Zettaset

Zettaset XCrypt Full Disk™ v8.5.2

Deployment and Maintenance Guide

Table of Contents

Overview

Zettaset XCrypt Full Disk is a partition-level encryption solution that delivers the security of the military-grade AES 256-bit encryption algorithm while yielding the high-performance ideal for bulk encryption and distributed environments.

XCrypt Full Disk encrypts entire partitions under the UNIX file system layer. When a partition is unlocked (by authenticating to a key server and retrieving the key) the file system is mounted and becomes available. All users with sufficient UNIX file system permissions can read and write the plaintext. Those without permissions cannot access the decrypted data.

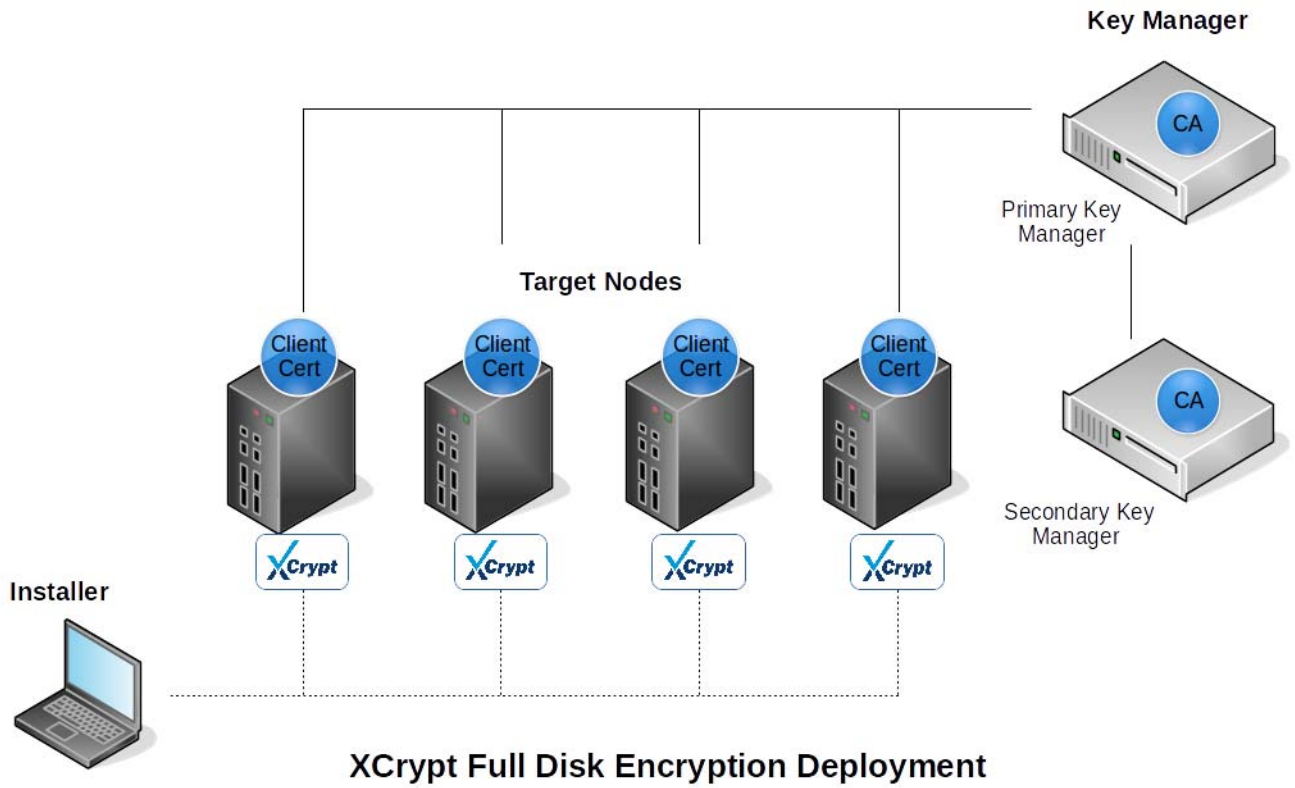
Installation is performed from the command line.

The deployment relies on three types of entities:

Installer - This is the device used to launch the initial Zettaset software installation. This node can be a target node, or a separate device with access to the target nodes. It must have the Zettaset software and license files, ansible, and the client and CA certificates needed to communicate with any 3rd-party Key Management device used. (No certificates are needed when using Zettaset's own Key Manager.) After the initial installation, the installer can be used to add new nodes, but it doesn't have any more managerial function.

Target Nodes - These are the nodes that contain the partitions to be encrypted. After the Zettaset installation, each node will contain the client and CA certificates needed to communicate with the Key Manager. Key rotation, decryption, and encryption of new partitions are done directly on the target nodes.

Key Manager - This is the secure device used to store keys for the encrypted nodes. It also contains the CA used for secure communication with the target nodes. You can use a 3rd-party Key Management device, or use Zettaset's software-based key server, which can be installed anywhere in your cluster. The 3rd-party Key Manager must be KMIP compliant.



XCrypt Full Disk Encryption Deployment

Part I: Deployment

This part contains information on the deployment of your Zettaset software and contains the following sections:

- Section 1: Deployment Prerequisites
- Section 2: Installing Pre-Requisites when Offline
- Section 3: Configuring the Inventory File
- Section 4: Manually Zeroizing Old Data
- Section 5: Installing Zettaset XCrypt Full Disk

Section 1: Deployment Prerequisites

Perform the following steps **on each target node** in your deployment:

1 Confirm that the operating system is either CentOS or RHEL 6.6 - 7.5 by viewing `/etc/redhat-release`:

```
$ cat /etc/issue.net
CentOS Linux release 7.2.1511 (Core)
```

2 Confirm that java 1.7 or higher is installed:

```
$ java -version
java version "1.7.0"
```

3 Install libselinux-python, 2.0.94 or higher.

```
$ yum install libselinux-python -y
```

4 Install cryptsetup if OS is CentOS or RHEL 6.x:

```
$ yum install cryptsetup-luks -y
```

5 Confirm that wget, 1.12 or higher, is installed:

```
$ wget --version
```

Install if needed:

```
$ yum install wget -y
```

6 Confirm that netstat is installed:

```
$ netstat --version
```

Install if needed:

```
$ yum install netstat -y
```

7 Update nss. nss must be version 3.21 or greater.

```
$ yum update nss -y
```

8 If encrypting an xfs file system, xfsprogs and xfsdump libraries must be installed on the node running xfs. The xfs partitions must be unmounted before installing Zettaset XCrypt Full Disk.

9 Open the ports used by your Key Manager. For example, when using the Zettaset software-based Key Manager open ports 6666 and 8789:

When using iptables:

```
$ iptables -I INPUT -p tcp --dport 6666 --syn -j ACCEPT
$ iptables -I INPUT -p tcp --dport 8789 --syn -j ACCEPT
$ service iptables save
$ service iptables restart
$ iptables -L -n # confirm
```

When using firewalld:

```
$ firewall-cmd --get-active-zones # use the active zone
$ firewall-cmd --zone=public --add-port=6666/tcp --permanent
$ firewall-cmd --zone=public --add-port=8789/tcp --permanent
$ firewall-cmd --reload
$ firewall-cmd --list-all # confirm
```

If using an external, third-party Key Manager, ensure that the necessary ports are open in your cluster.

10 When enabling KMIP HA on CentOS or RHEL 7.x, open ports 2181, 2888, and 3888 on the [zookeeper] nodes to establish communication between those devices. For example, if using firewalld:

```
$ firewall-cmd --zone=public --add-port=2181/tcp --permanent
$ firewall-cmd --zone=public --add-port=2888/tcp --permanent
$ firewall-cmd --zone=public --add-port=3888/tcp --permanent
$ firewall-cmd --reload
$ firewall-cmd --list-all # confirm
```

Then open port 24007 and one port per [kmip] node starting from 49152 on the [kmip] nodes.

```
$ firewall-cmd --zone=public --add-port=24007/tcp --permanent
$ firewall-cmd --zone=public --add-port=29152-29154/tcp --permanent
$ firewall-cmd --reload
```

11 Install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files:

a Download the file from

www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

or

www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

b Extract the jar files and install them in $JAVA_HOME/lib/security.

12 FIPS mode is only supported in CentOS or RHEL 7.x. If you are setting fips_mode to true:

a Confirm that the FIPS version openssl installed on all nodes is at least version 1.0.1e-fips.

b Install the Bouncy Castle Java Crypto Provider statically on all nodes.

13 A License Server port must be open, the default is 21800. To change the default value edit the following files:

- /usr/share/zts/config/license-config.xml (on the License Server nodes)
- /etc/zts/conf.default/license-server_ssl.xml (on the slave nodes)

Perform the following steps **on the installer node**, referred to as installer01 in the code samples.

1 Establish ssh trust between the installer node and all target nodes. This prevents errors when running ssh commands. To create ssh trust:

a Generate an ssh key for the installer, if not already present:

```
$ ssh-keygen
```

b Distribute the key to each target node:

```
$ ssh-copy-id target01
$ ssh-copy-id target02
```

2 Install ansible (any version between 1.7.2 and 2.4.2.0) on the installer node:

```
$ yum install markupsafe epel-release gcc python-pip -y
$ easy_install pip==1.5.6
```

```
$ pip install paramiko PyYAML jinja2 httplib2
$ pip install ansible==2.3.0
```

3 Install the Zettaset archive and license files:

```
$ scp -P 22 zts-xcrypt-full-disk-8.5.2.tar.gz root@installer01:/opt
$ scp -P 22 sample.license root@installer01:/opt
```

4 Extract the archive:

```
$ ssh installer01
$ cd /opt
$ tar zxvf zts-xcrypt-full-disk-8.5.2.tar.gz
```

5 Copy `hosts.inv.example` to `hosts.inv`.

```
$ cd /opt/zettaset/xcrypt-full-disk/8.5.2
$ cp hosts.inv.example hosts.inv
```

Section 2: Installing Pre-Requisites when Offline

When deploying Zettaset XCrypt Full Disk to a cluster that does not have access to the internet or a central package repository, use the Zettaset pre-installer to install the required rpms.

To use the pre-installer:

1 Copy the tar.gz file to all of the nodes on which the Zettaset software will be deployed and on the node that will serve as the Zettaset XCrypt Full Disk installer node.

2 Extract the archive file on each node:

```
$ tar -xvf zts-offline-preinstall.tar.gz
```

3 Prepare the installer node by executing the following command:

```
$ ./preinstall.py ansible.lst
```

This statement will install the rpms needed to run the Zettaset XCrypt Full Disk installation.

4 Prepare the nodes in the Zettaset deployment by executing the following command on each node:

```
$ ./preinstall.py deps.lst
```

This statement will install the rpms required by the Zettaset deployment.

You can then continue with the Zettaset XCrypt Full Disk installation.

Section 3: Configuring the Inventory File

The inventory file (`hosts.inv`) sets the configuration properties used for the installation. An annotated sample file (`hosts.inv.example`) is included with the Zettaset software. Additional information is provided here.

```
# ALL NODES
target01 encrypted_blockdev=/dev/sdb1 encrypted_mountpoint=/data1
encrypted_mountnames=crypt1 encrypted_preserve=n fstype=ext4 newfsargs=none
mountargs=none

target02 encrypted_blockdev=/dev/sdb1 encrypted_mountpoint=/data1
encrypted_mountnames=crypt1 encrypted_preserve=n fstype=ext4 newfsargs=none
mountargs=none

target03 encrypted_blockdev=/dev/sdb1 encrypted_mountpoint=/data1
encrypted_mountnames=crypt1 encrypted_preserve=n fstype=ext4 newfsargs=none
mountargs=none
```

In the `ALL NODES` section, list each node in your deployment using hostnames or IP addresses. For each node, include the following variables:

`encrypted_blockdev` - Enter the block device to be encrypted (`/dev/sdb1`). Disk partition name is expected. To use disk partition labels, set `use_labels=true`.

`encrypted_mountpoint` - Enter a mount point for the device (`/data1`). The mount point must exist before the install.

`encrypted_mountnames` - Enter a partition name. Each name must be unique for each partition on the node. (`crypt1`)

`encrypted_preserve` - Use one of the following values:

> `y` - Preserves existing data. The file system must be mounted before the install. If the partition is not mounted, the data will be overwritten. The partition must also be unmountable. If a process prevents the unmount, encryption cannot start. Only ext file systems can be preserved.
>
> `n` - Does not preserve existing data. The partition must be unmounted.
>
> `w` - Securely wipes the partition before the new encrypted file system is created. The partition must be unmounted.

`fstype` - Must be set to the type of file system to make when `encrypted_preserve` is `y` or `w`. Must be set to the existing file system type when `encrypted_preserve=n`. Typical file system types include ext4 and xfs.

`newfsargs` - A string of arguments to pass to the mkfs command. If there are spaces between multiple arguments, surround the string in double quotes (i.e., `"-b 2048 -d su=64k,sw=4"`). When no arguments are to be passed, set this value to `none`.

`mountargs` - A string of mount options to pass to the `crypt_mount.sh` script. If there are spaces between multiple arguments, surround the string in double quotes (i.e., `"noatime,inode64,allocsize=16m"`). When no arguments are to be passed, set this value to `none`.

`kmip_client_jks` - The location of the keystore that contains the client certificate. The keystore must be in this location on the installer node prior to installation.

`kmip_client_jks_password` - The password for the jks file.

When encrypting multiple partitions on a node, use commas to separate values. For `newfsargs`, use colons to separate values. Include values for all settings. For example:

```
target03 encrypted_blockdev=/dev/sdb1,/dev/sdb2 encrypted_mountpoint=/data1,/
data2 encrypted_mountnames=crypt1,crypt2 encrypted_preserve=n,n fstype=ext4,xfs
newfsargs=none:"-b size=2048 -d su=64k,sw=4" mountargs=none:"noatime,inode64"
```

The hostnames provided above must resolve. If some nodes are separated by a proxy (like if you are deploying to nodes in skytap from your laptop) use the `ansible_ssh_host` and `ansible_ssh_port` variables. Otherwise, do not use those variables.

```
# PRODUCT NAME
zts_product=xcrypt_full_disk
```

Displays the product name.

```
# SOFTWARE LICENSE
license_file=/path/to/your.license
```

Include the full path to the license file.

```
# FIPS MODE
fips_mode=false
```

Set to true to enable FIPS 140 mode. All ZTS processes will run in FIPS mode. `fips_mode` set to true is currently only supported for OS versions 7.x.

```
# DISK LABELS
use_labels=false
```

By default, the Zettaset software expects the `encrypted_blockdev` value used above to point to a disk partition, such as /dev/sdb. To use disk partition labels instead, set `use_labels=true`.

```
# CA CONFIGURATION
internal_ca=false
external_ca_cert_source=
ca_org_name=
ca_org_unit=
ca_org_email=
ca_org_locale=
ca_org_state=
ca_org_country=
```

A CA is required to authenticate nodes within your deployment. To use your pre-defined CA, set `internal_ca=false` and enter the full path to the CA PEM file in `external_ca_cert_source`. This is the location of the CA PEM file on the installer node.

While using an external CA, the `ca_org_*` values can be ignored.

```
# KMIP SERVER CONFIGURATION
internal_kmip=false
kmip_master_ip=172.24.166.20
kmip_server_port=9000
```

```
kmip_client_timeout=300000
kmip_compatible_user=true
install_ha=false
kmip_client_jks_test=
kmip_client_jks_test_passwd=
```

A KMIP server is required to process key requests. To use an external KMIP server, set `internal_kmip=false`, and set the `kmip_master_ip` and `kmip_master_port` to point to your third-party device.

Use `kmip_client_timeout` to configure the timeout setting, or just keep the default value of 300000.

When using an external KMIP server, use `kmip_client_jks_test` and `kmip_client_jks_passwd` to enter the jks path and password and check the KMIP server connectivity prior to installing XCrypt. Using these values will install a KMIP client on the installation node. Leave these values blank if you do not need to check external KMIP connectivity or install a KMIP client on the installation node.

```
# HSM CONFIGURATION
hsm_so_pin=12345678
hsm_user_pin=12345678
hsm_slot=1
hsm_lib_cfg_env_var=
hsm_lib_file=
```

An HSM is required for key storage. To create and use an internal, software-based HSM, select 4-8 alphanumeric characters for both `hsm_so_pin` and `hsm_user_pin`. Key `hsm_slot` pointed to slot 1.

To use a third-party HSM, use the security office and user pin values for `hsm_so_pin` and `hsm_user_pin`. Point `hsm_slot` to your device's slot number. Use `hsm_lib_cfg_env_var` to configure any environment variables required by your HSM, for example: `hsm_lib_cfg_env_var=env_var_name=value`. Use `hsm_lib_file` to point to the location of your pkcs11 module.

NOTE: When using a third party HSM, remember to secure the PINs in this file after the installation.

```
# NODE FUNCTIONS
[ca_master]
target01
[kmip]
target01
target02
[kmip_master]
target01
[slave]
target01
target02
target03
[license_server]
target04
[zookeeper]
target01
target02
target03
```

The bracketed values indicate the function a node will have in the deployment. Be sure that these settings agree with the other values in this file.

`[ca_master]` - The node used to store licenses and generate the CA. If using an external CA, set this value to a node within the cluster.

`[kmip]` - List of the KMIP server and backup server nodes. The first entry must be the `kmip_master`. Ignore when using an external KMIP server.

`[kmip_master]` - The KMIP master node. Must be the same as `kmip_master`_ip. Ignore when using an external KMIP server.

`[slave]` - List of the nodes that will have encrypted partitions.

`[license_server]` - List of the nodes where the License server will be installed. Must not intersect with `[kmip]` or `[slave]` nodes.

`[zookeeper]` - List of the zookeeper nodes used when KMIP HA is enabled. List at least three nodes. These nodes cannot be members of the `[kmip]` group.

Section 4: Manually Zeroizing Old Data

For each partition where you are not preserving existing data (where `encrypted_preserve=n`) and which previously held sensitive data, the partition should be wiped before being encrypted. Otherwise, some blocks may not be overwritten with encrypted data right away and could be recovered.

Use the umount and wipefs commands to unmount and zeroize.

```
$ umount <partition name>
$ wipefs <partition name>
$ cat /dev/zero > <partition name>
```

For example, to zeroize partition /dev/sbd:

```
$ umount /dev/sdb
$ wipefs /dev/sdb
$ cat /dev/zero > /dev/sdb
```

Section 5: Installing Zettaset XCrypt Full Disk

The installer sends the Zettaset XCrypt Full Disk libraries and configuration files to each target node. It also encrypts the nodes and partitions listed in the `hosts.inv` file.

1. Perform a sanity check on the inventory file. This will confirm that the settings in your file are valid.

   ```
   $ ./install_zts-xcrypt-full-disk.sh -vv -i hosts.inv check
   ```

2. Run the installer:

   ```
   $ ./install_zts-xcrypt-full-disk.sh -vv -i hosts.inv install
   ```

   This will create any KMIP and HSM servers needed, establish secure connectivity between all nodes and services, and encrypt partitions.

3. View each target node's block devices to confirm partition encryption. Your output will reflect your partitions.

   ```
   $ ssh target03 "lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT"
   NAME                    FSTYPE          SIZE  MOUNTPOINT
   └─sdc                   crypto_LUKS     5G
     └─crypt2 (dm-2)       xfs             5G    /data2
   ```

4. View the encryption key names by viewing `cryptinittab` on the target node:

   ```
   $ ssh target03 "cat /etc/zts/conf.default/cryptinittab"
   partition   mount point    mapper name  key name
   ```

```
/dev/sdc     /data2          crypt2        688eda48-337f-49fd
```

5 Backup the `cryptinittab` file for each encrypted node: this file is the only way to associate a key with a partition.

6 Review `install.log` when needed.

7 Remove any hsm pin values from the `hosts.inv` file.

Part II: Maintenance

This part contains information on the maintenance of your Zettaset deployment and contains the following sections:

- Section 1: Encrypting Additional Nodes
- Section 2: Maintaining the List of Licensed Nodes
- Section 3: Encrypting Additional Partitions
- Section 4: Mounting and Unmounting Encrypted Partitions
- Section 5: Key Rotation
- Section 6: Restoring Data/Disaster Recovery
- Section 7: Removing Encryption
- Section 8: Removing Non-Functioning Encrypted Partitions
- Section 10: Uninstalling Zettaset XCrypt Full Disk

Section 1: Encrypting Additional Nodes

After installing Zettaset XCrypt Full Disk, you may want to encrypt additional nodes. To encrypt additional partitions on an existing node, see Section 3: Encrypting Additional Partitions, below.

To encrypt additional nodes:

1 Edit `hosts.inv` on the installer node:
   a Add the new nodes to the `ALL NODES` section. Be sure to leave the old entries. Use the `encrypted_blockdev`, `encrypted_mountpoint`, `encrypted_mountnames`, `encrypted_preserve`, `fstype`, `newfsargs`, `mountargs`, `kmip_client_jks`, and `kmip_client_jks_password` variables to configure the encryption. Use the other variables as needed.
   b Comment out the old nodes in the `[slave]` section.
   c Add the new nodes to the `[slave]` section. Use either the hostnames or IP addresses.
   d Be sure that the hsm pin values are present.
2 Be sure that the installation node's ssh key has been distributed to the new node:
   ```
   $ ssh-copy-id target05
   ```
3 Run the installation, using '`-t add`':
   ```
   $ ./install_zts-xcrypt-full-disk.sh -vv -t add -i hosts.inv install
   ```
4 Backup the `cryptinittab` file for each encrypted node: this file is the only way to associate a key with a partition.

Section 2: Maintaining the List of Licensed Nodes

XCrypt maintains a list of licensed nodes for your deployment. When a new node is encrypted, it's automatically added to this list, provided there are licenses available. You can add, delete, and replace nodes from this list. You can also view the list itself. This management is done on your license server node.

To add a node:

```
$ /usr/share/zts/bin/edit_nodes.sh -a <new node>
```

New entries are validated against the list of nodes contained in the `hosts.inv` file, which is used during installation. If the user needs to add a new node not contained in `hosts.inv`, use `-f` to skip validation.

To delete a node:

```
$ /usr/share/zts/bin/edit_nodes.sh -d <existing node>
```

To replace a node:

```
$ /usr/share/zts/bin/edit_nodes.sh -r <existing node> <new node>
```

To list nodes:

```
$ /usr/share/zts/bin/edit_nodes.sh -l
```

Section 3: Encrypting Additional Partitions

After installing Zettaset XCrypt Full Disk on a node, you may want to encrypt additional partitions. Do this on the target node itself, there is no need to re-run the installer.

To encrypt additional partitions:

1. Create a new file system on the partition:

   ```
   $ mkfs.xfs /dev/sdf
   ```

2. Create the mount point:

   ```
   $ mkdir -p /data5
   ```

3. Mount the partition, if necessary (do not mount xfs partitions):

   ```
   $ mount /dev/sdf /data5
   ```

4. Use `crypt_setup.sh` to encrypt the partition:

   ```
   $ /usr/share/zts/bin/crypt_setup.sh -fstype <fs> -newfsargs <args> y|n|w      \
   <partition> <mapper> <mount point>
   ```

   where
   `y` = preserve the existing data
   `n` = do not preserve the existing data
   `w` = wipe the existing data

   For example,

   ```
   $ /crypt_setup.sh -fstype xfs -newfsargs none n /dev/sdf crypt5 /data5
   ```

5. Use crypt_mount.sh to mount partition:

   ```
   $ /usr/share/zts/bin/crypt_mount.sh mount
   ```

   NOTE: When mounting a partition that uses mount options, add those options to the partition's entry in `/etc/fstab` before mounting. For example, you would include the following line to `/etc/fstab`:

   ```
   <file system>      <mount point>       <type> <options>       <dump> <pass>
   /dev/mapper/crypt1 /var/lib/zts/slave1 xfs    noatime,inode64 0      2
   ```

6. View the block devices to confirm the partition encryption:

   ```
   $ ssh target01 "lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT"
   NAME                    FSTYPE       SIZE MOUNTPOINT
   ├─sdf                   crypto_LUKS  204M
   │ └─crypt5 (dm-5)       xfs          202M /data5
   ```

7. Backup the new key(s) on your key manager.

8. Backup the `cryptinittab` file for the encrypted node: this file is the only way to associate a key with a partition.

Section 4: Mounting and Unmounting Encrypted Partitions

To mount all encrypted partitions on a node:

1 Call crypt_mount.sh:

```
$ /usr/share/zts/bin/crypt_mount.sh mount
```

NOTE: When mounting a partition that uses mount options, add those options to the partition's entry in `/etc/fstab` before mounting. For example, you would include the following line to `/etc/fstab`:

```
<file system>      <mount point>       <type> <options>       <dump> <pass>
/dev/mapper/crypt1 /var/lib/zts/slave1 xfs    noatime,inode64 0      2
```

To unmount all encrypted partitions on a node:

1 Call crypt_mount.sh:

```
$ /usr/share/zts/bin/crypt_mount.sh unmount
```

Section 5: Key Rotation

You should periodically change encryption keys, in accordance with your organization's security policy. Key rotation can be done while the encrypted partitions are in use. Repeat this process for each node in your deployment.

1 View the current keys by viewing cryptinittab on the target node:

```
$ cat /etc/zts/conf.default/cryptinittab
partition   mount point   mapper name   key name
/dev/sdb    /data1        crypt1        8697-697e-46a7
/dev/sdc    /data2        crypt2        7efc-6dac-489d
/dev/sdd    /data3        crypt3        db85-1a74-83c6
/dev/sde    /data4        crypt4        a032-c464-14a7
```

2 Use crypt_rotate.sh to create and apply a new key:

```
$ /usr/share/zts/bin/crypt_rotate.sh
```

New keys are generated and applied to each encrypted partition on the host. The old keys are deleted from the key server.

3 View the new key names in cryptinittab:

```
$ cat /etc/zts/conf.default/cryptinittab
partition   mount point   mapper name   key name
/dev/sdb    /data1        crypt1        0410-9c1e-ee96
/dev/sdc    /data2        crypt2        47ba-b3c5-f096
/dev/sdd    /data3        crypt3        c100-96c2-7af8
/dev/sde    /data4        crypt4        10cd-0b0e-470f
```

Section 6: Restoring Data/Disaster Recovery

In the event that you need to restore plaintext data from a backup device to an encrypted partition as part of a disaster recovery scenario, you can perform the procedures outlined in Section 1: Encrypting Additional Nodes or Section 3: Encrypting Additional Partitions. Once completed, you can restore the data to the new partition. New keys will be used to encrypt the partition(s).

Section 7: Removing Encryption

To remove encryption from a partition, first back up the data while the partition is mounted. Then unmount the partition, remove encryption, and restore the backup.

1 Read the list of encrypted partitions:

```
$ cat /etc/zts/conf.default/cryptinittab
partition    mount point    mapper name    key name
/dev/sdb     /data1         crypt1         8697-697e-46a7
/dev/sdc     /data2         crypt2         7efc-6dac-489d
/dev/sdd     /data3         crypt3         db85-1a74-83c6
/dev/sde     /data4         crypt4         a032-c464-14a7
```

2 Back up the data.

3 Unmount the file system:

```
$ umount /data1
```

4 Unmap the encrypted partition:

```
$ cryptsetup remove crypt1
```

5 Edit `/etc/zts/conf.default/cryptinittab` and remove the partition entry.

6 Make a new file system on the partition and mount it:

```
$ mkfs.xfs /dev/sdb
```

```
$ mount /dev/sdb /data1
```

7 Copy your backup to the mount location.

Section 8: Removing Non-Functioning Encrypted Partitions

In the event that a partition is no longer functioning, because of a disk failure, for example, you can remove the encrypted partition from the Zettaset XCrypt Full Disk configuration on that node. Once removed, the partition will no longer be mounted. These instructions assume that the data on the partition is no longer accessible.

To remove the partition:

1 Establish an ssh connection to the node with the failed partition.

2 Stop the encryption service:

```
$ service zts-encrypt stop
or
$ systemctl stop zts-encrypt.service
```

3 Make a backup copy of the cryptinittab file:

```
$ cp /etc/zts/conf.default/cryptinittab /etc/zts/conf.default/cryptinittab.bkp
```

4 Remove the partition from the current cryptinittab file:

```
$ vi /etc/zts/conf.default/cryptinittab
partition    mount point    mapper name    key name
/dev/sdb     /data1         crypt1         8697-697e-46a7
/dev/sdc     /data2         crypt2         7efc-6dac-489d
/dev/sdd     /data3         crypt3         db85-1a74-83c6
/dev/sde     /data4         crypt4         a032-c464-14a7
```

```
Start the encryption service:
```

```
$ service zts-encrypt start
or
$ systemctl start zts-encrypt.service
```

5 Follow the directions in Section 3: Encrypting Additional Partitions when encrypting the replacement partitions.

Section 9: Recovering from Disk Failure within a RAID 5 Array

When deploying Zettaset on a disk within a RAID 5 array, no additional action is required in the event of a disk failure. The RAID recovery process will restore the encrypted contents of the failed disk.

Section 10: Uninstalling Zettaset XCrypt Full Disk

IMPORTANT! Back up all encrypted data prior to uninstalling. The data cannot be retrieved once the uninstall is complete.

1 Stop the Zettaset services on each node:

```
$ service zts-kmip stop
$ service pkcsslotd stop
$ service crlserver stop
$ service zts-encrypt stop
```

Use the `systemctl` commands when needed.

Some services, like zts-kmip, are not installed on all nodes. In these cases, attempting to stop the service has no ill effect. Confirm that the services have been stopped, or do not exist:

```
$ service zts-encrypt status
```

2 On each encrypted node:

a Read the list of encrypted partitions:

```
$ cat /etc/zts/conf.default/cryptinittab
partition   mount point   mapper name   key name
/dev/sdb    /data1        crypt1        8697-697e-46a7
/dev/sdc    /data2        crypt2        7efc-6dac-489d
/dev/sdd    /data3        crypt3        db85-1a74-83c6
/dev/sde    /data4        crypt4        a032-c464-14a7
```

b Back up the data.

c Unmount the encrypted partitions:

```
$ umount /data1
$ umount /data2
$ umount /data3
$ umount /data4
```

d Unmap the encrypted partitions:

```
$ cryptsetup luksClose crypt1
$ cryptsetup luksClose crypt2
$ cryptsetup luksClose crypt3
$ cryptsetup luksClose crypt4
```

e Reset the partition:

```
$ dd if=/dev/zero of=/dev/sdb count=2000
$ dd if=/dev/zero of=/dev/sdc count=2000
$ dd if=/dev/zero of=/dev/sdd count=2000
$ dd if=/dev/zero of=/dev/sde count=2000
```

f Confirm decryption on each partition:

```
$ cryptsetup isLuks /dev/sdb
Device /dev/sdb is not a valid LUKS device.
```

g Remove the entries from `/etc/zts/conf.default/cryptinittab`.

3 Remove the following rpms from all of the nodes in your cluster using the commands below, in order:

```
$ rpm -e zts-kmip-client
$ rpm -e zts-kmip-server
$ rpm -e luksipc
$ rpm -e opencryptoki
$ rpm -e opensc
$ rpm -e pkcs11Engine
$ rpm -e pkcs11wrapper
$ rpm -e zts-ca
$ rpm -e zts-xcrypt-full-disk
$ rpm -e zts-license
$ rpm -e zts-logback-xcrypt
$ rpm -e zts-setup
```

Confirm that the rpms have been removed:

```
$ rpm -q <rpm-name>
```

NOTE: `zts-kmip-server` and `zts-license` are not present on each node, but attempting to remove them has no ill effect. Also, avoid including the version number in the `rpm -e` command, as this fails.

4 Remove the Zettaset directories from each node in the cluster:

```
$ rm -rf /usr/share/zts
$ rm -rf /etc/zts
$ rm -rf /var/log/zts
```

5 Remove the `zettaset` directory from the installer node. Backup your hosts.inv file and save the software archive for future installations.

```
$ cp zettaset/8.5.2/hosts.inv /tmp
$ cp zts-xcrypt-full-disk-8.5.2.tar.gz /tmp
$ rm -rf <your install directory>/zettaset/
```

Part III: Troubleshooting

This part contains troubleshooting information you may need for your Zettaset deployment and contains the following sections:

- Section 1: Log Files
- Section 2: Configuration Files
- Section 3: Error Messages

Section 1: Log Files

The log files used by Zettaset XCrypt Full Disk are listed below.

Log (/var/log/zts)	Description
arcsight-root-hostname.out	Full system log for XCrypt Full Disk, with entries in Common Event Format (CEF).
crlserver.log	Record of certificate revocation list (CRL) server activity.
derby.log	Derby database log, used by the KMIP server.
kmip.log	Log of KMIP server interaction.
luks_mount.log, luks_rotate.log, luks_setup.log	Log of luks activity.

Section 2: Configuration Files

Should you need to change or view the configuration, the configuration files are shown below.

File	Description
/etc/zts/conf.default/kmip-server.properties	Configuration used by KMIP server. Only present on the KMIP server nodes identified in hosts.inv.
/etc/zts/conf.default/kmip-client.properties	Configuration used by KMIP clients.
hosts.inv	Configuration used for deployment, located in your install directory.

Section 3: Error Messages

Listed below are the possible error messages with notes on their resolution.

Error	Cause	Resolution
<partition> already in use as an encrypted filesystem	This partition has already been encrypted.	No need to re-apply encryption. Remove, or comment out, the encrypted partition entry from hosts.inv
can not unmount	Could not unmount the partition.	Use lsof and fuser to check the system usage and diagnose the problem.
cannot find enabled slots for <partition>	Could not access the partition slots.	Verify the encrypted_blockdev value in hosts.inv.

Error	Cause	Resolution
Cannot launch KmipServer, masterMoveDB Failed	Error moving the temp db to the keys.db.	Check the temp db and keys.db paths in kmip-server.properties. Be sure that both paths are accessible.
cryptinittab can not be read	Could not access cryptinittab.	Verify that /etc/zts/conf.default/cryptinittab is accessible.
DB close failed	Couldn't close the database.	Verify that the kmip_client_timeout value in /etc/zts/conf.default/kmip-client.properties is appropriate for your network.
db connect failed	Could not connect to the key database.	Check the location of the keys.db in /etc/zts/conf.default/kmip-server.properties.
device <partition> is too full to be resized	The partition does not have enough space to apply encryption to the existing data.	Free up space on the partition before continuing.
e2fsck of <partition> failed	Could not complete the file system check.	Verify that the partition is not mounted.
error calculating file system available space of N	Failed to calculate the size of partition.	Verify connectivity between the installer node and the target server. Check that the partition is mounted, if required by your hosts.inv settings, otherwise check that it is mountable.
error calculating file system size of <partition>	Failed to calculate the size of partition.	Verify connectivity between the installer node and the target server. Check that the partition is mounted, if required by your hosts.inv settings, otherwise check that it is mountable.
error encrypting <partition>-data preservation requested by partition is not mounted. Data is not encrypted.	The partition for which you are requesting data preservation is not mounted.	Mount the partition before encrypting.
error: unknown OS	The target OS is not supported.	Use targets that are either linux or freebsd.
failed to close database connection	Couldn't close the database connection.	Verify that the kmip_client_timeout value in /etc/zts/conf.default/kmip-client.properties is appropriate for your network.
failed to config jbdc	Could not connect to the key database.	Check the location of the keys.db in /etc/zts/conf.default/kmip-server.properties.
failed to connect to the pkcs11 module	Could not connect to the pkcs11 module.	Check the PKCS11 wrapped lib and PKCS11 lib locations in kmip-server.properties. Verify that the library exists and the path is accessible.

Error	Cause	Resolution
failed to initialize pkcs11	Could not initialize the pkcs11 library.	Check the PKCS11 wrapped lib and PKCS11 lib locations in kmip-server.properties. Verify that the library exists and the path is accessible.
failed to start the kmip server	KMIP server either failed to start, or the KMIP could not connect.	Review the configuration of /usr/zts/conf.default/kmip-server.properties, and /usr/zts/conf.default/kmip-client.properties.
HSM init failed	Could not initialize the pkcs11 library.	Verify that the pkcs11 values in /etc/zts/conf.default/kmip-server.properties are correct.
key from server does not match key we sent	The key returned from the KMIP server does not match the key sent to the server.	Check that the key name values are not changed by the KMIP server.
key get failed	KMIP client was not able the get the key from server.	Check the KMIP connectivity and be sure that the key name is correct.
<keyname> set failed with error	The KMIP was not able to set the key name.	The error message will contain additional information. Check the KMIP connectivity and be sure that the key name is unique.
KmipClient.initSSL() failed with Exception	Failed to establish SSH connectivity between the KMIP client and server.	Check the your KMIP Client and Server have common protocols and ciphersuites to use to establish SSH connectivity.
KmipClient.initSSL() failed with IOException	Failed to establish SSH connectivity between the KMIP client and server.	Verify that the KmipPort and KmipServer settings in the kmip-client.properties files are correct. Verify that the port setting in kmip-server.properties is correct.
KmipServer failed to launch	Could not load the kmip server properties file.	Verify the configuration in /etc/zts/conf.default/kmip-server.properties.
KmipServer logger configuration caught JoranException	There was a logging error.	Check the configuration of /etc/zts/conf.default/logback.xcrypt.xml.
KmipServer.initSSL(): SSL initialization failed	SSL communication between the KMIP server and client failed to initialize.	Check the KmipServer and KmipPort values in kmip-client.properties. Check the port setting in kmip-server.properties. Check that the KMIP server and client have common protocols and ciphersuites.
Luks open timed out, exiting	Could not access the encrypted partition.	Check the encrypted_mountnames and encrypted_mountpoint values in hosts.inv. Verify connectivity between the installer and the partition.
luksipc error	Could not preserve existing data, encryption process was halted.	Verify the encrypted_blockdev, encrypted_mountpoint, and encrypted_mountnames values in hosts.inv.
luksOpen error	Could not access the encrypted partition.	Check the encrypted_mountnames and encrypted_mountpoint values in hosts.inv.

Error	Cause	Resolution
mkfs error	Could not make a file system on this partition.	Check the encrypted_mountnames values in hosts.inv, verify that you can make a file system on this partition.
mount point does not exist	The mount point listed in hosts.inv does not exist.	Verify the encrypted_mountpoint values set in hosts.inv.
Network Timeout, closing socket	Connectivity with the server has been lost or slowed.	Check your network connectivity for impediments. Adjust the timeout setting in kmip-client.properties if needed.
partition is already encrypted, will not modify it	The partition selected for encryption in the hosts.inv file has already been encrypted by XCrypt Full Disk.	No need to re-apply XCrypt Full Disk. Remove, or comment out, the encrypted partition entry from hosts.inv.
Registration request failed	The attempt to save a key in the KMIP database failed.	View /var/log/zts/kmip.log and the system log for more details.
resize of <partition> failed-run fsck	Could not resize the partition.	Run fsck on the partition.
unmount error	Could not unmount the partition.	Use lsof and fuser to check the system usage and diagnose the problem.

Contacting Zettaset

Zettaset Corporate Headquarters
465 Fairchild Drive
Mountain View, California 94043
USA

Toll Free: +1-888-511-3736
Fax: +1-650-314-7950

Sales and Support
Email: sales@zettaset.com
Phone: +1-650-314-7927