THALES

CipherTrust Transparent Encryption

Data Transformation Guide

for DSM

Release: 7.5.0

Document Version 1

December 19, 2023

CipherTrust Transparent Encryption

Data Transformation Guide

Contents

Preface **5**
Audience 5
Assumptions 5
The CTE Agent Documentation Set 5
Document Conventions 5
Typographical Conventions 5
Notes, Tips, Cautions, and Warnings 6

Chapter 1: Data Transformation Overview **7**
The Data Transformation Process 7
How CipherTrust Protects Files 7
CTE Terminology 8
CTE Components 8
CTE Architecture 9
CTE Policies 9
Considerations with Bulk Data Transformation 10
Data Transformation Techniques 10
Copy and Restore Transformation Method 11
Restore Transformation Method 13
The CTE dataxform Utility Transformation Method 13
CTE Protection Policies 14
Overview of the dataxform Utility 16
Multithreading in the dataxform Utility 18
dataxform Space Requirements 18
dataxform Execution Time 19
dataxform and Sparse Files 20
dataxform and Duplicate Elimination 20
Dataxform and Linked Files 20
Automatic Data Transformation 21
Simplifying dataxform Data Transformation 21
Best Practices When Using dataxform 23
Summary 23

Chapter 2: Initial Data Encryption **24**
Data Encryption Overview 24
How to Decide What Method to Use 24
Restore Encryption Method 25
Copy Encryption Method 25

dataxform Encryption Method 26
Using the Copy or Restore Encryption Method on File Systems 27
Using the Copy or Restore Encryption Method on Block Devices 28
Using dataxform to Encrypt Your Data with the DSM 30

Chapter 3: Rekeying 34
Rekeying Overview 34
Rekeying with dataxform 34
Rekeying Using the Manual Copy Method 37

Appendix A: DataXform Reference 39
Common dataxform examples 39
Displaying dataxform Version Information 39
Verifying that a Guarded Directory Can be Rekeyed with dataxform 39
Estimating the dataxform Runtime Period 40
Manually Running dataxform on Specific Files 40
Running Automatic Data Transformation 42
Automatic Data Transformation Notes and Limitations 42
To run automatic dataxform 42
Cleaning Up a Previous dataxform Session 43
Checking for Hard-Link Files Inside the GuardPoint with dataxform 44
Monitoring dataxform 45
Using the Message Log Window 45
Using vordxf_* Files 45
Using dataxform_status* Files 47
Using dataxform_auto_config 50
Using dataxform_auto_lock 51
Recovering a Failed or Incomplete dataxform Session 51
Restarting an Incomplete Automatic dataxform Session 51
Recovering from a Failed dataxform Session 52
Automatic and Manual GuardPoints 56
dataxform Examples and Full Command Syntax 58
dataxform Command Syntax Examples 58
dataxform Command Parameters 60
Unencrypting Data 63

Preface

The Data Transformation Guide describes how to use CipherTrust Transparent Encryption (CTE) on hosts to transform GuardPoint data from clear text to encrypted text or from encrypted text to clear text.

Audience

The *Data Transformation Guide* is for security teams who want to rekey the existing GuardPoint data, who need to perform an initial encryption of their GuardPoint data, or who need to decrypt their GuardPoint data.

Assumptions

The *Data Transformation Guide* assumes that you have:

- CTE Agents installed and configured on the systems whose data you want to protect.
- Registered every installed agent with Vormetric Data Security Manager.
- Experience creating policies on the key manager you are using.
- Root access to the protected host containing the data to be encrypted or rekeyed.

The CTE Agent Documentation Set

The CTE for DSM guides are available at: CTE for DSM Documentation Site.

Document Conventions

The document conventions describe common typographical conventions and important notice and warning formats used in Thales technical publications.

Typographical Conventions

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

Table 3-1: Typographical Conventions

Convention	Usage	Example
bold regular font	GUI labels and options	Click the **System** tab and select **General Preferences**.
bold italic monospaced font	Variables or text to be replaced	https://<***Token Server name***>/admin/ Enter password: <***Password***>
`regular monospacedfont`	• Commands and code examples • XML examples	`session start iptarget=192.168.253.102`

Table 3-1: Typographical Conventions (continued)

Convention	Usage	Example
italic regular font	GUI dialog box titles	The *General Preferences* window opens.
	File names, paths, and directories	*/usr/bin/*
	Emphasis	*Do not* resize the page.
	New terminology	*Key Management Interoperability Protocol (KMIP)*
	Document titles	See *Data Transformation Guide* for information about CipherTrust Transparent Encryption.
quotes	• File extensions • Attribute values • Terms used in special senses	".js", ".ext" "true" "false", "0" "1+1" hot standby failover

Notes, Tips, Cautions, and Warnings

Notes, tips, cautions, and warning statements may be used in this document.

A Note provides guidance or a recommendation, emphasizes important information, or provides a reference to related information. For example:

> **Note**
> It is recommended to keep tokenization keys separate from the other encryption/decryption keys.

A tip is used to highlight information that helps you complete a task more efficiently, such as a best practice or an alternate method of performing the task.

> **Tip**
> You can also use Ctrl+C to copy and Ctrl+P to paste.

Caution statements are used to alert you to important information that may help prevent unexpected results or data loss. For example:

CAUTION
Make a note of this passphrase. If you lose it, the card will be unusable.

A warning statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data. For example:

WARNING
Do not delete keys without first backing them up. All data that has been encrypted with deleted keys cannot be restored or accessed once the keys are gone.

Chapter 1: Data Transformation Overview

This chapter provides a detailed overview of the data transformation process. It covers the following topics:

The Data Transformation Process 7
CTE Terminology 8
CTE Components 8
CTE Policies 9
Considerations with Bulk Data Transformation 10
Data Transformation Techniques 10
CTE Protection Policies 14
Overview of the dataxform Utility 16
Automatic Data Transformation 21
Best Practices When Using dataxform 23
Summary 23

The Data Transformation Process

Data transformation is used for:

- **Initial data transformation** — The first time you create a GuardPoint with existing data, you can tell CTE to encrypt the existing data. This means that the clear-text data in the GuardPoint becomes encrypted as cipher-text. Any new data you add to the GuardPoint is also encrypted immediately based on the policy applied to the GuardPoint
- **Rekeying** — If you have already encrypted the data in a GuardPoint, you can increase the security of that data by periodically changing the encryption key used to encrypt it. This process is called rekeying.
- **Reverse transformation** — Decrypting GuardPoint data from cipher-text to clear-text. (Not a common procedure.)

In addition to being disruptive to data center operations, data transformation is complex. It is strongly recommended that you read and understand this section before proceeding to the initial data transformation and rekey chapters.

How CipherTrust Protects Files

The CipherTrust Transparent Encryption Agent (CTE Agent) encrypts the data within a file one block at a time. It does not encrypt file metadata such a file's name or size, thus enabling administrators to manage files without being able to view or modify their contents. Whether initially encrypting files, rekeying them, or decrypting them, the CTE Agent must therefore:

- Read each block of file data to be transformed.
- Transform the block by encrypting, decrypting, or rekeying it.
- Write the transformed block, either to its original location, or to an alternate one.

CTE Terminology

The CTE documentation set uses the following terminology:

Term	Description
CTE	CipherTrust Transparent Encryption is a suite of products that allow you to encrypt and guard your data. The main software component of CTE is the CTE Agent, which must be installed on every host whose devices you want to protect. **Note** This suite was originally called Vormetric Transparent Encryption (VTE), and some of the names in the suite still use "Vormetric". For example, the default installation directory is /opt/vormetric/DataSecurityExpert/agent/ for Linux and AIX, and C:\Program Files\Vormetric\DataSecurityExpert\agent\ for Windows.
CTE Agent	The software that you install on a physical or virtual machine in order to encrypt and protect the data on that machine. After you have installed the CTE Agent on the machine, you can use CTE to protect any number of devices or directories on that machine.
key manager	An appliance that stores and manages data encryption keys, data access policies, administrative domains, and administrator profiles. Thales offers two key managers for use with CTE, the Vormetric Data Security Manager (DSM) and CipherTrust Manager.
host / client	In this documentation, host and client are used interchangeably to refer to the physical or virtual machine on which the CTE Agent is installed. The difference comes from the key manager you are using. The DSM refers to the machines as hosts, while the CipherTrust Manager refers to them as clients.
GuardPoint	A device or directory to which a CTE data protection and encryption policy has been applied. CTE will control access to, and monitor changes in, this device and directory, encrypting new or changed information as needed.

CTE Components

The CTE solution consists of two parts:

- The *CTE Agent software* that resides on each protected virtual or physical machine (host). The CTE Agent performs the required data encryption and enforces the access policies sent to it by the *key manager*. The communication between the CTE Agent and the key manager is encrypted and secure.

  After the CTE Agent has encrypted a device on a host, that device is called a *GuardPoint*. You can use CTE to create GuardPoints on servers on-site, in the cloud, or a hybrid of both.

- A *key manager* that stores and manages data encryption keys, data access policies, administrative domains, and administrator profiles. After you install the CTE Agent on a host and register it with a key manager, you can use the key manager to specify which devices on the host that you want to protect, what encryption keys are used to protect those devices, and what access policies are enforced on those devices.

  Thales offers two key managers that work with CTE:

  - CipherTrust Manager, Thales's next generation key manager that supports most CTE features on Linux and Windows, and all CTE features on AIX.
  - The *Vormetric Data Security Manager* (DSM), Thales's legacy key manager that supports all CTE features on Linux, Windows, and AIX.

Both key managers can be set up as either a security-hardened physical appliance or a virtual appliance. Both provide access to the protected hosts though a browser-based, graphical user interface as well as an API and a CLI.

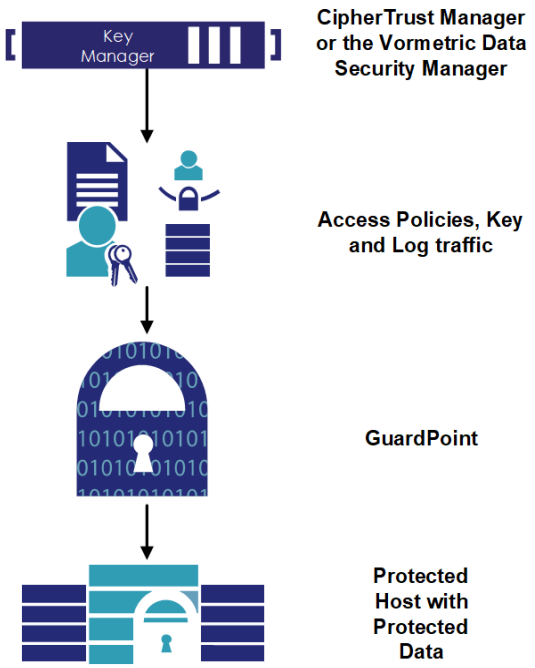
You must select one and only one key manager per host or host group. While you could have some hosts registered with a CipherTrust Manager and some registered with a DSM, you cannot have the same host registered to both a CipherTrust Manager and a DSM.

> **Note**
> For a list of CTE versions and supported operating systems, see the CTE Compatibility Portal or the *Compatibility Matrix for CTE Agent with CipherTrust Manager* and the *Compatibility Matrix for CTE Agent with Data Security Manager*.
>
> All All 7.2 documentation is available at CTE Docs. All 7.3 documentation is available at: CTE Doc Portal

CTE Architecture

A GuardPoint is usually associated with a Linux/AIX mount point or a Windows volume, but it may also be associated with a directory sub-tree.The CTE Agent sits between applications and the file system that hosts files within the GuardPoint. The CTE Agent intercepts every file access request and enforces the access and encryption rules in the policy associated with the GuardPoint.

Figure 1-1: CTE Architecture



CTE Policies

CTE policies consist of ordered lists of rules that specify:

- **Actors:** Users, groups, and processes that are permitted to access protected data.
- **Actions:** The actions available to authorized actors. For example create/delete, read/write, decrypt, modify permissions, and so on.

- **Files acted upon:** Policy rules may apply to entire directories and mount points, or only to files named in a specific way (for example, `.docx` files may be encrypted and restricted to read-only access by designated users, while other files may be stored clear and read and written by anyone).

In addition, each CTE policy specifies an encryption key used to encrypt blocks of file data when applications write them and decrypt them when they are read. (A special type of policy, called a rekey policy, includes an additional key used for re-encrypting data during rekeying.)

CTE encryption is transparent to applications. This means that the CTE Agent encrypts blocks of data as they are written, and decrypts data when they are read by authorized users and applications. This architecture separates administration of files from access to the data in them. Backup programs, for example, may be authorized to read files, but not view the data in them. Therefore, data can be backed up and taken off-site while remaining encrypted so that security is not breached.

Considerations with Bulk Data Transformation

For large file sets (hundreds of gigabytes or more), bulk transformation is time-consuming. Managing transformation time is important, because file set content must be frozen (inaccessible to applications) throughout the transformation process. Once transformation starts, it must continue until complete, so transformation time determines the window of data unavailability.

Two major components contribute to transformation time:

- **Number of blocks of file data** — Because CTE must read, transform, and rewrite each block of file data, this component can be estimated by multiplying the number of file blocks to be transformed by the average read, transformation, and write time for a block.
- **Number of files** — Because the CTE Agent transforms data file by file, each file must be “looked up,” opened, and closed during transformation, using underlying file system mechanisms. This typically requires multiple disk accesses. Therefore, file sets that consist of many small files, per-file overhead, can actually exceed file block transformation time.

Other factors, such as file system fragmentation, and load from concurrent applications, may also affect transformation time. Mainly, the number of blocks and number of files to be transformed are fundamental in that they cannot be reduced or eliminated.

Data Transformation Techniques

Two basic methods are available for initially encrypting and rekeying files protected by CTE:

- **Copy/Restore** — Using the operating system file copy utility, a protected host administrator can copy unprotected files into a location protected by a CTE GuardPoint with a standard production policy. Similarly, files already protected by CTE can be transformed (rekeyed) by copying them from their protected location to another location protected by a different encryption key. For details, see "Copy and Restore Transformation Method" on the facing page.
- **The CTE `dataxform` utility** — Every CTE Agent includes a utility program that can encrypt or transform protected files. The `dataxform` utility encrypts, rekeys, or decrypts data in place. For details, see "The CTE dataxform Utility Transformation Method" on page 13.

Both methods have advantages and limitations that make them suitable in different scenarios.

Notes

- CTE can also be configured to protect data at the disk level. For data protected in this way, only the copy transformation technique is available for encryption.
- Do not run more than one instance of dataxform (per file system) for performance reasons.

The table below summarizes the strengths and weaknesses of the two file set transformation methods.

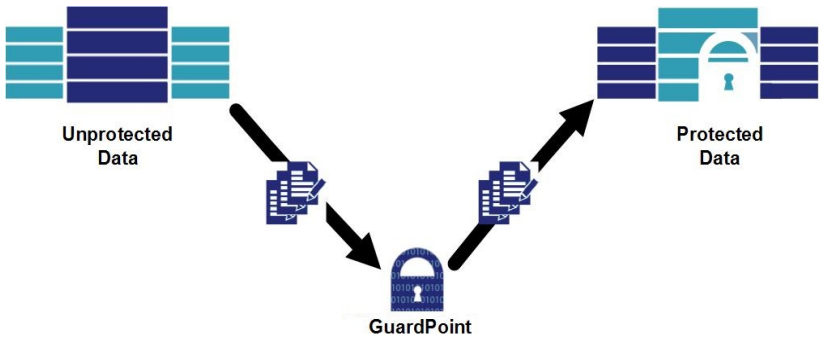
Issue	Copy Method	Dataxform method
Temporary storage required	Equal to size of file set.	Sufficient to hold a list of path names of files in file set.
Security	File data is unprotected while in copy utility's buffers.	File data is never outside the CTE GuardPoint.
Initial encryption	Files can be copied directly from source directory to a CTE-protected directory.	Files must be in a protected location before transformation.
Operational impact	No access to files during transformation. Path names or operating procedures must be adjusted after transformation.	No access to files during transformation. No other impact on operating procedures.
Recoverability	Restart copy operation at, or prior to, point of failure.	Files undergoing transformation at point of failure must be discovered from dataxform logs and restored from backup.

Copy and Restore Transformation Method

The copy method performs initial encryption, rekeying, and decryption by copying data from one directory, or GuardPoint, to another directory or GuardPoint.

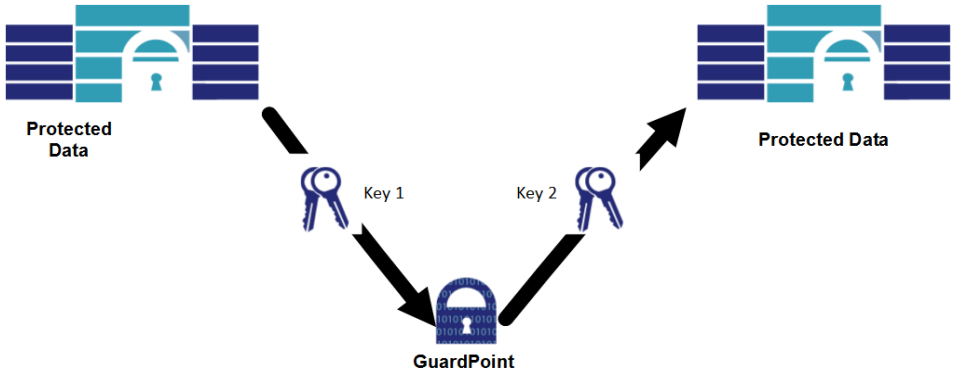
In the following figure, the administrator of the protected hosts encrypts a file set by copying it to a directory protected by a CTE GuardPoint with a standard policy. Encryption is transparent to the copy utility.

Figure 1-2: Initial Encryption



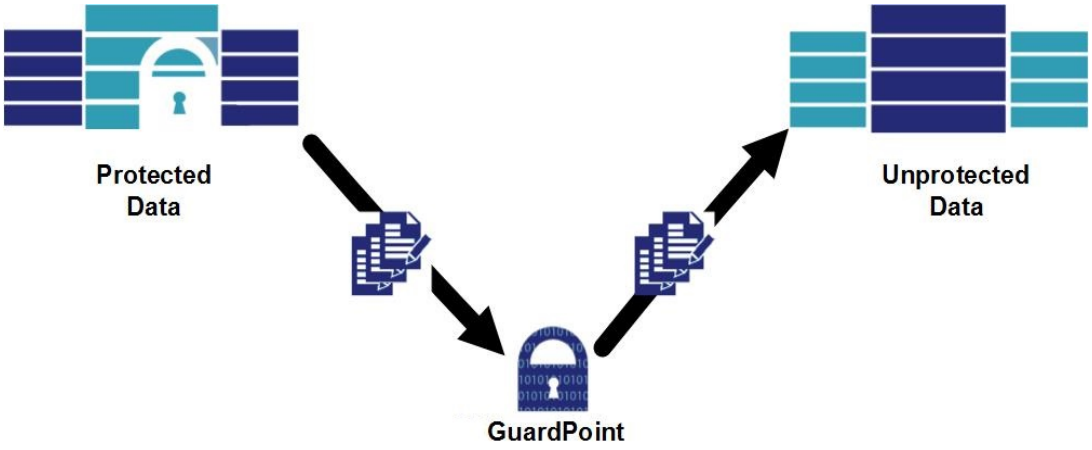
In the following figure, already-encrypted files protected by a CTE GuardPoint are rekeyed by copying them to a directory protected by another GuardPoint with a different encryption key. Both decryption and re-encryption are transparent to copy utilities.

Figure 1-3: Rekeying Protected Data



In the following figure, you can decrypt a protected file set by copying files from their protected location to unprotected directories. The CTE Agent decrypts file blocks before delivery to the copy utility for rewriting.

Figure 1-4: Decrypting Data by Copying



> **Note**
> Exercise care here. If the governing policy does not authorize the copy utility user to access data, CTE delivers encrypted file blocks to it.

Encrypting and rekeying files by copying has two important advantages:

- **Simplicity** — After you have installed a CTE Agent and GuardPoints are activated on a protected host, the protected host's administrator can encrypt, decrypt, or rekey file sets simply by copying them from one location to another. There are no procedures to learn, and no requirement to coordinate with your key manager's Security Administrator. Data transformation is simply another routine administrative task.
- **Recoverability** — If a copy-based transformation is interrupted, for example, by a power failure or system crash, the transformation resumes at or prior to the point of interruption. This is because all of the source files remain available and can be recopied, overwriting files at the destination that may have been only partially re-encrypted.

Offsetting these advantages are two limitations inherent to the copy method:

- **Storage resource consumption** — Copying a file set requires that both source and destination files exist simultaneously. Storage capacity sufficient for both must be available during initial encryption. For very large protected data sets, “extra” temporary storage may be a significant expense. However, a greater concern is likely to be the impact of moving production file sets as they are transformed. File data is unprotected while in the copy utility’s buffers.
- **Impact on operating procedures** — Original and copied file sets have different path names and/or network addresses. After transformation, either both file sets must be renamed (the old path to a new name, and the new path to the old name), or applications must be adapted to process the transformed data set at the new directory. For a small data center with a few protected file sets, some combination of these options is usually practical. For data centers with hundreds of protected file sets, the administrative complexity and consequent chance of error make copying a complex option.

See Chapter 2: "Initial Data Encryption" on page 24 and Chapter 3: "Rekeying" on page 34 for detailed operational information on the copy method.

Restore Transformation Method

A variation of the copy method is to make a backup of the files for transformation and restore the backup to the destination location. This works because:

- Backing up data causes it to be read and decrypted.
- Restoring data causes it to be written (re-encrypting it with an alternative key).
- CTE protection is transparent to backup programs.

This technique also creates a backup of the data set. However, a disadvantage is the time required to copy data twice (once from the source location to backup, and once from backup to destination location).

These considerations suggest that copying data to transform it is more suitable for initial encryption (and final decryption), and less so for rekeying. Additionally, the simplicity of recovering an interrupted transformation makes the copy/restore method useful in situations where the probability of interruption during transformation is significant.

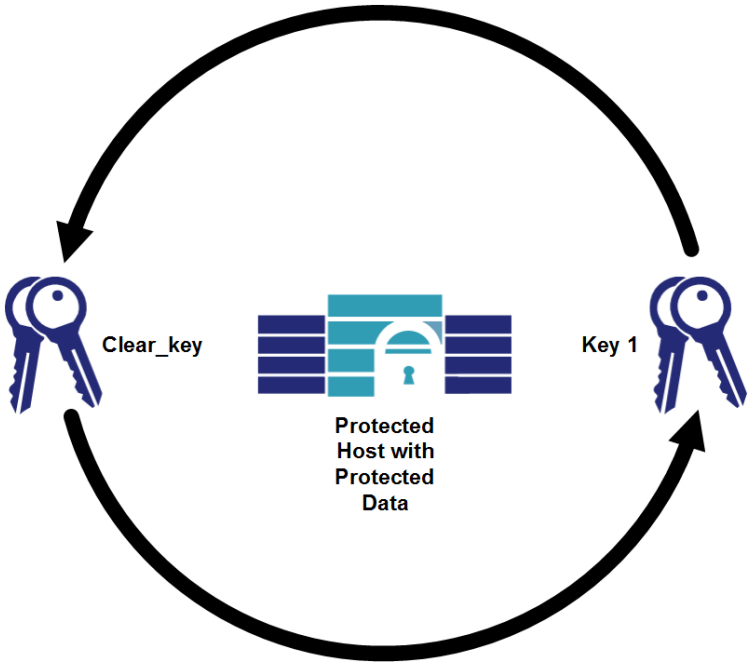
See "Restore Encryption Method" on page 25 for detailed operational information on the restore method.

The CTE dataxform Utility Transformation Method

The CTE `dataxform` utility transforms data-in-place and contains two components:

- User-mode that controls the overall operation.
- Kernel-mode that transforms files block-by-block.

Figure 1-5: Offline Rekey



Transforming data in place has two important advantages:

- **Minimal storage requirements** — Because `dataxform` transforms files in place, where they reside, it does not require temporary file storage. However, the utility does need storage in which to create a list of files for transformation.
- **Security**— The period of time that the data transformed by the `dataxform` utility appears in memory, outside the GuardPoint and therefore, unprotected, is shorter than with copying. This is significant for rekeying (compared to copying), which holds clear file data in memory between reading and rewriting. Moreover, `dataxform` requires coordination between the administrator for the protected host and the Security Administrators for your key manager, so that no one individual can subvert security during transformation.

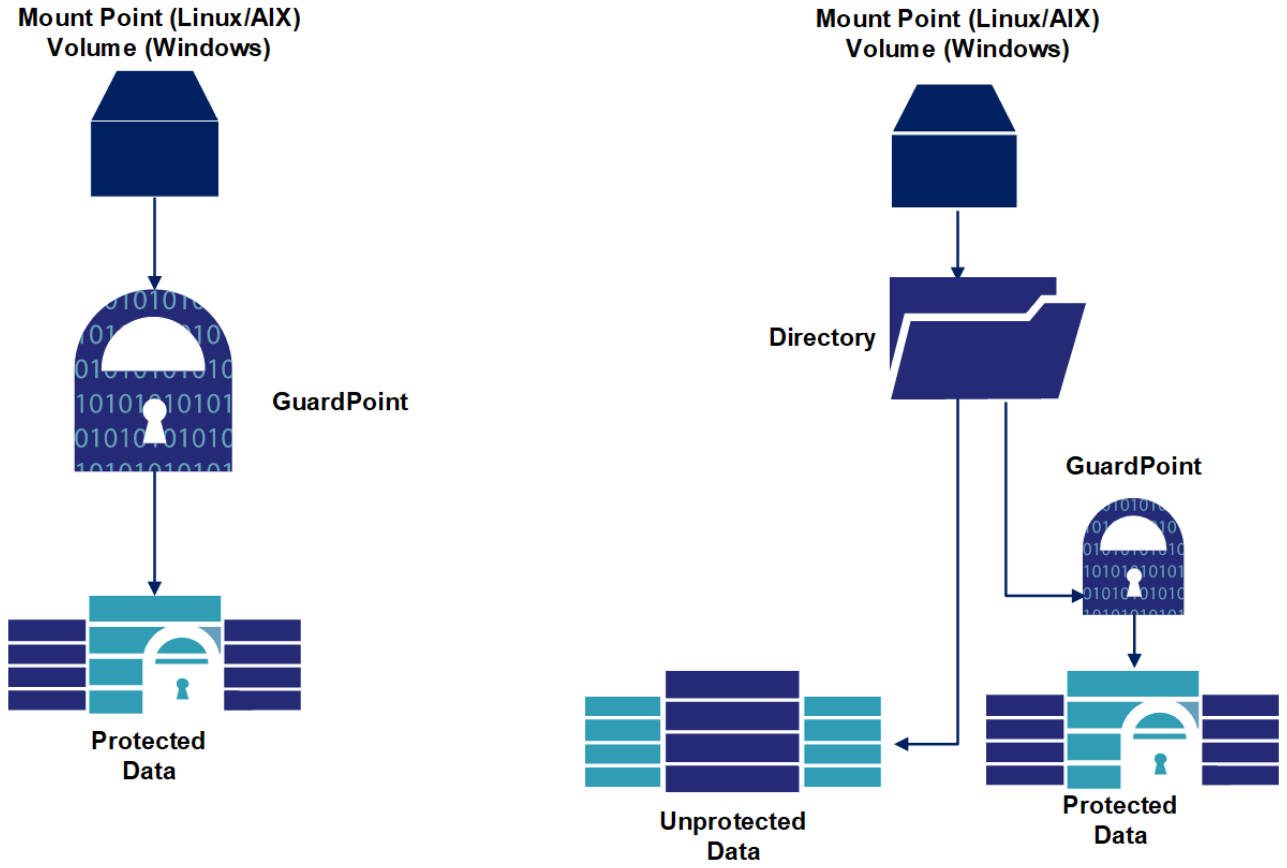
Offsetting these advantages is the complexity of recovering from an interrupted `dataxform` run. Because `dataxform` transforms files in-place, data in a file undergoing transformation at the time of a failure may be only partly transformed. There is no way to determine which blocks have been transformed and which have not. These files must be recreated after the `dataxform` runs from a backup copy. The protected host administrator must determine (by examining `dataxform` logs) which files may have been incompletely transformed, delete them from the transformed file set, and recreate them by selective copying from a backup.

CTE Protection Policies

The basic unit CTE data protection policy application is the GuardPoint. GuardPoints are typically associated with file system mount points, but may also be associated with directory sub-trees.

> **Note**
> Nested mount points within a directory, or mount points protected by a GuardPoint, are also protected in Linux and AIX environments.

Figure 1-6: CTE GuardPoints



All files in the directory hierarchy, below a GuardPoint, are subject to the GuardPoint's policy, which consists of rules that specify:

- **Protected files**: Filenames or filename patterns (example: *.dat) to which the policy applies.
- **Authorized users**: User(s) group(s), and application(s) permitted to access the protected files.
- **Permissions**: Actions permitted to users (example: create/delete, read/write, rename, decrypt).

Policies also specify the name of an encryption algorithm and a key for encrypting protected files. For example, a policy might specify that all Excel workbooks protected by a GuardPoint be encrypted using an AES256 key called EXCEL-KEY. Additionally, only users in group 128 have access to the files. All other files that are not encrypted, are freely accessible to all users.

CTE Agents use two types of policies:

- **Initial Data Transformation** — Dataxform policies contain the elements listed above, plus a data transformation key, used by the `dataxform` utility to rekey file data. Transformation policies contain strict access control rules that prevent application and user access to files during transformation. CTE only uses Dataxform policies for the initial transformation. Afterwards, you replace it with a production policy.

  Dataxform operates on a per-GuardPoint basis. For *initial encryption*, the `dataxform` policy specifies a `clear` production key (meaning that the utility does not decrypt data because the data is unencrypted) and a new data transformation key to encrypt the data.

- **Production/Standard** — Production policies contain the elements listed above. They protect data within GuardPoint(s) during day-to-day IT operations.

For *decryption*, the policy specifies a clear data transformation key (that is, the utility does not re-encrypt files as it rewrites them) and the current production key. A *rekeying transformation* policy specifies both a current production ("old") key and a transformation ("new") key.

The following tables show various policy components of a typical `dataxform` rekey policy.

Security Rule (Policy Rules)

Order	Resource	User	Process	Action	Effect	When	Browsing
1				key_op	Audit, Permit,		Yes
2				all_ops	Deny		Yes

Key Selection Rules (Production/Standard)

Order	Resource	Key
1		Original_Key

Data Transformation Rules (Key)

Order	Resource	Key
1		New_Key

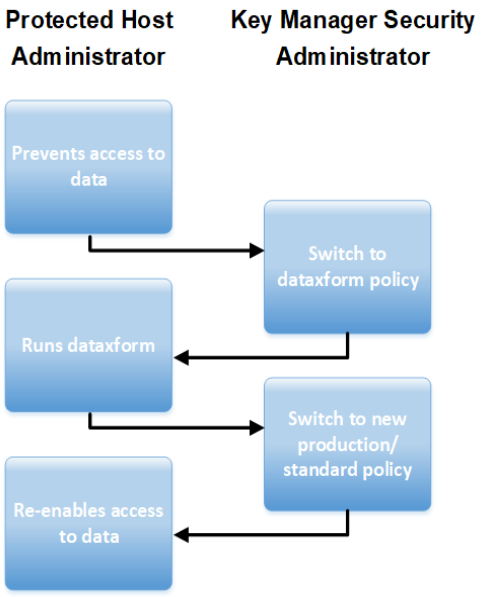
Overview of the dataxform Utility

The `dataxform` utility is installed on a protected host during CTE Agent installation. The utility:

- Reads each file block-by-block
- Uses the production key to decrypt each block
- Re-encrypts it with the data transformation key
- Rewrites it to its original location.

The following figure illustrates that transforming data with `dataxform` requires collaboration between the Protected Host Administrator and the Security Administrator for the key manager with which the host is registered.

Figure 1-7: Administrator Collaboration



1. The protected host administrator disables access to the files to be transformed, by stopping applications and/or databases that use them. They inform the key manager Security Administrator when they are inaccessible.
2. The key manager Security Administrator creates a `dataxform` policy with appropriate encryption key(s) and applies it to the GuardPoint.
3. The protected host administrator runs the `dataxform` utility on the GuardPoint directory with the appropriate parameters and options, and informs the key manager Security Administrator when the utility completes.
4. The key manager Security Administrator replaces the `dataxform` policy with a production/standard policy (or an initial test policy used to create the production policy). These policies use production keys that are the same as the encryption key used in the `dataxform` policy. After the switch, the key manager Security Administrator informs the protected host administrator.
5. The protected host administrator re-enables user access to the protected data.

Protected host and key manager Security Administrators must coordinate with each other to transform protected data sets using `dataxform`. While this makes it impossible for a single individual to subvert CTE security, close coordination can be difficult to arrange, particularly in large data centers with many protected hosts administered by different individuals. To partially relax this requirement without compromising security, `dataxform` execution can be automated. See "Automatic Data Transformation" on page 21.

In addition to rekeying protected files, you can use `dataxform` for the initial encryption and decryption of file sets. For initial encryption, the key manager Security Administrator specifies `clear_key` as the transformation policy's encryption key and a new encryption key as the production key. This instructs the utility not to decrypt files before "re-encrypting" them. Similarly, to decrypt protected data, the key manager Security Administrator specifies `clear_key`, as the transformation key and the existing encryption key as the production key, causing `dataxform` to decrypt, but bypass re-encryption.

The `dataxform` utility starts by creating a list of all files that may require transformation. To guarantee that the list remains correct until the transformation is completed, the admin must prevent all file access by including an "`all_ops`" "`deny`" rule in the policy. Without this rule, `dataxform` does not start.

When using `dataxform`, it is important to keep in mind the following issues:

- Ensure that the pre-transformation production policy, the `dataxform` policy, and post-transformation production policy, all specify the same files to be affected.
- Similarly, the `dataxform` policy must specify the same production key as the pre-transformation production policy. The post-transformation production policy must specify the same production key as the `dataxform` policy's transformation key.

> **Note**
> CTE does not cross-check key relationships. This is the responsibility of the key manager Security Administrator.

Multithreading in the dataxform Utility

The `dataxform` utility is almost always I/O bound. You can reduce end-to-end run time by configuring the utility to transform multiple streams of data concurrently, in separate kernel threads. `dataxform` can be multi-threaded in two dimensions:

- **File concurrency** — `dataxform` can transform up to 32 files in concurrent execution threads. Each time a kernel thread finishes transforming a file, it informs the user component, which responds with a command to transform the next file in its work list. Number of threads is set with the `--thd` option.
- **File chunking** — You can also configure the kernel component to divide individual files into chunks and transform up to 16 chunks concurrently. The chunk size defaults to 128 KB, but you can adjust it using the `--buf_size` option.‘

File concurrency is useful for transforming large numbers of files, but less-so with file sets that consist of a few large files. For the latter, chunking is typically more advantageous.

Concurrent transformation reduces run time, and therefore the period during which protected files are unavailable to applications. On the other hand, because files undergoing transformation at the moment of a system crash must be recovered from a backup, more active files means more time-consuming post-run recovery. Moreover, more concurrent transformation activity consumes more processing, memory, and I/O resources, which are unavailable to other applications running concurrently.

dataxform Space Requirements

Because it transforms data in place, `dataxform` must run to completion once it starts. If the utility does not run to completion, some files will have been completely transformed, some will not have been transformed, and some will only be partially transformed. Transformed files will be encrypted with the transformation key, not-yet-transformed ones with the pre-transformation production key, and those undergoing transformation will be in an indeterminate encryption state. To enable successful completion of an interrupted transformation, `dataxform` adopts two strategies:

- **Master file list**

  Before transforming the files in a set, the utility makes a disk-based list of path names. The list determines the order of transformation, and is also used to determine the restart point if transformation is interrupted. When `dataxform` finishes transforming a file set, it deletes the master file list, so only temporary storage for the list is required.

- **Status logging**

  Each time `dataxform` finishes transforming a file, the utility records the status of the transformation in the disk-based status file. Status includes the path names of files being transformed at the time of recording. This enables `dataxform` to restart after interruption and recover incompletely transformed files from a backup.

For large directories (e.g., those containing 100,000 or more files) the size of the `dataxform` file list can run to tens of gigabytes (the size is largely determined by file path name lengths). By default, `dataxform` stores its master file list in the directory in which it writes log entries. Before running `dataxform`, the protected host administrator should ascertain that the file system containing the logging directory contains sufficient space to hold a list of full path names within the GuardPoint (see "Monitoring dataxform" on page 45). If this is not practical, the administrator can designate an alternate location for the list and status files as a `dataxform` command line option.

dataxform Execution Time

During transformation, a file set must remain static. Therefore, it is inaccessible to users and applications. Once started, transformation must complete before admins can permit applications access to the transformed files. This includes any restarts and manual recovery of incorrectly transformed files. In effect, the transformation process determines the duration of the outage. There are two elements to consider when choosing a window of time during which transforming a data set does not adversely affect the business function it supports:

- **Length of run** — The run time, assuming that the run is problem-free.
- **Success of run** — Maximizing the chance of success (and therefore minimizing the need for time-consuming manual recovery).

Length of Run

The `dataxform` utility includes a dry run capability that uses a combination of sampling and calculation to estimate the duration of a problem-free run against a given file set. A dry run can execute while data is online, however, this results in less accurate runtime estimates. It counts both the number of files in the set and the amount of data they contain. It also performs some sample transformations of dummy files to estimate how long an actual transformation would run. The result of a dry run is an estimate of `dataxform` run time. In most cases, however, the estimate is conservative, provided that other system activity is minimal during the actual transformation. See "Estimating the dataxform Runtime Period" on page 40.

Success of Run

To maximize the chances of successful transformation, the protected host administrator should ensure that the required resources will be available during the run:

- **Storage space** — See "dataxform Space Requirements" on the previous page.
- **Kernel threads** — The utility uses kernel threads for actual file data transformation. The protected host administrator can specify as many as 512 concurrent file and data chunk transformation threads. The admin must also ensure that sufficient kernel threads are pre-configured in the operating system (usually at system startup time) so that the specification can be met in the presence of other system activity occurring during transformation. See "Multithreading in the dataxform Utility" on the previous page.
- **Processing power and I/O bandwidth** — You can maximize the power and bandwidth by limiting other system activity during transformation. I/O bandwidth, particularly disk accesses, is especially important for sets containing large numbers of files, because each file must be located and opened (both of which require disk accesses) in addition to having its data read and overwritten.

> **Notes**
> - Minimizing `dataxform` run time should be a priority because once the utility starts, it must transform the entire data set before users and applications can access it again.
> - Do not stop `dataxform` after it is started, as this may cause issues.
> - Make sure that files are not open before starting dataxform. If a file is busy, data might become corrupted.

dataxform and Sparse Files

Sparse files are files in which storage space is allocated in file block addresses, into which data is written. Most Linux and AIX file systems do not allocate space for file blocks until the blocks are actually written. Thus, if an application creates a file and writes the first and 1000th blocks, the second through 999th blocks are represented as a hole in the file system's data structures. When an application reads file blocks that have never been written, the file system returns zeros. Application reads from holes are thus indistinguishable from reads of file blocks that actually contain zeros. When an application writes data to file block addresses in the midst of a hole, the file system allocates storage space for the data, subdividing the hole into two smaller ones if necessary.

The CTE Agent and the `dataxform` utility cannot distinguish a file block that contains zeros from a hole—both return blocks containing zeros when read. When `dataxform` decrypts and re-encrypts such blocks and writes them back to their original locations, file systems allocate storage space for blocks that may previously have been holes. For large, mostly sparse files, this can result in run times and storage consumption that far exceed expectations based on pre-transformation file sizes. Therefore, `dataxform` provides administrators with two options for dealing with sparse files:

- **Recognize holes** — A protected host administrator can configure `dataxform` to detect and bypass the processing of file blocks that contain all zeros. The result is that holes remain holes, and file blocks that contained zeros prior to transformation continue to contain zeroed after transformation. Application reads of either return decrypted zeroed, and applications' first writes cause the file system to allocate storage and write encrypted data to them. See the `--preserve_sparse_files` option in the "dataxform Examples and Full Command Syntax" on page 58.
- **Ignore holes** — Alternatively, a protected host administrator can configure `dataxform` to ignore holes. The utility decrypts, re-encrypts, and rewrites all file blocks. Any file blocks that previously corresponded to holes have storage allocated for them, and thus, after transformation, files are no longer sparse. See the `--encrypt_sparse_file_holes` option in the "dataxform Examples and Full Command Syntax" on page 58.

In most cases, it is advantageous for the protected host administrator to configure `dataxform` to recognize holes, so that sparse files remain sparse. Failure to do so can result in longer than expected transformation times and post-transformation file sets that consume significantly more storage space than prior to transformation.

dataxform and Duplicate Elimination

A similar phenomenon can occur with files that are deduplicated or compressed by the under-lying file system or volume manager. If deduplication or compression boundaries do not align with the file blocks that are the units in which CTE encrypts data, the size of a transformed file set may be greater or less than that of the same file set prior to transformation.

Dataxform and Linked Files

When using `dataxform`, protected host administrators must be cognizant of the utility's treatment of linked files—files for which two or more directory entries point to a single data image. In general, the utility encrypts and rekeys linked files correctly, but the relationship of links to GuardPoints means that administrators must be aware of links and how the utility handles them prior to transformation. A link may be *hard*—it may be a directory entry that points to a file inode to which one or more other directory entries in the same file system also point. A link may be *soft or symbolic*—it may represent a file whose data consists of the path name of another file in the same or a different file system.

The `dataxform` utility can detect that a directory entry is a hard or soft link. It transforms any file with multiple hard links to it when it first encounters any of the links to the file. Thereafter, it skips the already-transformed file, and creates a skipped log entry. This is not an error, but an indication that `dataxform` recognizes that it has already transformed the file's data. Soft links are simply skipped.

There are two situations with linked files in which data corruption can potentially result:

- **External links** — Links in directories outside a GuardPoint, that point to files in directories within the GuardPoint.
- **Links to external files** — Links in directories, protected by a GuardPoint, that points to files in directories outside of the GuardPoint.

In these cases, the CTE Agent does not have complete control over application access to file data. For example, in a GuardPoint that protects a directory sub-tree (rather than a mount point), if a hard link in a directory outside the GuardPoint points to a file within the GuardPoint, the CTE Agent does not see every access point to the file's data. If the file is opened through the external link and data is written to it, the GuardPoint does not intercept the writes, and no encryption occurs. If the file is later opened through the protected path, and the data written from outside is read, the CTE Agent decrypts it, even though it was never encrypted. This situation does not occur with GuardPoints that protect entire file systems, because hard links can only refer to file data within the same file system name space as the files to which they point.

This problem does not occur with an external soft link, because a file opened through a soft link in a directory outside the GuardPoint is ultimately opened through its actual path, which lies within a protected directory.

Similarly, if a hard link in a protected directory refers to file data outside the GuardPoint, the `dataxform` kernel component opens it and transforms the data in it. If the file is subsequently opened through a path outside the GuardPoint, data is not decrypted as it is read, and therefore appears corrupt to applications. If applications access the file from outside the GuardPoint and write data to it, the GuardPoint intercepts subsequent reads through the link, and the CTE Agent decrypts data that was never encrypted.

To assist protected host administrators in dealing with linked files, the `dataxform` utility includes a facility for listing the hard linked files within a GuardPoint. Administrators can analyze these lists to determine whether the links cross GuardPoint boundaries and therefore represent potential for operational errors and data corruption.

See "Checking for Hard-Link Files Inside the GuardPoint with dataxform" on page 44 for more information.

Automatic Data Transformation

Using `dataxform` for initial encryption or rekeying of data is a two-party procedure requiring cooperation between the key manager Security Administrator and administrators of protected hosts. The key manager Security Administrator creates policies and applies them to GuardPoints before and after transformation. The protected host administrators disable access to protected file sets, run the `dataxform` utility, and re-enable file access after transformation. The two-party architecture preserves security by making it impossible for a single individual to subvert data protection.

In small data centers, key manager Security Administrators and protected host administrators typically work closely together and have an understanding of each others' priorities and constraints. In larger organizations, organizational and physical distances between them often exist. Moreover, a key manager cluster often manages data security and key management for dozens, or hundreds, of protected hosts.

Simplifying dataxform Data Transformation

CTE can be configured to partially automate data transformation with `dataxform`, reducing the need for administrator coordination. The administrator of a protected host enables automatic transformation of a protected data set by creating a file named `dataxform_auto_config` in the GuardPoint's root directory. This file contains information used to verify version compatibility with the CTE Agent, as well as some parameters to be input to `dataxform` (for example, the location of the disk space to be used to construct the utility's file list).

If a `dataxform_auto_config` file is present when the key manager Security Administrator activates a **dataxform** policy (one that contains both production and transformation keys), the CTE Agent in the protected system automatically starts `dataxform`. Conversely, the protected host administrator can disable automatic transformation by deleting the `dataxform_auto_config` file from a GuardPoint's root directory.

When `dataxform` execution completes (or aborts), it leaves behind status files that it uses to regulate subsequent executions. Whenever the `dataxform` starts, it looks for these files, and if it finds them, displays an informative message and exits without transforming any files. This prevents `dataxform` from running repeatedly. Prior to running `dataxform`, a protected host administrator must execute the utility's cleanup function to eliminate status files from previous runs (see "Cleaning Up a Previous dataxform Session" on page 43). If a transformation fails, the protected host administrator must repair the problem, complete the transformation, and *then* execute the cleanup function (see "Recovering a Failed or Incomplete dataxform Session" on page 51).

Even with automatic data transformation, the key manager Security Administrator must monitor `dataxform` progress (for example, by observing the audit log), and replace the GuardPoint's `dataxform` policy with a post-transformation production policy when the run completes. Protected host administrators remain responsible for blocking access to data (for example, stopping databases and applications or unmounting file systems) so applications do not have access to files. Finally, protected host administrators are responsible for re-enabling application access to files after transformation is complete and the key manager Security Administrator has replaced the `dataxform` policy with a post-transformation production policy.

To summarize, the key manager Security Administrator and protected host administrator interact during automatic data transformation as follows:

- **Enable automation** (protected host administrator)

  To enable automation, the protected host administrator creates a `dataxform_auto_config` file in the root directory protected by the GuardPoint. This is a one-time action. The `dataxform_auto_config` file needs to only be updated when parameters change, or deleted when the administrator wishes to disable automation.

- **Clean up from previous transformation** (protected host administrator)

  The protected host administrator executes the `dataxform` cleanup function (`--cleanup`) to enable transformation to begin automatically when a transformation policy is activated for the GuardPoint.

- **Disable access to data** (protected host administrator)

  The protected host administrator disables access to data, and informs the key manager Security Administrator that it is safe to replace the GuardPoint's pre-transformation production policy with a `dataxform` policy. The time between disabling access and activation of the `dataxform` policy is part of the overall window of data unavailability.

- **Monitor dataxform progress** (key manager Security Administrator)

  The key manager Security Administrator monitors the progress of the utility, and when the run is complete, replaces the `dataxform` policy with a new post-transformation production policy. Once the post-transformation production policy has been activated, the key manager Security Administrator notifies the protected host administrator that it is safe to re-enable application access to the protected file set. The time between completion of the `dataxform` run and re-enabling of data access is part of the overall window of file unavailability. See "Monitoring dataxform" on page 45 for operational details.

Partial automation of data transformation reduces the number of interactions between protected host and key manager Security Administrators. Expect that, over time, CTE will evolve to reduce the interactions to those required to maintain the fundamental security precepts of the software.

See "Running Automatic Data Transformation" on page 42 for detailed operational examples.

Best Practices When Using dataxform

The `dataxform` utility transforms files in place, overwriting file blocks as it transforms them. In-place transformation is advantageous, because it minimizes the temporary storage required to transform large file sets. However, if `dataxform` fails partway through, files being transformed at the moment of failure must be restored from backups after the run is complete and access to the data set has been re-enabled. This implies that a reliable, up-to-date backup copy of a protected file set is a necessary prerequisite to in-place transformation.

If a large file set must be fully backed up prior to running `dataxform`, backup time must be added to the estimated transformation time to calculate the window of unavailability. (Backup and transformation must be consecutive so that files do not change between the two.) For large data sets, backup time can be substantial. In no case, however, can one recommend bypassing the backup step. If files are worth protecting with CTE, they are presumably of significant value to the enterprise. Therefore, every reasonable effort must be made to protect them against loss as well as theft.

Summary

Periodic rekeying of encrypted data is increasingly becoming a regulatory or policy necessity, particularly in the health care and financial fields. In addition, a sometimes-overlooked problem in deploying online data encryption is the initial encryption of legacy data sets. There are two basic techniques for both initial encrypting and rekeying:

- Copy data from its source to a destination protected by the desired encryption policy.
- Encrypt or rekey data in place, overwriting it block by block.

As data center operations grow more complex, and as file sets grow larger, copying becomes a less viable option. Running `dataxform` requires cooperation between the key manager Security Administrator and administrators of protected hosts. File sets must remain offline for the duration of a `dataxform` run, so the expected outage window must be estimated, and everything possible done beforehand to ensure that the run will succeed. The utility includes facilities for estimating run time, and for discovering potential problems, such as linked files, prior to running the software.

Chapter 2: Initial Data Encryption

After installing and configuring CTE, one of the most common procedures is to encrypt the data you want to protect. This is called *initial data encryption*. There are two methods for encrypting data with CTE. The first is by copying or restoring your clear data into a GuardPoint with a production/standard encryption policy. The second is by using the `dataxform` utility to encrypt the data in place. This chapter describes both of these methods in the following sections:

Data Encryption Overview 24
Using the Copy or Restore Encryption Method on File Systems 27
Using the Copy or Restore Encryption Method on Block Devices 28
Using dataxform to Encrypt Your Data with the DSM 30

> **Note**
> Before using the procedures in this chapter read Chapter 1: "Data Transformation Overview" on page 7.

Data Encryption Overview

The first step in the data encryption process is to determine the optimal encryption method for your environment. There are three methods:

- **Copy**. Clear data is encrypted by copying it into a GuardPoint with an encryption policy.
- **Restore**. Clear data stored on a backup device is encrypted by restoring it to a GuardPoint with an encryption policy.
- **dataxform**. Data is encrypted in-place using the `dataxform` command line utility.

The optimal method depends on the following:

1. Whether you are encrypting data on a block device or directory.
2. The amount of disk space you have.
3. Speed of your backup devices.

> **Note**
> Whichever method you select, backup your data before encrypting it.

How to Decide What Method to Use

Some general rules:

- If the data you are encrypting is in a block device, you must use the Copy or Restore encryption method.
- If you can copy the data into an encrypting GuardPoint on the same disk and same volume, the Copy method is as fast as the `dataxform` method.
- If you can copy the data into an encrypting GuardPoint on the same disk, but different volume, or SAN or NAS, you can use the Copy method but it is slightly slower.
- The Copy method also requires disk space that is twice the size of the data you are encrypting.
- If the data you are encrypting comes from a backup device that is not a disk, for example, a storage array, you can use the Restore method.

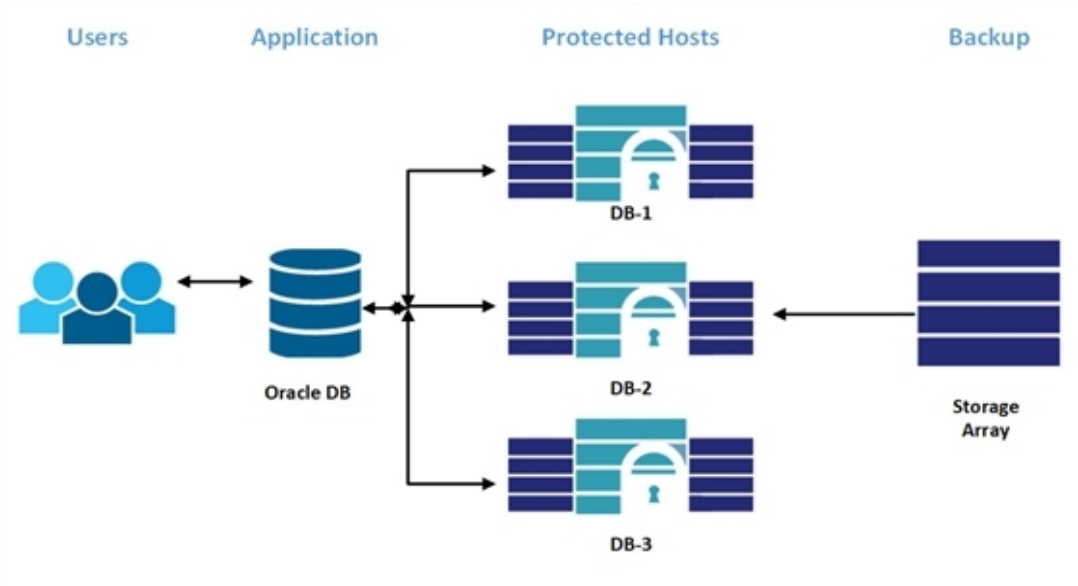
Restore Encryption Method

In this method, your sensitive data is backed up to a storage device. To encrypt the data:

1. Block access to the directory or device that will be encrypted.
2. Create a GuardPoint with an encryption policy on the directory or block device that will hold the protected data.
3. Restore the data from the backup device to the GuardPoint. As data is written into the GuardPoint, it is encrypted.
4. Replace the encryption policy with a production policy or the initial test policy.
5. Open access to the now-protected directory or block device.

In the following example, users access Oracle databases on the protected host.

Figure 2-1: Users Accessing a restored Oracle DB



To encrypt `\DB-3`:

1. Block user access to it.
2. Create an encryption GuardPoint on `\DB-3.....`
3. Restore the backup data from the backup media to `\DB-3`.
4. Restore access to the directory.

This method requires no extra disk space. The speed of this method depends on the speed of the restoration device.

Copy Encryption Method

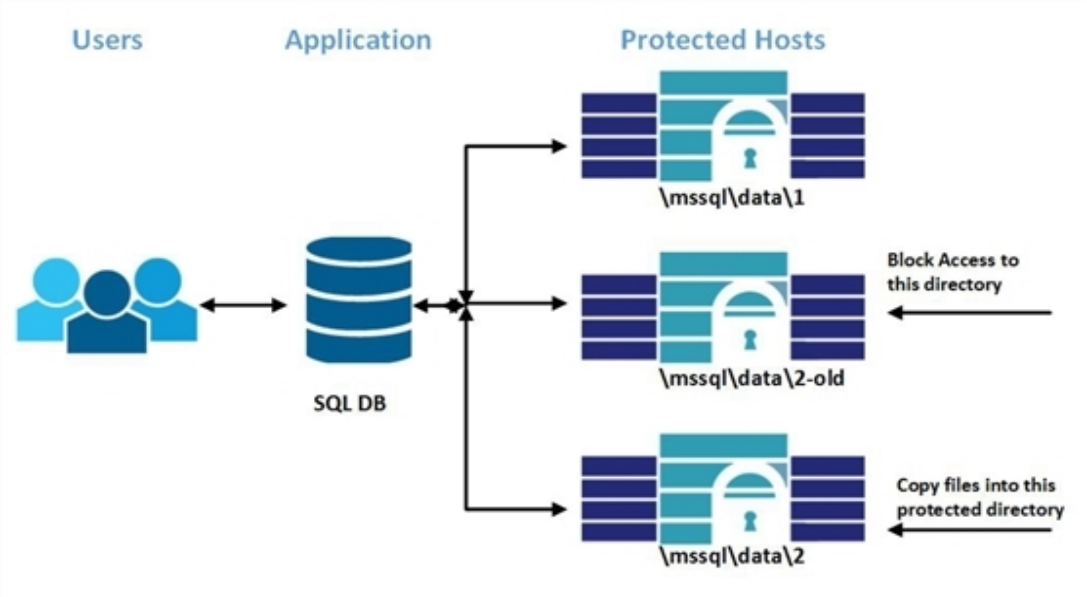
In this method, you encrypt clear data by copying it into a GuardPoint with an encryption policy. This method is generally faster than the restore encryption method. If the data you copy to the GuardPoint is on the same drive and volume as the GuardPoint, this method is comparable in speed to `dataxform`, which is about 2-4 GB per minute. If the data to be encrypted is accessed from a slower disk or a different volume, the encryption will be slightly slower.

The following is an example of the Copy Encryption process:

1. Block all access to the directory containing the data to be encrypted.
2. Rename that directory (example: from `\mssql\data\2` to `\mssql\data\2-OLD`).
3. Create a new directory for your sensitive data with the original directory path (`\mssql\data\2`) and block access to it.
4. Create a GuardPoint with an encryption policy on that directory.
5. Copy the sensitive data into the GuardPoint. Data in the GuardPoint is encrypted.
6. Replace the encryption policy with a production policy or the initial test policy.
7. Allow access to the new directory.

This method requires additional disk space at least as large as `\mssql\data\2-OLD`. The speed of the method depends on the speed of the copy.

Figure 2-2: Users using the copy method



dataxform Encryption Method

In this method, you encrypt data in place using the `dataxform` tool. In general, this method is the fastest. To generate an estimated time for encryption see "Estimating the dataxform Runtime Period" on page 40.

The following is an overview of the `dataxform` method:

1. Block all access to the directory containing the sensitive data.
2. Create a `dataxform` policy for the GuardPoint on this directory.
3. Run `dataxform` on the directory. After completion, the data in the GuardPoint is encrypted.
4. Remove the `dataxform` policy on the GuardPoint and replace it with a production policy.
5. Open access to the directory.

Using the Copy or Restore Encryption Method on File Systems

WARNING

If you apply an encryption GuardPoint to a folder containing files, those files remain unencrypted. If you try to access those files, they are then encrypted. If you attempt a write, you can potentially corrupt parts or the entire file. The only method to access those files in an unencrypted state is to disable or remove the GuardPoint.

Prerequisites

- Verify that there is a good backup of the data to be encrypted. This step is vital.
- You will need to stop ALL access and services to the data to be encrypted during part of this procedure, and access will NOT be restored until all of the data has been copied to the new location and the encryption process is complete. Make sure that you plan for this outage and that users know the data will be inaccessible for some time.
- Make sure you have enough empty storage space to copy the data.
- Make sure that you have a CTE production policy with the proper security rules defined for the new GuardPoint. The production policy needs to use the same encryption key that you will be using to initially encrypt the data.

Procedure

1. Log on to your key manager and, if necessary, switch to the domain containing the host you want to protect.
2. Identify the encryption key you want to use to encrypt the data or create a new encryption key.
3. Create an initial encryption policy for the GuardPoint. This policy will only be used to encrypt the data as it is copied into the new directory, after which you will replace the policy with a production policy.
   a. For **Policy Type**, select Standard.
   b. Add a security rule with:
      - **Action:** all_ops
      - **Effect:** Apply Key, Permit
   c. Add a Key Rule that specifies the encryption key you want to use. This key must match the one specified in the production policy you intend to apply to the GuardPoint after the data has been encrypted.
4. Create an empty directory for the protected data if necessary.
5. Create a GuardPoint on the empty directory.
   - For **Policy**, select the initial encryption policy you want to use.
   - In the **Type** field, select either **Directory (Auto Guard)** or **Directory (Manual Guard)**.

     If you select **Auto Guard**, CTE starts the guard process as soon as the policy is pushed to the host. You enable, disable, guard, and unguard the GuardPoint in your key manager.

     If you select **Manual Guard**, You guard the GuardPoint on the protected host with the `secfsd -guard <path>` command and unguard it with the `secfsd -unguard <path>` command. This gives you more control over when data transformations occur because CTE will not start encrypting or rekeying the device until you manually start the process.
   - For **Path**, enter the path to the directory or click Browse. For example, `/dev/sda1/data/HR` or `E:/data/HR`.

6. Stop ALL access and services to the data to be encrypted. Make sure no processes, services, or users are currently accessing the data. You cannot restore access to the data until the encryption process is complete.
7. Copy or restore the data from the old operational directory to the newly created GuardPoint. CTE encrypts the data as it is added to the GuardPoint.

   Make sure you wait until the encryption process has finished for all data in the GuardPoint before you continue with this procedure.
8. Disable the encryption policy from the new GuardPoint and apply your production policy. Your data is now fully encrypted and you can redirect access to the GuardPoint.
9. Start application testing and inform application teams that systems are ready for use. Everything should work exactly as before except that now the data is encrypted. Monitor the situation with your users.

Using the Copy or Restore Encryption Method on Block Devices

The process for using the Copy or Restore encryption method on block devices and raw disks is much the same as with file systems.

Information for Encrypting Block Devices

- The Oracle DBA defines a disk group with `secvm` disks/devices. Then `secvm` communicates with the DBA and updates the disk group information. Oracle ASM provides the mapping of `secvm` devices to physical disks/devices.
- All databases (table space) in a disk group must be encrypted. Do not mix encrypted and non-encrypted databases in a single disk group. Non-encrypted databases must be kept in separate disk groups that are not protected by a CTE Agent.
- If you want to apply GuardPoints globally to a set of hosts, configure the GuardPoints in a host/client group. This ensures that the same GuardPoints with the same policies are applied to all members of the host/client group. If you plan to configure the host in a host/client group, do not configure GuardPoints at the host level.
- Partitions are identified by their device name. Device names for partitions vary between platforms.

Prerequisites

- Verify that there is a good backup of the data to be encrypted. This step is vital.
- The block device receiving the protected data must be new or clean as all existing data will be unusable.
- You will need to stop ALL access and services to the data to be encrypted during part of this procedure, and access will NOT be restored until all of the data has been copied to the new location and the encryption process is complete. Make sure that you plan for this outage and that users know the data will be inaccessible for some time.
- Make sure that you have a CTE production policy with the proper security rules defined for the new GuardPoint. The production policy needs to use the same encryption key that you will be using to initially encrypt the data.

Procedure

1. Make sure that the block devices into which you plan to copy the data are empty and have enough storage space to contain the existing data.

   CAUTION
   Any existing data in the target devices will become unusable after this procedure is complete. Make sure that the target devices do not contain any data before you continue with this procedure.

2. Log on to your key manager and, if necessary, switch to the domain containing the host you want to protect.
3. Identify the encryption key you want to use to encrypt the data or create a new encryption key.
4. Create one or more initial encryption policies for the GuardPoints. These policies will only be used to encrypt the data as it is copied into the new block devices, after which you will replace the initial policies with production policies.
   a. For **Policy Type**, select Standard.
   b. Add a security rule with:
      - **Action:** all_ops
      - **Effect:** Apply Key, Permit
   c. Add a Key Rule that specifies the encryption key you want to use. This key must match the one specified in the production policy you intend to apply to the GuardPoint after the data has been encrypted.
5. Create a GuardPoint for each block device you want to encrypt. For each GuardPoint:
   - For **Policy**, select the initial encryption policy you want to use.
   - In the **Type** field, select either **Raw or Block Device (Auto Guard)** or **Raw or Block Device (Manual Guard)**.

     If you select **Auto Guard**, CTE starts the guard process as soon as the policy is pushed to the host. You enable, disable, guard, and unguard the GuardPoint in your key manager.

     If you select **Manual Guard**, You guard the GuardPoint on the protected host with the `secfsd -guard <path>` command and unguard it with the `secfsd -unguard <path>` command. At system startup, you must guard the device and then mount it. This gives you more control over when data transformations occur because CTE will not start encrypting or rekeying the device until you manually start the process.
   - For **Path**, enter the path to the device or click Browse. For example, `/dev/sda1` or `E:/`.

     If you click Browse, the selection dialog box shows the available block devices. Inactive partitions are displayed, but open partitions and currently guarded partitions are *not* displayed.

     > **Note:** Third-party applications can open raw devices in obscure ways, and may cause the Remote File Browser to ignore and not display supposedly inactive devices. For example, inactive raw devices in the Oracle DBCA disk discovery path are not displayed in the Remote File Browser, even when the devices are not assigned to a disk group. If `/dev/sd*` is configured in the DBCA disk discovery path and DBCA is running, inactive `/dev/sd*` devices are not displayed in the browser. This is because the devices are kept open by the Oracle process. To get around this problem, close DBCA and open the browser again. The devices are free, displayed in the browser, and available for selection.
6. Click **OK** to apply the policy to the GuardPoint.

   Wait until the policy has been applied to the device. This may take up to a minute.

7. Stop ALL access and services to the devices to be encrypted. Make sure no processes, services, or users are currently accessing the data. You cannot restore access to the data until the encryption process is complete.
8. Copy or restore the data from the old operational directory to the newly created GuardPoint. CTE encrypts the data as it is added to the GuardPoint.

   Make sure you wait until the encryption process has finished for all data in the GuardPoint before you continue with this procedure.
9. Disable the encryption policy from the new GuardPoint and apply your production policy. Your data is now fully encrypted and you can redirect access to the GuardPoint.
10. Start all services and restore access to the data that is now fully encrypted.

    Make sure that the applications and services access the newly created CTE-protected hosts. CTE encrypted raw or block protected hosts are accessed using the directory: `/dev/secvm/dev/xxxxx` where *xxxxx* is the original device name.
11. Start application testing and inform application teams that systems are ready for use. Everything should work exactly as before; however, monitor the situation with your users.

Using dataxform to Encrypt Your Data with the DSM

The `dataxform` utility is an executable that encrypts data-in-place in a GuardPoint. This section describes how to use `dataxform` for initial data encryption. For more details on `dataxform` and its capabilities see "Overview of the dataxform Utility" on page 16 and Appendix A: "DataXform Reference" on page 39.

> **Note**
> `dataxform` does not work on block devices.

The following procedure explains how to use dataxform if you are using the Vormetric Data Security Manager (DSM) as your key manager.

Notes and Limitations

- For detailed `dataxform` information, see "Overview of the dataxform Utility" on page 16 and "dataxform Examples and Full Command Syntax" on page 58.
- Guarding and transforming linked files is potentially dangerous. See "Checking for Hard-Link Files Inside the GuardPoint with dataxform" on page 44.
- Performance may be impacted (slow down) if you run multiple transforms concurrently on systems where the GuardPoints are on the same disk. On systems with a large number of CPUs running concurrently with default settings, even if the GuardPoints are on different disks, you may experience slower performance. If you execute multiple instances of `dataxform` manually, the instances run in parallel. If you execute multiple instances of `dataxform` automatically (see "Automatic Data Transformation" on page 21 and "Running Automatic Data Transformation" on page 42), the instances run consecutively.
- It is recommended that you run one `dataxform` session, for better performance.

Prerequisites

- Verify that there is a good backup of the data to be encrypted. This step is vital.
- Make sure that you have a CTE production policy with the proper security rules defined for the new GuardPoint. The production policy needs to use the same encryption key that you will be using to initially encrypt the data.

- You will need to stop ALL access and services to the data to be encrypted during part of this procedure, and access will NOT be restored until the encryption process is complete. Make sure that you plan for this outage and that users know the data will be inaccessible for some time. To estimate the outage duration, see "Estimating the dataxform Runtime Period" on page 40.

  In addition, make sure you know which `dataxform` options you want to use so that you are ready to run the command as soon as the GuardPoint is ready. This will keep the required downtime as short as possible. For a complete list of options, see "dataxform Examples and Full Command Syntax" on page 58.

- If you use Microsoft Volume Shadow Services (VSS) files and use `dataxform` to change the encryption keys, then you must make a VSS shadow copy before and after running `dataxform`. See "Automatic Data Transformation" on page 21.

Procedure

1. Log into the DSM Management Console and switch to the domain containing the host you want to protect.
2. Identify the encryption key you want to use to encrypt the data or create a new encryption key.
3. Create an initial encryption policy for the GuardPoint. This policy will only be used to encrypt the data as it is copied into the new directory, after which you will replace the policy with a production policy.
   a. In the top menu bar, click **Policies**.
   b. Above the policy table, click **Add**.
   c. For **Policy Type**, select Standard.
   d. For **Name**, make sure you use a name that clearly designates this as an initial-encryption policy and not a production policy. You will need to be able to find this policy name from the list of all available policies when you create the GuardPoint.
   e. Add a security rule with:
      - **Action:** key_op
      - **Effect:** Apply Key, Permit
   f. Add a Key Rule that specifies `clear_key` as the encryption key you want to use.
   g. Add a Data Transformation Rule that specifies the encryption key you want to use to encrypt the data. This key must match the one specified in the production policy you intend to apply to the GuardPoint after the data has been encrypted.
   h. Click **OK** to save the policy.
4. Stop ALL access and services to the data to be encrypted. Make sure no processes, services, or users are currently accessing the data. You cannot restore access to the data until the encryption process is complete.
5. Create a GuardPoint using the initial encryption policy. When you are creating the GuardPoint, for **Type**, select **Directory (Auto Guard)** or **Directory (Manual Guard)**. You cannot use `dataxform` with a block device.

   If you select **Auto Guard**, CTE starts the guard process as soon as the policy is pushed to the host. You enable, disable, guard, and unguard the GuardPoint in the key manager.

   If you select **Manual Guard**, You guard the GuardPoint on the protected host with the `secfsd -guard <path>` command and unguard it with the `secfsd -unguard <path>` command. This gives you more control over when data transformations occur because CTE will not start encrypting or rekeying the device until you manually start the process.

   Select **Directory (Manual Guard)** for Linux/AIX file system directories that must be manually guarded and unguarded in order to failover to a different node in a cluster. See "Automatic and Manual GuardPoints" on page 56 for details on this issue.

6. Click **OK** to apply the policy to the GuardPoint.

   Wait until the policy has been applied to the device. This may take up to a minute.

7. If you selected Auto Guard, disable the GuardPoint.

8. Log onto the host system and manually encrypt the data in the GuardPoint.

   a. Run `dataxform` with the desired options. For example, if you want to encrypt the GuardPoint `/opt/apps/dx2` GuardPoint, display the time taken for `dataxform` to complete each phase of the transformation process, and have `dataxform` preserve the last modified dates on the files, you would enter:

   ```
   # dataxform --rekey --print_stat --preserve_modified_time --gp /opt/apps/dx2

   Checking if data transform is supported for guardpoint /opt/apps/dx2
   Data transformation is supported on /opt/apps/dx2
   About to perform the requested data transform operation
   -- Be sure to back up your data
   -- Do not access files in the guard point during the transform process
   -- Please do not attempt to terminate the application

   Scan found 10005 files (273 KB) in 5 directories for guard point /opt/apps/dx2
   The current operation took 0 hours, 0 minutes and 1 seconds
   Transformed 10010 files (273 KB) of 10005 files (273 KB) for guard point
   /opt/apps/dx2
   The current operation took 0 hours, 0 minutes and 25 seconds
   Data transform skipped some files
   The file /opt/apps/dx2/hardlinkedfileLocal01 was skipped. It was an additional
   hard link
   The file /opt/apps/dx2/filenothere03 was skipped. It was a soft link
   The file /opt/apps/dx2/filenothere02 was skipped. It was a soft link
   The file /opt/apps/dx2/filenothere01 was skipped. It was a soft link
   The file /opt/apps/dx2/hardlinkedfileLocal02 was skipped. It was an additional
   hard link
   Number of additional hard links skipped: 2
   Number of soft links skipped: 3
   Missing 1 references to hard link /opt/apps/dx2/hardlinkfile01
   Missing 1 references to hard link /opt/apps/dx2/hardlinkfile02
   Missing 1 references to hard link /opt/apps/dx2/hardlinkfile03
   The data transform operation took 0 hours, 0 minutes and 25 seconds
   Data transform for guard point /opt/apps/dx2 finished but 5 files were skipped
   ```

   - View the `dataxform` run results in the local log, `/var/log/vormetric/vordxf _path_usr.log`, or in the **Logs** window.
   - View the list of files that were not transformed in `/var/log/vormetric/dataxform_status_skip-_path.log`.

   > **Note:** Low-power systems can run out of memory while running `dataxform`. If entries like "`[VMD] [ERROR] [1933564] [DXF4328E] Kernel component gave unexpected status 4.`" and "`[VMD] [ERROR] [3670108] [DXF4300E] Out of Memory`" are sent to the system messages file, lower the `--thd` parameter value. It takes longer to run, but `dataxform` uses less memory and completes successfully.

   b. Read the `dataxform` command line messages as it encrypts files. Specifically note messages that list files or folders that are skipped and the reasons why. The `dataxform` log file also contains this information. Use it to identify failed transformations (see "Monitoring dataxform" on page 45.

c. If a `dataxform` fails during transformation, you can usually rerun it and it resumes transformation beginning with the next file. The risk is that all files that were in-progress during the transformation may not be completely transformed at the point of failure. In this case you will have to restore it from your backup and transform it again.

> **Note:** If `dataxform` fails, do not clean up the GuardPoint or remove the `dataxform` status files. If you run `dataxform` again in the same GuardPoint, `dataxform` will use these files and resume operations where it left off.

d. After you have verified that the encryption is successful, clean up the `dataxform` session files before you run `dataxform` again on the same GuardPoint. See "Cleaning Up a Previous dataxform Session" on page 43.

9. After the dataxform session has been cleaned up, go back to the DSM Management Console and remove the GuardPoint.
10. Create a new GuardPoint using the desired production policy with the appropriate access control rules. Your data is now fully encrypted and you can redirect access to the GuardPoint.
11. Start all services and restore access to the now-encrypted data.
12. Inform application teams that systems are ready for use. Everything should work exactly as before; however, monitor the situation with your users.

Chapter 3: Rekeying

This chapter describes how to rekey your existing GuardPoints. It consists of the following sections:

Rekeying Overview 34
Rekeying with dataxform 34
Rekeying Using the Manual Copy Method 37

Rekeying Overview

Data encryption keys are the keys used to encrypt data in a GuardPoint. *Rekeying*, also called *key rotation*, is the process of changing the encryption key used to encrypt your GuardPoint data. Changing GuardPoint encryption keys increases security and is required in some organizations. Best practices covered in the National Institute of Standards and Technology (NIST) Special Publication 800-57 dictate that encryption keys should be rotated periodically to ensure the security of data from compromise. This security, however, comes at a cost – namely, downtime to perform the re-encryption of the data.

There are two methods for rekeying:

- **Using the dataxform utility** — Data is encrypted in-place using the `dataxform` utility. This method is fast and easy, but requires total and exclusive access to the GuardPoint. All users and applications are blocked from accessing the data until `dataxform` is finished executing. See "Rekeying with dataxform" below.
- **Manual copying** — A GuardPoint is created on a new directory or device and guarded using a new encryption key. Then the data is copied from the original GuardPoint to the new GuardPoint. During the copy, data is decrypted with the original key and encrypted with the new key when it is placed in the new GuardPoint. This method does not require exclusive access to the original GuardPoint, but requires extra storage space of at least the amount of data to be copied. It also requires more steps to ensure that the newly re-encrypted data is both protected by a policy using the new key and known to users or applications that require access to the data. See "Rekeying Using the Manual Copy Method" on page 37.

Rekeying with dataxform

Rekeying with `dataxform` requires that you change the current key of the production policy on your GuardPoint to a new key.

Notes and Limitations

- For detailed `dataxform` information, see "Overview of the dataxform Utility" on page 16 and "dataxform Examples and Full Command Syntax" on page 58.
- Guarding and transforming linked files is potentially dangerous. See "Checking for Hard-Link Files Inside the GuardPoint with dataxform" on page 44.
- Be aware that running more than one `dataxform` session per underlying disk may affect performance, and cause things to slow down. You can run multiple `dataxform` sessions provided they are going to different disks. If you run multiple `dataxform` sessions against the same disk, there will likely be performance impact.

- If you execute multiple instances of `dataxform` manually, the instances run in parallel. If you execute multiple instances of `dataxform` automatically (see "Automatic Data Transformation" on page 21 and "Running Automatic Data Transformation" on page 42), the instances run consecutively.

  > **Note:** `dataxform` is optimized to run as fast as possible, tuning itself to the computer automatically. To run more than one instance at once, you may need to reduce to number of execution threads allocated to each instance. See "dataxform Examples and Full Command Syntax" on page 58.

- If you change the encryption key on particular production policy, and if another GuardPoint on another host uses the same production policy, then that GuardPoint's data will be unreadable because it still uses the old key. To avoid this:
  - Transform all of the data in all of the GuardPoints on all of the Hosts that use the same production policy with the new key
  - Change the name of the production policy used on the GuardPoint on which you ran `dataxform`. The policy will then only apply to that GuardPoint and not the other GuardPoints using the policy of the original name.

Procedure

1. If `dataxform` was run on this GuardPoint previously, you must clean up those `dataxform` sessions. See "Cleaning Up a Previous dataxform Session" on page 43.
2. Log into your key manager management console.
3. Create a new key or identify an existing key that you want to use for re-encrypting the data.
4. Create a `dataxform` policy that specifies:
   - **Policy Type**: Standard.
   - **Name**: Something unique that you will be able to recognize in the list of available policies when you go to create the GuardPoint.
   - A Security Rule with **Action**: key_op and **Effect**: apply_key, permit
   - A Key Selection Rule that specifies the original key currently in use.
   - A Data Transformation Rule that specifies the new key you want to use.
5. Block all access to data in the GuardPoint that is to be re-encrypted.
6. Disable the production policy on the GuardPoint.
7. Add a new GuardPoint to the host with the same directory as the original GuardPoint, applying the `dataxform` policy to the GuardPoint.

   > **Note:** At this point, all access to the GuardPoint is denied except for `dataxform`.

8. From the command line on the protected host, run `dataxform` as `root` or admin user with at least the `--rekey` and `--gp` options.

   For example:

```
# dataxform --rekey --print_stat --preserve_modified_time --gp /opt/apps/dx2
Checking if data transform is supported for guard point /opt/apps/dx2
Data transformation is supported on /opt/apps/dx2
About to perform the requested data transform operation
-- Be sure to back up your data
-- Do not access files in the guard point during the transform process
-- Please do not attempt to terminate the application

Scan found 10005 files (273 KB) in 5 directories for guard point /opt/apps/dx2
The current operation took 0 hours, 0 minutes and 1 seconds
Transformed 10010 files (273 KB) of 10005 files (273 KB) for guard point
/opt/apps/dx2
The current operation took 0 hours, 0 minutes and 25 seconds
Data transform skipped some files
The file /opt/apps/dx2/hardlinkedfileLocal01 was skipped. It was an additional hard
link
The file /opt/apps/dx2/filenothere03 was skipped. It was a soft link
The file /opt/apps/dx2/filenothere02 was skipped. It was a soft link
The file /opt/apps/dx2/filenothere01 was skipped. It was a soft link
The file /opt/apps/dx2/hardlinkedfileLocal02 was skipped. It was an additional hard
link
Number of additional hard links skipped: 2
Number of soft links skipped: 3
Missing 1 references to hard link /opt/apps/dx2/hardlinkfile01
Missing 1 references to hard link /opt/apps/dx2/hardlinkfile02
```

```
Missing 1 references to hard link /opt/apps/dx2/hardlinkfile03
The data transform operation took 0 hours, 0 minutes and 25 seconds
Data transform for guard point /opt/apps/dx2 finished but 5 files were skipped
```

- View the `dataxform` run results in the local log, `/var/log/vormetric/vordxf_path_usr.log`, or in the **Logs** window.
- View the list of files that were not transformed in `/var/log/vormetric/dataxform_status_skip_path.log`.

> **Notes**
> - If `dataxform` fails, do not clean up the GuardPoint or remove the `dataxform` status files. If you run `dataxform` again in the same GuardPoint, `dataxform` will use these files to resume operation where it had left off.
> - Low-power systems can run out of memory while running `dataxform`. If entries like "`[VMD] [ERROR] [1933564] [DXF4328E] Kernel component gave unexpected status 4.`" and "`[VMD] [ERROR] [3670108] [DXF4300E] Out of Memory`" are sent to the system messages file, lower the `--thd` parameter value. It will take longer to run, but `dataxform` will use less memory and complete successfully.

9. Go back to your key manager management console and, in the production policy, change the key to use the new key instead of the original key.
10. Disable the GuardPoint that you created using the `dataxform` policy and re-enable the production policy that now has the new key.
11. Verify proper access to the data.

Rekeying Using the Manual Copy Method

If you change the encryption key on a production policy, and if another GuardPoint on another host uses the same production policy, then that GuardPoint's data becomes unreadable because the data is still encrypted/decrypted with the old key. To avoid this:

- Transform all of the data in all of the GuardPoints on all of the Hosts that use the same production policy with the new key
- Change the name of the production policy used on the GuardPoint on which you ran `dataxform`. The policy will then only apply to that GuardPoint and not the other GuardPoints using the policy of the original name.

Procedure

1. Identify a location where CTE can create a GuardPoint and has enough space to hold the data for rekeying.
2. Log into your key manager's management console.
3. Create a new key or identify an existing key that you want to use for re-encrypting the data.
4. Create a Standard production policy that specifies the following:
   - **Policy Type**: Standard.
   - **Name**: Something unique that you will be able to recognize in the list of available policies when you go to create the GuardPoint.
   - A Security Rule with **Action**: all_ops and **Effect**: apply_key, permit.
   - Any other security rules you need in your production policy.
   - A Key Selection Rule that specifies the new key you want to use.
5. Add a new GuardPoint that uses the new production policy on the protected host location you identified.

6. Copy or move the data from the original directory to the new GuardPoint.

   The data in the new directory is rekeyed as it is copied in.

7. Wait until all files have been copied and the rekey operation has completed on the new GuardPoint.
8. Direct all applications and users to use the new data location, or change the name of the new GuardPoint directory to the name of the original directory.
   - If you direct all applications and users to use the new data location, make sure that they use the production policy that contains the new encryption key.
   - If you change the name of the new GuardPoint directory to the name of the original directory use the following instructions:
     1. In the key manager management console, disable the original GuardPoint.
     2. Unguard the newly created GuardPoint with the rekeyed data.
     3. On the protected host, rename the original directory to a temporary name.
     4. Rename the newly created directory to the original directory name.
     5. Modify the original policy to use the newly created key.
     6. Enable the original GuardPoint with the original policy. This puts the original policy in effect with the new key on the newly transformed data.
9. Verify that the rekeyed data is accessible to users.

Appendix A: DataXform Reference

This appendix consists of the following sections:

Common dataxform examples 39
Cleaning Up a Previous dataxform Session 43
Checking for Hard-Link Files Inside the GuardPoint with dataxform 44
Monitoring dataxform 45
Recovering a Failed or Incomplete dataxform Session 51
Automatic and Manual GuardPoints 56
dataxform Examples and Full Command Syntax 58
Unencrypting Data 63

Common dataxform examples

Displaying dataxform Version Information

```
# dataxform --version

Build version: 78
Build Date: 2023-12-19
```

Verifying that a Guarded Directory Can be Rekeyed with dataxform

The `dataxform --rekey_supported --gp <guard point path>` command verifies that the specified GuardPoint is being guarded with a valid data transformation policy and is ready to be rekeyed with `dataxform`.

In the following example, the GuardPoint has a rekey policy:

```
# dataxform --rekey_supported --gp /opt/apps/dx2
Checking if data transform is supported for guard point
/opt/apps/dx2
Data transformation is supported on /opt/apps/dx2
```

In the following example, the GuardPoint has a standard policy and therefore cannot be rekeyed with `dataxform`.

```
# dataxform --rekey_supported --gp /opt/apps/apps1/doc
Checking if data transform is supported for guard point
/opt/apps/apps1/doc
The kernel component doesn't support data transform on /opt/apps/apps1/doc

Verify this is a guard point with valid data transformation policy, and check the system
log files for any other problems. It may be due to one or more of following reasons;
1.policy has no valid key rule(s), and/or 2. policy has no key_op rule, and/or 3. policy
has valid permit rule(s), and/or 4. policy rule that contains key_op in the action field
also specifies other actions.
```

> **Note**
> You can also get the message "`not a guard point or there is no data transformation rule`" when an administrator is inside the GuardPoint or accessing files in the GuardPoint. Check that no one is in the GuardPoint and that a rekey policy is applied to the GuardPoint. If a GuardPoint does not qualify for rekeying, check that a key is configured in the **Data Transformation Rules** tab of the assigned policy in the Management Console.

Estimating the dataxform Runtime Period

Rekeying the files in a GuardPoint can take a long time if there are thousands of files to be rekeyed. The `--deep_scan` argument on the `dataxform` command simulates and estimate how long a `dataxform` session will take on a specified GuardPoint. Enter all the arguments that you would normally use for the `dataxform` session, and assign the rekey policy to the GuardPoint, so `dataxform` accurately simulates the actual rekey process.

> **Note**
> `--deep_scan` is CPU and I/O intensive. Expect a drop in system performance while running `--deep_scan`. It can take a long time to complete--enough so that for very small file systems, you are better off running `dataxform` directly rather than trying to estimate how long it will take with the `--deep_scan` option.

The following shows a simple `dataxform` command line session with the `--deep_scan` argument.

```
# dataxform --deep_scan --gp /opt/apps/dx4

Checking if data transform is supported for guard point /opt/apps/dx4
Data transformation is supported on /opt/apps/dx4
About to perform a deep scan of the guard point
-- To enable consistent results, do not access the guard point during
   the scan
-- Transformation simulations will be performed, which may take some time
Do you wish to continue (y/n)?y

Status information for directories in guard point
Directory /opt/apps/dx4/ab_dir/ contains 140 files (1 GB)
Directory /opt/apps/dx4/ac_dir/ contains 734 files (21 GB)
Directory /opt/apps/dx4/aa_dir/ contains 40 files (15 MB)
Directory /opt/apps/dx4/ad_dir/ contains 37 files (70 MB)
Directory /opt/apps/dx4/ contains 46 files (915 MB)
Scan found 997 files (23 GB) in 5 directories for guard point /opt/apps/dx4
Estimated data transformation time for /opt/apps/dx4 is 1h 13m 6s
-- This is an estimated time using actual data
   transformations on test files
-- The actual transformation time may be more or less than this estimate
```

The second line of the `--deep_scan` output in the example above is "`Data transformation is supported on /opt/apps/dx4`". This line is displayed when the directory being scanned is an active GuardPoint with a rekey policy. This line is not displayed when the directory being scanned is a regular directory, a disabled GuardPoint, or an active GuardPoint with a regular, non-rekey policy.

The `--deep_scan` results are also echoed to `/var/log/vormetric/vordxf-_path_usr.log` and forwarded to the **Logs** windows.

Manually Running dataxform on Specific Files

Use the following procedure to manually execute `dataxform` on a specific set of files in a GuardPoint.

1. Back up the data in the GuardPoint.
2. If specific files are to be encrypted, create a file list.

   A file list is a text file that consists of the full path name of each file to be transformed. Enter one file path per line. If a file list is not specified, `dataxform` will rekey all the files in the GuardPoint.

3. Log on to the Management Console as an administrator of type Security Administrator with `Host` role permissions or type All.

   Note: Existing active GuardPoints must be disabled before running a manual data transformation.

4. For an existing GuardPoint, disable it. For new GuardPoints, go to the next step.
   a. Open the **GuardPoint** tab of the host with the GuardPoint to be transformed. The applied policies and GuardPoints of the host are displayed.
   b. Disable the GuardPoint that is currently in effect. Select the **Select** check box for the GuardPoint and click **Disable**.
   c. Confirm that the GuardPoint is disabled:
      - For Linux and UNIX systems: execute the `secfsd -status guard` command repeatedly until the GuardPoint is no longer displayed.
      - For Windows systems: on the task bar, right-click the Vormetric Tray Icon and click **View > File System > GuardPoints** until the GuardPoint is no longer displayed.
5. Create a `dataxform` *policy* and apply it to the now disabled or newly created GuardPoint. The `dataxform` policy specifies the following:
   - **Action**: key_op
   - **Effect**: apply_key, permit
   - **Key Selection Rules** key: The original key currently in use. Use clear_key if unencrypted.
   - **Data Transformation Rules** key: The new key. Use clear_key if decrypting.
6. Confirm that the GuardPoint is re-enabled:
   - For Linux and UNIX systems: execute the `secfsd -status guard` command repeatedly until the GuardPoint is displayed.
   - For Windows systems: On the task bar, right-click the Vormetric Tray Icon and click View > File System > GuardPoints until the GuardPoint is displayed.
7. Execute the `dataxform` command with the desired options on the host system. For example:

   ```
   #dataxform --rekey_list --file_list dx_fileList.txt
   --gp /home/apps/apps1/data --dir_recovery /root
   --dir_recovery allows you to specify where dataxform status files are placed.
   ```

8. (Optional) Monitor `dataxform` progress on the host system.

   ```
   # tail -f /var/log/vormetric/vordxf_path_usr.log
   ```

9. Wait until `dataxform` completes.
10. Disable or delete the `dataxform` policy and replace with a production policy. Reboot the host if you cannot disable or delete the rekey policy

CAUTION
Do not apply a policy that is configured for encryption to a directory that contains unencrypted files because, when `apply_key` is configured, the unencrypted files are encrypted when they are accessed. The data will be unusable if read and corrupted if saved.

Running Automatic Data Transformation

You can automatically transform your GuardPoint data by creating a `dataxform` policy and placing a file called `dataxform_auto_config` in the top-level directory of the GuardPoint.

The `dataxform_auto_config` file consists of one or two lines. The first line is mandatory. It specifies an internally used version number. Currently, the internal version number is `1`. Set the first line to "`version=1`". The second line is optional. The second line lists additional `dataxform` parameters. By default, `dataxform` executes the `--rekey` and `--gp` options. These options rekey all the files in the GuardPoint. Do not enter the `--rekey_list`, `--file_list`, or `--gp` options in the `dataxform_auto_config` file. An example `dataxform_auto_config` file is shown below:

```
version=1
--thd 4 --preserve_modified_time
```

Automatic Data Transformation Notes and Limitations

- Low-power systems can run out of memory while running `dataxform`. If entries like "`[VMD] [ERROR] [1933564] [DXF4328E] Kernel component gave unexpected status 4.`" and "`[VMD] [ERROR] [3670108] [DXF4300E] Out of Memory`" are sent to the system messages file, lower the `--thd` parameter value. It will take longer to run, but `dataxform` will use less memory and should complete successfully.
- *Automatic dataxform* processes directories and files according to the native sort order of the system. Automatic `dataxform` is aware of the files currently being processed. If the automatic `dataxform` process is interrupted, you can restart it and it will resume from roughly where it had left off. It will resume within a range of files generally equal to the number of open threads that were running, using the files listed in `./dataxform_status-_gp` and `./dataxform_status-alt-_gp`, and based on the system sort order. We strongly recommend that no one access the GuardPoint during data transformation because depending where `dataxform` is in the list of directories and files to process, the directories and files that you create can be skipped, and changes that you make to the files can go unnoticed. You will then have to manually transform the new or modified files. Additionally, if you do not configure manual `dataxform` properly, it is easy to rekey a file twice, thus corrupting that file. Automatic `dataxform`, on the other hand, is easier to recover.

To run automatic dataxform

1. Back up and block all access to the GuardPoint.
2. Create a `dataxform_auto_config` file.
3. Log on to the Management Console as an administrator of type Security Administrator with `Host` role permissions or type All.
4. Open the **GuardPoint** tab of the host with the GuardPoint to be transformed. The applied policies and GuardPoints of the host are displayed.
5. Disable the GuardPoint that is currently in effect. Select the **Select** option for the GuardPoint. Click **Disable**.
6. (Optional) Enter the `df` command on the host system repeatedly until the secfs mount for the GuardPoint is no longer displayed, or execute the "`secfsd -status guard`" command repeatedly until the GuardPoint is no longer displayed.
7. Copy the `dataxform_auto_config` file into the GuardPoint. Data transformation should start within seconds.

8. Apply the `dataxform` policy to the now disabled GuardPoint.

   (Optional) Execute the "`secfsd -status guard`" command repeatedly on the host system until the GuardPoint and rekey policy are displayed. You can also keep clicking the **Refresh** button in the **Edit Host** window, **GuardPoint** tab, until the green status ball is displayed.

9. (Optional) Monitor `dataxform` progress on the host system.

   `# tail -f /var/log/vormetric/vordxf_path_usr.log`

10. Check log files to verify successful `dataxform` completion.

    The `/var/log/vormetric/vordxf_path_usr.log` file lists the success or failure of `dataxform`, the files that were affected, and the actions taken. Refer to "Using dataxform_status* Files" on page 47 for details.

    Check the rekey status in the **Logs** window.

11. Disable or delete the `dataxform` policy. Reboot the host if you cannot disable or delete the rekey policy.

12. Delete the `dataxform_auto_config` file from the GuardPoint.

    If you do not delete the `dataxform_auto_config` file, the next time you apply a rekey policy to the GuardPoint, data transformation will begin immediately. It is better to copy the file into the GuardPoint when you are ready.

13. Apply a production policy to the GuardPoint. If the `dataxform` policy used an encryption key, be sure to use the same key in the production policy.

CAUTION

Do not apply a policy that is configured for encryption to a directory that contains unencrypted files because, when `apply_key` is configured, the unencrypted files are encrypted when they are accessed. The data will be unusable if read and corrupted if saved.

Cleaning Up a Previous dataxform Session

When `dataxform` execution completes (or aborts), it leaves behind status files that it uses to regulate subsequent executions. Whenever the `dataxform` starts, it looks for these files and if it finds them, either resumes running where it left off, or displays an informative message and exits without transforming any files. The exiting prevents `dataxform` from running repeatedly. Prior to running `dataxform`, a protected host administrator must execute the utility's "cleanup" function to eliminate status files from previous runs. If a transformation fails, the protected host administrator must repair the problem, complete transformation, and *then* execute the cleanup function.

WARNING

NEVER run a cleanup operation after a partial transformation, as you will be unable to resume transforming the remaining files.

When `dataxform` is run, it checks the status in the `dataxform_status-gp` file. If the status is "done", `dataxform` exits with "`data transform status for GuardPoint_dir: previous attempt completed`" error message. Remove the status file and other configuration files with the `--cleanup` option.

The command syntax for cleaning a previous `dataxform` session is:

```
# dataxform --cleanup --gp gp
```

For example:

```
# dataxform --cleanup --nq --gp /opt/apps/dx2
About to remove the data transformation status files
```

The `--cleanup` operation does not return a message indicating a successful completion. If there are no errors, the cleanup operation completed successfully, and you can now run `dataxform` on the GuardPoint.

Checking for Hard-Link Files Inside the GuardPoint with dataxform

Guarding and transforming linked files is potentially dangerous. Links require special consideration. Depending on how policies and keys are applied to both the link file/directory and the source file/directory, there is the potential for policy conflicts and key mismatches. A key mismatch will corrupt data files when they are saved. See also "Dataxform and Linked Files" on page 20 for more information.

With regard to links:

- There is only one instance of a file; where, with links, it can have multiple names. The instance is transformed when the first name for it is encountered. Subsequent names are recognized as aliases of the already transformed instance and are ignored.
- Never make a links to sources outside of the GuardPoint. If a file in a GuardPoint is linked to a file in another GuardPoint, and encryption is applied, cross-changes to the same file from different GuardPoints will use different keys and corrupt files.
- If a file in a GuardPoint is a link to a file that is not in any GuardPoint, then only standard operating system access controls are applied to the file outside the GuardPoint. That means that root, and possibly others, can access and modify the file instance without an encryption key and without restriction. Only the file instance in the GuardPoint should be accessed, otherwise the key will not be used to decrypt/encrypt the instance. If the instance is changed and saved outside of the GuardPoint, it will be corrupted.
- Security rules for link files inside the GuardPoint must be intelligently written to anticipate how files are accessed and how keys are applied. For example, you can create a file named `a.foo` and create a link to it named `a.bar`. If you write a security rule that applies one key to `*.foo` and another key to `*.bar`, the key applied depends upon how the file is accessed. Again, this is an opportunity for cross-changes and file corruption.

A GuardPoint should be inactive when you transform its contents. Make sure that linked files are also inactive.

The `--check_links` example below detected links. It found three hard links to files outside the GuardPoint and two to files inside the GuardPoint. The `dataxform` output includes the full path names of the hard-link files detected in the GuardPoint. Note the lines that read:

```
Found 1 references to hard link but expected 2
```

Because `dataxform` "expected" more links than it found, a likely culprit is that the file inside the GuardPoint is linked to a file outside the GuardPoint. We strongly recommend that links do not go outside the GuardPoint because policy and key application can become unpredictable and corrupt files.

Also, note the lines that read:

```
Found 2 references to hard link
```

These lines indicate that the link file and the source file both reside inside the GuardPoint.

```
# dataxform --check_links --gp /opt/apps/dx2
Status information for directories in guard point
Directory /opt/apps/dx2/ab_dir/ contains 2002 files (57 KB)
Directory /opt/apps/dx2/ac_dir/ contains 2000 files (57 KB)
```

```
Directory /opt/apps/dx2/ad_dir/ contains 2000 files (57 KB)
Directory /opt/apps/dx2/aa_dir/ contains 2000 files (57 KB)
Directory /opt/apps/dx2/ contains 2003 files (43 KB)
Scan found 10005 files (273 KB) in 5 directories for guard point /opt/apps/dx2
Scanning for hard links in directory /opt/apps/dx2
Found 1 references to hard link but expected 2
Hardlink reference 1 /opt/apps/dx2/hardlinkfile01
Found 1 references to hard link but expected 2
Hardlink reference 1 /opt/apps/dx2/hardlinkfile02
Found 1 references to hard link but expected 2
Hardlink reference 1 /opt/apps/dx2/hardlinkfile03
Found 2 references to hard link
Hardlink reference 1 /opt/apps/dx2/hardlinkedfileLocal02
Hardlink reference 2 /opt/apps/dx2/ab_dir/linkedfile02
Found 2 references to hard link
Hardlink reference 1 /opt/apps/dx2/hardlinkedfileLocal01
Hardlink reference 2 /opt/apps/dx2/ab_dir/linkedfile01
```

Monitoring dataxform

Dataxform activity is recorded in three places:

- The **Message Logs** window on Windows.
- `/var/log/vormetric/vordxf_`*`path_usr`*`.log`
- `/var/log/vormetric/dataxform_status-`*`gp`* and `dataxform_status_alt-`*`gp`* files

Using the Message Log Window

During a rekey operation, the key manager logs both the current encryption key and the new encryption key. Transformed files are listed four times because multiple operations occur during a rekey operation (the logging level is set to `INFO` and `audit` is enabled). This can result in an extremely large number of log entries. In the example below, a GuardPoint is opened with the `clear_key` key and saved with the `aes128` key.

Ensure that there are no errors in the key manager log that have to do with `dataxform` or `DXF`. Look for errors that contain strings like “denied” and “failed”. For example,

```
[DXF4376E] Data transform in guard point /opt/apps/dx9 failed for 6 files
```

```
[DXF4271E] Number of files in error due to a signal stopping the dataxform: 6
```

The example errors indicate that `dataxform` had been interrupted as it was actively transforming six files.

The `dataxform` messages indicate that `dataxform` is supported, `dataxform` detected a number of files, it transformed that same number of files, and then completed successfully.

If errors are generated, check the `dataxform_status_skip-_`*`gp`* file for a list of the files that were in the process of being transformed, but are now in an unknown state.

Using vordxf_* Files

Data transformation activity is recorded and prepared for transmission to the key manager. One file that records transformation activity is updated dynamically and should be checked first should any problems arise. You may want to use the “`tail -f`” command on the file to monitor the data transformation process in real-time.

The data transformation log file resides in `/var/log/vormetric` on the host system.

The base name of the data transformation log file is `vordxf_path_usr.log`, where *path* is the underscore-separated (_) full path to the GuardPoint directory and *usr* is the name of the user executing `dataxform`. Only the Linux or AIX `root` user, or the Windows `Administrator` user, can run `dataxform`. For example, `vordxf-_opt_apps_dx9_root.log`.

Always wait for the "`Data transform for guard point path completed ...`" message before making any changes to the GuardPoint.

```
...
2011-01-17 12:45:55.846 [VMD] [INFO ] [15792] [DXF4344I] Data transform version 4.4.1.0
2011-01-17 12:45:55.857 [VMD] [INFO ] [15792] [DXF4429I] Data transformation is
supported on /opt/apps/dx9
2011-01-17 12:45:55.886 [VMD] [INFO ] [15792] [DXF4378I] Scan found 63 files (1 KB) in 5
directories for guard point /opt/apps/dx9
2011-01-17 12:45:55.889 [VMD] [INFO ] [15792] [DXF4366I] The current operation took 0
hours, 0 minutes and 0 seconds
2011-01-17 12:45:56.220 [VMD] [INFO ] [15792] [DXF4380I] Transformed 69 files (1 KB) of
63 files (1 KB) for guard point /opt/apps/dx9
2011-01-17 12:45:56.224 [VMD] [INFO ] [15792] [DXF4366I] The current operation took 0
hours, 0 minutes and 1 seconds
2011-01-17 12:45:57.242 [VMD] [INFO ] [15792] [DXF4371I] Data transform skipped some
files
2011-01-17 12:45:57.255 [VMD] [INFO ] [15792] [DXF4201I] The file
/opt/apps/dx9/filenothere01 was skipped. It was a soft link
2011-01-17 12:45:57.283 [VMD] [INFO ] [15792] [DXF4201I] The file
/opt/apps/dx9/filenothere02 was skipped. It was a soft link
2011-01-17 12:45:57.286 [VMD] [INFO ] [15792] [DXF4201I] The file
/opt/apps/dx9/filenothere03 was skipped. It was a soft link
2011-01-17 12:45:57.289 [VMD] [INFO ] [15792] [DXF4200I] The file
/opt/apps/dx9/hardlink01 was skipped. It was an additional hard link
2011-01-17 12:45:57.292 [VMD] [INFO ] [15792] [DXF4200I] The file
/opt/apps/dx9/hardlink02 was skipped. It was an additional hard link
2011-01-17 12:45:57.299 [VMD] [INFO ] [15792] [DXF4200I] The file
/opt/apps/dx9/hardlink03 was skipped. It was an additional hard link
2011-01-17 12:45:57.302 [VMD] [INFO ] [15792] [DXF4257I] Number of additional hard links
skipped: 3
2011-01-17 12:45:57.307 [VMD] [INFO ] [15792] [DXF4258I] Number of soft links skipped: 3
2011-01-17 12:45:57.312 [VMD] [INFO ] [15792] [DXF4365I] The data transform operation
took 0 hours, 0 minutes and 2 seconds
2011-01-17 12:45:57.316 [VMD] [INFO ] [15792] [DXF4363I] Data transform for guard point
/opt/apps/dx9 finished but 6 files were skipped
```

`dataxform` logging is tied to VMD logging. The size of the `vordxf_path_usr.log` file is set by the **Maximum File Size** text-entry box on the **FS Log** tab of the key manager. The default maximum `vordxf_path_usr.log` file is 1 MB.

The following `vordxf_path_usr.log` excerpt shows a `dataxform` session that scans the directory and subdirectories for files to transform, performs a policy check with the key manager, and successfully transforms all but three files.

```
2011-01-10 14:37:03.970 [VMD] [INFO ] [26158] [DXF4344I] Data transform version 4.4.1.0
2011-01-10 14:37:03.997 [VMD] [INFO ] [26158] [DXF4345I] Data transform version 4.4.1.0,
using policy aes128_to_clear_dx on guard point /opt/apps/lib/dx4
2011-01-10 14:37:04.155 [VMD] [INFO ] [26158] [DXF4429I] Data transformation is
supported on /opt/apps/lib/dx4
2011-01-10 14:37:06.564 [VMD] [INFO ] [26158] [DXF4378I] Scan found 10000 files (273 KB)
in 5 directories for guard point /opt/apps/lib/dx4
2011-01-10 14:39:25.797 [VMD] [INFO ] [26158] [DXF4380I] Transformed 10003 files (273
```

```
KB) of 10000 files (273 KB) for guard point /opt/apps/lib/dx4
2011-01-10 14:39:35.949 [VMD] [INFO ] [26158] [DXF4371I] Data transform skipped some
files
2011-01-10 14:39:35.980 [VMD] [INFO ] [26158] [DXF4201I] The file
/opt/apps/lib/dx4/filenothere01 was skipped. It was a soft link
2011-01-10 14:39:35.986 [VMD] [INFO ] [26158] [DXF4201I] The file
/opt/apps/lib/dx4/filenothere02 was skipped. It was a soft link
2011-01-10 14:39:35.989 [VMD] [INFO ] [26158] [DXF4201I] The file
/opt/apps/lib/dx4/filenothere03 was skipped. It was a soft link
2011-01-10 14:39:36.000 [VMD] [INFO ] [26158] [DXF4258I] Number of soft links skipped: 3
2011-01-10 14:39:36.050 [VMD] [INFO ] [26158] [DXF4363I] Data transform for guard point
/opt/apps/lib/dx4 finished but 3 files were skipped
Check the status files in /var/log/vormetric, or the status file location you
configured, for dataxform_status_skip-_gp and dataxform_status_error-_gp files. The
following is the dataxform_status_skip-_gp file for the preceding example. These are
link files and link file cannot be transformed.
```

```
# cat dataxform_status_skip-_opt_apps_lib_dx4
Skipped, the name refers to a softlink : /opt/apps/lib/dx4/filenothere01
Skipped, the name refers to a softlink : /opt/apps/lib/dx4/filenothere02
Skipped, the name refers to a softlink : /opt/apps/lib/dx4/filenothere03
```

An error status file was not generated. The `dataxform_status_error-_gp` file is generated when `dataxform` fails to complete. It did complete successfully, so no error status file is generated. The error status file consists of the full path of each file that was being processed when `dataxform` failed. The files are not necessarily corrupt. You must check each file in the list to determine its current status.

Using dataxform_status* Files

Several `dataxform_status_*` files are used to run `dataxform`. One file you generate. The other files are generated by `dataxform` to track the `dataxform` session. Each `dataxform_status` file contains specialized records of the last or current `dataxform` session. The files are placed in `/var/log/vormetric` by default, but you can specify an alternate location using the `--dir_recovery` argument to `dataxform`.

The files are:

```
dataxform_status-_gp
dataxform_status_alt-_gp
dataxform_status_error-_gp
dataxform_status_skip-_gp
```

where, *gp* is the full path to the GuardPoint. The slash (`/`) and backslash (`\`) path delimiters are replaced with underscores (_).

```
# pwd
/var/log/vormetric
# ls dataxform_status*
dataxform_status_alt-_opt_apps_lib_dx1
dataxform_status_error-_opt_apps_lib_dx1
dataxform_status_skip-_opt_apps_lib_dx1
dataxform_status-_opt_apps_lib_dx1
dataxform_status-_opt_apps_lib_dx2_aa_dir
```

where `/opt/apps/lib/dx1` and `/opt/apps/lib/dx2/aa_dir` are the actual GuardPoint paths.

The format and use of each file are described below.

`dataxform_status-_gp` and `dataxform_status_alt-_gp` are used together to monitor data file processing. These status files indicate the threads that are started, the threads that are running, their status, and sequence.

For a `dataxform` failure or interruption, the `dataxform_status-_gp` and `dataxform_status_alt-_gp` files can be used to guesstimate the files that have been completed and the files that still must be transformed. Some manual verification will be needed to verify precisely the files that have and have not been transformed. You can use this information to create a file list and manually resume the `dataxform` session, or, you can just leave these status files in place and resume automatic `dataxform`. Automatic `dataxform` will determine where to resume.

The structure of the `dataxform_status-_gp` and `dataxform_status_alt-_gp` files is:

```
version=n
status=stat
operation=action
current=file
n in-progress files
seqno=n
hmac=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

where:

- `version=n` is an internally used version number and can be ignored.
- `status` is the current `dataxform` processing status. Status can be `done`, `in-progress`, `stopped`, or `undone`. An `in-progress` status indicates that `dataxform` is still processing data files. Do not work in, or access, the GuardPoint while `dataxform` is still `in-progress`. Otherwise, you risk corrupting data. A `stopped` status indicates that the last `dataxform` session was interrupted before it could complete. The `done` status means that `dataxform` completed. Note that, though `dataxform` completes, not all files are necessarily transformed. Check the log files for data files that have not been transformed. `undone` indicates that rekey from a file list had been performed.
- `operation=action` is the operation specified on the `dataxform` command line. It can be either `rekey` or `rekey_list`. Automatic `dataxform` is implicitly configured to operate with `rekey`.
- `current=file` is the full path name of the file currently being processed. When `status` is `done`, the `current` parameter is blank. In every other case, `current` will be set to some value, such as the default unset value `-1`.
- `n in-progress files` is the total number of files being concurrently processed by `dataxform`. Each file is transformed as a separate sub-process, or thread. `n` is the value of the `--thd` paramerter passed to the `dataxform` command, and it can be between 1 and 32. The default number of threads is 8 or the number of CPUs, whichever is less. For specifics on the `-thd` parameter option, see "dataxform Examples and Full Command Syntax" on page 58
- `seqno=n` is the sequence number. There are two status files, `dataxform_status-_gp` and `dataxform_status_alt-_gp`. The `dataxform` utility writes status information to one file. When a sub-process completes and a new data file is opened for transformation, the current status file is closed, and the other status file is opened with the new data file as the `current=file` file. The `seqno` number increments each time `dataxform` begins to process the next data file. If a `dataxform` session fails or is interrupted, check the `seqno` number. Use the status file with the higher `seqno` number to determine the files that were being processed when `dataxform` stopped. The status file with the lower `seqno` number indicates the last fully processed data file in the `current=file` parameter. The status file with the higher `seqno` number indicates the last opened data file in the `current=file` parameter. Most likely the last opened data file was not successfully transformed.
- `hmac=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx` is a check value used to ensure the integrity of the status files, and it can be ignored.

By default, the log files are generated by `dataxform` during manual and automatic data transformation and placed in `/var/log/vormetric` on Linux systems. An alternate location can be specified using the `--dir_recovery` argument to `dataxform`.

A successful `dataxform` status file looks like:

```
# cat ./dataxform_status-_opt_apps_lib_dx2_aa_dir
version=5
status=done
operation=rekey
current=
0 in-progress files
seqno=2004
hmac=6BI53B320C455A45E013AC21F353CE0BBCF159440B2B32F3F3
000FBB0BA7B3C5267D32D60385C786FEDA8C59D57F1C44
5588A9FCAB56CC642E8B2E5601D09CA7E9F6AFFA886ADR6
2A8FE559CF
C89B9F55G
3E65F4
I8FD574
```

Note that the `status` is `done`, there is no `current` file, and there are 0 files being processed. The `seqno` number is the number of files in the GuardPoint. There is no corresponding `dataxform_status_alt-_gp` file because that file is deleted when `dataxform` completes successfully. Also, look for a `dataxform_status_skip-_gp` file to see what files in the GuardPoint, if any, were detected but not transformed.

The `dataxform_status-_gp` and `dataxform_status_alt-_gp` files for a `dataxform` session are shown below. Some things to note:

- The number that precedes every file path in the status file is the offset into the `dataxform` file list. It is used to quickly locate entries if `dataxform` needs to be restarted. This number of can be ignored.
- The `dataxform` utility was running 8 sub-processes, as shown in the `8 in-process files` field. The 8 files being processed are shown beneath this entry.
- The `seqno` field indicates that `dataxform_status_alt-_Guard` is the true snapshot of what `dataxform` was doing while the files were captured or `dataxform` failed because it has the higher sequence number. The other status file, `dataxform_status-_Guard`, is older because it has the lower `seqno` number.

  This means that all of the files shown in `dataxform_status_alt-_Guard` were being processed when `dataxform` stopped.
- The final line with the prefix `hmac=` is a checksum field. This is used to verify that the file content is intact should the transform be interrupted and need to be restarted.

  All data in the status files after this line are 'noise'. This occurs for efficiency reasons. The `dataxform` program writes the status files using a simple block write of all the status information into the existing file, overwriting any entries already present. If, as often happens, the existing status file is larger than the data being written, then some of the stale entries may be visible after the final checksum line. The additional effort of removing this stale data for each file transformation would add a significant additional amount of time to the overall transformation, so this 'noise' is simply left and can be ignored.

```
# cat dataxform_status-_Guard
version=5
status=in-progress
operation=rekey
current=48986 /Guard/dir_47/boot/grub/stage2
8 in-progress files
48881 /Guard/dir_47/boot/grub/reiserfs_stage1_5
48936 /Guard/dir_47/boot/grub/fat_stage1_5
```

```
48333 /Guard/dir_47/boot/initramfs-2.6.32-71.el6.x86_64.img
48758 /Guard/dir_47/boot/vmlinuz-2.6.32-71.el6.x86_64
46899 /Guard/dir_5/boot/initramfs-2.6.32-71.el6.x86_64.img
48821 /Guard/dir_47/boot/config-2.6.32-71.el6.x86_64
48986 /Guard/dir_47/boot/grub/stage2
47316 /Guard/dir_5/boot/vmlinuz-2.6.32-71.el6.x86_64
seqno=738
hmac=2ED53B320C455A45E013AC21F353CE0BBCF159440B2B32F3F3
000FBB0AD9B3C5267D32D60385C786FEDA8C59D57F1C52
7788A9FCAB56CC642E8B2E5601D09CA7E9F6AFFA886ADE1
2A8FE559FC
C89B9F55E
3E65F5
D8FD544
```

```
# cat dataxform_status_alt-_Guard
version=5
status=in-progress
operation=rekey
current=49031 /Guard/dir_47/boot/grub/ufs2_stage1_5
8 in-progress files
49031 /Guard/dir_47/boot/grub/ufs2_stage1_5
48936 /Guard/dir_47/boot/grub/fat_stage1_5
48333 /Guard/dir_47/boot/initramfs-2.6.32-71.el6.x86_64.img
48758 /Guard/dir_47/boot/vmlinuz-2.6.32-71.el6.x86_64
46899 /Guard/dir_5/boot/initramfs-2.6.32-71.el6.x86_64.img
48821 /Guard/dir_47/boot/config-2.6.32-71.el6.x86_64
48986 /Guard/dir_47/boot/grub/stage2
47316 /Guard/dir_5/boot/vmlinuz-2.6.32-71.el6.x86_64
seqno=739
hmac=E113351BDB7497AAD150DEC57953DB0E57401404F564733E71
02A612158F65D7ACC2E1D9A75CB94ED91751397CDFD1CF
EE3483B58DFD6DD4DC722D
0CD47F89890DFA572DE7F3E4E9F
4
0A2
E78F4
F8
7B
DC8D603
```

Using dataxform_auto_config

The `dataxform_auto_config` file is user-generated and consists of one or two lines and is placed in the GuardPoint to initiate automatic data transformation. Usually, it contains just one line, “`version=1`”. The version is an internally used number that currently must be set to `1`. File usage is described in "Running Automatic Data Transformation" on page 42.

```
# cat dataxform_auto_config
version=1
```

Note
When you assign a rekey policy to a GuardPoint, a message like the following is issued in the `secfsd.log` file every 20 seconds until a `dataxform_auto_config` file is placed in the GuardPoint, manual `dataxform` is run on the GuardPoint, or a non-rekey policy is applied to the GuardPoint:
`/opt/vormetric/DataSecurityExpert/agent/secfs/.sec/current/bin/secfsd[29252]: Information: Mon Feb 21 11:54:37 20121 do_dataxform: no auto config file exists; not starting dataxform`. This information is not displayed in the **Logs** window.

Using dataxform_auto_lock

The `dataxform_auto_lock` file is a two-line file generated by `dataxform` and is placed in the GuardPoint during manual and automatic data transformation. The first line is the name of the host system on which `dataxform` is being executed. The second line indicates the `dataxform` status. Status is “in-progress”, “stopped”, or “done”. A stopped status indicates that the last `dataxform` session was stopped before it reached completion. An example of a successful `dataxform` run is:

```
# cat dataxform_auto_lock
done
```

If a `dataxform_auto_lock` file is present in a GuardPoint, and/or other `dataxform` configuration files are in `/var/log/vormetric` (except for `dataxform_auto_config`) `dataxform` will exit with an error. You cannot perform a new data transformation in the Guardpoint unless you delete the `dataxform_auto_lock` file and the `dataxform_*` files in `/var/log/vormetric`. The recommended way to remove old status files is to run `dataxform` with the `--cleanup` argument.

WARNING
Do not use `--cleanup` if a previous `dataxform` session did not complete successfully (see "Recovering a Failed or Incomplete dataxform Session" below).

Example of using the `--cleanup` argument to remove old status files:

```
# cat /opt/apps/lib/dx1/ad_dir/dataxform_auto_lock
aix4200
done
# dataxform --cleanup --nq --gp /opt/apps/lib/dx1/ad_dir
About to remove the data transformation status files
```

You can now run `dataxform` again and successfully rekey files in the GuardPoint.

Recovering a Failed or Incomplete dataxform Session

A `dataxform` session can fail for different reasons: `dataxform` can be canceled, the process is killed, the system crashes, and so on. The recovery process is similar for manual and automatic `dataxform`.

For more information, see:

- "Restarting an Incomplete Automatic dataxform Session" below
- "Recovering from a Failed dataxform Session" on the next page

Restarting an Incomplete Automatic dataxform Session

Although unlikely, if an automatic `dataxform` session fails to complete, your intervention may be required to allow it to resume.

In the case of a system restart--for example after a power failure--the automatic session will be resumed when the GuardPoint is mounted during the boot sequence. However, if the `dataxform` session terminated unexpectedly (for example, if the session had been killed), then the system will not restart it automatically, and the GuardPoint status needs to be reset so that another automatic session will be initiated.

The following steps reset the GuardPoint status:

1. Verify that the session really has terminated unexpectedly by verifying that no existing `dataxform` process is running. Use the standard system tools (`ps` or task manager).
2. From your key manager, disable the GuardPoint, then wait until the GuardPoint status on the CTE Agent no longer lists the GuardPoint.
3. From your key manager enable the GuardPoint.

An automatic `dataxform` session will start soon after this, and the session will continue from where it previously stopped.

Recovering from a Failed dataxform Session

The following is an example of how to recover from a failed `dataxform` session. The GuardPoint in this example is `/opt/apps/dx9`.

1. A manual `dataxform` session is run, then canceled using Ctrl-c:

```
# dataxform --rekey --nq --print_stat --preserve_modified_time --gp /opt/apps/dx9
Checking if data transform is supported for guard point /opt/apps/dx9
Data transformation is supported on /opt/apps/dx9
About to perform the requested data transform operation
-- Be sure to back up your data
-- Do not access files in the guard point during the transform process
-- Please do not attempt to terminate the application
Scan found 10003 files (273 KB) in 5 directories for guard point /opt/apps/dx9
The current operation took 0 hours, 0 minutes and 2 seconds
Shutting down data transform: received fatal signal 2
Transformed 1126 files (24 KB) of 10003 files (273 KB) for guard point /opt/apps/dx9
The current operation took 0 hours, 0 minutes and 7 seconds
Data transform got errors on some files
The file /opt/apps/dx9/datafile931 could not be transformed, a signal stopped the
data transform
The file /opt/apps/dx9/datafile1102 could not be transformed, a signal stopped the
data transform
The file /opt/apps/dx9/datafile1121 could not be transformed, a signal stopped the
data transform
The file /opt/apps/dx9/datafile1100 could not be transformed, a signal stopped the
data transform
The file /opt/apps/dx9/datafile1120 could not be transformed, a signal stopped the
data transform
The file /opt/apps/dx9/datafile1132 could not be transformed, a signal stopped the
data transform
Number of files in error due to a signal stopping the dataxform: 6
The data transform operation took 0 hours, 0 minutes and 7 seconds
Could not complete data transform for guard point /opt/apps/dx9, data transform was
interrupted by a signal
Data transform for guard point /opt/apps/dx9 finished but 6 files were not processed
due to errors
```

The `dataxform_status-_opt_apps_dx9`, `dataxform_status_error-_opt_apps_dx9`, and `dataxform_dir_list-_opt_apps_dx9` files are created in the log directory, or the GuardPoint, depending on the command line options. By default, all status and log files go to `/var/log/vormetric`. If `--status_gp` was included on the command line, the status files go in the GuardPoint.

The dataxform session log file, `/var/log/vormetric/vordxf-_opt_apps_dx9_root.log`, is updated. It displays the same basic information as displayed on the terminal screen.

2. The `dataxform` messages indicate that the session had been interrupted and that a number of files (6) are in an unknown state due to the interruption.
3. You may optionally use the `--recovery` option to generate a set of files that track the progress `dataxform` made in the GuardPoint (see later). However this step is not required and can be deferred until after a new `dataxform` session has been run to transform the remaining files.
4. Resume the data transformation run, using the same command as you used to start it before.

```
# dataxform --rekey --nq --print_stat --preserve_modified_time --gp /opt/apps/dx9
Checking if /opt/apps/dx9 is a guard point with a rekey policy applied /opt/apps/dx9
is a guard point with a rekey policy applied
Automatic data transform status for /opt/apps/dx9: previous attempt did not complete
Note: data from a previous dataxform run is being used
About to perform the requested data transform operation
-- Be sure to back up your data
-- Please do not attempt to terminate the application
Scan found 10003 files (273 KB) in 5 directories for guard point /opt/apps/dx9
Transformed 10001 files (273 KB) of 10003 files (273 KB) for guard point
/opt/apps/dx9
Data transform got errors on some files
The file /opt/apps/dx9/datafile931 could not be transformed, it was in progress in
the previous data transform
The file /opt/apps/dx9/datafile1102 could not be transformed, it was in progress in
the previous data transform
The file /opt/apps/dx9/datafile1121 could not be transformed, it was in progress in
the previous data transform
The file /opt/apps/dx9/datafile1100 could not be transformed, it was in progress in
the previous data transform
The file /opt/apps/dx9/datafile1120 could not be transformed, it was in progress in
the previous data transform
The file /opt/apps/dx9/datafile1132 could not be transformed, it was in progress in
the previous data transform
The data transform operation took 0 hours, 0 minutes and 7 seconds
Could not complete data transform for guard point /opt/apps/dx9, data transform was
interrupted by a signal
Data transform for guard point /opt/apps/dx9 finished but 6 files were not processed
due to errors
```

5. If for some reason this session also did not complete, performing the same operation again should continue from where the previous session ended. The status records are accumulated across each session.
6. Use the `--recovery` option to generate a set of files that track the progress `dataxform` made in the GuardPoint. Use these files later to determine which files in the GuardPoint have not been transformed.

WARNING

Do not run the `--recovery` option more than once, as a second run may overwrite vital information required to recover any files that did not transform correctly.

```
# dataxform --recovery --gp /opt/apps/dx9
Note: data from a previous dataxform run is being used
Number of files previously in error due to a signal stopping the dataxform: 6
Scan found 10010 files (273 KB) in 6 directories for guard point /opt/apps/dx9
Generating list of files previously transformed on /opt/apps/dx9
Data transform got errors on some files
The file /opt/apps/dx9/datafile931 was previously in error. A signal stopped the
dataxform
The file /opt/apps/dx9/datafile1102 was previously in error. A signal stopped the
dataxform
The file /opt/apps/dx9/datafile1121 was previously in error. A signal stopped the
dataxform
The file /opt/apps/dx9/datafile1100 was previously in error. A signal stopped the
dataxform
The file /opt/apps/dx9/datafile1120 was previously in error. A signal stopped the
dataxform
The file /opt/apps/dx9/datafile1132 was previously in error. A signal stopped the
dataxform
Number of files in error due to a signal stopping the dataxform: 6
The dataxform_files_todo-_opt_apps_dx9 and dataxform_files_done-_opt_apps_dx9 are
created, and the /var/log/vormetric/dataxform_status-_opt_apps_dx9 file may be
updated.

# cat dataxform_status-_opt_apps_dx9
version=5
status=done
operation=rekey
current=
0 in-progress files
seqno=2
hmac=5449453CBAEFCC01EC02542650D8C1040D762D213829F8B3CF967DC578320A475F705C11D3BAF74
D588630CE8078AF46
```

This file indicates that the `dataxform` session had completed.

7. The `/var/log/vormetric/vordxf-_opt_apps_dx9_root.log` file is updated. It displays the same basic information displayed on the terminal screen.

8. The primary files of interest are `dataxform_files_todo-_opt_apps_dx9`, `dataxform_files_done-_opt_apps_dx9`, and `dataxform_status_error-_opt_apps_dx9`.

   `dataxform_files_todo-_opt_apps_dx9` lists the files that `dataxform` had not touched and are yet to be transformed, and any files for which a transform was attempted but for some reason failed.

   `dataxform_files_done-_opt_apps_dx9` lists the files that `dataxform` rekeyed successfully. Leave these files alone.

   `dataxform_status_error-_opt_apps_dx9` lists the files that were being processed at the time an error occurred—for example, when `dataxform` was interrupted in the above example. These are the files that have to be checked individually to determine if they had been processed. They may or may not have been completely processed.

   ```
   # cat dataxform_status_error-_opt_apps_dx9
   Error, was in progress during a previous session: /opt/apps/dx9/datafile931
   Error, was in progress during a previous session: /opt/apps/dx9/datafile1102
   Error, was in progress during a previous session /opt/apps/dx9/datafile1121
   Error, was in progress during a previous session: /opt/apps/dx9/datafile1100
   Error, was in progress during a previous session: /opt/apps/dx9/datafile1120
   Error, was in progress during a previous session: /opt/apps/dx9/datafile1132
   ```

> **Note:** If more than one session restart was required to complete the `dataxform` run, there may be repeated entries in the above list.

```
# cat dataxform_files_todo-_opt_apps_dx9
/opt/apps/dx9/datafile931
/opt/apps/dx9/datafile1100
/opt/apps/dx9/datafile1102
/opt/apps/dx9/datafile1120
/opt/apps/dx9/datafile1121
/opt/apps/dx9/datafile1132

# head -12 dataxform_files_done-_opt_apps_dx9
/opt/apps/dx9/datafile1
/opt/apps/dx9/datafile2
/opt/apps/dx9/datafile3
/opt/apps/dx9/datafile4
/opt/apps/dx9/datafile5
/opt/apps/dx9/datafile6
/opt/apps/dx9/datafile7
/opt/apps/dx9/datafile8
/opt/apps/dx9/datafile9
/opt/apps/dx9/datafile10
/opt/apps/dx9/datafile11
/opt/apps/dx9/datafile12
```

9. Disable the GuardPoint with the rekey policy through your key manager. Do not re-apply the regular policy because you are not able to use the in the GuardPoint. A proper rekey policy restricts access to all the files in the GuardPoint. You have to disable the rekey policy so you can access all the files in the GuardPoint and determine their transformation status.
10. At this point, you should restore the files that were named in the error listing above from a backup, and run a transformation session for just these files, using the "todo" list (`dataxform_files_todo-_opt_apps_dx9`) to select which files are to be transformed. However, depending on the exact nature of the error reported, some entries may need to be removed from the "todo" list, for example, if somehow a file no longer exists, then attempting to transform it again will obviously not work.
11. If you are running automatic `dataxform`, remove the `dataxform_auto_conf` file from the GuardPoint.
12. Re-apply the rekey policy to the GuardPoint, but first be sure that your current directory is not the GuardPoint.
13. Verify that the policy has been successfully re-applied.

```
# secfsd -status guard
GuardPoint Policy Type ConfigState Status Reason
---------- ------ ---- ----------- ------ ------
/opt/apps/dx1 allowAllOps_fs local guarded guarded N/A
/opt/apps/dx3 denyAllOps_fs local guarded guarded N/A
/opt/apps/dx4 allowAllOps_fs local guarded guarded N/A
/dev/dsk/c0t0d0s7 allowAllOps_rd rawdevice guarded guarded N/A
/opt/apps/dx9 clear_to_aes128_dx local guarded guarded N/A
```

14. Run `dataxform` on just the files in the todo list. For example:

```
# dataxform --rekey_list --file_list var/log/vormetric/dataxform_files_todo-_opt_
apps_dx9 --nq --print_stat --preserve_modified_time --gp /opt/apps/dx9
Checking if data transform is supported for guard point /opt/apps/dx9
Data transformation is supported on /opt/apps/dx9
Previous status information does not relate to a --rekey_file operation.
About to perform the requested data transform operation
-- Be sure to back up your data
-- Please do not access files in the guard point during the
transform process
-- Please do not attempt to terminate the application
Starting data transform of /opt/apps/dx9 for files listed in
/var/log/vormetric/dataxform_files_todo-_opt_apps_dx9
The data transform operation took 0 hours, 1 minutes and 20 seconds
bash-3.00#
```

    In this case the `dataxform` completes successfully. The error file was removed because no errors were encountered in the rekey process.

15. Disable the rekey policy.

16. If you are satisfied with the successful completion, clean up the files left by the rekey process.

```
# dataxform --cleanup --gp /opt/apps/dx9
About to remove the data transformation status files
Do you wish to continue (y/n)? y
```

17. Apply a regular policy that uses the applied key.

18. Check that the access controls and encryption keys configured in the policy are working as expected.

Automatic and Manual GuardPoints

A CTE Agent GuardPoint is usually applied immediately after it is configured in the Management Console. This is called an Automatic GuardPoint. However, GuardPoints can also be applied later on a host system. This is called a Manual GuardPoint.

When would you want to apply the GuardPoint later? Consider the case of a 2-node protected host cluster configured as active/passive in a cluster environment, such as Veritas Cluster Server (VCS) or IBM PowerHA (formerly HACMP). There are two nodes, one which is currently active and the other that is currently inactive. Both nodes are locked. You apply GuardPoint protection to active nodes only. You should never apply a GuardPoint to a passive node. If the active node develops a problem and tries to switch over to the inactive node, the cluster process will fail to switch over because the mirror directory on the inactive node is currently mounted on the active node. The solution is for the cluster process to unmount (for example, unguard) the currently active node, place it in an inactive state, place the old inactive node in an active state, and then mount (for example, guard) the mirror directory on the newly active node. Inappropriate switching on any AIX system can spawn messages like:

- “invalid GuardPoint”
- “The directory is not on cluster file system partition shared across nodes”
- “secfsd Failed to unguard <dirpath> - will retry later”
- “Agent is calling clean for resource <resource name> because the resource became OFFLINE unexpectedly, on its own.”

Generally, when you get messages like these, check that only active nodes are properly guarded.

Automatic and Manual GuardPoints are set in the **Edit Host** window, **Guard File System** sub-window.

The GuardPoint type is usually set to `Directory (Auto Guard)` for file system based directories and to `Raw or Block Device (Auto Guard)` when applying GuardPoint protection to raw or block devices. When an auto GuardPoint is applied, regardless if it is a file system directory or a raw device, the change is pushed to the host system, and the GuardPoint is applied immediately. This is evident by using the `df` command to display `secfs` mounts (for example, GuardPoints) or `secfsd` to display the GuardPoints themselves. The `secfsd` output shows a guard type of `local` for directories configured with `Directory (Auto Guard)`.

```
# df
Filesystem                          1K-blocks     Used  Available Use%  Mounted on
/dev/mapper/VolGroup00-LogVol00  40123784  11352236   26733380  30%  /
/dev/sda1                          101086     14590 81277  16%  /boot
none                               254492         0     254492   0%  /dev/shm
/opt/vormetric/DataSecurityExpert/agent/secfs/.sec
                                 40123784  11352236   26733380  30%
/opt/vormetric/DataSecurityExpert/agent/secfs/.sec
/opt/apps/apps1/tmp              40123784  11352236   26733380  30% /opt/apps/apps1/tmp
/opt/apps/apps1/lib              40123784  11352236   26733380  30% /opt/apps/apps1/lib
/opt/apps/apps1/doc              40123784  11352236   26733380  30% /opt/apps/apps1/doc
```

```
# secfsd -status guard
GuardPoint           Policy        Type    ConfigState  Status  Reason
----------           ------         ----   ------------ ------  ------
/opt/apps/apps1/tmp  allowAllOps_fs           local   guarded    guarded    N/A
/opt/apps/apps1/lib  allowAllRootUsers_fs     local   guarded    guarded    N/A
/opt/apps/apps1/doc  allowAllOps-winusers1_fs local   guarded    guarded    N/A
```

When a manual GuardPoint is applied, regardless if it is a file system directory or a raw device, the change is pushed to the host system only. The host is aware of the GuardPoint but the host does not mount it. This is indicated in the `Type` column of the "`secfsd -status guard`" output. For example, the GuardPoint `/opt/apps/apps2/bin` has been configured with `Directory (Manual Guard)` so the guard type is set to "`manual`".

```
# secfsd -status guard
GuardPoint           Policy                   Type   ConfigState   Status   Reason
----------          ----------                ----   -----------   ------   -----
/opt/apps/apps1/tmp  allowAllOps_fs           local   guarded     guarded   N/A
/opt/apps/apps1/lib  allowAllRootUsers_fs     local   guarded     guarded   N/A
/opt/apps/apps1/doc  allowAllOps-winusers1_fs    local guarded    guarded   N/A
/opt/apps/apps2/bin  HR_policy01              manual unguarded  not guarded Inactive
```

Note the `Type` value. A `Type` of `manual` indicates a manual GuardPoint. A `Type of local` indicates an automatic GuardPoint.

A manually applied GuardPoint retains a yellow triangle status (Pending) in the Management Console until the GuardPoint is applied on the host. After the GuardPoint is applied on the host, and the host communicates the change to the server, the status changes to a green ball (Normal). It returns to the yellow triangle when the GuardPoint is manually unguarded.

Use the `secfsd` command to guard and unguard `Directory (Manual Guard)` and `Raw or Block Device (Manual Guard)` GuardPoints. The `secfsd` syntax is:

```
secfsd -guard <path>
secfsd -unguard <path>
```

> **Note**
> In zone-based CTE Agent deployments, such as Solaris Zones, always specify paths relative to the global zone, never the local zone. Also, you must guard and unguard manual GuardPoints in the global zone.

For example, to manually guard and unguard a file system directory:

1. Configure a GuardPoint with the type **Directory (Manual Guard)**.
2. Log onto the protected host with CTE Agent as the root user.
3. Wait until the configuration change is downloaded to the protected host.

   You can run the status command until you see the manual GuardPoint. For example:

   ```
   # secfsd -status guard
   GuardPoint         Policy           Type     ConfigState  Status       Reason
   ----------         ------           ----     -----------  ------       ------
   /opt/apps/etc      allowAllOps_fs   manual   unguarded    not guarded  N/A
   /opt/apps/lib/dx3  allowAllOps_fs   local    guarded      guarded      N/A
   ```

4. Enable the GuardPoint.

   ```
   # secfsd -guard /opt/apps/apps2/bin
   secfsd: Guard initiated
   ```

   The GuardPoint is active and the policy is enforced.

5. Disable the GuardPoint.

   ```
   # secfsd -unguard /opt/apps/apps2/bin
   secfsd: Unguard initiated
   ```

dataxform Examples and Full Command Syntax

`dataxform` is located at:

- Linux/AIX: `/opt/vormetric/DataSecurityExpert/agent/vmd/bin/dataxform`
- Windows: `C:\Program Files\Vormetric\DataSecurityExpert\agent\vmd\bin\dataxform`

Add the path for the directory that contains `dataxform` to the user `$PATH` environment variable so you can run `dataxform` without having to specify the full path. Also, a symbolic link named `/usr/bin/dataxform` is created during CTE Agent installation and `/usr/bin` is usually defined in the $PATH variable.

dataxform Command Syntax Examples

The following examples show the most common uses of `dataxform`. The full list of options appears in the table below.

Preparation and GuardPoint File Analysis Examples

Check the `dataxform` version

```
dataxform --version
```

Display basic GuardPoint file information

```
dataxform --scan --gp dir_path
```

where *`dir_path`* is the full path to a GuardPoint or an unguarded directory.

Lists the subdirectories in the GuardPoint or unguarded directory, the number of files in the top directory and in each subdirectory, and the total number of files and directories. The disk usage for each directory, and total disk usage, are also listed.

When run on a GuardPoint that has already been transformed but not cleaned, `--scan` also returns the number of hard links and soft links that were skipped in the previous `dataxform` session.

Display file information and create simulation files

```
dataxform --deep_scan --gp dir_path
```

where *`dir_path`* is the full path to a GuardPoint or an unguarded directory.

Lists the subdirectories in the GuardPoint or unguarded directory, the number of files in the top directory and in each subdirectory, and the total number of files and directories. The disk usage for each directory, and total disk usage, are also listed.

In addition, this command creates a set of simulation files of various sizes in the GuardPoint or directory, and uses these to estimate how long it would take to rekey the actual GuardPoint or directory. See also "Estimating the dataxform Runtime Period" on page 40.

Check file links

```
dataxform --check_links --gp guardpoint_path
```

where *`dir_path`* is the full path to a GuardPoint or an unguarded directory.

Lists the subdirectories in the GuardPoint or unguarded directory, the number of files in the top directory and in each subdirectory, and the total number of files and directories. The disk usage for each directory, and total disk usage, are also listed.

In addition, this argument scans for hard links in the GuardPoint or unguarded directory. This operation generates only a list of hard-link files.

Rekeying a GuardPoint Examples

Check to see if a GuardPoint is ready to be rekeyed

```
dataxform --rekey_supported --gp guardpoint_path
```

where *`guardpoint_path`* is the full path to the GuardPoint.

This command makes sure that the GuardPoint is valid, has a rekey policy applied, and is not being accessed by any users or processes.

For example: `dataxform --rekey_supported --gp /opt/apps/dx2`

Rekey All Files in a GuardPoint

```
dataxform --rekey --gp guardpoint_path [--nq]
```

where *`guardpoint_path`* is the full path to the GuardPoint and `--nq` is an optional parameter that tells `dataxform` to run the command silently (without displaying any prompts).

Rekey Specific Files in a GuardPoint

```
dataxform --rekey_list --file_list file --gp guardpoint_path
```

where:

- `--file_list` *`file`* specifies the name of the input file that contains a list of the files that you want `dataxform` to process.
- *`guardpoint_path`* is the full path to the GuardPoint.

Troubleshooting and Maintenance Examples

Clean up a GuardPoint after a previous `dataxform` session

```
dataxform --cleanup [--nq] --gp guardpoint_path
```

where:

- `--nq` is an optional parameter that tells `dataxform` to run the command silently (without displaying any prompts).
- *`guardpoint_path`* is the full path to the GuardPoint.

For example: `dataxform --cleanup --nq --gp /opt/apps/dx2`

Generate the files necessary to complete an interrupted `dataxform` session

```
dataxform --recovery [--file_list file] --gp guardpoint_path [--dir_recovery path]
[-nq]
```

where:

- `--file_list` *`file`* specifies the name of the output file you want `dataxform` to use. dataxform automatically appends `_done` to this file name.
- *`guardpoint_path`* is the full path to the GuardPoint.
- `--dir_recovery` *`path`* specifies the name of a custom output directory into which `dataxform` should write the recovery files. If this option is not specified, dataxform writes the output files to /var/log/vormetric on Linux/AIX or to the standard `logs` directory on Windows.
- `--nq` is an optional parameter that tells `dataxform` to run the command silently (without displaying any prompts).

For example:

```
dataxform --recovery --file_list my-output-file --gp /opt/apps/dx2 \
--dir_recovery /var/custom/output/dir
```

dataxform Command Parameters

Option	Description (Parameter)
`--buf_size`	Sets the size of the kernel buffers that are allocated to run `dataxform`. Buffer sizes range between 4 and 128 KB. The default buffer size is 128 KB. The selected default has been empirically determined to be the safest and most efficient. We strongly recommend that you do not change the default value. Specify an integer between 4 and 128 inclusive.
`--check_links`	Lists the subdirectories in the GuardPoint or regular directory, the number of files in the top directory and in each subdirectory, and the total number of files and directories. The disk usage for each directory, and total disk usage, are also listed. In addition, this argument scans for hard links in the GuardPoint or regular directory. Use the `--gp` option to specify the GuardPoint or regular directory to scan. No files are rekeyed. This operation generates only a list of hard-link files.

Option	Description (Parameter)
`--cleanup`	Deletes the status files that were generated by a previous `dataxform` session and that prevents you from running another `dataxform` session. It removes the `dataxform_auto_lock`, `dataxform files`, and `dataxform_status` files from a GuardPoint. It also deletes the `dataxform_status-`*`gp`* file from `/var/log/vormetric`. You must clean up the GuardPoint before you can run a new `dataxform` session, because if those files are detected by `dataxform`, `dataxform` will abort with the message: "`Automatic data transform status for gp: previous attempt completed.`" Use `--cleanup` to remove these files, or remove them manually, after a `dataxform` session. Include the `--gp` option to specify the GuardPoint. If the `--dir_recovery` argument was used to output the `dataxform_*` files to a different location, be sure to include `--dir_recovery` and the full path to the alternate directory when you run cleanup. If you are running automatic `dataxform`, disable or remove the GuardPoint before using the `--cleanup` argument. Enable or reapply the GuardPoint afterwards. You must do this because once automatic `dataxform` completes it is done. It no longer checks the GuardPoint for a `dataxform_auto_config` file. When the GuardPoint is enabled or reapplied, `dataxform` is reactivated and searches for the file.
`--deep_scan`	Lists the subdirectories in the GuardPoint or unguarded directory, the number of files in the top directory and in each subdirectory, and the total number of files and directories. The disk usage for each directory, and total disk usage, are also listed. In addition, this argument creates a set of simulation files of various sizes in the GuardPoint or directory, and uses these to estimate how long it would take to rekey the actual GuardPoint or directory. See also "Estimating the dataxform Runtime Period" on page 40. Use the `--gp` parameter with this parameter to specify the GuardPoint or directory path.
`--dir_recovery`	Allows you to specify where `dataxform` status files are placed. By default, the status files are placed in one of the following locations: • Without any arguments, the status files are placed in `/var/log/vormetric` • With the `--status_gp` argument, the status files are placed in the GuardPoint. • With the `--dir_recovery` argument, the status files are placed in a user-specified location. The name of the status files created by `--dir_recovery` are `dataxform_status-_`*`gp`* and `dataxform_status-alt_`*`gp`*, where *`gp`* is the underscore-separated path to the GuardPoint. For example, if the path to the GuardPoint is `/home/apps/lib/dx1`, then a status file name would be `dataxform_status-_home_apps_lib_dx1`. The `dataxform` file, `dataxform_auto_lock`, is always written to the GuardPoint.
`encrypt_sparse_file_holes`	Checks for "holes" in sparse files and fills and encrypts them. The `dataxform` utility processes each rekey block in 1k "chunks" (the rekey block is on a 1K boundary and multiples of 1K in size). Each "chunk" consisting of all zeros is considered a "hole". By default, holes on Linux systems are not rekeyed or written, thus keeping the file size small. The default on Linux systems is: `--preserve_sparse_files`.

Option	Description (Parameter)
`--file_list`	Used with `--recovery` to specify the output file name, or with `--rekey_list`, to specify the input file name. When used with `--recovery`, the output file name is automatically appended with "`_done`". `--file_list` takes one argument, *`file`*.
`--gp`	Specifies the full path to the GuardPoint directory to process.
`--mt`	The `--mt` option sets the maximum number of threads allowed to transform one file. Valid values are integers between 1 and 16, inclusive. We recommend that you do not change the default values for `--mt`. To increase `dataxform` performance, you may want to start with the default value and gradually increase the maximum threads value. See also "--thd" on the facing page.
`--nq`	The `--nq` ("no queries") option does rekeying without prompts. Without this option, `dataxform` prompts you to verify that you want to continue with the specified operation.
`--preserve_access_time`	Directs `dataxform` to leave the last-accessed time of files intact during the rekey process. This option is useful when the last-accessed time is used to trigger file backups.
`--preserve_modified_time`	Directs `dataxform` to leave the last-modified time of files intact during the rekey process. This option is useful when the last-modified time is used to trigger file backups.
`--preserve_sparse_files`	Checks for "holes" in sparse files and preserves them in the rekeyed file. The `dataxform` utility processes each rekey block in 1k "chunks" (the rekey block is on a 1K boundary and multiples of 1K in size). Each "chunk" consisting of all zeros is considered a "hole". Holes are not rekeyed nor written, therefore creating a sparse file, which can be considerably smaller in size. Sparse files are preserved by default. This `dataxform` option is provided only for backward compatibility.
`--print_stat`	Displays the time it takes `dataxform` to complete each phase of the transformation process.
`--recovery`	Generates the files needed to complete an interrupted `dataxform` session. The generated files are `dataxform_files_done-`*`_path`* and `dataxform_files_todo-`*`path`*, and they are placed in `/var/log/vormetric` on Linux systems. Use the `--gp` option to specify the GuardPoint.
`--rekey`	The `--rekey` option rekeys all the files in the GuardPoint specified by the `--gp` option.
`--rekey_list`	Same as `--rekey`, except that `dataxform` transforms only the files in the file specified in the `--file_list` option. This option is typically used to recover a failed `dataxform` session. Use the `--gp` option to specify the GuardPoint.
`--rekey_supported`	Checks the specified directory to determine if it is a valid GuardPoint, if a rekey policy is currently applied, and if anyone is currently accessing the directory. The `--gp` option specifies the GuardPoint to check. No files are rekeyed. This operation indicates only if the specified GuardPoint is ready to be rekeyed.

Option	Description (Parameter)
`--scan`	Lists the subdirectories in the GuardPoint or unguarded directory, the number of files in the top directory and in each subdirectory, and the total number of files and directories. The disk usage for each directory, and total disk usage, are also listed. When run on a GuardPoint that has already been transformed but not cleaned, `--scan` also returns the number of hard links and soft links that were skipped in the previous `dataxform` session. No files are rekeyed. Use the `--gp` parameter with this parameter to specify the GuardPoint or directory path.
`--status_gp`	Causes `dataxform` to put the status, run, and error files in the GuardPoint rather than in the log directory. See also "Using dataxform_status* Files" on page 47.
`--status_interval`	Sets the time interval (in seconds) at which data transformation status messages are sent to the key manager. The default is `300` seconds (5 minutes).
`--thd`	Sets the number of threads that `dataxform` can use to transform files. A “thread” equates to a file. The more threads specified, the more files that are transformed concurrently. You may process up to 32 files concurrently. Threads are numbered 0 through 31, but the values you enter are 1 through 32. 0 indicates all 32 threads. The default number of threads is 8 or the number of CPUs, whichever is less. If messages like “access denied” are displayed or files are being skipped, try reducing the number of threads. If the errors still occur after the number of threads is set to 1, the errors are not due to `dataxform` processes colliding. Most likely there is something wrong with the files or policy permissions. The default value has been empirically determined to be the safest and most efficient. We strongly recommend that you do not increase the default value. See also "--mt" on the previous page.
`--version`	Displays `dataxform` version information.

Unencrypting Data

You can use either of the following methods to unencrypt your data. The first requires you to copy the files to a temporary directory and the second requires you to use dataxform on the host system.

Copy Method

1. Stop all applications accessing the GuardPoint.
2. Log on as a user who can see clear text for all files in the GuardPoint.
3. Copy all files from the GuardPoint to a new directory that is not guarded.
4. In your key manager, unguard the GuardPoint. Make sure that it does not display in the key manager.
5. Delete all files in the original directory.
6. Copy all files from the unguarded/clear text directory to the original directory.
7. Start any application once all files have finished copying.

> **Note**
> Do not rename directories. This will not result in unencrypting directories.

Dataxform Method

1. Stop any application accessing the GuardPoint.
2. In your key manager, clone the original policy that you used to encrypt the data.
3. In the cloned policy, reverse the keys. Put the original key in for the key_selection rule and the clear_key in for the transformation rule
4. Guard the directory you want to unencrypt with the cloned policy.

   > **Note:** Make sure the status is green before you guard with that policy.

5. On the host system, run `dataxform --rekey` on the GuardPoint:

   ```
   # dataxform --rekey --gp /<dirName> --preserve_modified_time
   ```

   For example:

   ```
   # dataxform --rekey --gp /DataSecurity/mydir --preserve_modified_time
   ```

6. Run `dataxform --cleanup` on the GuardPoint:

   ```
   # dataxform --cleanup --gp /<dirName>
   ```

   For example:

   ```
   # dataxform --cleanup --gp /DataSecurity/myDir
   ```

7. In your key manager, unguard the Guard Point.

   The data should now be in clear text (unencrypted).

THALES

Contact us
For office locations and contact information,
visit cpl.thalesgroup.com/contact-us

> cpl.thalesgroup.com <