IBM® InfoSphere Guardium Database Encryption 3.0.0.1

Vormetric Tokenization Server (VTS)

Release 3.0.0.1

June 22, 2018-

IBM Guardium Database Encryption 3b corresponds to Vormetric Data Security Platform Release 6. Release 6 contains the Data Security Manager Release 6.0.2 and Vormetric Transparent Encryption Agent Release 6.0.2.

CONTENTS

Preface

P

The *Vormetric Tokenization Server Installation, Administration, and Programming Guide* describes how to install, configure, and manage VTS and provides guidelines and some REST API documentation for developers to use when VTS and implementing VTS into their applications.

The VTS AP service categories are:

- Tokenization
- Encryption
- Key Management.

NOTE: Additional REST API documentation is provided from the in product help.

DOCUMENTATION VERSION HISTORY

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

Documentation Changes

Product/Document Version	Date	Changes
2.2.0 GA	05/22/18	Update to GA version and format and style clean up.
2.2 Beta	04/9/18	Addition of crypto and key management services and APIs; redesigned GUI look-and-feel; new three-way authorization schema permitting user-based permissions to tokenize, encrypt, verify, sign, etc. with specific keys; support for client-certificate user authorization; support for custom character sets, including Chinese/Japanese/Korean (CJK) use cases; additional programmer's notes and links to external REST API documentation; all-new introduction and concepts material.
2.1.2 v2	1/29/2018	Minor updates, added notes to "Create a token group," updated install requirements section
2.1 v3	2/27/2017	Corrected VTS port configuration table, removed port 7024

ASSUMPTIONS

This guide addresses three primary audiences:

- **Network administrators** who install the Vormetric Tokenization Server (VTS) package and configure it using the VTS Administrator CLI;
- **VTS GUI administrators** who assign and manage users, keys, and permissions in the VTS GUI;
- **VTS application developers** who integrate VTS functionality into their application code using the VTS REST APIs.

RELATED DOCUMENTATION

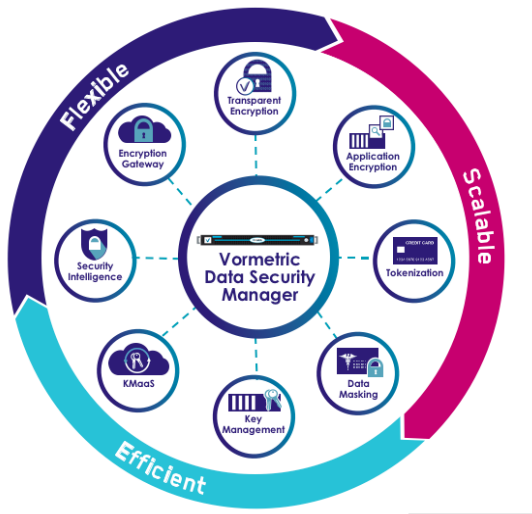
The following related documents are available to registered users on the Vormetric Web site at https://support.vormetric.com

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

VORMETRIC DATA SECURITY PLATFORM OVERVIEW

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

Figure 1: The Vormetric "Solar System"



The platform consists of products that share a common, extensible infrastructure. At the heart of the platform is the Guardium Database Encryption (GDE) Appliance, which coordinates policies, keys, and the collection of security intelligence, all of which is managed and observed through your browser. In addition to the GDE Appliance, the Vormetric Data Security Platform consists of the following products:

- **Vormetric Application Encryption (VAE)** provides a framework to deliver application-layer encryption such as column- or field-level encryption in databases, Big Data, or PaaS applications. VAE is an application encryption library providing a standards-based API to do cryptographic and encryption key management operations into existing corporate applications.
- **Vormetric Cloud Encryption Gateway (VCEG)** safeguards files in cloud-storage environments, including Amazon S3 and Box. VCEG encrypts sensitive data before it is saved to the cloud, enabling security teams to establish visibility and control around cloud assets.
- **Vormetric Key Management (VKM)** centralizes 3rd-party encryption keys and stores certificates securely. It provides standards-based enterprise encryption key management for Transparent Database Encryption (TDE), KMIP-compliant devices, and offers vaulting and inventory of certificates.
- **Vormetric Protection for Teradata Database** provides granular controls to secure assets in Teradata environments. It simplifies the process of using column-level encryption in your Teradata database. It provides documented, standards-based APIs and user-defined functions (UDFs) for cryptographic and key management operations.

- **Vormetric Security Intelligence** provides comprehensive logging combined with Security Information Event Management (SIEM) systems to detect advanced persistent threats and insider threats. In addition, the logs satisfy compliance and regulatory audits.
- **Vormetric Tokenization with Dynamic Data Masking (VTS)** replaces sensitive data in your database with unique identification symbols called tokens. This reduces the number of places that clear-text sensitive data reside, and thus reduces the scope of complying with PCI DSS and corporate security policies.
- **Vormetric Transparent Encryption (VTE)** secures any database, file, or volume across your enterprise without changing the applications, infrastructure, or user experience.

HOW TO GET HELP

For support and troubleshooting issues:

- help.thalesesecurity.com
- support@thalesesecurity.com
- (877) 267-3247

For Thales Sales:

- http://go.thalesesecurity.comcontact-data-security-specialist-form.html
- sales@thalesesec.net
- (888) 267-3732

Introduction to VTS

1

The Vormetric Tokenization Server (VTS) package is a platform-independent virtual appliance that comprises three security services and their associated tools. VTS works in conjunction with the **Guardium Database Encryption (GDE) Appliance.**, as well as (optionally) with AD/LDAP servers and remote logging servers.

This chapter includes the following sections:

What is VTS?

This section describes the core services and components that make up VTS, as well as highlighting the specific benefits of the tokenization feature.

VTS Core Services

VTS offers three service categories for handling sensitive data:

- **Tokenization service:** Tokenization is used for replacing sensitive data with tokenized, or encrypted, data. Tokenized data retains the format of the original data while protecting it from theft or compromise. For example, if a company accepts customer credit card numbers, those numbers can be tokenized and stored as random-looking numbers, then detokenized by an employee or application with the proper permissions.

  Tokenization is frequently used for sensitive data such as credit card numbers, social security numbers, drivers licenses, or other personally identifiable information (PII). *Data masking* can be applied to any detokenized data to hide sections of the data from different groups of users. For example, less-privileged database users might view only the last four digits of a detokenized credit card number, while a more privileged user could view the entire card number. The *Vormetric Tokenization* REST APIs are used to integrate this functionality into a developer's application.

The baseline tokenization service has been extended to include two additional service categories:

- **Key management services:** Using the GDE Appliance as the key provider and repository, *users*-- whether human or machine-- can be granted permission to create, modify, find, import, export, and/or destroy encryption keys. The *Vormetric Key Management* REST APIs are used to integrate this functionality into a developer's application.
- **Cryptographic services:** VTS also includes data encryption, decryption, sign, and verify services which can be tied to specific keys and user permissions. They can also be integrated into a developer's application using the *Vormetric Cryptographic* REST APIs.

Components in a VTS Deployment

A VTS deployment consists of the following components:

Vormetric Tokenization Server

This virtual appliance can:

- Receive REST API calls from the three service categories and return decrypted/detokenized data to requesting client
- Tokenize/detokenize, encrypt/decrypt, sign, and verify data using keys from the GDE Appliance
- Apply dynamic data masking as appropriate
- Authenticate users via user name/password
- Authenticate users via client certificates (optional)
- Apply user or user group profiles to determine user permissions
- Associate user permissions with particular keys

The VTS includes:

- **VTS Administrative CLI** used by the network administrator when installing and configuring the server.
- **VTS GUI** used by a GUI administrator to manage users, groups, permissions, data masks, and GDE-based keys.
- **REST APIs** that cover tokenization, cryptography, and key management.

The server can be installed as an ISO or OVA image.

Guardium Database Encryption (GDE) Appliance. The GDE Appliance must be installed and configured before deploying VTS. It provides and retains the keys used for tokenizing and encrypting data.

LDAP/Active Directory Server (Optional)

AD/LDAP can be used to import and authenticate VTS users. It is accessed via LDAP or LDAPS.

Remote Logging Server (Optional)

The network administrator can configure the VTS to send system log messages to a remote logging server.

Tokenization Benefits

Vormetric Tokenization:

- Replaces sensitive data in databases with tokens. This reduces the number of places in which plain text credit card numbers reside, and thus reduces the scope of complying with the Payment Card Industry Data Security Standard (PCI DSS) and corporate security policies.
- Preserves the format of data in a way that reduces the operational impact associated with encryption and other obfuscation techniques. For example, you can tokenize a credit card field in a database, yet keep the tokenized information in a format that is compatible with associated applications.
- Enables outsourcing application testing and running analytics without giving access to sensitive assets because the format of the data has been preserved. To outsource, you can create a copy of the production database and give that copy to the outsourced development team.
- Creates strong separation of duties between privileged administrators and data owners. In this way, IT administrators, such as hypervisor, cloud, storage, and system administrators can perform their tasks without access to the sensitive data residing on those systems.
- Enables dynamic data masking—the ability to establish varying levels of data redaction for different database users. For example, you can enable customer service personnel to access the last four digits of a customer’s credit card number, while an accounts payable representative can access the full credit card number. Integrates tokenization users with existing LDAP-based identity directories. Security teams can efficiently set granular tokenization policies for specific users and groups.

Understanding VTS Authentication & Authorization

With VTS 2.2, users are authenticated, granted permissions (to tokenize, detokenize, encrypt, hash, etc.), and associated with particular keys, using a three-way permissions matrix. It is important to understand the underlying assumptions about users, user groups, permissions, and keys in order to set up your VTS project appropriately.

What is a VTS "user?"

VTS makes no official distinctions between "human user accounts," for real people, and "service user accounts," representing daemons, scripts, processes, applications, or groups within applications with programmatic access to VTS services. Both are referred to as "users."

From the point of view of the VTS system, the VTS REST APIs, and the application developer who wishes to incorporate VTS services into an application, a basic user account consists of:

- **user name/identifier**
- **password or client certificate**[1]

These attributes form the basis of user authentication.

NOTE: As a best practice in VTS, it is recommended to collect individual users into logical user groups and assign permissions at the group level.
See also "Understanding VTS User Groups & Permissions" on page 10.

User Profiles

Human users

Of course, some users of VTS are real people. There are, first of all, the administrators, staff, and developers who use the VTS interface and/or the VTS APIs to install, configure, integrate, and maintain V TS services.

There are also human "end-users," such as employees in a customer service department who may be assigned to a VTS user group authorized to detokenize credit cards and to view the last four digits of the card, based on the data mask that was applied to their user group.

Service users

1. If authentication is done from a client certificate, the password is not used by VTS.
See "Client Certificates & User Authentication" on page 6 for more detail.

However, most VTS usage consists of service users, where various servers and applications pass authentication, tokenization, encryption, and key information between themselves to complete a transaction of some kind.

For example:

A company may set up a structure to assign a financial application as User1 with a certain number of keys to perform certain operations. Within that same application, User2 might be a process that uses key2 to tokenize/detokenize credit cards, and User3 might be used to encrypt/decrypt social security numbers with key3.

VTS Authentication Basics

As described in “What is a VTS “user?”” on page 4, authentication is based on credentials— either user name and password or a client certificate.

VTS will first seek to validate whether the credentials match entries in its local database. If not, and if AD/LDAP is configured, then it will check against AD/LDAP settings. When authentication is successful, the system will also check for VTS role assignment[1], any key association, and any active permissions.

AD/LDAP & User Authentication

Companies can choose to use Active Directory/LDAP to store both human and service user accounts for VTS, to have a centralized way to manage them. If this approach is chosen, then the implementation involves cooperation between:

- **The network administrator**, to configure the connection between AD/LDAP and VTS;
- **The AD/LDAP administrator** and other organizational decision makers who decide which users should be imported into the VTS database, how to group them, what user group names to assign them in AD/LDAP, and what types of permissions should be granted on a group and/or individual basis;
- **The VTS GUI administrator**, who must manually enter the designated user group names into the VTS GUI before the user accounts are imported.
  See also “Understanding VTS User Groups & Permissions” on page 10.

It is also possible to create local user accounts directly in VTS. This is useful in test or demo installations of VTS, or might be used for enhanced security, as no AD/LDAP administrator would have access to the local user credentials.

1. See “VTS roles “Superuser,” “Staff,” and “Active”” on page 7 for details.

Client Certificates & User Authentication

Client certificates can be used for authentication. The certificate itself is a credential and can be safer than a password, as it cannot be easily shared, duplicated, written on a sticky note, or otherwise carelessly exposed.

Companies might enable simple client cert authentication, to verify that a machine has a valid cert to connect with VTS. Or they may choose a more sophisticated implementation.
With VTS 2.2 it is possible to use the common name (`CN`) inside a client cert and enter it into VTS as a user name, thereby taking advantage of the ability to assign granular permissions and/or associate particular keys to a client certificate. This is known as using a client certificate "with identities."

To implement client certs with granular permissions, an organization would need to:

- Have established an in-house Certificate Authority (CA);
- Have an exportable CA cert;
- Issue any number of client certs (signed by the CA) that each contain a common name (`CN`).
- Go to the VTS CLI to:
- Upload the CA cert to VTS;
- Enable using client certs with ID.(See "Enable the Client Authentication Feature (optional)" on page 43.).
- Use the VTS graphical user interface (GUI) for:
- Creating users and entering common names as user names (or import from Active Directory for LDAP)
- Managing user permissions (If importing from LDAP, you can use the VTS CLI setting to designate a user group as the "active" user group thereby activating all members of the group.)
- Associate key(s) with the cert/user.
- Assign permissions to the cert/user.

VTS Authorization Basics

VTS employs a multi-layered security model with separation of duties and granular permissions settings built in to it. It is also flexible, adaptive to individual corporate structures and policies. This structure implies some inherent complexity, but the concepts below form the foundational assumptions.

VTS roles “Superuser,” “Staff,” and “Active”

When a user is created VTS, it can be assigned one or more of the following core VTS roles:

- **Superuser:** Can perform both read and write actions in the VTS GUI.
  Superusers are GUI administrators who manage users, user groups, keys, tokenization groups, tokenization templates, data masks, and character sets. Superusers associate users with key(s) and assign the crypto, key, and tokenization permissions to other users. Superusers are automatically also “Staff.”
- **Staff:** Users with read-only permission to the VTS GUI.
- **Active:** Users must be “active” in order to have encryption, key, or tokenization permissions. Active users are those authorized to use the VTS REST APIs.

The role-based permissions (Active/Staff/Superuser) are incremental. With no role assigned, the user cannot access any VTS functionality. (This is equivalent to being deleted, but the information is retained in the database for possible reactivation.)

NOTE: When users are created directly in the VTS GUI, they are automatically assigned the status “active,” which can be disabled as needed.

If users are imported from AD/LDAP, the “active” designator is handled differently. See “Understanding VTS User Groups & Permissions” on page 10 for details.

As of VTS 3.0.0.1, if Active is unchecked, the user cannot log in to the GUI even if assigned Superuser/Staff status.

Key-aware Permissions Matrix

VTS 3.0.0.1 allows the GUI Administrator to assign permissions based on user (or user group), action, and key, similar to the English sentence:
“User `vtsuser1` may encrypt, decrypt, tokenize and detokenize with symmetric key `Sym_key1`.

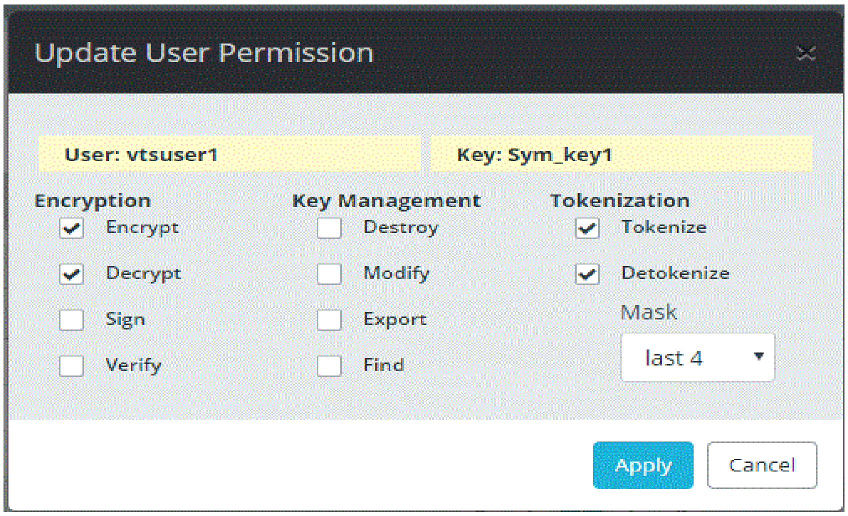


Figure 1: Permissions available on symmetric keys.

NOTE: Permissions can be assigned at the user and the user group level, and are additive. See also “Understanding VTS User Groups & Permissions” on page 10.

Different types of keys (symmetric, asymmetric, and opaque objects) have different permissions available. See also “Understanding VTS Keys” on page 10.

Tokenize and detokenize functionality require a couple of additional steps.
See also “Tokenization Groups, Templates, & Masks: How Do They Fit In?” on page 13.

Other (None Key Dependent) Permissions

Users or user groups can be authorized to dynamically create and/or import key names and key material on the GDE Appliance, under the “Other Permissions” column of the Permissions main page.

NOTE: See also “Static vs. Dynamic Key Management Creation” on page 12.

- **Create** key: creates the key name and key material in the GDE Appliance; the key name will be listed in the VTS GUI.
- **Import** key: used to import any key from one GDE Appliance to another, or from an external system into the GDE Appliance. The imported key name will be listed in the VTS GUI.

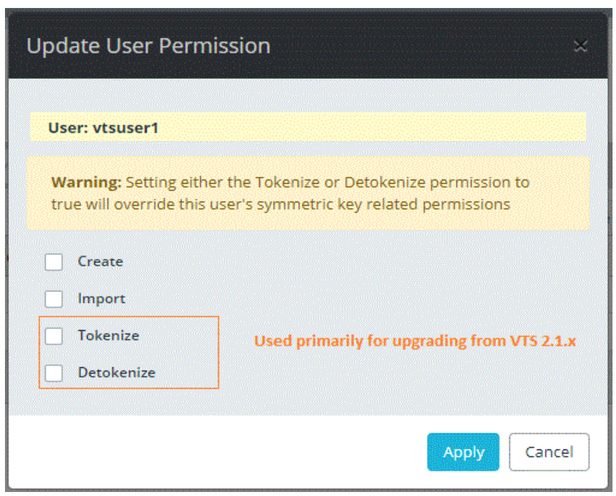


Figure 2: “Create” and “Import” keys are distinct from the legacy Tokenize/Detokenize function.

NOTE: The *Other Permissions* assignment window also includes check boxes for **Tokenize** and **Detokenize**, along with a warning about overriding symmetric key assignments (Figure 2).

This Tokenize/Detokenize feature is used during upgrade from earlier versions of VTS, as it grandfathers in the global tokenize/detokenize feature that was used previously.

NOTE: See “Other Permissions: Tokenize/Detokenize” on page 68 for usage tips.

Understanding VTS User Groups & Permissions

- **VTS role-based user groups:**
  The three core VTS roles (Superuser/Staff/Active) are also three required core VTS user groups. An organization may choose to assign different group names, but the meanings are retained (see "VTS roles "Superuser," "Staff," and "Active"" on page 7 for the definitions). VTS GUI administrators must create at least three user groups that correspond to the roles Superuser/Staff/Active.
- **Assigning users to user groups two different ways:**
  If importing from AD/LDAP, users will be sorted into their role groups from the CLI.
  If entering users directly in the GUI, the administrator will assign the role(s) on the user's page and then check the appropriate group(s).
- **Adding more user groups:**
  Additional groups can be added to VTS for organizational purposes.
- **Assigning group permissions:**
  Key-based and non-key-dependent permissions can be assigned at the user group level or the user level, but assigning at the group level is recommended.
- **Permissions are additive:**
  Individual and group permissions are additive. For example, suppose Joe is a member of the user group "Finance," which has permission to tokenize credit card numbers, and Joe is also a team lead granted user-level permission to detokenize. In total, Joe can tokenize and detokenize.

Users, User groups, & Tokenization Groups

To allow for abbreviated Tokenization REST API calls and to enable division of an organization into sub-organization either by location or department, VTS adds a so-called "tokenization group" concept. So, there could be a user group "Finance" in San Francisco, and another "Finance" user group in Paris, each associated with its own tokenization group.

NOTE: See also "Tokenization Groups, Templates, & Masks: How Do They Fit In?" on page 13.

Understanding VTS Keys

VTS supports three types of keys: **symmetric**, **asymmetric**, and **opaque objects**, each of which uses different algorithms and can be used for different actions.

In VTS 2.2, user permissions (to encrypt, decrypt, sign, verify, hash, or tokenize/detokenize) must be tied to particular keys. The *Vormetric Key Management* APIs provide an array of static key management functions, as well as the ability dynamically to create or import keys to the GDE Appliance.

Key Types, Algorithms, & Key States

The application developer, the GDE Administrator, and the VTS GUI Administrator work together to implement key handling in VTS.

Table 1: Key Summary Table

Key Type	Description	Algorithms	State
Symmetric	oct or octet sequence. Used for: • tokenize • detokenize • encrypt • decrypt Symmetric keys are also used for Tokenization, and while the symmetric keys are typical AES keys with 128 or 256 bits of key material, the algorithms are FPE (FF3) and FF1. **Note:** Versioned keys may not be used for tokenization, only non-versioned keys.	A128ECB - AES 128 ECB cipher mode A256ECB - AES 256 ECB cipher mode A128CBC - AES 128 CBC cipher mode A256CBC - AES 256 CBC cipher mode A128CBCPAD - AES 128 CBC-PAD cipher mode A256CBCPAD - AES 256 CBC-PAD cipher mode A128CTR - AES 128 CTR (counter) cipher mode A256CTR - AES 256 CTR (counter) cipher mode	At this time, only symmetric keys support states. On create and import, only a preactive or active state may be given. On a key update all states can be given except for preactive. The following states are available: • Preactive • Active - Default when a key is created • Suspended • Compromised • Deactivated • Destroyed
Asymmetric	RSA key type. Used for: • encrypt • decrypt • sign • verify	RSA	

Table 1: Key Summary Table (Continued)

Key Type	Description	Algorithms	State
Opaque object	octet array. This is a Thales extension (not part of JOSE RFC-7518 spec). Used for • sign • verify Does not have the option of store on server.	HS1 - HMAC-SHA-1 HS224 - HMAC-SHA-224 HS256 - HMAC-SHA-256 HS384 - HMAC-SHA-384 HS512 - HMAC-SHA-512	N/A

Static vs. Dynamic Key Management Creation

Static Work Flow

In most cases, it is recommended to implement your application using static keys— that is, keys that will change their key material rarely, and thus rarely be created or destroyed in the GDE Appliance.

The implementation tasks for static key creation are:

- Decide which actions will be performed (encrypt/decrypt, tokenize/detokenize, sign/verify) and which types of keys will be needed to support those actions.
- **GDE Administrator:** Create VTS-specific keys in the GDE Appliance, giving them meaningful names.
  NOTE: it is recommended to keep tokenization keys separate from other encryption/decryption keys.
- **VTS GUI Administrator:** Enter the appropriate key names into the VTS GUI, to be stored in the local VTS database and used for assigning encryption and tokenization permissions.
- **Application Developer:** Use the *Vormetric Key Management, Cryptography,* and *Tokenization* APIs to perform the necessary functions in the application being integrated with VTS.
  Do not use the REST API `create` or `import` key functions.
- It is recommended to use a static key model with tokenization.

Dynamic Work Flow

In some cases, an organization may decide to automate key creation, rotation, or key import in the GDE Appliance, using the appropriate *Vormetric Key Management* APIs.

Caution: If a key is deleted, all of the ciphertext the key was instrumental in creating is rendered unreadable.

In this case, the implementation team would need to:

- Decide which actions will be performed (encrypt/decrypt, tokenize/detokenize, sign/verify) and which types of keys will be needed to support those actions.
- **Application Developer:** Use the *Vormetric Key Management* and *Cryptography* APIs to perform the necessary functions in the application being integrated into VTS, including `create` and/or `import` key functions.
- **VTS GUI Administrator:** Designate a user (possibly a non-human user) and grant the user permission to create and/or import keys.

Tokenization Groups, Templates, & Masks: How Do They Fit In?

In tokenization, sensitive data— such as credit card numbers or social security numbers— are replaced with an encrypted “token” which retains the format of the original data. When detokenizing, different users can be authorized to view the entire detokenized/unencrypted value, or authorized to see only a part of the value, with the rest hidden by a *data mask*.
VTS includes tokenization groups, templates, and data masks as convenience features to make implementing tokenization easier.

What is a Tokenization Group?

A tokenization group is a way to compartmentalize VTS at a department level or site level. Each tokenization group has a symmetric key associated with it. While the key management and encryption services refer to keys directly, the `tokenize/detokenize` services refer to a key indirectly – via the tokenization group.

NOTE: See also "Manage Tokenization Groups" on page 57.

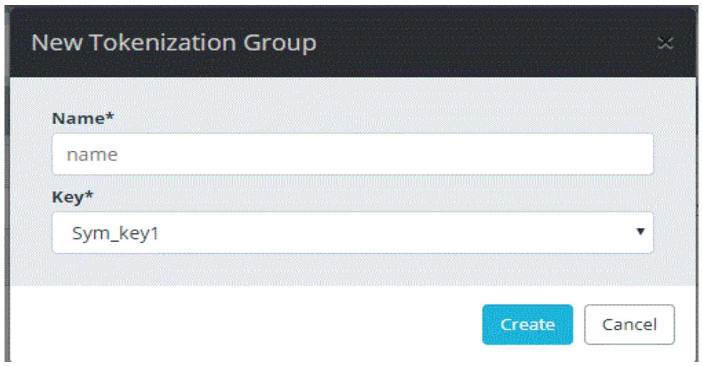


Figure 3: Have at least one symmetric key created when you create a tokenization group.

NOTE: Remember that all symmetric keys used for tokenization must be unversioned keys.

Tokenization Template

Tokenization templates are used to supply parameters to the tokenization/detokenization operation which otherwise would have to be supplied individually in the REST API call, thus making the REST API call both easier to use and shorter. The VTS GUI administrator collaborates with the application developer to create tokenization templates with the appropriate settings, and to give the template a name that developers will use in their code.

NOTE: See also "Manage Tokenization Templates" on page 62.

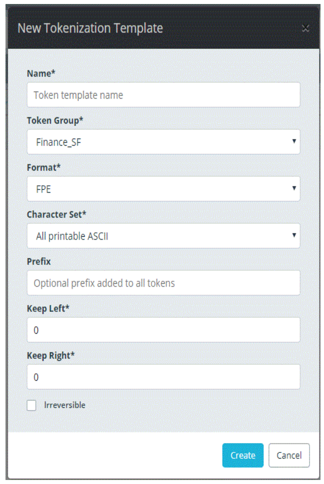


Figure 4: Create tokenization templates after creating tokenization groups and character sets.

Using Data Masks

When a user or user group is given detokenization privileges, a data mask must be applied that defines how much of the unencrypted data they are allowed to see. To create a mask that shows the entire data string, the VTS GUI Administrator would enter very large values in the **Show Left** and **Show Right** fields (999999).

NOTE: See also "Manage Data Masks" on page 64.

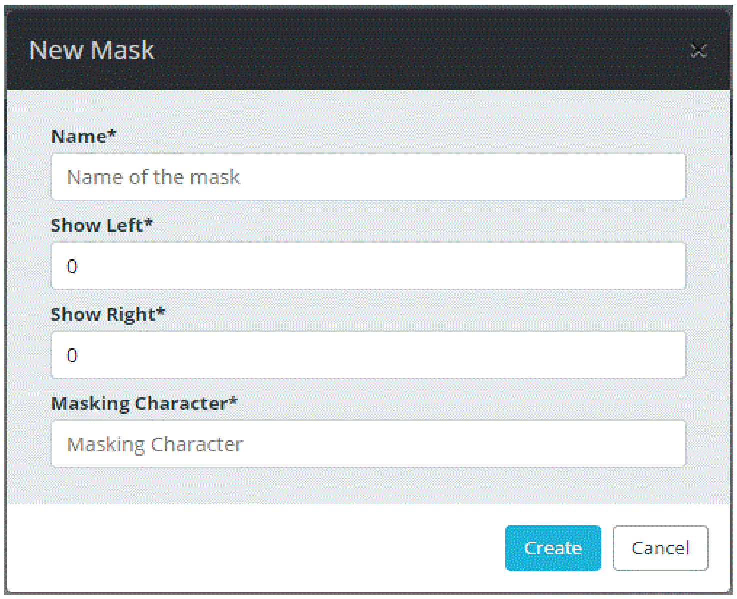


Figure 5: New data mask creation.

About the VTS API

VTS includes three sets of REST API services: Encryption, Key Management, and Tokenization.

Tokenization/detokenization are simple-to-use REST APIs which provide format-preserving encryption and proprietary "RANDOM" tokenization, whereas the Encryption APIs offer a plethora of traditional non-format-preserving encryption methods, as well as two format-preserving ones (FF3 and FF1)

Encryption and Key Management APIs are described in the chapter "Encryption & Key Management API Programming Notes" on page 71, and in an online reference guide. The Tokenization APIs (including the reference documentation, examples, and error codes) are described in the chapter "Tokenization REST API" on page 93.

Installing and Upgrading VTS

2

This chapter describes how to install VTS. It is used by the
VTS network administrator and contains the following sections:

Overview

The VTS solution consists of two required components and two optional components:

- *Vormetric Tokenization Server* (*VTS*)
- *Vormetric Data Security Manager (GDE Appliance)*,
- *Active Directory (AD)/LDAP server* (optional)
- *Remote Logging server* (optional)

This chapter focuses on installing the VTS, with additional sections about upgrade, backup, and recovery. Chapter 3, "Configuring the VTS System" on page 27, describes the follow-up configuration steps the network administrator performs to connect all the components and establish baseline settings.

VTS Basic Component Architecture

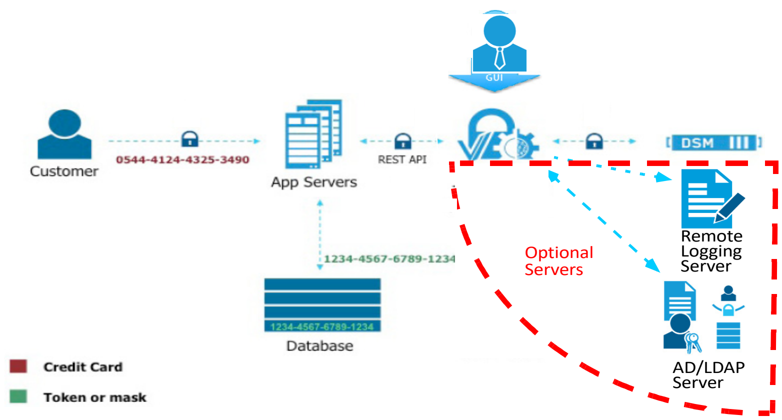
VTS can be installed as a virtual machine (the more typical case) or as software on bare metal.

The Vormetric Tokenization Server does most of the tokenization work and acts as a server for the additional cryptographic and key management API calls and features. It uses the other components for specific tasks. It obtains encryption keys from the GDE Appliance and it uses those encryption keys to encrypt sensitive data. For users created in the AD/LDAP server, the Tokenization Server uses the AD/LDAP server to validate those users when they request data to be tokenized, detokenized, encrypted, etc.. Users can also be created with the VTS GUI. These users are validated by the VTS itself.

Once the system has been installed and configured, the VTS supports the RESTful API calls from applications. Log messages can optionally be sent to a remote logging server.

IMPORTANT: We highly recommend setting up a production VTS in an environment with at least two nodes. This is described in "“Configure VTS Nodes using “Cluster” CLI Commands”" on page 27.

Figure 6: VTS component architecture



Installation Prerequisites

The VTS solution requires installing multiple components. The following sections describe each component and the information you must gather before integrating them into the Vormetric Tokenization solution. The integration steps themselves are described in

- “GDE Appliance Requirements” on page 19
- “Vormetric Tokenization Server Requirements” on page 19
- (Optional) “Active Directory/LDAP Server Requirements” on page 20
- (Optional) “Remote Logging Server” on page 21
- “Port Configuration” on page 22

GDE Appliance Requirements

The GDE Appliance provides the Vormetric Tokenization Server with encryption keys to encrypt the sensitive

For specific GDE Appliance version compatibility requirements, check the *Release Notes* for Vormetric Tokenization Server. Note that VTS does not require a dedicated GDE Appliance; it can be shared with other VDS applications. To install a new GDE Appliance, see the *Guardium Database Encryption (GDE) ApplianceInstallation and Configuration Guide*.

Gather this GDE Appliance information for system configuration:

1. GDE Appliance fully qualified hostname:
   ____
2. GDE Appliance IP address:
   ____

Vormetric Tokenization Server Requirements

Vormetric Tokenization Servers are virtual machines.

Requirements:

- VMware ESXi Server 5.5 or higher.
- Vormetric Tokenization Server `.ova` file. Examples: `vts-2.1.1.1649.ova` or `vts-2.1.1.1649.iso.` We recommend the following minimum virtual hardware configuration:
  Number of CPU cores: 4
  RAM: 16 GB minimum. 24 GB recommended if VTS will be used to tokenize credit card numbers longer than 16 digits.
  Hard Disk Space: 75 GB

Gather this Vormetric Tokenization Server information for system configuration:

1. Tokenization Server `.ova` image or :
   ____
2. Tokenization Server system hostname:
   ____
3. Tokenization Server IP address:
   ____

4. Tokenization Server subnet prefix length:
   ____

5. Tokenization Server default gateway:
   ____

6. DNS Server IP address: ____

NOTE: If you don't use DNS to resolve hostnames, you can modify the `/etc/hosts` files. See "Host name resolution without DNS" on page 31.

Active Directory/LDAP Server Requirements

An optional AD/LDAP server can be used to generate users and authenticate their credentials when they request services. See "Understanding VTS Authentication & Authorization" on page 4 and "AD/LDAP & User Authentication" on page 5 for a conceptual basis on how AD/LDAP is used. Requirements:

- Access to the AD/LDAP server containing users and groups that will have VTS privileges.
- For LDAPS, the LDAP server's CA certificate needs to be imported into the Tokenization Server.
- If you use the three example Active Directory groups (vtsUsers, vtsManager, and vtsSuperuser) below, create these groups before doing "Configure an AD/LDAP Server (optional)" on page 38.

Gather this AD/LDAP information for system configuration:

Table 2: AD/LDAP configuration information

Name	Description	Your Site Information
LDAP Server URI	Hostname, port and protocol of LDAP server. **Example**: ldaps://ads.example.com:636	
binding DN	FQDN of user that has access to the LDAP Server **Example**: vts0@ads.example.com	
binding password	The credential of above user **Example**: H838%i2hz9*%^86&^*	
user search scope	Search criteria of user object in AD/LDAP server **Example**: dc=corp,dc=example,dc=com	

Table 2: AD/LDAP configuration information (Continued)

Name	Description	Your Site Information
user group search scope	Search criteria of user group inside AD/LDAP **Example**: cn=Users,dc=corp,dc=example, dc=com	
user search filter	Name of user field in AD/LDAP **Example**: sAMAccountName for AD, uid for LDAP	
Tokenization Server active user group	Group that user can access Tokenization Server RESTful API **Example**: cn=vtsUsers,cn=User,dc=corp, dc=example, dc=com	
Tokenization Server super user group	Group that can access the GUI and REST APIs. **Example**: cn=vtsSuperuser,cn=User,dc=corp, dc=example, dc=com	
group member selector	UNIX/Linux LDAP only. The LDAP attribute which designates a group member within a group. **Example**: member or memberUID	
group object class	UNIX/Linux LDAP only. The LDAP attribute that identifies a group of users. **Example**: posixGroup or groupOfNames	
user prefix	UNIX/Linux LDAP only. The LDAP attribute to be used to prefix the username when searching for username in the LDAP database **Example**: uid, cn, sn	
CA Certificate	For LDAPS only: Access to root and intermediate CA certificates for the LDAPS server in .pem file format. May require a Certificate Management tool	

Remote Logging Server

Optionally, you can redirect log messages to a remote logging server. **Gather this remote logging server information for system configuration:**

1. Remote logging server name or IP address:____
2. Remote logging server port (TCP):____

Port Configuration

If the Vormetric Tokenization Server must communicate with other components through a firewall, open the ports in the firewall as shown in the following tables. Table 3 shows incoming traffic, and Table shows outgoing traffic.

NOTE: For a complete list of ports required for the GDE Appliance, see the *GDE Appliance Installation and Configuration Guide*.

Table 3 describes each port you must open for incoming communication to the VTS.

Table 3: Incoming Ports to Configure for Vormetric Tokenization Server

22	TCP	Management Console →VTS	Administration CLI SSH access.
443	TCP	Browser → VTS Requester → VTS CLI → VTS	HTTPS access for Administration GUI and REST/JSON API. Download the zip file to perform upgrade.
5432	TCP	VTS ↔ VTS	PostgreSQL Bi-Directional Replication (BDR) port for database replication across nodes.

Table 4 describes each VTS port you must open for outgoing communication from the VTS.

Table 4: Outgoing Ports to Configure for Vormetric Tokenization Server

User-defined	TCP	VTS →Log Server	Remote logging
123	UDP	VTS ↔ NTP Server	Optional network time synchronization
389	TCP	VTS → LDAP Server	Optional LDAP connection
636	TCP	VTS → LDAPS Server	Optional LDAPS (LDAP over TLS) connection
8080	TCP	VTS → GDE Appliance	Default TCP/IP port for HTTP that is used once to perform the initial certificate exchange between a VTS host and GDE Appliance.
8443	TCP	VTS → GDE Appliance	TCP/IP port through which the VTS communicates with the GDE Appliance to exchange configuration. The VTS establishes a secure connection to the GDE Appliance via certificate exchange using this port.
8444	TCP	VTS → GDE Appliance	Upload VTS log messages to GDE Appliance
8446	TCP	VTS → GDE Appliance	Configuration exchange (policy/key pull) using Elliptic Curve Cryptography (Suite B)

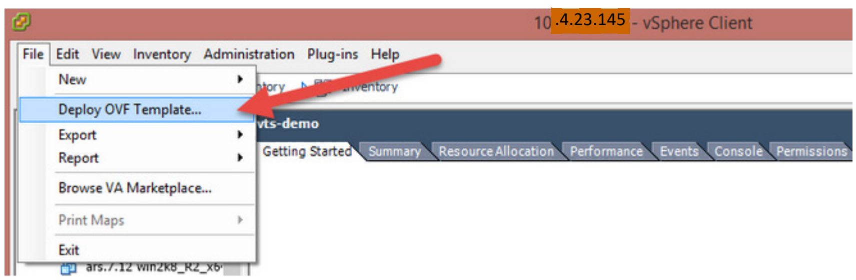
Table 4: Outgoing Ports to Configure for Vormetric Tokenization Server (Continued)

8447	TCP	VTS → GDE Appliance	VTS uploads log messages to GDE Appliance using Elliptic Curve Cryptography

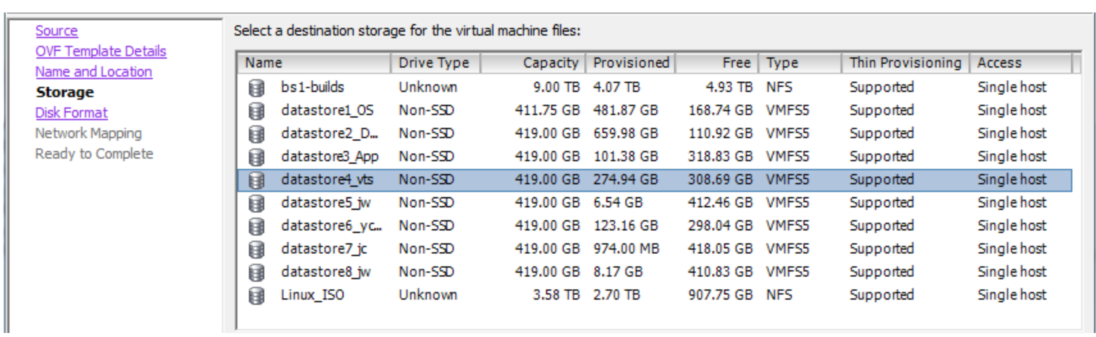
Install VTS as a Virtual Machine on VMWare vSphere

Vormetric Tokenization Servers are virtual machines running on a VMware ESXi server.

1. Open the VMware vSphere Client.
2. Click **File >Deploy OVF template**.

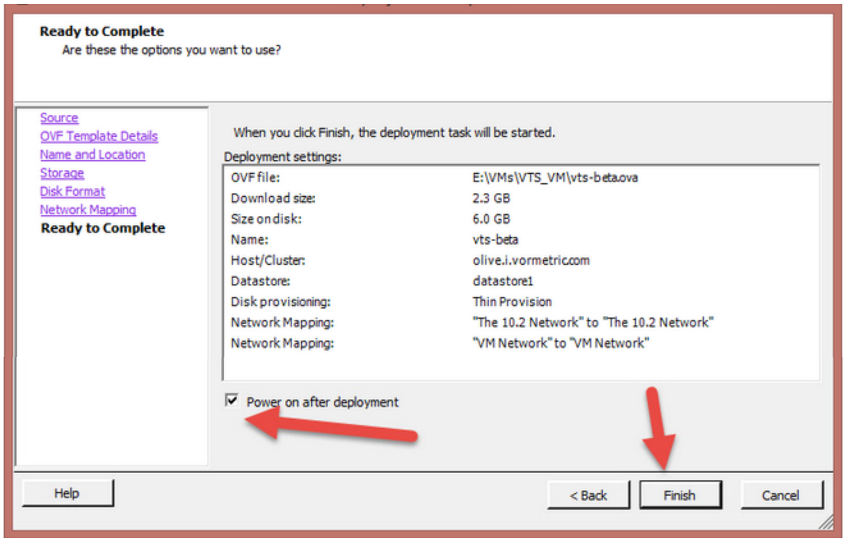


3. Click **Browse** and locate the OVF file (example:).
   Select the file and click **Next**. The *OVF Template Details* page appears.
4. Click **Next**. The *Name and Location* window opens.
5. Type in a name for the Tokenization Server and then click **Next**.
   The *Storage* window opens.



6. Select a destination storage profile for the virtual machine files. (We assume that the VM storage profiles have been defined and added to the datastore. See the vSphere documentation.) Click **Next**. The *Disk Format* window displays.
7. Select **Thin Provision** and **Next**. The *Network Mapping* window displays.

8. There are two NICs in the VM. Select the desired network map and click **Next**.
9. Check the box marked **Power on after deployment**.
   Click **Finish** to deploy the Virtual Appliance. This deployment can take a few minutes.



10. At the message **Completed Successfully**, click **Close**.
    The main screen of the vSphere Client appears.

At this point, the core installation is complete.

For a multi-node setup, you can repeat this installation process for additional machines. You would then configure each node separately and join them in a cluster as described in <xyz cluster>.

To begin configuration, click the **Console** tab of the vSphere Client to log in to the VTS CLI as `cliadmin`, change the default password, accept the Service Level Agreement, and begin configuring VTS settings using the VTS CLI (see Chapter).

Upgrade

This section gives generic instructions for how to upgrade the VTS. For specific instructions, refer to the release notes for the VTS version to which you are upgrading.

Notes for Upgrading a Cluster

- All nodes in the cluster must be upgraded to the same VTS version. A VTS cluster must not operate nodes using different VTS versions.
- When upgrading nodes in a cluster, all VTS operations must be suspended. If this is not possible, the user must perform a rolling upgrade. The nodes to be upgraded must be taken offline, for example taken out of the load balancer, before an upgrade is performed. When the load balancer is ready to use the upgraded nodes, all other nodes not yet upgraded must be taken offline. At any one time, the load balancer must not serve nodes running different VTS versions.
- While a node upgrade is in progress, it must not be interrupted. If it got interrupted or did not complete successfully, the user must discard the node by removing it from the cluster. Create a new VM, upgrade it to the desired version, and rejoin it to the cluster. When joining the node to the cluster, it must have the same version with all the nodes in the cluster.
  - All nodes in the cluster must be upgraded and must complete the upgrade before resuming operations.

Upgrade Steps

If you have multiple nodes in a cluster, you can upgrade all nodes at the same time.

1. Obtain the upgrade zip file from your Vormetric/Thales sales or support representative, and place the file on a server that is accessible from VTS over either HTTP or FTP.
2. Although the upgrade zip file is encrypted and digitally signed, the HTTP server can use SSL (HTTPS) and basic authentication. The HTTPS server can also use a self-signed certificate.
3. Log on as `cliadmin` to the VTS virtual machine that you want to upgrade. In the following example command, substitute your own VTS hostname.

```
$ ssh -l cliadmin <Tokenization Server hostname>
cliadmin@<Tokenization_Server_IP_address> password: <Enter the cliadmin
password>
0000:vormetric$
network      Networking configuration
system       System configuration
cluster      Cluster configuration
vae          VAE configuration
auth         LDAP configuration
security     Security configuration
remotelog    Remote logging configuration
maintenance  System maintenance utilities
exit         Exit
```

4. Navigate to the *maintenance* menu. Type:

```
0000:vormetric$ maintenance
```

5. Navigate to the *vts* menu. Type:

```
0001:maintenance$ vts
VTS Configuration
vts>
```

6. Upgrade the VTS software with the `upgrade` command. In the example command below, substitute the location of your own upgrade zip file.

```
vts> upgrade <Upgrade Package URL>
```

Examples:

```
upgrade https://example.com/vts-upgrade-<version>.zip
upgrade https://username:password@example.com/vts-upgrade-<version>.zip
upgrade ftp://example.com/vts-upgrade-<version>.zip
```

Watch the output of the `upgrade` command while the upgrade is underway. After several progress messages, you should see a success message like the following:

```
Upgrade complete.
```

Configuring the VTS System

3

This chapter describes how the network administrator uses the VTS CLI to configure the VTS environment, establish connections between the various components, and other system-level tasks. It contains the following sections:

Access the CLI

The network administrator uses the VTS CLI to configure the VTS environment.

1. Access the CLI through a terminal emulator like Putty.
2. On first access, enter the default login and password:
   login: `cliadmin`
   Password: `cliadmin123`
3. On first access, you will be prompted to change the password.
   Save and store your new password securely.
4. On first access, you will be prompted to review and sign the License Agreement.

Table 5: CLI navigation Cheatsheet

Navigation	Command	Example
Starting prompt	`vormetric$`	

Table 5: CLI navigation Cheatsheet (Continued)

Navigation	Command	Example
List CLI categories or the commands within categories	`?`	vormetric$ **?** **network** Networking configuration **system** System configuration **cluster** Cluster configuration **vae** VAE configuration **auth** LDAP configuration **security** Security configuration **remotelog** Remote logging configuration **maintenance** System maintenance utilities **exit** Exit
Open a category and display its commands	`<category name> ?`	vormetric$ **network** network$ **?** **ip** Interface configuration **dns** Set the DNS servers **host** Configure /etc/hosts file **ping** Ping ip address or hostname **traceroute** Trace route to ip or hostname **rping** Send ARP request to neighbor host **arp** Manipulate the system ARP **checkport** Check opening port with nc **up** Return to previous menu **exit** Exit
Get help: See command usage and example output	`<enter the command without a value>`	0003:network$ **ping** Usage: ping IPADDRESS 0004:network$ checkport Usage: checkport HOST_NAME PORT [timeout {1..600}] 0005:network$ **login** Usage: login username
Have CLI complete a category name, command, or argument when typing	Press `<tab>`	Enter enough characters to uniquely identify a category, command, or argument, and then press the **<Tab>** key. The VTS CLI will complete it for you.
Return to the main level	`up`	
End the CLI session	`exit`	

NOTE: You must enter a category to execute any of its commands. For example, the `reboot` command is in the `system` category, so you would enter `system`, then enter `reboot`.

The configuration steps in the following sections employ a subset of the CLI commands.
See “VTS CLI Reference” on page 109 for the full CLI command reference.

Basic Configuration Steps

NOTE: For multi-node installations, the basic configuration steps in this section must be completed on each node before you "Configure VTS Nodes using "Cluster" CLI Commands" as described on page -34.

Set the VTS Network Settings

Network settings are required only for VMware and bare metal installations, and only if you decide to use a static IP address. Do not attempt to set the IP address and gateway of the network interface if you are using VTS in an Azure or Amazon cloud.

Configure the VTS network settings:

- "Set the IP address for the VTS virtual machine."
- "Assign the default gateway."
- "Assign the DNS server," or use "Host name resolution without DNS."

Check the annotations you made under "Vormetric Tokenization Server Requirements" on page 19 for the IP, gateway, and DNS values you will need.

Set the IP address for the VTS virtual machine

1. Using the VTS CLI, navigate to the *network* commands menu.

```
0000:vormetric$ network
0001:network$
```

2. Set the IP address for VTS virtual machine. There are two ethernet cards, eth0 and eth1. Assign an IP address to eth0 with the following command:

```
0001:network$ ip address set <Tokenization Server IP address>/<address prefix length>
dev eth0
```

NOTE: 1) The address prefix length corresponds to the network class to which this host can connect. For class A use 8, for class B use 16, for class C use 24.
2) If you are connected via eth0, you may be disconnected at this step. You will be prompted to reconnect to the new IP address. Enter `Yes`.

3. Verify the interface settings. Type:

```
0002:network$ ip address show
```

Assign the default gateway

1. Add the IP address for the default gateway:

```
0003:network$ ip route add default via <IP address for the default
gateway, ex: 192.168.0.1> dev eth0
```

2. Verify the route settings. Type:

```
0004:network$ ip route show
Main routing table
192.168.0.0/16 dev eth0  proto kernel  scope link  src 192.168.20.11
192.168.0.0/16 dev eth1  proto kernel  scope link  src 192.168.99.122
169.254.0.0/16 dev eth0  scope link  metric 1002
169.254.0.0/16 dev eth1  scope link  metric 1003
default via 192.168.0.1 dev eth0
ip route show SUCCESS
```

Assign the DNS server

If you are not resolving hostnames with a DNS server, see "Host name resolution without DNS" on page 31.

1.If you are using DNS, set the primary DNS server for the Vormetric Tokenization Server. Type:

```
0005:network$ dns dns1 <IP address for dns1>
```

2. If you have a second or third DNS server, set them for the Vormetric Tokenization Server. Type:

```
0006:network$ dns dns2 <IP address for dns2>
```

3. If you want to set the search domain, type :

```
0007:network$ dns search <search_domain>
```

4. Show the DNS settings. Type:

```
0008:network$ dns show
search i.vormetric.com

DNS show SUCCESS
```

Host name resolution without DNS

DNS is one method of host name resolution. If you do NOT use a DNS server to resolve host names, do one the following:

- **Modify the `hosts` file on the Vormetric Tokenization Server.** Include other Vormetric Tokenization Servers, the AD/LDAP server, the GDE Appliance, and remote syslog server. Use names like *serverx.domain.com,* enter the host names and matching IP addresses in the `/etc/hosts` file on the Vormetric Tokenization Servers using the `host` command under the `network` menu of the VTS CLI. For example:

```
0011:network$SUCCESS: add host
0012:network$ host show
name=localhost6.localdomain6

SUCCESS: show host
```

You must do this on *each* VTS, since entries in the `hosts` file are not replicated across Vormetric Tokenization Servers.

OR

- **Use IP addresses:** You may use IP addresses or the FQDN to identify the hosts--except for the GDE Appliance, which must be identified by its FQDN.

NOTE: Network settings are not applicable for Azure VMs.

Set the VTS hostname

Even though you have set a VTS host name when you created the virtual machine, you must also set a host name in the VTS system using the CLI. This is used by the GDE Appliance when you register the host there.

1. In the VTS CLI, navigate to the *system commands* menu. Type:

```
0010:vormetric$ system
```

2. Set the VTS hostname with the `setinfo` command:

```
setinfo hostname <Fully Qualified Host Name>
```

Example:

```
0011:system$ setinfo hostname vts-5.vormetric.com
SUCCESS: setinfo hostname. If the Security Server certificate is
already generated, please re-sign the server certificate to reflect
hostname changes.
```

NOTE: If you change the VTS hostname, either generate a new self-signed server certificate or import a server certificate (see below).

Import a server certificate for TLS

By default, the system automatically uses a self-signed certificate for HTTPS. You must regenerate the self-signed certificate (for a test or demo installation) or install a 3rd party certificate from a CA (for production) so that others know you are verified.

Option 1: Regenerate a self-signed server certificate

To regenerate the self-signed certificate run:

```
0012:security$ server-cert createcertss
```

Enter answers to the informational questions and restart the web server when asked.

NOTE: If your server requires a particular TLS protocol, you can set it following the instructions in xyz.

Option 2: Import authenticated third-party server cert (recommended)

To import an authenticated 3rd-party CA certificate to the VTS, follow the steps below.

Use an SSH terminal like **xterm** or **PuTTY** to do this.

1. On the tokenization CLI, go the *Security* category and enter `server-cert generatecsr` to generate a certificate request:

   ```
   0013:security$ server-cert generatecsr
   ```

2. Follow the system prompts. A certificate request displays on the secure shell:

   ```
   ------ BEGIN CERTIFICATE REQUEST -------
   . . .

   ------ END CERTIFICATE REQUEST ------
   ```

3. Copy and paste the certificate request including the "REQUEST" lines onto a third-party certificate request form, such as GoDaddy, Thwart, or Verisign.
4. After getting the certificate, run `security importcert` on the CLI command line:

   ```
   0014:security$ server importcert
   ```

   You will be prompted to continue. Type `yes`.
5. Copy and paste the entire certificate including the CERTIFICATE lines shown below.

   ```
   -----BEGIN CERTIFICATE-----
   . . .
   -----END CERTIFICATE-----
   ```

6. Type "Ctrl-d" to end the input.
7. The VTS will import the certificate and try to verify that the imported certificate matches the private key generated at the time the certificate was generated.
8. The system will prompt you to restart the web server. Type `yes`.
9. The web server component of the VTS will be restarted and the new certificate will take effect. The system is set up using the third-party server certificate for TLS connections.

Register VTS with the GDE Appliance (mandatory)

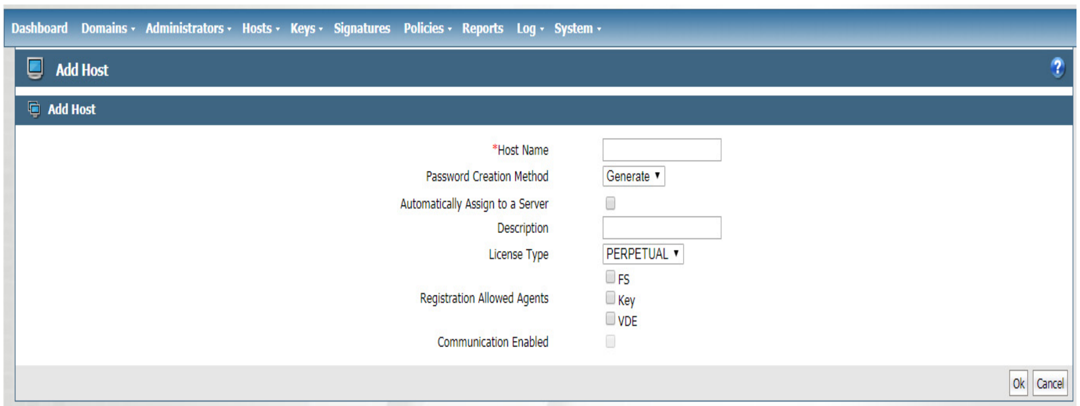
See "GDE Appliance Requirements" on page 19 for a list of the prerequisites before configuring.

1. If there is no DNS, add the hostname of your GDE Appliance into `/etc/hosts`. Go to the `network` submenu in the VTS CLI and enter: `host add <dsm hostname> <dsm IP Address>`

   ```
   0012:network$ host add dsm1 192.168.34.12
   SUCCESS: add host
   ```

2. Ask the GDE Appliance Administrator to add the VTS hostname to GDE Appliance database as follows.

On the GDE Appliance Management Console click **Hosts > Add**. Enter:
Host Name: See "Set the VTS hostname".
Agent type: Key
License type: Set as per your GDE Appliance licensing agreement
Communication Enabled: Checked
All other parameters can be left as is. Click **OK**.



3. In the GDE Appliance Management Console, confirm that settings are correct. Click on the host name to go to the *Edit Host* page. Under the *Key Agent* column, check the **Communication Enabled** and **Registration Allowed** check boxes, and click **Ok**.

4. Return to the VTS CLI and register the VTS with the GDE Appliance.
Obtain the GDE Appliance hostname from the *GDE Management Console> Dashboard> Server Name*.
Then go to the `vae` category of the CLI and run:

```
0012:network$ up
0013:vormetric$ vae
0014:vae$ register <dsm hostname>
```

Configure VTS Nodes using "Cluster" CLI Commands

A VTS image can be deployed once, as a stand-alone instance or node, and used for demo or test purposes. In a production environment, it is recommended to repeat the installation procedure on multiple nodes. Perform the "Basic Configuration Steps" on each node.
You then configure the nodes to form a cluster.

The following instructions use a hypothetical scenario where three images were freshly installed, and describe the mandatory steps to:

“On Instance 1”

- “Create node1”
  - The first time the database is created, you are also prompted to “Create VTS GUI Admin credentials”.
- “Add nodes 2 and 3”

“On Instance 2”

- “Add nodes 1 and 3”
- “Join the cluster”

“On Instance 3:”

- “Add nodes 1 and 2”
- “Join the cluster”

Upon completion, all three nodes in the cluster are interconnected and the three databases are replicated.

NOTE: All the commands to accomplish these steps are grouped under the `cluster` category in the VTS CLI. The `cluster$ create <node IP address>` actually creates and configures the initial database. Therefore, even for a standalone, single-instance VTS installation, at least one “cluster” step is required.

If your environment already has one VTS VM installed and configured, and the database has already been created, then skip ahead to adding nodes and joining the cluster.
See “Adding Nodes to an Existing Cluster” on page 38.

Before you begin:

Have the IP addresses of each VTS node on hand.

VTS supports up to four nodes.

On Instance 1

Create node1

1. "Access the CLI through a terminal emulator like Putty." and enter `cluster` to access the cluster category of commands.
2. Enter `cluster $create <node IP address>`
   This will launch the process of creating the first VTS database, along with other VTS configuration steps behind the scenes.

```
0015:cluster$ create <node1_IP_address>
Stopping postgresql-9.4 service: [ OK ]
Saving config files
Initializing database ... OK
. . .
```

Create VTS GUI Admin credentials

The first time you create the VTS database, you will be prompted to create a superuser (GUI Administrator) and assign a password to that administrator.

1. When prompted to create a superuser, enter `yes`.
2. Assign a name for the GUI Administrator. Leave blank to use `'root'`.
3. Enter and confirm a password. **Save and store securely**!

```
You have installed Django's auth system, and don't have any superusers
defined. Would you like to create one now? (yes/no): yes
Username (leave blank to use 'root'): vtsroot   (login name for VTS GUI)
Email address:
Password:   (password for VTS GUI)
Password (again):   (password for VTS GUI)
Superuser created successfully.
Pre-populating vts database...
. . .
Create cluster SUCCESS.
```

Access the VTS GUI

Test the node by accessing the VTS GUI: **`https://node_IP_address/admin`**

Next Steps:

For a single-node installation, you have now completed the required configuration steps and can proceed to "Advanced Configuration Steps" on page 38, as needed for your environment. Finish with "Restart the Vormetric Tokenization Server" on page 46.

For a multi-node installation, continue the cluster configuration steps below.

Add nodes 2 and 3

In the VTS CLI, enter `cluster$ add_node` with the IP addresses of nodes 2 and 3.

```
0015:cluster$ add_node node2_IP_address
0016:cluster$ add_node node3_IP_address
```

On Instance 2

Once the `cluster$ create` command has been used on one node, run `cluster$ add_node` and `cluster$ join` on each additional instance.

Add nodes 1 and 3

In the VTS CLI, enter `cluster$ add_node` with the IP addresses of nodes 1 and 3.

```
cluster$ add_node node1_IP_address
cluster$ add_node node3_IP_address
```

Join the cluster

1. In the VTS CLI, enter `cluster$_join` with the IP addresses of node 2 and node 1.

```
cluster$ join <node2_IP_address> <node1_IP_address>
```

2. "Access the VTS GUI" to test the node.

On Instance 3:

Add nodes 1 and 2

In the VTS CLI, enter `cluster$ add_node` with the IP addresses of nodes 1 and 2.

```
cluster$ add_node node1_IP_address
cluster$ add_node node2_IP_address
```

Join the cluster

1. In the VTS CLI, enter `cluster$_join` with the IP addresses of node 3and node 1.

```
cluster$ join <node3_IP_address> <node1_IP_address>
```

2. "Access the VTS GUI" to test the node.

For a 3-node installation, you have now completed the required configuration steps and can proceed to "Advanced Configuration Steps" on page 38, as needed for your environment.
Finish with "Restart the Vormetric Tokenization Server" on page 46.

VTS supports up to four nodes.

Adding Nodes to an Existing Cluster

When a new node is added to an existing cluster, user needs to do the following:

1. On each node of the existing cluster, add `<new_node_IP>`
2. On the new node, add IP addresses of all the nodes respectively
3. Run join `<new_node_IP>` `<any_node_IP_in_cluster>`

IMPORTANT: All cluster must run the add_node command for the new node to be added. If one node is missed, the join command will hang and the replication will not complete.

Advanced Configuration Steps

The procedures in this section are optional, depending on your organization's needs and existing infrastructure.

Configure an AD/LDAP Server (optional)

The Active Directory/LDAP server is optional. However, you can use it to import VTS users and VTS GUI Administrators. See "Active Directory/LDAP Server Requirements" on page 20 before configuring. You must configure the AD/LDAP server on every node in your VTS HA cluster.

1. Go to the `auth` submenu in the VTS CLI.

```
vormetric$ auth
auth$ ?
host          Setup LDAP Server IP
bind          LDAP User & Credential
user_scope    LDAP User Search Scope
group_scope   LDAP Group Search Scope
user_filter   LDAP User Search Filter
group_member  LDAP Group Member Selector
active_user   LDAP Active User Group
super_user    LDAP Super User Group
ca-cert       CA Certificate for built-in LDAPS client
enable        Enable LDAP support
disable       Disable LDAP support
show          Show db information
```

2. Like all CLI commands, you can type the command without a parameter and a usage example displays:

```
auth$ host
usage: host ldap_server_url, e.g. ldap://192.168.118.99:389
auth$ group_scope
usage: group_scope group_search_scope. e.g. CN=users,DC=vts,
DC=vormetric,DC=com
```

3. Disable, then enable LDAP support in the Tokenization Server:

```
auth$ disable
Note: You choose to disable the LDAP support in VTS, the existing LDAP
parameters will be saved for future use. Unless, You like to also clear
them out.

Please confirm that you like to proceed to clear out LDAP configuration
(yes|no)[no]:yes
auth$ enable
```

If you disable LDAP support, then you can use only the VTS GUI to create users and groups.

4. Set the AD/LDAP server URI with the `host` command:

```
host ldaps://<hostname_or_IP_address>:<Port. Default: 636>
```

or

```
host ldap://<hostname_or_IP_address>:<Port. Default: 389>
```

Example:

```
0015:auth$ host ldap://192.168.118.99:389
Set LDAP Server URI SUCCESS
```

5. Set the binding DN of the user who has access to the AD/LDAP server with password. Example:

   `auth$ bind <FQDN of user with access to LDAP Server.>`

   Example:

```
0016:auth$ bind vts0@vts.vormetric.com
Enter Password :
Enter Password again :
Set LDAP Bind User DN SUCCESS
```

6. Set user search scope using `user_scope` command:

   `user_scope <User search scope>`

   Example:

```
0017:auth$ user_scope dc=vts,dc=vormetric,dc=com
Set LDAP USER SEARCH SCOPE SUCCESS
```

7. Set group search scope using `group_scope` command:

   `group_scope <Group search scope>`

   Example:

```
0018:auth$ group_scope CN=Users, dc=vts,dc=vormetric,dc=com
Set LDAP GROUP SEARCH SCOPE SUCCESS
```

8. Set attribute field for user search filter using `user_filter` command:

   `user_filter <Name of user field in AD/LDAP>`

   Example:

```
0019:auth$ user_filter sAMAccountName
Set LDAP USER SEARCH FILTER SUCCESS
```

9. Set user prefix name using `user_prefix` command. The user prefix is the attribute which should be used to prefix the user name when searching for the user name in the LDAP database.

   `user_prefix <User prefix name. Example: uid or cn or sn>`

Example:

```
0019:auth$ user_prefix uid
Set LDAP USER PREFIX SUCCESS
```

10. Set an LDAP group member filter using `group_member` command. This configuration only affects the search for a Unix/Linux LDAP Server. It is not applicable to a Windows Active Directory Server.

`group_member` <***Group member selector. Attribute: "member" (fullDN) or "memberUid" (uid only)***>

Example:

```
0019:auth$ group_member membUid
Set LDAP GROUP MEMBER SELECTOR SUCCESS
```

11. Set Group Object Class name using `group_objectclass` command. The group object class is the attribute which identifies a group of users.

`group_objectclass` <***Group object class. Attribute: posixGroup or groupOfNames***>

Example:

```
0019:auth$ group_objectclass posixGroup
Set LDAP GROUP OBJECTCLASS SUCCESS
```

12. Set Tokenization Server active user group using `active_user` command:

```
active_user <Group whose members can access the RESTful API. Example:
vtsUsers>
```

In this example, any user who wants access to the VTS RESTful API must belong to the AD group *vtsUsers* within *Users* of the domain *vts.vormetric.com*:

```
0020:auth$ active_user cn=vtsUsers,cn=Users,dc=vts,dc=vormetric,dc=com
Set LDAP ACTIVE USER GROUP SUCCESS
```

13. Set VTS super user group using `super_user` command:

`super_user` <***Group that can access the Tokenization Server GUI & APIs. Example: vtsSuperuser***>

In this example, any user who requires access to the VTS GUI and REST APIs must belong to the AD group *vtsSuperuser* within *Users* of domain *vts.vormetric.com*:

```
0021:auth$ super_user cn=vtsSuperuser,cn=Users,dc=vts,dc=vormetric,
dc=com
Set LDAP SUPERUSER USER GROUP SUCCESS
```

14. *For LDAPS support only*: Import the CA certificate chain of the LDAP server to the built-in LDAPS client of the VTS. A Certificate Management tool like DigiCert or Certificate Manager tool is recommended and VTS Administration CLI command `ca-cert` to import the certificate into the VTS.

    a. Using a Certificate Management tool like DigiCert or Certificate Manager tool, download your intermediate and root certificates.

    b. Run `ca-cert import` to import the certificate trust chain. Paste the entire body of each certificate in the order shown in the example, then terminate with `Ctrl-D` when finished.

    Example:

```
            0022:auth$ ca-cert import
            Please confirm that you would like to proceed to import the
            LDAP Server CA certificate? (yes|no):yes
            Please paste content of the certificate and press CTRL-D
            when finished.
            -----BEGIN CERTIFICATE-----
            (Your Intermediate certificate: DigiCertCA.crt)
            -----END CERTIFICATE-----
            -----BEGIN CERTIFICATE-----
            (Your Root certificate: TrustedRoot.crt)
            -----END CERTIFICATE-----
Saving LDAP Server CA Certificate succeed.
Validating CA certificate ....
/tmp/cacert.pem: c = US, ST = CA, L = San Jose, O = Vormetric Inc., OU
= organizationalunit, CN = vts-qa, emailAddress = edleup@vormetric.com
error 18 at 0 depth lookup: self signed certificate
OK
Certificate validation succeed, Import LDAP Server CA Certificate
SUCCESS.
```

15. Execute the `show` command to list the LDAP configuration:

```
0023:auth$ show
LDAP Enable: true
LDAP Server URI:  ldap://192.168.118.99:389
LDAP BIND DN: vts0@vts.vormetric.com
LDAP BIND Password: kj43221
LDAP USER SEARCH SCOPE: dc=vts,dc=vormetric,dc=com
LDAP GROUP SEARCH SCOPE: CN=Users,dc=vts,dc=vormetric,dc=com
LDAP USER SEARCH FILTER: sAMAccountName
LDAP ACTIVE USER GROUP: cn=vtsUsers,cn=Users,dc=vts,dc=vormetric,dc=com
LDAP SUPER USER GROUP:
Show LDAP Configuration SUCCESS
```

16. Restart the web server component of the Tokenization Server:

```
0024:auth$ up
0025:vormetric$ system
0026:system$ server restart
Do you want to restart the server software ? (yes|no)[no]:yes
Restarting now...
SUCCESS: The tokenization server is restarted.
```

Enable the Client Authentication Feature (optional)

In addition to Basic Auth (username and password), the VTS server can authenticate clients using client certificates. To enable, client-certificate authentication must be enabled on every node of the VTS cluster. Perform the procedure below on each node.

NOTE: See also: “Client Certificates & User Authentication” on page 6 and “Enable Client Authentication” on page 156.

Obtain Client CA Key and Certificate

1. Log in to the VTS CLI.
2. Go to the `security` menu and run: `client-certificate upload-ca-cert`

3. At the prompt, paste the text of the certificate. For example:

```
vormetric$ security
security$ client-certificate upload-ca-cert
Please paste content of certificate and press CTRL-D when finished.
-----BEGIN CERTIFICATE-----
MIIDrTCCAxagAwIBAgIBADANBgkqhkiG9w0BAQQFADCBnDEbMBkGA1UEChMSVGhl
...
-----END CERTIFICATE-----
```

4. Verify the uploaded certificate by using the `client-certificate show` command.

```
security$ client-certificate show
```

Enable client cert authentication with or without identities

Client certificate authentication has two modes:

- Enable without identities
- Enable with identities

To enable without identities, run:

```
0003:security$ client-certificate enable-without-id
```

To enable with identities, run:

```
0004:security$ client-certificate enable-with-id
```

NOTE: VTS GUI administrator will need to enter the common name (CN) from the cert as a user name. See “Manage VTS Users” on page 51.

To disable client certificate authentication, run:

```
0005:security$ client-certificate disable
```

To check if client certificate authentication is enabled or not, run:

```
0006:security$ client-certificate status
```

Sample use in code

To use client-certificate authentication, include the client key and CA certificate when running the REST API.

Example:

```
$ curl --tlsv1.2 -k --key client.key --cert client.crt \
-X POST -u username:password \
-d'{"tokengroup" : "t1" , "data" : "9453677629008564",
"tokentemplate" : "Credit Card" }' \
https://192.168.12.88/vts/rest/v2.0/tokenize
```

Specify the TLS protocol version (optional)

Clients may require different versions of TLS protocol. You may specify what version of SSLTLS is supported by VTS. If you opt in to this feature, you must specify the TLS version on every node in your VTS HA cluster.

To do this, log on to the VTS CLI and run:

```
security >ssl-protocol [ tls1.2 | tls1.1 |tls1.0 ]
```

where the user specifies the lowest protocol version allowed. TLS versions above the specified version are also allowed:

`tlsv1.0` supports tlsv1.0, tlsv1.1, and tlsv1.2
`tlsv1.1` supports tlsv1.1, and tlsv1.2
`tlsv1.2` supports tlsv1.2

Example:

```
0035:vormetric:$ security
0036:security$ ssl-protocol tlsv1.2
0037:security$ show
SLL Protocols: TLSv1.1 TLSv1.2
. . .
```

Redirect log messages to a remote log server (optional)

Optionally, you can redirect log messages to a remote logging server. If you opt in to this feature, it is recommended that you redirect log messages off every node in your VTS HA cluster to the remote log server. You can also add SIEM software (example: Splunk, QRadar) for logging. See the *VDS Security Intelligence User Guide.*

1. Go to the `remotelog` category in the VTS CLI:

```
0034:vormetric$ remotelog
remote  Enable/Disable remote logging
host    Set remote rsyslogd server and port
show    Show logging information
up      Return to previous menu
exit    Exit
```

2. Show the current remote log settings:

```
0035:remotelog$ show
remote Logger Enabled: : false
Show Logger Configuration SUCCESS
```

3. To enable or disable the remote logging re-direct, use the `remote` command:

```
0036:remotelog$ remote
usage: remote [enable/disable]
0037:remotelog$ remote enable
Successfully enabled remote syslog.
Shutting down system logger:                        [  OK  ]
Starting system logger:                             [  OK  ]
```

4. Set the remote log server hostname and port, use `host` command:

```
0038:remotelog$ host
usage: host [HOSTNAME/IP] port]
0039:remotelog$ host 192.168.33.51 10514
Successfully updated rsyslog configuration.
Shutting down system logger:                        [  OK  ]
Starting system logger:                             [  OK  ]
```

You do not need to restart the VTS after changing the remote log server configuration.

NOTE: The current log level is set to 'INFO' level for operations and 'DEBUG' level for other packages.

Restart the Vormetric Tokenization Server

After all CLI configuration is completed:

1. Navigate to the *system commands* menu:

```
0010:vormetric$ system
0011:system $ ?
setinfo   Set system information
security  configure security system now
reboot    Reboot the system now
server    Manage tokenization server
up        Return to previous menu
exit      Exit
```

2. Restart the web server component of the VTS:

```
0012:system$ server
usage:   server [ restart | start | stop | status ]
0013:system $ server restart
Do you want to restart the server software? (yes|no)[no]:yes
Restarting now...
SUCCESS: The tokenization server is restarted.
```

VTS Administration in the GUI

4

This chapter contains the following sections:

Log in to the VTS GUI

The VTS GUI is used to create tokenization users/groups, specify user/group permissions, create Token Templates and create Token Groups. Log on to the VTS Administrative GUI as follows:

1. Point your browser to the Tokenization Server:

```
https://<VTS name or IP address>/admin/
```

2. Log in using the Admin user name and password you defined when you created the VTS cluster (see "Create VTS GUI Admin credentials" on page 36).

3. The VTS GUI Home page is displayed (Figure 7).

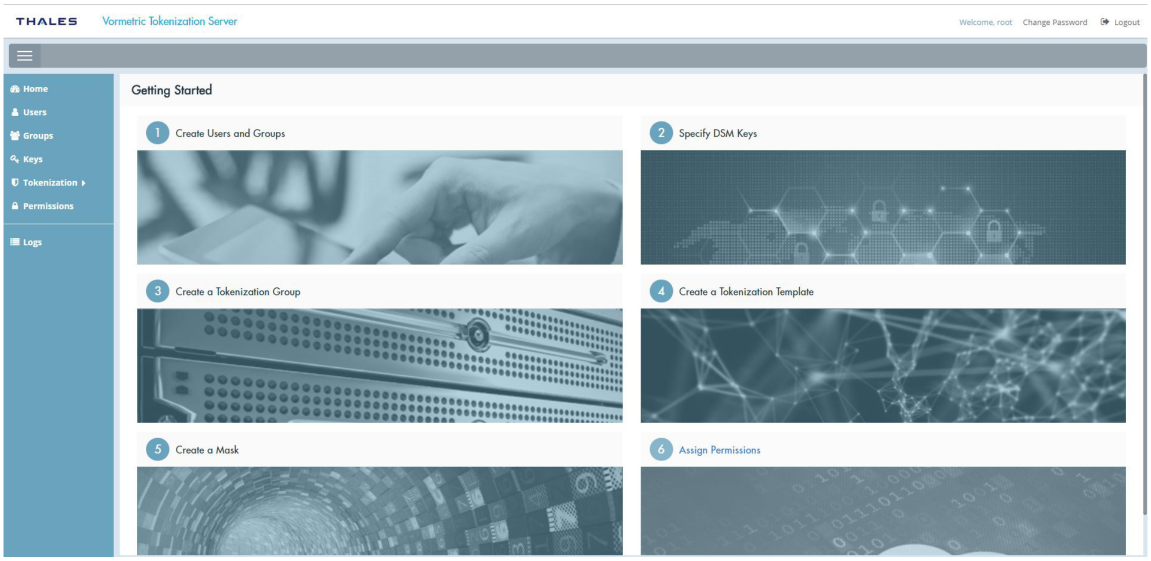


Figure 7: VTS Home page.

Troubleshooting: Tokenization Server Administration GUI login failure

- **nginx 504 Gateway Time-out** errors displays when I try to log on.
  - Problem: The Tokenization Server is not connected to the LDAP server.
  - Solution: Reboot the Tokenization Server. Open the VTS CLI. Execute the following commands: **Auth > disable**. Then **System > reboot**.

Manage VTS User Groups

Review the following sections to familiarize yourself with user groups in VTS:

- "VTS roles "Superuser," "Staff," and "Active"" on page 7
- "Understanding VTS User Groups & Permissions" on page 10

Note that you will want at least three user groups, to correspond to the VTS roles "Superuser," "Staff," and "Active."

If using AD/LDAP, user groups must be defined on the LDAP system and the user group names must be entered into the VTS GUI before importing users.

Create a User Group

To create a user group:

1. “Log in to the VTS GUI” and select **User Groups** from the navigation bar.
   The All Groups list is displayed.

Figure 8: The All Groups list.

2. Select **New Group** and enter a name on the resulting page. Click **Create**.
   The new group name will appear in the All Groups list and will be selectable from User pages and the Groups Permissions page.

Delete a User Group

If you delete a user group to which users and/or permissions have been associated, the User Group will be removed from any User belonging to it, and will delete any permissions assigned to it.
“Log in to the VTS GUI” and select **Groups** from the navigation bar (Figure 8).

3. Select the check box by a group, click **Delete**, and confirm Delete.

Manage VTS Users

Review the following sections to familiarize yourself with users in VTS:

- “What is a VTS “user?”” on page 4
- “VTS Authentication Basics” on page 5
- “VTS Authorization Basics” on page 7

You can “Create a User Manually” or “Import a User From an AD/LDAP Server”.

We recommend creating at least one user in the GUI to get a feel for how it is done.

Create a User Manually

Create Basic User Information

1. “Log in to the VTS GUI” and select **Users** from the navigation bar.
   The list of All Users is displayed.

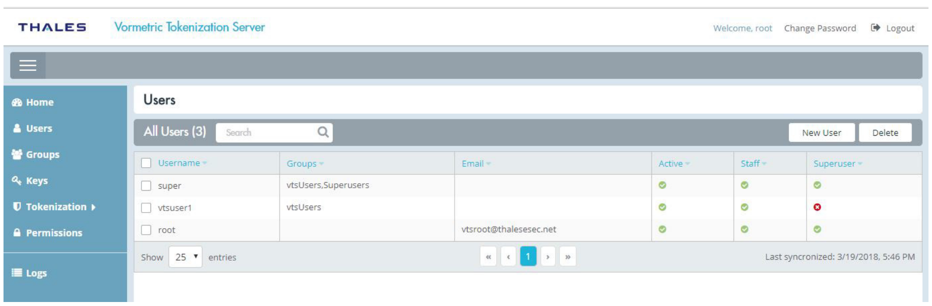


Figure 9: List of all users

2. Select **New User** and enter the following attributes:
   - **User Name:** required
   - **Password:** required.
     Note: if you are creating a user based on the common name in a client certificate, the password will be ignored by the system but must still be entered here.
   - **Email:** optional
   - **VTS Role:** optional. Select **Superuser** and/or **Staff**.
     “**Active**” will be automatically enabled when the user is created, and can be disabled when you “Update User Information”.
     See “VTS roles “Superuser,” “Staff,” and “Active”” for details.
3. Click **Create**.
   The user will be displayed as available in the All Users list and the Permissions matrix. Complete the user profile when you “Update User Information”.

NOTE: User names and passwords are case-sensitive. Application code that calls the VTS RESTful APIs must pass user names and passwords in the same case that are defined in the VTS GUI.

Update User Information

After you “Create Basic User Information”, you can add more attributes to the basic user, including changing the VTS role and assigning to a user group.

1. "Log in to the VTS GUI", select **Users** from the navigation bar, and click on the User Name of interest.

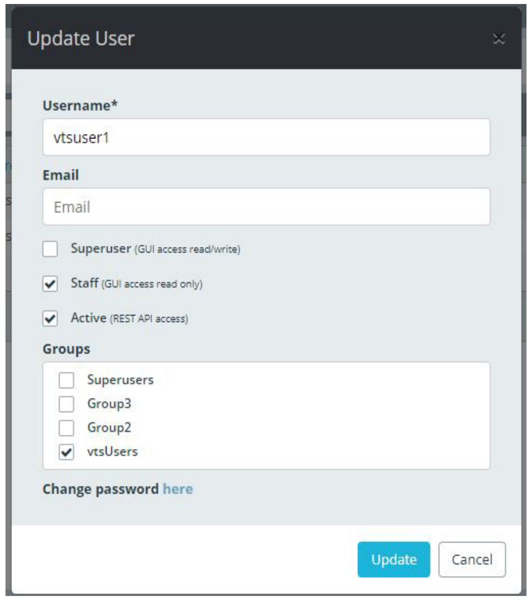


Figure 10: Update user page.

2. Change VTS role, User Group assignment, or Email as needed. Click **Change password here** to update the user's password.
3. Click **Update.**

Delete a User Manually

1. If you delete a user to which permissions have been associated, all permissions linked to the user would be deleted. It is generally recommended to uncheck "Active" status rather than delete users. "Log in to the VTS GUI" and select **Users** from the navigation bar (Figure 8).
2. Select the check box by a user, click **Delete**, and confirm Delete.

Import a User From an AD/LDAP Server

Review the following sections to familiarize yourself with users in VTS:

- "What is a VTS "user?"" on page 4
- "VTS Authentication Basics" on page 5
- "VTS Authorization Basics" on page 7

Have your AD/LDAP connection to VTS set up and configured using the VTS CLI, as described in "Configure an AD/LDAP Server (optional)" on page 38.

Create the appropriate user groups in the VTS GUI, as described in "Create a User Group" on page 51.

Then:

The AD/LDAP administrator can set up their AD/LDAP server to import and authenticate users and user groups.

To import Vormetric Tokenization users from an AD/LDAP server:

1. Create three AD/LDAP server groups, one for REST API users (Tokenization/Crypto/Key Management) and the other two for VTS administrators who require GUI access.
   The group which has been designated as the "Superuser" group in the CLI will have read/write access in the GUI, whereas the group which has been designated as the "Staff" group in the CLI will have read-only access in the GUI.

   Populate each group with the desired REST API ("active") users and VTS administrators.

   The name of the name of the group which will have permission to access the REST APIs must match what was specified in with the active_user command of the auth submenu of the VTS CLI.
   See "Configure an AD/LDAP Server (optional)."

   The name of the VTS users with read-only access to the GUI must match what was specified in the staff_user command of the auth submenu of th eCLI.

   The name of the VTS administrators' group must match what was specified in the **`super_user`** command of the `auth` submenu of the VTS CLI.

   Example AD\LDAP server group names:

   `vtsUsers` - Users in this group can create and/or access data through the RESTful API.
   `vtsStaff` - Users in this group can access the VTS GUI (read only).

   `vtsAdmin` - Users in this group can access the VTS GUI (read/write)

NOTE: staff_user is valid only if the user also has active_user.
super_user is valid only if the user has active_user and staff_user.
Use corporate policy to define whether or not staff_ and super_ users should actually be granted REST API permissions and disallow by policy when necessary to maintain corporate standards or separation of duties.

2. Create three group permissions in the VTS GUI with the same names as the LDAP groups (example: `vtsUsers` , `vtsStaff`, `vtsAdmin`). Members of these AD/LDAP groups inherit these permissions.

   An application program which uses Basic Auth (username/password) will try to authenticate to LDAP using these credentials. An application program which uses a client certificate for authentication will use the Bind user's username and password for looking up the client certificate's common name (CN) in LDAP. If the credentials are authorized, the LDAP user is imported into the user table of the VTS. Thus, there is no need manually to create the LDAP users in the GUI. Once the user has been added to the VTS, permissions can be modified with another group permission or individually.

NOTE: User names and passwords are case-sensitive. Application code that calls the REST APIs must pass user names and passwords in the same case that are defined in the AD/LDAP server.

Specify GDE Appliance Keys

Review "Understanding VTS Keys" to understand the definition and usage of symmetric keys, asymmetric keys, and opaque objects.

Create a Key Name

Use the VTS GUI to enter a key name that corresponds to a key that has been created in the GDE Appliance and are used for VTS functions. Once the name has been entered into the VTS local database, the key is available to associate with user and user group permissions.
Symmetric keys will also be available to associate with tokenization groups.

NOTE: The very first symmetric key entered into VTS needs to be entered from the GUI of the first cluster node. Please enter this key as soon as possible. It will be used to create large lookup tables for the "RANDOM" tokenization mode.

1. “Log in to the VTS GUI” and select **Keys** from the navigation bar.
   The Keys List is displayed, with the **Symmetric Keys**, **Asymmetric Keys**, and **Opaque Objects** tabs in the top navigation.

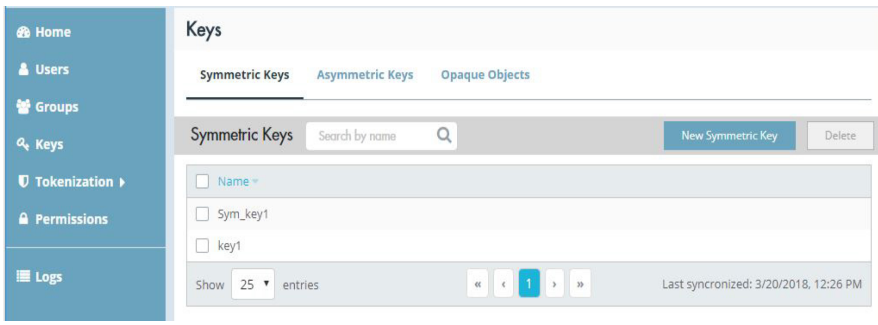


Figure 11: Keys list in the Symmetric Keys tab.

2. Select the tab for the type of key you want (e.g. Symmetric Keys) and click the **New <Symmetric Key>** button.
3. Enter the key name that was used in the GDE Appliance and click **Create**.
   The key name is displayed on the Keys List and will show as available in the Permissions matrix and Tokenization Group pages.

Delete a Key Name

Deleting a key name from the VTS local database does not delete the key material (nor the key name) from the GDE Appliance. Deleting a key name will delete all tokenization groups, tokenization templates and permissions associated to it.

1. “Log in to the VTS GUI”, select **Keys** from the navigation bar.
   The Keys List is displayed, with the Symmetric Key, Asymmetric Keys, and Opaque Objects tabs in the top navigation (Figure 11).
2. Select the tab for the type of key you want (e.g. Symmetric Keys), click the check box by a desired key, and click **Delete**. Confirm and click Delete again.

NOTE: The very first symmetric key entered into the VTS can serve as a regular key for all encryption, signing and tokenization operations, but it is also used to build internal cryptographic tables and hence must not (and cannot) be deleted.

Tokenization Setup

Overview

Review “Tokenization Groups, Templates, & Masks: How Do They Fit In?” on page 13 for the concepts behind setting up tokenization in the VTS GUI. While the GUI does not constrain you to one particular implementation path, a logical sequence might be:

- “Specify GDE Appliance Keys” for tokenization (must be symmetric keys).
- “Create a Tokenization Group” and associate it with the appropriate key(s).
- “Define a Custom Character Set” as needed (optional).
- “Create a data mask” for detokenization.
- “Create a tokenization template” and associate the applicable tokenization group and character set.
- **Create users and user groups** as needed.
(See “Manage VTS User Groups” on page 50 and “Manage VTS Users” on page 51.)
- **Grant tokenization privileges to users and user groups** by going to their permissions page, selecting the symmetric key associated with the appropriate tokenization group, and checking the **Tokenize** box. (See “Assign Key-based Permissions” on page 65.)
- **Grant detokenization privileges to users and user groups** in the same way, assigning a **data mask** from the drop-down menu.

NOTE: See “Other (None Key Dependent) Permissions” on page 9 for an explanation of the Tokenize/Detokenize check boxes listed under “Other Permissions” in the Permissions matrix. This feature is included primarily for legacy and upgrade purposes.

Manage Tokenization Groups

A tokenization group is a vehicle to compartmentalize a VTS server into distinct data spaces, each of which is protected by an encryption key. A tokenization group is used as a parameter in the *Tokenization* API calls, `tokenize` and `detokenize`. It allows application programmers to create islands of sensitive data that are available only to members of a particular tokenization group. To use tokenization, you must create at least one tokenization group.

Overview

Example:
A credit card company accepts business from three retail outlets called *Store1*, *Store2*, and *Store3*. The three retail outlets require tokenization of the customer credit card numbers used at their store. You could enable this by creating three tokenization groups called `Store1`, `Store2`, and `Store3` (the `tokengroup` parameter is case-sensitive). The application code processing sensitive data must specify one of these `tokengroup` parameters in each `tokenize` and `detokenize` calls (see "Tokenization REST API" on page 93).
For example:

```
curl --tlsv1.2 -X POST -u EdYee:EdYee_password -d'{"tokengroup" : "store1" ,
"data" : "9453677629008564", "tokentemplate" : "Credit Card" }'
https://192.168.12.88/vts/rest/v2.0/tokenize
```

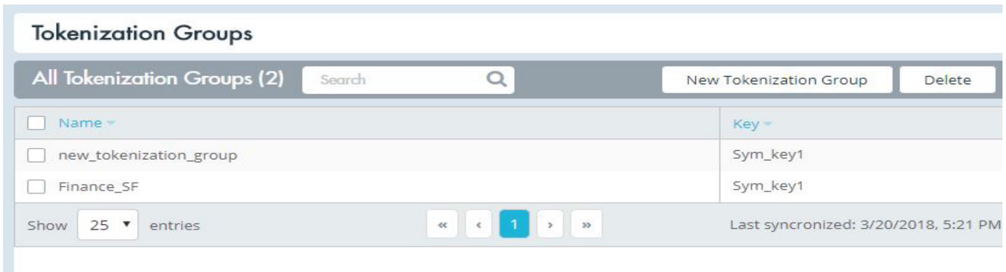
To enable tokenization services for each group, an implementation team would:

- Decide how many tokenization groups are required and what their precise names will be (Application programmer and VTS GUI Administrator)
- Create tokenization groups that match the names to be used in the application code: `Store1`, `Store2`, and `Store3`.
  (VTS GUI Administrator)
- Go to the and:
  - Create tokenization group encryption keys
  - Use key *Template*: **Default_SQL_Symmetric_Key_Template** for those keys
  - Set the *Key Type* to **Cached on Host**.
    Be sure the administrator knows that the first key must be created on the first node installed, not on secondary or subsequent cluster nodes. **Do not delete this key.**
    (GDE Appliance Administrator)
- Specify the appropriate `tokengroup` parameter in the `tokenize` and `detokenize` API calls. Using our example credit card company, the application code would specify `Store1`, `Store2`, or `Store3` for a REST call.
  (Application programmer)

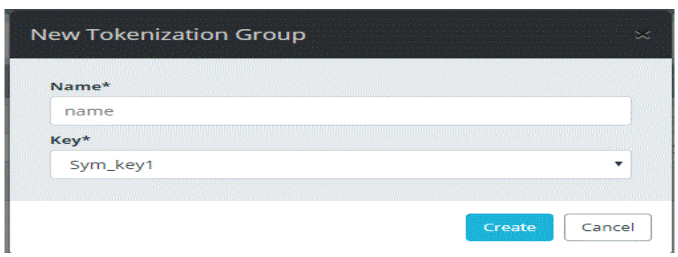
Create a Tokenization Group

Have at least one symmetric key created in the GDE Appliance and entered in the VTS GUI, as described in "Specify GDE Appliance Keys" on page 55. \

1. "Log in to the VTS GUI", select **Tokenization>Tokenization Groups**.
   The All Tokenization Groups list is displayed.



2. Click the **New Tokenization Group** button.
   The New Tokenization Group creation page is displayed.



3. Enter a Tokenization Group **Name**.

NOTE: Tokenization group names are case-sensitive. Application code must pass tokenization group names in the same case that are defined in the VTS GUI.

4. Select a **Symmetric Key** from the drop-down list.

Warning! Do not delete keys used for tokenization. If you delete a key, you will have to recreate the corresponding tokenization group to regain access to your tokens.
Do not use auto-key-rotation features with tokenization.

5. Click **Create**.

Delete a Tokenization Group

Before deleting a tokenization group, make sure that tokens created with the symmetric key associated with this tokenization group are not stored anywhere within (or outside) of the organization. Once the tokenization group used to create these tokens has been deleted, they can no longer be detokenized.

Manage Character Sets

VTS 2.2 has introduced enhanced support for tokenizing international characters. In earlier versions, VTS supported three default character sets:

- All digits
- Alphanumeric
- All printable ASCII

These remain available, while the ability to create custom character sets has been added.

Define a Custom Character Set

To define a custom character set, you need to know the unicode character range for the language(s) you want to support. For example, the range for Arabic is 0600-06FF, while the range for Thai is 0E00 - 0E7F

NOTE: Learn more about character ranges here: http://jrgraphix.net/research/unicode.php.

To make a custom character set available for tokenization in VTS:

1. Log in to the VTS GUI and select **Tokenization> Character sets**.

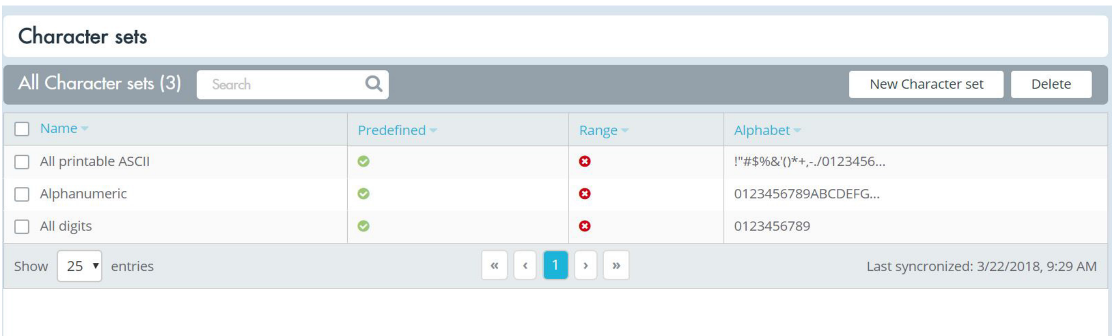


Figure 12: All character sets list is displayed.

2. Click **New Character set**.

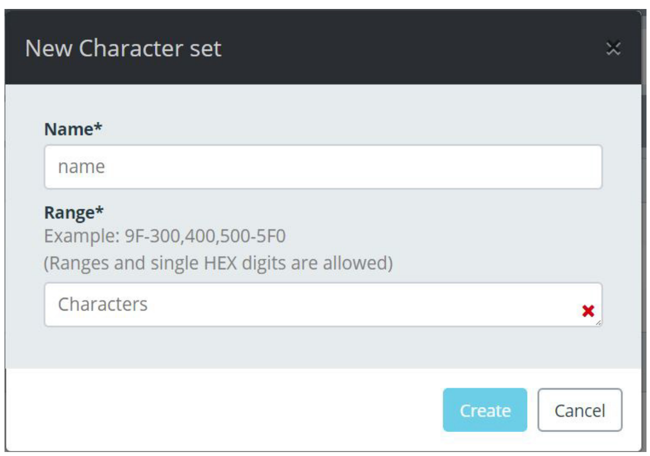


Figure 13: New character set page.

3. Enter a **name** and **range** or set of ranges for your set.
   The maximum size of a custom UTF character set definition string is 1536.
   This implies either 100 ranges, or 200 individual characters, or some combination thereof—for instance, 50 ranges plus 100 individual characters.
4. Click **Create**.
   The new set will appear in the All Character Sets list and be available to apply to tokenization templates.

To edit a character set, click on its name in the list. Predefined character sets cannot be edited and should not be deleted.

Apply a Character Set

The three predefined character sets always appear in the drop-down list of character sets when you create a tokenization template. Once a custom set has been defined, it will appear in the tokenization template creation menu as well.

NOTE: See “Create a tokenization template” on page 62.

Delete a Character Set

Before deleting a character set, make sure that tokens created with this character set are not stored anywhere within (or outside) of the organization. Once the character set used to create these tokens has been deleted, they can no longer be detokenized, unless the character set is recreated very precisely. Even the order of the characters in the character set matters.

Prototyping tokenization with a user-defined UTF-8 character set

VTS 2.2 supports UTF-8 inside the valuename field for "`data`" and "`token`".

Generate a file that has the JSON request in it containing this UTF-8 code, and then submit it via cURL.

A file can be submitted in cURL via the

```
--data-binary @filename option.curl --tlsv1.2 -k -X POST -u
superuser:ssl12345  https://127.0.0.1/vts/rest/v2.0/tokenize   --data-
binary @tokenizerequest.txt
```

Manage Tokenization Templates

Tokenization templates allow you to define the specifics of how you want your data tokenized and detokenized. For example, you can specify that your data be Luhn checked before tokenizing, or that the last four digits of a data string be left unencrypted, or that an identifying prefix be attached to all tokens.

To use tokenization, at least one tokenization template must be created.

Create a tokenization template

1. In the VTS GUI, select **Tokenization> Tokenization Templates.**
2. Click **New Tokenization Template.**
   See Figure 4 on page -15.

3. Enter the following values:

Name - Give the new template a name. This name will be used in the API POST commands `tokenize` and `detokenize`.

Tokenization group - The tokenization group for which this template can be used. You must create at least one template for each tokenization group.

Format - Specify which of the following formats to use when tokenizing data:

- Format-preserving encryption (FPE) with or without a Luhn check. A Luhn check ensures that a submitted number is a valid credit card number. Note: FPE is an alternate name for FF3.
- Format-preserving encryption (FF1) with or without a Luhn check. A Luhn check ensures that a submitted number is a valid credit card number.
- Random or Random with Luhn check. This format is optimized for use with numbers from 9 digits (without a Luhn digit, for instance SSNs) to 19 digits (including a Luhn digit), like credit card numbers or tax ID numbers. If the data includes more free-form text, such as names and addresses, either the FPE or FF1 format is preferable.

NOTE: In a random or random-Luhn format, the template uses a pseudo-random number generator (PRNG) instead of an encryption algorithm. In addition, the very first symmetric key configured in the VTS GUI is not a true encryption key, but instead acts as a seed value for the PRNG.

- One of the supported date formats. Use this to anonymize dates. Between the month, date, and year portions of the date format, the input data can include one of the following separators: slash (/), comma (,), hyphen(-), or space ( ). Date formats are used along with the Start Year and End Year fields.

NOTE: In a date format, if a two-digit year format is selected (for example, DDMMYY), the VTS assumes that the date is in the twenty-first century; that is, it assumes the first two digits of the year are 20.

Keep Left - The number left digits to leave unencrypted. You need at least a total of two digits to tokenize. Cannot be left blank.

Keep Right - The number right digits to leave unencrypted. You need at least a total of two digits to tokenize. Cannot be left blank.

One-Time Use Token - (Optional) Check this box if you never want the token to be detokenized. For example, if you have production data that you want to tokenize for test or development, and you know you'll never want to decrypt it.

Character Set- The character set to use with tokenization.

Prefix - (Optional) An optional prefix to be added to all tokens. Example: **CC-**1234-1234-1234-1234 or **SSN:**123-23-1234. Note that the format is no longer preserved with this feature.

Start Year and **End Year** - The range of time within which all dates in the input data are assumed to fall.

NOTE: After changing Start Year and End Year, do not change them again right away. Wait at least a few minutes. Modifying a date range too soon (that is, before the VTS database finishes processing the first date range) can cause errors.

- **Irreversible** - **CAUTION.** Using this checkbox means that the token created using this template can never be detokenized. For example, you may have production data you want to tokenize for test or development and you know you will never want to decrypt it.

4. Click **Create**.

   To edit a tokenization template, click on a template name.

Sample: Use a tokenization template in the REST API

Once you specify a tokenization template, it can be used in the RESTful API application code. Example:

```
curl --tlsv1.2 -X POST -u EdYee:EdYee_password -d'{"tokengroup" : "store1" , "data" :
"9453677629008564", "tokentemplate" : "vtsUsersTemplate" }'
https://192.168.12.88/vts/rest/v2.0/tokenize
```

Delete a tokenization template

Before deleting a tokenization template, make sure that tokens created with this tokenization template are not stored anywhere within (or outside) of the organization. Once the tokenization template used to create these tokens has been deleted, they can no longer be detokenized, unless the tokenization template is recreated very precisely.

Manage Data Masks

Tokenization data masks hide specified parts of detokenized data. A mask is required for detokenization, even if the user is authorized to view the entire unencrypted string of data.

Create a data mask

Review "Using Data Masks" on page 15**.**

To create a mask:

1. "Log in to the VTS GUI", select **Tokenization > Masks**.
   The All Masks list is displayed.

2. Click New Mask and enter the information on the Mask definition page.
   Some examples of data masks you could create:
   - A mask called SHOW_LAST_4 that shows the last four digits of a tokenized credit card: XXXX-XXXX-XXXX-7897
   - A mask called SHOW_FIRST_3 that shows only the first three digits of a social security number: 565-XX-XXXX.
   - A mask that shows everything (ALL_CLEAR): set **Show First** and **Show Last** values to 999999.
3. Click **Create**.

Delete a data mask

When a Mask is deleted, all user and user group permissions associated with that mask will be deleted.

Warning! This could result in production down time if a user is using the system and their permissions are deleted. They will be denied tokenization and detokenization rights. Be VERY sure these permissions are not in use before deleting the mask.

Apply Permissions to Users and User Groups

Overview

Review the following sections:

- "Understanding VTS Authentication & Authorization" on page 4
- "Understanding VTS Keys" on page 10

With VTS 2.2, users are authenticated, granted permissions (to tokenize, detokenize, encrypt, hash, etc.), and associated with particular keys, using a three-way permissions matrix.

Assign Key-based Permissions

The GUI Administrator assigns permissions based on user (or user group), action, and key, similar to the English sentence:

"User `vtsuser1` may encrypt, decrypt, tokenize and detokenize with symmetric key `key1`.

Before permissions can be assigned, you must:

- Create user groups and users
- "Specify GDE Appliance Keys"

Then:

1. Log in to the VTS GUI and select **Permissions**. The User Permissions list is displayed.
   If necessary, click **User Group** to see the User Group Permissions list.
   You can click the tabs to see the Symmetric Keys, Asymmetric Keys, and Opaque Objects assigned to a user (or group).

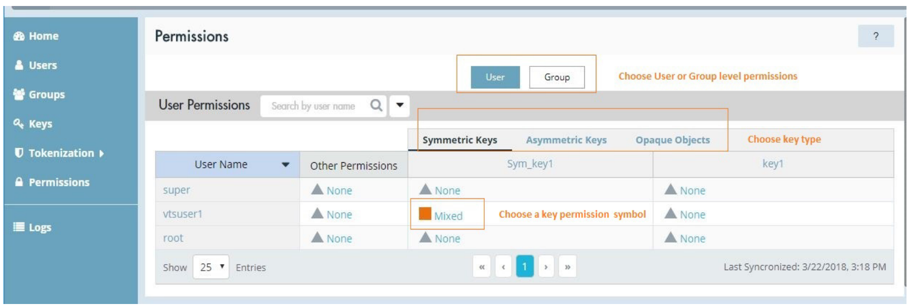
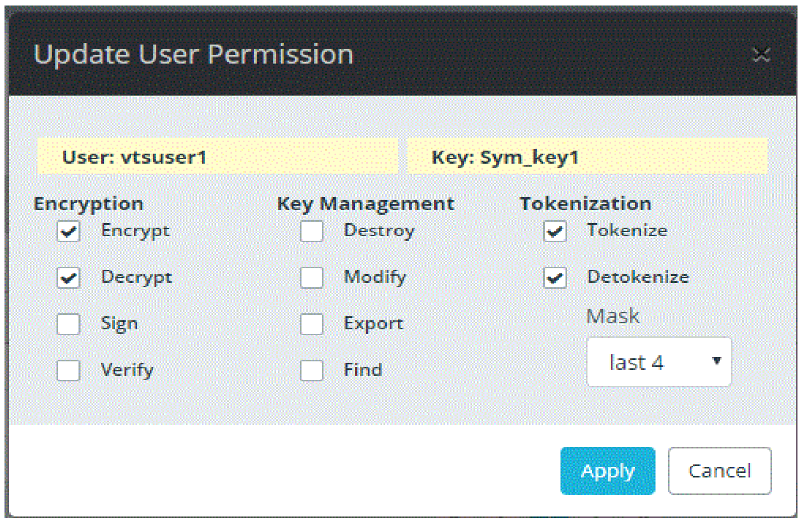


Figure 14: User Permissions list page.

2. Click a user name to see a user permissions summary page. You can review and change permissions there or from the list page. To continue from the list page:
3. Click the permission symbol for a particular user's key. The user permission matrix is displayed.



4. Assign permissions as need and click **Apply**.

Assign Other Permissions

The permissions in the "Other Permissions" category are those that are not tied to specific keys.

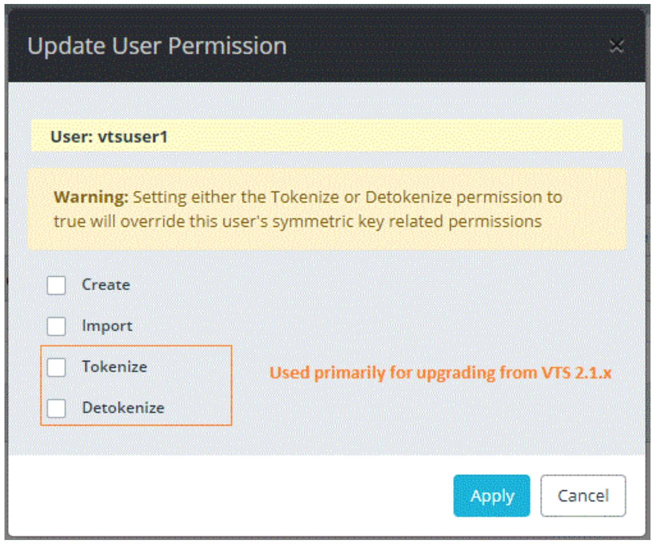
Create & Import Keys

Review "Static vs. Dynamic Key Management Creation" on page 12.

To grant a user or user group permission to create keys on the GDE Appliance or import keys to the GDE Appliance:

1. Log in to the VTS GUI and click **Permissions**. The User Permissions list is displayed. (If necessary, click **Group** to access the User Group Permissions list.)

2. Click the permission symbol for a user in the **Other Permissions** column.
   The Other Permissions assignment window is displayed.



3. Check **Create** and **Import**, as desired, and click **Apply**.

Other Permissions: Tokenize/Detokenize

Checking tokenize/detokenize here applies privileges at a global level, on all symmetric keys in the system. If your system is upgraded from an earlier version, any users who had been granted tokenize/detokenize privileges will have these boxes checked automatically. Remember to de-select the global check boxes and reassign privileges on a per-key basis to take advantage of VTS 2.2's more granular structure.

Log messages

Most tokenization and detokenization log messages are in the file `tokenization.log`. You can get this via web download with credentials from the file `tokenization.log`. Example:

```
https://vtshost.vormetric.com/log/tokenization.log
```

These log messages also go to the syslog with the facility code `daemon`.

Other log files:

`django.request.log` - Logs any user request coming from the Tokenization Server GUI.

`uwsgi_vts.log` - Debug log.

`clish.log` - CLI activities and events.

View the Tokenization Server logs by clicking on the **Logs** icon on the Tokenization Server GUI. (Audit log messages also go to the syslog with the facility code `user`.)

NOTE: To set the VTS log level, see "Set Log Levels" on page 100.

- **Action Time** specifies when the event occurred.
- **User** is the Tokenization user.
- **Content Type** is the database table that was modified. *Tenant* involves Group Tokens, *Key* is encryption keys, *user* is for users, *groups* for groups and so on.
- **Object** is the specific database object that was created or changed.
- **Action** is the action that was performed.
- **Change Message** states what database field was changed.

NOTE: You cannot sort by "Change Message" in the *Audit Log* window.

syslog

Audit log messages go to the syslog with the facility code `user`.

`tokenization.log` messages go to the syslog with the facility code `daemon`.

CLI log messages go to the syslog with the facility code `local3`.

Key Rotation

Regular encryption key rotation can increase security and also satisfy some PCI DSS requirements. To rotate VTS keys, ask your GDE Appliance Administrator to run the GDE Appliance CLI command `security gencert` on the GDE Appliance that creates and stores the VTS keys.

Versioned keys are not permitted for tokenization.

Encryption & Key Management API Programming Notes

5

VTS includes three sets of REST API services: Encryption, Key Management, and Tokenization. The Tokenization APIs (including the reference documentation, examples, and error codes) are described in the chapter "Tokenization REST API" on page 93.

This chapter provides concepts, guidelines, and examples for the Encryption and Key Management APIs. There are also online reference documents for these two sets of services.

This chapter contains:

Reference Documentation

The Vormetric Encryption and Key Management services have built-in online documentation. To access this documentation, the user must be logged in.

Vormetric Encryption RESTful API documentation

`https://<VTS hostname or IP address>/vts/crypto/v1/doc`

Vormetric Key Management RESTful API documentation

`https://<VTS hostname or IP address>/vts/km/v1/doc`

Encryption and Key Management API Concepts

The concepts in this section provide background context for API programmers who will use the Encryption and Key Management APIs in their code.

TES Resource Names (TESRN)

A TESRN is a reference to a resource with the Crypto and Key Management services, specifically a way to address keys. These are used throughout all the cryptographic and key management operations that require a key reference.

It has the following format:

```
tesrn:<managing-service>:[<domain>][:<resource>]:[resource-
type]:<resource-id>[:<version>|<public>|<private>]
```

The intention is to be able to identify where the resource resides and whether the service can handle such or not. There exists of course some abbreviated ways to access keys:

By service default

In the case of the VTS crypto and key management services, A key can be referenced by simply supplying the label: *foo123* key would be *"foo123"*. In TESRN format it would be *"tesrn:vts::label:f00123.* Note that the domain is empty and the resource type, which is absent from the name will default to "key".

Other default key types

In addition, there may be cases where the caller may wish to use a alias, UUID"s or MUID"s instead of a key label. For there cases, the caller simply needs to refer each respectively as illustrated in the following examples:

By Alias:

```
alias:myfoo
```

TESRN format:

```
tesrn:vts::alias:myfoo
```

By UUID:

```
uuid:7fa96ada-18c6-11e8-8dab-0800277415ff
```

TESRN format:

```
"tesrn:vts::uuid:7fa96ada-18c6-11e8-8dab-0800277415ff"
```

By *MUID*:

```
muid:
```

```
urn:
```

TESRN format:By *Label*:

```
"tesrn:vts::muid:91eb92c2-18c6-11e8-accb-0800277415ff91ec0054-
18c6-11e8-b01e-0800277415ff"
```

```
label:foo123
```

```
"tesrn:vts::label:foo123"
```

Legacy Resource Names

While not specific to the VTS services, it is worth mentioning WSOP (nShield Web Services Option Pack) as a future interoperability consideration. For WSOP keys are accessed by UUID only. These key references are expressed as:

```
urn:c1d6f102-18c6-11e8-b2f9-0800277415ff
```

Algorithms and Key Types

The Vormetric Encryption and Key Management APIs make references throughout the specification to algorithm types. The algorithm types have names which conform to the JOSE RFC-7518 Specification, plus a decision to extend the algorithm namespace (maintaining the JOSE syntax) to meet customer needs.

NOTE: The algorithm name will be **case insensitive** for use with the services. The complete list of algorithm names is:

Key types also conform to the JOSE RFC-7518 section 6.1 specification.

NOTE: Though this section describes the key types is case sensitive for JOSE, the current implementation is **case insensitive** for key types.

Table 6: Key Types

Key Type	Description	Algorithms	State
Symmetric	oct or octet sequence. Used for: • tokenize • detokenize • encrypt • decrypt Symmetric keys are also used for Tokenization, and while the symmetric keys are typical AES keys with 128 or 256 bits of key material, the algorithms are FPE (FF3) and FF1. **Note:** Versioned keys may not be used for tokenization, only non-versioned keys.	A128ECB - AES 128 ECB cipher mode A256ECB - AES 256 ECB cipher mode A128CBC - AES 128 CBC cipher mode A256CBC - AES 256 CBC cipher mode A128CBCPAD - AES 128 CBC-PAD cipher mode A256CBCPAD - AES 256 CBC-PAD cipher mode A128CTR - AES 128 CTR (counter) cipher mode A256CTR - AES 256 CTR (counter) cipher mode	At this time, only symmetric keys support states. On create and import, only a preactive or active state may be given. On a key update all states can be given except for preactive. The following states are available: • Preactive • Active - Default when a key is created • Suspended • Compromised • Deactivated • Destroyed
Asymmetric	RSA key type. Used for: • encrypt • decrypt • sign • verify	RSA	

Table 6: (Continued)Key Types

Key Type	Description	Algorithms	State
Opaque object	Octet array. This is a Thales extension (not part of JOSE RFC-7518 spec). Used for • sign • verify Does not have the option of store on server. Size limit: 0-4096 bytes.	HS1 - HMAC-SHA-1 HS224 - HMAC-SHA-224 HS256 - HMAC-SHA-256 HS384 - HMAC-SHA-384 HS512 - HMAC-SHA-512	N/A

NOTE: EC Key type is an elliptic curve cryptographic key, which is currently not supported.

Opaque Objects

An opaque object is a generic form of object that is securely stored on the GDE Appliance and is currently used by the underlying VAE PKCS11 library to implement some extensions that are not available natively on the GDE Appliance.

Currently, VTS and VAE primarily use opaque objects to represent HMAC keys. They are created or imported using the same endpoint as for symmetric and asymmetric keys, except that the key type (kty) must be opaque. If a wrapped key is provided, this will be the key material used to create the opaque object key. If only a size is given, the VTS cryptographic service will generate a key using the GDE Appliance true random number generator.

The size limit for an object is 4096 bytes. When size is omitted, or set to zero, and key bytes are present, the size is computed from the size of the byte sequence. When an opaque object has empty key bytes with no specific size defined, it creates an object.

In this release opaque objects can only be used for signing and verifying. Opaque objects support cryptographic header versions 2.1 and 2.7 only.

Cryptographic Headers

Cryptographic headers is a convenient feature that allows a ciphertext to be decrypted without knowledge of the specific key name and/or version that was used to encrypt it. When encrypting with headers enabled, the ciphertext includes bytes containing key information inserted before the encrypted data. To enable cryptographic headers, the ***header*** attribute must appear in the POST body for sign, verify, encrypt and decrypt calls.

NOTE: The two versions of headers supported in this release are 2.1 and 2.7.

Verify and decrypt also take an ***all*** header attribute value to let the service figure out the header version in a more transparent way for the caller.

Versioned Keys

Key versioning is available for symmetric keys in the current release of VTS 2.2 with the VAE 6.0.2 library. The rules for versioned keys is different from conventional symmetric keys. As a general rule of thumb:

- The label represents the key class. A reference to the key by label will always refer to the most recent activated key version.
- Every key version has its own UUID and MUID
- A key can be deleted by UUID or MUID
- Deleting version zero (0) of a key class by UUID or MUID, deleted the entire key class (all versions)
- Versioned keys can be used for signing, verifying, encryption and decryption
- To create a new versioned key, the create endpoint must have an ***action*** attribute set to **create.** A version number must also be supplied.

Some limitations for this release are:

- Inability to find a particular version of a key by label and version number
- Internal Key IDs are not supported

Key States

Symmetric and asymmetric keys support states. The following states are available:

- Preactive - When a key was created to have a preactive state.
- Active - default when a key is created
- suspended - a key has been suspended.
- compromised - key is compromised
- deactivated - key is deactivated
- destroyed - key is destroyed

To set a key to the suspended, compromised, deactivated state, one must call the modify endpoint (`PATCH`). A key can be set to the active state again, only if it is in the suspended state.

Valid Key Operations

This table lists the valid key operations from the API. If does not include permissions related to authorization. See the Authorization section below for more details.

Table 7: Valid Key Operations

Operation	Symmetric Key	Versioned Symmetric Key	Asymmetric Key	Opaque Object
Create	yes	yes	yes	yes
Import	yes	yes	no	yes
Destroy	Key delete custom attribute delete	Delete key by UUID or MUID only Delete key of version 0 deletes entire set Delete custom attribute	Key delete custom attribute delete	Key delete custom attribute delete
Modify	custom attributes add alias key states key migration to versioned key	custom attributes add alias key states key rotation lifespan interval	custom attributes	custom attributes add alias
Find	By label/name By UUID By MUID By alias	By label/name - last version By UUID By MUID By alias	public key only By label/name	By label/name By UUID By MUID By alias
Export	clear_key kid for plain bytes wrapping key with A256CBC or A256CBCPAD	clear_key kid for plain bytes wrapping key with A256CBC or A256CBCPAD	no	no
Encrypt	Standard encryption cryptographic headers	Standard encryption cryptographic headers	Standard encryption only	no
Decrypt	Standard decryption cryptographic headers	Standard decryption cryptographic headers	Standard decryption only	no

Table 7: Valid Key Operations (Continued)

Operation	Symmetric Key	Versioned Symmetric Key	Asymmetric Key	Opaque Object
Sign	HMAC signing cryptographic headers	HMAC signing cryptographic headers	RSA signing only	HMAC signing cryptographic headers
Verify	HMAC verifying cryptographic headers	HMAC verifying cryptographic headers	RSA verifying only	HMAC verifying cryptographic headers

Wrapping Keys

When importing or exporting a key, it is recommended to use a wrapping key in order not to compromise the key itself. In the current version of the VTS Key Management module a wrapping key must be an AES 256-bit key and only the CBC or CBCPAD ciphers may be used. The ideal method for wrapping a key on both import and export is to use asymmetric keys, however such is currently unsupported by the underlying libraries and services which implement the VTS Key Management module.

Importing an exporting keys require that the user has a wrapping key and that such key is available on both the client and the GDE Appliance.

Importing With Wrapping Key

When importing with a wrapping key, the following steps need to be taken:

- Create a 256-bit AES key that is known to the caller and the GDE Appliance. This is the wrapping key.
- Encrypt the key bytes with the wrapping key using an AES-CBC or AES-CBCPAD cipher. This is the wrapped key.
- Call the create endpoint an supply the base64 encoded wrapped key and the wrapping key ID.

Exporting With Wrapping Key

When exporting with a wrapping key, the following steps need to be taken:

- Create a 256-bit AES key that is known to the caller and the GDE Appliance. This is the wrapping key.
- Call the export endpoint, supplying a wrapping key ID. This will return a wrapped key.
- Decrypt the base64 encode wrapped key using the wrapping key on the client side.

Importing and Exporting Using Clear Bytes

The API offers the ability to perform import and export function using raw key bytes, however such method is not recommended. In order to import a key using clear bytes, simply omit the wrapping key ID or given the ***clear_key*** label. For exporting, give the ***clear_key*** label as the wrapping key ID. In both cases, the ***alg*** and the ***iv*** attributes must be omitted.

Authorization

The usage of any key on VTS 2.2 requires user or group level authorization for a particular key. Authorization. The following sections describe tasks associated with authorization:

- "Creating & Importing Keys" on page 79
- "Managing Keys" on page 79
- "Managing Keys" on page 79
- "Cryptographic Operations" on page 80
- "Authorization Operations" on page 80

Creating & Importing Keys

This category covers both key import and key creation. A user or group that has either import, create or both will be allowed to add keys to the VTS. The difference between import from create is in the ability to add a wrapped key representing the key itself to the key creation request. This ability to create and import keys is set for each user through the VTS Administration UI.

If a user or group has create and/or import permissions, when invoking the corresponding API endpoint will create a key with all the permissions for key management and cryptographic operations. Any other user needing to perform operations on such keys, can only be added through the VTS Administration UI. If the user does not have permission to create or import, the corresponding creation endpoint will return an error.

Managing Keys

A user or group may perform each of the following key management operations if he or she has permissions to perform such. This can be configured for each individual key from inside the VTS Administration UI. The allowed key management operations are:

- Destroy - destroy a key
- Modify - modify a keys attributes
- Find - lookup or retrieve a specific key
- Export - export a specific key. This only applies to symmetric keys.

Cryptographic Operations

A user or group may perform each of the following cryptographic operations using a specific key if he or she has permissions to perform such as configured for each individual key from inside the VTS Administration UI. The allowed cryptographic operations available are:

- Encrypt - encryption using a key
- Decrypt - decryption using a key
- Sign - sign with a given key
- Verify - verify a signature with a given key

Authorization Operations

This table shows the possible authorization operations when a key is or was created or imported via the API. The owner is the user which creates the key. This table only applied to authorization. The Valid Key Operations table should be used to determine what are valid operations not taking authorization into account.

Table 8: Authorization Operations

Operation	Owner	Group	Other user	Other Group
Create	yes, if owner has create permission Owner can specify on create the permissions allowed though the ***usage*** attribute If ***usage*** attribute is not supplied, key will be created with all permissions	yes, if owner has create permission granted through a group Owner can specify on create the permissions allowed though the ***usage*** attribute If ***usage*** attribute is not supplied, key will be created with all permissions	N/A	N/A
Import	yes, if owner has import permission Owner can specify on import the permissions allowed though the ***usage*** attribute If ***usage*** attribute is not supplied, key will be imported with all permissions	yes, if owner has import permission granted through a group Owner can specify on import the permissions allowed though the ***usage*** attribute If ***usage*** attribute is not supplied, key will be imported with all permissions	N/A	N/A

Table 8: Authorization Operations (Continued)

Operation	Owner	Group	Other user	Other Group
Destroy	yes, if owner gave it permission on creation Key if removed from the VTS Authorization table unless it is a versioned key	No	No	No
Export	yes, if owner gave it permission on import or create	No	No	No
Find	yes, if owner gave it permission on import or create	No	No	No
Modify	yes, if owner gave it permission on import or create	No	No	No
Encrypt	yes, if owner gave it permission on import or create	No	No	No
Decrypt	yes, if owner gave it permission on import or create	No	No	No
Sign	yes, if owner gave it permission on import or create	No	No	No
Verify	yes, if owner gave it permission on import or create	No	No	No

Date Handling

Throughout the API specification, dates are all represented in RFC-3339 format. Any date that appears within the request or result body in JSON format will be in this format. As a general pointer, when using different programming languages in writing client code to access the APIs, we prepared the following:

Go

The time struct/interface in golang is already marshaled/unmarshaled for JSON in RFC339.

Python

Python requires the installation of a package that implements rfc3339. A *"pip install rfc3339"* will address most users needs.

```
Python rfc3339
import rfc3339, time
rfc3339.rfc3339(time.time())
```

PHP

RFC3339 is a predefined constant in date for `PHP: date(DATE_RFC3339)`

Bash/cURL

RFC-3339 dates can be obtained by typing: *date --rfc-3339=seconds*

C++/boost

```
#include <boost/date_time.hpp>
std::cout.imbue(std::locale(std::cout.getloc(), new
boost::local_time::local_time_facet("%Y-%m-%dT%H:%M:%S%F%Q")));
std::cout <<
boost::local_time::local_microsec_clock::local_time(boost::local_time::ti
me_zone_ptr()) << std::endl;
```

Successful Responses

All successful requests will return a 200 HTTP status. All calls return a response body, except for destroy (`DELETE`) of a specific key which will have an empty body.

In a batch or vectorized call, where a series of requests are done simultaneously, the result is always 200 and each element of the vector contains the corresponding result for each request, whether successful of not.

Error Responses and Error Codes

Error responses from the cryptographic and key management layer only returns a JSON body with the following format. Anything different from this format is from the NGINX reverse proxy or other components.

Error Response Example:


```
{"error":"TES_INVALID_OBJECT_HANDLE","message":"find
object returned an invalid
handle","status":400,"timestamp":"2018-03-
08T11:56:38.793756418-08:00"}
```


Table 9: Response Parameters

	Description
`error`	This response parameter indicates an error is returned. The error codes start with a TES_ for service related error checking and a CKR_ for PKCS11 related error codes encountered.
`message`	
`status`	This response parameter provides the HTTP status code associated with the error. For a list of possible error codes see: Table 10, " Error HTTP Status Codes," on page 83.
`timestamp`	

Table 10: Error HTTP Status Codes

	Description
400	An HTTP 400 Bad Request status indicating that the user supplied a malformed or invalid request.
401	An HTTP 401 Unauthorized status indicating that the user need to login or authenticate before the request can be handled.
403	An HTTP 403 Forbidden status indicating that a logged in user does not have authorization to access or perform an operation on the given resource.
404	•An HTTP 404 Not Found status indicating that the URL or resource (TESRN) references does not exist.
500	An HTTP 500 Server Internal Error indicating that an error occurred on the service side and that it may be available later.

Key Management Examples

Key Creation

Key creation allows to create a key whose contents (key material or bytes) is generated by the GDE Appliance.

This example shows how to create an AES 128-bit key named foo123key:

Request:

```
POST /vts/km/v1/keys {
"size": 128, "name": "foo123key", "kty": "oct"
}
```

Request:

```
200 OK
{"attributes":{"usage":["encrypt","decrypt","sign","verify","destroy","
modify","find","export"],"size":16},"metadata":{"key_management":{"uuid
":"9f3c2775-405b-33e8-91af-
904faa0e551f","caching_duration":44640,"state":"active","lifespan_unit"
:"days","muid":"9f3c2775-405b-33e8-91af-904faa0e551f39b6aee0-6821-39f9-
a186-
9a9904abff32","caching_enabled":true,"name":"foo123key"}},"kid":"tesrn:
vts::label:foo123key"}
```

Key Import

Key import is basically the same as a key create, however the request contains information on the `wrappedkey` and the wrapping key.

This request imports an AES 128-bit key named foo1234key with key bytes equal to: “`HelloWorldAlways`”.

Request:

```
POST
/vts/km/v1/keys{"size":128,"name":"foo1234key","kty":"oct","wrappedkey"
:"SGVsbG9Xb3JsZEFsd2F5cw=="}
```

Response:

```
200 OK
{"attributes":{"usage":["encrypt","decrypt","sign","verify"],"size":16}
,"metadata":{"key_management":{"uuid":"9f3c2775-405b-33e8-91af-
904faa0e551f","caching_duration":44640,"state":"active","lifespan_unit"
:"days","muid":"9f3c2775-405b-33e8-91af-904faa0e551f39b6aee0-6821-39f9-
a186-
9a9904abff32","caching_enabled":true,"name":"foo123key"}},"kid":"tesrn:
vts::label:foo1234key"}
```

Key Delete

This examples deletes a key named foo123key.

NOTE: Upon success, the HTTP status code must be 200.

Request:

```
DELETE /vts/km/v1/keys/foo123key
```

Response:

```
200 OK
```

Key Update

Request:

```
POST /vts/km/v1/keys/fookey {"state": "suspended"}
```

Response:

```
200 OK
{"attributes":{"usage":["encrypt","decrypt","sign","verify","destroy","
find"],"size":16},"metadata":{"key_management":{"uuid":"bfd2a7b7-20db-
31d7-b22f-
01e625a54961","caching_duration":44640,"state":"suspended","versioning_
enabled":1,"lifespan_unit":"days","muid":"bfd2a7b7-20db-31d7-b22f-
01e625a549618b0ce745-fcb2-3718-97b3-
d4a0b415ac5d","caching_enabled":1,"name":"fookey"}},"kid":"tesrn:vts::l
abel:fookey"}
```

Key Retrieval

Request:

```
GET /vts/km/v1/keys/fookey
```

Response:

```
200 OK
{"attributes":{"usage":["encrypt","decrypt","sign","verify","find","des
troy"],"size":16},"metadata":{"key_management":{"uuid":"bfd2a7b7-20db-
31d7-b22f-
01e625a54961","caching_duration":44640,"state":"suspended","versioning_
enabled":1,"lifespan_unit":"days","muid":"bfd2a7b7-20db-31d7-b22f-
01e625a549618b0ce745-fcb2-3718-97b3-
d4a0b415ac5d","caching_enabled":1,"name":"fookey"}},"kid":"tesrn:vts::l
abel:fookey"}
```

Key Export

Request:

```
POST /vts/km/v1/keys/export/fookey {"alg": "A256CBCPAD", "params": {"iv":
"REREREREREREREREREREREA=="}, "kid": "tesrn:vts::label:foo123key"}
```

Response:

```
200 OK
{"exported_key":{"attributes":{"usage":["encrypt","decrypt","sign","ver
ify","destroy","find"],"size":16},"metadata":{"key_management":{"uuid":
"a0415d15-74d5-3d90-b755-
53d88b6b2c31","caching_duration":44640,"state":"active","lifespan_unit"
:"days","muid":"a0415d15-74d5-3d90-b755-53d88b6b2c315276f6c8-ecde-30aa-
87b9-
17d63a432d94","caching_enabled":1,"name":"fookey"}},"kid":"tesrn:vts::l
abel:fookey"},"wrappedkey":"pXSyQ1eGDEucWA1Mvmwix5tTGezs73Hr92hecKtn5Bc
="}
```

Cryptographic Operation Examples

Simple SHA-256 Digest Example

Request:

```
POST /vts/crypto/v1/digest
{"alg": "S256", "payload": "Ef+4MkU6ukzQSMbHxH4rxoVF/xKHyg=="}
```

Response:

```
200 OK
{"digest": "RfdwuYKwBn0l03ZMGB7r59S+P+T9/s8ufPJfcSQXNfE="}
```

Encryption

AES-256 CBC Encryption Example

Request:

```
POST /vts/crypto/v1/encrypt{
    "plaintext": "JLns9BI4Pa7TTrCaeJIjQQ==",
    "alg": "A256CBC",
    "params": {
        "iv": "tler+lrCnchFRJDCFs1F/A=="},
    "kid": "mykey123"
}
```

Response:

```
200 OK
{u"ciphertext": "SZASu6Ts+01IMtb14+djhg==", "params": {}}
```

Simple AES-256 CTR Encryption with Headers Example

Request:

```
POST /vts/crypto/v1/encrypt
{"plaintext": "JLns9BI4Pa7TTrCaeJIjQQ==", "alg": "A256CTR",
"params": {"iv": "tler+lrCnchFRJDCFs1F/A=="}, "kid":
"myversionedkey", "headers"="hdr_v2.7"}
```

Response:

```
200 OK
{"ciphertext":
"UktNQzIxMAD/////AAAABXV1aWQAAAAAEDkKfAiGhjeLkp50NDgjPQT/////AAAA
A2l2AAAAABBhYmMxMjNkZWZnaDY3ODkw/////wAAAAVjc3VtAAAAACAbtDVcSesXu
7SYqoZxNR+hC6Cq8H3gE5xv5vvSzRhUrKcwnBu8", "params": {}}
```

Decryption

Simple AES-256 CBC Encryption Example

Request:

```
POST /vts/crypto/v1/decrypt
{"ciphertext":
"wInPTVRugkxP0KBm46yCo8lpQMaDDRol5bEWt9Mz629MKh7IqB9roh36ZkyIumLi
", "alg": "A256CBC", "params": {"iv": "ek5LmcwOJATvux5PoSaZ6w=="},
"kid": "fookey"}
```

Response:

```
200 OK
{"plaintext":
"4Oidw01cBKKcF1tK4ZBdZg7yDe9KbvBP1npfs382HT+yz0Fb3CeCh/eBIhH7a1rb
", "params": {}}
```

Simple AES-256 CTR Decryption with Headers Example

Request:

```
POST /vts/crypto/v1/decrypt
{"ciphertext":
"UktNQzIxMAD/////AAAABXV1aWQAAAAAEDkKfAiGhjeLkp50NDgjPQT/////AAAA
A2l2AAAAABBhYmMxMjNkZWZnaDY3ODkw/////wAAAAVjc3VtAAAAACAbtDVcSesXu
7SYqoZxNR+hC6Cq8H3gE5xv5vvSzRhUrKcwnBu8", "alg": "A256CTR",
"params": {"iv": "tler+lrCnchFRJDCFs1F/A=="}, "headers"="all"}
```

Response:

```
200 OK
{"plaintext": "JLns9BI4Pa7TTrCaeJIjQQ==", "params": {}}
```

Sign

Simple HMAC-SHA256 sign example

Request:

```
POST /vts/crypto/v1/sign
{"alg": "HS256", "payload": "aGVsbG8gd29ybGQh", "kid": "mykey"}
```

Response:

```
200 OK
{""signature": "13pCxjIXjY8MNhsx69szyUOVraQc6Tn0C0irqCutOLs="}
```

Verify

Simple HMAC-SHA256 Verify Example

Request:

```
POST /vts/crypto/v1/verify
{"alg": "HS256", "signature":
"13pCxjIXjY8MNhsx69szyUOVraQc6Tn0C0irqCutOLs=", "payload":
"aGVsbG8gd29ybGQh", "kid": "mykey"}
```

Response:

```
200 OK
{"valid": 1}
```

Random

Simple Random Example

The following example shows a simple request to generate an 8 byte random number:

Request:

```
POST /vts/crypto/v1/random
{"seed": "q8qqyw==", "size": 8}
```

Response:

```
200 OK
{"random": "152LM5GzL8U="}
```

Versioned Key Management Examples

This section is dedicated into showing how to use versioned keys within the Cryptographic and Key Management set of APIs.

Versioned Key Creation

Versioned Key Creation Example

Creation of an AES 256-bit versioned key named *vkey256* with each version of the key with a lifespan of 7 days.

Request:

```
POST /vts/km/v1/keys
{"size": 256, "name": "vkey256", "kty": "oct",
"versioning_enabled": true, "version": 0, "version_action":
"create", "lifespan_interval": 7}
```

Response:

```
200 OK
{"attributes": {"usage": ["encrypt", "decrypt", "sign", "verify",
"destroy", "modify", "find", "export"],
"size": 16},
"metadata": {"key_management": {"uuid": "9f3c2775-405b-33e8-91af-
904faa0e551f",
"caching_duration": 44640,
"state": "active",
"lifespan_unit": "days",
"muid": "9f3c2775-405b-33e8-91af-904faa0e551f39b6aee0-6821-39f9-
a186-9a9904abff32",
"caching_enabled": True,
"name": "vkey256"}},
"kid": "tesrn:vts::label:vkey256"}
```

Versioned Key Import Example

Creation of an AES 256-bit versioned key named *vkey256* with each version of the key with a lifespan of 7 days.

Request:

```
POST /vts/km/v1/keys
{"size": 256, "name": "vkey256", "kty": "oct",
"versioning_enabled": true, "version": 0, "lifespan_interval": 7,
"wrappedkey": "SGVsbG9Xb3JsZEFsd2F5cw=="}
```

Response:

```
200 OK
{"attributes": {"usage": ["encrypt", "decrypt", "sign", "verify",
"destroy", "modify", "find", "export"],
"size": 16},
"metadata": {"key_management": {"uuid": "9f3c2775-405b-33e8-91af-
904faa0e551f",
"caching_duration": 44640,
"state": "active",
"lifespan_unit": "days",
"muid": "9f3c2775-405b-33e8-91af-904faa0e551f39b6aee0-6821-39f9-
a186-9a9904abff32",
"caching_enabled": True,
"name": "vkey256"}},
"kid": "tesrn:vts::label:vkey256"}
```

Standard Key to Version Key Migration Example

Creation of an AES 256-bit versioned key named *vkey256* with each version of the key with a lifespan of 7 days.

Request:

```
PATCH /vts/km/v1/keys/existingkey
{"version_action": "migrate"}
```

Response:

```
200 OK
{
  "attributes": {
    "size": 32
  },
  "metadata": {
    "key_management": {
      "uuid": "7f460ca4-04e2-37d9-8d57-b2c36fbed55e",
      "caching_duration": 44640,
      "state": "active",
      "versioning_enabled": true,
      "muid": "7f460ca4-04e2-37d9-8d57-b2c36fbed55ecd222707-123a-
3df5-baef-c9b1f77ac3fb",
      "caching_enabled": 1,
      "name": "existingkey"
    }
  },
  "kid": "tesrn:vts::uuid:7f460ca4-04e2-37d9-8d57-b2c36fbed55e"
}
```

Version Key Rotation Example

Rotation of an existing versioned key named *vkey256* with each version of the key with a lifespan of 7 days.**Request:**

```
POST /vts/km/v1/keys
{"size": 256, "kid": "vkey256", "kty": "oct",  "version": 1,
"version_action": "rotate"}
```

Response:

```
200 OK
{"attributes": {"usage": ["encrypt", "decrypt", "sign", "verify",
"destroy", "modify", "find", "export"],
"size": 16},
"metadata": {"key_management": {"uuid": "d0a031de-5102-3bba-80f4-
7019306da03e",
"caching_duration": 44640,
"state": "active",
"lifespan_unit": "days",
"muid": "d0a031de-5102-3bba-80f4-7019306da03ed0a031de-6821-39f9-
a186-9a9904abff32",
"caching_enabled": True,
"name": "vkey256"}},
"kid": "tesrn:vts::label:vkey256"}
```

6 Tokenization REST API

Learn about the VTS Tokenization API.

The Vormetric Tokenization API consists of:

- `tokenize` - A POST call that tokenizes data. A user submits sensitive data, the Vormetric Tokenization Server returns a token. The token is then stored in the production database.
- `detokenize` - A POST call that detokenizes data. A user submits a token to the Vormetric Tokenization Server and the actual data is returned detokenized. Permission settings for that user dynamically specify the appropriate data mask.
- `log` - Changes the log level. Four different log levels (error, info, debug and trace) can be selected. Chose 'error' or 'info' for normal operation, or 'debug' when instructed to do so by Thales e-Security support.

This chapter includes the following sections:

NOTE: See "Encryption & Key Management API Programming Notes" on page 71 for information about the Encryption and Key Management REST APIs.

Using the API

To enable tokenization, you do not have to change your database schema, because Vormetric Tokenization preserves the format of your data. However, you must change your application code.

When a user makes a request to tokenize or detokenize, the username and password must be provided. This is a part of a REST request. The username/password is validated by either the AD/LDAP server or VTS, depending on where the user is defined.

NOTE: The data passed to and from the Tokenization Server—user name, tokenization group name, tokens, and sensitive data is case-sensitive, and the application code that passes data to the Tokenization Server must be aware of this.

Create a Token

Description

Use the `tokenize` endpoint to encrypt a data string to a token using format-preserving encryption (FPE or FF1).

Table 11: Tokenize Endpoint Details

Data	Description
HTTP Request	`https://`*`VTS_IP_Address`*`/vts/rest/v2.0/tokenize`
Method	POST
VTS IP Address	The IP address of the VTS to access with the API call.
Response Format	JSON
Required Authentication	A valid user name and password must be passed as an argument. The values are case sensitive.

Table 12: Tokenize Response Parameter

Parameter	Type	Description
`token`	string	Token for the sensitive data. **Example value**: 2134234509874567
`status`	string	•**Succeeded** (if request successful) •**Error** (if request failed).
`reason`	string	If status is **Error** a **reason** is displayed. Otherwise the **reason** is omitted.

Table 13: Tokenization Parameters

Parameter	Required	Type	Description
`tokengroup`	Y	string	Defines a group name space in the configuration database. For example, let's say you are a credit card company and you want the VTS to support three departments supporting Store1, Store2, and Store3. You want employees in these departments to have access to data for those respective stores. Two actions are required: • The application code must specify a different `tokengroup` parameter in the POST calls for each group supported (example: `Store1`, `Store2`, and `Store3`). The `tokengroup` parameter is case-sensitive. • The VTS GUI administrator must create token groups that match the `tokengroup` parameter names used in the application code. If you do not support multiple token groups, then `tokengroup` will always be the same value. **Example value**: `store1`
`data`	Y	string	The data string to tokenize. The argument considers case and is limited to 128KiBs **Example value**: `1432445695309134`
`tokentemplate`	Y	string	VTS GUI Administrator-defined name for a group of properties that define tokenization operation. Properties include the token group to which the template applies, the tokenization format (date, FPE or FPE with Luhn check, or FF1 or FF1 with Luhn check, or Random (lookup) or Random with Luhn check), number of leftmost or rightmost characters to *not* tokenize, whether you wish to never detokenize a tokenized entry, the character set used for tokenization, and an optional prefix for tokens. **Example value**: `Credit Card`

Create Token CURL Example

NOTE: If you use the `--tlsv1.2` CURL Command syntax option, you must use CURL version 7.34 or higher.

Example Request:

```
curl -k  -X POST -u Donald_Duckmiller:Panda45# -d'{"tokengroup" :
"vtsUsers-t" , "data" : "9453677629008564", "tokentemplate" :
"vtsUsersTemplate" }' https://192.168.118.140/vts/rest/v2.0/tokenize
```

NOTE: The user name (`Donald_Duckmiller`) and tokengroup (`vtsUsers-t`) are case sensitive.

Example Response

```
{"token":"CC-0332335756020172","status":"Succeed"}
```

IMPORTANT: If FPE-luhn, FF1-luhn, or Random-luhn is selected in the Change Token Template page of the VTS GUI, the data must pass a Luhn check or tokenization will fail with the error message:
`{"status":"error","reason":"Luhn check failed for input data."}`

Tokenizing Multiple Data Items

Up to 1,000 data items can be tokenized in a single `curl` command by submitting a JSON array in the REST request. For example, to tokenize four data items you might enter the following in a `postcnn.txt` file:

Example Request:

```
cat postccn.txt

[{"tokengroup" : "vtsUsers-t" , "data" : "9453677629008564",
"tokentemplate" : "vtsUsersTemplate" },{"tokengroup" : "vtsUsers-
t" , "data" : "9453677629008564", "tokentemplate" :
"vtsUsersTemplate" },{"tokengroup" : "vtsUsers-t" , "data" :
"9453677629008564", "tokentemplate" : "vtsUsersTemplate"
},{"tokengroup" : "vtsUsers-t" , "data" : "9453677629008564",
"tokentemplate" : "vtsUsersTemplate" }]

curl -k -X POST -u Donald_Duckmiller:Panda45# --data-binary
@postccn.txt https://192.168.118.140/vts/rest/v2.0/tokenize
```

Warning! Newlines are NOT allowed in the input.

Example Response:


```
[{"token":" 9465976792116170","status":"Succeed"},
{"token":" 9465976792116170","status":"Succeed"},
{"token":" 9465976792116170","status":"Succeed"},
{"token":" 9465976792116170","status":"Succeed"}]
```


The tokenized data is returned in the CURL Command syntax response, in order of submission.

Get Detokenize Resource Data

Description

Get detokenized data from a resource..

Table 14: Detokenize Endpoint Details

Data	Description
HTTP Request	`https://`*`VTS_IP_Address`*`/vts/rest/v2.0/detokenize`
Method	POST
VTS IP Address	The IP address of the VTS to access with the API call.
Response Format	JSON
Required Authentication	A valid user name and password must be passed as an argument. The values are case sensitive.

Table 15: detokenize Request Parameters

Parameter	Required	Type	Description
`token`	Y	string	The token to detokenize. **token** is case-sensitive. **Example value**: 6029541314537206

Table 15: detokenize Request Parameters

Parameter	Required	Type	Description
`tokengroup`	Y	string	Defines a group name space in the configuration database. For example, let's say you are a credit card company and you want the VTS to support three departments supporting Store1, Store2, and Store3. You want employees in these departments to have access to data for those respective stores. Two actions are required: • The application code must specify a different `tokengroup` parameter in the POST calls for each group supported (example: `Store1`, `Store2`, and `Store3`). The `tokengroup` parameter is case-sensitive. • The VTS GUI administrator must create token groups that match the `tokengroup` parameter names used in the application code. If you do not support multiple token groups, then `tokengroup` will always be the same value. **Example value**: `store1`
`tokentemplate`	Y	string	VTS GUI Administrator-defined name for a group of properties that define tokenization operation. Properties include the token group to which the template applies, the tokenization format (either FPE or FPE with Luhn check, or FF1 or FF1 with Luhn check), number of leftmost or rightmost characters to *not* tokenize, whether you wish to never detokenize a tokenized entry, the character set used for tokenization, and an optional prefix for tokens. **Example value**: `Credit Card`

Table 16: Detokenize Response Parameters

Parameter	Type	Description
`data`	string	The detokenized sensitive data. May be masked depending on the configuration. **data** is case sensitive and limited to 128KiB characters.
`status`	string	**Succeeded** (if request successful) or **Error** (if request failed).
`reason`	string	If status is **Error**, then a **reason** will be displayed. Otherwise the **reason** will be omitted.

Example:

NOTE: If you use the `--tlsv1.2` CURL option, you must use CURL version 7.34 or higher.

Request:

```
curl -k -X POST -u tok-user1:Panda45# -d'{"tokengroup" : "vtsUsers-t" ,
"token" : "CC-0332335756020172", "tokentemplate" : "vtsUsersTemplate" }'
https://192.168.118.140/vts/rest/v2.0/detokenize
```

If the client authentication feature is enabled (See also "Upgrade" on page 24.) the example syntax is as follows:

```
curl -k --key client.key --cert client.crt -X POST -u tok-
user1:Panda45# -d'{"tokengroup" : "vtsUsers-t" , "token" :
"CC-0332335756020172", "tokentemplate" : "vtsUsersTemplate"
}' https://192.168.118.140/vts/rest/v2.0/detokenize
```

Response:

```
{"data":"9453677629008564","status":"Succeed"}
```

Detokenizing Multiple Data Items

Up to 1,000 data items can be detokenized in a single curl command by adding multiple data items. For example, to detokenize four data items, you might enter the following in the posttoken.txt file:

Example:

Request:

```
cat posttoken.txt

[{"tokengroup" : "vtsUsers-t" , "token" : "CC-0332335756020172",
"tokentemplate" : "vtsUsersTemplate" },{"tokengroup" :
"vtsUsers-t" , "token" : "CC-0332335756020172", "tokentemplate"
: "vtsUsersTemplate" },{"tokengroup" : "vtsUsers-t" , "token" :
"CC-0332335756020172", "tokentemplate" : "vtsUsersTemplate"
},{"tokengroup" : "vtsUsers-t" , "token" : "CC-
0332335756020172", "tokentemplate" : "vtsUsersTemplate" }]

curl -k -X POST -u tok-user1:Panda45# --data-binary
@posttoken.txt https://192.168.118.140/vts/rest/v2.0/detokenize
```

Response:

```
[{"data":"9453677629008564","status":"Succeed"},{"data":"9453677
629008564","status":"Succeed"},{"data":"9453677629008564","statu
s":"Succeed"},{"data":"9453677629008564","status":"Succeed"}]
```

The detokenized data is returned in the CURL response in corresponding order of submission.

NOTE: Newlines are NOT allowed in the input.

IMPORTANT: If FPE-luhn, FF1-luhn, or Random-luhn is selected in the Change Token Template page of the VTS GUI, the data must pass a Luhn check or detokenization will fail with the error message:
`{"status":"error","reason":"Luhn check failed for input data."}`

Set Log Levels

Description

Set the log level for the REST API.

Table 17: Log Endpoint Details

Data	Description
HTTP Request	`https://VTS_IP_Address/vts/rest/v2.0/log`
Method	POST
VTS IP Address	The IP address of the VTS to access with the API call.
Response Format	JSON
Required Authentication	None

Table 18: Log Request Parameter

Parameter	Required	Type	Description
`loglevel`	Y	Enum	`loglevel` can take one of the following values: • `error` - Default value used in normal production environment. • `debug` - Use only when directed by support. • `info` - Do not use unless maximum logging is required. NOTE: The performance of tokenization and detokenization operations will be severely affected by a log level of “debug” and substantially affected by a log level of “info.” For maximum throughput, use a loglevel of “error.” **Example**: debug

Table 19: Log Response Parameters

Parameter	Type	Description
status	string	`Success` (if request successful) or `Error` (if request failed).

Example:

NOTE: If you use the `--tlsv1.2` cURL option, you must use CURL version 7.34 or higher.

Request:

```
curl -k --tlsv1.2 -X POST https://127.0.0.1/vts/rest/v2.0/log --data-
binary '{"loglevel":"debug"}'
```

Response:

```
{"status":"success"}
```

Tokenization Examples

Java Client Example

```
package com.vormetric.app;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;

import org.apache.commons.codec.binary.Base64;

import com.jayway.jsonpath.*;

import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLSession;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;

public class VTSClientSample {

public static void main(String[] args) {
        new VTSClientSample().DoIt();
}
 @SuppressWarnings("null")
private void DoIt(){
   String credential = Base64.encodeBase64String("vtsuser1:Panda45".getBytes());
   String ccn[] = {
   "5471949763376677","5545127221796024","5353800637453171","5139918930790601",
"5267003853942275"
   };
try {

// Tokenize request
String https_url = "https://vts-qa/vts/rest/v2.0/tokenize/";
URL myurl = new URL(https_url);
HttpsURLConnection con = (HttpsURLConnection)myurl.openConnection();
String ccNum = "9453677629008564";
String jStr =
"{\"data\":\""+ccNum+"\",\"tokengroup\":\"t1\",\"tokentemplate\":\"Credit
Card\"}";
con.setRequestProperty("Content-length", String.valueOf(jStr.length()));
con.setRequestProperty("Content-Type","application/json");
con.setRequestProperty("Authorization","Basic "+credential);
con.setRequestMethod("POST");
con.setDoOutput(true);
con.setDoInput(true);
DataOutputStream output = new DataOutputStream(con.getOutputStream());
output.writeBytes(jStr);
output.close();
BufferedReader rd=new BufferedReader(new InputStreamReader(con.getInputStream()
));
```

```
tring line = "";
String strResponse = "";
while ((line = rd.readLine()) != null) {
strResponse=strResponse+line;
}
rd.close();
String token = JsonPath.read(strResponse, "$.token").toString();
con.disconnect();
System.out.println("Tokenize server: "+https_url);
System.out.println("Tokenize request: "+jStr);
System.out.println("Tokenize response: "+strResponse);
// Bulk tokenize
String jStrArray ="[";
for (int i=0;i<ccn.length;i++) {
jStrArray = jStrArray +
"{\"data\":\""+ccn[i]+"\",\"tokengroup\":\"t1\",\"tokentemplate\":\"Credit
Card\"}";
if (i<ccn.length-1) {
jStrArray = jStrArray + ",";
}
}
jStrArray=jStrArray+"]";

con = (HttpsURLConnection)myurl.openConnection();
con.setRequestProperty("Content-length", String.valueOf(jStrArray.length()));
con.setRequestProperty("Content-Type","application/json");
con.setRequestProperty("Authorization","Basic "+credential);
con.setRequestMethod("POST");
con.setDoOutput(true);
con.setDoInput(true);
output = new DataOutp
```

C# Client Example

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Text;
using System.Net;
using System.IO;
using System.Threading.Tasks;
using RestSharp;
using RestSharp.Authenticators;
using System.Windows.Forms;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace VTS2ClientSample
{
   class Program
   {
static public bool MyRemoteCertificateValidationCallback(System.Object
sender, X509Certificate certificate, X509Chain chain, SslPolicyErrors
sslPolicyErrors)
      {
         return true; // unsafe! Do not do this in production!
      }

      static void tokenize(RestClient client)
      {
         var request = new RestRequest("vts/rest/v2.0/tokenize",
Method.POST);
         request.RequestFormat = DataFormat.Json;
         request.AddBody(new { tokengroup = "t1", tokentemplate = "Credit
Card", data = "1234567812345670" });
         // execute the request
         IRestResponse response = client.Execute(request);
         var responsestatus = response.ResponseStatus;
         var content = response.Content; // raw content as string

         MessageBox.Show("Tokenize API returns: " + content,
responsestatus.ToString(), MessageBoxButtons.OK);
      }

      static void detokenize(RestClient client)
      {
         var request = new RestRequest("vts/rest/v2.0/detokenize",
Method.POST);
         request.RequestFormat = DataFormat.Json;
         request.AddBody(new { tokengroup = "t1", tokentemplate = "Credit
Card", token = "1234567812345670" });
         // execute the request
         IRestResponse response = client.Execute(request);
         var responsestatus = response.ResponseStatus;
         var content = response.Content; // raw content as string
```

```
 MessageBox.Show("Detokenize API returns: " + content,
responsestatus.ToString(), MessageBoxButtons.OK);
      }

      static void Main(string[] args)
      {
         ServicePointManager.SecurityProtocol =
SecurityProtocolType.Tls12;
         ServicePointManager.ServerCertificateValidationCallback =
MyRemoteCertificateValidationCallback; // unsafe! Do not do this in
production!

         var client = new RestClient("https://10.10.10.10");
         client.Authenticator = new HttpBasicAuthenticator("username",
"password");

         tokenize(client);

         detokenize(client);
      }
   }
}
```

Tokenization API Call Failure Error Messages and Workarounds

These error messages may occur when an Tokenization API call is made.

Table 20: Error Message Reference

Message	Description
--tlsv1.2: is unknown	Only CURL version 7.34 or later supports the --tlsv1.2 option. If you get this error message, upgrade to CURL version 7.34 or higher.
API call aborted and no JSON vector elements are processed	These are error messages which pertain to the whole JSON vector. In other words, in these cases no vector elements are processed at all. The whole operation is aborted
User name and/or password mismatch.	Due to a wrong user name and/or password, access to the Tokenization Server is denied.
Unknown JSON keyword encountered. The valid keyword for the log request is loglevel.	For the REST API called "`log`", the only valid keyword on the left of the JSON `valuename-value` tuple is "`loglevel`".
Could not parse JSON input, perhaps it is too long.	An input size of 1 MByte cannot be exceeded. It could also mean that the JSON input is malformed.
User name and/or password missing	Either the user name, or the password, or both are missing from the request.
This user does not have permission to tokenize.	While the user exists and the password is correct, this particular user does not have the permission to tokenize. Consider assigning this user the permission to tokenize via the GUI.
User name found, but the user is not marked active.	While the user exists and the password is correct, this particular user is not marked as an active user. Consider marking this user an active user in the GUI.
Invalid operation. Use `/vts/rest/v2.0/tokenize` or `/vts/rest/v2.0/detokenize`	An invalid URL was specified. Allowed operations are `/vts/rest/v2.0/tokenize`, `/vts/rest/v2.0/detokenize`, or `/vts/rest/v2.0/log`.
No password specified.	While a user name was specified, the password is missing.
User name not found.	The specified user name exists neither in the local database which is configured via the GUI, nor in the optional LDAP database.
LDAP user found in local database, but LDAP is disabled.	Presumably due to a prior use of LDAP, an LDAP user was cached in the local database, but LDAP has since been disabled, disallowing authentication with this user. Prior to using LDAP users for authentication, at least one of the groups this user belongs to according to the LDAP database must be configured as a user group via the GUI.

Table 20: Error Message Reference (Continued)

Message	Description
API call aborted, one JSON vector element fails, others are processed	These messages pertain to only one element of the JSON vector. In these cases, one vector element may result in an error, but all other vector elements are still processed.
The length of the token exceeds 128 Kbytes.	The length of a token is limited to 128 KiB. However, the token supplied was longer than 128 KiB.
Unknown JSON keyword encountered. Valid keywords for the detokenize request are tokengroup, token and tokentemplate.	For the detokenize operation, only three specific keywords are allowed on the left side of a JSON tuple: "tokengroup", "token", and "tokentemplate". Please review and correct your JSON string.
The length of the data exceeds 128 Kbytes.	The length of the input data is limited to 128 KiB. However, the input data supplied was longer than 128 KiB.
Unknown JSON keyword encountered. Valid keywords for the tokenize request are tokengroup, data and tokentemplate.	For the tokenize operation, only three specific keywords are allowed on the left side of a JSON tuple: • `tokengroup` • `data` • `tokentemplate`
Please review and correct your JSON string.	A token group was specified in the JSON input, but it doesn't exist in the GUI-administered configuration database. Consider adding this token group via the GUI.
The requested key name was not found on the GDE Appliance	A key name was configured via the GUI and now used in an operation, however this key name does not exist on the GDE Appliance.
The token group specified was not found in the database	Consider creating this key name on the GDE Appliance.
tokengroup not specified in JSON request.	The JSON tuple named "`tokengroup`" is missing from a JSON vector element. A tokengroup must be specified.
The requested combination of tokentemplate and tokengroup was not found in the tokentemplate table.	Either the tokengroup specified has not been entered via the GUI, or the tokentemplate has not been entered via the GUI, or the token template of the same name has been entered, but for a different tokengroup. Please ensure that the tokentemplate you are trying to use has been entered for the tokengroup you are trying to use.
It is not allowed to specify the FPE-luhn token format with a non-zero keepright value.	After accounting for `keepleft` and `keepright`, not enough input characters are left to successfully tokenize the input. When you use the FPE token format, the number of input characters must exceed the sum of the Keep Left value and the Keep Right value by at least two. When you use the FPE-luhn token format, the number of input characters must exceed the sum of the Keep Left value and the Keep Right value by at least three.

Table 20: Error Message Reference (Continued)

Message	Description
There are not enough input characters for the detokenize operation.	At least two input characters are needed in the token in order to try to detokenize it.
Token format FPE-luhn is only compatible with an alphabet consisting of the 10 decimal digits (in ASCII encoding).	The FPE-luhn token format cannot be used with an alphabet containing letters and/or special characters. It must be used with an alphabet only containing the 10 digits.
After accounting for the keepleft and keepright requirements, not enough input characters left to tokenize.	When you use the FPE token format, the number of input characters must exceed the sum of the keepleft value and the keepright value by at least two.
Luhn check failed for input data.	The token format FPE-luhn was specified, but the input data sports an incorrect Luhn checksum. Please check your input data, or change the token format from FPE-luhn to FPE.
After accounting for keepleft and keepright, there are not enough input characters present to successfully tokenize.	When you use the FPE token format, the number of input characters must exceed the sum of the keepleft value and the keepright value by at least two. When you use the FPE-luhn token format, the number of input characters must exceed the sum of the keepleft value and the keepright value by at least three.
Token does not begin with the prefix specified in the database.	While a certain token prefix is specified in the token template which is being used in the JSON input, the token seen in the JSON input does not sport this particular token prefix.
The specified token only contains the prefix and no payload.	While the token seen in the JSON input sports the correct token prefix, there are no other characters present in the token.
Luhn check failed for input token.	The token format FPE-luhn was specified, but the token specified sports an incorrect Luhn checksum. Please check your input data.
There are not enough input characters present to successfully detokenize.	When you use the FPE token format, the number of input characters, not counting the token prefix, must exceed the sum of the keepleft value and the keepright value by at least two. When you use the FPE-luhn token format, the the number of input characters, not counting the token prefix, must exceed the sum of the keepleft value and the keepright value by at least three.

A

VTS CLI Reference

The Vormetric Tokenization Server Command Line Interface (CLI) is used to configure the VTS network and do other system-level tasks.

This appendix documents the following CLI categories and their options:

To access the CLI, see "Access the CLI through a terminal emulator like Putty." on page 27.

For tips on navigating the CLI, see "On first access, you will be prompted to review and sign the License Agreement." on page 27.

Network Management

Use the `network` category to set, modify, or delete system IP addresses, and to set up Vormetric Tokenization Servers.

NOTE: Each server is assigned a unique IP address.

Table 21: Network Category Commands

	Description
`ip`	This category provides CLI commands to configure the VTS network interface settings.
`dns`	This category provides CLI commands to configure the VTS DNS server(s).
`host`	This category provides CLI commands to manage the host configuration via the /etc/hosts file.
`ping`	Pings an IP address, host name, or FQDN.
`traceroute`	Traces route to IP address or host name.
`rping`	Sends an ARP (Address Resolution Protocol) request to a neighbor host.
`arp`	Displays the system ARP cache.
`checkport`	Checks local and remote TCP port status.

Example: Show IP address:[1]

Request:

```
network$ip address show
```

Response:

```
Device        Prefix             Broadcast            Label
eth0          192.168.20.11/16      192.168.255.255         none
eth1          192.168.99.122/16     192.168.255.255         none

Show ip address SUCCESS
```

1. See also: “Configure the Network Interface” on page 111.

Configure the Network Interface

Use the `ip` command to configure the VTS network interface.

Table 22: Network Interface Command Arguments

Command	Description
`link`	Sets the physical components of the network interface, such as connection speed, mode, and MTU.
`address`	Initializes or sets the IP address for a network interface.
`route`	Configures network routing for Token Server.
`diag`	Initiate LINUX "`ip`" commands to diagnose network issues.

Configure the IP Address

Use the `address` argument to initialize (set to default) or set different addresses on the interface.

Example: Syntax

```
ip address {init|set} ip_address/address_prefix_length dev {eth0 |
eth1} [label {diag | this}]
```

```
ip address {show|flush} {eth0|eth1} [label {diag|this}]
```

Table 23: Address Command Parameters

Parameter	Description
`init`	Sets the system interfaces to the original settings from manufacturing.
`set`	Sets an IP address on a specific network interface.
`dev`	Specifies the network interface to which to apply the IP address. The network interfaces are eth0 and eth1.
`flush`	Removes the IP addresses on the specified interface.
`init`	IP address for the device that is being added; the value can be: 0-32.
`label`	Specifies that a label is being applied to this interface assignment.
`show`	Displays the current addresses on the interfaces.
`this`	Sets the interface to use a dynamic IP address.

Table 23: Address Command Parameters (Continued)

Parameter	Description
`diag`	Sets the interface to use a fixed (static) IP address.

Example: Change the eth0 IP Address:

1.Initiate the `network` category CLI options:

```
vormetric$ network
```

2. Call the existing IP address:

```
network$ ip address init 192.168.99.77/16 dev eth0
```

Example: Set the IP address for the `eth1` network interface and assigns it a new IP address:

```
network$ ip address set 192.168.99.122/16 dev eth1 label diag
```

Example: List all assigned IP addresses:

Request:

```
network$ ip address show
```

Response:

```
Device         Prefix             Broadcast              Label
eth0           192.168.20.11/16     192.168.255.255            none
eth1           192.168.99.122/16    192.168.255.255            none

Show ip address SUCCESS
```

NOTE: This command is useful for verifying device network address configuration updates.

Run Diagnostics (`diag`)

The `ip diag` command is the equivalent of running the Linux ip command. Check the ip man pages on a Linux system for details.

Syntax:

```
ip diag [option]
```

Example: Show Diagnostics

Request:

```
network$ ip diag -V
```

NOTE: -V is a CLI flag to return a verbose response. This flag is required to return the details shown in the response example.

Response:

```
ip utility, iproute2-ss061002
ip diag SUCCESS
```

Example: Show multicast addresses

Request:

```
network$ ip diag maddr show
```

Response:

```
1: lo
   inet  224.0.0.1
   inet6 ff02::1
2: eth0
   link  33:33:00:00:00:fb
   link  01:00:5e:00:00:fb
   link  01:00:5e:00:00:01
   link  33:33:ff:60:f9:3e
   link  33:33:00:00:00:01
   inet  224.0.0.251
   inet  224.0.0.1
   inet6 ff02::fb
   inet6 ff02::1:ff60:f93e
   inet6 ff02::1
ip diag SUCCESS
```

Configure IP Links

The `ip link` command establishes how the various interfaces connect to the other nodes in the network. `ip link` is used to specify the bandwidth of the `eth0` and `eth1` interfaces and sets the Maximum Transmission Unit (MTU).

Example Syntax

```
ip link set (eth0|eth1) speed
(auto|10mb_half|10mb_full|100mb_half|100mb_full|1000mb_half|1000mb_ful
l)
```

```
ip link set {eth0|eth1} [mtu {100…1500}] [{up|down}]
[speed{auto|10mb_half|10mb_full|100mb_half|100mb_full|1000mb_half|1000
mb_full}]
```

```
ip link show [eth0|eth1]
```

Table 24: IP Link Command Arguments

	Description
`eth0`	Network interface card 1.
`eth1`	Network interface card 2.
`mtu`	Sets the Maximum Transmission Unit value. The default MTU is 1500.
`set`	Enables the parameter settings below for the ip link command.
`show`	Displays information about the IP link connections.
`speed`	Sets the link speed of the interface.

NOTE: Use auto detect to set the data rate of all interfaces and set the MTU value to the default (1500).

Example: Configure and activate an ethernet interface

The following example configures the `eth1` interface to operate at 100 Mb/s, in full-duplex mode, and activates the interface for network accessibility.

Request:

```
network$ ip link set eth1 speed 100mb_full
```

Response:

```
ip link speed SUCCESS
```

Example: Show IP all system IP links

Request:

```
network$ ip link show
```

Response:

```
Device  State   MTU     Mediatype       Speed
eth0    UP      1500    copper
eth1    DOWN    1500    copper

SUCCESS: show ip link
```

Configure IP Routes

Set up IP routes using the `ip route` command. If the `eth0` and `eth1` interfaces are set on the same subnet, a netmask is optional. If they are on separate subnets, include the netmask for the other subnet.

NOTE: Configure a default route connection outside of the subnet.

Syntax:

```
ip route {add|delete|replace} [ip_prefix|default]
[via {ipaddress}] [dev {eth0|eth1}] [src {ipaddress}]
```

Table 25: IP Route Command Arguments

	Description
`add`	Adds a static route.
`delete`	Deletes a static route.
`replace`	Changes the table, gateway and/or source of an existing IP route.
`show`	Displays all the currently configured routes.

Example: Show IP Routes

Request:

```
network$ ip route show
```

Response:

```
Main routing table
192.168.0.0/16 dev eth0  proto kernel  scope link  src 192.168.20.11
192.168.0.0/16 dev eth1  scope link  src 192.168.99.122
169.254.0.0/16 dev eth0  scope link  metric 1002
169.254.0.0/16 dev eth1  scope link  metric 1003
default via 192.168.0.1 dev eth0
ip route show SUCCESS
```

Example: Delete an IP route and displays the results

Request:

```
network$ ip route del 192.168.0.0/16 dev eth1
```

Response:

```
ip route SUCCESS
```

Example: Show routing table

Request:

```
network$ ip route show
```

Response:

```
Main routing table
192.168.0.0/16 dev eth0  proto kernel  scope link  src 192.168.2011
169.254.0.0/16 dev eth0  scope link  metric 1002
169.254.0.0/16 dev eth1  scope link  metric 1003
default via 192.168.0.1 dev eth0
ip route show SUCCESS
```

Example: Add an IP route to the `eth1` interface and displays the result:

1.Add the IP Route:

Request:

```
network$ ip route add 192.168.0.0/16 dev eth1
```

Response:

```
ip route SUCCESS
```

2. Show the result:

Request:

```
network$ ip route show
```

Response:

```
Main routing table
192.168.0.0/16 dev eth1  scope link
192.168.0.0/16 dev eth0  proto kernel  scope link  src 192.168.20.11
169.254.0.0/16 dev eth0  scope link  metric 1002
169.254.0.0/16 dev eth1  scope link  metric 1003
default via 192.168.0.1 dev eth0
ip route show SUCCESS
```

Configure the Default Route

A default route specifies the gateway to which IP packets are sent when the local routing table is unable to resolve a destination.

NOTE: Always configure a default route.

Example: Configure a default route on the `eth0` interface:

Request:

```
network$ ip route add default via 192.168.0.254 dev eth0
```

Response:

```
WARNING: Changing static network ip route requires network
configuration to be restarted.
```

Type `yes` to confirm the route ip change:

```
Continue? (yes|no)[no]:yes
```

```
Successfully updated network ip routing gateway.
ip route SUCCESS
```

Type `yes` to restart :

```
Do you want to restart networking interface (yes|no)[no]:yes
```

```
Shutting down interface eth0:                               [  OK  ]
Shutting down interface eth1:                               [  OK  ]
Shutting down loopback interface:                           [  OK  ]
Bringing up loopback interface:                             [  OK  ]
Bringing up interface eth0:  Determining if ip address 192.168.20.11
is already in use for device eth0...
                                                            [  OK  ]
Bringing up interface eth1:  Determining if ip address 192.168.99.122
is already in use for device eth1...
                                                            [  OK  ]
SUCCESS: Restart Networking Interface Successfully.
```

NOTE: Ignore the separation of routes into unique tables. All routes are considered members of the main routing table, as reflected in the Management Console. Separate routing tables have been deprecated.

Display Assigned IP Routes

The `ip route show` command displays the IP routes that have been assigned to the system. Use the `ip route show` command to verify the changes you made to the IP route tables:

Example:

Request::

```
network$ ip route show
```

Response:

```
Main routing table
192.168.0.0/16 dev eth0  proto kernel  scope link  src
192.168.20.11
192.168.0.0/16 dev eth1  proto kernel  scope link  src
192.168.99.122
169.254.0.0/16 dev eth0  scope link  metric 1002
169.254.0.0/16 dev eth1  scope link  metric 1003
default via 192.168.0.254 dev eth0
ip route show SUCCESS
```

Set DNS Domain Name Servers

The `dns` command sets the DNS domain servers used by the VTS.

NOTE: This is equivalent to editing the `/etc/resolv.conf` file.

Syntax

```
dns [search domainname] [dns1 ip] [dns2 ip] [dns3 ip]
dns [search domainname]
dns [dns1 ip] [dns2 ip] [dns3 ip]
dns clear
dns show
```

The `dns` command includes the following elements:

Table 26: DNS Command Arguments

Parameter	Description
`clear`	Removes all the DNS settings.
`dns1`	Specifies settings for domain server 1.
`dns2`	Specifies settings for domain server 2.
`dns3`	Specifies settings for domain server 3.
`search`	Sets the domain server name.
`domainname`	The name of the domain server.
`ip`	The IP address for the specified domain server.
`show`	Show all the currently configured Domain Name Servers.

NOTE: Configure just the DNS server name, just the DNS server IP addresses, or both the DNS server name and IP addresses.

Example: Configure the domain and IP address and lookup

The following example sets the domain to `i.vormetric.com` and the `dns1` lookup IP address to `192.168.2.254:`

Request:

```
network$ dns search i.vormetric.com dns1 192.168.2.254
```

Response:

```
DNS SUCCESS
```

Example: Use the `dns show` command to confirm the configuration:

Request:

```
network$ dns show
```

Response:

```
search i.vormetric.com
nameserver 192.168.2.254
DNS show SUCCESS
```

Example: Use the `dns clear` command to remove all DNS settings:

Request:

```
network$ dns clear
```

Response:

```
dns SUCCESS
```

Configure Hosts

Use the `host` command VTS to add and remove static IP addresses to and from th VTS `/etc/hosts` file. By default, only hosts with resolvable host names or FQDNs can be configured in the Tokenization Server database.

Table 27: Host Command Arguments

	Description
`add`	Inserts the *`ipAddress:host`* pair in `/etc/hosts`.
`delete`	Removes the *`ipAddress:host`* pair from `/etc/hosts`.
`show`	Shows the `/etc/hosts` file except for blank lines, comment lines. Displayed entries are not sorted.

Add IP Address

Use this command to add a host to the VTS. When executed he record is written to the VTS `etc/hosts` file.

Syntax:

```
host add name ipAddress
```

Run the `host` allows the VTS to communicate with other VTS and hosts without using DNS.

NOTE: VTS Administrators cannot edit system files directly. All host file updates are written via the host command and arguments.

Run the host command to name a host with a valid network host name without DNS so that the network host name resolves to a valid IP address.

Delete a Host

```
host delete name
```

Show Configured Hosts

```
host show
```

`name` is the host name of a host, and `ipAddress` is the IP address to use to contact that host.

NOTE: Host names cannot contain spaces and IP addresses must be in the standard *xxx.xxx.xxx.xxx* format.

Example: Add IP Address

The following example adds a *`ipAddress:host`* pair to the `/etc/hosts` file and displays all configured *`ipAddress:host`* pairs.

Request:

```
network$ host show
name=localhost6.localdomain6

SUCCESS: show host
```

Response:

```
SUCCESS: add host
network$ host show
name=localhost6.localdomain6

SUCCESS: show host
```

The following example deletes a host from the `/etc/hosts` file:

```
SUCCESS: deleted host
0012:network$ host show
name=localhost6.localdomain6
SUCCESS: show host
```

Ping

The `ping` command sends ICMP (Internet Control Message Protocol) echo request packets (`ECHO_REQUEST`) to a specified network host. The `ping` command uses the ICMP protocol's mandatory echo request datagram to elicit an ICMP echo response (`ECHO_RESPONSE`) from a host or gateway. The `ping` command sends six packets to the network host and then reports the results.

Syntax

```
ping {ipAddress | FQDN}
```

Example

The following example sends a `ping` request to the host `vmlinux04_RH5`:

```
network$ ping www.vormetric.com
PING www.vormetric.com (50.57.226.77) 56(84) bytes of data.
64 bytes from 50-57-226-77.static.cloud-ips.com
(50.57.226.77): icmp_seq=1 ttl=48 time=190 ms
64 bytes from 50-57-226-77.static.cloud-ips.com
(50.57.226.77): icmp_seq=2 ttl=48 time=187 ms
...
--- www.vormetric.com ping statistics ---
7 packets transmitted, 6 received, 14% packet loss, time
6152ms
rtt min/avg/max/mdev = 143.625/178.151/190.223/15.776 ms
ping SUCCESS
```

Track a Route Package

The Internet is a large and complex aggregation of network hardware that is connected together by gateways. Tracking the route a packet follows (or finding a gateway that is discarding your packets) can be difficult.

The `traceroute` command uses the IP-protocol time field to elicit an ICMP time exceeded (`TIME_EXCEEDED`) response from each gateway along the path to a specified host.

Specify the target IP address or FQDN. The `traceroute` command supports a timeout option.

Syntax

```
traceroute (ipAddress|FQDN) [timeout 1-60]
```

Example: The following example sends a traceroute command request to an IP address:

```
network$ traceroute 192.168.60.7
traceroute to 192.168.60.7 (192.168.60.7), 30 hops max, 40 byte
packets
1 192.168.244.3  3000.605 ms !H  3000.571 ms !H  3000.548 ms !H
Traceroute Completed
```

Send ARP Request

The `rping` command sends Address Resolution Protocol (ARP) requests to a neighbor host, pings the address on the device interface by ARP packets and informs how many users are using a particular IP address.

Syntax

```
rping ipAddress {eth0|eth1}
```

Example

```
network$ rping 192.168.244.4 eth0
ARPING 192.168.244.4 from 192.168.244.3 eth0
Unicast reply from 192.168.244.4 [00:0C:29:36:9E:B3]  2.518ms
Unicast reply from 192.168.244.4 [00:0C:29:36:9E:B3]  0.817ms
Unicast reply from 192.168.244.4 [00:0C:29:36:9E:B3]  0.866ms
Sent 3 probes (1 broadcast(s))
Received 3 response(s)
Arping SUCCESS
```

Display ARP Cache

The `arp` command displays the current Address Resolution Protocol (ARP) cache of the Tokenization Server.

Syntax

```
arp
```

Example: Displays the current ARP cache:

```
network$ arp
192.168.60.25 dev eth0 lladdr 00:08:a1:59:c1:cc REACHABLE
192.168.0.254 dev eth0  FAILED
192.168.110.104 dev eth0 lladdr 00:17:31:6f:58:16 STALE

link info
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue \
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
qlen 1000\    link/ether 00:0c:29:60:f9:3e brd ff:ff:ff:ff:ff:ff
3: sit0: <NOARP> mtu 1480 qdisc noop \    link/sit 0.0.0.0 brd 0.0.0.0
arp SUCCESS
```

If a connection is STALE, ping it and check again. It should change to REACHABLE. If it does not change, or it changes to FAILED, the connection is no longer available.

Scan Network Port

The `checkport` command is used to scan a port on a network-accessible system to verify that a TCP connection can be made to the system using the specified port. It does not guarantee that you can log on, just that a communication channel can be opened on the VTS or on a host. It is typically used to check the status and availability of the ports through which to administrate and run Vormetric Data Security. These are ports like 22, 7024, 8443, 8444, and 8445. The `checkport` command returns the transport layer protocol and the service using that port. The transport layer protocol is always TCP. The service is a system service like ssh, vmsvc, and *.

A “`connection refused`” message can be returned for various reasons, such as a port is not assigned and/or is not in a LISTEN state.

NOTE: If a Vormetric Data Security port refuses a connection, you must troubleshoot the TCP connection.

The `checkport` activity is logged in the Management Console, and is displayed when operating outside of a domain. A sample *Logs* window entry is shown below.

```
18713  2010-08-27 13:07:11.944 PDT  I  vmSSA05  CLI0003I: [cliadmin] network
checkport vmlinux101 7024
```

When `checkport` is executed in the Management Console interface, rather than on the command line, the log entry is appended with "`timeout` *x*", where *x* is either the value you entered on the command line or the default timeout.

Syntax

```
checkport host port [timeout x]
```

where,

host is an IP address, FQDN, hostname, or even "localhost". Typically, it is a valid Tokenization Server, as configured in the Management Console.

port is a single TCP port number or a range of port numbers. A port number range is a hyphen/dash-separated list and is entered in the form "startnum-endnum". For example, `8440-8449`.

x is an integer between 1 and 600, inclusive. It is the timeout threshold and is expressed in seconds. The default is `180` seconds.

Example

The following example checks the availability of the Vormetric Data Security port (`8445`):

```
network$ checkport vmSSA06 8445
Connection to vmSSA06 8445 port [tcp/*] succeeded!
SUCCESS: invoked checkport(nc) command.
```

The following example checks the availability of the Vormetric Data Security port (`7024`) used to download configuration data to an agent host:

```
network$ checkport solaris120 7024
Connection to solaris120 7024 port [tcp/vmsvc] succeeded!
SUCCESS: invoked checkport(nc) command.
```

The following example checks the availability of a range of ports on the local system, which is also a Tokenization Server, and includes a 10 second timeout.

```
network$ checkport solaris120 7024
Connection to solaris120 7024 port [tcp/vmsvc] succeeded!
SUCCESS: invoked checkport(nc) command.
```

System Category Commands

The `system` configuration category enables you to set the VTS host name, enable/disable the console port, create certificates, restart the VTS, and reboot/shut down the VTS.

Enter the `system` configuration category by typing:

```
vormetric1000$ system
```

Table 28: System Category Commands

`setinfo`	Sets the host name or FQDN of the VTS.
`shutdown`	Stops the Token Server and powers off the appliance.
`reboot`	Reboots the Token Server.
`server`	Provides the options to restart, start, and stop the Token Server as well as the option to check the status of the Token Server.

Define the VTS

The `setinfo` command enables you to set the host name of the VTS and display appliance-related information such as the hardware UUID, serial number, and uptime.

The assigned name is used to identify the appliance and identify the certificate owner. If you change the host name after generating the CA signer and VTS certificates, you must regenerate the certificates because the host name is used in the certificates to identify the VTS.

Syntax

```
setinfo [show | hostname HOSTNAME | sshbanner ]
```

Table 29: Setinfo Command Arguments

	Description
`hostname`	Sets the hostname for your system. This option takes one argument, the network name to assign the appliance.
`sshbanner`	Defines the `/etc/ssh/ssh-banner` file. Available only on Vormetric Tokenization Servers. Edit the banner shown when logging on to the VTS CLI.
`show`	Shows the current `setinfo` settings.

Example: Set the VTS host name to `vmSSA001`:

```
system$ setinfo host vts-683
SUCCESS: setinfo hostname. If the Security Server certificate is already
generated, please re-sign the server certificate to reflect the hostname
changes.
```

Get VTS Information

The `setinfo show` command displays general appliance information. The following example was created on a Tokenization Server.

Request:

```
system$ setinfo show
```

Example Response:

```
hostname = vts-683
UUID = 564DC6B2-F953-F395-DC61-9C9E833FC99C
serial number =
part number =
uptime =  21:58:30 up 2 days,  9:17,  2 users,  load average: 0.00, 0.00,
0.00
on virtual machine = true
ssh banner =
Show setinfo SUCCESS
```

System Management

VTS CLI administrators can start and stop the VTS based on the need for maintenance intervals, test cycles and other system administration tasks.

Table 30: System Management Commands

Parameter	Description
`restart`	Restarts the VTS.
`start`	Start the VTS.
`stop`	Stops the VTS.
`status`	Display the VTS running status.

Shutdown The VTS

The `shutdown` command stops the VTS virtual machine and saves all configuration changes. Upon successful execution of this command the appliance can be safely turned off.

Warning! Use this command successfully before turning off the appliance.

Syntax

```
shutdown
```

Example

The following example shuts the system down:

Request:

```
system$ shutdown
```

Response:

Type `y` when prompted to complete the shutdown event.

```
Do you want to shutdown the system ? (y/n):y
Shutting down now...Type y to confirm the shutdown request.
Shutdown SUCCESS
```

Restart the VTS

The reboot command reboots the VTS virtual machine.

NOTE: See "Restart the VTS" for additional restart options.

Syntax

```
reboot
```

Example

The following example reboots the system immediately:

Request:

```
system$ reboot
```

Response:

Type `y` when prompted to complete the restart event.

```
Reboot the system y/n
Rebooting now...
Reboot SUCCESS
system$
Broadcast message from root (Sun Feb 9 02:44:20 2014):
The system is going down for reboot NOW!
```

Cluster Management

The `cluster` category is a subset of VTS commands for use with cluster management.

Table 31: Cluster Management Commands

Command	Description
`add_node`	Add a node that can later create or be joined to an existing cluster. This command must be used on a node before that node can join a cluster.
`create`	Create a VTS cluster.
`join`	Join a node to an existing VTS cluster.
`remove_node`	Remove a node from a cluster.
`show`	Show cluster settings.

Add Cluster Nodes

This command allows other nodes in the cluster to communicate with this node. Use this command on a new VTS node before creating (`cluster$ create`) or joining (`cluster$ join`) a VTS cluster. his command must be run *on* each node in the cluster *for* each node in the cluster.

Syntax

```
add_node node_IP_address
```

Example

The following examples shows the process to add three nodes to a cluster using the `add_node` command:

Node 1:

```
cluster$ add_node node2_IP_address
cluster$ add_node node3_IP_address
cluster$ add_node node1_IP_address
```

Node 2:

```
cluster$ add_node node1_IP_address
cluster$ add_node node3_IP_address
cluster$ add_node node2_IP_address
```

Node 3:

```
cluster$ add_node node1_IP_address
cluster$ add_node node2_IP_address
cluster$ add_node node3_IP_address
```

Create a VTS Cluster

Creates a VTS cluster. Run only once on the first cluster node. Run `add_node` on the node before running `create`.

Syntax

```
create node_IP_address
```

Example

```
cluster$ create 192.168.55.55
Stopping postgresql-9.4 service: [ OK ]
Saving config files
Initializing database ... OK
. . .
Would you like to create one now? (yes/no): yes    (type yes here)
Username (leave blank to use 'root'): vtsroot    (login name for VTS GUI)
Email address:
Password:   (password for VTS GUI)
Password (again):   (repeat password for VTS GUI)
Superuser created successfully.
Pre-populating vts database...
. . .
Create cluster SUCCESS.
```

Join Nodes to an Existing Cluster

Joins a node to an existing VTS cluster. You must run `add_node`[1] before running `join`.

Syntax

```
join IP_address_of_another_cluster_node
```

Example

```
cluster$ join 192.168.55.56
```

Remove Cluster Node

Use the `remove_node` command to remove a node from a VTS cluster.

If a node has crashed and has been recreated, it can be joined back to the cluster by running the join command. The node rejoining must first be removed from the cluster before it can be joined. For more information see See “Join Nodes to an Existing Cluster” on page 131.

1. See “Add Cluster Nodes” on page 129.

Syntax

```
remove_node Ip_address
```

NOTE: This command must be run on all cluster nodes.

Example

```
cluster$ remove_node 192.168.55.56
```

Show Nodes

The `show` command returns information about the cluster nodes.

Syntax:

```
show {nodename|nodes|connections|replication_status|
replication_slots|conflict_status}
```

Table 32: Cluster Show Command Arguments

Parameter	Description and Usage Notes
`nodename`	Supply the string that uniquely identifies the node.
nodes	List all nodes assigned to the cluster.
connections	List all connections.
replication_status	Show replication status.
replication_slots	Show replication slots.
conflict_status	Show conflict status.

Example: Show Nodes

Request:

```
cluster$ show nodes
```

Response:

```
node_name  | node_sysid          | node_status | node_local_dsn
------------+---------------------+-------------+-----------------
 10-3-2-242 | 6249025496938457592 | r           | host=192.168.2.242
 10-3-2-245 | 6249084678005133436 | r           | host=192.168.2.245
 10-3-2-247 | 6249123651018108642 | r           | host=192.168.2.247
 10-3-2-248 | 6249131625684990743 | r           | host=192.168.2.248
(4 rows)
Show cluster SUCCESS.
(Nodes status: http://bdr-project.org/docs/stable/catalog-bdr-nodes.html)
```

Request:

```
cluster$ show connections
```

Response:

```
 conn_sysid    | conn_is_unidirectional | conn_dsn
-------------------+-----------------------+---------------------
 6249025496938457592 | f                     | host=192.168.2.242
 6249025720274348403 | f                     | host=192.168.2.245
 6249084678005133436 | f                     | host=192.168.2.245
 6249123651018108642 | f                     | host=192.168.2.247
 6249131625684990743 | f                     | host=192.168.2.248
(5 rows)
```

Example: Cluster Status

Request:

```
cluster$ show replication_status
```

Response:

```
client_addr|
application_name                         |   state   |sync_state
-----------+-----------------------------------------+-----------+---------
-
 192.168.2.245|bdr (6249084678005133436,1,16387,):receive| streaming | async
 192.168.2.247|bdr (6249123651018108642,1,16387,):receive| streaming | async
 192.168.2.248|bdr (6249131625684990743,1,16387,):receive| streaming | async
(3 rows)
```

Example: Replication Status

Request:

```
cluster$ show replication_status
```

Response:

```
client_addr|
application_name                          |   state   |sync_state
----------+---------------------------------------+----------+---------
-
 192.168.2.245|bdr (6249084678005133436,1,16387,):receive| streaming | async
 192.168.2.247|bdr (6249123651018108642,1,16387,):receive| streaming | async
 192.168.2.248|bdr (6249131625684990743,1,16387,):receive| streaming | async
(3 rows)
```

Example: Conflict Status

Request:

```
  show conflict_status
```

Response:

```
cluster$
remoteid                                                     |
nr_insert_conflict | nr_update_conflict | nr_delete_conflict
---------------------------------------+------------+------------+
bdr_6249025720274348403_1_16387_16387_ |          0 |           0 |
      0
bdr_6249025966319626654_1_16387_16387_ |          0 |           0 |
      0
bdr_6249027021650246005_1_16387_16387_ |          0 |           0 |
      0
(3 rows)
```

VAE Management Commands

The `vae` category includes commands to register the Tokenization Server to the GDE Appliance..

Table 33: VAE Management Commands

Command	Description
`register`	Register the current VTS host to the GDE Appliance.
`show`	Show registration information.
`test`	Test to be sure the VTS has been successfully registered with the GDE Appliance.
`dsmkeepalive`	Enable/Disable DSM keepalive

Register to the GDE Appliance

Register the current host to the GDE Appliance:

Syntax:

```
register <DSM FQDN>
```

Example Request:

```
vae$ register jdog-dsm-1150.example.com
```

Response:

```
Welcome to the Vormetric Key Agent
Registration Program.

Agent Type: Vormetric Key Agent
Agent Version: 6.0.0.7
```

Show GDE Appliance Host Registration

Display information related to the registration of this host with the GDE Appliance.

Syntax:

```
show
```

Request:

```
vormetric# show
```

Response:

```
VAE has registered to remote DSM Server.
DSM Server Host: jdog-dsm-1150
VAE Password: *********
VAE Password:
Show VAE Configuration SUCCESS
```

test

Test to be sure the VTS has been successfully registered with the GDE Appliance. Also validates that the given key has been successfully created on the GDE Appliance.

Syntax

```
test <keyname>
```

Example

```
vormetric# test key1
Test encryption was successful.
```

GDE Appliance Connection Management

Use the dsmkeepalive command to maintain the connection between the VTS server and the GDE Appliance.

Syntax:

```
dsmkeepalive [enable | disable | status]
```

Following are the possible arguments:

- enable
- disable
- status

Example - Disable a Connection:

Request:

```
dsmkeepalive disable
```

Response:

```
DSM Keepalive is disabled
```

Example - Check Connection Status

Request:

```
dsmkeepalive status
```

Response:

```
DSM Keepalive has been disabled
```

LDAP Authentication

The `auth` category includes commands to set the LDAP Server.

Table 34: LDAP commands

Command	Description
`host`	Set LDAP server hostname or IP address.
`bind`	LDAP User & Credential.
`user_scope`	LDAP User Search Scope.
`group_scope`	LDAP Group Search Scope.
`user_filter`	LDAP User Search Filter.

Table 34: LDAP commands (Continued)

Command	Description
user_prefix	Attribute which should be used to prefix the user name when searching for the user name in the LDAP database.
group_member	LDAP Group Member Selector.
group_objectclass	Attribute which identifies a group of users.
active_user	LDAP Active User Group.
super_user	LDAP Super User Group.
ca-cert	CA Certificate for built-in LDAPS client.
enable	Enable LDAP support.
disable	Disable LDAP support.
show	Show db information.

Configure the LDAP Host

The `host` command allows you to set the LDAP server hostname or IP address.

Syntax

```
host name <ldap_server_uri>[a]
```

a. ***Example: ldap://192.168.118.98:389***>

Example

```
auth$ host ldap://192.168.118.99:389
Set LDAP Server URI SUCCESS
```

Set the Binding DN

Set the binding DN of the user who has access to the AD/LDAP server.

Syntax

```
auth$ bind <FQDN of user with access to LDAP Server.>
```

Example

```
auth$ bind vts0@vts.vormetric.com
Enter Password :
Enter Password again :
Set LDAP Bind User DN SUCCESS
```

Set the User Search Scope

Set the Tokenization Server user search scope.

Syntax

```
user_scope <Token User search scope>
```

Example

```
auth$ user_scope dc=vts,dc=vormetric,dc=com
Set LDAP USER SEARCH SCOPE SUCCESS
```

Set the Group Search Scope

Set group search scope.

Syntax

```
group_scope <Group search scope>
```

Example

```
0018:auth$ group_scope CN=Users,dc=vts,dc=vormetric,dc=com
Set LDAP GROUP SEARCH SCOPE SUCCESS
```

Set the User Search Filter

Set user search filter.

Syntax

```
user_filter <Name of user field in AD/LDAP>
```

Example

```
0019:auth$ user_filter sAMAccountName
Set LDAP USER SEARCH FILTER SUCCESS
```

Set the User Prefix

Attribute which should be used to prefix the user name when searching for the user name in the LDAP database. In some cases, it is ‘cn’, in other cases, it is ‘uid’. In rare cases, it could even be ‘sn’.

Syntax

```
user_prefix <cn or uid or sn>
```

Example

```
0019:auth$ user_prefix uid
Set LDAP USER PREFIX SUCCESS
```

Set Group Member

Set group member filter.

Syntax:

```
group_member <Group member selector. Example: memberUid or member>
```

Example

```
auth$ group_member membUid
Set LDAP GROUP MEMBER SELECTOR SUCCESS
```

Group Objects

The attribute which identifies a group of users. In some cases, it is ‘posixGroup’. In other cases, it is ‘groupOfNames’.

Syntax

```
group_objectclass <posixGroup or groupOfNames>
```

Example

```
auth$ group_objectclass posixGroup
Set LDAP GROUP OBJECTCLASS SUCCESS
```

Set Active User

Set Tokenization Server active user group.

Syntax

```
active_user <Group whose members can access the RESTful API. Example:
vtsUsers>
```

Example

```
auth$ active_user cn=vtsUsers,cn=Users,dc=vts,dc=vormetric,dc=com
Set LDAP ACTIVE USER GROUP SUCCESS
```

Set the Super User Group

Set super user group. Members of this group can change Tokenization Server database schema.

Syntax

```
super_user <Group that can change the Tokenization Server database schema.
Example: vtsAdmin>
```

Example

```
auth$ super_user cn=vtsAdmin,cn=Users,dc=vts,dc=vormetric,dc=com
Set LDAP UPER USER GROUP SUCCESS
```

Manage Root Certificate

Use the `ca-cert` command Import or clear the CA root certificate of LDAP server for the built-in LDAPS client.

Syntax

```
ca-cert clear
ca-cert import
```

Examples:

Clear Certificate

```
auth$ ca-cert clear
Please confirm that you would like to clear LDAP Server CA certificate?
(yes|no):yes
Clear LDAP Server CA Certificate SUCCESS.
```

Import Certificate

```
auth$ ca-cert import
Please confirm that you would like to proceed to import the LDAP Server CA
certificate? (yes|no):yes
Please paste content of the certificate and press CTRL-D when finished.

-----BEGIN CERTIFICATE-----
MIIEczCCA1ugAwIBAgIBADANBgkqhkiG9w0BAQQFADO098EAkGA1UEBhMCR0Ix
EzARBgNVBAgTClNvbWUtU3RhdGUxFDASBgNVBAoTC00EgQFADsTHRkMTcwNQYD
aXR5MRQwEgYDVQQDEwtCZXN0IENBIEx0ZDAeFw0wMDsdff3TUwMTZaFw0wMTAy
MDQxOTUwMTZaMIGHMQswCQYDVQQGEwJHQjETMBEGA1dff3T29tZS1TdGF0ZTEU
. . .
xQUE/C0pWWm6gDkwd5D0DpMDJRqV/weoZ4wC6B73f5uYBcsbLhGYHaXJeSD6Kr
BMAV7Gzdc4VspS6ljrAhbiiawdBiQlQmsBeFz9JkF48BoGNb3l8BoGN+qMa56Y
It8una2gY4l2O//on88r5IWJlm1L0oA8e4fR2yrBHX8BoGNads8BoGNyNrwGi/
7vQMfXdGsRrXNGRGnX+HoX
-----END CERTIFICATE-----
Saving LDAP Server CA Certificate succeed.
Validating CA certificate ....
/tmp/cacert.pem: c = US, ST = CA, L = San Jose, O = Vormetric Inc., OU =
organizationalunit, CN = vts-qa, emailAddress = edleup@vormetric.com
error 18 at 0 depth lookup: self signed certificate
OK
Certificate validation succeed, Import LDAP Server CA Certificate SUCCESS.
```

Enable LDAP Support

Use the `enable` command to activate LDAP support features.

Syntax

```
enable
```

Example

```
auth$ enable
LDAP Server Support Enabled SUCCESS
```

Disable LDAP Support

Use the `disable` command to deactivate LDAP support.

Syntax

```
disable
```

Example

```
auth$ disable
LDAP Server Support Disabled SUCCESS
```

Show LDAP Server Details

Use the `show` command to return information about the configured LDAP server.

Syntax

```
show
```

Example

```
auth$ show
LDAP Enabled true
LDAP Server URI: ldap://192.168.118.98:389
LDAP BIND DN: vts0@vts.vormetric.com
LDAP BIND Password: ************
LDAP USER SEARCH SCOPE: dc=vts,dc=vormetric,dc=com
LDAP GROUP SEARCH SCOPE: CN=Users,DC=vts,DC=vormetric,DC=com
LDAP USER SEARCH FILTER: sAMAccountName
LDAP ACTIVE USER GROUP: CN=vtsUsers,CN=Users,DC=vts,DC=vormetric,DC=com
LDAP SUPER USER GROUP:
Show LDAP Configuration SUCCESS
```

Syslog Configuration

The `remotelog` category includes commands to set the syslog Server.

Table 35: Remotelog Parameters

Parameter	Description
`remote enable|disable`	Enable/disable remote rsyslog server
`host name/ip port`	Set remote rsyslog server and port.
`show`	Show logger information
`up`	Return to previous menu
`exit`	Exit

Enabling the Remote Logging Server

Use the remote command to Enable or disable rsyslog server.

Syntax

```
remote [enable |disable]
```

Example

```
remotelog$ remote enable
Successfully enabled remote syslog.
Shutting down system logger:                            [  OK  ]
Starting system logger:                                 [  OK  ]
```

Configure the Remote Logging Server

Use the `host name/ip port` command to set remote rsyslog server and port.

Syntax

```
host <HOSTNAME/IP] port>
```

Example

```
remotelog$ host 192.168.24.54 1514
Successfully updated rsyslog configuration.
Shutting down system logger:                                [  OK  ]
Starting system logger:                                     [  OK  ]
```

Get Logger Details

Use the `show` command to retrieve logged information.

Syntax

```
show
```

Example

```
vormetric# show
Log Level: INFO
Remote Logger Enable: : false
rsyslogd Host: 192.168.24.54
rsyslogd port: 1514
Show Logger Configuration SUCCESS
```

VTS Maintenance Commands

The `maintenance` category is used to set time parameters, configure GUI login behavior, and upgrade Tokenization Server software.

Enter the `maintenance` category by typing `maintenance` at the VTS CLI command prompt:

```
maintenance$
```

The `maintenance` category consists of the following commands:

Table 36: Maintenance options

Command	Description
`showver`	Displays the VTS versions that are on the system and indicates the version that is currently running.

Table 36: Maintenance options

Command	Description
`ntpdate`	Configures one or more NTP servers with which to synchronize the system clock.
`date`	Sets the system date.
`time`	Sets the system time.
`gmttimezone`	Sets the system time zone.
`diag`	In development.
`vts`	Sets various VTS settings, including rate limiting and login retries; or upgrades VTS software.

Get the VTS Version

The `showver` command displays the version of the VTS.

Syntax

```
showver
```

Example

Enter the `showver` command without any arguments to display installed images and the current image. For example:

```
maintenance$ showver
ver_count=0
cur_ver=show version SUCCESS
```

Configure NTP Synchronization

When NTP is configured and enabled, at one hour intervals the VTS CLI daemon synchronizes the system clock of the VTS with the first available NTP server. If, within one second, the VTS cannot connect with the NTP server, the VTS CLI daemon tries the next NTP server in the list. The NTP server can reside in any time zone.

Syntax

```
ntpdate sync | add server | delete server | on | off | show a
```

a. sync, add, delete, on, off, and show are literals that are entered as shown or in abbreviated form.

- add—adds the named NTP server to the list of servers to contact for time synchronization. At least one server must be configured before you can enable (turn on) time synchronization. You may configure up to four NTP servers.
- delete—removes the named NTP server from the list of servers to contact for time synchronization. Time synchronization is disabled (turned off) when the last NTP server is removed from the list.
- off—disables time synchronization and leaves the current NTP server list intact. You can re-enable synchronization without having to reconfigure the NTP servers.
- on—enables NTP time synchronization. At least one NTP server must be configured before you can enable synchronization.
- *server*—is the network identity (FQDN, hostname, or IP address) of a network accessible NTP server.
  - show–displays the NTP server configuration and state. The ntpdate show command does not sort the output. It displays all the configured NTP servers in the same order that they were added to the Tokenization Server database.
  - sync—forces clock synchronization with the first available NTP server.

Examples

Display the default NTP configuration environment:

```
maintenance$ ntpdate show
Total ntpdate server number  : 0
ntpdate is off
ntpdate SUCCESS
```

Add NTP servers:

```
maintenance$ ntpdate add 192.168.78.100
ntpdate SUCCESS
maintenance$ ntpdate add i.vormetric.com
ntpdate SUCCESS
```

Enables NTP synchronization:

```
maintenance$ ntpdate on
ntpdate SUCCESS
```

Display the fully-configured NTP environment:

```
maintenance$ ntpdate show

Total ntpdate server number  : 4
ntpdate server [1] : 192.168.78.100
ntpdate server [2] : i.vormetric.com
ntpdate server [3] : 192.2.78.100
ntpdate server [4] : 192.168.244.109
ntpdate is on
ntpdate SUCCESS
```

Synchronize the appliance clock with the first available NTP server clock:

```
maintenance$ ntpdate sync
ntpdate SUCCESS
```

Swap the last two NTP servers in the list to change access order:

```
maintenance$ ntpdate delete 192.168.78.100
ntpdate SUCCESS
maintenance$ ntpdate add 192.168.78.100
ntpdate SUCCESS
```

Shows the current NTP configuration status:

```
maintenance$ ntpdate show
Total ntpdate server number  : 4
ntpdate server [1] : 192.168.78.100
ntpdate server [2] : i.vormetric.com
ntpdate server [3] : 192.168.244.109
ntpdate server [4] : 192.168.78.100
ntpdate is on
ntpdate SUCCESS
```

Configure Date Settings

The `date` command in the `maintenance` category is used to set or to display the date on the system. The `date` command without any arguments displays the current system date. If a parameter is included with the `date` command it resets the system date to the specified date.

Syntax

```
date MM/DD/YYYY
date
```

Example

To set the date on the system to December 20th, 2014, enter the following:

```
maintenance$ date 12/20/2014
```

The following example displays the system date:

```
maintenance$ date
month=Dec day=20 year=2014
Show system date SUCCESS
```

Configure Time Settings

The `time` command sets or to displays the time on the system using a 24-hour clock. When no parameters accompany the `time` command, it displays the current system time. If a parameter is included with the time command, it resets the system time to the specified value.

Syntax

```
time HH:MM:SS
time
```

Example

To set the time on the system enter the following:

```
maintenance$ time 02:23:00
```

This sets the system to 2:23 AM.

The following example uses the time command to display the system time:

```
maintenance$ time
hour=18 min=22 sec=38 zone=PDT
Show system time SUCCESS
```

Set the Time Zone

The `gmttimezone` command in the maintenance category is used to set the system time zone. If a parameter is included with the `gmttimezone` command, it sets the time to the zone specified. To see a list of supported time zones, enter `gmttimezone list`.

Syntax:

```
gmttimezone {list|show|set zonename}
gmttimezone list
gmttimezone show
```

Get Timezone List

To list and set the `gmttimezone` on the system enter the following:

```
maintenance$ gmttimezone list
...
(GMT-07:00) America/Phoenix (Mountain Standard Time)
(GMT-07:00) America/Shiprock (Mountain Standard Time)
(GMT-07:00) America/Yellowknife (Mountain Standard Time)
(GMT-08:00) America/Dawson (Pacific Standard Time)
(GMT-08:00) America/Los_Angeles (Pacific Standard Time)
(GMT-08:00) America/Tijuana (Pacific Standard Time)
(GMT-08:00) America/Vancouver (Pacific Standard Time)
(GMT-08:00) America/Whitehorse (Pacific Standard Time)
(GMT-08:00) Pacific/Pitcairn
(GMT-09:00) America/Anchorage
...
```

Configure Timezone:

```
maintenance$ gmttimezone set America/Tijuana
Set timezone SUCCESS
0031:maintenance$ gmttimezone show
Timezone is set to : America/Tijuana
Show timezone SUCCESS
```

Show Configured Timezone:

```
maintenance$ gmttimezone show
Timezone is set to : US/Pacific
Show timezone SUCCESS
```

Get System Diagnostics

The `diag` command in the `maintenance` category displays VTS logs, CPU and memory usage, available disk space, daemon status, OS version, and system uptime.

Syntax

```
diag [log [ list | view LOG_FILE_NAME] | vmstat | diskusage | daemon |
osversion | uptime ]
```

- `log`—shows log file information. Use `log list` to see the available log file names and `log view` to show the contents of a specified log file.
- `vmstat`—shows CPU and memory usage of the Tokenization Server virtual machine.
- `diskusage`—shows disk space on the Tokenization Server. It is the equivalent to the UNIX `df` command.
- `daemon`—shows monitoring status of daemon processes running on the Tokenization Server.
- `osversion`—shows the system kernel version. It is the equivalent to the UNIX `uname -a` command.
- `uptime`—shows how long the Tokenization Server has been running since the last reboot, the current number of administrators logged into the Tokenization Server, and CPU load usage. It is the equivalent to the UNIX `uptime` command.

Configure the VTS

The `vts` command in the `maintenance` category opens a sub-menu with additional commands to:

- Limit the number of tokenization or detokenization operations in a given time period
- Limit the number of records to process at once when using the batch data transformation utility
- Enable or disable cross-origin resource sharing.
- Set the number of retries and lockout period for logging in to the VTS GUI
- Upgrade the Tokenization Server software

Syntax

```
vts
```

At this prompt, use one of the following syntaxes:

```
{maxrequestspersecond | maxrequestsperhour | maxrequestsperday}
{show | delete | set <limit>}

batchsizelimit {show | delete | set <limit>}

cliadmin {passwd [<password>]}

cors {enable | disable | show}

weblogin [lockout|maxtries [show|set {num}]] | quit

upgrade URL
```

- `maxrequestspersecond`, `maxrequestsperhour`, or `maxrequestsperday` — manages rate limiting of tokenization and detokenization operations. The `show` command displays the current rate limit. The `delete` command removes the rate limit. The `set <rate>` command tells how many tokenization or detokenization requests are permitted in the given time period (every second, every hour, or every day). VTS enforces this limit with an accuracy level within 5%. If the limit is exceeded, the HTTP error code 503 is returned.
- `batchsizelimit`—the maximum permitted number of records to be submitted in one API call. Often used along with rate limiting, but can also be used alone.
- `cliadmin`—changes the cliadmin password. If no password is given in the command line, the user is prompted for a new password and its confirmation.
- `cors`—depending on security requirements, enable or disable cross-origin resource sharing. When enabled, code in one domain can make requests to code in a different domain. This enables JavaScript code in a browser to call the VTS directly to tokenize or detokenize data.
- `weblogin`—manages login behavior of the VTS GUI. The `maxtries set num` command sets the maximum number of VTS GUI login attempts that are allowed. The `lockout set num` command sets the length of time (in seconds) to lock out the user after the maximum number of login attempts are used, preventing further login attempts. The `show` option can be used to display the current setting for `maxtries` or `lockout`.

  If you change the lockout period or maximum retries, you will be prompted to restart the Tokenization Server. Make sure there are no tokenization operations running before restarting.

Caution: Do not set `maxtries` to 0. The user will be locked out. No login attempts will be allowed.

- `upgrade`—downloads new software and upgrades the current VTS node.

NOTE: For more information, see “Upgrade” on page 24.

- `quit`—exits the `vts` submenu.

Examples:

Set Rate Limiting

At most 100 tokenization or detokenization requests are permitted per hour:

```
maintenance$ vts
vts> maxrequestsperhour set 100
```

Set Batch Size

This example shows a size limit of 20:

```
maintenance$ vts
vts> batchsizelimit set 20
```

Upgrade the VTS

```
maintenance$ vts
vts> upgrade https://example.com/vts-upgrade-<version>.zip
```

Set Max Login Tries

Set the maximum number of login tries before being locked out:

```
maintenance$ vts
vts> weblogin maxtries set 5
Login maxtries has been set. VTS app needs to be restarted for changes to
take effect.
```

Show Max Login Setting

Show the maximum number of login tries before being locked out:

```
maintenance$ vts
vts> weblogin maxtries show
5 attempts
```

Show Lockout Time Setting

Show how long the VTS GUI login is locked out after administrator exceeds maximum number of login attempts:

```
maintenance$ vts
vts> weblogin lockout show
2 minutes
```

Reset the period in seconds for VTS GUI lockout. To set it to 100 minutes:

```
maintenance$ vts
vts> weblogin lockout set 6000
```

Security Configuration

The `security` category is used to set security parameters.

Enter the `security` category by typing:

```
vormetric$ security
security$
```

The `security` category consists of the following commands:

Table 37: Security Options

Command	Description
`client-certificate`	Enables client authentication by requiring clients to pass a client key and CA certificate to VTS.
`server-cert`	Creates the self-signed certificate or certificate signing request and import signed certificate.
`ssl-protocol`	Specifies what version of the SSL (TLS) is supported by the VTS.
`show`	Show the security settings.

Enable Client Authentication

Enables client authentication by requiring clients to pass a client key and client certificate to VTS. Optionally, the common name (CN) in the client certificate can be used to establish a user identity.

- `upload-ca-cert:` Enterprises that want to use client certificates are encouraged to set up and maintain a CA (certificate authority) of their own. Once a CA has been set up, a CA certificate must be exported from it and uploaded into the VTS in PEM format (ASCII format).
- `remove-ca-cert:` This function is used rarely, if ever. A previously uploaded CA certificate is deleted from the VTS.
- `show-ca-cert:` The CA certificate can be viewed. Certificates are not secret. This function can be used to verify which CA certificate is installed on this VTS.
- `enable-with-id:` This functionality is new in VTS 2.2. In addition to validating a client certificate against the (previously installed) CA certificate, this mode of operation extracts a user name from the CN (common name) field of the client certificate, thereby obviating the need for submitting a password in the REST API. The client certificate acts as an “access badge” or passport. Once a user name has been extracted from the client certificate, it is still

looked up in the local database and/or AD/LDAP, and is subject to a check for user role (Superuser/staff/active) and permissions checks.

- `enable-without-id:` This client certificate mode is compatible with previous releases of VTS. The client certificate is checked against the (previously installed) CA certificate, but Basic Auth (i.e. a combination of username and password) is used to establish the user's identity.
- `disable:` With client certificate support disabled in VTS, VTS will not even request a client certificate from a client computer and simply relies on Basic Auth to establish a user's identity.
- `status:` The status of the client certificate-related security settings is displayed.

Example:

```
security$ client-certificate

Client Certificate

usage: client-certificate [upload-ca-cert|remove-ca-cert|show-ca-
cert|enable-with-id|enable-without-id|disable|status]

options:

    upload-ca-cert      Upload CA certificate

    remove-ca-cert      Remove CA certificate

    show-ca-cert        Show CA certificate information

    enable-with-id      Enable verify client with identities

    enable-without-id   Enable verify client without identities

    disable             Disable verify client

    status              Show status
```

Generate TSL Credentials

The system category server-cert command creates the TLS credentials used to authenticate VTS and their agents.

Table 38: Server-cert Parameters

Parameter	Descriptions
`generatecsr`	Generates the Certificate Signing Request for the VTS.

Table 38: Server-cert Parameters

Parameter	Descriptions
`importcert`	Import a trusted CA-signed certificate.
`createcertss`	Regenerate a self-signed certificate.

Generate Certificate

The `server-cert generatecsr` command generates the certificate signing request on the VTS. Run this utility when the VTS needs to be set up as a trusted server and verified by a CA. After generating a request, use the certificate to obtain a trusted CA-signed certificate.

The information that you provide is displayed when the signer-certificate is viewed. You are prompted to specify:

- Your organizational unit, which is frequently a department or group name
- Organization name, which is frequently the company name
- City or locality in which the organization is located
- State or province in which the organization is located
- The country in which the organization is located

After you enter this data, the utility creates a certificate signing request which is displaced on the console.

Syntax

```
server-cert generatecsr
```

Example:

```
security$ server-cert generatecsr
Continue to generate certificate signing request? (yes|no)[no]:yes
Generating a 2048 bit RSA private key
.........+++
writing new private key to 'server.key'
-----
You are about to be asked to enter information that will be
incorporated into your certificate request. What you are about to
enter is what is called a Distinguished Name or a DN. There are
quite a few fields but you can leave some blank. For some fields
there will be a default value, If you enter '.', the field will be
left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:San Jose
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Eng
Organizational Unit Name (eg, section) []:Eng 2
Common Name (e.g. server FQDN or YOUR name) []:Greg
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:     Do not set the password
An optional company name []:
Successfully generated certificate signing request.
Copy and past the contents displayed into the online enrollment form
when requested.
-----BEGIN CERTIFICATE REQUEST-----
MIICtTCCAZ0CAQAwWjELMAkGA1UEBhMCVVMxCzAJBgNVBAgMAkNBMREwDwYDVQQH
... ==
-----END CERTIFICATE REQUEST-----
```

Import Certificate

The VTS CLI `server-cert importcert` command imports a CA-signed certificate for the current VTS.

The utility does the following:

- Accepts a certificate pasted into the console.
- Verifies that is matches the previously generated private key.
- uploads the certificate to be used by the TLS for web access
- You are prompted to specify:
- Your organizational unit, which is frequently a department or group name
- Organization name, which is frequently the company name
- City or locality in which the organization is located
- State or province in which the organization is located

- The country in which the organization is located

After you enter this data, the utility imports the certificate, completes the installation process, and then restarts the web server component of the VTS. After which, you are returned to the CLI prompt.

Syntax

```
server-cert importcert
```

Example

```
security$ server-cert importcert
Continue to import third-party certificate? (yes|no)[no]:yes
Please copy/paste your certificate into the terminal (Ctrl-d to end
input):
-----BEGIN CERTIFICATE-----
MIIEczCCA1ugAwIBAgIBADANBgkqhkiG9w0BAQQFAD..AkGA1UEBhMCR0Ix...
..HoX
-----END CERTIFICATE-----
```

Create a Self-Signed Certificate

This utility creates a self-signed certificate.

The utility does the following, in the following order:

- Generates the certificate signing request and the key
- Signs the certificate signing request and creates a self-signed certificate.
- Upload the certificate to be used by TLS.
- Restarts the web server.

Syntax

```
server-cert createcertss
```

Example

```
security$ server-cert createcertss
Continue to create self-signed certificate? (yes|no)[no]:yes
Generating RSA private key, 2048 bit long modulus
...................................+++
e is 65537 (0x10001)
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:San Jose
Organization Name (eg, company) [Internet Widgits Pty
Ltd]:Vormetric
Organizational Unit Name (eg, section) []:Eng
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:            Do not set the password.
An optional company name []:
Signature ok
subject=/C=US/ST=CA/L=San Jose/O=Vormetric/OU=Eng
Getting Private key
Successfully created self-signed certificate for Tokenization
Server.
SSL Certificate on VTS updated successfully.
Do you want to restart web server (yes|no)[no]:yes
Stopping nginx:                                            [  OK  ]
Starting nginx:                                            [  OK  ]
```

Specify TLS Protocol

Specify the SSL (TLS) protocol version supported by VTS.

Syntax[1]

```
ssl-protocol [ tls1.2 | tls1.1 |tls1.0 ]
```

1. `tlsv1.0` supports tlsv1.0, tlsv1.1, and tlsv1.2. `tlsv1.1` supports tlsv1.1, and tlsv1.2. `tlsv1.2` supports tlsv1.2

Example

```
ssl-protocol tlsv1.2
```

Show Security Settings

Show the `security` settings.

Syntax

```
show
```

Example

```
security$ show
SLL Protocols: TLSv1.1 TLSv1.2
ssl_client-cert is disabled!
ssl_verify_client is disabled!
Show Security Configuration SUCCESS
```