THALES

Vormetric Data Security Platform

Vormetric Tokenization Server

Installation, Administration, and Programming Guide

Version 2.3.0

APRIL 2, 2019

DOCUMENT VERSION 1

Vormetric Data Security Platform
Vormetric Tokenization Server Installation, Administration, and Programming Guide
Version 2.3.0
April 2, 2019, Document Version 1

Contents

PREFACE

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 implementing VTS into their applications.

The VTS API service categories are:

- Tokenization
- Encryption (Crypto)
- 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.

Table 1: Documentation Changes

Product/Document Version	Date	Changes
2.3.0	04/02/19	Added a clarification on key input sizes and their relationship with encryption modes Added information about importing and exporting keys Added new chapter: Admin REST API Added GCM code samples for encrypt, decrypt, and version key creation Added information about updated performance monitoring Updated CLI Reference with summaries of new CLI hierarchy of command categories Updated Installation and Configuration chapters
2.2.2 patch	11/13/18	•Added new topic: Create Super User •Minor adits to About Logging •Updates to describe how credentials are processed for Azure
2.2.2	10/12/18	2.2.2 documentation features and fixes, and new documentation framework and format.
2.2.0 GA	05/22/18	Update to GA version and format and style clean up.

Table 1: Documentation Changes

Product/Document Version	Date	Changes
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
2.1.1 v1	6/30/2017	Added support for VTS instances in AWS and Azure.

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

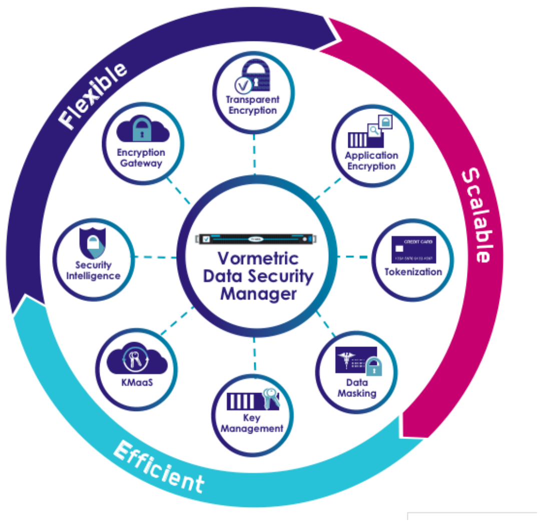
The following related documents are available to registered users on the Vormetric Web site at https://support.vormetric.com

1. **Data Security Manager (DSM) 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 Data Security Manager (DSM), which coordinates policies, keys, and the collection of security intelligence, all of which is managed and observed through your browser. In addition to the DSM, the Vormetric Data Security Platform consists of the following products:

- **Vormetric Application Encryption (VAE)** provides a framework to deliver application-layer encryption such as column- or field-level encryption in databases, Big Data, or PaaS applications. VAE is an application encryption library providing a standards-based API to do cryptographic and encryption key management operations into existing corporate applications.
- **Vormetric Cloud Encryption Gateway (VCEG)** safeguards files in cloud-storage environments, including Amazon S3 and Box. VCEG encrypts sensitive data before it is saved to the cloud, enabling security teams to establish visibility and control around cloud assets.
- **Vormetric Key Management (VKM)** centralizes 3rd-party encryption keys and stores certificates securely. It provides standards-based enterprise encryption key management for

Transparent Database Encryption (TDE), KMIP-compliant devices, and offers vaulting and inventory of certificates.

- **Vormetric Protection for Teradata Database** provides granular controls to secure assets in Teradata environments. It simplifies the process of using column-level encryption in your Teradata database. It provides documented, standards-based APIs and user-defined functions (UDFs) for cryptographic and key management operations.
- **Vormetric Security Intelligence** provides comprehensive logging combined with Security Information Event Management (SIEM) systems to detect advanced persistent threats and insider threats. In addition, the logs satisfy compliance and regulatory audits.
- **Vormetric Tokenization with Dynamic Data Masking (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://enterprise-encryption.vormetric.com/contact-sales.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:

- Vormetric Data Security Manager (DSM)
- AD/LDAP servers (optionally)
- Remote logging servers (optionally)
- vormetric>Load balancers (optionally)

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.

- **Key management services:** Using the Vormetric Data Security Manager (DSM) 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
- DSM (Data Security Manager) is a secure key-store for VTS and other Thales eSecurity products. For more information see the *Vormetric Data Security Manager (DSM) Installation and Configuration Guide*.
- LDAP/Active Directory Server (Optional)
- Remote Logging Server (Optional)
- Load Balancer (Optional)

Vormetric Tokenization Server

The VTS server is a virtual appliance that manages the tokenization tasks for the DSM.

The Vormetric Data Security Manager (DSM) must be installed and configured before deploying VTS.

The virtual appliance handles the following tokenization tasks:

- 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 DSM
- 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 overall administrator to install, configure, and maintain the VTS server(s) and cluster.
- **VTS GUI** used by a GUI administrator to manage users, groups, permissions, data masks, and DSM-based keys.
- **REST APIs** that cover tokenization, cryptography, and key management.

The server can be installed as an ISO or OVA image, shared as an Amazon Web Services Machine Image (AMI), or instantiated from the Microsoft Azure Marketplace.

Vormetric Data Security Manager (DSM)

IMPORTANT: The DSM 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)

An 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.

Load balancer (optional)

To achieve High-Availability and scalable performance, VTS can be set up with multiple servers (nodes) to form a cluster. The cluster can operate as one coherent VTS system to be served by a load balancer.

Tokenization benefits

Vormetric tokenization provides the following benefits:

- 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 and Authorization

In VTS, 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 and permissions" on page 10.

User profiles

Users can be human or servers.

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

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.

1. If authentication is done from a client certificate, the password is not used by VTS. See "Client certificates and user authentication" on page 6 for more detail.
1. See "VTS roles" on page 7 for details.

AD/LDAP and 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 and permissions" on page 10.

NOTE: 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.

Client certificates and 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 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 50.)
- 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.)

Associating key(s) with the cert/user

Assigning 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

When a user is created in 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.)

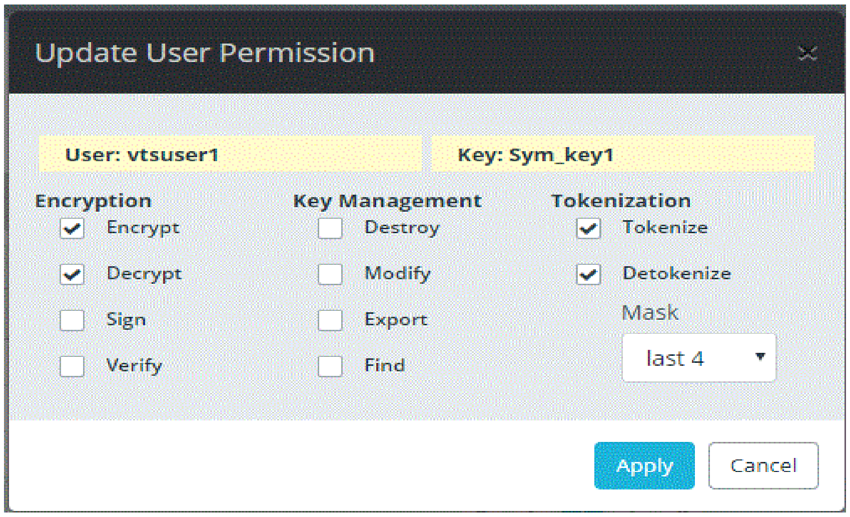
NOTE: When users are created directly in the VTS GUI, they are automatically assigned the status "active," which can be disabled as needed.

If users are imported from AD/LDAP, the "active" designator is handled differently. See "Understanding VTS user groups and permissions" on page 10 for details.

As of VTS 2.3.0, if Active is unchecked, the user cannot log in to the GUI even if assigned Superuser/Staff status.

Key-aware permissions matrix

As of VTS 2.3.0, the GUI Administrator can 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`.


Update User Permission

User: vtsuser1
Key: Sym_key1

Encryption
- [✓] Encrypt
- [✓] Decrypt
- [ ] Sign
- [ ] Verify

Key Management
- [ ] Destroy
- [ ] Modify
- [ ] Export
- [ ] Find

Tokenization
- [✓] Tokenize
- [✓] Detokenize

Mask: last 4

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 and 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.

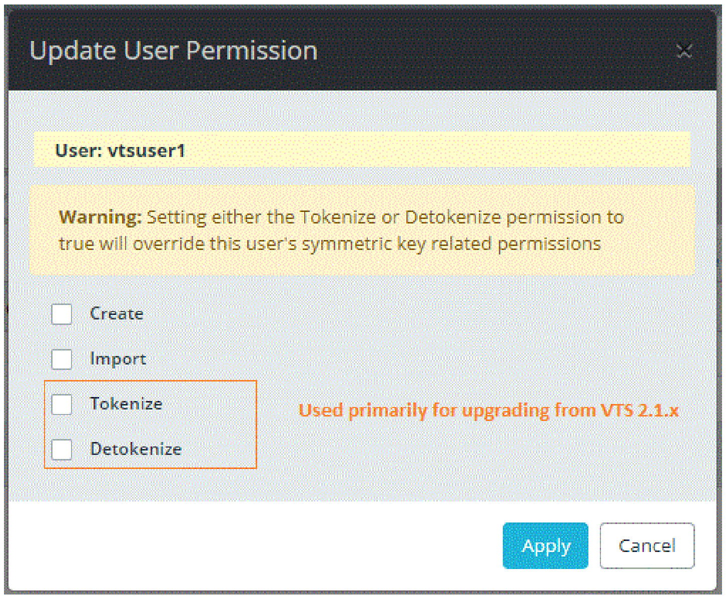
Tokenize and detokenize functionality require a couple of additional steps. See also “Tokenization Groups, Templates, and Masks” on page 13.

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 DSM, under the “Other Permissions” column of the Permissions main page.

NOTE: See also “Creating static and dynamic keys” on page 12.

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



Warning: Setting either the Tokenize or Detokenize permission to true will override this user's symmetric key related permissions

- [ ] Create
- [ ] Import
- [ ] Tokenize
- [ ] Detokenize

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 77 for usage tips.

Understanding VTS user groups and 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" 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, and 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, and Masks" 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.

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 DSM.

Key types, algorithms, and key states

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

Table 1: Key Summary Table

	Description	Algorithms	State
Symmetric	Oct or octet sequence. Used to: • 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 A128GCM - AES 128 GCM cipher mode A256GCM - AES 256 GCM 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
	RSA key type. Used for: • encrypt • decrypt • sign • verify		
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

Key input sizes

For CBC and ECB modes of encryption, the input size should be a multiple of the block size. Note that the AES block size is 16 bytes (128 bits).

If you do not use an input length that is a multiple of the block size, use CTR or CBC-PAD encryption.

Creating static and dynamic keys

Create keys with either a static aor dynamic work flow:

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 DSM.

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.
- **DSM Administrator:** Create VTS-specific keys in the DSM, 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.

IMPORTANT: 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 DSM, using the appropriate *Vormetric Key Management* APIs.

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, and Masks

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 "Managing tokenization groups" on page 66.

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 "Managing tokenization templates" on page 71.

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 “Managing data masks” on page 74.

Figure 5: New data mask creation.

About the VTS API

VTS includes the following sets of REST API services: Encryption, Key Management, Tokenization, and Administration.

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 and Key Management API Programming Notes” on page 91, 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 115. The Administration APIs are described in “Admin REST API (Beta)” on page 137.

These APIs are also described in an online reference guide, under **Documentation** in the VTS Administration GUI navigation bar.

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 the following required components and optional components:

- Vormetric Tokenization Server (VTS)
- Vormetric Data Security Manager (DSM),
- Active Directory (AD)/LDAP server (optional)
- Remote Logging server (optional)
- Load Balancer (optional)

NOTE: This chapter focuses on installing the VTS, with additional sections about upgrade, backup, and recovery. Chapter 3, "Configuring the VTS System" on page 33, describes the follow-up configuration steps the network administrator performs to connect all the components and establish baseline settings.

VTS basic component architecture

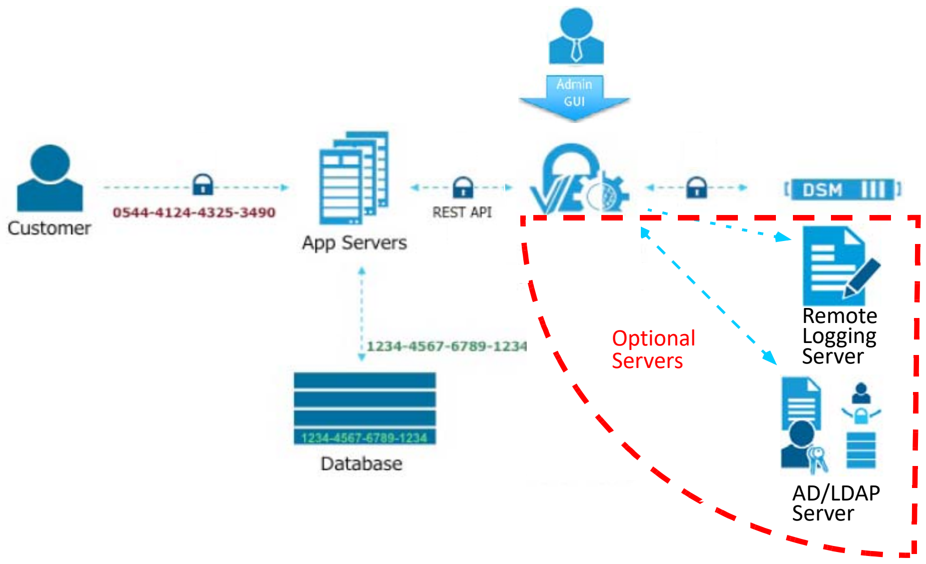
VTS can be installed as a virtual machine (the more typical case) or as software on bare metal.

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 DSM 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: A minimum of two nodes is recommnded. This configuration ensures that the data is retained online, and provides for optimum availability. For higher performance, use a cluster with more than two nodes. It is also recommended to use a load balancer to automatically distribute load to all nodes in the cluster.

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 “Choose an installation type and deploy” on page 24.

- “DSM requirements” on page 19
- “Vormetric Tokenization Server requirements” on page 19
- (Optional) “Active Directory/LDAP server requirements” on page 21
- (Optional) “Remote logging server” on page 22
- “Port configuration” on page 23

DSM requirements

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

For specific DSM version compatibility requirements, check the *Release Notes* for Vormetric Tokenization Server. Note that VTS does not require a dedicated DSM; it can be shared with other VDS applications. To install a new DSM, see the *Vormetric Data Security Platform Data Security Manager (DSM) Installation and Configuration Guide*.

Gather this DSM information for system configuration:

1. DSM fully qualified hostname: ____
2. DSM IP address: ____

Vormetric Tokenization Server requirements

Replace entire section:

Vormetric Tokenization Server can be installed on a physical server (bare metal) or as a virtual machine.

To run as a virtual machine, VTS supports the following Hypervisors:

- VMware ESXi 6.0 or higher
- Amazon AWS EC2
- Microsoft Azure
- Microsoft Hyper-V

- Google Cloud Compute Engine

VTS builds are available in the following formats:

- ISO -- for physical machines, VMware ESXi, or Hyper V
- OVA -- for VMware ESXi
- VHD -- for Hyper V and Azure
- AMI -- for Amazon AWS
- GCP -- for Google Cloud Compute Engine

Minimum system requirements:

RAM:

- 16GB minimum for RANDOM mode = none
- 24GB for RANDOM mode = numeric or alphanumeric
- 32 GB for for RANDOM mode = both numeric and alphanumeric

CPU:

- 4 CPUs, x86, > 2.0Ghz recommended

Disk:

If VTS is installed as a virtual machine, the disk size is automatically created and allocated when the virtual machine is created from the build image.

If VTS is installed on a physical machine, a minimum of 85GB disk is required. ISO installation will fail if the disk is less than 85GB. Installation will also fail if RAM is less than 16GB.

Networking:

VTS supports DHCP or static IP address. IPv6 is not supported, only IPv4 is supported at this time.

VTS supports DNS or static entries in */etc/hosts* file. It is recommend to use DNS if available.

Gather this Vormetric Tokenization Server information for system configuration:

1. Tokenization Server `.ova` image, `.iso` image, Amazon AWS ID, or Azure ID:
   ____
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 "The /etc/hosts file" on page 37.

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 and Authorization" on page 4 and "AD/LDAP and user authentication" on page 6 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 45.

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	
user group search scope	Search criteria of user group inside AD/LDAP **Example**: cn=Users,dc=corp,dc=example, dc=com	

Table 2: AD/LDAP Configuration Information

Name	Description	Your Site Information
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):____

Load balancer requirements

A cluster of VTS nodes can be installed and run as one coherent VTS system using a load balancer. The load balancer will act as the main entry point for tokenization and crypto services to distribute load across all nodes in the cluster. This will allow scalability and reliability of the system.

VTS supports the following methods of load balancing:

- Round-robin - requests are distributed in a round-robin fashion.
- Least-connected -- allows the load balancer to control the load on some VTS nodes.
- IP-hash -- the client's IP address is used to determine what server in a server group should be selected for the client's request. This can be used for client's session sticky or persistent.

The load balancer must be able to use TLS1.1 and higher.

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 4 shows outgoing traffic.

NOTE: For a complete list of ports required for the DSM, see the *DSM 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	This port is required if a cluster of more than one node is used. This port is used exclusively for communication between the VTS nodes in a cluster. If the cluster of nodes is inside a VPC or subnet, access to port 5432 should only be confined within the subnet or VPC. It is recommended that this 5432 port is firewalled off from anything other than the cluster 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 → DSM	Default TCP/IP port for HTTP that is used once to perform the initial certificate exchange between a VTS host and DSM.
8443	TCP	VTS → DSM	TCP/IP port through which the VTS communicates with the DSM to exchange configuration. The VTS establishes a secure connection to the DSM via certificate exchange using this port.
8444	TCP	VTS → DSM	Upload VTS log messages to DSM
8446	TCP	VTS → DSM	Configuration exchange (policy/key pull) using Elliptic Curve Cryptography (Suite B)
8447	TCP	VTS → DSM	VTS uploads log messages to DSM using Elliptic Curve Cryptography

Choose an installation type and deploy

There are several options for the network administrator to install the Vormetric Tokenization Server:

- “Install VTS as a virtual machine on VMware vSphere” on page 24
- “Install VTS on bare metal” on page 25
- “Install VTS in the AWS cloud” on page 25
- “Install VTS in the Azure cloud” on page 27
- Install VTS on Hyper-V
- Install VTS in the Google Cloud (GCM)

Install VTS as a virtual machine on VMware vSphere

VTS can use a VMware ESXi hypervisor to create a virtual machine using the ISO or OVA build image. VTS supports ESXi version 6.0 or higher. Refer to VMware ESXi documentation on how to create a VM using an ISO or OVA image.

It is recommended to use Thick Provisioning for hard disks.

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 "Configuring VTS Nodes" on page 41.

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 "Configuring the VTS System" on page 33).

Install VTS on bare metal

Although typically installed as a virtual machine, the Vormetric Tokenization Server is also provided as an ISO image that can be installed on a bare metal host.

A bare metal host is a computer that does not run hypervisor software. Do not install the operating system on the host before installing VTS software, because the OS is included along with the VTS software in the installation package. When selecting the operating system, specify 64-bit CENTOS version 7 or higher.

Make sure the computer meets the following minimum system requirements:

- Number of CPU cores: 4
- RAM: 15 GB minimum for RANDOM mode none. 24 GB recommended if VTS will be used to tokenize credit card numbers longer than 16 digits. 24 GB if using numerical RANDOM mode or alphanumeric RANDOM mode, or 32 GB if using both modes.
- Hard disk space: 85 GB

To do a bare metal installation:

1. Obtain the installation file `vormetric-tokenization.iso.`
2. Refer to the machine's documentation on how to boot and install from an ISO image.

Install VTS in the AWS cloud

Prerequisites

The following components are necessary for running a VTS in the cloud on Amazon Web Services.

1. A VPC (Amazon's *Virtual Private Cloud*). If you plan to create a cluster of VTS machines, it is recommended that your cluster nodes use a VPC so that they communicate only with each

other within the private cloud. Cluster communication does not need to be, and should not be, exposed on the Internet.

2. A Subnet. If you choose to use a VPC, you will need to create a subnet and associate that subnet to that VPC. This subnet is where your VTS cluster nodes will get the IP addresses.
3. A Security Group. Set your security group with the following minimum settings:

   a: Incoming port 443 - Web and REST API access

   b: Incoming port 22 - CLI access

   c: Incoming port 5432 - Optional. Set this only if the machines in your cluster are not using a VPC to communicate with each other.

   If you will be accessing your VTS strictly inside the VPC, you will still need a security group, but you can block all incoming ports. If your DSM is located inside the same VPC, you can block both incoming and outgoing ports.

Create the VTS instance

Follow this procedure to create the VTS.

1. Choose your AWS region
2. Locate the AMI of the VTS image
3. Select the AMI image and launch an instance.
4. Select Instance type: `m4.xlarge` - with recommended minimums of 4 vCPUs and 16GB memory
5. Select the VPC to use for the installation.
6. Select the subnet.
7. Select **Auto-assign public IP**, if applicable.
8. Set the primary IP address. This will be the IP address of the VPC subnet, and it will be used by the VM for its cluster node IP address.
9. In the **Add Storage** menu, select `delete on termination`, if applicable. Do not change the size of the root volume.
10. No not select a key pair. VTS currently does not support the use of key pairs.

Configuration notes

To begin configuration, `ssh` into the instance as `cliadmin` user, using the public IP address or the VPC IP address, and follow the steps in "Configuring the VTS System" on page 33.

If you will be creating a cluster of VTS machines, create an instance for each node in the cluster. You would configure each node separately and join them in a cluster as described in "Configuring VTS Nodes" on page 41. Note that when configuring a multi-node cluster, you will use the **VPC IP** address for the `node_IP_address`.

Install VTS in the Azure cloud

Create the Azure VM

1. Log in to the Azure web portal at `https://portal.azure.com`.
2. Go to the Marketplace screen and search for **Vormetric Tokenization Server.** Select the desired version and click the **Create** button.
3. Supply the following information to create a virtual machine:
   - **VM name:** name of the VM you want to use, for example, `my-vts-2.3.0-azure-vm`
   - **VM disk type:** SSD or HDD.
   - **User name:** Not required by VTS but Azure needs this data to create a VM. This user name entry is not retained by VTS, and is not used as a VTS credential.
   - **Authentication type:** Also not required by VTS but required for Azure to create a VM. This authentication type entry is not retained by VTS, and does affect the VTS configuration.
   - **Resource group:** You may create a new resource group or use an existing one. If you plan to create a cluster of VTS machines, it is recommended that all cluster nodes use the same resource group, so that they can communicate with one another without having to use a public static IP address or enable an incoming port in the firewall.
   - **Location:** For example, “West-US”
   - **Subnet and subnet address range**
   - **Static or dynamic public IP address**: Use a static IP address if you expect your cluster to span multiple resource groups

Configuration notes

To begin configuration, `ssh` into the instance as the `cliadmin` user using the public IP address, and follow the steps in “Configuring the VTS System” on page 33.

Note that on Azure, you will not “Set the VTS network settings” on page 35. Azure automatically configures these parameters for the VTS VM:

- Internal and subnet IP addresses are provided to the VM through DHCP.
- For VTS clusters, create an instance for each node in the cluster. Then configure each node separately and join the nodes in a cluster.
- When creating or joining a cluster, if you have all the cluster nodes in the resource group, you must use the same **internal or subnet IP** address used by the VM.
- When adding a node to the cluster, you must always use the internal or subnet IP address of the new VTS VM node.

The network security group defines the firewall rules of the VTS virtual machine. At a minimum, incoming ports must allow port 22 (SSH) and port 443 (HTTPS).

If your cluster nodes are located in different resource groups, you also need to add port 5432 to the allowed incoming ports. However, if all your cluster nodes are in the same resource group, do not add port 5432, because nodes inside a resource group do not need any firewall permissions to communicate with one another.

Install VTS on Hyper-V

Install the VTS on a Hyper-V server using an .iso or VHD file.

Download the .iso or VHD file to the installation server:

Use the following steps to install the VTS on a Hyper-V server using an .iso or VHD image file.

1. Open Hyper-V Manager.
2. Select **Action** > **New** > **Virtual Machine**.
3. Choose **Generation 1**.

NOTE: Generation 2 is not supported.

4. Set the start up memory to 16GB.
5. Choose a network adapter.
6. For the VM, choose one of the following based on the type of file used:
    - `.iso` image - Use an existing hard disk: Select VTS VHD image file you've downloaded or copied.
    - VHD file - Install an operating system from a bootable CD/DVD-ROM: Navigate to and select the downloaded `.iso` file.
7. Select **Finish** to create the Virtual Machine
8. For `.iso` only: Select and start the Virtual Machine. This VM boots up the mounted ISO image file and the VTS installation begin

Deploy VTS in a Docker image (beta)

IMPORTANT: Docker container support is currently a beta-level implementation.

If you are using Docker Containers, you can deploy a VTS image within a Docker container. To deploy VTS:

1. Import the VTS Docker image:

```
# cat vts-<version.build>.docker.tar.gz | docker import -
vts:<version.build>
```

Example:

```
# cat vts-2.3.0.99.docker.tar.gz | docker import - vts:2.3.0.99
```

2. Update the image:

```
# docker build -f Dockerfile -t vts:<latest_version> .
```

Example:

```
# docker build -f Dockerfile -t vts:2.3.0.99 .
```

3. Create and run the VTS container:

```
# docker run --name <container_name> --hostname <container_name> -d -p
822:22 -p 443:443 -it vts:<latest_version>
```

Example:

```
# docker run --name myvts1 --hostname myvts1 -d -p 822:22 -p 443:443 -it
vts:2.3.0.125
```

4. Log in to the VTS container as cliadmin:

```
# ssh -p 822 cliadmin@docker-<host-ip-address>
```

Example:

```
# ssh -p 822 cliadmin@docker-192.168.1.45
```

5. Browse the VTS container:

```
# https://docker-<host-ip-address>
```

Example:

```
# https://docker-192.168.1.45
```

6. Stop the container, type:

```
# docker stop <container_name>
```

Example:

```
# docker stop myvts1
```

7. Start the container, type:

```
# docker start <container_name>
```

Example:

```
# docker start myvts1
```

8. Remove the container, type:

```
# docker rm <container_name>
```

Example:

```
# docker rm myvts1
```

Upgrading VTS

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.

IMPORTANT: You can upgrade to VTS version 2.3.0 only from version 2.2.2. VTS v2.2.2 introduced the ability to back up and restore VTS configurations. This capability is required to upgrade to v2.3.0.

After running VTS v2.2.2, follow these steps to upgrade to v2.3.0:

1. Back up your VTS configuration. The backup includes users and groups, permissions, and settings. You can perform this backup either through the VTS GUI or REST APIs. Refer to "Back Up and Restore the VTS" on page 81 for instructions on how to back up VTS 2.2.2.
2. Download VTS 2.3.0 binaries from the Thales support portal, and install this version.
3. Restore to v2.3.0 the 2.2.2 backup. Refer to "Back Up and Restore the VTS" on page 81 for instructions on how to restore a backup.

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 in 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>
[...]
Vormetric Tokenization Server Main Menu
vormetric> ?
    Command              Description
    =======              ===========
    auth                 Authentication Setup
    cluster              Cluster Setup
    network              Network Setup
    system               System Setup
    vae                  VAE Configuration
    vts                  VTS Configuration
    quit|q|up|<ctrl-d>   Quit or return to previous menu
    exit                 Exit application

    Enter <command> to display usage
vormetric>
```

4. At the main menu, enter:

```
vormetric> vts upgrade --url <upgrade_zip_url>
```

Another option is to navigate to the vts submenu and enter the upgrade command directly:

```
vormetric> vts
vts> upgrade <upgrade_zip_url>
```

5. 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.
   ```

   Depending on the type of upgrade, the system will be rebooted at the end for changes to take effect. You are prompted at the beginning of the upgrade for confirmation.

   There are other options to perform an upgrade. Run `upgrade --help` for details:

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.

CLI navigation cheat sheet

Navigation	Command	Example
Starting prompt	`vormetric>`	

List CLI categories or the commands within categories	**?**	`vormetric>` **`?`**
		`auth` `Authentication setup`
		`cluster` `Cluster Setup`
		`network` `Network Setup`
		`system` `System Setup`
		`vae` `VAE Configuration`
		`vts` `VTS Configuration`
		`quit|q|up|<ctrl-d>` `Quit or return to previous menu`
		`exit` `Exit application`

Open a category and display its commands	*`<category name>`* **`?`**	`vormetric>` **`cluster`**
		`network>` **`?`**
		`add` `Register a new node to a cluster`
		`create` `Create New Cluster`
		`join` `Join node to a cluster`
		`remove` `Remove node from cluster`
		`show` `Show cluster status`
		`quit|q|up|<ctrl-d>` `Quit or return to previous menu`
		`exit` `Exit application`

Get help: See command usage and example output	`<enter the command without a value>`	`network>` **`checkport`**
		`usage: checkport [-h] [--host HOST] [--port PORT [PORT ...]]`
		`Check port connection status of a host`
		`Optional arguments:`
		`-h, --help` `show this help message and exit`
		`--host HOST` `hostname or IP address of host`
		`--port PORT [PORT ...]`
		`port numbers to scan`

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`**	

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

Basic Configuration Steps

The configuration steps in the following sections employ a subset of the CLI commands. See "VTS CLI Reference" on page 85 for a summary of CLI commands. For complete usage details on any CLI command, run the command with the "`--help`" option.

NOTE: For multi-node installations, the basic configuration steps in this section must be completed on each node before you "Configuring VTS Nodes" as described on page -41.

Set the VTS network settings

Configure the network setting for VMware and bare metal installations with a static IP implementation.

IMPORTANT: Network settings are required only for VMware and bare metal installations, with a static IP implementation. Do not attempt to set the IP address and gateway of the network interface if you are using VTS in an Azure, Amazon, or Google cloud. Cloud providers assign the IP addresses to the VM through DHCP.

Configure the VTS network settings:

- "Set the IP address for the VTS virtual machine."
- "Assign the DNS server," or use "The /etc/hosts file."

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.

```
vormetric> network
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:

```
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:

```
network> ip address show
```

Assign the DNS server

If you are not resolving hostnames with a DNS server, see "The /etc/hosts file" on page 37.

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

```
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:

```
network> dns dns2 <IP address for dns2>
```

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

```
network> dns search <search_domain>
```

4. Show the DNS settings. Type:

```
network> dns --show
DNS resolver is using DHCP
Generated by DHCP:
    nameserver 192.168.100.199
    nameserver 192.0.0.299
```

The /etc/hosts file

If you have hostnames that do not use DNS, you can use the local `/etc/hosts` file to resolve IP addresses. To update this file, run the CLI "`system hosts`" command.

```
vormetric> system hosts
usage: hosts [-h] [--show] [--add IP [HOSTNAME ...]] [--remove IP]

Update hosts file

optional arguments:
  -h, --help            show this help message and exit
  --show                Show hosts file entries
  --add IP [HOSTNAME ...]
                        Add/update hosts entry

  --remove IP           Remove hosts entry
```

- **Modify the `hosts` file on the Vormetric Tokenization Server.** Include other Vormetric Tokenization Servers, the AD/LDAP server, the DSM, 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:

```
network> host add rgrimes 192.168.99.999
   SUCCESS: add host
network> host show
name=localhost6.localdomain6
name=linux32-48215.hilltop_colo.com ip=192.168.99.599
name=rgrimes ip=192.168.99.999
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 DSM, which must be identified by its FQDN.

NOTE: Network settings are not applicable for Azure VMs.

Set the VTS hostname

You must set a hostname for the VTS VM. This hostname must match the hostname created on the DSM. The VTS will not be able to register to the DSM if those names do not match.

To set the hostname of the VTS VM, run the `system hostname` command with the `--set` option.

1. In the VTS CLI, enter:

```
vormetric> system hostname
usage: hostname [-h] [--show] [--set NAME]

Set/show hostname

optional arguments:
  -h, --help  show this help message and exit
  --show      Show System Hostname
  --set NAME  Set System Hostname
```

Import a server certificate for TLS

Use the `vts server_certificate` command in the VTS CLI to import a server certificate. Enter the command without any options to see a detailed help message that explains all arguments to the command (some of which are deleted from the example shownb below to save space):

```
vormetric> vts server_certificate

usage: server_certificate [-h] [--import {certificate,key}]
                          [--create {csr,selfsigned,key}] [--size SIZE]
                          [--show {key,certificate}] [-y] [--country C]
                         [--state ST] [--locality L] [--organization O]
                          [--organizational_unit OU] [--common_name 
HOSTNAME]
                          [--email EMAIL] [--days DAYS]

Server Certificate Utility

optional arguments:
  -h, --help            show this help message and exit
  --import {certificate,key}
                        import a certificate or private key
  --create {csr,selfsigned,key}
                        create a certificate signing request, a self-
signed certificate, or a new private key.
                        a certificate signing request is created using
[ . . .]
```

Option 1: Regenerate a self-signed server certificate

To generate a self-signed certificate run the `vts server_certificate` command with the `--create selfsigned` option

```
vormetric> vts server-certificate --create certificate
```

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

If your server requires a particular TLS protocol, you can set it following the instructions in "Specify the TLS protocol version (optional)" on page 52.

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

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

.Register VTS with the DSM (mandatory)

See "DSM requirements" on page 19 for a list of the prerequisites before configuring.

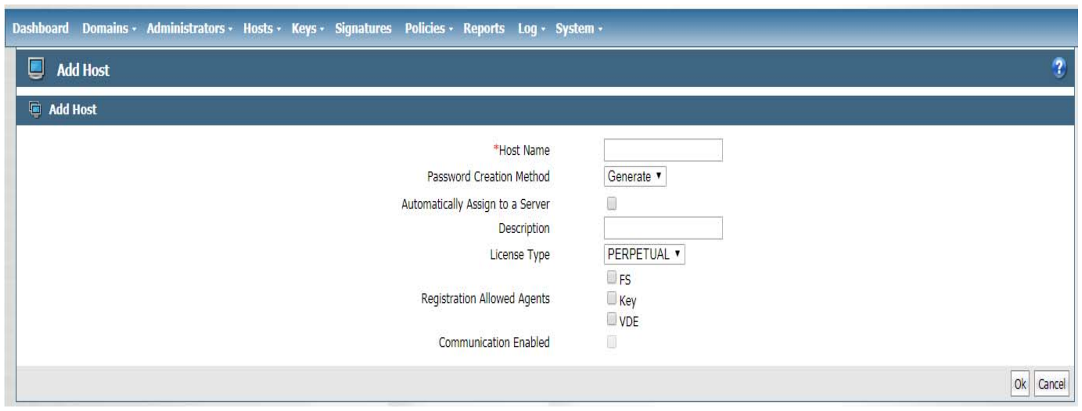
1. If there is no DNS, add the hostname of your DSM into `/etc/hosts`. Go to the `network` submenu in the VTS CLI and enter: `host add <dsm hostname> <dsm IP Address>`

```
network> host add dsm1 192.168.34.12
```

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

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

3. Click **Ok**.



*Host Name	
Password Creation Method	Generate
Automatically Assign to a Server	☐
Description	
License Type	PERPETUAL
Registration Allowed Agents	☐ FS ☐ Key ☐ VDE
Communication Enabled	☐

4. In the DSM 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**.
5. Return to the VTS CLI and register the VTS with the DSM.

   a. Obtain the DSM hostname from the *DSM Management Console> Dashboard> Server Name*.

   b. Then go to the `vae` category of the CLI and run:

```
network> up
vormetric> vae
vae> register <dsm hostname>
```

Configuring VTS Nodes

A VTS image can be deployed once, as a stand-alone instance or node, or as a cluster of nodes, and used for demo or test purposes. In a production environment, it is recommended to repeat the installation procedure on multiple nodes.

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

If your environment already has one VTS VM installed and configured, and the database has already been created, then skip to “Adding nodes to an existing cluster” on page 45.

NOTE: Before you begin make sure you have the IP addresses of each VTS node for the cluster.

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.

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 45.

Before you begin

Have the IP addresses of each VTS node on hand.

NOTE: VTS supports up to ten nodes in a cluster.

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 `create` *`<node IP address>`*
   This launches the process of creating the first VTS database, along with other VTS configuration steps behind the scenes.

```
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 "Configure an AD/LDAP Server (optional)" on page 45, as needed for your environment. Finish with "Restart the Vormetric Tokenization Server" on page 54.

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.

```
cluster$ add_node node2_IP_address
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

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>
```

“Access the VTS GUI” to test the node.

On Instance 3

Add nodes 1 and 2

n 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

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

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

"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 "Configure an AD/LDAP Server (optional)" on page 45, as needed for your environment.

Finish with "Restart the Vormetric Tokenization Server" on page 54.

NOTE: VTS supports up to ten nodes in a cluster.

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`*>.

NOTE: Every node in the cluster must run the add_node command for each new node to be added. If one node is missed, the `join` command hangs and the replication does not complete.

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 21 before configuring. You must configure the AD/LDAP server on every node in your VTS HA cluster.

1. Go to the `auth` command category in the VTS CLI to see the `ldap` command.

```
vormetric> auth
auth$ ?
    Command              Description
    =======              ===========
    ldap                 LDAP Setup
```

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

```
auth> ldap
usage: ldap [-h] [--show] [--setup] [--enable {true,false}]
               [--server SERVER_URI] [--bind_dn BIND_DN]
               [--bind_password PASSWORD] [--user_search_filter FILTER]
               [--user_search_scope SCOPE] [--group_search_scope SCOPE]
               [--active_user_group GROUP] [--staff_user_group GROUP]
               [--super_user_group GROUP]
               [--ca_cert {import,remove,show}] [-y]


LDAP Setup


optional arguments:
-h, --help            show this help message and exit
--show                show LDAP settings
[ . . . ]
```

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

```
auth$ ldap --enable false
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$ ldap --enable true
```

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:

```
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:

```
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:

```
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:

```
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:

```
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:

```
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:

```
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:

```
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*:

```
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*:

```
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: kj4322l
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)

See also: “Client certificates and user authentication” on page 6.

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.

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:

```
0028:vormetric> security
0029:security$ client-certificate upload-ca-cert
Please paste content of certificate and press CTRL-D when finished.

-----BEGIN CERTIFICATE-----

MIIDrTCCAxagAwIBAgIBADANBgkqhkiG9w0BAQQFADCBnDEbMBkGA1UEChMSVGhl
```

```
IFNhbXBsZSBDb21wYW55MRQwEgYDVQQLEwtDQSBEaXZpc2lvbjEcMBoGCSqGSIb3
DQEJARYNY2FAc2FtcGxlLmNvbTETMBEGA1UEBxMKTWV0cm9wb2xpczERMA8GA1UE
CBMITmV3IFlvcmsxCzAJBgNVBAYTAlVTMRQwEgYDVQQDEwtUU0MgUm9vdCBDQTAe
Fw0wMTEyMDgwNDI3MDVaFw0wMjEyMDgwNDI3MDVaMIGcMRswGQYDVQQKExJUaGUg
U2FtcGxlIENvbXBhbnkxFDASBgNVBAsTC0NBIERpdmlzaW9uMRwwGgYJKoZIhvcN
AQkBFg1jYUBzYW1wbGUuY29tMRMwEQYDVQQHEwpNZXRyb3BvbGlzMREwDwYDVQQI
EwhOZXcgWW9yazELMAkGA1UEBhMCVVMxFDASBgNVBAMTC1RTQyBSb290IENBMIGf
MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDaiAwfKB6ZBtnTRTIo6ddomt0S9ec0
NcuvtJogt0s9dXpHowh98FCDjnLtCi8du6LDTZluhlOtTFARPlV/LVnpsbyMCXMs
G2qpdjJop+XIBdvoCz2HpGXjUmym8WLqt+coWwJqUSwiEba74JG93v7TU+Xcvc00
5MWnxmKZzD/R3QIDAQABo4H8MIH5MAwGA1UdEwQFMAMBAf8wHQYDVR0OBBYEFG/v
yytrBtEquMX2dreysix/MlPMMIHJBgNVHSMEgcEwgb6AFG/vyytrBtEquMX2drey
six/MlPMoYGipIGfMIGcMRswGQYDVQQKExJUaGUgU2FtcGxlIENvbXBhbnkxFDAS
BgNVBAsTC0NBIERpdmlzaW9uMRwwGgYJKoZIhvcNAQkBFg1jYUBzYW1wbGUuY29t
MRMwEQYDVQQHEwpNZXRyb3BvbGlzMREwDwYDVQQIEwhOZXcgWW9yazELMAkGA1UE
BhMCVVMxFDASBgNVBAMTC1RTQyBSb290IENBggEAMA0GCSqGSIb3DQEBBAUAA4GB
ABclymJfsPOUazNQO8aIaxwVbXWS+8AFEkMMRx6O68ICAMubQBvs8Buz3ALXhqYe
FS5G13pW2ZnAlSdTkSTKkE5wGZ1RYSfyiEKXb+uOKhDN9LnajDzaMPkNDU2NDXDz
SqHk9ZiE1boQaMzjNLu+KabTLpmL9uXvFA/i+gdenFHv
-----END CERTIFICATE-----
```

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

```
0002: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:

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

To enable with identities, run:

```
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 "Managing VTS Users" on page 59.

To disable client certificate authentication, run:

```
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 SSL/TLS 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`:

```
vormetric:> security
security> ssl-protocol tlsv1.2
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:

```
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:

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

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

```
remotelog$ remote
usage: remote [enable/disable]
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:

```
remotelog> host
usage: host [HOSTNAME/IP] port]
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:

```
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:

```
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.
```

4 VTS Administration in the GUI

This chapter contains the following sections:

Logging into the VTS GUI

Use the VTS administration GUI to

- create tokenization users and groups
- specify permissions for users and groups
- create token templates
- create token groups

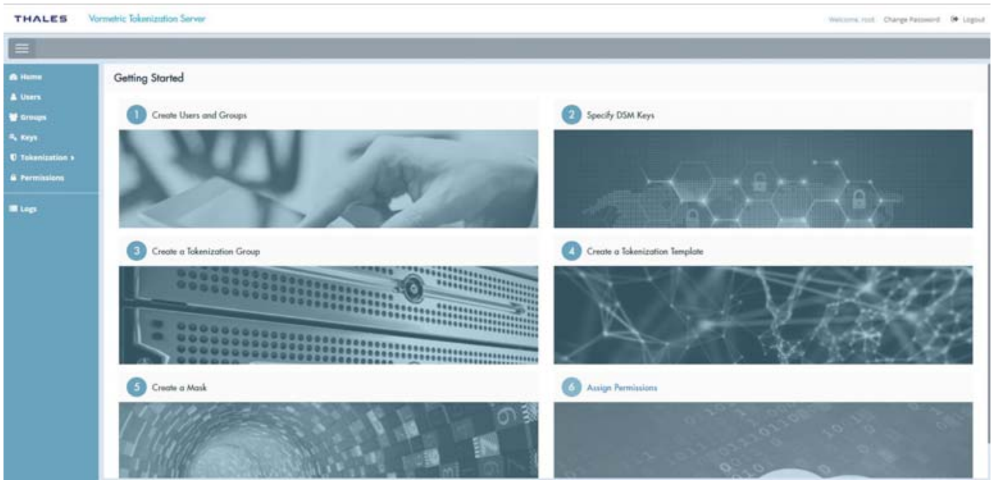
To log into the VTS administration GUI:

1. Point your browser to the Tokenization Server:

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

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 43).

3. The VTS GUI Home page is displayed.



Troubleshooting

A Tokenization Server Administration GUI login failure is indicated by the following error message:

```
nginx 504 Gateway Time-out error
```

In this scenario, consider the following problem/solution option:

- Problem: The Tokenization Server is not connected to the LDAP server.
- Solution: Reboot the Tokenization Server. Open the VTS CLI. Execute the following commands:
  1.**Auth** > **disable**
  2. **system** > **reboot**

Creating and Managing Passwords

The VTS uses a dynamic password policy control mechanism to ensure system security.

The VTS leverages a password checker library. If you try to create a password that does not pass the proactive password checking provided by this library, you are prompted to try another string. The policy can change dynamically based on several factors. The dynamic nature of the checking algorithm provides the most secure possible password policy options. When creating a system password, use the following guidelines:

- Use at least 1 digit, 1 uppercase letter, 1 lowercase letter, 1 special character, and use at least 8 characters in total.
- Do not use your name, the company name, or other familiar names in the password string.
- Do not use your birthday or any other personally or professionally meaningful dates in the password string.
- Include a combination of the following character types in the password string:
  - Lowercase letters
  - Uppercase letters
  - Numbers
  - Special characters

Managing VTS User Groups

User groups enable you to assign the same permissions to multiple users.

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

- "VTS roles" on page 7
- "Understanding VTS user groups and permissions" on page 10

If using AD/LDAP users, create at least three user groups, to correspond to the VTS roles:

- Superuser
- Staff
- Active

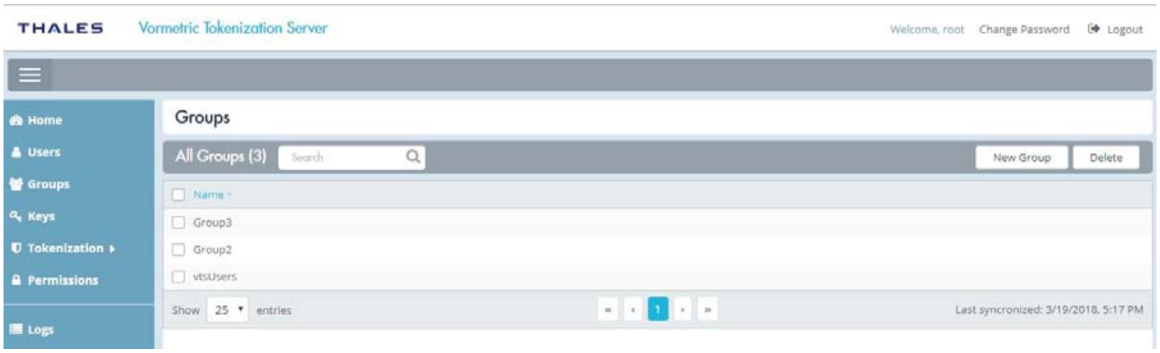
User groups must be defined on the LDAP system. The user group names must be entered into the VTS GUI before importing users.

Creating user groups

To create a user group:

1. Log into the VTS GUI and select **User Groups** from the navigation bar.
   The All User Groups list is displayed.

Figure 7: List of User Groups



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

Assigning masks

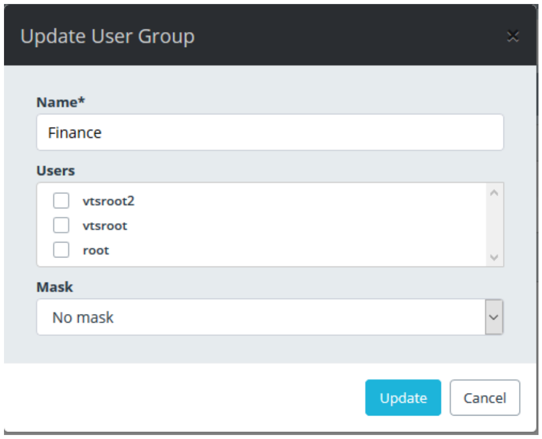
You can assign a mask to a group when initially creating the user group (at the *New User Group* dialog), or any time later to an existing user group, by clicking the user group and adding a mask assignment at the *Update User Group* dialog.

Adding users to user groups

To add a user to an existing user group:

1. From the VTS GUI select **User Groups**.
2. Click on group from the list.
3. Select users to add to the group and click **Update**.

Figure 8: Update User Group Dialog


Update User Group

Name*
Finance

Users
- [ ] vtsroot2
- [ ] vtsroot
- [ ] root

Mask
No mask

Deleting a user group

To delete a user group:

1. Log into the VTS GUI and select **User Groups** from the navigation bar.
2. Select the check box by a group
3. Click **Delete**
   A dialog asks you to confirm the **Delete**.

IMPORTANT: If you delete a user group to which users and/or permissions have been associated, the user group is removed from any user belonging to it, and will delete any permissions assigned to it.

Managing 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 "Creating a user manually" or "Importing users from an AD/LDAP server".

NOTE: Thales recommends creating at least one user in the GUI before attempting the task manually.

Creating a user manually

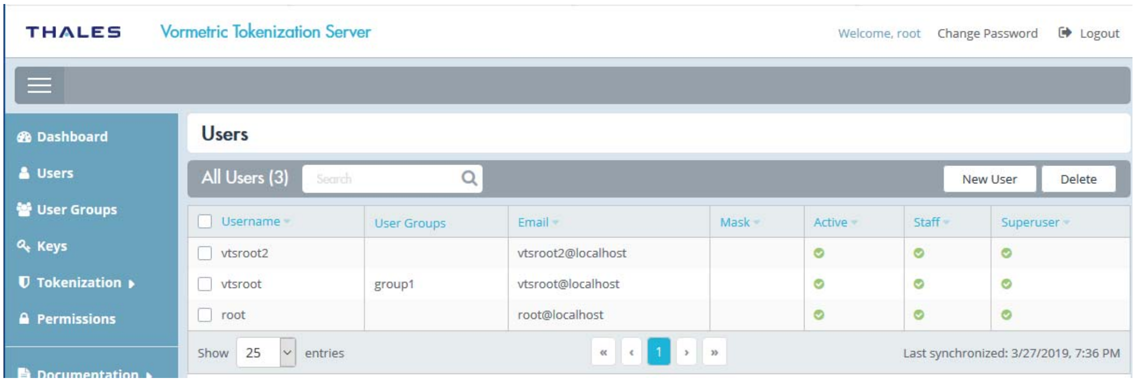
To create a user manually:

1. "Creating basic user information" on page 60
2. "Updating user information" on page 61

Creating basic user information

1. Log into the VTS GUI and select **Users** from the navigation bar.
   The list of All Users is displayed.

Figure 9: All Users List



Username	User Groups	Email	Mask	Active	Staff	Superuser
vtsroot2		vtsroot2@localhost		✓	✓	✓
vtsroot	group1	vtsroot@localhost		✓	✓	✓
root		root@localhost		✓	✓	✓

Show 25 entries

Last synchronized: 3/27/2019, 7:36 PM

2. Select **New User**.
   In the New User dialog, fill in the required information and click **Create**.

IMPORTANT: The User Name must contain only English letters, numbers, and the following special characters: "@" (at-sign), "." (period), "+" (plus-sign), "-" (hyphen), or "_" (underscore).

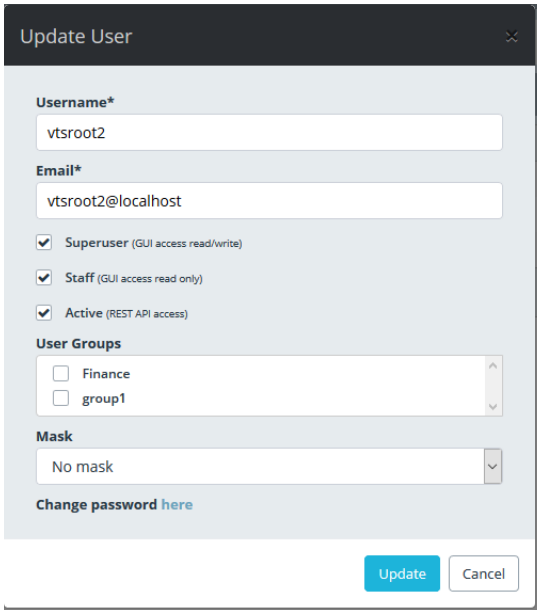
NOTE: User names and passwords are case-sensitive. Application code that calls the VTS RESTful APIs must pass user names and passwords in the same case that are defined in the VTS GUI.

Updating user information

After you "Creating 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 into the VTS GUI, and select **Users** from the navigation bar.

2. Click on the User Name of interest.



Update User

Username*
vtsroot2

Email*
vtsroot2@localhost

- [x] Superuser (GUI access read/write)
- [x] Staff (GUI access read only)
- [x] Active (REST API access)

User Groups
- [ ] Finance
- [ ] group1

Mask
No mask

Change password here

3. Change VTS role, User Groups assignment, Mask, or Email, as needed.
4. Click **Change password here** to update the user's password.
5. Click **Update.**

Delete a user manually

If you delete a user to which permissions have been associated, all permissions linked to the user are also deleted.

1. It is generally recommended to uncheck "Active" status rather than delete users. Log into the VTS GUI, and select **Users** from the navigation bar (Figure 2).
2. Select the check box by a user, click **Delete**, and confirm Delete.

Importing users 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

AD/LDAP user import prerequisites

Before you import users complete the following tasks:

- 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 45.
- Create the appropriate user groups in the VTS GUI, as described in "Creating user groups" on page 57.
- Set up the 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 that has been designated as the "Superuser" group in the CLI will have read/write access in the GUI, whereas the group that has been designated as the "Staff" group in the CLI will have read-only access in the GUI.
2. Populate each group with the desired REST API ("active") users and VTS administrators.

NOTE: The name with permission to access the REST APIs must match that specified 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 mae of the name of the group whtch 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_user and super_ users should actually be granted REST API permissions and disallow by policy when necessary to maintain corporate standards or separation of duties.

3. 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 that 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.

Unlocking user accounts

As a security feature, users are locked out after six failed login attempts.

If a user fails to login after six attempts, the account is locked for 15 minutes (to prevent brute force attacks). The maximum number of attempts and cool-off time duration can be modified in the CLI.

To unlock a user account:

1. Click **Logs** in the navigation menu.
2. Select the **Locked Accounts** tab.
   The locked accounts are displayed
3. Click **Unlock** for the desired account(s).

NOTE: All successful and failed login attempts are displayed under the **Access Logs** tab in the **Logs** menu. Select **Success** or **Failed** from the pull-down menu to filter the list so that only the selected access type is shown

Specify DSM keys

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

Creating a key name

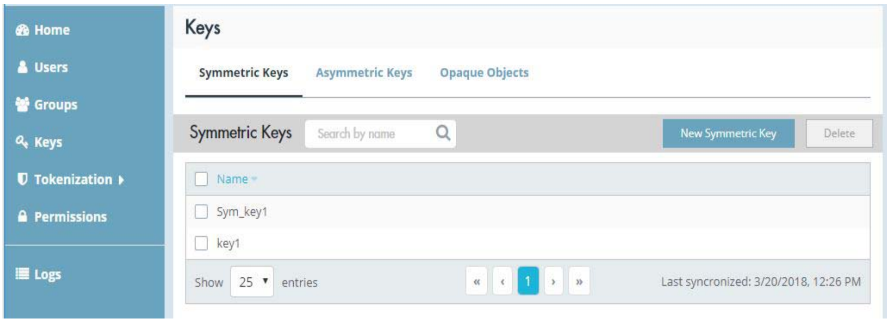
Use the VTS GUI to enter a key name that corresponds to a key that has been created in the DSM and is to be used for VTS functions.

After the name has been entered into the VTS local database, the key is available to associate with user and user group permissions. Symmetric keys are 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. After "Logging into the VTS GUI", 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 10: Keys List in the Symmetric Keys Tab



2. Select the tab for the type of key you want (for example, **Symmetric Keys**) and click the **New Symmetric Key** button.

IMPORTANT: In VTS, the very first key generated cannot be renamed, edited, or destroyed. There is a warning when creating the key. Once this initial setup is complete, you create subsequent keys with the normal *New Symmetric Key* UI, for application FPE, and the like.

3. Enter the key name that was used in the DSM 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.

Deleting key names

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

1. 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 10).
2. Select the tab for the type of key you want (for example, **Symmetric Keys**), click the check box by the desired key, and click **Delete**.
3. 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

Review "Tokenization Groups, Templates, and Masks" on page 13 for the concepts behind setting up tokenization in the VTS GUI.

IMPORTANT: Before setting up tokenization make sure you have specified the DSM keys

Although the GUI does not constrain you to one particular implementation path, one logical sequence is:

- "Specify DSM keys" for tokenization (must be symmetric keys).
- "Creating tokenization groups" and associate it with the appropriate key(s).
- "Creating custom character sets" as needed (optional).
- "Creating a data mask" for detokenization.
- "Creating tokenization templates" and associate the applicable tokenization group and character set.
- **Create users and user groups** as needed.
  (See "Managing VTS User Groups" on page 57 and "Managing VTS Users" on page 59.)
- **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 "Assigning key-based permissions" on page 75.)
- **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 8 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.

Managing tokenization groups

A *tokenization group* is a way 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.

Use case 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 115).
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:

1.Decide how many tokenization groups are required and what their precise names will be (Application programmer and VTS GUI Administrator)

2. Create tokenization groups that match the names to be used in the application code: `Store1`, `Store2`, and `Store3`.
(VTS GUI Administrator)

3. Go to the DSM and:

4. Create tokenization group encryption keys.

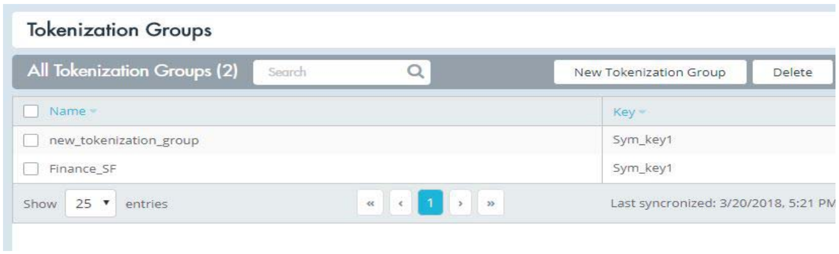
- Use key *Template*: **Default_SQL_Symmetric_Key_Template** for those keys
- Set the *Key Type* to **Cached on Host**.
  IMPORTANT: On a DSM (version 6.1 or later), make sure to choose the CBC Encryption Mode and not CBC_CS1.
  Be sure the administrator knows that the first key must be created on the first node installed, not on secondary or subsequent cluster nodes. **Do not delete this key.**
  (DSM Administrator)

5. 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)

Creating tokenization groups

Have at least one symmetric key created in the DSM and entered in the VTS GUI, as described in "Specify DSM keys" on page 64.

1.Select **Tokenization** > **Tokenization Groups**.
The All Tokenization Groups list is displayed.


Tokenization Groups

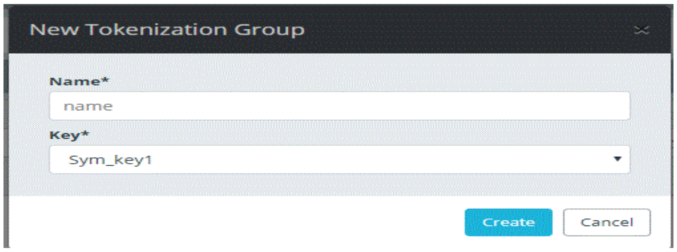
All Tokenization Groups (2)

Name	Key
new_tokenization_group	Sym_key1
Finance_SF	Sym_key1

Show 25 entries

Last syncronized: 3/20/2018, 5:21 PM

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


New Tokenization Group

Name*: name

Key*: Sym_key1

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**.

Deleting tokenization groups

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.

Warning! Once the tokenization group used to create these tokens has been deleted, they can no longer be detokenized.

To delete a tokenization group:

1. Log into the VTS administration interface.
2. Select **Tokenization** > **Tokenzation Groups**.
3. Select the group and click **Delete**.

Managing character sets

VTS now includes enhanced support for tokenizing international characters. In earlier versions, VTS supported three default character sets:

- All digits
- Alphanumeric
- All printable ASCII (not available for RANDOM Mode)

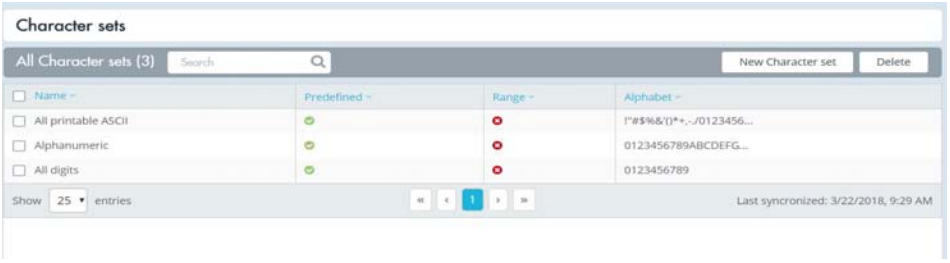
For FPE (FF3) and FF1 modes, you can also define a custom character set, which allows you to tokenize and detokenize UTF strings.

Creating custom character sets

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.

To create a custom character set:

1.Select **Tokenization** > **Character sets**.



Character sets

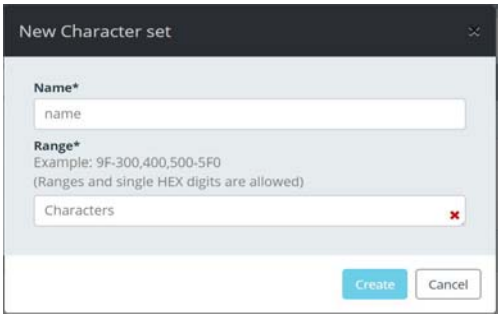
All Character sets (3)

Name	Predefined	Range	Alphabet
All printable ASCII	✓	✗	!"#$%&'()*+,-./0123456...
Alphanumeric	✓	✗	0123456789ABCDEFG...
All digits	✓	✗	0123456789

Show 25 entries

Last syncronized: 3/22/2018, 9:29 AM

2. Click **New Character set**.



New Character set

Name*
name

Range*
Example: 9F-300,400,500-5F0
(Ranges and single HEX digits are allowed)
Characters

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 appears in the All Character Sets list, and is available to apply to tokenization templates.

Editing character sets

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

Applying character sets

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 appears in the tokenization template creation menu as well.

NOTE: See "Creating tokenization templates" on page 71.

Deleting character sets

Before deleting a character set, make sure that tokens created with this character set are not stored anywhere within (or outside) of the organization.

Caution: After 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.

To delete a character set:

1.Navigate to **Tokenization** > **Character Sets**.

2. Select the character set from the list, and click **Delete**.

Prototyping tokenization

VTS 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 as shown here:

```
--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
```

Managing 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.

Creating tokenization templates

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`.
- **Token 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.

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.

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.

- **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.
- **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.

- **Irreversible** - (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.
- **Allow empty or 1 character inputs** - Enables null, empty, and one-character inputs, which are passed through. The token value returned is the orginal input value. The single-digit input supports both characters and numerals, and is available for all formats except RANDOM.

4. Click **Create**.

Minimum input characters with keepleft or keepright

If you configure using keepleft or keepright, note the following minimum numbers of input characters, according to tokenization format:

- For FPE: 2 characters plus the number of keepleft or keepright characters
- For RANDOM Numeric: 9 characters plus the number of keepleft or keepright characters
- For RANDOM Alphaumeric: 5 characters plus the number of keepleft or keepright characters

IMPORTANT: 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. This is shown in the following 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
```

Deleting tokenization templates

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.

Caution: After 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.

To delete a tokenization template:

1.Navigate to **Tokenization** > **Tokenization Templates**.

2. Select the tokenization template from the list, and click **Delete**.

Managing 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.

Creating a data mask

Review “Using data masks” on page 15**.**

To create a mask:

1., Select **Tokenization** > **Masks**.
The All Masks list is displayed.

2. Click **New Mask** and enter the information on the *New Mask* dialog.
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 Left** and **Show Right** values to 999999.

3. Click **Create**.

Deleting 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

Review the following sections:

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

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

NOTE: See additional help on the *Permissions* page by clicking the "**i**" icon in the upper right corner.

Assigning 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.

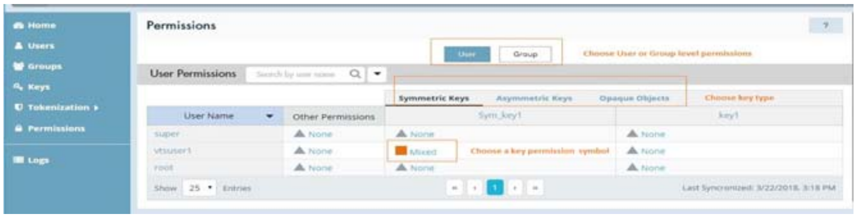
Before permissions can be assigned, you must:

- Create user groups and users
- "Specify DSM keys"

Use the following steps to assign key-based permissions:

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).

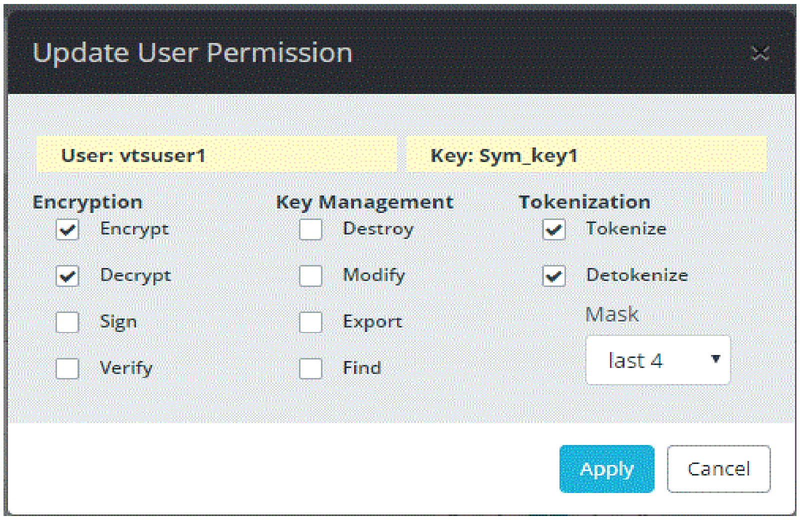
Figure 11: User Permissions List Page



User Name	Other Permissions	Sym_key1	key1
super	▲ None	▲ None	▲ None
vtsuser1	▲ None	🟧 Mixed	▲ None
root	▲ None	▲ None	▲ None

Last Syncronized: 3/22/2018, 3:18 PM

2. Click a user or group 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.


Update User Permission

User: vtsuser1

Key: Sym_key1

Encryption
- [✓] Encrypt
- [✓] Decrypt
- [ ] Sign
- [ ] Verify

Key Management
- [ ] Destroy
- [ ] Modify
- [ ] Export
- [ ] Find

Tokenization
- [✓] Tokenize
- [✓] Detokenize
- Mask: last 4

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

Assigning other permissions

The permissions in the "Other Permissions" category are those that are not tied to specific keys. They are assigned only once per user group, and apply to all key groups. Actions that are not supported in the API are not displayed.

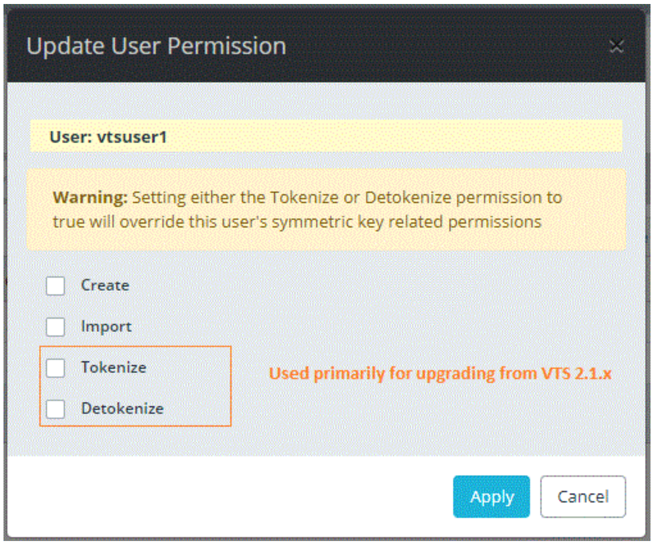
Creating and importing keys

IMPORTANT: Review "Creating static and dynamic keys" on page 12.

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

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



Update User Permission

User: vtsuser1

Warning: Setting either the Tokenize or Detokenize permission to true will override this user's symmetric key related permissions

- [ ] Create
- [ ] Import
- [ ] Tokenize
- [ ] Detokenize

Used primarily for upgrading from VTS 2.1.x

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

NOTE: The **Create** permission is assigned only once per user and applies to all type of keys.
The **Import** permission is assigned only once per user and applies to symmetric keys and opaque objects.

Other permissions: tokenize/detokenize

Select tokenize/detokenize to apply 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 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 the more granular structure of VTS configuration options .

Dashboard API Monitoring

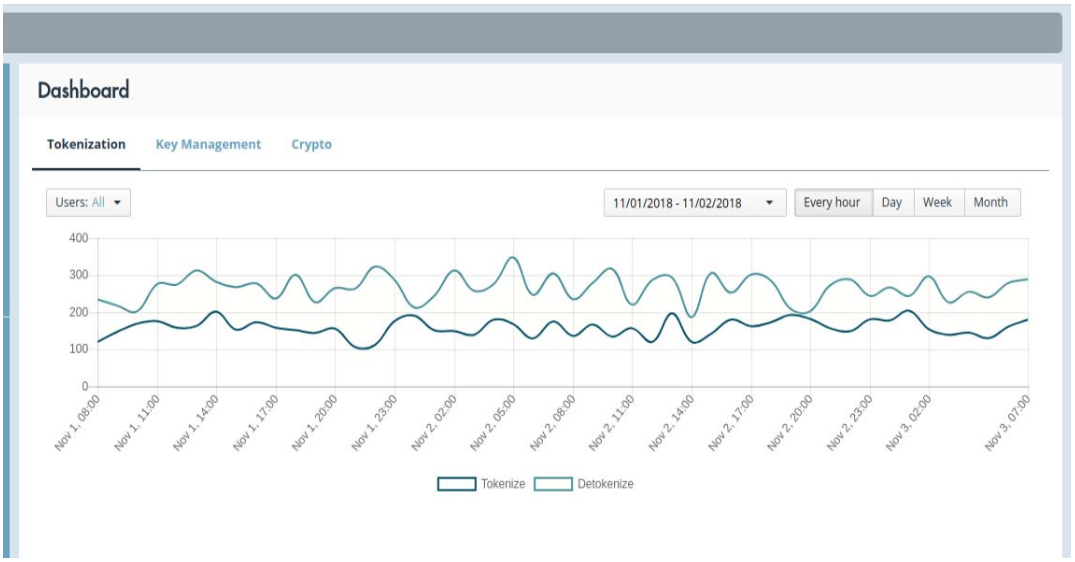
The VTS Dashboard provides visual monitoring for:

- Tokenization activity
- Key Management activity
- Cryptography activity

Activity data is updated hourly, and can be sorted and filtered by:

- User
- Date range
- Hour
- Day
- Week
- Month

Figure 12: API Activity on the Dashboard



Administrator History Logs

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.

IMPORTANT: Log data added by the server or server framework are not included in the log files

---

View the Tokenization Server logs by clicking **Logs** icon in the Tokenization Server GUI. (Audit log messages also go to the syslog with the facility code `user`.) From here you can:

- View audit logs, access logs (login and logout attempts), active sessions, and locked accounts
- Export the logs

View audit logs

To view the audit logs:

1. Log into the VTS GUI.
2. Click **Logs** in the navigation pane.
3. Click the **Audit Logs** tab.
   Details are shown for each audited event, including:
   - **Date** 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 Name** is the specific database object that was created or changed.
   - **Modification** is the action that was performed.

NOTE: To set the VTS log level, see "Set Log Levels" on page 123.

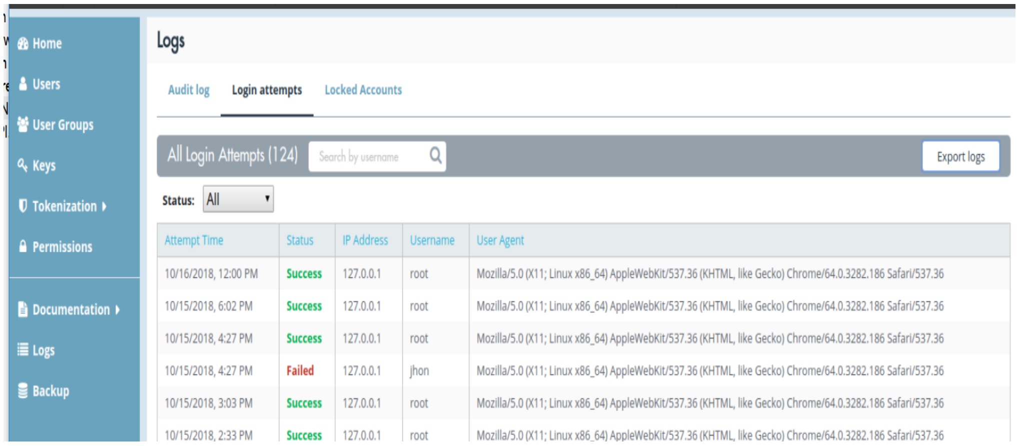
View access logs

The access log shows information about all attempts to log into the server, and all logouts from it. The related informaiton includes the access time, the IP address, username, and user agent (typically, a web browser). By default, all login attempts are listed, but the list can be filtered (by Status) to show only successful or only failed attempts.

To view the access logs:

1. Log into the VTS GUI.
2. Navigate to **Logs**.
3. Click the **Access Logs** tab.
4. Filter the listed results with the **Type** pull-down menu. When viewing the **Logins** access type, use the additional **Status** pull-down menu to filter by Successful or Failed login attempts.

Figure 13: Logs Page Showing Login Attempts



Attempt Time	Status	IP Address	Username	User Agent
10/16/2018, 12:00 PM	Success	127.0.0.1	root	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36
10/15/2018, 6:02 PM	Success	127.0.0.1	root	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36
10/15/2018, 4:27 PM	Success	127.0.0.1	root	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36
10/15/2018, 4:27 PM	Failed	127.0.0.1	jhon	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36
10/15/2018, 3:03 PM	Success	127.0.0.1	root	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36
10/15/2018, 2:33 PM	Success	127.0.0.1	root	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36

Download access or audit logs

To download the history logs for an administrator:

1. Log into the VTS GUI.
2. Navigate to **Logs**.
3. Click the **Audit Logs** or **Access Logs** tab
4. Click the **Export** button for the Audit or Access Logs.

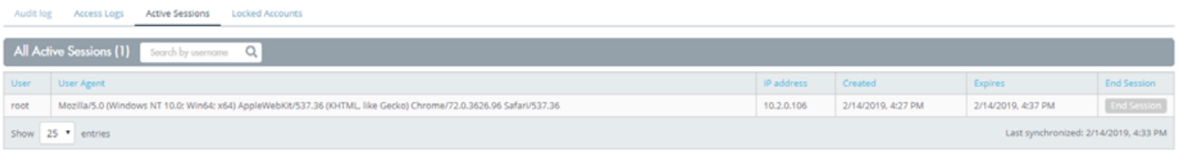
Show active sessions

To see a list of active sessions of users currently logged into the server:

1. Log into the VTS GUI

2. Click **Logs** in the navigation pane.
3. Click the **Active Session** tab.
4. To end a user's session on the GUI, click the **End Session** button in the row for that user.

Figure 14: List of Active Sessions



User	User Agent	IP address	Created	Expires	End Session
root	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.96 Safari/537.36	10.2.0.106	2/14/2019, 4:27 PM	2/14/2019, 4:37 PM	End Session

Show 25 entries

Last synchronized: 2/14/2019, 4:33 PM

Show locked accounts

If a user fails to enter valid credentials more than five times, the user is locked from accessing the VTS for 15 minutes. View the details of locked accounts, including when the lock was activated and when it was released. Superusers can unlock an account before the 15 minute time limit is completed.

To view and manage locked accounts:

1. Log into the VTS GUI.
2. Click **Logs** in the navigation pane.
3. Click the **Locked Accounts** tab.
4. Unlock an account by clicking **Unlock** in the row of the desired user.

System logging

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`.

Back Up and Restore the VTS

Create regular backups of the server to restore the VTS in the event of failure.

Even in a clustered scenario, system failure can occur. To minimize this risk back up your VTS implementation regularly using the administration interface or the REST API.

Use the VTS GUI to create a backup before upgrading your VTS version, and before any major server configuration changes. Maintain a regular backup schedule using the VTS REST API.

Restore the state of your VTS implementation using the administration interface. Backups restore the VTS to the saved version in the following scenarios:

- Restoring the application after system failure
- Restoring the application after upgrade failure
- Restoring the application due to unrecoverable system configuration errors.

NOTE: A backup cannot restore a system using a VTS version older than 2.2.2. A backup cannot be applied to any version of the VTS software that is older than the version in use when the backup was created.

Back up the VTS in the GUI

Use the administration interface to create and save a backup.

Use the following steps to create a system archive:

1. Select **Backup** from the administration interface left hand navigation.
2. Activate the **Create Backup** tab.
3. Accept the default values, and click **Backup now**.

The archive generates and is saved to your default download folder. The file name is created with the following syntax:

`backup_`*`year-month-day`*.

For example: `backup_2018-10-20`

Move the backup to your storage directory.

Managing backup tasks

Use the REST API to create a backup and automate system backup tasks.

You can use the Backup API to generate backups via any valid HTTP request. The syntax requires authentication using user credentials.

See the reference documentation for more information.

Restoring the VTS from a backup

If a complete system restore is required, make sure the VTS installation is complete before beginning this task.

NOTE: Make sure the VTS version leveraged for the backup is the same as the backup or newer than the backup, and is at least version 2.2.2

Use the following steps to restore the VTS to it's last recorded working state using a backup.

1. Select **Backup** from the administration interface left hand navigation.
2. Activate the **Restore** tab.
3. Click **Choose File** and navigate to the required backup archive.
4. Confirm the following back up details:
   - VTS Version
   - Backup date
   - DSM hostname
   - Encryption key name
5. Click Restore.

NOTE: Depending on the size of the backup, the restore process may take some time.

If this node is a member of an existing cluster, it is removed from the cluster. Any existing data on this node is destroyed.

A new cluster is created with this node as the first node but with data restored from the backup.

When the restore completes, this node has all data from the backup including login credentials.

NOTE: If user forgot the login credentials in the backup, run the `createsuperuser` CLI command to create a new user.

This node becomes a new node of a new cluster. It is no longer a member of any previous cluster.

✓

IMPORTANT: If this node was a member of an existing cluster prior to the restore, it must be removed from that cluster. Failure to remove this node from its previous cluster prevents new nodes from joining that cluster.

The VTS is restored to the state of archive. When the task is complete, a banner appears, indicating that the restore is complete. The VTS reboots.

When the login screen appears, enter your credentials to access the restored application.

Key Rotation

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

Versioned keys are not permitted for tokenization.

VTS CLI Reference

5

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

This chapter summarizes the following CLI categories and the commands within each:

- "LDAP Authentication (auth ldap) Category" on page 85
- "Cluster Management Category" on page 86
- "Network Management Category" on page 86
- "System Configuration Category" on page 88
- "VAE Configuration Category" on page 89
- "VTS Configuration Commands" on page 89

This chapter provides a high-level overview of each command category. For complete usage details on each command and its options, run the command with the "`--help`" option. For tips on navigating the CLI, see "CLI navigation cheat sheet" on page 34.

NOTE: To access the CLI, see "Access the CLI through a terminal emulator like Putty." on page 33.

LDAP Authentication (auth ldap) Category

The `auth` category contains only the `ldap` command, whose options configure and review the LDAP Server settings.

As an alternative to specifying the various LDAP settings as options to the `ldap` command (for example, `auth ldap --server ldap://hostname:636`), you can run an interactive wizard that prompts you for each setting. Run this wizard with the `ldap --setup` command.

Cluster Management Category

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

Table 5: Cluster Management Commands

	Description
`add`	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`	Remove a node from a cluster.
`show`	Show cluster settings.

Network Management Category

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 6: Network Category Commands

Command	Description
`arp`	Manipulate the system ARP cache.
`arping`	Send ARP requests to a neighbor host
`checkport`	Check port connection status of a host
`dns`	Show or configure settings for the VTS DNS server(s).
`ethtool`	Query network driver and hardware settings
`ifconfig`	Show or configure a network interface
`ip`	Show or configure the VTS network interface settings.
`netstat`	Print network connections

Table 6: Network Category Commands

Command	Description
`nslookup`	Query internet name servers
`ntpservice`	NTP service
`ping`	Pings an IP address, host name, or FQDN.
`route`	Set static route
`service`	Start/stop/restart network service
`set`	Configure network device interface
`setup`	Network setup wizard
`show`	Show network device configuration
`traceroute`	Traces route to IP address or host name.

Example:

Request:

```
network> ip address
```

Response:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:47:22:72 brd ff:ff:ff:ff:ff:ff
    inet 10.3.201.140/16 brd 10.3.255.255 scope global dynamic eth0
       valid_lft 216sec preferred_lft 216sec
    inet6 fe80::20c:29ff:fe47:2272/64 scope link
       valid_lft forever preferred_lft forever

WARNING! Changes made using this command are not persistent. Use the setup command
to change network settings permanently.
```

System Configuration Category

The `system` configuration category enables you to configure the VTS host settings (such as, hostname, console port, or timezone), and view and manage other common system adminstration tasks for the server (such as, reboot or shut down, or view disk space usage or currently logged-in users).

Table 7: System Category Commands

	Description
`banner`	Configure SSH banner
`cpuid`	Show detailed CPUid information
`cpuinfo`	Show CPU information
`daemon`	Show daemon services that are running
`date`	Print or Set the system date and time
`df`	Report file system disk space usage
`free`	Display amount of free and used memory in the system
`host`	DNS lookup utility
`hostname`	Set/Show Hostname
`hosts`	Update hosts file
`lastlogin`	Show last logged-in users
`meminfo`	Show memory information
`reboot`	Reboot system
`shutdown`	Shut down system
`terminal`	Show or terminate terminal (CLI) sessions
`timezone`	Set or show timezone
`top`	Display Linux processes
`uname`	Print system information
`uptime`	Show how long the system has been running
`vmstat`	Report virtual memory statistics
`who`	Show who is logged into the system

VAE Configuration Category

The vae category includes commands to register the Tokenization Server to the DSM..

Table 8: VAE Management Commands

	Description
`certificates`	VAE certificates utilities
`dsmkeepalive`	Enable or Disable DSM keepalive
`register`	Register the current VTS host to the DSM.
`show`	Show registration information.
`test`	Test to be sure the VTS has been successfully registered with the DSM.

VTS Configuration Commands

The vts category is used to configure the VTS, set time parameters, configure GUI login behavior, and upgrade Tokenization Server software.

The vts category consists of the following commands:

Table 9: VTS Category Commands

Command	Description
`batch_size_limit`	Tokenization Batch size limit.
`cliadmin`	Cliadmin user setup
`client-certificate`	Enables client authentication by requiring clients to pass a client key and CA certificate to VTS.
`cors`	Class-Origin Resource Sharing (CORS)
`create_superuser`	Create a superuser local account for Web administrative GUI access
`logfile`	List, view, or tail log files
`loglevel`	Set log level
`randommode`	Random Mode utility
`rate_limiting`	Limit the amount of tokenization requests

Table 9: VTS Category Commands

Command	Description
`remotelog`	Set remote logging
`restore`	Restore system from backup
`server-certificate`	Creates the self-signed certificate or certificate signing request and import signed certificate.
`service`	VTS Service utility
`show`	Show the security settings.
`ssl-protocol`	Specifies what version of the SSL (TLS) is supported by the VTS.
`upgrade`	Upgrade VTS
`weblogin`	Web Login Setup

Encryption and Key Management API Programming Notes

6

VTS includes three four sets of REST API services: Encryption, Key Management, Tokenization, and Admin. The Tokenization APIs (including the reference documentation, examples, and error codes) are described in the chapter "Tokenization REST API" on page 115. The Admin APIs are described in the chapter "Admin REST API (Beta)" on page 137.

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.

NOTE: The data input format for the APIs is base64.

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 some abbreviated ways to access keys:

Reference keys by label

Quickly get information about a key by using by using the key label in your request: *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, defaults to "key".

Other default key types

In addition, there may be cases where the caller may wish to use an alias, UUIDs, or MUIDs instead of a key label. For these cases, the caller simply needs to refer to 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:7fa96ada-18c6-11e8-8dab-0800277415ff
```

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

TESRN format:

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

By *Label*:

```
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. 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 10: 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 A128GCM - AES 128 GCM cipher mode A256GCM - AES 256 GCM 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 10: (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 DSM and is currently used by the underlying VAE PKCS11 library to implement some extensions that are not available natively on the DSM.

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 becomes the key material used to create the opaque object key. If only a size is given, the VTS cryptographic service generates a key using the DSM 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.

NOTE: 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.

Table 11: Opaque Object Creation Rules

Size	Key Bytes	Result
size = 0 (or omitted)	"" (or omitted)	returns error
size = 0 (or omitted)	present	sets size to the length of key bytes

Table 11: Opaque Object Creation Rules

Size	Key Bytes	Result
size > 0	length does not match size	returns error
size > 0	"" (or omitted)	generates Opaque Object with size bit key
size > 0	length does not match size	returns error

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 2.3 with the VAE 6.2.0 library. The rules for versioned keys are different from conventional symmetric keys. In general:

- 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, deletes 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 - Akey has been suspended
- Compromised - Key is compromised
- Deactivated - Key is deactivated
- Destroyed - Key is destroyed

To set a key to the suspended, compromised, or deactivated state, 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. It does not include permissions related to authorization. See the Authorization section below for more details.

Table 12: 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

Table 12: Valid Key Operations (Continued)

Operation	Symmetric Key	Versioned Symmetric Key	Asymmetric Key	Opaque Object
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
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 key requires that the user has a wrapping key, and that such key is available on both the client and the DSM.

Importing with wrapping key

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

1.Create a 256-bit AES key that is known to the caller and the DSM. This is the wrapping key.

2. Encrypt the key bytes with the wrapping key using an AES-CBC or AES-CBCPAD cipher. This is the wrapped key.

3. 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:

1.Create a 256-bit AES key that is known to the caller and the DSM. This is the wrapping key.

2. Call the export endpoint, supplying a wrapping key ID. This will return a wrapped key.

3. 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.

To import a key using clear bytes, omit the wrapping key ID or give 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

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

- "Creating and importing keys" on page 99
- "Managing keys" on page 100
- "Cryptographic operations" on page 100
- "Authorization operations" on page 100

Creating and importing keys

This category covers both key import and key creation. A user or group that has either import, create, or both is 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

Table 13, “Authorization Operations,” on page 101, shows the possible authorization operations when a key is or was created or imported via the API. The owner is the user who creates the key. This table only applies to authorization. The Valid Key Operations table should be used to determine what are valid operations, without taking authorization into account.

Table 13: 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
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

Table 13: Authorization Operations (Continued)

Operation	Owner	Group	Other user	Other Group
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

Size

All sizes throughout the API refer to size in bytes, with the exception of the key creation/import where the size is in bits. The maximum size for the random generator endpoint in 4096 bytes.

Payload limits

The total payload for any request is 5 MB.

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 return a 200 HTTP status.

All calls return a response body, except for destroy (DELETE) of a specific key which has 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.

For 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 14: 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 15, “ Error HTTP Status Codes,” on page 104.
timestamp	

Table 15: Error HTTP Status Codes

Key Management Examples

Key creation

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

IMPORTANT: You cannot create a key via the REST API and make it a seed key in one step. You must create the key and add it via the VTS GUI in separate steps. You cannot just pass in "seedkey" as a keyword to the create key REST API. If you do, the key is created, but it is not a seed key.

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"
}
```

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":"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 example 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 examples

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/////AAAAA2l
2AAAAABBhYmMxMjNkZWZnaDY3ODkw/////wAAAAVjc3VtAAAAACAbtDVcSesXu7SYqoZ
xNR+hC6Cq8H3gE5xv5vvSzRhUrKcwnBu8", "params": {}}
```

Decryption examples

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"}
```

Tokenization REST API

7

Learn about the VTS Tokenization API in this chapter. Also note that the Tokenization APIs are also described in an online reference guide, under **Documentation** in the VTS Administration GUI navigation bar.

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 eSecurity support.

This chapter includes the following sections:

NOTE: See "Encryption and Key Management API Programming Notes" on page 91 for information about the Encryption and Key Management REST APIs. The Admin REST APIs are described in "Tokenization REST API" on page 115.

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 VTS must be aware of this.

NOTE: Tokenization input can be a normal text format.

Create a Token

Description

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

Table 16: tokenize Endpoint Details

	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 17: tokenize Parameters

			• 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.

Table 18: tokenize Response Parameters

		•**Succeeded** (if request succeeded) •**Error** (if request failed)

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 19: detokenize Endpoint Details

	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 20: detokenize Request Parameters

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

Table 20: detokenize Request Parameters

Parameter	Required	Type	Description
`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 21: detokenize Response Parameters

	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`		**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 “Upgrading VTS” on page 30.) 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:

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 22: log Endpoint Details

	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 23: log Request 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 are 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.”

Table 24: log Response Parameter

	Type	Description
status	string	`Success` (if request succeeded) 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 25: 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 25: 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 DSM	A key name was configured via the GUI and now used in an operation, however this key name does not exist on the DSM .
The token group specified was not found in the database	Consider creating this key name on the DSM .
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 25: 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.

API Reference

For the complete Tokenization API reference, open the VTS GUI and navigate to:

Documentation > **Tokenization API**

Key Management Service REST API

8

Use the Vormetric Token Server (VTS) KMS REST API to manage VTS keys.

Requirements

To use the KMS REST API use:

- The Vormetric Token Server 2.3.0 or newer
- Any DSM version compatible with VTS 2.3.0

Base URL

Following is the base URL for KMS REST API operations:

```
/vts/km/v1
```

Authentication

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JavaScript Object Notation (JSON) object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or MACed and/or encrypted.

Retrieve a Token Request

Request:

```
curl -X POST \
-H "Content-Type: application/json" \
-d '{"username":"admin","password":"Password123!"}' \
https://{server_address}/api-token-auth/
```

Response:

```
{ "token": "eyJ0eXA [...] KnA74" }
```

Refresh a Token Request

Request:

```
curl -X POST \
-H "Content-Type: application/json" \
-d '{"token":"<EXISTING_TOKEN>"}' \
https://{server_address}/api-token-refresh/
```

Response:

```
{ "token": "eyJ0eXA [...] KnA74" }
```

Operations

Table 26: Paths

Path	Operations	Description
/keys	POST	Create or Import a key.
/keys/{id}	DELETE	Delete the path specified key.
	GET	Retrieve the path specified key.
	PATCH	Update the path specified key.
	POST	Export the path specified key.

Object Format

The response format for all requests is a JSON object.

API Reference

For the complete API reference, open the VTS GUI and navigate to:

Documentation > **Key Management API**

Response Codes

HTTP status codes are grouped into five categories:

- 1xx for informational
- 2xx for success
- 3xx for redirection
- 4xx for client errors
- 5xx for server errors

9 Admin REST API (Beta)

The VTS Administration REST API allows you to manage the administration of the VTS. After completing the administrative tasks, you will be able to run the tokenize and detokenize data commands.

To use the Admin REST API you must be using:

- The Vormetric Tokenization Server 2.3.0 or newer
- Any DSM version compatible with VTS 2.3.0

✓ **IMPORTANT:** The Admin REST API is currently available only at a beta level of implementation.

Base URL

Following is the base URL for Admin REST API operations:

```
https://<server_address>/api/
```

Authentication

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JavaScript Object Notation (JSON) object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or MACed and/or encrypted.

Retrieve a token request

Request:

```
curl -X POST \
-H "Content-Type: application/json" \
-d '{"username":"admin","password":"Password123!"}' \
https://{server_address}/api-token-auth/
```

Response:

```
{ "token": "eyJ0eXA [...] KnA74" }
```

Refresh a token request

Request:

```
curl -X POST \
-H "Content-Type: application/json" \
-d '{"token":"<EXISTING_TOKEN>"}' \
https://{server_address}/api-token-refresh/
```

Response:

```
{ "token": "eyJ0eXA [...] KnA74" }
```

Responses

This section describes request response codes and response pagination details.

Response codes

HTTP status codes are grouped into five categories:

- 1xx for informational
- 2xx for success
- 3xx for redirection
- 4xx for client errors
- 5xx for server errors

Pagination

Requests that return multiple items will be paginated to 25 items.

Responses include a "`count`" field, which specifies the total number of objects available via pagination. Responses also include a "`next`" and "`previous`" field. If "`count`" is superior to 25, these fields contain a URL path to access the objects located respectively after or before.

Request:

```
GET https://{{server_address}}/api/v1/users/?limit=100&offset=400
```

Response:

```
HTTP 200 OK
{
  "count": 1023
  "next": "https://{server_address}/api/users/?limit=100&offset=500",
  "previous": "https://{server_address}/api/users/?limit=100&offset=300",
  "results": [ … ]
}
```

Operations

The VTS Admin REST API supports the following operations:

Searching

To search objects, add the attribute ?search= at the end of the URL followed by the pattern you want to look for. All the objects containing that pattern in one of their attributes will be included in the Response.

Request:

```
GET https://{{server_address}}/api/users/?search=robert
GET https://{{server_address}}/api/groups/?search=finance
```

Ordering

To query object in a specific order, add the attribute `?ordering=` at the end of the URL followed by the attribute you want to order by.

Ordering examples:

Order users by username

```
GET https://{{server_address}}/api/v1/users/?ordering=username
```

You may also specify reverse orderings by prefixing the field name with '-', like so:

```
GET https://{{server_address}}/api/v1/users/?ordering=-username
```

Multiple orderings may also be specified:

```
GET https://{{server_address}}/api/v1/users/?ordering=username,email
```

Object format

The response format for all requests is a JSON object.

Creating objects

To create a new object, send a POST request to the class URL containing the contents of the object. For example:

```
curl -X POST \
     -H "{'Authorization':'JWT '+<token>}" \
     -H "Content-Type: application/json" \
     -d '{"name":"User1", ...}' \
     https://{server_address}/api/{some_ressource}/
```

Retrieving objects

To retrieve a list of objects, send a GET request to the class URL. For example:

```
curl -X GET \
     -H "{'Authorization':'JWT '+<token>}" \
     -H "Content-Type: application/json" \
     https://{server_address}/api/{some_ressource}/
```

Updating objects

To edit an object, send a PUT request to the URL containing the new content with the object's ID. For example:

```
curl -X PUT \
     -H "{'Authorization':'JWT '+<token>}" \
     -H "Content-Type: application/json" \
     -d '{"name":"User1", ...}' \
     https://{server_address}/api/{some_ressource}/:id
```

Deleting objects

To delete an object, send a DELETE request to the class URL with the object's ID. For example:

```
curl -X DELETE\
     -H "{'Authorization':'JWT '+<token>}" \
     -H "Content-Type: application/json" \
     https://{server_address}/api/{some_ressource}/:id
```

Backing up VTS

Schedule automatic backups of the VTS.

```
GET https://<VTS_IP_Address>/api/backup/create
```

This request must be preceded by a successful authentication request. Use the secure token provided with the response in the *<auth token>* portion of this example..

Table 27: Set backup Endpoint Details

	Description
HTTP Request	`https://`*`VTS_IP_Address`*`/api/backup/create`
Method	GET
VTS IP Address	The IP address of the VTS to access with the API call.
Response Format	JSON
Required Authentication	Yes

Example cURL Requests

Authentication:

```
curl -X POST\
     -H "Content-Type: application/json" \
     -d ‘{username”:”vtsuser”,”password”:”Password123!”}’ \
     https://vts_server_address/api/api-token-auth
```

Backup request:

```
curl -X GET\
     -H ‘{Authorization”::”JWT <token>”}’\
     -d ‘{username”:”vtsuser”,”password”:”Password123!”}’ \
     https://vts_server_address/api/backup/create -o backup.zip
```

API Reference

For the complete API reference open the VTS GUI and navigate to:

Documentation > **Admin API**