IBM®

# Guardium Data Encryption 3.0

# Application Encryption User Guide

**Release 3**

IBM Guardium Data Encryption 3.0 is the same product as Vormetric Data Security (VDS) Release 6.0.1. VDS Release 6.0.1 consists of Data Security Manager Release 6.0.1 and Vormetric Encryption agent Release 6.0.1. The Key Vault and Key Agent features of Vormetric Data Security are not available with Guardium Data Encryption 3.0.

Vormetric Data Security

# Vormetric Data Security Platform

## Vormetric Application Encryption (VAE)

### Installation and API Reference Guide

Version 5.2.5 Patch

50-1000040-02

DECEMBER 16, 2016
DOCUMENTATION VERSION 1

Vormetric Data Security Platform

*Vormetric Application Encryption (VAE) Installation & API Reference Guide*

Release 5, Version 5.2.5 Patch

December 16, 2016, Documentation Version 1

FAILURE OF ANY TRANSMISSION HARDWARE, TECHNOLOGY, OR SYSTEM USED BY LICENSEE OR ANY LICENSEE CUSTOMER. VORMETRIC SHALL HAVE NO LIABILITY FOR, AND LICENSEE SHALL DEFEND, INDEMNIFY, AND HOLD VORMETRIC 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 VORMETRIC 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 VORMETRIC.

Protected by U.S. patents:

6,678,828

6,931,530

7,143,288

7,283,538

7,334,124

Vormetric 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 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

The *Vormetric Application Encryption (VAE) Installation & API Reference Guide* describes the functionalities and features of VAE. It provides descriptions, syntax, and usage examples for each of the actions and data types for the VAE solution. This document describes the Vormetric implementation of the PKCS #11 standard.

## DOCUMENTATION VERSION HISTORY

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

**Table 1:** Documentation Changes

| | Date | Changes |
|---|---|---|
| 5.2.3 v1 | 7/15/15 | First release for 5.2.3. Fixed section on verifying proper installation on UNIX and Linux systems. |
| 5.2.3 v2 | 8/31/15 | Included information for the new feature of Counter (CTR) Mode block cipher and fixed various technical and editorial issues. |
| 5.2.4 v1 | 2/16/16 | Included information for the new feature of Format Preserving Encryption (FPE), Shared Secret Registration, and fixed technical and editorial issues. |
| 5.2.4 v2 | 3/23/16 | Fixed some typos and information errors with the naming of the binaries for installation. |
| 5.2.5 v1 | 6/6/16 | Fixed some typos and grammatical errors. |
| 5.2.5 v2 | 8/5/16 | Fixed some typos and added note in Installation section about key cache only in memory and not written to disk. |
| 5.2.5 v3 | 9/27/16 | Added Digest functions and FIPS. Partial copy edit to fix wording. |
| 5.2.5 Patch v1 | 12/16/16 | Update System Requirements. Fix Return Values in Digest functions. |

## ASSUMPTIONS

This documentation assumes knowledge of C and PKCS #11.

The system administrator must have root permissions for the systems on which Vormetric Application Encryption (VAE) software is installed.

# RELATED DOCUMENTS

The following documents are available to registered users on the Vormetric Web site at:

https://help.vormetric.com

- *Vormetric Transparent Encryption Agent Installation and Configuration Guide*
- *Vormetric Data Security Platform Administrators Guide*

# TYPOGRAPHICAL CONVENTIONS

This section lists the common typographical conventions for Vormetric technical publications.

***Typographical Conventions***

| Convention | Usage | Example |
|---|---|---|
| **bold regular font** | GUI labels and options. | Click the **System tab** and select **General Preferences**. |
| ***bold italics serif font*** | variables or text to be replaced | `https://<`***Token Server name***`>/admin/`<br>`Enter password: <`***Password***`>` |
| `regular monotype font` | Command and code examples<br>XML examples | Example:<br>`session start`<br>`iptarget=192.168.253.102` |
| *italic regular font* | GUI dialog box titles | The *General Preferences* window opens. |
| | File names, paths, and directories | */usr/bin/* |
| | Emphasis | *Do not resize the page.* |
| | New terminology | *CDF (Carousel Definition Format)* |
| "quotes" | File extensions<br>Attribute values<br>Terms used in special senses | "`.js`", "`.ext`"<br>"true" "false", "0"<br>"1+1" hot standby failover |

# GUIDE TO VAE 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**. In most cases, you will probably use an existing GDE Appliance, but use this document if you need to install a new Data Security Manager (GDE Appliance).
2. **Vormetric Application Encryption (VAE) Installation and API Reference Guide**. (This book.) For developers who want to use application encryption with Vormetric's implementation of PKCS #11.
3. **Vormetric Security Intelligence Configuration Guide**. Use this to integrate your Vormetric VAE events logs with the ArcSight ESM, Splunk, or IBM QRadar
4. **Vormetric Data Security (VDS) Platform Event and Log Messages Reference**. A listing of all the VDS Platform event and log messages with severity, long and short form, and description.

## Searching through all the documents

Technical information for Vormetric products can be spread across many documents. Instead of searching through each individual document to find the information you need, you can use the following procedure to search all of the VTE documents with a single search in Windows (the same process should work for UNIX/Linux):

1. Copy all the .pdf files of a specific product into a single directory. For example, using the Vormetric Transparent Encryption:

```
C:\Documents\PDFs\5.2.3>dir
Admin_Guide_V1.pdf
Agent_Install_&_Config_Guide_v2.pdf
Data_Transformation_Guide_v1.pdf
DSM_Automation_Reference_v1.pdf
DSM_Install_Guide_v1.pdf
Event_&_Log_Messages_Ref_v1.pdf
GettingStarted_v1.pdf
VSI_Reference_v1.pdf
RN_DSM.pdf
RN_Linux.pdf
RN_RHEL7.pdf
RN_UNIX.pdf
RN_Windows.pdf
VDSCompatibilityMatrix.pdf
```

2. Bring up Adobe Reader or Adobe Acrobat.
3. Open any pdf file from that directory: **File > Open > Select File**.
4. Click **Edit > Advanced Search**.
5. Under "*Where would you like to search?*" click "**All PDF Documents in**", then select the directory containing all the VTE PDF files. In this case, `C:\Documents\PDFs\5.2.3`
6. In the "*What word or phrase would you like to search for?*" enter the search phrase and click search.

You can do this with any set of PDF files.

## 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 (DSM), which coordinates policies, keys, and the collection of security intelligence, all of which is managed and observed through your browser. In addition to the DSM, 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** 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

Vormetric's Master Software License and Hardware Purchase Agreement (“MSLA”) 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 Vormetric's MSLA, shall be superseded by the definitions and terms of the MSLA. 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:

- https://help.vormetric.com
- http://support.vormetric.com
- support@vormetric.com
- 877-267-3247

For Vormetric Sales:

- http://enterprise-encryption.vormetric.com/contact-sales.html
- sales@vormetric.com
- 888-267-3732

# Vormetric Application Encryption

1

Vormetric Application Encryption (VAE) enables data encryption at the application level. Applications can use VAE to encrypt a column in a database or a field, such as credit card numbers or Social Security numbers. VAE can also be used to encrypt an entire data file or directory.

VAE provides some of the same functionality as Vormetric Transparent Encryption (VTE), but they differ significantly. VTE does not support application-level data encryption, but it supports encryption of an entire data file or directory. VAE and VTE both offer this file-level encryption, but they use different mechanisms for it.

**NOTE:** The Vormetric Application Encryption software contains the same components as the Vormetric Key Agent, which is used for non-Oracle or non-MS-SQL applications. The installer and configuration for VAE refer to the Key Agent because the installation and configuration process is the same.

**NOTE:** VAE does not have a daemon that pings the GDE Appliance periodically, so it will not know about changes on the GDE Appliance.
If the configuration of the GDE Appliance is changed, such as adding a failover GDE Appliance, the VAE agent must be restarted bycalling `C_Initialize`.

## Product Overview

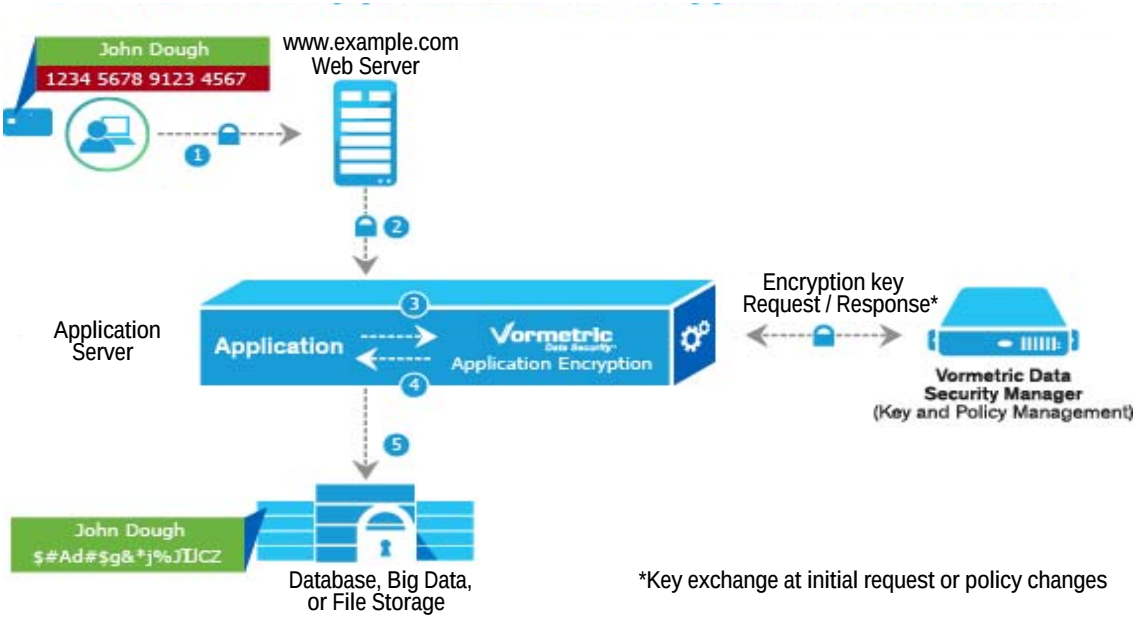
### Application Encryption Workflow

The Application Encryption workflow is shown in Figure 1.

1. A user submits personal information to purchase items from a website.
2. The web server sends personal information to an application server.
3. The application calls Vormetric Application Encryption (VAE) to encrypt the sensitive personal data.

4. VAE returns the encrypted value to the application.
5. The application stores the encrypted value.

**Figure 1:** Application Encryption Workflow




## Components

Vormetric Application Encryption (VAE) makes use of several other components of the Vormetric Data Security (VDS) Platform, including:

### 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. The Key Agent is sometimes called the Vormetric Application Encryption Agent.

## Functionality

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

The APIs in the Vormetric library are a subset of the PKCS #11 specification version 2.20. They are platform-independent to cryptographic tokens, and are traditionally used for HSMs (hardware security modules) and smartcards.

VAE supports single and multi-part encryption and decryption using RSA 1024, RSA 2048, 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.

VAE provides API support for the following:

- Create a key
- Find a key
- Destroy a key
- Export a key
- Import a key
- Encrypt data
- Decrypt data
- Sign data
- Verify data signature
- Compute the digest for data with a key (HMAC)

See “API Reference” on page 33 for a complete list of the supported APIs.

### Key Cache Options

You can choose to store keys on the GDE Appliance only or allow them to also be cached in memory on the host. By default, newly created keys are cached on the host. For greater security, you can keep the keys on the GDE Appliance; however, doing this will affect performance. Table 1 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/high-volume transactions of data. |

| | Key stored on the GDE Appliance | Key cached on host |
|---|---|---|
| Security | - Most secure. The key never leaves the GDE Appliance.<br>- In compliance with PCI security standards | - Key is cached on the client in memory.<br>- In compliance with PCI security standards |

## Supported Agent Operating Systems

The following table lists the supported VAE Agent Agent Operating Systems and languages. Vormetric provides sample code that shows how to call the VAE APIs from C, Java, and C#.

| Operating System | 32/64-bit | Application Encryption (Languages) | | |
|---|---|---|---|---|
| | | C | Java (JDK8)[2, 3] | .NET [7] |
| Windows<br>Server 2008 Standard | 32-bit | No | Oracle JDK[2, 3, 6] | Yes [4, 5] |
| Windows<br>2012 R2 | 64-bit | No | Oracle JDK[6] | Yes |
| Linux<br>Red Hat 6u4 | x86_64-bit | Yes [1] | Open JDK[2, 3] | No |
| Linux<br>Red Hat 6u5<br>2.6.32-431.el6.x86_64 | x86_64-bit | Yes | Open JDK | No |
| Linux<br>Red Hat 7.1 | x86_64-bit | Yes | Open JDK | No |
| Linux<br>Ubuntu 12.04<br>3.11.0-15-generic | x86_64-bit | Yes | Open JDK | No |
| Linux<br>Ubuntu 16 | x86_64-bit | Yes | Open JDK | No |
| Linux<br>(SUSE 10) SLES 10 2.6.16.60-0.85.1-smp | x86_64-bit | Yes | Open JDK | No |
| Linux<br>(SUSE 11) SLES 11SP1<br>2.6.32.12-0.7-default | x86_64-bit | Yes | Open JDK | No |
| Linux<br>(SUSE 11) SLES 11SP2Plus<br>3.0.13-0.27-default | x86_64-bit | Yes | Open JDK | No |
| Linux<br>(SUSE 12) | x86_64-bit | Yes | Open JDK | No |
| AIX 7.1 | NA | Yes | Yes (limited support) | No |

1. Supported through the shared library.
2. Supported through the Sun PKCS#11 `sun.security.pkcs11.wrapper`, which calls through to the C library. This is part of the standard Java install. No additional steps are required.
3. Tested with Java 8.
4. Supported through a shim layer that also calls through to the C library. This is a separate PKCS11interop.dll that we ship with our product and is part of the install.
5. Visual Studio 2010. Visual Studio 2012.
6. Vormetric Key Agent requires the use of the sunpkcs11.jar library, which is available only in the 32-bit version of Java 7. It is available in both 32 and 64-bit versions of Java 8.
7. Vormetric supports NET 4.0 and later.

## GDE Appliance and Agent Software Version Compatibility

The following table shows compatibility between the GDE Appliance software versions and Vormetric Application Encryption agent versions.

**Table 2:** GDE Appliance and Agent Software Compatibility

| GDE Appliance Version | Vormetric Application Encryption (VAE) | |
|---|---|---|
| | **5.2.4** | **5.2.5** |
| **5.3.1** | Compatible | Compatible |
| **5.3** | Compatible | Compatible |
| **5.2.3** | Compatible | Not Compatible |
| **5.2.2** | Not Compatible | Not Compatible |

# Vormetric Application Encryption Installation

2

This chapter describes how to install and configure Vormetric Application Encryption (VAE) on Windows and Linux systems. It contains the following sections:



## Overview

Installing the key agent installs the shared library that enables application encryption. The library (`libvorpkcs11.so` on Linux or `vorpkcs11.dll` on Windows) provides functions that the customer can call to do application encryption and key management.

### Assumptions

- The IP address, routing configuration, and DNS addresses for the hosts where Vormetric Encryption Agents are installed allow connectivity to all DSMs.

**NOTE:** Since the key cache on the VAE agent is only in memory and not written to disk, the VAE (key) agent must ALWAYS connect to the GDE Appliance upon a reboot to retrieve necessary keys. The challenge-response/generate password option (in which the user can type in a password to unlock the agent when the GDE Appliance is off-line) is available only for VTE agents and is NOT supported with the VAE (key) agent.

Because we don't support the host password for challenge response/generate password or a manual passphrase with the VKM (key) agent, there is no work-around for an offline GDE Appliance situation. For continuous access, make sure you have at least 1 healthy GDE

Appliance online in your HA GDE Appliance configuration to serve up the MEK to decrypt table/tablespace keys.

## Installation Plan

Following are the high-level steps for installing and configuring VAE.

1. Verify with the GDE Appliance Security Administrator that the host where you will install the agent has been added to the GDE Appliance.
2. Collect configuration information.
3. Set up the host name resolution method.
4. Open firewall ports, if applicable.
5. Choose the installation method (Silent or Iterative).
6. Select the random number generation method.
7. Decide if you want to enable the anti-cloning functionality.
8. Install the agent software.
9. Verify the installation.

## Agent Install Checklist

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

**Table 3:** VAE Agent Installation Checklist

| Checklist item | Status |
|---|---|
| Obtain the Installation file from Vormetric support.<br>The format for the VAE Agent file name is:<br>`vee-key-<product-version-build-system>.exe`<br>**Examples:**<br>Windows: `vee-key-5.2.5-win64.exe`<br>Linux: `vee-key-5.2.5-rh7-x86_64.bin` | |
| Fully Qualified Domain Name (FQDN) of the GDE Appliance | |
| IP address or Fully Qualified Domain Name (FQDN) of the host | |
| Subnet mask of the host | |

| Checklist item | Status |
|---|---|
| Root password for the host | |
| If using Shared Secret Registration, get from the **GDE Appliance Security Administrator**:<br>1) The shared secret password<br>2) Domain<br>3) Host group if applicable<br>4) A description for the host. | |
| If using the Fingerprint Registration ask **GDE Appliance Administrator** to add host to GDE Appliance database and check **Registration Allowed** check box.<br>After checking the fingerprint, select the **Communication Enabled** check box. | |
| Host system clock set to the correct time zone | |

# Before You Begin

- The Vormetric Data Security Manager (GDE Appliance) must be installed and configured before the VAE Agent is installed.
- The GDE Appliance version must be 5.2.3 or later.
- Do not install the VAE Agent on network-mounted volumes like NFS.

## Determine Your Agent Registration Method

Protected hosts can be registered with the GDE Appliance using either the *Fingerprint method* or the default *Shared Secret method*.

- **The Fingerprint method**—requires the GDE Appliance Security Administrator to add the FQDN or IP address of each protected host to the GDE Appliance database before registering the agent.

  After registration, the installer of the agent passes the CA certificate to the GDE Appliance Security Administrator to verify that the protected host and GDE Appliance share valid certificates.

  If you choose the Fingerprint method, ask the GDE Appliance Security Manager to add the FQDN or IP address of the protected host to the GDE Appliance database before registering that host.
- **The Shared Secret method**—requires the GDE Appliance Security Administrator to create a *shared secret*—a case-sensitive string of characters—for auto-registering a domain or host group.

Agent Installers use the shared secret to add and register protected hosts to the GDE Appliance for a domain or host group. The GDE Appliance Security Administrators can optionally add host names or IP addresses to the GDE Appliance database, but this is done automatically using the Shared Secret Registration method, and there is no need to verify that the protected host and GDE Appliance share valid certificates. Multiple protected hosts can be added dynamically with a single shared secret password during the agent installation and registration process.

If you choose the Shared Secret method, ask the GDE Appliance Security Manager to create a shared secret for the domain or host group in which the new protected host will reside. Then, get the shared secret and the validity period (one hour, day, week, or month) and register within that period.

There is a "**Require that hosts are first added**" checkbox in GDE Appliance Shared Secret creation page. If this box is checked, the hosts must be manually added to the GDE Appliance database.

## Host Name Resolution

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

You can also modify the *hosts* file on the DSM 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 DSMs and the protected hosts:
  - **Modify the *hosts* file on the DSM:** To use names like *serverx.domain.com,* enter the host names and matching IP addresses in the `/etc/hosts` file on the DSM using the `host` command under the *network* menu. For example:

    ```
    0011:network$ host add jblaster-dev1 10.3.42.12
    SUCCESS: add host
    0012:network$ host sh
    name=localhost6.localdomain6 ip=::1
    name=linux32-48215.sacdbackup.com ip=10.3.48.215
    name=jblaster-dev1 ip=10.3.42.12
    SUCCESS: show host
    ```

    You must do this on *each* DSM, since entries in the **hosts** file are not replicated across DSMs.
  - **Modify the *hosts* file on the protected hosts:** Enter the DSM host names and matching IP addresses in the `/etc/hosts` file on the protected host. *You must do this on EACH protected host making sure to add an entry for all DSM nodes (if using HA).*

OR

- **Use IP addresses:** If using IP addresses as protected host names, you must enable Agent IP in the DSM (see "`agentip`" `Enable/Disable` in the "`network`" menu in the CLI). 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 have to use an IP address when Agent IP is enabled.

## Using the CLI

**NOTE:** The Vormetric Data Security Manager (GDE Appliance) is sometimes referred to in the code as a "`Security Server`" or a "`Data Security Server`" or sometimes as just the "`Server`".

Access the CLI menu as follows:

1. Start the serial console application (for example, HyperTerminal).
2. If the login prompt is not displayed, press the **Enter** key to wake up the connection.
3. Log on to 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
```

# Initial Setup

The following steps describe how to install the VAE Agent for the first time.

The 5.2.5 VAE Agent will work only with the 5.2.3 GDE Appliance and later.

## To Install the VAE Agent on Windows

**NOTE:** The minimum length for the PIN is 8 characters and maximum length is 63 characters. The PIN cannot contain the $ symbol.

1. Log on to the host as a Windows user with administrative privileges.
2. Copy the installation file onto the Windows system.
3. Double-click the installation file. The *Welcome* window opens. Verify the version of VAE you are installing.
4. Click **Next**. The License Agreement appears.
5. Accept the License Agreement and then click **Next**. The *Destination Folder* window opens.
6. Click **Next** to accept the default folder. *The Ready to Install the Program* screen opens.

**NOTE:** If you have a Vormetric Transparent Encryption (VTE) (file system) agent already installed on the system, you cannot change the Destination Folder.

7. Click **Install**. The agent software installs. This may take a few minutes.
8. When the install is finished, the *Install Shield Wizard Completed* screen opens. **Register Vormetric Key Agent (64 bit) now** is selected.

**NOTE:** If you have a specific plan to register the agent later, clear the check box for **Register Vormetric Key Agent (64 bit) now**, click **Finish**, and then skip the rest of the steps in this section.

9. Click **Finish** to start the registration process. The *Register Host* window opens. Click **Next**. Verify the correct agent type is selected.
10. Click **Next**. Type the name of the GDE Appliance. This name must match the **Server name** on the **Dashboard** of the GDE Appliance.

**NOTE:** If you already have a file system agent installed on the system, you do not need to register the agent again unless you want to activate or deactivate cloning prevention.

11. Click **Next**. Enter the name of the machine hosting the agent. You can type the name or select it from the drop-down menu. This name must match the host name added to the GDE Appliance database by the Security Administrator.
12. Click **Next.** Enter a password. This is the PIN that will authenticate this device in the PKCS #11 applications you write.

**NOTE:** If you are in a cluster environment, enter the same password for all cluster nodes.

13. Click the **Enable hardware association** box to enable the cloning prevention function.
14. Click **Register**. The *Register Host* window displays the VAE Agent version and a pop-up window displays the signer CA certificate fingerprint.

    At this stage of the installation, the host administrator and GDE Appliance Security Administrator must exchange information to confirm that the agent host and GDE Appliance share valid certificates.
15. **Host Admin:** Send the fingerprint to the GDE Appliance Security Administrator and wait for confirmation.
16. **GDE Appliance Security Admin**:

    a: Log onto the GDE Appliance Management Console and navigate to the domain where the host was added.

    b: Click the **Dashboard** tab.

    c: Match the fingerprint from the Host Admin with the appropriate **fingerprint** on the Dashboard.

    d: Advise the Host Admin of the results.
17. **Host Admin:** If the fingerprints match, type ‘Y’ and then press **Enter**.
18. **Host Admin:** The fingerprint for the certificate is displayed. Pass this fingerprint to the GDE Appliance Security Admin.
19. **GDE Appliance Security Admin**:

    a: Click the **Host** tab.

    b: Click the name of the host where you just installed the agent. The **Edit host** screen opens.

    c: Match the fingerprint from the Host Admin with the fingerprint in the **Certificate Fingerprint** column.

    d: Advise the Host Admin of the results.

20. **Host Admin:** If the fingerprints match, click **OK**. A message that the installation was a success is displayed.
21. **GDE Appliance Security Admin**: On the GDE Appliance, select the **Communication Enabled** check box for the host.

**NOTE:** After installation, the PIN or password must be provided by the application developer when prompted during `C_Login`. ***Do not forget this password.*** To change the password, you must re-register the agent.

### To Verify the Installation on Windows

Verify the installation by checking agent processes.

1. In the system tray, right-click the Vormetric icon.
2. Select Status. Review the information in the Vormetric Status window to confirm the correct agent or agents are installed and registered.

## To Install the VAE Agent on Linux/UNIX

**NOTE:** Password minimum length for PIN is 8 characters and maximum length is 63 characters.

1. Log on to the host where you will install the VAE Agent. You must have root access.
2. Copy or mount the installation file to the host system.
3. Start the installation. At the prompt, type

   **`./vee-key-<product-version-build-system>.bin`**

   **Example:** >

   **`./vee-key-5.2.5-rh7-x86_64.bin`**
4. Type '**y**' and press **Enter** to accept the Vormetric License Agreement. The installation proceeds.

**NOTE:** The Vormetric Data Security Manager (GDE Appliance) is sometimes referred to in the code as a "`Security Server`" or a "`Data Security Server`" or sometimes as just the "`Server`".

```
Welcome to the Vormetric Key Agent
Registration Program.
Agent Type: Vormetric Key Agent
Agent Version: 5.2.5.39
```

```
In order to register the Vormetric Key Agent
with a Vormetric Data Security Server:
    1) you must know the host name of the machine running the
       Security Server (the host name is displayed on the
       Dashboard window of the Management Console), and
    2) unless you intend to use the 'shared secret' registration method,
       the agent's host machine must be pre-configured on the
       Security Server as a host with the 'Reg. Allowed'
       checkbox enabled for this agent type on the Hosts window
       of the Management Console.
```

5. At the following prompt, type '**y**' or press **Enter** to proceed with registration, or type '**n**' if you specifically plan to register later.

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

The following dialog will display:

```
Please enter the primary Security Server host name: GDE Appliance-dg-15208.com
You entered the host name GDE Appliance-dg-15208.com
Is this host name correct? (Y/N) [Y]: Y
Please enter the host name of this machine, or select from the following list.
If using the "fingerprint" registration method, the name you provide must
precisely match the name used on the "Add Host" page of the Management
Console.
[1] h55119.dg.com
[2] 10.3.55.119
Enter a number, or type a different host name or IP address in manually:
What is the name of this machine? [1]:
You selected "h55119.dg.com".


Would you like to register to the Security Server using a registration shared
secret (S) or using fingerprints (F)? (S/F) [S]:
What is the registration shared secret?
Please enter the domain name for this host: key_encrypt_domain
Please enter the host group name for this host, if any:
Please enter a description for this host: example
Shared secret    : ********
Domain name      : key_encrypt_domain
Host Group       : (none)
```

```
Host description : example
Are the above values correct?  (Y/N) [Y]:
```

6. 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**. A list of host names appears.
   c. From the list, type the number of the host name of the local machine, or manually enter the host name, 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, press **Enter** to enable cloning prevention functionality.

   **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 can be
rectified by running this registration program again.
Do you want to enable this functionality?  (Y/N) [Y]:
```

8. (Linux only) At the following prompt:
   - Typical install: type “**n**” to use the default random number generator.

   **Example:**

```
Do you want to configure the agent to use /dev/random for the source of random
numbers? Doing so has security benefits but could cause significant delays
during installation and startup. The default behavior (answering No) is to use
/dev/urandom, which has no associated delay.
Would you like to use /dev/random for random numbers? (Y/N) [N]:
Generating certificate...done.
```

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

9. The CA certificate fingerprint is displayed.

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

10. Enter the password for the VAE (Key Agent) library and confirm your entry.

```
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 GDE Appliance-dg-15208.com.
```

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

**NOTE:** After installation, the PIN or password must be provided by the application developer when prompted during `C_Login`. ***Do not forget this password.*** To change the password, you must re-register the agent.

## To Verify the Installation on Linux

The best way to check if everything is correctly installed:

1. Register the agent against a GDE Appliance.
2. Run sample code such as `create_key`.

3. Check that the key is on the GDE Appliance by using the GDE Appliance GUI after running the `create_key` sample.

# Modify the Key Cache

You can modify two settings in the key cache on the GDE Appliance.

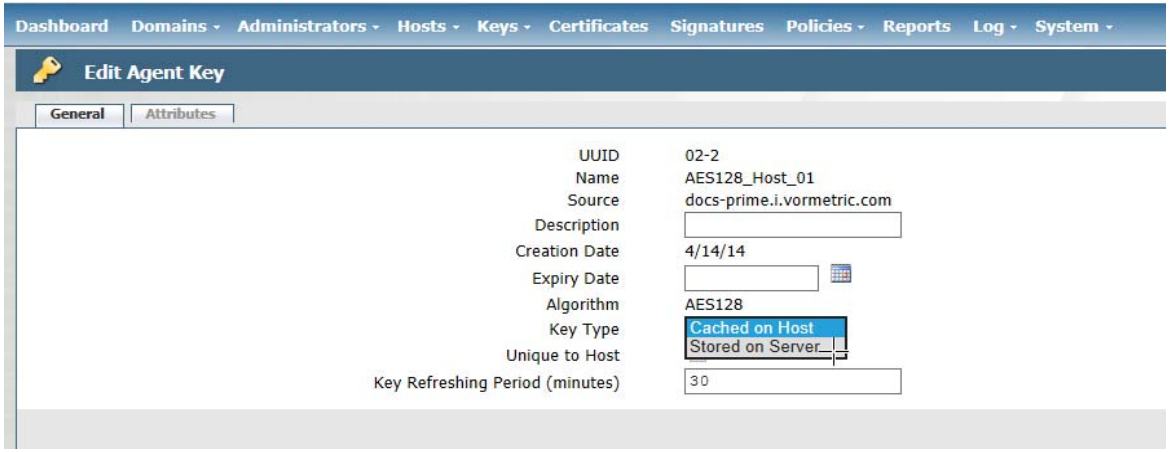
By default, keys are cached on the host, with a time-to-live value of 40320 minutes (4 weeks).

- 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.

## 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.

**Figure 2:** Edit Agent Key window




4. In the **Key Type** drop down, 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 to exist in the local key cache before it is refetched from the GDE Appliance. Then click **Ok**.

# 3 Using the VAE API

This chapter is for application developers. It describes how to use the Vormetric Application Encryption (VAE) APIs to integrate VAE functionality with your applications. It contains the following sections:



## Sample Code

VAE provides code examples for C and Java. The sample code demonstrates the following functionality:

- Create a symmetric key
- Search for and delete a key
- Encrypt and decrypt a single message
- Encrypt and decrypt a file
- Import a key into the GDE Appliance
- Export a wrapped key from the GDE Appliance
- Create an asymmetric key pair and sign

Download and extract the most recent samples and documentation from:

https://help.vormetric.com

Place the files in the following directory:

```
<install directory>/vormetric/DataSecurityExpert/agent/pkcs11/
```

C sample files are located at:

```
.../samples/c_samples
```

Java sample files are located at:

```
.../samples/java_samples
```

Refer to the README file in the samples directory for more information.

### Compiling and Running Sample Code in c_samples

To compile and run the sample code provided in `c_samples`, use `gmake`. (Previously, the `make` command was used on Linux). Use the following commands.

To clean all the files:

```
gmake clean
```

To compile:

```
gmake
```

To run all samples:

```
gmake PASSWORD=<your password here> run
```

## Location of Libraries, Samples, and Logs

The default location of the library files on Linux systems is:

```
/opt/vormetric/DataSecurityExpert/agent/pkcs11/lib
```

Linux code samples:

```
<install directory>/vormetric/DataSecurityExpert/agent/pkcs11/samples
```

On a Windows 32-bit machine, find the sample and library files as follows:

- Samples:

```
C:\Program Files\Vormetric\DataSecurityExpert\agent\pkcs11\samples
```

- Library installation directory:

```
C:\Program Files\Vormetric\DataSecurityExpert\agent\pkcs11\bin
```

On a Windows 64-bit machine, find the sample and library files as follows:

- 32-bit samples:

```
C:\Program Files(x86)\Vormetric\DataSecurityExpert\agent\pkcs11\samples
```

- 32-bit library installation directory:

```
C:\Program Files(x86)\Vormetric\DataSecurityExpert\agent\pkcs11\bin
```

- 64-bit samples:

```
C:\Program Files\Vormetric\DataSecurityExpert\agent\pkcs11\samples
```

- 64-bit library installation directory:

```
C:\Program Files\Vormetric\DataSecurityExpert\agent\pkcs11\bin
```

The Windows logging file uses the following path:

```
%\ProgramData\Vormetric\DataSecurityExpert\agent\log
```

## Running in FIPS Mode

Unlike some code libraries, the VAE library always runs in FIPS mode. The library meets FIPS 140-2 Level 1 requirements. It is not necessary to call any additional functions or take any other steps to enable FIPS mode.

## Verifying Successful API Initialization

Each time the application linked to the VAE library starts up, the VAE library checks itself to be sure that the cryptographic functions are working correctly and the library integrity has not been compromised. This self-test runs when the cryptographic library is loaded, before any `pkcs11` library function calls.

To check whether the API passed this test, view the log files:

- Linux: `/var/log/vorpkcs11/selftest.log`
- Windows: `vorpkcs11_startup.log` in the folder `C:\ProgramData\Vormetric\DataSecurityExpert\agent\log\`

If the self-test is successful, the following messages appear in the log file for each startup of the application running the library:

Linux:

```
VAE Library (45771): Tue Sep 20 14:40:21 2016 [SUCCESS] selftest passed!
VAE Library (45771): Tue Sep 20 14:40:21 2016 [SUCCESS] integrity check passed!
```

Windows:

```
VAE Library: Tue Sep 20 14:40:21 2016 [SUCCESS] selftest passed!
VAE Library: Tue Sep 20 14:40:21 2016 [SUCCESS] integrity check passed!
```

After this verification step, the application can access the cryptographic functions.

If the verification fails, the VAE library logs a message and exits with status 255. The messages logged my either be related to an integrity failure or a Know Answer Test (KAT) failure.

Linux failure messages:

```
VAE Library (45771): Tue Sep 20 14:40:21 2016 [FAILURE] selftest failed!
```

```
VAE Library (45771): Tue Sep 20 14:40:21 2016 [TERMINATE] exiting due to
failure!

VAE Library (45771): Tue Sep 20 14:40:21 2016 [FAILURE] integrity check
failed!

VAE Library (45771): Tue Sep 20 14:40:21 2016 [TERMINATE] exiting due to
failure!
```

Windows error messages:

```
VAE Library: Tue Sep 20 14:40:21 2016 [FAILURE] selftest failed!

VAE Library: Tue Sep 20 14:40:21 2016 [TERMINATE] exiting due to failure!

VAE Library: Tue Sep 20 14:40:21 2016 [FAILURE] integrity check failed!

VAE Library: Tue Sep 20 14:40:21 2016 [TERMINATE] exiting due to failure!
```

# Python Integration and Sample Code

## Overview

This section shows you how to integrate the Vormetric Application Encryption (VAE) library with Python.

**NOTE:** Python integration is not supported for CTR and FPE mode.

Python is a popular general purpose scripting language that allows you to do the following with the VAE library:

- Create a key
- Delete a key
- Single-part encrypt/decrypt
- Multi-part encrypt/decrypt
- Import a key (to the GDE Appliance)
- Sign

## Prerequisites

Before using the VAE library with Python you must have the following:

- The Vormetric VAE library must be installed and registered with a Data Security Manager
- Python 2.6 or 2.7
- PyKCS11 version 1.2.4
- argparse version 1.2.1

## Run a sample

The following samples are available for Python:

| Source code archive | Change from previous archive |
|---|---|
| `python_sample_8-29-2014.tar` | None. |
| `python_sample_9-5-2014.tar` | Changed 1 to True in vpkcs11_sample_helper.py and vpkcs11_sample_find_delete_key.py |

| Source code archive | Change from previous archive |
|---|---|
| `python_samples_9-16-2014.tar` | Various bug fixes. |
| `python_sample_9-19-2014.tar` | bug fixes for:<br>PKCS-970 Win32 vpkcs11_sample_encrypt_decrypt_multipart.py<br>PKCS-977 Python Sampe READ ME file revised<br>PKCS-959 Python Sample, print out of -h option updated<br>PKCS-962 python encrypt-decrypt script with key created in GDE Appliance |
| `python_samples_9-22-2014.tar` | PKCS-984 python vpkcs11_sample_create_keypair_sign.py with existing key. |

To run the samples, un-tar the files into a directory of your choice. Once un-tared, you should see the following files:

- `multipart_test_file.txt`
- `README_vpkcs11_python_samples.txt`
- `requirements.txt`
- `vpkcs11_sample_create_keypair_sign.py`
- `vpkcs11_sample_create_key.py`
- `vpkcs11_sample_create_object.py`
- `vpkcs11_sample_encrypt_decrypt_multipart.py`
- `vpkcs11_sample_encrypt_decrypt.py`
- `vpkcs11_sample_find_delete_key.py`
- `vpkcs11_sample_helper.py`

Use the un-tared files for the following functions:

## Create a key

The `vpkcs11_sample_create_key.py` file demonstrates how to create a key on the Data Security Manager through the VAE library. The keyname is specified in the program "vpkcs11_sample_python_key". The program first searches the GDE Appliance for the existence of a key with the same name using `C_FindObjects.` If the key already exists on the GDE Appliance, we will not create it again. If the key does not exist in the GDE Appliance, we will create it.

## Delete a key

The `vpkcs11_sample_find_delete_key.py` file demonstrates how to search for and delete a key on the GDE Appliance.

## Single-part encryption and decryption

The `vpkcs11_sample_encrypt_decrypt.py` demonstrates single-part encryption/decryption. A keyname is specified in the program, then the program ensures that the key exists on the GDE Appliance. The program then takes the plaintext specified in the command line from the user, encrypts the text, then decrypts the encrypted text and compares the plaintext and the decrypted text to ensure that they are the same.

## Multi-part encryption

The `vpkcs11_sample_encrypt_decrypt_multipart.py` file demonstrates the following: A keyname is specified in the program. The program ensures that the key exists. Next, the program takes the filename specified in the command line, encrypts the file and writes it out. A different version of the program takes the encrypted file, decrypts it, and writes it out to a different file. You can verify that the two files are the same using the following command:

```
>diff <originalfile> <orginalfile>_decrypted.txt
```

## Import a Key into the Data Security Manager (GDE Appliance)

The `vpkcs11_sample_create_key.py` file demonstrates how to import a key into the GDE Appliance. 32 bytes are specified in the program and the program then imports these 32 bytes into the GDE Appliance to create a symmetric key.

## Sign

The `vpkcs11_sample_create_keypair_sign.py` file demonstrates how to create a key pair on the GDE Appliance and use it to sign some data.

# Metadata Logging and Sample Code

## Overview

This section shows you how to use metadata logging with the Vormetric Application Encryption (VAE) library.

Metadata logging is used to pass extra information (such as user name, user login ID or any other user-specified information), associated with the user who makes a particular function call.

Two calls are required for a corresponding function to create metadata logging. In the first call, the input buffer contains the user metadata to be logged. The corresponding buffer length passed must be zero. The second call follows the regular PKCS #11 calling conventions by passing in the allocated buffer and its corresponding length.

### C Sample

The following example is for `vpkcs11_sample_metadata_logging.c`:

```
encryptAndDecrypt ()
. . .
CK_BYTE*         cipherText = NULL_PTR;
CK_ULONG         cipherTextLen = 0;
CK_ULONG         metaDataLen = 256;
CK_CHAR          metadata[] = "META: This is a test
metadata/Encryption: user: tester: hostname" ;
CK_CHAR          metadata2[] = "META: This is a test
metadata/Decryption: user: tester: hostname" ;

/* For C_Decrypt */
CK_BYTE*         decryptedText = NULL_PTR;
CK_ULONG         decryptedTextLen = 0;
/* General */
CK_RV rc = CKR_OK;
. . .
cipherText = (CK_BYTE *)calloc( 1, sizeof(CK_BYTE) * metaDataLen );
if(cipherText != NULL)
{
sprintf((char*)cipherText, "%s", (char *)metadata);
}
printf ("Plain Text length: %d\n", (int)sizeof( plainText));
/* first call C_Encrypt by pass in metadata and obtain cipherText buffer
size upon CKR_OK return */
```

```
rc = FunctionListFuncPtr->C_Encrypt(
                              hSession,
                              plainText, sizeof (plainText),
                              cipherText,&cipherTextLen
                              );
if (rc != CKR_OK){
                    printf ("C_Encrypt failed\n");
                    return rc;
}
else
{
printf ("C_Encrypt succeed, cipherTextLen is : %ld \n",
(long)cipherTextLen);
cipherText = (CK_BYTE *)calloc( 1, sizeof(CK_BYTE) * cipherTextLen );
}
. . .
```

### Java Sample

Metadata samples for Java can be found in the following files:

- `EncryptDecryptMetaData.java`
- `EncryptDecryptFileMeta.java`

## JAVA - AIX 7.1 System Samples

AIX samples will be made available only at customer request (with limitations – some Java samples are not supported on the AIX platform). VAE provides limited support for AIX.

IBM JDK has its own PKCS #11 implementation, which differs from the Oracle Sun implementation. A different set of samples is therefore necessary on AIX due to the differences. In addition, certain functionalities are not supported in the IBM PKCS #11 implementation.

**Samples that don't work with the IBM PKCS #11 implementation are:**

- Encryption/decryption with AES CTR mode
- Encryption/decryption with Vormetric FPE mode
- Encryption/decryption with metadata prepending for logging

**Samples that work with the IBM PKCS #11 implementation are:**

- Create an AES256 key
- Create an RSA2048 key pair

- Delete a key
- Encrypt and decrypt message with an AES256 key
- Encrypt and decrypt file with an AES256 key
- Sign and verify message with an RSA2048 key pair
- Wrap and export a key from Data Security Manager
- Encrypt and decrypt messages with an RSA2048 key pair

# Troubleshooting

## HP-UX11v3: Apache Ant 1.8.4 Requirements

Apache Ant 1.8.4, the latest version supported on HP-UX 11v3, does not work with Java 8 source.

As a workaround, use the following command-line flag to force Ant to use the javac1.7 compiler:

```
prompt> ant -Dbuild.compiler=javac1.7
```

On any other platform, if you have Java 8 source code, upgrade your Ant version to 1.9x.

# Vormetric Encryption Algorithms

4

This chapter describes the encryption algorithms used with the Vormetric Application Encryption (VAE) Agent on Windows and Linux systems.

## Symmetric Block Cypher Modes

**ECB Mode**—used to encrypt small, non-repeating data sets, such as a PIN value on the magnetic strip of a credit card. Use if you have short plaintext and you aren't worried about dictionary attacks.

Size: multiples of 16 bytes.

**NOTE:** The key must be Cached on Host.

**CBC Mode**—used when encrypting items such as MS-Word documents or MS-Excel spreadsheets. Use if you have long plaintext. By chaining, it avoids the disadvantages of dictionary attacks experienced by ECB mode.

Size: multiples of 16 bytes.

**CBC_PAD Mode**—use if you have long plain-text. By chaining, it avoids the disadvantages of dictionary attacks experienced by ECB mode.

Size: arbitrary

Block cipher modes expect plaintext in blocks of multiples of 16. Since data in the real world is not necessarily in multiples of 16 bytes, padding schemes are used to make the data length 16 bytes. If you specify a padding scheme, the plaintext is padded BEFORE encryption.

You can specify a padding scheme in the CKM mechanism in `C_EncryptInit` or `C_DecryptInit`.

- `CKM_AES_CBC`

  No padding. The input must be a multiple of the block size (16 bytes) otherwise an error is returned. The output is the same size as the input.

- `CKM_AES_CBC_PAD`
  - This uses PKCS#5 padding (also known as PKCS#7 padding) where the plaintext is padded to the 16-byte boundary before encryption.

- Because of the padding scheme used, the plaintext length is derivable from the ciphertext.
- PKCS#5 padding is a scheme where the plaintext is padded with the number of padded bytes. The user can look into the last bytes of the decrypted text to see how many bytes they must strip off.
- Plaintext that is already byte-aligned are padded to the next block. For example, 16 bytes of plaintext is padded to 32 bytes.
- In the following example the block size is 16 bytes and padding is required for 4 bytes.

  Example:

```
... | DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD | DD DD DD DD DD DD DD
DD DD DD DD 04 04 04 04 |
```

**CTR Mode**—used when a transmission protocol such as VPN must transmit encrypted data over a network. Allows starting of decryption in the middle of cypher-text.

Size: arbitrary

**NOTE:** For CTR mode, Java users must run Java version 1.7 and up. The sun.security.pkcs11.wrapper started supporting CTR mode in 1.7.

**NOTE:** The key must be Cached on Host.

**FPE Mode**—used to encode within a given input length and within a given ASCII or Unicode character set. Most suitable if you do not want to change the database schema.

- For FPE Mode, the tweak and alphabet must exactly match in the encrypt and decrypt calls for the ciphertext to be decrypted to the original plaintext.
- FPE does not work on 1-character plaintexts. You must have at least 2 characters in the plaintext to be encrypted.
- In FPE, character sets cannot have duplicate characters.
- In FPE, the characters in the plaintext must be in the alphabet specified.
- When FPE is used with an ASCII character set, only ASCII codes 0 – 127 are supported. If a larger character set is desired, use a range of Unicode characters.

**NOTE:** The key must be Cached on Host.

# API Reference

5

## API List

| | |
|---|---|
| "General Purpose Functions" on page 35 | |
| "C_Initialize" | Initializes the pkcs11 library. |
| "C_Finalize" | Called to indicate that an application is finished with using the pkcs11 library. |
| "C_GetInfo" | Provides manufacturer and version information about the library. |
| "C_GetFunctionList" | Tells the application (caller of the pkcs11 library) which APIs are supported. |
| "Slot and Token Management Functions" on page 40 | |
| "C_GetSlotList" | Provides list of available slots. It can be a single default slot. |
| "C_GetSlotInfo" | Provides information about a particular slot in the system. |
| "C_GetTokenInfo" | Provides information about a specific token. |
| "C_GetMechanismList" | Provides a list of mechanisms supported. For example, AES_CBC. |
| "C_GetMechanismInfo" | Provides information about a mechanism possibly supported by the token. |
| "Session Management Functions" on page 47 | |
| "C_OpenSession" | Starts a cryptographic session with a specific token. Session can be defined as read-write session. |
| "C_CloseSession" | Closes a cryptographic session. |
| "C_CloseAllSessions" | Closes all sessions on a slot. |
| "C_GetSessionInfo" | Provides information about a session. |
| "C_Login" | Logs a user into a token. |
| "C_Logout" | Logs a user out from a token. |
| "Object Management Functions" on page 55 | |
| "C_WrapKey" | Supports key export where a symmetric key is wrapped with another symmetric key on the GDE Appliance and then exported. |
| "C_CreateObject" | Imports key bytes into the GDE Appliance and creates a key on the GDE Appliance. |
| "C_DestroyObject" | Destroys an object. |
| "C_FindObjectsInit" | Initializes a search for a token and session objects that match a template. |
| "C_FindObjects" | Finds an object from a specific token. |
| "C_FindObjectsFinal" | Terminates a search for a token and session objects. |
| "C_GetAttributeValue" | Gets attribute value of a specific Security Object. Argument specifies the Security Object ID. |

| | |
|---|---|
| "C_SetAttributeValue" | Sets attribute value of a specific Security Object. Argument specifies the Security Object ID. |
| "C_GenerateKey" | Generates a secret or symmetric key. |
| "C_GenerateKeyPair" | Generates a key pair. Arguments specify RSA type and length of the key pair. |
| "Digest and MACing Functions" on page 73 | |
| "C_DigestInit" | Initializes a digest operation. |
| "C_Digest" | Compute data digest in a single part. Must call C_DigestInit before calling C_Digest. |
| "C_DigestKey" | Specifies HMAC key for digest operation. Must call C_DigestInit before calling C_DigestKey. |
| "C_DigestUpdate" | Continues a multi-part digest operation. Must call C_DigestInit and optionally C_DigestKey before calling C_DigestUpdate. |
| "C_DigestFinal" | Finishes a multi-part digest operation. |
| "Signing and MACing Functions" on page 79 | |
| "C_SignInit" | Initializes a signature operation. |
| "C_Sign" | Signs data in a single part. |
| "C_Verify_Init" | Verifies the signature initialization operation. |
| "C_Verify" | Verifies the signature. |
| "Encryption Functions" on page 84 | |
| "C_EncryptInit" | Initializes an encryption operation. |
| "C_Encrypt" | Encrypts single part data. Must call `C_EncryptInit` before calling `C_Encrypt`. |
| "C_EncryptUpdate" | Continues a multi-part encryption. Must call `C_Encrypt` before calling `C_EncryptUpdate`. |
| "C_EncryptFinal" | Finishes a multi-part encryption. |
| "Decryption Functions" on page 93 | |
| "C_DecryptInit" | Initializes a decryption operation. |
| "C_Decrypt" | Decrypts encrypted data in a single part. Must call `C_DecryptInit` before calling `C_Decrypt`. |
| "C_DecryptUpdate" | Decrypts multi-part encrypted data. Must call C_Decrypt before calling C_DecryptUpdate. |
| "C_DecryptFinal" | Finishes a multi-part decryption to `C_DecryptInit.` |

# General Purpose Functions

# C_Initialize

## Description

Initializes the pkcs11 library.

```
CK_DEFINE_FUNCTION(CK_RV, C_Initialize)(
  CK_VOID_PTR pInitArgs
);
```

**NOTE:** Only the applications written in C will call C_Initialize. Java and .NET versions will not. For more examples, see the sample code.

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_VOID_PTR pInitArgs | Must be NULL. |

**NOTE:** We do not support application level or OS level mutexes at this time.

## Output Parameters

None

## Return Values

CKR_ARGUMENTS_BAD

CKR_CANT_LOCK

CKR_CRYPTOKI_ALREADY_INITIALIZED

CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR

CKR_HOST_MEMORY

CKR_NEED_TO_CREATE_THREADS

CKR_OK

# C_Finalize

## Description

Called to indicate that an application is finished with using the pkcs11 library.

```
CK_DEFINE_FUNCTION(CK_RV, C_Finalize)(
   CK_VOID_PTR pReserved
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_VOID_PTR pReserved | Must be NULL. |

## Output Parameters

None.

## Return Values

CKR_ARGUMENTS_BAD

CKR_CRYPTOKI_NOT_INITIALIZED

CKR_FUNCTION_FAILED

CKR_GENERAL_ERROR

CKR_HOST_MEMORY

CKR_OK

# C_GetInfo

## Description

Provides manufacturer and version information about the library.

```
CK_DEFINE_FUNCTION(CK_RV, C_GetInfo)
  (CK_INFO_PTR pInfo);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_INFO_PTR pInfo | Pointer to a CK_INFO structure to receive the information. |

## Output Parameters

C_GetInfo will fill in the CK_INFO structure with relevant information about the library.

CK_INFO provides general information about Cryptoki. It is defined as follows:

| Parameter | Description/Value |
|---|---|
| cryptokiVersion | Cryptoki interface version number, for compatibility with future revisions of this interface. |
| manufacturerID | ID of the Cryptoki library manufacturer. Must be padded with the blank character (' '). Should ***not*** be null-terminated. |
| flags | Bit flags reserved for future versions. Must be zero for this version. |
| libraryDescription | Character-string description of the library. Must be padded with the blank character (' '). Should ***not*** be null-terminated. |
| libraryVersion | Cryptoki library version number. |

## Return Values

CKR_ARGUMENTS_BAD

CKR_CRYPTOKI_NOT_INITIALIZED

CKR_FUNCTION_FAILED

CKR_GENERAL_ERROR

CKR_HOST_MEMORY

CKR_OK

# C_GetFunctionList

## Description

Allows the application (caller of the Application Encryption Library) to find the list of APIs that are supported and their addresses.

```
CK_DEFINE_FUNCTION(CK_RV, C_GetFunctionList)(
   CK_FUNCTION_LIST_PTR_PTR ppFunctionList
);
```

**NOTE:** Only the applications written in C will call `C_GetFunctionList`. Java and .NET versions will not. For more examples, see the sample code.

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_FUNCTION_LIST_PTR_PTR ppFunctionList;` | Pointer to a value that will receive a pointer to `CK_FUNCTION_LIST` structure that contains function pointers for all the API routines in the library. |

## Output Parameters

`ppFunctionList` will be filled in with the address of the list of function pointers from the PKCS#11 library.

## Return Values

`CKR_ARGUMENTS_BAD`

`CKR_FUNCTION_FAILED`

`CKR_GENERAL_ERROR`

`CKR_HOST_MEMORY`

`CKR_OK`

# Slot and Token Management Functions

# C_GetSlotList

## Description

Provides list of available slots. It can be a single default slot. In C, call this function twice to get the actual slot list. The first time, it returns the number of available slots. Allocate memory to the available slots, and then the second call returns the actual slot list.

```
CK_DEFINE_FUNCTION(CK_RV, C_GetSlotList)(
   CK_BBOOL tokenPresent,
   CK_SLOT_ID_PTR pSlotList,
   CK_ULONG_PTR pulCount);
);
```

## Input Parameters

**NOTE:** Only the applications written in C will call `C_GetSlotList`. Java and .NET versions will not. For more examples, see the sample code.

First call to `C_GetSlotList(CK_FALSE)` to retrieve the number of slots.

| Parameter | Description/Value |
|---|---|
| `CK_BOOL tokenPresent` | Must be CK_FALSE. |
| `CK_SLOT_ID_PTR pSlotList` | Must be NULL_PTR. |
| `CK_ULONG_PTR pulCount` | Pointer to an unsigned long to hold slot count. |

Second call to `C_GetSlotList` to retrieve the actual slot list.

| Parameter | Description/Value |
|---|---|
| `CK_BOOL tokenPresent` | CK_TRUE |
| `CK_SLOT_ID_PTR` | Pointer to the space allocated to hold the slot based on the count retrieved above. |
| `CK_ULONG_PTR pulCount` | Pointer to a CK_ULONG holding the number of slots allocated above. |

## Output Parameters

`pulCount` points to the number of slots available. It is always 1 for our library.

## Return Values

`CKR_ARGUMENTS_BAD`

CKR_BUFFER_TOO_SMALL

CKR_CRYPTOKI_NOT_INITIALIZED

CKR_FUNCTION_FAILED

CKR_GENERAL_ERROR

CKR_HOST_MEMORY

CKR_OK

# C_GetSlotInfo

## Description

Provides information about a particular slot in the system.

```
CK_DEFINE_FUNCTION(CK_RV, C_GetSlotInfo)(
   CK_SLOT_ID slotID,
   CK_SLOT_INFO_PTR pInfo
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SLOT_ID slotID` | Must be 0. |
| `CK_SLOT_INFO_PTR pInfo` | Pointer to a CK_SLOT_INFO object. |

## Output Parameters

The structure where `CK_SLOT_INFO_PTR` points will be filled in with relevant information about the token.

## Return Values

`CKR_ARGUMENTS_BAD`

`CKR_CRYPTOKI_NOT_INITIALIZED`

`CKR_DEVICE_ERROR`

`CKR_FUNCTION_FAILED`

`CKR_GENERAL_ERROR`

`CKR_HOST_MEMORY`

`CKR_OK`

`CKR_SLOT_ID_INVALID`

# C_GetTokenInfo

## Description

Provides information about a specific token.

```
CK_DEFINE_FUNCTION(CK_RV, C_GetSlotInfo)(
   CK_SLOT_ID slotID,
   CK_SLOT_INFO_PTR pInfo );
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SLOT_ID slotID` | Must be slot 0. |
| `CK_TOKEN_INFO_PTR pInfo` | Pointer to a CK_TOKEN_INFO object. |

## Output Parameters

The structure where `CK_TOKEN_INFO_PTR` points will be filled in with relevant information about the token.

## Return Values

`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_DEVICE_ERROR`
`CKR_DEVICE_MEMORY`
`CKR_DEVICE_REMOVED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_OK`
`CKR_SLOT_ID_INVALID`
`CKR_TOKEN_NOT_PRESENT`
`CKR_TOKEN_NOT_RECOGNIZED`
`CKR_ARGUMENTS_BAD`

# C_GetMechanismList

## Description

Provides a list of mechanisms supported.

```
CK_DEFINE_FUNCTION(CK_RV,C_GetMechanismList)(
  CK_SLOT_ID slotID,
  CK_MECHANISM_TYPE_PTR pMechanismList,
  CK_ULONG_PTR pulCount
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SLOT_ID slotID` | Must be slot 0. |
| `CK_MECHANISM_TYPE_PTR pMechanismList` | Pointer to memory to hold a list of `CK_MECHANISM_TYPE`. |
| `CK_ULONG_PTR pulCount` | Number of `CK_MECHANISM_TYPE` objects the memory above can hold. |

## Output Parameters

The `CK_MECHANISM` structure where `CK_MECHANISM_TYPE_PTR` points will be filled in with the list of supported mechanism types.

## Return Values

`CKR_BUFFER_TOO_SMALL`
`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_DEVICE_ERROR`
`CKR_DEVICE_MEMORY`
`CKR_DEVICE_REMOVED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_OK, CKR_SLOT_ID_INVALID`
`CKR_TOKEN_NOT_PRESENT`
`CKR_TOKEN_NOT_RECOGNIZED`
`CKR_ARGUMENTS_BAD`

# C_GetMechanismInfo

## Description

Provides information about a mechanism possibly supported by the token.

```
CK_DEFINE_FUNCTION(CK_RV, C_GetMechanismInfo)(
  CK_SLOT_ID slotID,
  CK_MECHANISM_TYPE type,
  CK_MECHANISM_INFO_PTR pInfo
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SLOT_ID slotID` | Must be 0. |
| `CK_MECHANISM_TYPE type` | Mechanism type to retrieve info for. |
| `CK_MECHANISM_INFO_PTR pInfo` | Pointer to a `CK_MECHANISM_INFO` structure. |

## Output Parameters

The `CK_MECHANISM_INFO` structure where `pInfo` points is filled in with information about a particular mechanism.

## Return Values

`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_DEVICE_ERROR`
`CKR_DEVICE_MEMORY`
`CKR_DEVICE_REMOVED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_MECHANISM_INVALID`
`CKR_OK`
`CKR_SLOT_ID_INVALID`
`CKR_TOKEN_NOT_PRESENT`
`CKR_TOKEN_NOT_RECOGNIZED`
`CKR_ARGUMENTS_BAD`

# Session Management Functions

# C_OpenSession

## Description

Starts a cryptographic session with a specific token.

```
CK_DEFINE_FUNCTION(CK_RV, C_OpenSession)(
  CK_SLOT_ID slotID,
  CK_FLAGS flags, CK_VOID_PTR pApplication,
  CK_NOTIFY Notify,
  CK_SESSION_HANDLE_PTR phSession
);
```

## Supported Functionality

Only read/write sessions are supported. Maximum number of read/write sessions is 1000.

No read-only sessions are supported.

Each thread must run in its own session for multi-threaded applications.

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SLOT_ID slotID` | Must be 0. |
| `CK_FLAGS flags` | Must be 0. |
| `CK_VOID_PTR pApplication` | String to describe session name, or NULL. |
| `CK_NOTIFY Notify` | NULL |
| `CK_SESSION_HANDLE_PTR phSession` | Pointer to a session ID. |

## Output Parameters

The session ID pointer (`CK_SESSION_HANDLE_PTR phSession`) will be filled in with the session handle.

## Return Values

CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED

CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY

# C_CloseSession

## Description

Closes a cryptographic session.

```
CK_DEFINE_FUNCTION(CK_RV, C_CloseSession)(
  CK_SESSION_HANDLE hSession
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session to close, identified by the session handle. |

## Output Parameters

None.

## Return Values

CKR_CRYPTOKI_NOT_INITIALIZED

CKR_DEVICE_ERROR

CKR_DEVICE_MEMORY

CKR_DEVICE_REMOVED

CKR_FUNCTION_FAILED

CKR_GENERAL_ERROR

CKR_HOST_MEMORY

CKR_OK, CKR_SESSION_CLOSED

CKR_SESSION_HANDLE_INVALID

# C_CloseAllSessions

## Description

Closes all sessions on a slot. C programming language only.

```
CK_DEFINE_FUNCTION(CK_RV, C_CloseAllSessions)(
  CK_SLOT_ID slotID
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SLOT_ID slotID` | Must be 0. |

## Output Parameters

None.

## Return Values

`CKR_CRYPTOKI_NOT_INITIALIZED`

`CKR_DEVICE_ERROR`

`CKR_DEVICE_MEMORY`

`CKR_DEVICE_REMOVED`

`CKR_FUNCTION_FAILED`

`CKR_GENERAL_ERROR`

`CKR_HOST_MEMORY, CKR_OK`

`CKR_SESSION_CLOSED`

`CKR_SESSION_HANDLE_INVALID`

`CKR_ARGUMENTS_BAD`

# C_GetSessionInfo

## Description

Provides information about a session.

```
CK_DEFINE_FUNCTION(CK_RV, C_GetSessionInfo)(
  CK_SESSION_HANDLE hSession,
  CK_SESSION_INFO_PTR pInfo
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION _HANDLE hSession` | Session handle. |
| `CK_SESSION_INFO_PTR pInfo` | Pointer to a `CK_SESSION_INFO` object. |

## Output Parameters

The structure where `CK_SESSION_INFO_PTR` points will be filled in with relevant information about the session.

## Return Values

`CKR_CRYPTOKI_NOT_INITIALIZED`

`CKR_DEVICE_ERROR`

`CKR_DEVICE_MEMORY`

`CKR_DEVICE_REMOVED`

`CKR_FUNCTION_FAILED`

`CKR_GENERAL_ERROR`

`CKR_HOST_MEMORY`

`CKR_OK`

`CKR_SESSION_CLOSED`

`CKR_SESSION_HANDLE_INVALID`

`CKR_ARGUMENTS_BAD`

# C_Login

## Description

Logs a user into a token. The PIN is the PIN entered during Key Agent registration.

```
CK_DEFINE_FUNCTION(CK_RV, C_Login)(
  CK_SESSION_HANDLE hSession,
  CK_USER_TYPE userType,
  CK_UTF8CHAR_PTR pPin,
  CK_ULONG ulPinLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session to log the user into. |
| CK_USER_TYPE userType | CKU_USER (We support only CKU_USER) |
| CK_CHAR_PTR pPin | Password used to register the key agent with the Data Security Manager. |
| CK_ULONG ulPinLen | Length of the PIN. |

## Output Parameters

None.

## Return Values

CKR_ARGUMENTS_BAD
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_OK
CKR_OPERATION_NOT_INITIALIZED
CKR_PIN_INCORRECT
CKR_PIN_LOCKED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY_EXISTS
CKR_USER_ALREADY_LOGGED_IN
CKR_USER_ANOTHER_ALREADY_LOGGED_IN
CKR_USER_PIN_NOT_INITIALIZED
CKR_USER_TOO_MANY_TYPES
CKR_USER_TYPE_INVALID

# C_Logout

## Description

Logs a user out from a token. You can call `C_logout()` multiple times in a row.

```
CK_DEFINE_FUNCTION(CK_RV, C_Logout)(
  CK_SESSION_HANDLE hSession
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle that you want to log out of. |

## Output Parameters

None.

## Return Values

`CKR_CRYPTOKI_NOT_INITIALIZED`

`CKR_DEVICE_ERROR`

`CKR_DEVICE_MEMORY`

`CKR_DEVICE_REMOVED`

`CKR_FUNCTION_FAILED`

`CKR_GENERAL_ERROR`

`CKR_HOST_MEMORY`

`CKR_OK`

`CKR_SESSION_CLOSED`

`CKR_SESSION_HANDLE_INVALID`

# Object Management Functions

# C_WrapKey

## Description

Supports key export on the GDE Appliance where a symmetric key is wrapped with another symmetric key on the GDE Appliance and then exported. This function supports metadata logging.

```
CK_DEFINE_FUNCTION(CK_RV, C_WrapKey)(
  CK_SESSION_HANDLE hSession,
  CK_MECHANISM_PTR pMechanism,
  CK_OBJECT_HANDLE hWrappingKey,
  CK_OBJECT_HANDLE hKey,
  CK_BYTE_PTR pWrappedKey,
  CK_ULONG_PTR pulWrappedKeyLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_MECHANISM_PTR pMechanism` | The mechanism pointer only supports CKM_AES_CBC_PAD mode. It requires a 16 byte i.v. The structure looks like this: `{ CKM_AES_CBC_PAD,iv,16};` |
| `CK_OBJECT_HANDLE hWrappingKey` | Symmetric key used to wrap the key. |
| `CK_OBJECT_HANDLE hKey` | Symmetric key to export. |
| `CK_BYTE_PTR pWrappedKey` | Points to an array to put the wrapped key bytes. |
| `CK_ULONG_PTR ulWrappedKeyLen` | Length of the above array. |

## Output Parameters

The array where `CK_BYTE_PTR pWrappedKey` points will be filled in with the wrapped key bytes.

## Return Values

| | |
|---|---|
| CKR_ARGUMENTS_BAD | CKR_KEY_SIZE_RANGE |
| CKR_BUFFER_TOO_SMALL | CKR_KEY_UNEXTRACTABLE |
| CKR_CRYPTOKI_NOT_INITIALIZED | CKR_MECHANISM_INVALID |
| CKR_DEVICE_ERROR | CKR_MECHANISM_PARAM_INVALID |
| CKR_DEVICE_MEMORY | CKR_OK, CKR_OPERATION_ACTIVE |
| CKR_DEVICE_REMOVED | CKR_PIN_EXPIRED |
| CKR_FUNCTION_CANCELED | CKR_SESSION_CLOSED |
| CKR_FUNCTION_FAILED | CKR_SESSION_HANDLE_INVALID |
| CKR_GENERAL_ERROR | CKR_USER_NOT_LOGGED_IN |
| CKR_HOST_MEMORY | CKR_WRAPPING_KEY_HANDLE_INVALID |
| CKR_KEY_HANDLE_INVALID | CKR_WRAPPING_KEY_SIZE_RANGE |
| CKR_KEY_NOT_WRAPPABLE | CKR_WRAPPING_KEY_TYPE_INCONSISTENT |

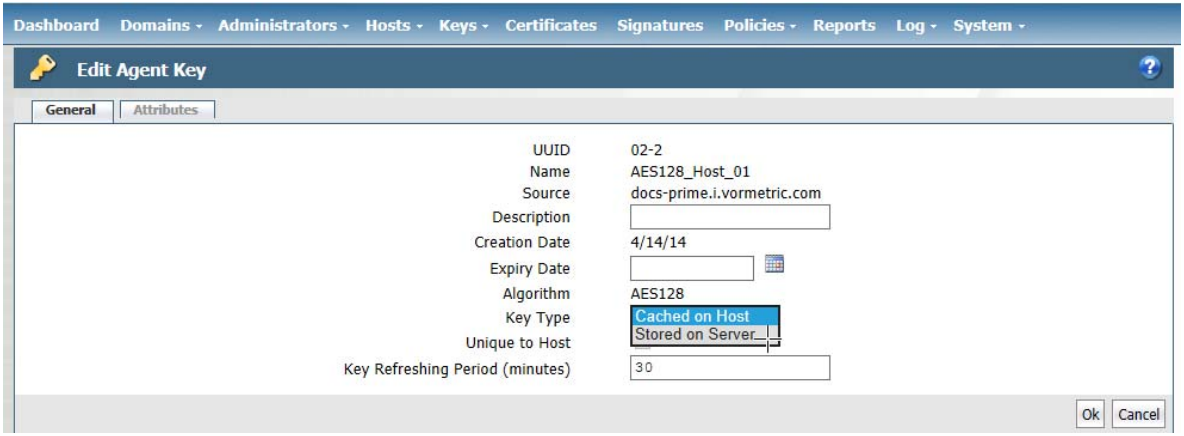
# C_CreateObject

## Description

Imports key bytes into the GDE Appliance and creates a symmetric key on the GDE Appliance. Maximum length of the key name is 64 characters. There is no minimum length.

```
CK_DEFINE_FUNCTION(CK_RV, C_CreateObject)(
  CK_SESSION_HANDLE hSession,
  CK_ATTRIBUTE_PTR pTemplate,
  CK_ULONG ulCount,
  CK_OBJECT_HANDLE_PTR phObject
);
```

**Figure 3:** Edit Agent Key window




By default, keys are stored on the host. If you don't want them stored on the host, log on to the GDE Appliance Console, switch to the domain with the desired host, click the **Keys** tab and select the key name from the **Name** list. In the **Key Type** drop down menu in the **Edit Agent Key** window, select **Stored on Server.**

### Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_ATTRIBUTE pTemplate` | Same template as `C_GenerateKey`, with the addition of `CKA_VALUE` and `CKA_VALUE_LEN`, which hold the key bytes and the length of the key bytes respectively. |
| `CK_ULONG ulCount` | Size of `aesKeyTemplate`. |
| `CK_OBJECT_HANDLE phObject` | Generated key handle. |

| | | |
|---|---|---|
| `CKA_LABEL` | `CK_UTF8CHAR` | |
| `CKA_APPLICATION` | `CK_UTF8CHAR` | |
| `CKA_CLASS` | `CK_OBJECT_CLASS` | |
| `CKA_KEY_TYPE` | `CK_KEY_TYPE` | |
| `CKA_VALUE` | `CK_BYTE` | |
| `CKA_VALUE_LEN` | `CK_ULONG` | |
| `CKA_TOKEN` | `CK_BBOOL` | |
| `CKA_ENCRYPT` | `CK_BBOOL` | |
| `CKA_DECRYPT` | `CK_BBOOL` | |
| `CKA_SIGN` | `CK_BBOOL` | |
| `CKA_VERIFY` | `CK_BBOOL` | |
| `CKA_WRAP` | `CK_BBOOL` | |
| `CKA_UNWRAP` | `CK_BBOOL` | |
| `CKA_EXTRACTABLE` | `CK_BOOL` | |
| `CKA_ALWAYS_SENSITIVE` | `CK_BBOOL` | |
| `CKA_NEVER_EXTRACTABLE` | `CK_BBOOL` | |
| `CKA_SENSITIVE` | `CK_BBOOL` | |

### Output Parameters

`CK_OBJECT_HANDLE hGenKey` will be filled in with the handle of the newly generated key.

### Return Values

| | |
|---|---|
| CKR_ARGUMENTS_BAD | CKR_GENERAL_ERROR |
| CKR_ATTRIBUTE_READ_ONLY | CKR_HOST_MEMORY |
| CKR_ATTRIBUTE_TYPE_INVALID | CKR_OK, CKR_PIN_EXPIRED |
| CKR_ATTRIBUTE_VALUE_INVALID | CKR_SESSION_CLOSED |
| CKR_CRYPTOKI_NOT_INITIALIZED | CKR_SESSION_HANDLE_INVALID |
| CKR_DEVICE_ERROR | CKR_SESSION_READ_ONLY |
| CKR_DEVICE_MEMORY | CKR_TEMPLATE_INCOMPLETE |
| CKR_DEVICE_REMOVED | CKR_TEMPLATE_INCONSISTENT |
| CKR_DOMAIN_PARAMS_INVALID | CKR_TOKEN_WRITE_PROTECTED |
| CKR_FUNCTION_FAILED | CKR_USER_NOT_LOGGED_IN |

# C_DestroyObject

## Description

Deletes a key.

Warning! This function should be used VERY CAREFULLY. Once a key is deleted, there is no way to recover the key and any data encrypted by that key will be lost.

```
CK_DEFINE_FUNCTION(CK_RV, C_DestroyObject)(
  CK_SESSION_HANDLE hSession,
  CK_OBJECT_HANDLE hObject
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_OBJECT_HANDLE hObject` | Handle of the key to be deleted. |

## Output Parameters

None.

## Return Values

`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_DEVICE_ERROR`
`CKR_DEVICE_MEMORY`
`CKR_DEVICE_REMOVED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_OBJECT_HANDLE_INVALID`
`CKR_OK, CKR_PIN_EXPIRED`
`CKR_SESSION_CLOSED`
`CKR_SESSION_HANDLE_INVALID`
`CKR_SESSION_READ_ONLY`
`CKR_TOKEN_WRITE_PROTECTED`

# C_FindObjectsInit

## Description

Initializes a search for a token and session objects that match a template. We only support searching for a key by the CKA_LABEL, which corresponds to the keyname on the GDE Appliance. The search template must have CKA_LABEL as its required attribute.

```
CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsInit)(
  CK_SESSION_HANDLE hSession,
  CK_ATTRIBUTE_PTR pTemplate,
  CK_ULONG ulCount
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session handle. |
| CK_ATTRIBUTE_PTR pTemplate | Attribute(s) to search for. |
| CK_ULONG ulCount | Attribute template count. |

## Output Parameters

None.

## Return Values

CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_OK
CKR_OPERATION_ACTIVE
CKR_PIN_EXPIRED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID

# C_FindObjects

## Description

Finds an object on the GDE Appliance. The only attribute we support is KEY_LABEL, which corresponds to the key name on the GDE Appliance. We return at most 1 key.

```
CK_DEFINE_FUNCTION(CK_RV, C_FindObjects)(
     CK_SESSION_HANDLE hSession,
     CK_OBJECT_HANDLE_PTR phObject,
     CK_ULONG ulMaxObjectCount,
     CK_ULONG_PTR pulObjectCount
);
```

**NOTE:** The GDE Appliance honors only the public key name. To query for a public key, pass in the public key name and CKO_PUBLIC_KEY. To query for the private key, pass in the public key name and CKO_PRIVATE_KEY.

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session handle. |
| CK_OBJECT_HANDLE_PTR phObject | Pointer to the buffer to put the object handles. |
| CK_ULONG ulMaxObjectCount | The maximum number of objects to put into the buffer. |
| CK_ULONG_PTR pulObjectCount | Pointer to CK_ULONG where the number of objects found is returned. |

## Output Parameters

| Parameter | Description/Value |
|---|---|
| CK_OBJECT_HANDLE_PTR phObject | Returns a single key that matches the CKA_LABEL. |
| CK_ULONG_PTR pulObjectCount | This is always 1 the first time the API finds a match. Subsequent calls return a uplObjectCount of 0, indicating no other objects matched that key name. |

### Return Values

| | |
|---|---|
| CKR_ARGUMENTS_BAD | CKR_GENERAL_ERROR |
| CKR_CRYPTOKI_NOT_INITIALIZED | CKR_HOST_MEMORY |
| CKR_DEVICE_ERROR | CKR_OK |
| CKR_DEVICE_MEMORY | CKR_OPERATION_NOT_INITIALIZED |
| CKR_DEVICE_REMOVED | CKR_SESSION_CLOSED |
| CKR_FUNCTION_FAILED | CKR_SESSION_HANDLE_INVALID |

# C_FindObjectsFinal

## Description

Terminates a search for a token and session objects.

```
CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsFinal)(
  CK_SESSION_HANDLE hSession
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session handle. |

## Output Parameters

None.

## Return Values

CKR_CRYPTOKI_NOT_INITIALIZED

CKR_DEVICE_ERROR

CKR_DEVICE_MEMORY

CKR_DEVICE_REMOVED

CKR_FUNCTION_FAILED

CKR_GENERAL_ERROR

CKR_HOST_MEMORY

CKR_OK

CKR_OPERATION_NOT_INITIALIZED

CKR_SESSION_CLOSED

CKR_SESSION_HANDLE_INVALID

# C_GetAttributeValue

## Description

Gets attribute value of a specific Security Object. Argument specifies the Security Object ID.

```
K_RV C_GetAttributeValue ( CK_SESSION_HANDLE   hSession,
                     CK_OBJECT_HANDLE hObject,
                     CK_ATTRIBUTE_PTR pTemplate,
                     CK_ULONG ulCount
                     ) ;
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_OBJECT_HANDLE hObject` | Object to get the attribute for. |
| `CK_ATTRIBUTE_PTR pTemplate` | Template containing the attributes to search for. Points to the CK_ATTRIBUTE structure. |
| `CK_ULONG ulCount` | Number of attributes in the template. |

## Output Parameters

`CK_ATTRIBUTE_PTR pTemplate` will be filled in with the values for the attributes searched for.

**NOTE:** Due to a known issue, on AIX, this function always returns CKA_END_DATE as 01/01/1970.

## Return Values

`CKR_ARGUMENTS_BAD`
`CKR_ATTRIBUTE_SENSITIVE`
`CKR_ATTRIBUTE_TYPE_INVALID`
`CKR_BUFFER_TOO_SMALL`
`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_DEVICE_ERROR`
`CKR_DEVICE_MEMORY`
`CKR_DEVICE_REMOVED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_OBJECT_HANDLE_INVALID`
`CKR_OK`
`CKR_SESSION_CLOSED`
`CKR_SESSION_HANDLE_INVALID`

# C_SetAttributeValue

## Description

Sets attribute value of a specific Security Object. Argument specifies the Security Object ID.

```
CK_DEFINE_FUNCTION(CK_RV, C_SetAttributeValue)(
     CK_SESSION_HANDLE hSession,
     CK_OBJECT_HANDLE hObject,
     CK_ATTRIBUTE_PTR pTemplate,
     CK_ULONG ulCount
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session handle. |
| CK_OBJECT_HANDLE hObject | Object handle. |
| CK_ATTRIBUTE_PTR pTemplate | Attribute template for the attributes to set. |
| CK_ULONG ulCount | Number of attributes in the template. |

## Output Parameters

None

## Return Values

CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_OBJECT_HANDLE_INVALID
CKR_OK, CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_WRITE_PROTECTED
CKR_USER_NOT_LOGGED_IN

# C_GenerateKey

## Description

Generates a symmetric key. Maximum length of the key name is 64 characters. There is no minimum length.

By default, keys are stored on the host or on the GDE Appliance. If you don't want them stored on the host, log on to the GDE Appliance Console, click the **Keys** tab and select the key name from the **Name** list. From the **Edit Agent Keys** window, in the **Key Type** drop-down menu, select **Stored on Server.**

```
CK_DEFINE_FUNCTION(CK_RV, C_GenerateKey)(
     CK_SESSION_HANDLE   hSession,
     CK_MECHANISM_PTR pMechanism,
     CK_ATTRIBUTE_PTR pTemplate,
     CK_ULONG ulCount,
     CK_OBJECT_HANDLE_PTR phKey
);
```

## Supported functionality

AES-128 and AES-256 symmetric encryption keys are supported.

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_MECHANISM_PTR pMechanism` | Key generation mechanism; must have the following values:<br>`{CKM_AES_KEY_GEN, NULL_PTR, 0};` |
| `CK_ULONG ulCount` | The number of attributes in the template. |
| `CK_OBJECT_HANDLE_PTR phKey` | Pointer to a key handle. |
| `CK_ATTRIBUTE_PTR pTemplate` | See below: |

| Attribute name | Type | Value |
|---|---|---|
| `CKA_LABEL` | `CK_UTF8CHAR` | Key name; also displayed on the Data Security Manager |
| `CKA_APPLICATION` | `CK_UTF8CHAR` | Description of the application generating the key name, can be NULL. |
| `CKA_CLASS` | `CK_OBJECT_CLASS` | Must be `CKO_SECRET_KEY`. |
| `CKA_KEY_TYPE` | `CK_KEY_TYPE` | Must be `CKK_AES`. |

| | | |
|---|---|---|
| CKA_TOKEN | CK_BBOOL | Must be false. |
| CKA_ENCRYPT | CK_BBOOL | Must be true. |
| CKA_DECRYPT | CK_BBOOL | Must be true. |
| CKA_SIGN | CK_BBOOL | Must be false. |
| CKA_VERIFY | CK_BBOOL | Must be false. |
| CKA_WRAP | CK_BBOOL | Must be true. |
| CKA_UNWRAP | CK_BBOOL | Must be false. |
| CKA_EXTRACTABLE | CK_BBOOL | Must be false. |
| CKA_ALWAYS_SENSITIVE | CK_BBOOL | Must be false. |
| CKA_NEVER_EXTRACTABLE | CK_BBOOL | Must be true. |
| CKA_SENSITIVE | CK_BBOOL | Must be true. |

### Output Parameters

The structure where `CK_OBJECT_HANDLE_PTR` phKey points is filled in with the newly created key handle.

### Return Values

`CKR_ARGUMENTS_BAD`
`CKR_ATTRIBUTE_READ_ONLY`
`CKR_ATTRIBUTE_TYPE_INVALID`
`CKR_ATTRIBUTE_VALUE_INVALID`
`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_DEVICE_ERROR`
`CKR_DEVICE_MEMORY`
`CKR_DEVICE_REMOVED`
`CKR_FUNCTION_CANCELED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_MECHANISM_INVALID`
`CKR_MECHANISM_PARAM_INVALID`
`CKR_OK`
`CKR_OPERATION_ACTIVE`
`CKR_PIN_EXPIRED`
`CKR_SESSION_CLOSED`
`CKR_SESSION_HANDLE_INVALID`
`CKR_SESSION_READ_ONLY`
`CKR_TEMPLATE_INCOMPLETE`
`CKR_TEMPLATE_INCONSISTENT`
`CKR_TOKEN_WRITE_PROTECTED`
`CKR_USER_NOT_LOGGED_IN`

# C_GenerateKeyPair

## Description

Generates an asymmetric key pair. Arguments specify RSA type and length of the key pair.

```
CK_DEFINE_FUNCTION(CK_RV, C_GenerateKeyPair)(
     CK_SESSION_HANDLE hSession,
     CK_MECHANISM_PTR pMechanism,
     CK_ATTRIBUTE_PTR pPublicKeyTemplate,
     CK_ULONG ulPublicKeyAttributeCount,
     CK_ATTRIBUTE_PTR pPrivateKeyTemplate,
     CK_ULONG ulPrivateKeyAttributeCount,
     CK_OBJECT_HANDLE_PTR phPublicKey,
     CK_OBJECT_HANDLE_PTR phPrivateKey
);
```

## Supported functionality

RSA-1024 and RSA-2048 asymmetric key pairs are supported.

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_MECHANISM_PTR pMechanism` | Key generation mechanism must have the following values:<br>`{ CKM_RSA_PKCS_KEY_PAIR_GEN, NULL_PTR, 0 };` |
| `CK_ATTRIBUTE_PTR pPublicKeyTemplate` | `{CKA_LABEL, publicKeyLabel, sizeof(publicKeyLabel) },/* Keyname */`<br>`{CKA_CLASS, &pubkey_class, sizeof(pubkey_class)},`<br>`{CKA_ENCRYPT, &bTrue, sizeof(bTrue)},`<br>`{CKA_VERIFY, &bTrue, sizeof(bTrue)},`<br>`{CKA_WRAP, &bTrue, sizeof(bTrue)},`<br>`{CKA_TOKEN, &bTrue, sizeof(bTrue)},`<br>`{CKA_PUBLIC_EXPONENT, publicExponent, sizeof(publicExponent)},`<br>`{CKA_MODULUS_BITS, &modulusBits, sizeof(modulusBits)}` |

| Parameter | Description/Value |
|---|---|
| CK_ULONG ulPublicKeyAttributeCount | Size of the key template `publicKeyTemplate`. |
| CK_ATTRIBUTE_PTR privateKeyTemplate | {CKA_LABEL, privateKeyLabel, sizeof(publicKeyLabel) },/* Keyname*/<br>{CKA_CLASS, &privkey_class, sizeof(privkey_class)},<br>{CKA_TOKEN, &bTrue, sizeof(bTrue)},<br>{CKA_PRIVATE, &bTrue, sizeof(bTrue)},<br>{CKA_SENSITIVE, &bTrue, sizeof(bTrue)},<br>{CKA_DECRYPT, &bTrue, sizeof(bTrue)},<br>{CKA_SIGN, &bTrue, sizeof(bTrue)},<br>{CKA_UNWRAP, &bTrue, sizeof(bTrue)} |
| CK_ULONG ulPrivateKeyAttributeCount | Size of the key template `privateKeyTemplate`. |
| CK_OBJECT_HANDLE_PTR phPublicKey | Filled in with the newly created public key handle. |
| CK_OBJECT_HANDLE_PTR phPrivateKey | Filled in with the newly created private key handle. |

## Output Parameters

`CK_OBJECT_HANDLE_PTR phPublicKey` is filled in with the object handle of the public key.

`CK_OBJECT_HANDLE_PTR phPublicKey` is filled in with the object handle of the private key.

## Return Values

`CKR_ARGUMENTS_BAD`
`CKR_ATTRIBUTE_READ_ONLY`
`CKR_ATTRIBUTE_TYPE_INVALID`
`CKR_ATTRIBUTE_VALUE_INVALID`
`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_DEVICE_ERROR`
`CKR_DEVICE_MEMORY`
`CKR_DEVICE_REMOVED`
`CKR_DOMAIN_PARAMS_INVALID`
`CKR_FUNCTION_CANCELED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_MECHANISM_INVALID`
`CKR_MECHANISM_PARAM_INVALID`
`CKR_OK`
`CKR_OPERATION_ACTIVE`
`CKR_PIN_EXPIRED`
`CKR_SESSION_CLOSED`
`CKR_SESSION_HANDLE_INVALID`
`CKR_SESSION_READ_ONLY`
`CKR_TEMPLATE_INCOMPLETE`
`CKR_TEMPLATE_INCONSISTENT`
`CKR_TOKEN_WRITE_PROTECTED`
`CKR_USER_NOT_LOGGED_IN`

# Digest and MACing Functions

# C_DigestInit

## Description

Initializes a digest operation.

```
CK_DEFINE_FUNCTION(CK_RV, C_DigestInit)(
   CK_SESSION_HANDLE hSession,
   CK_MECHANISM_PTR pMechanism,
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session handle. |
| CK_MECHANISM_PTR pMechanism | Which cryptographic hash function to use. One of the following values:<br>{ CKM_SHA256, NULL_PTR, 0 },<br>{ CKM_SHA384, NULL_PTR, 0 },<br>{ CKM_SHA512, NULL_PTR, 0 }<br>OR<br>{ CKM_SHA256_HMAC, NULL_PTR, 0 } |

## Output Parameters

None.

## Return Values

CKR_ARGUMENTS_BAD
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OK
CKR_SESSION_HANDLE_INVALID

# C_Digest

## Description

Compute data digest in a single part. Must call C_DigestInit before calling C_Digest.

```
CK_DEFINE_FUNCTION(CK_RV, C_DigestInit)(
   CK_SESSION_HANDLE hSession,
   CK_BYTE_PTR pPart,
   CK_ULONG ulPartLen,
    CK_BYTE_PTR pDigest,
    CK_ULONG_PTR pulDigestLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session handle. |
| CK_BYTE_PTR pPart | Pointer to the digest input data. |
| CK_ULONG ulPartLen | Length of the data. |
| CK_BYTE_PTR pDigest | Results in the message digest. |
| CK_ULONG_PTR pulDigestLen | The byte count of the digest. |

## Output Parameters

None.

## Return Values

CKR_ARGUMENTS_BAD
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OK
CKR_OPERATION_NOT_INITIALIZED
CKR_SESSION_HANDLE_INVALID

# C_DigestKey

## Description

Specifies HMAC key for digest operation. Must call C_DigestInit before calling C_DigestKey.

```
CK_DEFINE_FUNCTION(CK_RV, C_DigestKey)(
   CK_SESSION_HANDLE hSession,
   CK_OBJECT_HANDLE hKey
);
```

**NOTE:** This function does not work when key is set to `Store on Server`. The key must be set to `Cached on Host`.

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_OBJECT_HANDLE hKey` | Handle to the 256-bit private key. |

## Output Parameters

None.

## Return Values

`CKR_ARGUMENTS_BAD`
`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_FUNCTION_CANCELED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_KEY_FUNCTION_NOT_PERMITTED`
`CKR_KEY_HANDLE_INVALID`
`CKR_KEY_INDIGESTIBLE`
`CKR_KEY_SIZE_RANGE`
`CKR_MECHANISM_PARAM_INVALID`
`CKR_OK`
`CKR_SESSION_HANDLE_INVALID`

# C_DigestUpdate

## Description

Continues a multi-part digest operation. Must call C_DigestInit before calling C_DigestUpdate. Can optionally call C_DigestKey before calling C_DigestUpdate.

```
CK_DEFINE_FUNCTION(CK_RV, C_DigestUpdate)(
  CK_SESSION_HANDLE hSession,
  CK_BYTE_PTR       pPart,
  CK_ULONG          ulPartLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_BYTE_PTR pPart` | Pointer to the digest input data. |
| `CK_ULONG ulPartLen` | Length of the data. |

## Output Parameters

None.

## Return Values

`CKR_ARGUMENTS_BAD`
`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_FUNCTION_CANCELED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_KEY_FUNCTION_NOT_PERMITTED`
`CKR_KEY_HANDLE_INVALID`
`CKR_KEY_SIZE_RANGE`
`CKR_KEY_TYPE_INCONSISTENT`
`CKR_OK`
`CKR_OPERATION_NOT_INITIALIZED`
`CKR_SESSION_HANDLE_INVALID`

# C_DigestFinal

## Description

Finishes a multi-part digest operation. This works only with the local key cache. Not supported if crypto is done remotely on the GDE Appliance.

```
CK_DEFINE_FUNCTION(CK_RV, C_DigestFinal)(
   CK_SESSION_HANDLE hSession,
   CK_BYTE_PTR pDigest,
   CK_ULONG_PTR pulDigestLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session handle. |
| CK_BYTE_PTR pDigest | Results in the message digest. |
| CK_ULONG_PTR pulDigestLen | The byte count of the digest. |

## Output Parameters

None.

## Return Values

CKR_ARGUMENTS_BAD
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_KEY_HANDLE_INVALID
CKR_KEY_SIZE_RANGE
CKR_KEY_TYPE_INCONSISTENT
CKR_OK
CKR_OPERATION_NOT_INITIALIZED
CKR_SESSION_HANDLE_INVALID

# Signing and MACing Functions

# C_SignInit

## Description

Initializes a signature operation.

```
CK_DEFINE_FUNCTION(CK_RV, C_SignInit)(
   CK_SESSION_HANDLE hSession,
   CK_MECHANISM_PTR pMechanism,
   CK_OBJECT_HANDLE hKey
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_MECHANISM_PTR pMechanism` | The mechanism should have the following parameters `;{ CKM_RSA_PKCS, NULL_PTR, 0 }`. |
| `CK_OBJECT_HANDLE hKey` | Handle to the RSA private key. |

## Output Parameters

None.

## Return Values

`CKR_ARGUMENTS_BAD`
`CKR_CRYPTOKI_NOT_INITIALIZED`
`CKR_DEVICE_ERROR`
`CKR_DEVICE_MEMORY`
`CKR_DEVICE_REMOVED`
`CKR_FUNCTION_CANCELED`
`CKR_FUNCTION_FAILED`
`CKR_GENERAL_ERROR`
`CKR_HOST_MEMORY`
`CKR_KEY_FUNCTION_NOT_PERMITTED`
`CKR_KEY_HANDLE_INVALID`
`CKR_KEY_SIZE_RANGE`
`CKR_KEY_TYPE_INCONSISTENT`
`CKR_MECHANISM_INVALID`
`CKR_MECHANISM_PARAM_INVALID`
`CKR_OK`
`CKR_OPERATION_ACTIVE`
`CKR_PIN_EXPIRED`
`CKR_SESSION_CLOSED`
`CKR_SESSION_HANDLE_INVALID`
`CKR_USER_NOT_LOGGED_IN`

# C_Sign

## Description

Signs data in a single part.

```
CK_DEFINE_FUNCTION(CK_RV, C_Sign)(
  CK_SESSION_HANDLE hSession,
  CK_BYTE_PTR       pData,
  CK_ULONG          ulDataLen,
  CK_BYTE_PTR        pSignature,
  CK_ULONG_PTR      pulSignatureLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| CK_SESSION_HANDLE hSession | Session handle. |
| CK_BYTE_PTR pData | Points to the data. |
| CK_ULONG ulDataLen | Length of data. |
| CK_BYTE_PTR pSignature | Pointer to buffer to receive the signed data. |
| CK_ULONG_PTR pulSignatureLen | Pointer to buffer length. |

## Output Parameters

The CK_BYTE structure where pSignature points is filled in with the signed data.

The CK_ULONG structure where pulSignatureLen points is filled in with the signed data length.

## Return Values

CKR_ARGUMENTS_BAD
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DATA_INVALID
CKR_DATA_LEN_RANGE
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_OK
CKR_OPERATION_NOT_INITIALIZED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_USER_NOT_LOGGED_IN
CKR_FUNCTION_REJECTED

# C_Verify_Init

## Description

Initializes a verification operation.

```
CK_DEFINE_FUNCTION(CK_RV, C_VerifyInit)
  (CK_SESSION_HANDLE hSession,
  CK_MECHANISM_PTR pMechanism,
  CK_OBJECT_HANDLE hKey);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| hSession | Session handle. |
| pMechanism | Pointer to the structure that specifies the verification mechanism. |
| hKey | Handle for the public key of the key pair used for verification. |

## Output Parameters

None.

## Return Values

CKR_CRYPTOKI_NOT_INITIALIZED

CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR

CKR_KEY_FUCNTION_NOT_PERMITTED, CKR_KEY_HANDLE_INVALID

CKR_MECHANISM_INVALID

CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID

CKR_USER_NOT_LOGGED_IN

# C_Verify

## Description

Performs a single-part verification operation. To perform verification, the signature is verified with the key, the handle of which is provided in `C_Verify_Init` and compared with the `pData`.

```
CK_DEFINE_FUNCTION(CK_RV, C_Verify)
  (CK_SESSION_HANDLE hSession,
  CK_BYTE_PTR pData,
  CK_ULONG ulDataLen,
  CK_BYTE_PTR pSignature,
  CK_ULONG ulSignatureLen);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `hSession` | Session handle. |
| `pData` | Pointer to the data. |
| `ulDataLen` | Length of the data. |
| `pSignature` | Ponter to the signature. |
| `ulSignatureLen` | Length of the signature. |

## Output Parameters

`None.`

## Return Values

`CKR_OK`

`CKR_SIGNATURE_INVALID`

## Error Codes

`CKR_CRYPTOKI_NOT_INITIALIZED`

`CKR_SIGNATURE_LEN_RANGE`

# Encryption Functions

# C_EncryptInit

## Description

Initializes an encryption operation. `C_EncryptInit` must be called before `C_Encrypt` or `CEncryptUpdate` is called.

```
CK_DEFINE_FUNCTION(CK_RV, C_EncryptInit)(
  CK_SESSION_HANDLE hSession,
  CK_MECHANISM_PTR  pMechanism,
  CK_OBJECT_HANDLE  hKey
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_MECHANISM_PTR pMechanism` | The pointer to the structure that specifies the encryption mechanism.<br><br>We support the following mechanisms:<br>`CKM_AES_ECB`<br>`CKM_AES_CBC`<br>`CKM_AES_CBC_PAD`<br>`CKM_AES_CTR`<br>`CKM_VORMETRIC_FPE` (in Java, use: `0x80004001L`)<br>**Note:** For `CKM_VORMETRIC_FPE`, the IV contains the following 3 field values:<br>- tweak (8 bytes)<br>- character set. ASCII or Unicode characters. The following Unicode encodings are supported: UTF-8, UTF-16, UTF-32, big-endian or little-endian byte order. Be sure there are no duplicate characters in the character set.<br>- radix (character set size ).The size can be from 2 to 65535. For more information, see samples.<br>**Note:** FPE and CTR do not work when key is set to `Store on Server`. The key must be set to `Cached on Host`. |
| `CK_OBJECT_HANDLE hKey` | The handle to the encryption key to do the encryption. |

## Output Parameters

None.

## Return Values

CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_ARGUMENTS_BAD
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_KEY_FUNCTION_NOT_PERMITTED
CKR_KEY_HANDLE_INVALID
CKR_KEY_SIZE_RANGE
CKR_KEY_TYPE_INCONSISTENT
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_PIN_EXPIRED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_USER_NOT_LOGGED_IN

# C_Encrypt

## Description

Encrypts single part data. Must call `C_EncryptInit` before calling `C_Encrypt`.

```
CK_DEFINE_FUNCTION(CK_RV, C_Encrypt)(
  CK_SESSION_HANDLE hSession,
  CK_BYTE_PTR       pData,
  CK_ULONG          ulDataLen,
  CK_BYTE_PTR       pEncryptedData,
  CK_ULONG_PTR      pulEncryptedDataLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_BYTE_PTR pData` | Pointer to data to be encrypted. |
| `CK_ULONG ulDataLen` | Length of the data to be encrypted. |
| `CK_BYTE_PTR pEncryptedData` | Pointer of the buffer to put the data.<br>First call: Pointer to the metadata to be passed and logged on the application's behalf. Must start with "META:"; input parameter.<br>Second call: Pointer to the resulting encrypted data; output parameter. |
| `CK_ULONG_PTR pulEncryptedDataLen` | Pointer to encrypted buffer length.<br>First call: Pointer to encrypted buffer length. Encrypted buffer length is zero for metadata logging.<br>Second call: actual buffer length used for encrypted data, written encrypted data length. |

## Output Parameters

When input parameter `pEncryptedData` is NULL, this function returns a calculated output buffer length value pointed to by `pulEncryptedDataLen`. For C program only.

## Return Values

`CKR_OK`

## Error Codes

CKR_ARGUMENTS_BAD
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DATA_INVALID
CKR_DATA_LEN_RANGE
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_OPERATION_NOT_INITIALIZED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_PIN_INVALID
CKR_DEVICE_REMOVED
CKR_MECHANISM_INVALID

# C_EncryptUpdate

## Description

Continues a multi-part encryption. This works only with the local key cache. Not supported if crypto is done remotely on the GDE Appliance.

```
CK_DEFINE_FUNCTION(CK_RV, C_EncryptUpdate)(
  CK_SESSION_HANDLE hSession,
  CK_BYTE_PTR       pPart,
  CK_ULONG          ulPartLen,
  CK_BYTE_PTR       pEncryptedPart,
  CK_ULONG_PTR      pulEncryptedPartLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_BYTE_PTR pPart` | Pointer to the data to be encrypted. |
| `CK_ULONG ulPartLen` | Length of the data. |
| `CK_BYTE_PTR pEncryptedPart` | Pointer to the data buffer.<br>First call: Pointer to the metadata to be passed and logged on the application's behalf. Must start with "META:"; input parameter.<br>Second call: Pointer to the resulting encrypted data; output parameter. |
| `CK_ULONG_PTR pulEncryptedPartLen` | Pointer to encrypted buffer length.<br>First call: Pointer to encrypted buffer length. Encrypted buffer length is zero for metadata logging.<br>Second call: actual buffer length used for encrypted data, written encrypted data length. |

## Output Parameters

When input parameter `pEncryptedPart` is NULL, this function returns a calculated output buffer length value pointed to by `pulEncryptedPartLen`.

## Return Values

`CKR_OK`

## Error Codes

| | |
|---|---|
| CKR_ARGUMENTS_BAD | CKR_DEVICE_ERROR |
| CKR_BUFFER_TOO_SMALL | CKR_DEVICE_MEMORY |
| CKR_CRYPTOKI_NOT_INITIALIZED | CKR_DEVICE_REMOVED |
| CKR_DATA_INVALID | CKR_FUNCTION_CANCELED |
| CKR_DATA_LEN_RANGE | |

# C_EncryptFinal

## Description

Finishes a multi-part encryption. This works only with the local key cache. Not supported if crypto is done remotely on the GDE Appliance.

```
CK_DEFINE_FUNCTION(CK_RV, C_EncryptFinal)(
  CK_SESSION_HANDLE hSession,
  CK_BYTE_PTR       pLastEncryptedPart,
  CK_ULONG_PTR      pulEncryptedPartLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_BYTE_PTR pLastEncryptedPart` | Pointer to any remaining data to be encrypted.<br>First call: Pointer to the metadata to be passed and logged on the application's behalf. Must start with "META:"; input parameter.<br>Second call: Pointer to the resulting encrypted data; output parameter. |
| `CK_ULONG_PTR pulLastEncryptedPartLen` | Buffer length pointer.<br>First call: Pointer to encrypted buffer length. Encrypted buffer length is zero for metadata logging.<br>Second call: actual buffer length used for encrypted data, written encrypted data length. |

## Output Parameters

When input parameter `pLastEncryptedPart` is NULL, this function returns a calculated output buffer length value pointed to by `pulLastEncryptedPartLen`. For C program only.

## Return Values

`CKR_ARGUMENTS_BAD`

`CKR_BUFFER_TOO_SMALL`

`CKR_CRYPTOKI_NOT_INITIALIZED`

`CKR_DATA_INVALID`

`CKR_DATA_LEN_RANGE`

`CKR_DEVICE_ERROR`

`CKR_DEVICE_MEMORY`

```
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
```

# Decryption Functions

# C_DecryptInit

## Description

Initializes a decryption option. `C_DecryptInit` must be called before `C_Decrypt` or `C_DecryptUpdate`.

```
CK_DEFINE_FUNCTION(CK_RV, C_DecryptInit)(
  CK_SESSION_HANDLE   hSession,
  CK_MECHANISM_PTR  pMechanism,
  CK_OBJECT_HANDLE    hKey
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_MECHANISM_PTR pMechanism` | Pointer to the mechanism for decryption. Must match the mechanism used to encrypt, including the correct IV if using CKM_AES_CBC or CKM_AES_CBC_PAD or the correct nonce and counter concatenated if using CKM_AES_CTR.<br>We support the following mechanisms:<br>CKM_AES_ECB<br>CKM_AES_CBC<br>CKM_AES_CBC_PAD<br>CKM_AES_CTR<br>CKM_VORMETRIC_FPE (in Java, use: 0x80004001L)<br>**Note:** For CKM_VORMETRIC_FPE, the IV contains the following 3 field values:<br>- tweak (8 bytes)<br>- character set (ASCII characters 0-127)<br>- radix (character set size ) for more information see samples |
| `CK_OBJECT_HANDLE hKey` | Handle of the decryption key; (same as encryption key for symmetric encryption). |

## Output Parameters

None.

## Return Values

| | |
|---|---|
| CKR_ARGUMENTS_BAD | CKR_KEY_SIZE_RANGE |
| CKR_CRYPTOKI_NOT_INITIALIZED | CKR_KEY_TYPE_INCONSISTENT |
| CKR_DEVICE_ERROR | CKR_MECHANISM_INVALID |
| CKR_DEVICE_MEMORY | CKR_MECHANISM_PARAM_INVALID |
| CKR_DEVICE_REMOVED | CKR_OK |
| CKR_FUNCTION_CANCELED | CKR_OPERATION_ACTIVE |
| CKR_FUNCTION_FAILED | CKR_PIN_EXPIRED |
| CKR_GENERAL_ERROR | CKR_SESSION_CLOSED |
| CKR_HOST_MEMORY | CKR_SESSION_HANDLE_INVALID |
| CKR_KEY_FUNCTION_NOT_PERMITTED | CKR_USER_NOT_LOGGED_IN |
| CKR_KEY_HANDLE_INVALID | CKR_PIN_INVALID |

# C_Decrypt

## Description

Decrypts encrypted data in a single part. The operation must have been initialized by a prior call to `C_DecryptInit`.

```
CK_DEFINE_FUNCTION(CK_RV, C_Decrypt)(
  CK_SESSION_HANDLE hSession,
  CK_BYTE_PTR       pEncryptedData,
  CK_ULONG          ulEncryptedDataLen,
  CK_BYTE_PTR       pData,
  CK_ULONG_PTR      pulDataLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_BYTE_PTR pEncryptedData` | Pointer to the encrypted data to be decrypted. |
| `CK_ULONG ulEncryptedDataLen` | Length of the encrypted data to be decrypted. |
| `CK_BYTE_PTR pData` | Pointer to the buffer for the decrypted text.<br>First call: pointer to the metadata to be passed and logged on application's behalf. Must start with "META:"; input parameter.<br>Second call: pointer to the resulting decrypted data; (output parameter). |
| `CK_ULONG_PTR pulDataLen` | Pointer to the length of the buffer for the decrypted text.<br>First call: Pointer to decrypted buffer length. Decrypted buffer length is zero for metadata logging.<br>Second call: actual buffer length used for decrypted data; length of the written plaintext. |

## Output Parameters

When input parameter `pData` is NULL, this function returns a calculated output buffer length value pointed to by `pulDataLen`. For C program only.

## Return Values

`CKR_OK`

## Error Codes

| | |
|---|---|
| CKR_ARGUMENTS_BAD | CKR_MECHANISM_INVALID |
| CKR_BUFFER_TOO_SMALL | CKR_FUNCTION_FAILED |
| CKR_CRYPTOKI_NOT_INITIALIZED | CKR_GENERAL_ERROR |
| CKR_DEVICE_ERROR | CKR_HOST_MEMORY |
| CKR_DEVICE_MEMORY | CKR_OPERATION_NOT_INITIALIZED |
| CKR_DEVICE_REMOVED | CKR_SESSION_CLOSED |
| CKR_ENCRYPTED_DATA_INVALID | CKR_SESSION_HANDLE_INVALID |
| CKR_ENCRYPTED_DATA_LEN_RANGE | CKR_USER_NOT_LOGGED_IN |
| CKR_FUNCTION_CANCELED | CKR_MECHANISM_INVALID |

# C_DecryptUpdate

## Description

Decrypts multi-part encrypted data. The operation must have been initialized by a prior call to `C_DecryptInit`. This works only with the local key cache. Not supported if crypto is done remotely on the GDE Appliance.

```
CK_DEFINE_FUNCTION(CK_RV, C_DecryptUpdate)(
  CK_SESSION_HANDLE hSession,
  CK_BYTE_PTR       pEncryptedPart,
  CK_ULONG          ulEncryptedPartLen,
  CK_BYTE_PTR       pPart,
  CK_ULONG_PTR      pulPartLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_BYTE_PTR pEncryptedData` | Pointer to the encrypted data. |
| `CK_ULONG ulEncryptedDataLen` | Length of the encrypted data. |
| `CK_BYTE_PTR pData` | Pointer to a buffer to write the decrypted data.<br>First call: Pointer to the metadata to be passed and logged on the application's behalf. Must start with "META:"; input parameter.<br>Second call: Pointer to the resulting encrypted data; output parameter. |
| `CK_ULONG_PTR pulDataLen` | Buffer length pointer.<br>First call: Pointer to buffer length. Buffer length is zero for metadata logging.<br>Second call: actual buffer length used for encrypted data, written encrypted data length. |

## Output Parameters

When input parameter `pData` is NULL, this function returns a calculated output buffer length value pointed to by `pulDataLen`. For C program only.

## Return Values

`CKR_OK`

## Error Codes

| | |
|---|---|
| CKR_ARGUMENTS_BAD | CKR_FUNCTION_CANCELED |
| CKR_BUFFER_TOO_SMALL | CKR_FUNCTION_FAILED |
| CKR_CRYPTOKI_NOT_INITIALIZED | CKR_GENERAL_ERROR |
| CKR_DEVICE_ERROR | CKR_HOST_MEMORY |
| CKR_DEVICE_MEMORY | CKR_OPERATION_NOT_INITIALIZED |
| CKR_DEVICE_REMOVED | CKR_SESSION_CLOSED |
| CKR_ENCRYPTED_DATA_INVALID | CKR_SESSION_HANDLE_INVALID |
| CKR_ENCRYPTED_DATA_LEN_RANGE | CKR_USER_NOT_LOGGED_IN |

# C_DecryptFinal

## Description

Finishes a multi-part decryption. This works only with the local key cache. Not supported if crypto is done remotely on the GDE Appliance.

```
CK_DEFINE_FUNCTION(CK_RV, C_DecryptFinal)(
  CK_SESSION_HANDLE hSession,
  CK_BYTE_PTR       pLastPart,
  CK_ULONG_PTR      pulLastPartLen
);
```

## Input Parameters

| Parameter | Description/Value |
|---|---|
| `CK_SESSION_HANDLE hSession` | Session handle. |
| `CK_BYTE_PTR pLastPart` | Pointer to a buffer to write the decrypted data.<br>First call: Pointer to the metadata to be passed and logged on the application's behalf. Must start with "META:"; input parameter.<br>Second call: Pointer to the resulting decrypted data; output parameter |
| `CK_ULONG_PTR pulLastPartLen` | Buffer length pointer.<br>First call: Pointer to decrypted buffer length. Encrypted buffer length is zero for metadata logging.<br>Second call: actual buffer length used for decrypted data, written encrypted data length. |

## Output Parameters

`pLastPart` will be filled in with the decrypted text.

`pulLastPartLen` will be filled in with the length of the decrypted text.

When input parameter `pLastPart` is NULL, this function returns a calculated output buffer length value pointed to by `pulLastPartLen`. For C program only.

## Return Values

CKR_ARGUMENTS_BAD
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_ENCRYPTED_DATA_INVALID
CKR_ENCRYPTED_DATA_LEN_RANGE
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_OK
CKR_OPERATION_NOT_INITIALIZED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_USER_NOT_LOGGED_IN

# GLOSSARY

**access control**
The ability of Vormetric Transparent Encryption (VTE) to control access to data on protected hosts. Access can be limited by user, process (executable), action (for example read, write, rename, and so on), and time period. Access limitations can be applied to files, directories, or entire disks.

`admin` **administrator**
The default DSM administrator created when you install the DSM. `admin` has DSM System Administrator privileges and cannot be deleted.

**Administrative Domain**
(domains). A protected host or group of protected hosts on which an DSM administrator can perform security tasks such as setting policies. Only DSM administrators assigned to a domain can perform security tasks on the protected hosts in that domain. The type of VTE tasks that can be performed depends on the type of administrator. See also **"local domain"**.

**administrator**
See **"DSM Administrator and types"**.

**All Administrator**, **Administrator of type All**
The DSM Administrator with the privileges of all three administrator types: *System*, *Domain* and *Security*.

**appliance**
The DSM server. Often referred to as a *DSM hardware appliance*, which is a hardened DSM server provided by Vormetric, or as a *DSM virtual appliance*, which is the software version of the DSM to be deployed by the customers as a virtual machine.

**asymmetric key cryptography**
See *public key cryptographic algorithm.*

**asymmetric key pair**
A public key and its corresponding private key used with a public key algorithm. Also called a key pair.

**authentication**
A process that establishes the origin of information, or determines the legitimacy of an entity's identity.

**authorization**
Access privileges granted to an entity that convey an "official" sanction to perform a security function or activity.

**block devices**
Devices that move data in and out by buffering in the form of blocks for each input/output operation.

**catch-all rule**
The last policy rule that applies to any GuardPoint access attempt that did not fit any of the other rules in the policy.

**certification authority or CA**
A trusted third party that issues digital certificates that allow a person, computer, or organization to exchange information over the Internet using the public key infrastructure. A digital certificate provides identifying information, cannot be forged, and can be verified because it was issued by an official trusted agency. The certificate contains the name of the certificate holder, a serial number, expiration dates, a copy of the certificate holder's public key (used for encrypting messages and digital signatures) and the digital signature of the certificate-issuing authority (CA) so that a recipient can verify that the certificate is real. This allows others to rely upon signatures or assertions made by the private key that corresponds to the public key that is certified. The CA must be trusted by both the owner of the certificate and the party relying upon the certificate.

**challenge-response**
When a protected host is disconnected from the DSM, the GuardPoint data is not accessible to users. Challenge-response is a password-based procedure that allows users to gain access to their GuardPoint data during disconnection. Users run a utility, `vmsec challenge`, a seemingly random string (the challenge) is displayed. The user calls this in to their DSM Security administrator. The administrator returns a counter-string (the response) that the host user must enter to decrypt guarded data.

**Character device**
See ***"raw device."***

**ciphertext**
Data in its encrypted form. Ciphertext is the result of encryption performed on plaintext using an algorithm, called a cipher.

**cleartext or plaintext**
Data in its unencrypted form.

**cryptographic algorithm**
A computational procedure that takes variable inputs, including a cryptographic key, and produces ciphertext output. Also called a cipher. Examples of cryptographic algorithms include AES, ARIA, and DES.

**cryptographic key**
See "**encryption key**."

**cryptographic signature**
See "**signing files**."

**Database Encryption Key (DEK)**
A key generated by Microsoft SQL when TDE is enabled.

**Data Security Manager (DSM)**
Sometimes called the *Security Server* or *appliance*. A Vormetric server that acts as the central repository and manager of encryption keys and security policies. Receives instructions and configuration from administrators through a GUI-based interface called the *Management Console*. Passes and receives information to and from VTE Agents. Available as a complete hardened hardware system (*DSM Appliance*) or as software solution installed on a UNIX box (*software-only DSM*).

**dataxform**
A utility to encrypt data in a directory. Short for "data transform."

**DB2**
A relational model database server developed by IBM.

**Decryption**
The process of changing ciphertext into plaintext using a cryptographic algorithm and key.

**Digital signature**
A cryptographic transformation of data that provides the services of origin authentication, data integrity, and signer non-repudiation.

**domains**
See *administrative domains*.

**Domain Administrator**
The second-level DSM administrator created by a DSM *System Administrator*. The DSM *Domain Administrator* creates and assigns DSM *Security Administrators* to domains and assigns them their security "**roles**". See "**DSM Administrator and types**".

**Domain and Security Administrator**
A hybrid DSM administrator who is has the privileges of a DSM Domain Administrator and Security Administrator.

**DSM**
See ***"Data Security Manager (DSM)."***

**DSM Administrator and types**
Specialized system security administrators who can access the Vormetric DSM Management Console. There are five types of DSM administrators:

*DSM System Administrator* - Creates/removes other DSM administrators of any type, changes their passwords, creates/removes, domains, assigns a Domain Administrator to each domain. Cannot do any security procedures in any domain.

*Domain Administrator* - Adds/removes DSM Security Administrators to domains, and assign roles to each one. Cannot remove domains and cannot do any of the domain security roles.

*Security Administrator* - Performs the data protection work specified by their roles. Different roles enable them to create policies, configure hosts, audit data usage patterns, apply GuardPoints, and so on.

*Domain and Security Administrator* - Can do the tasks of DSM Domain and Security Administrators.

*All* - Can do the tasks of all three of the DSM administrative types

**DSM Automation Utilities**
Also called VMSSC. A set of command line utilities that is downloaded and installed separately on the protected host or any networked machine. These utilities can be used by advanced users to automate DSM processes that would normally be done with the Management Console. See the *DSM Automation Reference* for complete details.

**DSM CLI**
A command line interface executed on the DSM to configure the DSM network and perform other system-level tasks. See the *DSM Command Line Interface* documentation

**DSM CLI Administrator**
A user who can access the DSM CLI. DSM CLI Administrators are actual system users with real UNIX login accounts. They perform tasks to setup and operate the DSM installation. They do not have access to the Management Console.

**DSM database**
A database associated with the DMS containing the names of protected hosts, policies, GuardPoints, settings, and so on.

**DSM System Administrator**
The highest level of DSM administrator. This administrator creates/removes other DSM administrators of any type, creates/removes domains, and assigns a Domain Administrator to each domain. The DSM System Administrator cannot perform any security procedures in any domain or system. This administrator is not related to computer or network system administrators.

**EKM**
See "**Encryption Key Management (EKM)**."

**Encryption**
The process of changing plaintext into ciphertext using a cryptographic algorithm and key.

**encryption agent**
See *Vormetric Transparent Encryption agent*.

**encryption key**
A piece of information used in conjunction with a cryptographic algorithm that transforms plaintext into ciphertext, or vice versa during decryption. Can also be used to encrypt digital signatures or encryption keys themselves. An entity with knowledge of the key can reproduce or reverse the operation, while an entity without knowledge of the key cannot. Any VDS policy that encrypts GuardPoint data requires an encryption key.

**Encryption Key Management (EKM)**
An API library specification provided by Microsoft that defines a software framework that allows hardware security module (HSM) providers like Vormetric to integrate their product with the Microsoft SQL Server.

**failover DSM**
A secondary DSM that assumes the policy and key management load when a protected host cannot connect to the primary DSM or when a protected host is specifically assigned to the failover DSM. A failover DSM is almost identical to the primary DSM, having the same keys, policies, protected hosts, and so on.

**file signing**
See *signing files*.

**File System Agent**
A Vormetric software agent that resides on a host machine and allows administrators to control encryption of, and access to, the files, directories and executables on that host system. For example, administrators can restrict access to specific files and directories to specific users at specific times using specific executables. Files and directories can be fully encrypted, while the file metadata (for example, the file names) remain in cleartext. Also called the "**VTE Agent**".

**FQDN**
Fully qualified domain name. A domain name that specifies its exact location in the tree hierarchy of the Domain Name Server (DNS). For example: `example.vormetric.com`.

**GPFS**
General Parallel File System is a high-performance shared-disk clustered file system developed by IBM.

**GuardPoint**
A location in the file system hierarchy, usually a directory, where everything underneath has a Vormetric data protection policy applied to it. The File System Agent intercepts any attempt to access anything in the GuardPoint and uses policies obtained from the DSM to grant or deny the access attempt. Usually, depending on the policies, data copied into a GuardPoint is encrypted, and only authorized users can decrypt and use that GuardPoint data.

**Hardware Security Module or HSM**
A tamper-resistant hardware device that stores keys and provides stringent access control. It also provides a random number generator to generate keys. The DSM Appliance can come with an embedded Hardware Security Module.

**host locks**
Two Management Console options, **FS Agent Locked** and **System Locked,** that are used to protect the File System Agent and certain system files. File System Agent protection includes preventing some changes to the File System Agent installation directory and preventing the unauthorized termination of File System Agent processes.

**host password**
This is not a regular login or user password. This is the password entered by a host system user to unlock a GuardPoint when there is no DSM connection. This password decrypts cached keys when the DSM is not accessible. The host must also be configured with **Cached on Host** keys. See "**challenge-response**".

**initial test policy**
A first data security policy applied to a GuardPoint that is used to gather directory access information so DSM Security Administrators can create a permanent operational policy. The initial test policy encrypts all data written into the GuardPoint; decrypts GuardPoint data for any user who access it; audits and creates log messages for every GuardPoint access; reduces log message "noise" so you can analyze the messages that are important to you for tuning this policy; is run in the "Learn Mode" which does not actually deny user access, but allows you to record GuardPoint accesses.

After enough data is collected, the DSM Security Administrator can modify the initial test policy into an operational policy.

**Key Agent**
A Vormetric agent that provides an API library supporting a subset of the PKCS#11 standard for key management and cryptographic operations. It is required for the following products: Vormetric Key Management (VKM), Vormetric Tokenization, Vormetric Application Encryption (VAE), Vormetric Cloud Encryption Gateway (VCEG). Sometimes called the *VAE Agent*.

**key group**
A key group is a collection of asymmetric keys that are applied as a single unit to a policy.

**key management**
The management of cryptographic keys and other related security objects (for example, passwords) during their entire life cycle, including their generation, storage, establishment, entry and output, and destruction.

**key template**
A template that lets you quickly add agent keys or third-party vault keys by specifying a template with predefined attributes. You can define specific attributes in a template, then you can call up the template to add a key with those attributes.

**key shares**
When data is backed up or exported from VTE (for example, symmetric keys or DSM database backups), they can be encrypted in a wrapper key needed to restore the exported data on the new machine. Wrapper keys can be split and distributed to multiple individuals. Each split piece of the wrapper key is called a *key share*. Decrypting the data requires that some specified number of the individuals that received key shares contribute their key share to decrypt the data.

**key wrapping**
A class of symmetric encryption algorithms designed to encapsulate (encrypt) cryptographic key material. The key wrap algorithms are intended for applications such as protecting keys while in untrusted storage or transmitting keys over untrusted communications networks. Wrapper keys are broken up into *key shares*, which are pieces of a wrapper key. Key shares are divided amongst two or more *custodians* such that each custodian must contribute their key share in order to assemble a complete wrapper key.

**Key Vault**
A Vormetric product that provides passive key vaulting. It securely stores symmetric and asymmetric encryption keys from any application and tracks key expiration dates.

**Learn Mode**
A DSM operational mode in which all actions that would have been denied are instead permitted. This permits a policy to be tested without actually denying access to resources. In the Learn Mode, all GuardPoint access attempts that would have been denied are instead permitted. These GuardPoint accesses are logged to assist in tuning and troubleshooting policies.

**local domain**
A DSM domain in which DSM administration is restricted to Domain Administrators or Security Administrators assigned to that domain. To access a local domain in the Management Console, a DSM administrator must specify their local domain upon login.

**Management Console**
The graphical user interface (GUI) to the DSM.

**Master encryption key (MEK)**
The encryption key for Oracle Database used to encrypt secondary data encryption keys used for column encryption and tablespace encryption. Master encryption keys are part of the Oracle Advanced Security Transparent Data Encryption (TDE) two-tier key architecture.

**MEK**
See *Master encryption key.*

**Microsoft SQL Server**
A relational database server, developed by Microsoft.

**Microsoft SQL Transparent Data Encryption (MS-SQL TDE)**
Microsoft SQL Server native encryption for columns and tables.

**multi-factor authentication**
An authentication algorithm that requires at least two of the three following authentication factors:
1) something the user knows (for example, password); 2) something the user has (example: RSA SecurID); and 3) something the user is (example: fingerprint). VTE implements an optional form of multi-factor authentication for Management Console users by requiring DSM administrators to enter the token code displayed on an RSA SecurID, along with the administrator name each time the administrator logs on to the Management Console.

**multitenancy**
A VTE feature that enables the creation of multiple local domains within a single DSM. A local domain is a DSM domain in which DSM administration is restricted to Domain Administrators or Security Administrators assigned to that domain. This allows Cloud Service Providers to provide their customers with VTE administrative domains over which the customer has total control of data security. No other administrators, including CSP administrators, have access to VTE security in a local domain.

**offline policy**
Policies for Database Backup Agents. *Online policies* are for the File System Agent.

**one-way communication**
A VTE feature for an environment where the DSM cannot establish a connection to the agent, but the agent can establish a connection to the DSM. For example, the protected host is behind a NAT so protected host ports are not directly visible from the DSM, or the protected host is behind a firewall that prohibits incoming connections, or the protected host does not have a fixed IP address as in the cloud. When an agent is registered with one-way communication, changes made for that protected host on the DSM are not pushed to the protected host, rather as the protected host polls the DSM it will retrieve the change.

**online policies**
Policies for the File System Agent. *Offline policies* are for Database Backup Agents.

**policy**
A set of security access and encryption rules that specify who can access which files with what executable during what times, and whether or not those files are encrypted. Policies are created by DSM Security Administrators, stored in the DSM, and implemented on protected hosts by a File system Agent. See "**rule (for policies)**".

**policy tuning**
The process of creating a simple Learn Mode policy that allows any protected host user to access a GuardPoint; to examine who accesses the GuardPoint, what executables they use, and what actions they require; and to modify the policy such that it allows the right people, using the right executable, performing the right action to do their job, and prevent anyone else from inappropriate access.

**process set**
A list of processes that can be used by the users in a user set associated with a policy rule.

**protected host**
A host on which a VTE Agent is installed to protect that host's data.

**public key cryptographic algorithm, public key infrastructure**
A cryptographic system requiring two keys, one to lock or encrypt the plaintext, and one to unlock or decrypt the ciphertext. Neither key can do both functions. One key is published (*public key*) and the other is kept private (*private key*). If the lock/encryption key is the one published, the system enables private communication from the public to the unlocking key's owner. If the unlock/decryption key is the one published, then the system serves as a signature verifier of documents locked by the owner of the private key. Also called asymmetric key cryptography.

**raw device**
A type of block device that performs input/output operations without caching or buffering. This results in more direct access.

**register host**
The process of enabling communication between a protected host and the DSM. Registration happens during agent installation. Before registration can happen, the host must be added to the DSM database.

**rekeying**
The process of changing the encryption keys used to encrypt data. Changing keys enhances data security and is a requirement to maintain compliance with some data security guidelines and regulations. Also called *key rotation*.

**roles**
A set of Management Console permissions assigned to DSM Security Administrators by DSM Domain Administrators. There are five roles: *Audit* (can generate and view logging data for file accesses), *key* (can create, edit, and delete keys), *Policy* (can create, edit, and delete policies), *Host* (can configure, modify, and delete protected hosts and protected host groups), and *Challenge & Response* (can generate a temporary password to give to a protected host user to decrypt cached encryption keys when connection to the DSM is broken).

**RSA SecurID**
A hardware authentication token that is assigned to a computer user and that generates an authentication code at fixed intervals (usually 60 seconds). In addition to entering a static password, Management Console administrators can be required to input an 8-digit number that is provided by an external electronic device or software.

**rule (for policies)**
Every time a user or application tries to access a GuardPoint file, the access attempt passes through each rule of the policy until it finds a rule where all the criteria are met. When a rule matches, the *effect* associated with that rule is enforced. A rule consists of five access criteria and an effect. The criteria are Resource (the file/directories accessed), User (the user or groups attempting access), Process (the executable used to access the data), When (the time range when access is attempted) and Action (the type of action attempted on the data, for example read. write, rename and so on). *Effect* can be permit or deny access, decrypt data access, and audit access attempt. See *policy*.

**secfs**
1) The File System Agent initialization script. 2) An acronym for Vormetric Secure File System agent. It generally refers to the kernel module that handles policies (locks, protected host settings, logging preferences) and keys, and enforces data security protection.

**secvm**
A proprietary device driver that supports GuardPoint protection to raw devices. `secvm` is inserted in between the device driver and the device itself.

**Security Administrator**
The third-level DSM administrator who does most of data protection work like creating policies, configuring protected hosts, auditing data usage patterns, applying GuardPoints and other duties. The privileges of each Security Administrator is specified by the roles assigned to them by the Domain Administrator. See *roles*. See “**DSM Administrator and types**”.

**Security Server**
See “**DSM**”.

**separation of duties**
A method of increasing data security by creating customized DSM administrator roles for individual DSM administrators such that no one administrator has complete access to all encryption keys in all domains of all files.

**signing files**
File signing is a method that VTE uses to check the integrity of executables and applications before they are allowed to access GuardPoint data. If file signing is initiated in the Management Console, the File System Agent calculates the cryptographic signatures of the executables that are eligible to access GuardPoint data. A tampered executable, such as a Trojan application, malicious code, or rogue process, with a missing or mismatched signature, is denied access. Also called *cryptographic signatures*.

**Suite B mode**
A set of publicly available cryptographic algorithms approved by the United States National Security Agency (NSA). These algorithms enhance security by adding up to 384-bit encryption to the communication between the Web browser and the DSM, the DSM and Agent, and between DSMs in HA environments.

**Symmetric-key algorithm**
Cryptographic algorithms that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in comparison to public-key encryption.

**System Administrator (DSM)**
See **"DSM Administrator and types"**.

**Transparent Data Encryption (TDE)**
A technology used by both Microsoft and Oracle to encrypt database content. TDE offers encryption at a column, table, and tablespace level. TDE solves the problem of protecting data at rest, encrypting databases both on the hard drive and consequently on backup media.

**user set**
A named list of users on which a policy rule applies.

**VAE Agent**
See **"Key Agent"**.

**vmd**
Acronym for Vormetric Daemon, vmd is a process that supports communication between the DSM and kernel module.

**VMSSC or Vormetric Security Server Command Line Interface**
See *DSM Automation Utilities*.

**Vormetric Application Encryption (VAE)**
A product that enables data encryption at the application level as opposed to the file level as is done with VTE.

Where VTE encrypts a file or directory, VAE can encrypt a column in a database or a field in an application. VAE is essentially an API library for key management and cryptographic operations based on PKCS#11. See the *Vormetric Application Encryption Installation and API Reference Guide*.

**Vormetric Cloud Encryption Gateway (VCEG)**
Vormetric product that safeguards files in cloud storage environments, including Amazon Simple Storage Service (Amazon S3) and Box. The cloud security gateway solution encrypts sensitive data before it is saved to the cloud storage environment, then decrypts data for approved users when it is removed from the cloud.

**Vormetric Data Security Platform or VDS Platform**
The technology platform upon which all other Vormetric products—Vormetric Transparent Encryption (VTE), Vormetric Application Encryption (VAE), Vormetric Key Management (VKM), Vormetric Cloud Encryption Gateway (VCEG), Vormetric Tokenization Server (VTS), Vormetric Key Management (VKM), and Vormetric Protection for Teradata Database—are based.

**Vormetric Encryption Expert or VEE**
Earlier name of the Vormetric Transparent Encryption (VTE) product. It may sometimes appear in the product GUI or installation scripts.

**Vormetric Key Management (VKM)**
Vormetric product that provides a standards-based platform for storing and managing encryption keys and certificates from disparate sources across the enterprise. This includes Vormetric encryption keys, 3rd-party software keys, KMIP device keys and so on.

**Vormetric Protection for Teradata Database**
Vormetric product that secures sensitive data in the Teradata environment.

**Vormetric Security Intelligence**
Vormetric product that provides support for Security Information and Event Management (SIEM) products such as ArcSight, Splunk and QRadar. Provides solutions that monitor real-time events and analyze long-term data to find anomalous usage patterns, qualify possible threats to reduce false positives, and alert organizations when needed. Documented in the VDS Platform Security Intelligence User Guide.

**Vormetric Tokenization Server (VTS)**
Vormetric product that replaces sensitive data in your database (up to 512 bytes) with unique identification symbols called tokens. Tokens retain the format of the original data while protecting it from theft or compromise.

**Vormetric Transparent Encryption or VTE**
Vormetric product that protects data-at-rest. Secures any database, file, or volume without changing the applications, infrastructure or user experience.

**Vormetric Vault**
A virtual vault to store 3rd-party encryption keys, certificates and other security objects.

**VTE Agent**
Vormetric agents that are installed on protected hosts to implement data protection. See “**File System Agent**”.

**wrapper keys**
See “**key wrapping**”.

**WSDL**
Web Services Description Language.

# Index

## Symbols



## A



## C

## D



## E

## F



## H



## I



## J



## K



## L



## M



## N



## O

## P



## R



## S



## T



## U

## V



## W