

IBM®
Guardium Data Encryption 3.0.0.1

Teradata Protection User Guide

Release 3.0.0.1

IBM Guardium Data Encryption 3.0.0.1 is the same product as Vormetric Data Security (VDS) Release 6.0.3. VDS Release 6.0.3 consists of Data Security Manager Release 6.0.3 and Vormetric Encryption agent Release 6.0.3. The Key Vault and Key Agent features of Vormetric Data Security are not available with Guardium Data Encryption 3.0.0.1.

Vormetric Data Security Platform

Vormetric Protection for Teradata Database

Installation and Reference Guide

Version 6.0.2

Vormetric Data Security Platform
Vormetric Protection for Teradata Database
Version 6.0.2
June 22, 2018, *Installation and Reference Guide v1*
Copyright 2009 – 2018. Thales e-Security, Inc. All rights reserved.

NOTICES, LICENSES, AND USE RESTRICTIONS

Vormetric, Thales, and other Thales trademarks and logos are trademarks or registered trademark of Thales e-Security, Inc. in the United States and a trademark or registered trademark in other countries.

All other products described in this document are trademarks or registered trademarks of their respective holders in the United States and/or in other countries.

The software ("Software") and documentation contains confidential and proprietary information that is the property of Thales e-Security, Inc. The Software and documentation are furnished under license from Thales and may be used only in accordance with the terms of the license. No part of the Software and documentation may be reproduced, transmitted, translated, or reversed engineered, in any form or by any means, electronic, mechanical, manual, optical, or otherwise.

The license holder ("Licensee") shall comply with all applicable laws and regulations (including local laws of the country where the Software is being used) pertaining to the Software including, without limitation, restrictions on use of products containing encryption, import or export laws and regulations, and domestic and international laws and regulations pertaining to privacy and the protection of financial, medical, or personally identifiable information. Without limiting the generality of the foregoing, Licensee shall not export or re-export the Software, or allow access to the Software to any third party including, without limitation, any customer of Licensee, in violation of U.S. laws and regulations, including, without limitation, the Export Administration Act of 1979, as amended, and successor legislation, and the Export Administration Regulations issued by the Department of Commerce, or in violation of the export laws of any other country.

Any provision of any Software to the U.S. Government is with "Restricted Rights" as follows: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277.7013, and in subparagraphs (a) through (d) of the Commercial Computer-Restricted Rights clause at FAR 52.227-19, and in similar clauses in the NASA FAR Supplement, when applicable. The Software is a "commercial item" as that term is defined at 48 CFR 2.101, consisting of "commercial computer software" and "commercial computer software documentation", as such terms are used in 48 CFR 12.212 and is provided to the U.S. Government and all of its agencies only as a commercial end item. Consistent with 48 CFR

12.212 and DFARS 227.7202-1 through 227.7202-4, all U.S. Government end users acquire the Software with only those rights set forth herein. Any provision of Software to the U.S. Government is with Limited Rights. Thales is Thales eSecurity, Inc. at Suite 710, 900 South Pine Island Road, Plantation, FL 33324.

THALES PROVIDES THIS SOFTWARE AND DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT OF THIRD PARTY RIGHTS, AND ANY

Vormetric Data Security User Guide

WARRANTIES ARISING OUT OF CONDUCT OR INDUSTRY PRACTICE. ACCORDINGLY, THALES DISCLAIMS ANY LIABILITY, AND SHALL HAVE NO RESPONSIBILITY, ARISING OUT OF ANY FAILURE OF THE SOFTWARE TO OPERATE IN ANY ENVIRONMENT OR IN CONNECTION WITH ANY HARDWARE OR TECHNOLOGY, INCLUDING, WITHOUT LIMITATION, ANY FAILURE OF DATA TO BE PROPERLY PROCESSED OR TRANSFERRED TO, IN OR THROUGH LICENSEE'S COMPUTER ENVIRONMENT OR ANY FAILURE OF ANY TRANSMISSION HARDWARE, TECHNOLOGY, OR SYSTEM USED BY LICENSEE OR ANY LICENSEE CUSTOMER. THALES SHALL HAVE NO LIABILITY FOR, AND LICENSEE SHALL DEFEND, INDEMNIFY, AND HOLD THALES HARMLESS FROM AND AGAINST, ANY SHORTFALL IN PERFORMANCE OF THE SOFTWARE, OTHER HARDWARE OR TECHNOLOGY, OR FOR ANY INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS, AS A RESULT OF THE USE OF THE SOFTWARE IN ANY ENVIRONMENT. LICENSEE SHALL DEFEND, INDEMNIFY, AND HOLD THALES HARMLESS FROM AND AGAINST ANY COSTS, CLAIMS, OR LIABILITIES ARISING OUT OF ANY AGREEMENT BETWEEN LICENSEE AND ANY THIRD PARTY. NO PROVISION OF ANY AGREEMENT BETWEEN LICENSEE AND ANY THIRD PARTY SHALL BE BINDING ON THALES.

Protected by U.S. patents:

6,678,828

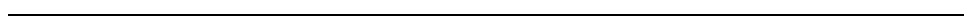
6,931,530

7,143,288

7,283,538

7,334,124

Thales Data Security includes a restricted license to the embedded IBM DB2 database. That license stipulates that the database may only be used in conjunction with the Thales Vormetric Security Server. The license for the embedded DB2 database may not be transferred and does not authorize the use of IBM or 3rd party tools to access the database directly.



Contents

Preface	5
Documentation Version History	5
Assumptions	5
Guide to Vormetric Protection for Teradata Database Documentation	6
Vormetric Data Security Platform—Overview	6
Service Updates and Support Information	7
Sales and Support	8
1 Vormetric Protection for Teradata Database (VPTD)	1
Overview	1
VPTD Operation	2
Normal Mode	2
Fast Mode	4
Whitelisting and Blacklisting VPTD Users	4
Creating Profiles for Invoking UDFs	5
Components	5
Data Security Manager (GDE Appliance)	5
Key Agent	5
Vormetric Local Cryptoserver Daemon	5
PKCS#11 Application Encryption Library	6
Key Cache Options	6
2 Installing and Deploying VPTD	7
Prerequisites for Installing VPTD on a Teradata Node	7
Installation Overview	8
Vormetric Protection for Teradata Installation Checklist	8
How to Get the FQDN of the GDE Appliance	9

Teradata Node Name Resolution	9
Using the Admin CLI	10
Example	10
Obtaining a Data Encryption Key for Your VPTD Deployment	11
Creating a Data Encryption Key	11
Using an Encryption Key Not Created with GDE Appliance 2.0 release 2e or later	12
Add the Teradata Node to the GDE Appliance Database	12
Install Vormetric Protection for Teradata on the Teradata Node	13
Install Vormetric Protection for Teradata	14
Verify the Installation	16
Modify the Key Cache	16
Install VPTD in Silent Mode	17
Configure Vormetric Protection for Teradata	18
Deploy Vormetric Protection for Teradata Database on a Teradata Node	22
Automated install over a Teradata cluster	25
Upgrade VPTD	25
3 Vormetric Protection for Teradata Database UDFs	27
Invoking the UDFs	27
encrypt_cbc()	28
decrypt_cbc()	29
encrypt_fpe()	29
decrypt_fpe()	30
encrypt_string()	31
decrypt_data()	32
encrypt_char()	32
Description	32
Example:	33
decrypt_char()	33
Description	33
Example:	34
profiles.conf	35
Using profiles.conf with Unicode Block Cipher UDFs	35
Using profiles.conf with FPE	36

Dynamic Masking for Tokenization	37
Define Masks	37
Support partial tokenization capability	38
Add prefix and suffix to a token	38
Generating of one-time use token	39
FPE-Luhn token type	39
4 VPTD Ongoing Operations	41
Log Messages	41
Troubleshooting	41
Be Sure Teradata User Can Access /tmp/vormetric	41
Cache Key on Host When Turning Off udf_aes	42
Set Width of BTEQ Session to Avoid Truncated UDF Output	42
Flush the Cache to Remove Old Profiles	42
Properly Escape Characters in Input Strings	43
A Locking Down Internet-facing Servers Supporting VPTD	45
Security Tips for Vormetric Protection for Teradata Database	45
To Disallow SSH Password Login and Use a Key Pair Login	47
B Uninstalling VPTD	49
Uninstalling Vormetric Protection for Teradata Database	49
Automated uninstall over a Teradata cluster	50

Preface

This manual describes how to install and implement Vormetric Protection for Teradata Database (VPTD) on your Teradata nodes.

Documentation Version History

The following table describes the changes made for each document version.

Documentation Changes

Document Version	Date	Changes
6.0.2. v1	5/10/18	Updated illustrations. Added new features: Dynamic masking, install, uninstall and upgrade over a TD cluster.
5.2.5 v2	1/25/17	Add integrated installer (vptd-*).
5.2.5 v1	12/16/16	GA release of 5.2.5 document.
5.2.5 Beta v1	9/2/16	Added UDFs for Unicode block cipher and FPE encryption: <code>encrypt_cbc()</code> , <code>decrypt_cbc()</code> , <code>encrypt_fpe()</code> , and <code>decrypt_fpe()</code> .

Assumptions

This documentation assumes knowledge of the Teradata database.

The system administrator must have root permissions for the systems on which Vormetric Protection for Teradata Database software is installed.

Guide to Vormetric Protection for Teradata Database Documentation

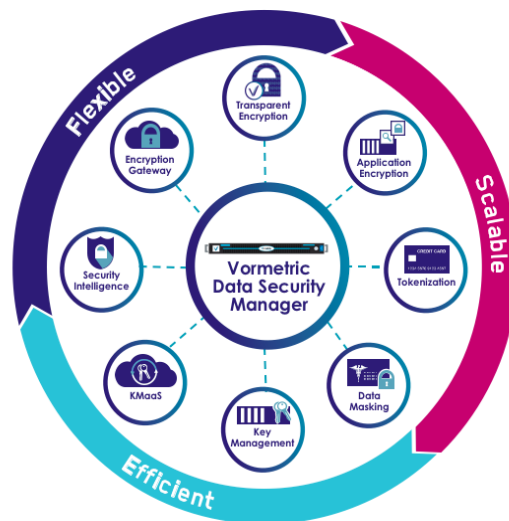
The following related documents are available to registered users on the Vormetric Web site at <https://support.vormetric.com>

1. **Data Security Manager (GDE Appliance) Installation and Configuration Guide.** Use this to install the Data Security Manager.
2. **Vormetric Protection for Teradata Database Installation and Reference.** This document. For security professionals who want to protect their Teradata Database.
3. **Vormetric Security Intelligence Configuration Guide.** Use this to integrate your Vormetric Tokenization event logs with the ArcSight ESM, Splunk, or IBM QRadar
4. **Vormetric Data Security (VDS) Platform Event and Log Messages Reference** (~700 pages, 0% pictures). A listing of all the VDS Platform event and log messages with severity, long and short form, and description.

Vormetric Data Security Platform—Overview

The Vormetric Data Security (VDS) Platform protects data wherever it resides. The platform solves security and compliance issues with encryption, key management, privileged user access control, and security intelligence logging. It protects data in databases, files, and Big Data nodes across public, private, hybrid clouds and traditional infrastructures.

Figure 1: The Vormetric “Solar System”



The platform consists of products that share a common, extensible infrastructure. At the heart of the platform is the Data Security Manager (GDE Appliance), which coordinates policies, keys,

and the collection of security intelligence, all of which is managed and observed through your browser. In addition to the GDE Appliance, the Vormetric Data Security Platform consists of the following products:

- **Vormetric Application Encryption (VAE)** provides a framework to deliver application-layer encryption such as column- or field-level encryption in databases, Big Data, or PaaS applications. VAE is an application encryption library providing a standards-based API to do cryptographic and encryption key management operations into existing corporate applications.
- **Vormetric Cloud Encryption Gateway (VCEG)** safeguards files in cloud-storage environments, including Amazon S3 and Box. VCEG encrypts sensitive data before it is saved to the cloud, enabling security teams to establish visibility and control around cloud assets.
- **Vormetric Key Management (VKM)** centralizes 3rd-party encryption keys and stores certificates securely. It provides standards-based enterprise encryption key management for Transparent Database Encryption (TDE), KMIP-compliant devices, and offers vaulting and inventory of certificates.
- **Vormetric Protection for Teradata Database** provides granular controls to secure assets in Teradata environments. It simplifies the process of using column-level encryption in your Teradata database. It provides documented, standards-based APIs and user-defined functions (UDFs) for cryptographic and key management operations.
- **Vormetric Security Intelligence** provides comprehensive logging combined with Security Information Event Management (SIEM) systems to detect advanced persistent threats and insider threats. In addition, the logs satisfy compliance and regulatory audits.
- **Vormetric Tokenization with Dynamic Data Masking (VTS)** replaces sensitive data in your database with unique identification symbols called tokens. This reduces the number of places that clear-text sensitive data reside, and thus reduces the scope of complying with PCI DSS and corporate security policies.
- **Vormetric Transparent Encryption (VTE)** secures any database, file, or volume across your enterprise without changing the applications, infrastructure, or user experience.

Service Updates and Support Information

The license agreement that you have entered into to acquire the Thales products ("License Agreement") defines software updates and upgrades, support and services, and governs the terms under which they are provided. Any statements made in this guide or collateral documents that conflict with the definitions or terms in the License Agreement, shall be superseded by the definitions and terms of the License Agreement. Any references made to "upgrades" in this guide or collateral documentation can apply either to a software update or upgrade.

Sales and Support

For support and troubleshooting issues:

- <http://help.thalesecurity.com>
- <https://www.thalesecurity.com/support/contact-support>
- support@thalesecurity.com
- (877) 267-3247

For Thales Sales:

- <http://enterprise-encryption.vormetric.com/contact-sales.html>
- sales@thalesec.net
- (408) 433-6000

Vormetric Protection for Teradata Database (VPTD)

1

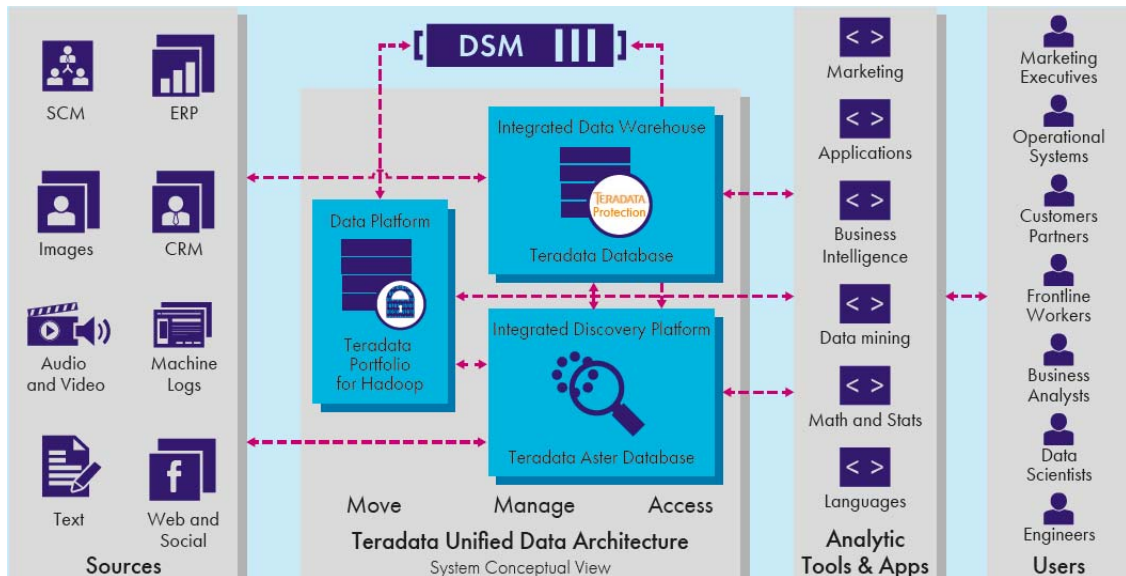
Vormetric Protection for Teradata Database (VPTD) is essentially Vormetric Application Encryption (VAE) in the Teradata environment. It enables column-level encryption of Teradata databases. This chapter describes how VPTD provides protection in the Teradata database environment. It contains the following sections:

- “Overview” on page 1
- “PKCS#11 Application Encryption Library” on page 6

Overview

A Teradata database consists of a Parsing Engine (PE) and any number of Access Module Processors (AMP) that exist on *nodes*. Nodes are servers that can host multiple virtual AMPs (VAMPs) and, optionally, the parsing engine itself.

Figure 1: Typical Teradata deployment



A Teradata site consists of one or more nodes. Each Teradata site requires a GDE Appliance. The following Teradata **User Defined Functions** (UDFs) must be provided by Vormetric and installed in the Teradata database:

- `encrypt_cbc()` – Encrypts a string provided in Unicode format
- `decrypt_cbc()` – Decrypts data that was encrypted using `encrypt_cbc()` and returns a string in Unicode format
- `encrypt_fpe()` – Performs format preserving encryption on a string provided in Unicode format
- `decrypt_fpe()` – Decrypts data that was encrypted using `encrypt_fpe()`
- `encrypt_string()` – Provided for backward compatibility. Encrypts a string provided in Latin characters
- `decrypt_data()` – Provided for backward compatibility. Decrypts data that was encrypted using `encrypt_string()` and returns a string in Latin characters
- `encrypt_char()` – Encrypts Latin character sets. Note that this is not compatible with the ciphertext output of `encrypt_string()` for the given plain text and key combination.
- `decrypt_char()` – Decrypts Latin character sets.

Each Teradata node requires a:

- Vormetric Key Agent
- Vormetric Application Encryption (VAE) Teradata installation file, which contains a Cryptoserver
- Connection to the GDE Appliance

VPTD Operation

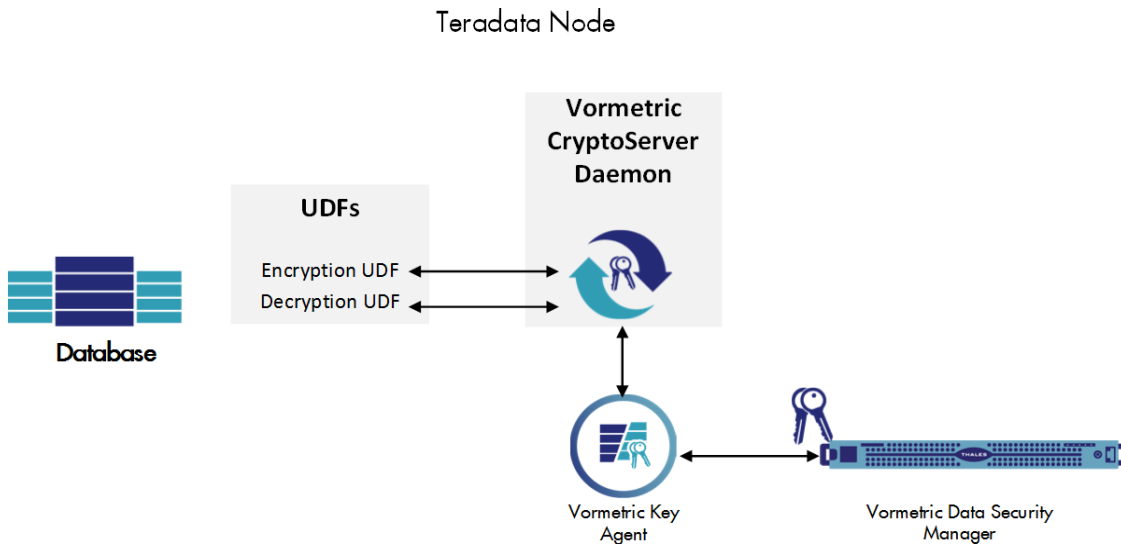
Once these components are installed, you have access to the UDFs that can be used to encrypt and decrypt Teradata database columns.¹ These UDFs can run in two modes: **Normal Mode** and **Fast Mode**.

Normal Mode

The following figure shows VPTD running in the Normal Mode.

1. You also have access to the VAE API library. See the *VAE Installation Guide and API Reference* for details.

Figure 2: Teradata node with VPTD running in the Normal Mode



In the Normal mode, `encrypt_cbc()`, `encrypt_fpe()`, or `encrypt_string()` sends a cleartext string to the Vormetric Local Cryptoserver Daemon. The Key Agent obtains a key from the GDE Appliance and encrypts the cleartext string. The encrypted string is then returned.

Likewise, `decrypt_cbc()`, `decrypt_fpe()`, or `decrypt_data()` sends an encrypted text string to the Vormetric Local Cryptoserver Daemon. The Key Agent obtains the key from the GDE Appliance and decrypts the string. The cleartext string is then returned.

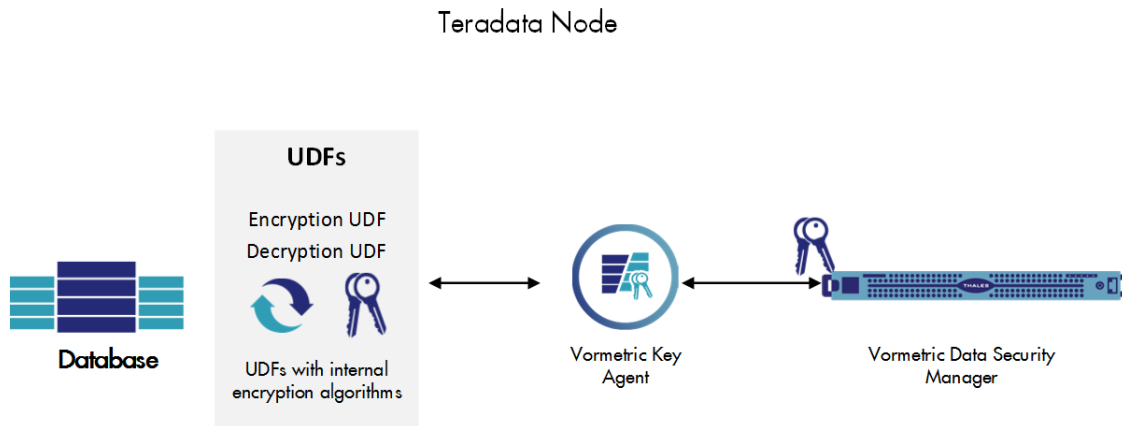
For more information about performance details and how to enable Normal and Fast Mode, see .

See [“Vormetric Protection for Teradata Database UDFs” on page 27](#) for examples of how to use these UDFs.

Fast Mode

The following figure shows VPTD running in the Fast Mode.

Figure 3: Teradata node with VPTD running in the Fast Mode



In the Fast Mode, the encryption UDF `encrypt_cbc()`, `encrypt_fpe()`, or `encrypt_string()` encrypts a cleartext string locally inside the UDF. The encrypted string is then returned.

Likewise, the decryption UDF `decrypt_cbc()`, `decrypt_fpe()`, or `decrypt_data()` decrypts an encrypted text string locally inside the UDF. The cleartext string is then returned.

For more information about performance details and how to enable Normal and Fast Mode, see [“Configure Vormetric Protection for Teradata”](#) on page 18.

See [“Vormetric Protection for Teradata Database UDFs”](#) on page 27 for examples of how to use these UDFs.

Whitelisting and Blacklisting VPTD Users

To control which users can execute the UDFs, create black lists and white lists for the UDFs. Add users to the `deny_decrypt.conf` black list and they will be barred from executing the decryption UDFs. You could also add users to the `allow_decrypt.conf` white list, and only those users can execute decryption UDFs. See [“Deploy Vormetric Protection for Teradata Database on a Teradata Node”](#) on page 22 for deployment details.

Whitelisting and blacklisting VPTD users is particularly useful for preventing internal threats to your Teradata database. By whitelisting approved users of the middleware (the business logic), all others, including the Teradata DB administrators, are blocked from accessing sensitive data. See [Figure 7: “High Security Deployment for Teradata Deployment”](#) on page 45.



Creating Profiles for Invoking UDFs

To streamline the invocation of the `encrypt_cbc()`, `decrypt_cbc()`, `encrypt_fpe()` and `decrypt_fpe()` UDFs, VPTD uses the file `profiles.conf`. This file contains named profiles that include several values required as input to these UDFs. Provide the profile name as a parameter when invoking the UDF. The UDF looks up the profile name and uses the parameters that are grouped under that profile name.



NOTE: Profiles are not supported for the `encrypt_string()` and `decrypt_data()` UDFs.

For more information, see [“encrypt_char\(\)” on page 32](#).

Components

This section describes the components of the VDS product that are relevant to Application Encryption.

Data Security Manager (GDE Appliance)

The GDE Appliance is the central component of the VDS Platform. It consists of a policy engine and a central key and policy manager. The GDE Appliance stores and manages host encryption keys, data access policies, administrative domains, and administrator profiles.

Key Agent

The Vormetric Key Agent provides a library that implements the PKCS#11 interface. This library is a dynamically loadable library (dll) on Windows and a shared object (so) on Linux and UNIX. The Key Agent's PKCS#11 library communicates over a secure channel to the GDE Appliance for all significant functionality. This is sometimes called the Vormetric Application Encryption Agent.

Vormetric Protection for Teradata Database supports single and multi-part encryption and decryption using AES 128, and AES 256. It does not store or manage other kinds of cryptographic objects as described in the PKCS #15 specification such as certificates, passwords, or any custom made PKCS#11 objects.

Vormetric Local Cryptoserver Daemon

In the Normal mode, Vormetric Local Cryptoserver Daemon accepts requests from the UDFs, forwards them to the Vormetric Key Agent for processing, and returns the Key Agent output to the respective UDF.

PKCS#11 Application Encryption Library

In addition to the Teradata UDFs, VPTD also provides APIs in the Vormetric library that are a subset of the PKCS#11 specification version 2.20. They are platform-independent cryptographic tokens, and are traditionally used for HSMs (hardware security modules) and smartcards.

These APIs support real-time I/O encryption and decryption of “at rest” data and log files. Data is encrypted by adding these APIs to existing applications.

VAE APIs support the following functionality:

- create a key
- find a key
- destroy a key
- export a key
- import a key
- encrypt
- decrypt
- sign
- wrap key

Key Cache Options

You can choose to store keys on the GDE Appliance or export them to a local key cache. By default, keys are stored in the local key cache. For greater security, you can keep the keys on the GDE Appliance; however, that will affect performance. Table 1 on page 6, compares the advantages of each option.

Table 1: Key storage options

	Key stored on the GDE Appliance	Key cached on host
Performance	- Network round trip slows performance.	- Very fast for bulk encryption or high volume transactions of data.
Security	- Most secure. The key never leaves the GDE Appliance. - In compliance with PCI security standards.	- Key is stored on the client in memory. - In compliance with PCI security standards.

Installing and Deploying VPTD

This chapter describes how to install and configure the Vormetric Protection for Teradata Database on your Teradata host. It contains the following sections:

- “Prerequisites for Installing VPTD on a Teradata Node” on page 7
- “Installation Overview” on page 8
- “Vormetric Protection for Teradata Installation Checklist” on page 8
- “Add the Teradata Node to the GDE Appliance Database” on page 12
- “Install Vormetric Protection for Teradata on the Teradata Node” on page 13
- “Install VPTD in Silent Mode” on page 17
- “Configure Vormetric Protection for Teradata” on page 18
- “Deploy Vormetric Protection for Teradata Database on a Teradata Node” on page 22
- “Automated install over a Teradata cluster” on page 25
- “Upgrade VPTD” on page 25

Prerequisites for Installing VPTD on a Teradata Node

The following is required to install Vormetric Protection for Teradata Database:

- A hardware GDE Appliance 3.0 or later, (to find your GDE Appliance version, open the Management Console and choose **System > About**). For GDE Appliance installation instructions, see *GDE Appliance Installation and Configuration Guide*.
- Access to the GDE Appliance from all Teradata nodes.
- The Teradata nodes must run Teradata 16.10, 16.00, 15.00, 15.10, 14.00, or 14.10 on SUSE Linux Enterprise Server (SLES) 10 or SLES 11 nodes. For each node you need:
 - Root password to Teradata nodes
 - The IP address or FQDN of the node
 - Installation executable file for Vormetric Protection for Teradata version 6.0.2 for Teradata. For example:

```
vptd16.10-6.0.2-94-sles11sp1-x86_64.bin
```

This installs the following:

- Vormetric Protection for Teradata, including the Vormetric Key Agent software
- UDFs
- Vormetric Local Cryptoserver Daemon
- Sample installation script (*install_udfs.bteq.sample*)
- Sample UDF profile configuration file (*profiles.conf.sample*)
- Sample Cryptoserver configuration file (*vormetric_local_crypto_server.conf.sample*)
- Mask configuration file (*mask.conf.sample*)

Installation Overview

The following are the high-level steps for installing Vormetric Protection for Teradata Database on a Teradata node.

1. Collect configuration information in the [“Vormetric Protection for Teradata Installation Checklist”](#) on page 8.
2. [“Add the Teradata Node to the GDE Appliance Database”](#) on page 12.
3. [“Install Vormetric Protection for Teradata on the Teradata Node”](#) on page 13.
4. [“Configure Vormetric Protection for Teradata”](#) on page 18.
5. [“Deploy Vormetric Protection for Teradata Database on a Teradata Node”](#) on page 22.

Vormetric Protection for Teradata Installation Checklist

Use this table to verify prerequisites and collect the information you need for the installation.

Table 2: VPTD Installation Checklist

Checklist item	Status
Hardware GDE Appliance 3.0 or later	
Obtain Vormetric Protection for Teradata installation file from Vormetric. Example: (vptd16.10-6.0.2-94-sles11sp1-x86_64.bin)	
Host system clock set to the correct time zone.	

Table 2: VPTD Installation Checklist

Checklist item	Status
Fully Qualified Domain Name (FQDN) of the GDE Appliance (page 9)	
Root password for the Teradata node	
IP address or Fully Qualified Domain Name (FQDN) of the Teradata node. (page 9)	
Symmetric encryption key, cached on the host. (page 11)	

How to Get the FQDN of the GDE Appliance

Log in to the Management Console and look at the dashboard.

1. Open a browser and enter the GDE Appliance URL.

This is either the hostname if configured in DNS, or its IP over HTTPS of the GDE Appliance.

2. Enter the default login and password. The default login is `admin`. The default password is `admin123`.



NOTE: You are asked to change the default password upon first login. Remember this new password or you will not be able to log in again!

3. The FQDN is at the top.



Teradata Node Name Resolution

You can map a Teradata node name to an IP address using a Domain Name Server (DNS). DNS is the most preferred method of host name resolution.

You can also modify the `hosts` file on the GDE Appliance or identify a host using only the IP address.

- If you use DNS to resolve host names, use the FQDN for the host names.
- If you do NOT use a DNS server to resolve host names, do the following on all of the GDE Appliances and the protected Teradata nodes:
 - **Modify the `host` file on the GDE Appliance:** To use names like `serverx.domain.com`, enter the node names and matching IP addresses in the `/etc/hosts` file on the GDE Appliance

using the `host` command under the `network` menu of the Admin CLI (see [See “Using the Admin CLI” on page 10](#)). For example:

```
0011:network$ host add grimes-dev1 192.168.42.12
SUCCESS: add host
0012:network$ host sh
name=localhost6.localdomain6 ip>:::1
name=linux32-48215.sacdbackup.com ip=192.168.48.215
name=rgrimes-dev1 ip=192.168.42.12
SUCCESS: show host
```

You must do this on *each* GDE Appliance, since entries in the host file are not replicated across GDE Appliances.

- **Modify the `hosts` file on the protected hosts:** Enter the GDE Appliance host names and matching IP addresses in the `/etc/hosts` file on the Teradata node. *You must do this on EACH protected node making sure to add an entry for all GDE Appliance nodes (if using HA).*

OR

- **Use IP addresses:** If using IP addresses as protected Teradata node names, you must enable Agent IP in the GDE Appliance (see “agentip” Enable/Disable in the “network” menu in the CLI. This is described in [“Using the Admin CLI” on page 10](#)). With Agent IP enabled, you can have some hosts identified by host name and some by IP simultaneously. In other words, they don't all need to use an IP address when Agent IP is enabled.

Using the Admin CLI

Access the CLI menu as follows:

1. Start the serial console application.
2. If the login prompt is not displayed, press the **Enter** key to wake up the connection.
3. Log into the appliance. The default System Administrator name and password are `cliadmin` and `cliadmin123`.

Example

At the prompt, type `cliadmin` followed by the password. At the `Vormetric$` prompt type “`network`”. See the example below:

```
0001:network$ agentip show
agent ip address support : off
SUCCESS: agent ip address support showed.
0002:network$ agentip on
WARNING: The Security Server will restart automatically after enabling
agent IP address support!
Continue? (yes/no)[no]:yes
```



```
SUCCESS: Agent IP address support is enabled and the server restarted.  
0003:network$ agentip show  
agent ip address support : on  
SUCCESS: Agent IP address support showed.  
0004:network
```

Obtaining a Data Encryption Key for Your VPTD Deployment

This section contains two subsections:

- [“Creating a Data Encryption Key” on page 11](#)
- [“Using an Encryption Key Not Created with GDE Appliance 2.0 release 2e or later” on page 12](#)

Creating a Data Encryption Key

1. Go to **Keys > Agent Keys > Keys** in the Management Console to open the *Agent Keys* window.
2. Click **Add** to open the **Add Agent Key** window.
3. Enter a key name, description, and security algorithm.
 - **Name:** Name of key. 64 character limit.
 - **Description:** Optional key description. 265 character limit.
 - **Template:** A key template with a set of pre-defined attributes. To create a valid Teradata key, select **Default_SQL_Symmetric_Key_Template** and do not change any of the custom attribute values.
 - **Algorithm:** Algorithm used to create the key.
 - **Key Type:** Location for the encryption key. **Stored on Server** keys are downloaded to non-persistent memory on the host. Each time the key is needed, the host must retrieve the key from the GDE Appliance. **Cached on Host** downloads and stores (in an encrypted form) the key in persistent memory on the host. For performance reasons, **Cached on Host** is highly recommended for Teradata installations. For Fast Mode (configuration file setting `'udfaes on'`, see [“Deploy Vormetric Protection for Teradata Database on a Teradata Node” on page 22](#)), key type **Cached on Host** is MANDATORY
 - **Unique to Host:** This checkbox is displayed when **Cached on Host** is selected. When enabled, it makes the encryption key unique. The key is downloaded to the host, encrypted using the host password, and stored. These keys are used for locally attached devices, as files encrypted by them can be read only by one machine. Do not enable this checkbox for cloned systems, RAID configurations, clustered environments, or any environment that uses host mirroring. Requires that **Key Creation Method** is set to *Generate*.
 - **Key Creation Method:** Select to generate a key using a random seed (**Generate**) or by **Manual Input**.

- **Expiry Date:** Date the key expires.
- **Key Refreshing Period (minutes):** Used only with Oracle Database TDE and Microsoft SQL Server TDE. How long to keep the key in the local key cache before it is refreshed.

Example:

Name: Key1
Description: Teradata key
Algorithm: AES256

All other values are the default.

4. Click **OK**. Your new key is created and displayed in the *Agent Keys* window.

Selected	UUID	Name	Algorithm	Key Type	Creation Date	Expiry Date	Source	Description
<input type="checkbox"/>	02-51	Key1	AES256	Cached on Host	Jan 03, 2015		ip-10-0-5-135.ec2.internal	Teradata key

Add Delete Page 1 of 1

5. Create as many keys as desired.

Using an Encryption Key Not Created with GDE Appliance 2.0 release 2e or later

If you try to use a key created with GDE Appliance 2.0 release 2d or another environment in the VPTD environment, it will not work. You must first modify the key's attributes in the GDE Appliance Management Console GUI (**Keys > [Select a key] > Attributes**):

Cryptographic Usage Mask	127
Object Type	Symmetric Key
x-VormApplicationName	Vormetric Key Agent
x-VormCanBePlainText	true
x-VormCanNeverBeExported	true
x-VormCanNeverBePlaintext	true
x-VormCanObjectPersist	true

Add the Teradata Node to the GDE Appliance Database

Your Teradata node names must be added to the GDE Appliance database before Vormetric Protection for Teradata is installed. This section describes how to do this. To add the Teradata node to the GDE Appliance database, you must have the Teradata node's Fully Qualified Domain Name (FQDN—54 character max) or IP address.

1. Log on to the Management Console as a Security Administrator with *Key* and *Policy* roles or as an administrator of type *All*.

2. Switch to the domain containing the Teradata node you want to protect. Click **Domains > Switch Domains**. The *Switch Domains* window opens.
3. Select the domain that will contain the Teradata host and click **Switch to domain**. The domain in which you are working is displayed in the upper right corner of the Management Console.



4. Select **Hosts > Hosts** in the menu bar. An empty *Hosts* window opens.
6. Click **Add**. The *Add Host* window opens.
7. Enter the following information:
 - **Host Name:** Enter FQDN of the Teradata Node. The IP address or host name will work, but the FQDN changes the least. Host names cannot contain an underscore.
 - **Description:** Optional. Enter text to identify the node or its function. Limited to 256 characters.
 - **Registration Allowed Agents:** Select the agents that will run on the node system. In this case it will be **Key**.
 - **License Type:** Choose the type of license that will run on this host. Options are **Perpetual**, **Term**, and **Hourly**, depending on the system license.
8. Click **Ok**. You are returned to the *Hosts* window.
9. Click the hostname link that you just added to the GDE Appliance database. This opens the **General** tab of the *Edit Host* window. Make sure the **Registration Allowed** and **Communication Enabled** checkbox are checked for the Key Agent.
10. Your node is added to the GDE Appliance database.
11. Repeat for all your protected Teradata nodes.

Install Vormetric Protection for Teradata on the Teradata Node

The following steps describe how to install Vormetric Protection for Teradata for the first time.

Install Vormetric Protection for Teradata

1. Log on with root access to the host where you will install Vormetric Protection for Teradata.
2. Copy or mount the installation file to the host system.

3. Start the installation. Type:

```
# ./vptd-<product-version-build-system>.bin
```

Example: *vptd16.10-6.0.2-94-sles11sp1-x86_64.bin*

4. The text of the License Agreement appears. Page through the agreement, then type *y* and press **Enter** to accept. The installation proceeds.
5. At the agent registration prompt, type *y* or press **Enter** to proceed with registration, or type *n* if you plan to register later.

```
Do you want to continue with agent registration? (Y/N) [Y]:
```

6. Continue to follow the prompts:

a: Enter the fully qualified host name of the primary GDE Appliance, and then press **Enter**.

b: Verify the host name, and then press **Enter**.

c: The installer shows a numbered list of host names and prompts you to choose one. From the list, type the number of the host name of your local machine, and then press **Enter**. This name must match the name of the host that was added to the GDE Appliance by the Security Administrator.

Example:

```
Please enter the host name of this machine, or select from the
following list. The name you provide must precisely match the
name used on the "Add Host" page of the Management Console.
```

```
[1] host1.example.com
[2] sys41017-priv.example.com
[3] sys41017-vip.example.com
[4] 10.3.41.127
```

```
Enter a number, or type a different host name or IP address in
manually:
```

```
What is the name of this machine? [1]: 1
```

```
Generating certificate...done.
```

```
Signing certificate...done.
```

7. At the following prompt, choose whether you want to register to the Security Server using a shared secret or using fingerprints.

Would you like to register to the shared Security Server using a registration shared secret (S) or using fingerprints (F)? (S/F) [S]:

The rest of this procedure is based on fingerprint registration. The advantages and disadvantages of each security method are beyond the scope of this document. Refer to the GDE Appliance documentation for complete details.

8. At the prompt, press **Enter** to enable cloning prevention ability, by associating the installation with existing hardware.

Example:

```
It is possible to associate this installation with the hardware
of this machine. If selected, the agent will not contact the
GDE Appliance or use any cryptographic keys if any of this
machine's hardware is changed. This means that if the machine
is copied (for example, a clone of a virtual machine), the copy
will not function correctly. This can be rectified by running
this registration program again.
```

```
Do you want to enable this functionality? (Y/N) [Y]:
```

9. The CA certificate fingerprint is displayed:

The following is the fingerprint of the CA certificate. Please verify that it matches the fingerprint shown on the Dashboard page of the Management Console. If they do not match, it can indicate an unsuccessful setup or an attack.

```
4C:10:12:72:1B:56:7D:93:A4:71:0D:93:B9:47:CF:A0:8F:D4:5A:E9
```

```
Do the fingerprints match? (Y/N) [N]: Y
```

At this stage of the installation, you, the host administrator, and GDE Appliance Security Administrator must exchange information to confirm that the agent host and GDE Appliance share valid certificates. This step verifies that nobody is intercepting and modifying traffic between the GDE Appliance and agent. It is a security feature.

- a. **Host Admin:** Send the fingerprint to the GDE Appliance Security Administrator and wait for confirmation.
- b. **GDE Appliance Security Admin:**
 - Log on to the GDE Appliance Management Console and navigate to the domain where the Teradata node was added.
 - Click the **Dashboard** tab.
 - Match the fingerprint from the Host Admin with the **EC CA fingerprint** on the Dashboard.
 - Advise the Host Admin of the results.
- c. **Host Admin:** If the fingerprints match, type *y* and then press **Enter**. "Installation success." is displayed.



NOTE: If this is a Key Agent installation, you are prompted for a password for the Key Agent library.

10. Enter the password for the Key Agent library and confirm your entry. Password minimum length for PIN is 8 characters and maximum length is 63 characters.

Please enter a password for the Key Agent library.

Accesses to this library will be protected by this password.

NOTE: If using Oracle RAC, passwords must be the same on all nodes.

Please enter password :

Enter again to confirm :

Successfully registered the Vormetric Key Agent with the primary Vormetric Data Security Server on test-vormetric.com.



NOTE: Do not forget this password. You will need it later during Vormetric Protection for Teradata Database deployment. To change the password, you must re-register the agent.

Verify the Installation

The following steps verify installation.

1. Run the following commands on the Teradata node to confirm that all components of VPTD were entered into the rpm database, which means that VPTD was installed successfully.

```
rpm -qa | grep vae-td
```

```
rpm -qa | grep vee-key
```

2. Open the GDE Appliance Edit Host window and verify that **Registration Allowed** and **Communication Enabled** are selected for the Key Agent.

Modify the Key Cache

You can modify two settings in the key cache at application level or on the GDE Appliance.

By default, keys are cached on the host, with a time-to-live value of 10080 minutes (7 days).

- You can choose to cache the key on the GDE Appliance or on the host.
- You can also modify how long the key stays in the local key cache before it is re-fetched from the GDE Appliance.

By accepting the default setting “Cached on Host”, the cryptographic key will be cached within the Key Agent library on the particular host (Teradata node).

The setting “Stored on Server” means that the key never leaves the GDE Appliance and thus all cryptographic operations with this key are performed on the GDE Appliance itself, not on the host.

These two settings are a trade off between speed and security. In most cases, end users will not be willing to accept the performance penalty of the “Stored on Server” setting, which is extreme.

To Modify the Key Cache on the GDE Appliance:

1. Log on to the GDE Appliance as an administrator of type Security Administrator.
2. Click the **Domains** tab, and switch to the domain where the key is installed.
3. Click **Keys > Agent Keys**, and then click on the name of the key you want to modify. If you have many keys, search for a key on the Management Console using the **Keyname contains:** field. The *Edit Agent Key* window opens.
4. In the **Key Type** drop-down menu, select **Cached on Host** or **Stored on Server** (GDE Appliance).
5. In the **Key Refreshing Period (minutes)** field, enter the number of minutes you want the key in the local key cache before it is refreshed from the GDE Appliance.
6. Click **Ok**.

Install VPTD in Silent Mode

A Silent Mode installation means that all requisite input is provided in a file and VPTD is invoked using the `-s` option. To install VPTD in Silent Mode:

1. Create a config file containing following fields:

```
SERVER_HOSTNAME=<hostname>
AGENT_USEIP=Yes
PKCS11_PASSWORD=<password>
```

2. Obtain VPTD and copy it to the directory that contains the config file that you just created.
3. Type:

```
# ./vptd<product-version-build-system>.bin -s <config file path>
```

Example

```
# ./vptd16.10-6.0.2-53-sles11sp1-x86_64.bin -s <config file path>
```

The following fields are optional for the Config file:

Table 3: Options for the Silent Mode installation configuration file

Name	Function	Required
SERVER_HOSTNAME	Host name of GDE Appliance	Yes
SERVER_IP	Synonym for SERVER_HOSTNAME	No
AGENT_USEIP	Uses IP address instead of hostname	No
AGENT_HOST_NAME	Uses the name of the machine instead of IP	No
AGENT_HOST_PORT	Port number to expose	No
STRONG_ENTROPY	Use /dev/random on linux	No
PKCS11_STANDALONE	Non-server mode for PKCS11 agent	No
PKCS11_PASSWORD	Password for PKCS11 agent	Yes (pkcs11 only)
ONEWAY_COMMS	Set when one-way communication required	No
USEHWSIG	Associate hardware to keys+certs (1 0)	No
SHARED_SECRET	Use given registration shared secret	No
HOST_DOMAIN	Registration domain	Yes (If SS provided)
HOST_GROUP	Registration host group	No
HOST_DESC	Host description	No

Configure Vormetric Protection for Teradata

After installing VPTD as described in [“Install Vormetric Protection for Teradata on the Teradata Node” on page 13](#), do the following steps to configure VPTD and get the system up and running.



NOTE: This procedure must be done only once regardless of the number of nodes.

1. Choose between Normal Mode and Fast Mode.

Before configuring Vormetric Protection for Teradata, you must first determine whether you will run in the Normal Mode or the Fast Mode. This decision will affect the configuration settings.

The table below outlines the various attributes of each mode and how they are enabled.

Table 4: Comparison of Normal Mode and Fast Mode

	Normal Mode (<i>udfaes off</i>)	Fast Mode (<i>udfaes on</i>)
Teradata Protected Mode (comment out the <i>alter function</i> lines in <i>install_udfs.bteq</i>)	<ul style="list-style-type: none"> Supported by 5.2.2 and later releases. Encryption/decryption done on cryptoserver. 	<ul style="list-style-type: none"> 5.2.3 and later releases. Encryption/decryption done locally inside the respective UDF.
Teradata Unprotected Mode (Default)	<ul style="list-style-type: none"> Not generally used. 	<ul style="list-style-type: none"> Recommended for maximum performance. Encryption/decryption done locally inside the respective UDF.

Specify the Teradata Protected or Unprotected Mode in the BTEQ file. Specify the *udfaes* parameter (Fast Mode or Normal Mode) in the Vormetric Local Cryptoserver Daemon configuration file (see [“Deploy Vormetric Protection for Teradata Database on a Teradata Node” on page 22](#)).

2. Create a Basic Teradata Query (BTEQ) script from the provided sample to install the UDFs.
 - a. Change directories:

```
# cd /opt/vormetric/DataSecurityExpert/agent/pkcs11/teradata/udfs/
```

- b. Copy the UDF sample script *install_udfs.bteq.sample* to a file named *install_udfs.bteq* in the same directory. Keep *install_udfs.bteq.sample* as a reference, and use *install_udfs.bteq* as your working BTEQ script.

- c. Edit `install_udfs.bteq` as per the embedded instructions. Replace the words in capital letters with real values. For example, change `USERNAME` and `PASSWORD` to a real user name and password.

```
bteq << $EOF
* Replace Teradata USERNAME and PASSWORD with a site-specific username and password.
.logon USERNAME,PASSWORD;
* Replace DBC with the database from which you want to derive the vormetric user.
* Note that the USER who installs the UDFs into the system must be set to the latin
* char set. After the UDFs are installed, the character set for this particular
* user MAY be changed to something else, for instance UNICODE. But during the UDF
* installation, the user's character set MUST be latin.
create user vormetric from DBC as perm=10000000 password=SOMEPASSWORD default character set
latin;
grant create function on vormetric to vormetric;
grant alter function on vormetric to vormetric;
grant drop function on vormetric to vormetric;
grant execute function on vormetric to public;
.logoff
.logon vormetric,SOMEPASSWORD;
```

```
database vormetric;

replace function encrypt_string (inputString varchar(16384), inputKeyname varchar(256))
returns varbyte(16400) specific encrypt_string language c no sql not deterministic parameter
style sql called on null input external name 'co:udf_encrypt_string:./udf_encrypt_string.o';

replace function decrypt_data (inputString varbyte(16400), inputKeyname varchar(256))
returns varchar(16384) specific decrypt_data language c no sql not deterministic parameter
style sql called on null input external name 'co:udf_decrypt_data:./udf_decrypt_data.o';

replace function encrypt_char (inputString varchar(16384), inputKeyname varchar(256),
inputCharcolumnsize INTEGER) returns varbyte(16000) specific encrypt_char language c no sql
not deterministic parameter style sql called on null input external name
'co:udf_encrypt_char:./udf_encrypt_char.o';

replace function decrypt_char (inputString varbyte(16000), inputKeyname varchar(256),
inputCharcolumnsize INTEGER) returns varchar(16384) specific decrypt_char language c no sql
not deterministic parameter style sql called on null input external name
'co:udf_decrypt_char:./udf_decrypt_char.o';

replace function encrypt_cbc (inputString varchar(8192) CHARACTER SET UNICODE, inputKeyname
varchar(256)) returns varbyte(16400) specific encrypt_cbc language c no sql not
deterministic parameter style sql called on null input external name
'co:udf_encrypt_cbc:./udf_encrypt_cbc.o';

replace function decrypt_cbc (inputString varbyte(16400), inputKeyname varchar(256)) returns
varchar(8192) CHARACTER SET UNICODE specific decrypt_cbc language c no sql not deterministic
parameter style sql called on null input external name
'co:udf_decrypt_cbc:./udf_decrypt_cbc.o';

replace function encrypt_fpe (inputString varchar(8192) CHARACTER SET UNICODE, inputKeyname
varchar(256)) returns varchar(8192) CHARACTER SET UNICODE specific encrypt_fpe language c no
sql not deterministic parameter style sql called on null input external name
'co:udf_encrypt_fpe:./udf_encrypt_fpe.o';

replace function decrypt_fpe (inputString varchar(8192) CHARACTER SET UNICODE, inputKeyname
varchar(256)) returns varchar(8192) CHARACTER SET UNICODE specific decrypt_fpe language c no
sql not deterministic parameter style sql called on null input external name
'co:udf_decrypt_fpe:./udf_decrypt_fpe.o';

* Comment out the following lines to run the UDFs in a separate process.
* By default, UDFs run in a separate process ("protected mode"), which incurs a performance
penalty of up to 20x or 25x or 30x
alter function encrypt_string execute not protected;
alter function decrypt_data execute not protected;
alter function encrypt_char execute not protected;
alter function decrypt_char execute not protected;
alter function encrypt_cbc execute not protected;
alter function decrypt_cbc execute not protected;
alter function encrypt_fpe execute not protected;
alter function decrypt_fpe execute not protected;

.logoff
.quit
```

- d. If you want to run in the Teradata Protected Mode, comment out the *alter function* lines. Leave them as is to run in the Teradata Unprotected Mode.

e. By default the BTEQ script installs the UDFs in the newly created database called *vormetric*. You may change the installation script to install them in the location of your choice.

f. Run the BTEQ script. Example:

```
# [/opt/vormetric/DataSecurityExpert/agent/pkcs11/teradata/udfs]#  
./install_udfs.bteq
```

3. After this script has successfully run, the UDFs are installed in the newly created database with the default name *vormetric*.

Deploy Vormetric Protection for Teradata Database on a Teradata Node

After installing Vormetric Protection for Teradata Database on the node, you must configure and deploy it.



NOTE: This procedure must be done for every node.

1. Log in as root user on the node.
2. Optional: Set up VPTD white lists or black lists.

Decide whether you are *blacklist-centric*—you have a list of users to block access to a command and all others are unblocked, or *whitelist-centric*—you have a list of users to allow access to a command and all others are blocked. Choose which orientation makes the most sense for your situation, then create the following four files in */etc/vormetric* with 600 permissions (read/write by root only):

```
allow_encrypt.conf  
allow_decrypt.conf  
deny_encrypt.conf  
deny_decrypt.conf
```

Add the logins of users, one line per user, to either deny (blacklist) or allow (whitelist) access to do encrypting or decrypting. User names must be entirely capitalized. The wildcard character (*) indicates “all users” and it is allowed in the whitelist, but not the blacklist. No other regular expressions are permitted. Make your configuration blacklist-centric by putting just a * in the whitelist and adding user names in the blacklist. Make it whitelist-centric by leaving your blacklist empty and putting user names in the whitelist.

If you are whitelist-centric, only users in `allow_decrypt.conf` will be able to decrypt data, and only users in `allow_encrypt.conf` will be able to encrypt data. If you are blacklist-centric, only users in `deny_decrypt.conf` will not be able to decrypt data, and only users in `deny_encrypt.conf` will be not able to encrypt data.

We recommend whitelisting approved users of the middleware (the business logic). This will block all others, including the Teradata DB administrators from accessing sensitive data.

The Linux system administrator must then replicate these files to all other Teradata nodes.

3. Create a new profile configuration file based on the provided sample.

```
# cd /etc/vormetric
# cp profiles.conf.sample profiles.conf
```

Keep `profiles.conf.sample` as a reference, and use `profiles.conf` as your working configuration file. Modify this file to serve your purposes. For more information, see [“encrypt_char\(\)” on page 32](#).

4. Create a Vormetric Local Cryptoserver Daemon configuration file.

```
# cd /etc/vormetric
# cp vormetric_local_crypto_server.conf.sample
  vormetric_local_crypto_server.conf
```

Keep `vormetric_local_crypto_server.conf.sample` as a reference, and use `vormetric_local_crypto_server.conf` as your working configuration file.

5. Modify the following lines in the Vormetric Local Cryptoserver Daemon configuration file `/etc/vormetric/vormetric_local_crypto_server.conf`:

```
Copyright (c) 2018 by <Your_Company>
iv <Initialization_Vector>
loglevel <desired_loglevel>
timeout <desired_whitelist-blacklist_polling_interval>
cryptoconf_timeout <in min>
udfaes <on_or_off>
```

where:

Your_Company: Name of your company.

IV: Initialization vector, a 16 byte 32 hex char number. This must be the same for all nodes.

Example: `0149AB83FC68D3C1395ABC4692DF931C`

Loglevel: Desired log level for logging events. Examples: `Debug`, `info`, `warn`, `error` or `fatal`.

Timeout: The period, in minutes, between when the white list and black list are polled for changes and possible refresh. The default of 5 minutes should be adequate, but if you change your black/white list, you must wait up to 5 minutes before the changes take effect.

Cryptoconf_timeout: BTEQ session cache for IV is cleared after this interval.

udfaes: Specifies Normal Mode (`udfaes off`) or Fast Mode (`udfaes on`).

6. Invoke the Vormetric Local Cryptoserver with the `-e` command line option. Example:

```
# /opt/vormetric/DataSecurityExpert/agent/pkcs11/teradata/bin/vormetric
_loca_crypto_server -e
```

You are prompted for the PIN which you created in the last step of [“Install Vormetric Protection for Teradata on the Teradata Node” on page 13](#). This PIN is written in encrypted form to the last line of `vormetric_local_crypto_server.conf`.

7. If you are using the Monit process supervision tool, then configure the `/etc/monitrc` file.

Add these five lines:

```
check process vormetric_local_crypto_server with pidfile
/var/run/vormetric_local_crypto_server.pid
  start program = "/etc/init.d/vormetric_local_crypto_server start"
  stop program = "/etc/init.d/vormetric_local_crypto_server stop"
  if failed unixsocket /tmp/vormetric then restart
  if 5 restarts within 5 cycles then timeout
```

8. Start the Vormetric Local Cryptoserver Daemon:

```
# /etc/init.d/vormetric_local_crypto_server start
```

Teradata database users now have access to the UDFs (see [“Vormetric Protection for Teradata Database UDFs” on page 27](#)) and the VAE C API library.

9. Test the UDFs. In the following example, replace SOMEPASSWORD with your actual Teradata user password. The following example tests one of the UDFs. You might want to issue additional SQL requests to test all of them.

```
[/etc/vormetric]# bteq .logon vormetric,vormetric
Teradata BTEQ 14.10.00.10 for LINUX. PID: 11978
Copyright 1984-2014, Teradata Corporation. ALL RIGHTS RESERVED.
Enter your logon or BTEQ command:
.logon vormetric,SOMEPASSWORD
*** Logon successfully completed.
*** Teradata Database Release is 14.10.03.02
*** Teradata Database Version is 14.10.03.02
*** Transaction Semantics are BTET.
*** Session Character Set Name is 'ASCII'.
*** Total elapsed time was 2 seconds.
BTEQ -- Enter your SQL request or BTEQ command:
select vormetric.encrypt_string('hello world','KEY1');
select vormetric.encrypt_string('hello world','KEY1');
*** Query completed. One row found. One column returned.
*** Total elapsed time was 2 seconds.
encrypt_string('hello world','KEY1')
-----
490C35C33A5E07587ED4A6AFB15A16C9
```

Automated install over a Teradata cluster

This feature detects a cluster and ensures that the installation is performed on all of the nodes in the cluster. The user does not have to do anything to make the installer look for the Cluster. This feature is built in. If the Cluster exists, the installer will find it. However, any change made in the configuration file is not propagated automatically on all nodes. It must be done manually on all of the nodes by the Admin.

When trying to install VPTD on a node, the installer detects the presence of the cluster and tries to install the same VPTD on the other nodes. This requires that the installation is performed in the "silent install" mode. This means that all requisite input is given in a file and VPTD is invoked with the `-s` option. If you use the Fingerprint method of registration, then the user must ensure that all of the nodes are added to the GDEAppliance prior to VPTD install.

Upgrade VPTD

The intent of this feature is to detect a cluster and ensure that the upgrade is performed on all of the nodes in the cluster.



NOTE: You must ensure that the crypto server daemon is not running on any single node before uninstallation begins.

When trying to upgrade the VPTD on a node, the installer detects the presence of the cluster and tries to upgrade the same VPTD on the other nodes. If there is a single node, then it upgrades the VPTD from that node as well.

You can upgrade VPTD from v5.2.5.xx to 6.0.2.xx or 6.0.2.xx to 6.0.2.xx. It does not update the existing user modified configuration files. It adds the new configuration files, sample configuration files with new features and binary files.

After the upgrade, the user must follow the manual steps to remove the existing UDFs from the database and perform the new UDFs installation.



Vormetric Protection for Teradata Database UDFs

Vormetric Protection for Teradata Database supports several UDFs to perform encryption and decryption operations on the Teradata database. This chapter contains the following sections:

- [encrypt_cbc\(\) on page 28](#)
- [decrypt_cbc\(\) on page 29](#)
- [encrypt_fpe\(\) on page 29](#)
- [decrypt_fpe\(\) on page 30](#)
- [encrypt_string\(\) on page 31](#)
- [decrypt_data\(\) on page 32](#)
- [encrypt_char\(\) on page 32](#)
- [decrypt_char\(\) on page 33](#)
- [profiles.conf on page 35](#)
- [Dynamic Masking for Tokenization on page 37](#)

Invoking the UDFs

To call one of the user-defined functions (UDFs) described in this chapter, you must invoke the Teradata BTEQ query tool using the following command:

ASCII login:

```
# bteq [.logon <UserID>,<Password>]
```

Unicode login:

```
# bteq -e UTF8 -c UTF16 [.logon <UserID>,<Password>]
```

You can then call the UDFs as follows:

```
BTEQ -- Enter your SQL request or BTEQ command:  
select vormetric. <UDF name>( <UDF parameters>);
```

See the examples in the rest of this chapter.

You can also perform other BTEQ functions at the prompt. For more information about BTEQ, see Teradata documentation.

encrypt_cbc()

Description

Given a cleartext string in Unicode format and a profile name (see [encrypt_char\(\)](#) on page 32), returns its encrypted equivalent.

```
function encrypt_cbc (inputString varchar(8192) CHARACTER SET UNICODE,
inputKeyName varchar(256)) returns varbyte(16400)
```

The input string must adhere to BTEQ requirements, such as escaping special characters. For example, if the input string contains an apostrophe, use a double apostrophe: 'It''s a beautiful day'.

Example

The following example encrypts a credit card number. This assumes that a profile has been set up in *profiles.conf* with the name `ccnum` and the encryption method `aes_cbc_pad`.

```
# select vormetric.encrypt_cbc('1234-9876-5678-6543', 'ccnum');

# select vormetric.encrypt_cbc('1234-9876-5678-6543', 'ccnum');
```

System Response

```
*** Query completed. One row found. One column returned.
*** Total elapsed time was 2 seconds.
```

```
vormetric.encrypt_cbc('1234-9876-5678-6543', 'ccnum')
-----
BE08C791A97E8FAC725DA39AFC071BD30A32C70C514E00B8D6D420501EF8B9D60F06537
CE93
```

decrypt_cbc()

Description

Given an encrypted string created using `encrypt_cbc()` and a profile name (see [encrypt_char\(\)](#) on page 32), returns its decrypted cleartext.

```
function decrypt_cbc (inputString varbyte(16400), inputKeyName
  varchar(256)) returns varchar(8192) CHARACTER SET UNICODE
```

Example

The following example decrypts a credit card number. It is assumed that a profile has been set up in `profiles.conf` with the name `ccnum` and the encryption method `aes_cbc_pad`.

```
# BTEQ -- Enter your SQL request or BTEQ command:
select
vormetric.decrypt_cbc('BE08C791A97E8FAC725DA39AFC071BD30A32C70C514E00B8D6D420
501EF8B9D60F06537CE93'xb,'ccnum');

#select
vormetric.decrypt_cbc('BE08C791A97E8FAC725DA39AFC071BD30A32C70C514E00B8D6D420
501EF8B9D60F06537CE93'xb,'ccnum');

*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.

vormetric.decrypt_cbc('BE08C791A97E8FAC725DA39AFC071BD30A32C70C514E00B8D6
D420501EF8B9D60F06537CE93'xb,'ccnum')
-----
1234-9876-5678-6543
```

encrypt_fpe()

Description

Given a cleartext string in Unicode or Latin characters and a profile name (see [encrypt_char\(\)](#) on page 32), returns its encrypted equivalent.

```
function encrypt_fpe (inputString varchar(8192) CHARACTER SET UNICODE,
  inputKeyName varchar(256)) returns varchar(8192) CHARACTER SET UNICODE
```

To accept Latin characters as input, the `encrypt_fpe()` call must refer to a profile that uses a predefined Latin character set. For more information, see [Using profiles.conf with FPE on page 36](#).

If there are characters in the plain text input which are not specified in the character set, they will be left in their current positions, unchanged, in the tokenized output. If the plain text input

is less than 2 characters (not counting characters that are not specified in the character set), the output of the UDF is the same as the plaintext input. FPE can tokenize only strings with 2 or more characters.

The input string must adhere to BTEQ requirements, such as escaping special characters. For example, if the input string contains an apostrophe, use a double apostrophe: 'It''s a beautiful day'.

Example

The following example encrypts a customer's first and last name. It is assumed that a profile has been set up in *profiles.conf* with the name `tokenize_name` and the encryption method `fpe`.

```
# select vormetric.encrypt_fpe('John Doe','tokenize_name');

# select vormetric.encrypt_fpe('John Doe','tokenize_name');

*** Query completed. One row found. One column returned.
*** Total elapsed time was 2 seconds.

vormetric.encrypt_fpe('John Doe','tokenize_name')
-----
OHBU cCj
```

decrypt_fpe()

Description

Given an encrypted string created using `encrypt_fpe()` and a profile name, returns its decrypted cleartext.

```
function decrypt_fpe (inputString varchar(8192) CHARACTER SET UNICODE,
inputKeyname varchar(256)) returns varchar(8192) CHARACTER SET UNICODE
```

Example

The following example decrypts a customer's first and last name. It is assumed that a profile has been set up in *profiles.conf* with the name `tokenize_name` and the encryption method `fpe`.

BTEQ -- Enter your SQL request or BTEQ command:

```
# select vormetric.decrypt_fpe('OHBU cCj','tokenize_name');

# select vormetric.decrypt_fpe('OHBU cCj','tokenize_name');

*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.

vormetric.decrypt_fpe('OHBU cCj','tokenize_name')
```

 John Doe

encrypt_string()



NOTE: The `encrypt_cbc()` and `encrypt_fpe()` UDFs are recommended. Use of `encrypt_string()` is not recommended.

Description

Given a cleartext string in Latin characters and a key, returns its encrypted equivalent.

```
function encrypt_string (inputString varchar(16384), inputKeyName
    varchar(256)) returns varbyte(16400)
```

The input string must adhere to BTEQ requirements, such as escaping special characters. For example, if the input string contains an apostrophe, use a double apostrophe: 'It''s a beautiful day'.



NOTE: The initialization vector (IV) for `encrypt_string()` is taken from the file `/etc/vormetric/vormetric_local_crypto_server.conf`.

Example

BTEQ -- Enter your SQL request or BTEQ command:

```
# select vormetric.encrypt_string('clear text', 'KEY1');
# select vormetric.encrypt_string('clear text', 'KEY1');
```

```
*** Query completed. One row found. One column returned.
*** Total elapsed time was 2 seconds.
```

```
vormetric.encrypt_string('clear text','KEY1')
```

 97E7BCEB7D8EA55A59BC83BD44609B0A

decrypt_data()

Description

Given an encrypted string created using `encrypt_string()` and the key used to encrypt data, returns its decrypted cleartext.

```
function decrypt_data (inputString varbyte(16400), inputKeyname
  varchar(256)) returns varchar(16384)
```



NOTE: Can not be used to decrypt data that was created using `encrypt_cbc()` or `encrypt_fpe()`. Use `decrypt_cbc()` or `decrypt_fpe()` instead.

Example

BTEQ -- Enter your SQL request or BTEQ command:

```
# select vormetric.decrypt_data('97E7BCEB7D8EA55A59BC83BD44609B0A'xb,'KEY1');
```

```
# select vormetric.decrypt_data('97E7BCEB7D8EA55A59BC83BD44609B0A'xb,'KEY1');
```

```
*** Query completed. One row found. One column returned.
```

```
*** Total elapsed time was 1 second.
```

```
vormetric.decrypt_data('97E7BCEB7D8EA55A59BC83BD44609B0A'XB,'KEY1')
```

```
-----
clear text
```

encrypt_char()

Description

Given a cleartext string in Latin characters, a key and column size returns its encrypted equivalent.

```
function encrypt_char (inputString varchar(16384), inputKeyname varchar(256),
  inputCharcolumnsize INTEGER) returns varbyte(16000)
```



NOTE: The initialization vector (IV) for `encrypt_char()` is taken from the file `/etc/vormetric/vormetric_local_crypto_server.conf`. A sample IV = `000102030405060708090A0B0C0D0E0F`.

Example:

```
CREATE SET TABLE VORMETRIC.testvar2 ,NO FALLBACK ,
      NO BEFORE JOURNAL,
      NO AFTER JOURNAL,
      CHECKSUM = DEFAULT,
      DEFAULT MERGEBLOCKRATIO
      (column1 INTEGER,
       column2 CHAR(13)
      );
```

BTEQ -- Enter your SQL request or BTEQ command:

```
select  column1,vormetric.encrypt_char(column2, 'KEY-NEW' ,13) as
enc_col2 from testvar2;
```

```
select  column1,vormetric.encrypt_char(column2, 'KEY-NEW' ,13) as
enc_col2 from testvar2;
```

```
*** Query completed. 3 rows found. 2 columns returned.
```

```
*** Total elapsed time was 1 second.
```

```
column1 enc_col2
```

```
-----
3 FB8CAA2F658963C73233A59589633B22
1 4C69855E490704EFF93DFEED957201F3
2 5F252180E7F86AE27F7050E3CE514115
```

decrypt_char()

Description

Given an encrypted string created using encrypt_char(), the key used to encrypt data and column size returns its decrypted cleartext.

```
function decrypt_char (inputString varbyte(16000), inputKeyname varchar(256),
inputCharcolumnsize INTEGER) returns varchar(16384)
```



NOTE: Cannot be used to decrypt data that was created using `encrypt_string()`, `encrypt_cbc()` or `encrypt_fpe()`. Use `decrypt_data()`, `decrypt_cbc()` or `decrypt_fpe()` with those UDFs instead.

Example:

See the `encrypt_char` example code to create a table.

BTEQ -- Enter your SQL request or BTEQ command:

```
select column1, '[' || vormetric.decrypt_char(
vormetric.encrypt_char(column2, 'KEY-NEW' ,13) , 'KEY-NEW' ,13)
|| ']' as enc_dec_col2 from testvar2;
```

```
select column1, '[' || vormetric.decrypt_char(
vormetric.encrypt_char(column2, 'KEY-NEW' ,13) , 'KEY-NEW' ,13)
|| ']' as enc_dec_col2 from testvar2;
```

*** Query completed. 3 rows found. 2 columns returned.

*** Total elapsed time was 1 second.

```
column1 enc_dec_col2
```

```
-----
```

```
3 [1236974      ]
```

```
1 [1234567890123]
```

```
2 [rt          ]
```



NOTE: Column size is maintained and filled with trailing spaces in output., which would not be possible using `encrypt_string/decrypt_data`.

profiles.conf

To streamline the invocation of the `encrypt_cbc()`, `decrypt_cbc()`, `encrypt_fpe()` and `decrypt_fpe()` UDFs, the file `profiles.conf` is used. This file contains named profiles that include several values required as input to these UDFs. Provide the profile name as a parameter when invoking the UDF. The UDF looks up the profile name and uses the parameters that are grouped under that profile name.

An example file named `profiles.conf.sample` is provided with the installation software. Make your own copy of this file and change the name to `profiles.conf`.

Be sure to set the file permissions for `profiles.conf` appropriately. The file should be owned by the root user, who can create and edit the file. Other users should be able to view the file, so they can see the profile names in order to reference them in their own UDF calls. It is not recommended to allow users other than root to edit `profiles.conf`. Set the file permissions to 644.



NOTE: Profiles are not supported for the `encrypt_string()` and `decrypt_data()` UDFs.

Using profiles.conf with Unicode Block Cipher UDFs

Profiles that are to be used with the UDFs `encrypt_cbc()` and `decrypt_cbc()` must contain the following key-value pairs:

```
method = aes_cbc_pad
iv = Initialization Vector Number
keyname = Name of the Vormetric encryption key to use
```

Following is a sample `profiles.conf` file that contains two profiles named `ccnum` and `address`:

```
[ccnum]
method = aes_cbc_pad
iv = 000102030405060708090A0B0C0D0E0F
keyname = KEY_1

[address]
method = aes_cbc_pad
iv = 0F0E0D0C0B0A09080706050403020100
keyname = KEY_1
```

With these profile definitions, the following UDF calls can be made:

```
encrypt_cbc('1234-9876-5678-6543', 'ccnum')
encrypt_cbc('2860 Junction Avenue, San Jose, CA 95134', 'address')
```

Using profiles.conf with FPE

Profiles that are to be used with the UDFs `encrypt_fpe()` and `decrypt_fpe()` must contain the following key-value pairs:

```
method = fpe
tweak = 8-byte tweak / auto
charset = One of the values specified in [character_sets]
keyname = Name of the Vormetric encryption key to use
```

If `tweak` is set to `auto`, the tweak is generated automatically for you.

In the `[character_sets]` section of `profiles.conf`, specify one or more named character sets which can be used to tokenize the cleartext input. Define each character set as one or more comma-separated Unicode ranges in UTF16 Big Endian format. Then, in `charset`, provide one of the names from this section.

Example

Following is a sample `profiles.conf` file for use with `encrypt_fpe()` and `decrypt_fpe()`. It defines several character sets, including a Latin character set, then defines two profiles named `tokenize_name` and `tokenize_address`,

```
[character_sets]
latin = 0020-007E
alphanumeric = 0030-0039,0041-005A,0061-007A
hindi = 0905-0939,0958-0961
custom = 0905-0939,0030-0039

[tokenize_name]
method = fpe
tweak = D8E7920AFA330A73
charset = latin
keyname = KEY_FPE_1

[tokenize_address]
method = fpe
tweak = auto
charset = alphanumeric
keyname = KEY_FPE_1
```

With these profile definitions, the following UDF calls can be made:

```
encrypt_fpe('John Doe', 'tokenize_name')
encrypt_fpe('2860 Junction Avenue, San Jose, CA 95134', 'tokenize_address')
```

Dynamic Masking for Tokenization

Masking allows the VPTD administrator to define one mask per user. The masking rules trigger during detokenization, using the UDF, `decrypt_fpe()`. It contains the following features:

- Define masks
- Support partial tokenization capability
- Add prefix and suffix to a token
- Support generation of one-time use token
- Support FPE-Luhn token type

Define Masks

Masks are defined using the configuration file: `/etc/vormetric/masks.conf`. The format of `masks.conf` is similar to that of `profiles.conf`. It is an `.ini` file with sections and key-value pairs. The section name represents the Teradata username. Each section contains the following key value pairs representing the mask definition:

- **Showfirst:** A numeric value representing the number of characters at the beginning which display in the detokenized output. By default, this value is zero.
For example, a `showfirst` value of 4 means that the output would look like 3454-XXXX-XXXX-XXXX.
- **Showlast:** A numeric value representing the number of characters at the end which display in the detokenized output. By default, this value is zero.
For example, a `showlast` value of 4 means that the output would look like XXXX-XXXX-XXXX-8462
- **Maskchar:** The masking character used in the detokenized output. This is mandatory for a valid mask definition.



Warning! If `masks.conf` is not found, or a username is not found in `masks.conf`, then the detokenized output will be shown in its entirety.

Example:

```
# A sample masks.conf
[DBC]
showfirst = 4
showlast = 4
```

```
maskchar = 'X'  
[VORMETRIC]  
showfirst = 0  
showlast = 0  
maskchar = '*'
```

Support partial tokenization capability

You can now have partial tokenization on the input string. This is governed by the profile given as a parameter to the UDF. The following key-value pairs allow partial tokenization:

- **Keepleft:** A numeric value denoting the number of characters which will remain as they are at the beginning of the input string, while the rest of the string is tokenized.

For example, a `keepleft` value of 4 for input 1234-5678-3456-7890 results in output similar to: 1234-6573-7412-8831.

- **Keepright:** A numeric value denoting the number characters which will remain as they are at the end of the input string, while the rest of the string is tokenized.

For example, a `keepright` value of 4 for input 1234-5678-3456-7890 might result in output similar to: 6438-6573-7412-7890.

Example:

```
[tokenize_ccnum]  
method = fpe  
tweak = auto  
charset = digits  
keyname = KEY_FPE_CC  
keepleft = 4
```

Add prefix and suffix to a token

You can add a prefix and/or suffix to the tokenized output through a new key-value pair: **prefix** and **suffix**, using the `profiles.conf` file. If the prefix is defined as a string "Credit Card number: " then the output results in output similar to: Credit Card number: 1234-6352-1738-2343.

The size of the prefix/suffix is limited to 10 characters.

Example:

```
[tokenize_ccnum]  
method = fpe  
tweak = auto  
charset = digits
```

```
keyname = KEY_FPE_CC  
keepleft = 4 prefix = "Credit Card number: "
```

Generating of one-time use token

If you need to generate an irreversible token that can never be detokenized:

1. Use the key-value pair: irreversible.
2. Set it to **yes** to generate a one-time token.

Example:

```
[irreversible_token]  
method = fpe  
tweak = auto  
charset = digits  
keyname = KEY_FPE_CC  
irreversible = yes
```

FPE-Luhn token type

An Luhn check is a checksum algorithm used to verify if a credit card number is valid. Users use FPE so that legacy applications do not break while processing the encrypted data. Running a standard FPE for a credit card number might result in a string which looks like a credit card number, but might fail the Luhn check. VPTD now supports generating an FPE token which will pass the Luhn check. For this, VPTD had added a method called `fpeluhn` in `profiles.conf`. The charset for this profile is ASCII digits, and the keepright directive will not work with this token type.

FPE-Luhn in Teradata is compatible with VTS. It requires 3 or more characters to tokenize.

```
[tokenize_ccnum]  
method = fpeluhn \\ New method to enforce Luhn check validation for the output  
tweak = auto  
charset = digits  
keyname = KEY_FPE_CC
```



VPTD Ongoing Operations

This chapter describes ongoing operations for Vormetric Protection for Teradata Database. This chapter contains the following sections:

- [Log Messages on page 41](#)
- [Troubleshooting on page 41](#)

Log Messages

See `/var/log/messages` for error messages originating from the Vormetric Local Cryptoserver Daemon.

See `/var/log/vormetric/pkcs11_root.log` for information messages originating when encrypt or decrypt operations are performed by the local Cryptoserver Daemon on behalf of the UDFs.

Troubleshooting

During ongoing operation of Vormetric Protection for Teradata Database, the following issues may arise if the appropriate steps are not taken to avoid them. Be aware of these considerations and take the recommended steps to ensure smooth operation.

Be Sure Teradata User Can Access `/tmp/vormetric`

The Vormetric Teradata UDFs and the local Vormetric Cryptoserver communicate through the named socket `/tmp/vormetric`. The UDFs run in the Teradata context, so this socket must have access permission set to 666 to allow access to the Teradata user. If only the root user has permission to access `/tmp/vormetric`, the UDF can not communicate with the Cryptoserver and the UDF will fail.

Cache Key on Host When Turning Off `udf_aes`

If using the `encrypt_fpe()` and `decrypt_fpe()` UDFs with the `udf_aes` option off, be sure the encryption key created on the GDE Appliance is cached on the host.

Otherwise, the UDF generates the following error:

```
error '500 - C_EncryptInit failed'
```

Set Width of BTEQ Session to Avoid Truncated UDF Output

If you are using `encrypt_cbc()` to encrypt long strings, use the following steps to ensure the output of the encryption UDF is not truncated. If the BTEQ session width is too narrow, characters can be lost from the end of the returned string.

1. Invoke BTEQ with the following command:

```
# bteq -e UTF8 -c UTF16
```

2. At the BTEQ prompt, run the following command. Instead of 1000, substitute any value that ensures the width is sufficient:

```
# .set width 1000
```

3. Run the following command:

```
# .set session charset "utf8"
```

Flush the Cache to Remove Old Profiles

When you change the files `profiles.conf`, `mask.conf`, or `vormetric_local_crypto_server.conf`, the updates are not necessarily recognized immediately. Information from an older configuration file could be cached. To be sure the most up-to-date information is available, flush the BTEQ cache using the following steps.



Caution: The database is restarted during this procedure. If any critical operations are underway, wait until they have finished.

After changes are made to either `profiles.conf` or `vormetric_local_crypto_server.conf`:

1. Restart the Cryptoserver.
2. Run the following command:

```
# tpareset -f flushing
```


3. Run the following command:

```
# pdestate -a
```

4. Look for output similar to the following. If this output is not seen, repeat the `pdestate -a` command.

```
PDE state is RUN/STARTED.  
DBS state is 4: Logons are enabled - Users are logged on
```

Properly Escape Characters in Input Strings

If an improperly formatted string is passed as input to BTEQ, such as when calling the UDFs `encrypt_fpe` or `encrypt_cbc`, the BTEQ session stops and waits indefinitely. The input string must adhere to BTEQ requirements, such as escaping special characters. For example, if the input string contains an apostrophe, use a double apostrophe: `'It''s a beautiful day'`. For more information about the requirements for input strings, see Teradata BTEQ documentation.



Locking Down Internet-facing Servers Supporting VPTD



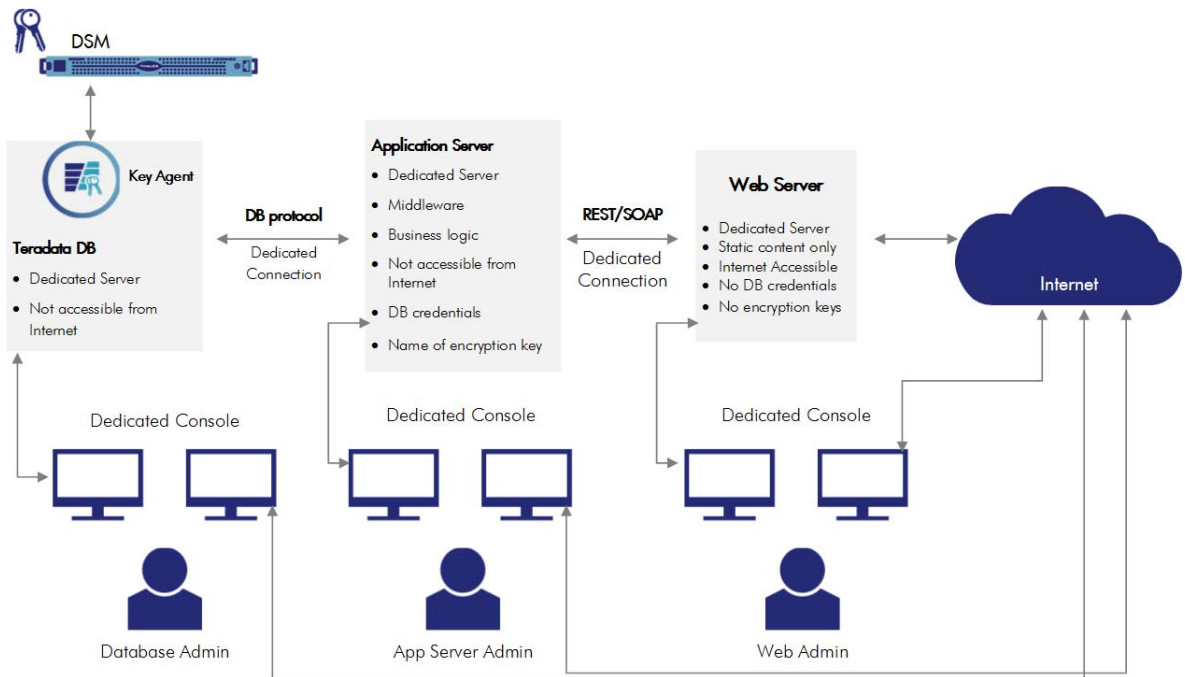
This appendix contains the following sections:

- “Security Tips for Vormetric Protection for Teradata Database” on page 45
- “To Disallow SSH Password Login and Use a Key Pair Login” on page 47

Security Tips for Vormetric Protection for Teradata Database

We recommend having at least two dedicated servers—a Web Server and an Application Server—between the Teradata DB and the Internet. This is shown in the following figure.

Figure 7: High Security Deployment for Teradata Deployment



- **Web Server.** The Internet-facing Web Server should be dedicated (a separate machine—virtual ok—with separate address space, IP, firewall, and so on). You must assume the Web Server will be penetrated, and so ensure that it does not contain DB credentials, encryption key names, or any sensitive data. It can contain static content such as static web pages, graphics and video, but it should not contain business logic or have direct access to the Teradata database. It should have such security features as firewalls, but you should not overload it with functionality.

The Web Server's access to the Teradata DB should be through an Application Server. The Web Server's access to the secure data should come in the form of REST or SOAP API calls to the Application Server, which are fairly traceable, testable, and can be limited in scope.

- **Application Server** should be dedicated and should not be accessible from the Internet. It will have Teradata DB credentials and the name of encryption keys, but not the encryption keys themselves. It will contain the organization's business logic, and when the Application Server receives REST/SOAP requests from the Web Server, it will retrieve the required data from the Teradata DB and send it to the Web Server which will send it along to the requester.
- **Teradata DB** should be dedicated and inaccessible from the Internet. It should pass data only to the Application Server with the appropriate credentials. It should contain the Vormetric Application Encryption (VAE) agent, which is connected to the GDE Appliance.

To support separation of duties, each of the above servers should be administered by a different administrator, a Web Administrator, Application Server Administrator, and Database Administrator. The Web Administrator will not have access to the database, nor access to the key name. The Database Administrator has access to the encrypted data, but lacks the key name to decrypt the data. The Application Server Administrator could decompile the middleware and extract the database credentials and the key name(s), but this is not a straightforward process as long as the application logic is compiled.

Each administrator should perform their administrative duties on their respective servers using a UNIX/Linux or OSX machine, not a Windows machine, which is more vulnerable to hacking. Furthermore, this administrator machine should have an air gap (physical and electronic isolation) from other computers.

The connection from these server administration machines to the dedicated servers should use the following security enhancements:

- Disable unused services such as printer daemons, mail servers and so on.
- Restrict the IP addresses that can log in. For example, only allow connections from corporate headquarters. This can be configured in the SSH configuration and settings.
- Change the SSH port from the default 22 to a random port number (example: 50712). In `/etc/ssh/sshd_config` change port 22 to port 50712
- Disallow SSH password login and use a key pair login instead (see below).

To Disallow SSH Password Login and Use a Key Pair Login

To set up SSH login to disallow passwords and use only key-pair login, first change the following two parameters in the `/etc/ssh/sshd_config` file:

```
PasswordAuthentication no
ChallengeResponseAuthentication no
```

The SSH server will only allow login for users with an SSH key which has been added to `~/.ssh/authorized_keys` file.

Generate user-specific SSH keys by using `ssh-keygen`, which yields a private key file and a public key file (example: `id_rsa` and `id_rsa.pub`).

The private key file must remain in the workstation you use to access your production server. The public key file, however, needs to be appended to the `authorized_keys` file on the server.

Specify a password during key file generation. We strongly recommend password-protecting your private key. If you don't password-protect your private key, anyone with access to your computer conceivably can SSH (without being prompted for a password) to your account on any remote system that has the corresponding public key. You may use an SSH key without a password for automated administration tasks such as administrative shell scripts. However, restrict the scope of these keys by prefixing them with the particular command for which they will be used. For example:

```
command="rsync --server --sender -vIHogDtprxe.iLsf --numeric-ids .
/srv" ssh-rsa AAAA ...
```

These SSH keys used for automation tasks should not use root access. If root access is required, use `sudo`.

You may consider denying root logins by setting the `PermitRootLogin` to `no` in the `sshd_config` file, or allow them only for logins which are tied to a particular command using the `forced-commands-only` option.

Uninstalling VPTD

This appendix contains the following sections:

- “Uninstalling Vormetric Protection for Teradata Database” on page 49
- “Automated uninstall over a Teradata cluster” on page 50

Uninstalling Vormetric Protection for Teradata Database

When trying to uninstall VPTD on a node, the uninstaller detects the presence of the cluster and will try to uninstall VPTD from the other nodes that contain the same version of VPTD. If there is only a single node, then it uninstalls VPTD from that node.

The uninstall script also ensures that the crypto server daemon is not running on any single node before uninstallation begins.



NOTE: You cannot call the uninstaller from within `/opt/vormetric`. You must be outside of this location.

1. Uninstall the VAE executable on Teradata:

```
rpm -qa | grep vae
vae-td14.10-6.0.2-94
rpm -e vae-td14.10-6.0.2-94
```

2. Uninstall files in `/opt/vormetric/DataSecurityExpert/agent` using `/opt/vormetric/DataSecurityExpert/agent/key/bin/uninstall`. These include:
 - Vormetric key agent
 - `/opt/vormetric/DataSecurityExpert/agent/pkcs11/teradata/udfs/install_udfs.bteq.sample`
 - `/opt/vormetric/DataSecurityExpert/agent/pkcs11/teradata/udfs/<renamed bteq.sample file>`
 - `/opt/vormetric/DataSecurityExpert/agent/pkcs11/Teradata/bin/vormetric_local_crypto_server`
3. Uninstall any files that are present in `/etc/vormetric/` with `rm -f`.

4. Remove Teradata UDFs by default in a specific database named “vormetric”. Either drop the database or remove the UDFs (user-defined functions) by issuing the following commands in the bteq shell:

```
DROP FUNCTION encrypt_cbc;
DROP FUNCTION decrypt_cbc;
DROP FUNCTION encrypt_fpe;
DROP FUNCTION decrypt_fpe;
DROP FUNCTION encrypt_string;
DROP FUNCTION decrypt_data;
DROP FUNCTION encrypt_char;
DROP FUNCTION decrypt_char;
```

5. Remove the `init.d` script to start the local crypto server:
 - Delete `/etc/init.d/vormetric_local_crypto_server`

Automated uninstall over a Teradata cluster

The intent of this feature is to detect a cluster and ensure that the uninstallation is performed on all of the the nodes in the cluster.



NOTE: You must ensure that the crypto server daemon is not running on any single node before uninstallation begins.

When trying to uninstall VPTD on a node, the uninstaller would detect the presence of the cluster and try to uninstall the same VPTD on the other nodes. If there is a single node then it will also uninstall the VPTD from that node as well.

The uninstaller resides in

```
/opt/vormetric/DataSecurityExpert/agent/pkcs11/teradata/bin
```

To uninstall VPTD:

1. Type:

```
# /opt/vormetric/DataSecurityExpert/agent/pkcs11/teradata/bin
/vptd_uninstall
```


2. Remove Teradata UDFs by default in a specific database named “vormetric”. Either drop the database or remove the UDFs (user-defined functions) by issuing the following commands in the bteq shell:

```
DROP FUNCTION encrypt_cbc;  
DROP FUNCTION decrypt_cbc;  
DROP FUNCTION encrypt_fpe;  
DROP FUNCTION decrypt_fpe;  
DROP FUNCTION encrypt_string;  
DROP FUNCTION decrypt_data;  
DROP FUNCTION encrypt_char;  
DROP FUNCTION decrypt_char;
```



Warning! You cannot call the uninstaller from `/opt/vormetric` or `opt/vormetric` sub-directories. You can call the uninstaller from any other directory.

