

# Dell EMC Isilon: Using Transparent Data Encryption with Isilon HDFS

## Abstract

With the introduction of Dell EMC OneFS v8.2, HDFS Transparent Data Encryption (TDE) is now supported to allow end-to-end data protection in Hadoop clusters using Dell EMC Isilon for HDFS storage. This paper covers the steps required for setting up and validating TDE with Isilon HDFS.

August 2021

## Revisions

Date	Description
December 2019	Initial release
June 2021	Ranger KMS / Active Directory configurations and OneFS 8.2.2 Hadoop TDE operational changes.
August 2021	PowerScale SyncIQ to replicate Hadoop encryption zones

## Acknowledgements

Author: Kirankumar Bhusanurmath, Analytics Solution Architect  
Boni Bruno, CISSP, CISM, CGEIT, Chief Solutions Architect

The information in this publication is provided “as is.” Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Copyright © 2021 Dell Inc. or its subsidiaries. All Rights Reserved. Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners. [8/5/2021] [Technical White Paper] [H18083.2]

# Table of contents

1	Introduction.....	5
2	Transparent Data Encryption overview .....	6
3	Configuring HDFS TDE with Isilon .....	7
3.1	Creating an Encryption Zone Key.....	7
3.1.1	KMS setup Isilon OneFS .....	10
3.2	Adding a Key Management Service URL To Isilon .....	11
3.3	Configuring an Encryption Zone .....	12
3.3.1	Isilon OneFS 8.2.....	12
3.3.2	Isilon OneFS 8.2.2 and later version .....	13
4	Validating the HDFS TDE configuration with Isilon .....	14
4.1	Listing Encryption Zones .....	14
4.2	Writing and Reading from Encryption Zones.....	14
5	Enabling Ranger KMS Audit with Isilon.....	18
6	PowerScale SyncIQ for data replication with TDE enabled .....	22
6.1	Solution validation.....	22
6.1.1	Setup Hadoop cluster .....	22
6.1.2	PowerScale SyncIQ root directory.....	22
6.1.3	PowerScale SyncIQ policy .....	22
6.1.4	Functional testing.....	23
7	Performance tests .....	25
8	Conclusion.....	26
A	Appendix.....	27
A.1	Using Isilon OneFS with Key Trustee KMS and CDH.....	27
A.2	Configure KMS URL in OneFS.....	28
B	Technical support and resources .....	30
B.1	Additional Isilon OneFS Documentation.....	30

## Executive summary

Data encryption is mandatory for many government, financial, and regulatory entities, worldwide, to meet privacy and other security requirements. The Hadoop Distributed File System (HDFS) is the primary data storage system used by Hadoop applications. HDFS Transparent Data Encryption (TDE) improves data security in Hadoop Big Data environments by allowing users to encrypt files or directories with HDFS. TDE prevents unauthorized users to read HDFS files at the operating system level.

With the introduction of Isilon OneFS v 8.2, HDFS Transparent Data Encryption (TDE) is now supported to allow end-to-end data protection in Hadoop clusters using Dell EMC Isilon Scale-out NAS for HDFS storage. The TDE implementation with Isilon HDFS allows transparent encryption of data that is read from or written to any HDFS storage node in an Isilon Hadoop cluster. Transparent means that end-users are unaware of the encryption/decryption processes, and *end-to-end* means that data is encrypted at-rest and in-transit. Both encryption capabilities are very important to meeting various security, regulatory, and compliance requirements.

This paper covers the steps required for setting up and validating Transparent Data Encryption for Hadoop clusters using Dell EMC Isilon Scale-out NAS for HDFS data storage.

## Audience

This document is intended for organizations interested in implementing transparent data encryption with Isilon HDFS. Solution architects, system administrators and others interested readers within those organizations constitute the target audience.

# 1 Introduction

Integrating HDFS with an *external*, enterprise-level keystore is the first step to deploying transparent data encryption (TDE). This is because separation of duties between a key administrator and an HDFS administrator is a very important aspect of this feature. However, most keystores are not designed for the encrypt/decrypt request rates seen by Hadoop workloads.

This led to the development of a new service, called the **Hadoop Key Management Server (KMS)**, which serves as a proxy between HDFS clients and the backend keystore. Both the keystore and Hadoop KMS must use Hadoop's KeyProvider API to interact with each other and with HDFS clients.

This document demonstrates how to setup and validate HDFS TDE with Dell EMC Isilon Scale-out NAS running OneFS v 8.2. This is the first release of Isilon OneFS to support HDFS TDE.

## 2 Transparent Data Encryption overview

When using TDE, data read from and written to special HDFS directories, called *encryption zones*, is transparently encrypted and decrypted without requiring changes to user application code. The contents of encryption zones are transparently encrypted upon write and transparently decrypted upon read.

Three types of keys are needed for encrypting and decrypting files:

	Key type	Key function	Key association
1.	Data Encryption Key (DEK)	Plain Text Key Used to Encrypt and Decrypt HDFS files	One per file.
2.	Encrypted Data Encryption Key (EDEK)	Encrypted DEK	One per DEK
3.	Encrypted Zone Key (EZK)	Encrypts/Decrypts DEK	One per Encryption Zone

Each encryption zone is associated with a single encryption zone key, which is specified when the zone is created. Each file within an encryption zone has its own unique *data encryption key* (DEK). DEKs are never handled directly by HDFS. HDFS only handles an *encrypted data encryption key* (EDEK). HDFS clients decrypt an EDEK, and then use the subsequent DEK to read and write data. During this transfer, Isilon HDFS data nodes simply see a stream of encrypted bytes.

The following diagram illustrates how encryption zone keys (EZ keys), data encryption keys (DEKs), and encrypted data encryption keys (EDEKs) are used to encrypt and decrypt HDFS files on Isilon.



## 3 Configuring HDFS TDE with Isilon

HDFS TDE requires a Key Management Service (KMS). The KMS is responsible for storing encryption keys. The KMS also provides a REST API and access control on the keys stored in the KMS database. There are variants of KMS products available, Apache Hadoop KMS, Cloudera Key Trustee KMS, IBM Watson KMS, and Ranger KMS to name a few. Any compatible Hadoop KMS will work with Isilon.

In this paper, I refer to the Ranger KMS provided with Hortonworks Data Platform (HDP) since more Dell EMC customers tend to use it with Hadoop. For information on how to install Ranger KMS, please refer to this [HDP link](#).

The remainder of this section covers the steps needed to configure HDFS TDE with Isilon.

### 3.1 Creating an Encryption Zone Key

HDFS TDE introduces the concept of an **encryption zone** (EZ), which is a directory in HDFS whose contents will be automatically encrypted on write and decrypted on read.

Encryption zones always start off as empty directories, and tools such as **distcp** with the **-skipcrccheck -update** flags can be used to add data to a zone - these flags are required because encryption zones are being used. Every file and subdirectory copied to an encryption zone will be encrypted.

The first step in creating an encryption zone is to create an encryption zone key. Assuming the KMS is installed and configured for Hadoop, you can easily create an encryption zone key on the KMS itself.

An example of the key creation screen is shown for reference below.

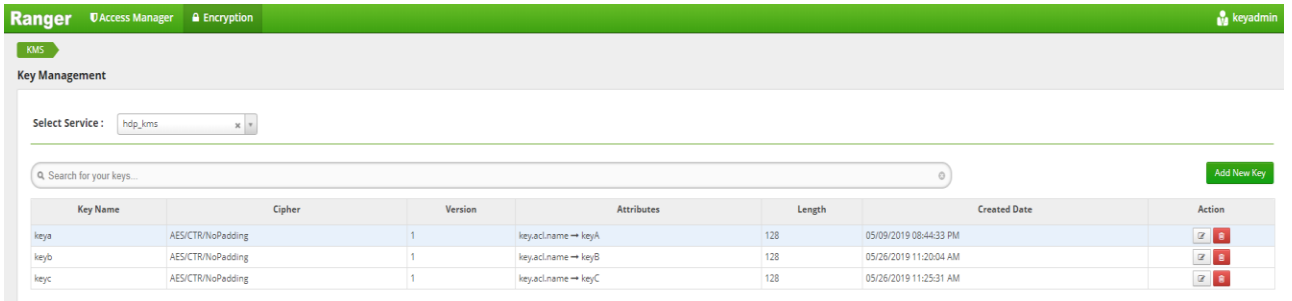
The screen shot is specific to Ranger KMS:

The screenshot shows the Ranger KMS interface for creating a new key. The breadcrumb navigation is KMS > hdp\_kms > Key Create. The form is titled 'Key Detail' and contains the following fields:

- Key Name \***: A text input field.
- Cipher**: A dropdown menu with 'AES/CTR/NoPadding' selected.
- Length**: A text input field with '128' entered.
- Description**: A text area.
- Attributes**: A table with two columns: 'Name' and 'Value'. There is a red 'x' button to the right of the table.
- A '+' button is located below the attributes table.
- At the bottom of the form are 'Save' and 'Cancel' buttons.

After specifying the key name and key properties, click “save” to create the key.

The resulting screen will list the new key, below is an example screen shot of a key listing on Ranger KMS:



You can also create an encryption zone key from any client in the Hadoop cluster by issuing the following command:

```
[hadoop_client]$ hadoop key create <keyname>
```

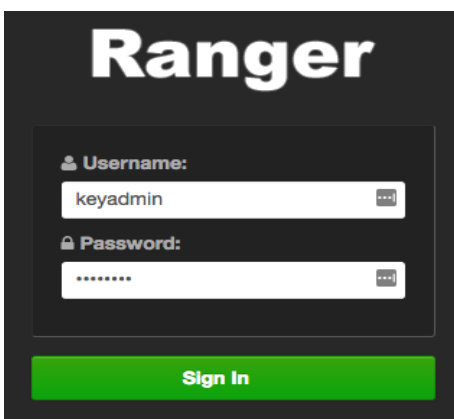
The above command will create a key on the KMS and an acknowledgment message as shown below:

```
<keyname> has been successfully created with options
{cipher='AES/CTR/NoPadding', bitlength=128, description='null',
attributes='null'}. KMSPClientProvider has been updated.
```

If you get the following authorization error after issuing the hadoop key create command:

```
<keyname> has not been created.
org.apache.hadoop.security.authorize.AuthorizationException:
User: ambari-qa not allowed to do 'CREATE_KEY' on '<keyname>'
```

It means the Hadoop user issuing the command does not have permissions defined in the KMS policy to create keys. The error message will contain the user name (e.g. user ambari-qa in the output above). To resolve the issue, you will need to log into the KMS and create a policy for the specified user.



The default admin user for Ranger KMS is 'keyadmin' and the default password is 'keyadmin'.

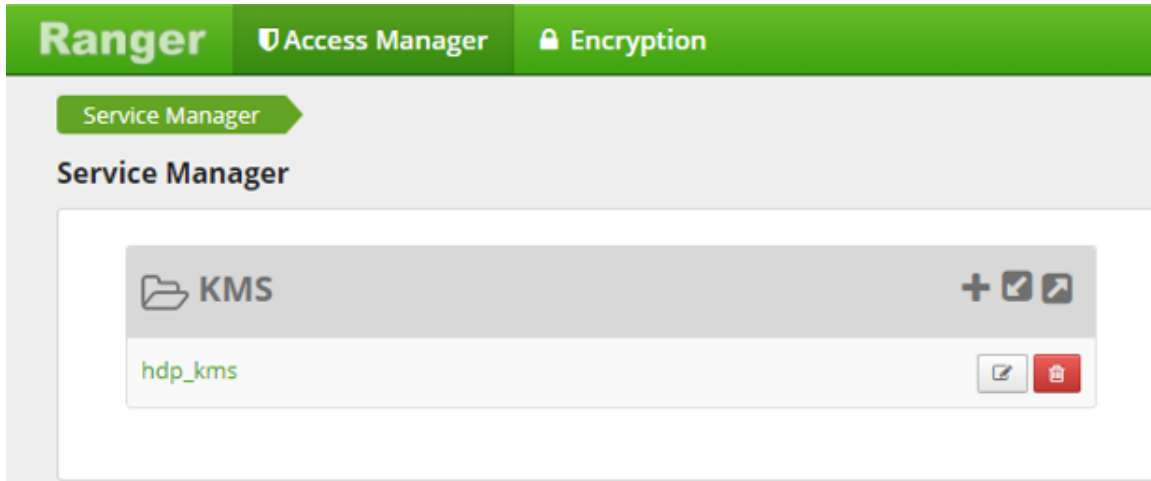
The default KMS GUI port is TCP port 6080.



Access the Ranger User Interface from a browser.

For example: [http://RANGER\\_FQDN\\_ADDR:6080](http://RANGER_FQDN_ADDR:6080)

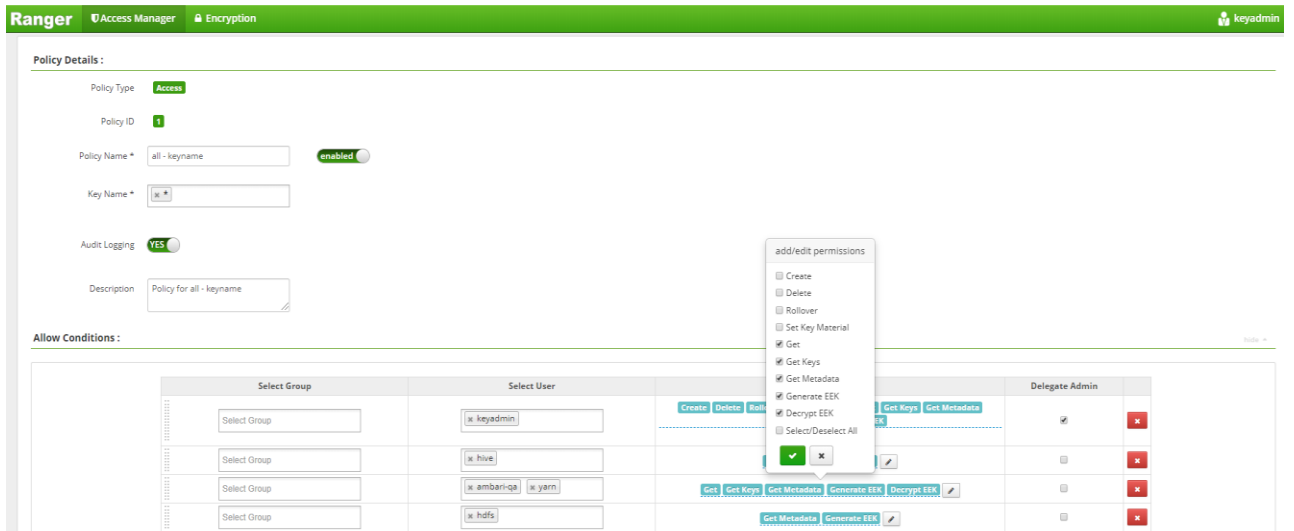
After signing into the KMS, you will see the available Service Manager for the Hadoop HDP cluster as shown below:



Click on the KMS link.

In the above example, **hdp\_kms** is the associated KMS for the HDP Hadoop cluster.

At this point you can edit the KMS policy:



Now you can add or remove users and edit permissions to meet your security requirements. To allow users to create keys, they need the 'create' permission assigned, to allow users to decrypt data, they need the 'Decrypt EEK' permission assigned and so on.

The above policy shown applies to all keys since a wildcard '\*' is specified for the key name. If creating different policies for different keys, pay close attention to the groups and users defined for each policy to ensure the policy is compliant with your security requirements.

### 3.1.1 KMS setup Isilon OneFS

Isilon OneFS uses different user to Get Metadata from the KMS when different authentication providers are used. Make sure the KMS has a policy to allow the user to "Get Metadata", without the stated hdfs permissions, an encryption zone cannot be created on Isilon. The user also needs to be able to "Generate EEK" to allow authorized users to place files into an HDFS encryption zone on Isilon.

**Note:** In the KDC set the delegation token renew duration equal to the delegation token max expired time

#### 3.1.1.1 Authentication provider MIT KDC

When the authentication provider is MIT KDC for the Hadoop and Isilon cluster, then Isilon OneFS uses the "hdfs" user to GET Metadata from the KMS. Make sure the KMS has a policy to allow the "hdfs" user to "Get Metadata".

An example MIT KDC/KMS policy is shown below for reference:

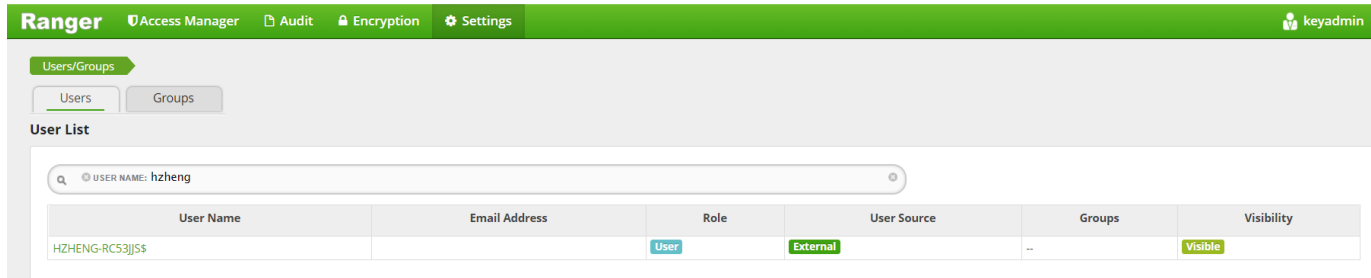
Allow Conditions :

Select Group	Select User	Permissions	Delegate Admin
Select Group	* keyadmin	Create Delete Rollover Set Key Material Get Generate EEK Decrypt EEK Get Keys Get Metadata	<input checked="" type="checkbox"/>
Select Group	* hive	Get Metadata Decrypt EEK	<input type="checkbox"/>
Select Group	* ambari-qa * yarn	Get Get Keys Get Metadata Generate EEK Decrypt EEK	<input type="checkbox"/>
Select Group	* hdfs	Get Metadata Generate EEK	<input type="checkbox"/>

#### 3.1.1.2 Authentication provider Microsoft AD

When the authentication provider is Microsoft AD for the Hadoop and Isilon cluster, then Isilon OneFS uses the AD Machine Account of the Isilon and Hadoop cluster. Make sure the KMS has a policy to allow the "AD\_MACHINE\_ACCOUNT\$" followed by "\$" to "Get Metadata".

**Note:** Make sure the AD Machine Account that is PowerScale cluster name is visible in the Ranger KMS Users/Group as shown in the below image, if not visible then manually has to be created with the "\$".



An example AD/KMS policy is shown below for reference: "HZHENG-RC53JJS\$" is the AD machine account of Isilon and Hadoop cluster.

Allow Conditions :

Select Group	Select User	Permissions	Delegate Admin	
Select Group	keyadmin   hdfs   rangerkms	Create   Delete   Rollover   Set Key Material   Get Get Keys   Get Metadata   Generate EEK   Decrypt EEK	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Select Group	hdfs   <b>HZHENG-RC53JJS</b>	Create   Get Keys   Get Metadata   Generate EEK Decrypt EEK	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Select Group	hive	Get Metadata   Decrypt EEK	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Select Group	yarn-ats   yarn   ranger	Get   Get Keys   Get Metadata   Generate EEK Decrypt EEK	<input checked="" type="checkbox"/>	<input type="checkbox"/>

**Note:** The GUI interface shown above is specific to the Ranger KMS that is included with HDP, other KMS products will have different GUIs and options and the process to edit policies will be different. What is important to remember is the need to have authorization policies that allow the appropriate user access to keys.

### 3.2 Adding a Key Management Service URL To Isilon

Currently with OneFS v 8.2, TDE can only be configured on Isilon using the CLI, configuring TDE using the Isilon WebUI is not supported. For TDE to work with Isilon, the Hadoop KMS must be defined on Isilon OneFS.

To configure the KMS information on Isilon OneFS, issue the following OneFS CLI command:

```
<isilon-cli># isi hdfs crypto settings modify --kms-url=<url-string> --zone=<hdfs-zone-name> -v
```

**Note:** The <url-string> needs to specify the port of the KMS and not the KMS GUI port. The default Ranger KMS port is 9292, do not specify the KMS GUI port of 6080 in the <url-string>. For example, the url-string syntax could be --kms-url=<http://FQDN-OF-KMS-SERVER:9292>

Confirm the KMS configuration on Isilon OneFS by issuing the following OneFS CLI command:

```
<isilon-cli># isi hdfs crypto settings view --zone=<hdfs-zone-name>
```

An example output of the command is shown below in green for reference:

```
<isilon-cli># isi hdfs crypto settings view --zone=hdfs
Kms Url: http://kms.domain.com:9292
```

After the KMS information is configured, the next step is to create a directory and configure an encryption zone on Isilon.

## 3.3 Configuring an Encryption Zone

### 3.3.1 Isilon OneFS 8.2

Before you can create an encryption zone, you need to create a directory for the encryption zone. This directory **MUST** reside beneath the specified *hdfs-root* directory designated for Isilon HDFS. For example, if the *hdfs-root* directory for Isilon HDFS is */ifs/hdfs*, the directories designated for encryption zones must reside directly under the */ifs/hdfs* directory.

Creating a directory in Isilon OneFS CLI is similar to creating a directory in UNIX or Linux, simply run the `mkdir` command as shown below:

```
<isilon-cli># mkdir /ifs/hdfs/<directory-name>
```

After creating the directory for your encryption zone, create the encryption zone with the following OneFS CLI command:

```
<isilon-cli># isi hdfs crypto encryption-zones create --path=<directory-path> --key-name=<key> --zone=<hdfs-zone-name> -v
```

---

**Note:** The `<directory-path>` should match the directory you created previously for the encryption zone, e.g. */ifs/hdfs/A* and the `<key>` must match the `<key>` created on the KMS, e.g. *keyA*. Also, the KMS must have a policy to allow the "hdfs" user access to "Get Metadata". Without this policy, Isilon cannot create an encryption zone and an error will be thrown.

---

Below is an example of the error thrown if the KMS server is missing a "hdfs" user policy.

```
<isilon-cli># isi hdfs crypto encryption-zones create --path=/ifs/hdfs/A --key-name=keyA --zone=hdfs -v
Create Encryption Zone call failed: GetKeyMetaData: KMS return HTTP status: 403;
Remote exception message: User:hdfs not allowed to do 'GET_METADATA' on 'keyA';
Request: http://<FQDN-OF-KMS>:9292/kms/v1/key/keyA/_metadata?user.name=hdfs
```

Below is an example of how two encryption zones (*/B* and */C*) are created on Isilon OneFS. These commands only work when the KMS is configured with the correct policy for the "hdfs" user as shown in the KMS section of this paper:

```
Isilon-cli # mkdir /ifs/hdfs/B
Isilon-cli # isi hdfs crypto encryption-zone create --path=/ifs/hdfs/B --key-name=keyB --zone=hdfs

Isilon-cli # mkdir /ifs/hdfs/C
Isilon-cli # isi hdfs crypto encryption-zone create --path=/ifs/hdfs/C --key-name=keyC --zone=hdfs
```

As stated in this paper, the encryption zone must be created on Isilon using the OneFS CLI. If the stated procedures are not followed using the Isilon OneFS CLI, and instead the commands to create an encryption zone are issued directly from a Hadoop client using the following hadoop command line tool:

```
Hadoop-client>$ hdfs crypto -createZone -keyName <key> -path <path>
```

The following error will be generated:

```
RemoteException: Unknown RPC: createEncryptionZone
```

Similarly, the Isilon HDFS log will contain the following error if you try to create the encryption zone directly from a client:

```
isilon: 2019-05-26T23:01:29-04:00 <30.6> isilon hdfs[3798]: [hdfs] RPC V9 user:
root exception: org.apache.hadoop.ipc.RpcNoSuchMethodException cause: Unknown
RPC: createEncryptionZone
```

Currently, Isilon OneFS v 8.2 does not implement the 'createEncryptionZone' remote procedure call, so make sure to create the encryption zone on Isilon using the OneFS CLI to avoid the errors shown above.

### 3.3.2 Isilon OneFS 8.2.2 and later version

Before you can create an encryption zone, you need to create a directory for the encryption zone. This directory **MUST** reside beneath the specified *hdfs-root* directory designated for Isilon HDFS.

Run Hadoop mkdir command to create the directory for the encryption zone as shown below:

```
<Hadoop-client># sudo -u hdfs hdfs dfs -mkdir -p /<directory-name>
```

After creating the directory for your encryption zone, create the encryption zone with the following hdfs command on the Hadoop hdfs client or edge node of the Hadoop cluster:

```
<Hadoop-client># sudo -u hdfs hdfs crypto -createZone -keyName <key> -path
<directory-path>
```

---

**Note:** The <directory-path> should match the directory you created previously for the encryption zone, e.g. /ifs/hdfs/A and the <key> must match the <key> created on the KMS, e.g. keyA. Also, the KMS must have a policy to allow the "hdfs" user access to "Get Metadata". Without this policy, Isilon cannot create an encryption zone and an error will be thrown.

---

Below is an example of the error thrown if the KMS server is missing a "hdfs" user policy.

```
<Hadoop-client># sudo -u hdfs hdfs crypto -createZone -keyName keyA -path /A
Create Encryption Zone call failed: GetKeyMetaData: KMS return HTTP status: 403;
Remote exception message: User:hdfs not allowed to do 'GET_METADATA' on 'keyA';
Request: http://<FQDN-OF-KMS>:9292/kms/v1/key/keyA/_metadata?user.name=hdfs
```

Below is an example of how two encryption zones (/B and /C) are created on Isilon OneFS. These commands only work when the KMS is configured with the correct policy for the "hdfs" user as shown in the KMS section of this paper:

```
<Hadoop-client># sudo -u hdfs dfs -mkdir /B
<Hadoop-client># sudo -u hdfs hdfs crypto -createZone -keyName keyB -path /B

<Hadoop-client># sudo -u hdfs dfs -mkdir /C
<Hadoop-client># sudo -u hdfs hdfs crypto -createZone -keyName keyC -path /C
```

## 4 Validating the HDFS TDE configuration with Isilon

Once the keys and policies are created on KMS and the encryption zones are defined on Isilon OneFS, you can validate the TDE configuration.

### 4.1 Listing Encryption Zones

You can list the encryption zones on Isilon with the following OneFS CLI command:

```
<Hadoop-client># sudo -u hdfs hdfs crypto -listZones
Path                Key Name
-----
/ifs/hdfs/A         keyA
/ifs/hdfs/B         keyB
/ifs/hdfs/C         keyC
-----
Total: 3
```

Once the encryption zones are defined and verified on Isilon OneFS, any Hadoop client will be able to list the encryption zones on Isilon using standard Hadoop command line tools.

To list encryption zones from a Hadoop client, use the following command:

```
Hadoop-client>$ hdfs crypto -listZones

/A keyA
/B keyB
/C keyC
```

---

**Note:** The client encryption zone listing will not show the hdfs-root directory path as seen on Isilon. This is normal and expected.

---

### 4.2 Writing and Reading from Encryption Zones

With the encryption zones configured and available to all Hadoop clients, authorized users will be able to write and read files to the encrypted zones. The ambari-qa user is the default smoke-test user included with HDP, testing TDE with the ambari-qa user requires a KMS policy to allow the ambari-qa user to get keys and metadata as well as generate and decrypt EEK's.

For reference, below is an example Ranger KMS policy that includes the ambari-qa user and the required permissions to write and read files from defined encryption zones:



With the above policy saved to the KMS, the ambari-qa user will be able to write and read a file to and from the created encryption zone on Isilon (e.g. /A ) from any client node in the Hadoop cluster.

The output below shows the current user id and contents of a local ambari-qa test.txt file - the contents of the test file “Can you read this?” is shown for reference.

```
ambari-qa>$ id
```

```
uid=1013(ambari-qa) gid=1041(hadoop) groups=1041(hadoop),100(users)
```

```
ambari-qa>$ cat test.txt
```

```
Can you read this?
```

This test file is placed to encryption zone /A on Isilon by the Hadoop ambari-qa user using the hdfs (-put) command – communication over the network and the test file stored on Isilon is encrypted transparently.

```
ambari-qa>$ hdfs dfs -put test.txt /A
```

```
ambari-qa>$
```

A directory listing (-ls) shows the file was successfully placed by the ambari-qa user to encryption zone /A on Isilon. The (-cat) command is used to read the contents of the file and shows the decrypted information. The encryption and decryption of files are transparent to authorized Hadoop users as shown in the example output below:

```
Ambari-qa>$ hdfs dfs -ls /A
```

```
Found 1 items
```

```
-rw-r--r--    3 ambari-qa hadoop          19 2019-05-27 23:34 /A/test.txt
```

```
-bash-4.2$ hdfs dfs -cat /A/test.txt
```

```
Can you read this?
```

To verify the test.txt file is encrypted on Isilon, an Isilon administrator can access the directory on Isilon as the privileged root user and display the contents of the test.txt file directly in the /A directory from the Isilon CLI.

```
isilon# id
```

```
uid=0(root)gid=0(wheel) groups=0(wheel),5(operator),10(admin),20(staff),70(ifs)
```

```
isilon# cat /ifs/hdfs/A/test.txt
```

```
PM$%D#a
```

As shown in the highlighted yellow section above, the contents of the test.txt file are encrypted, and the original text cannot be seen by the privileged root user on Isilon.

Depending how the permissions on are set on individual encryption zones, read and write access can either be allowed or restricted. For example, the /A encryption zone and the test file itself has the following permissions defined:

```
$ hdfs dfs -ls -d /A
```

```
drwxr-x---  - ambari-qa hadoop          0 2019-05-27 23:34 /A
```

```
-bash-4.2$ hdfs dfs -ls /A
```

```
Found 1 items
```

```
-rwxr-x---  3 ambari-qa hadoop          19 2019-05-27 23:34 /A/test.txt
```

The permissions shown allows read and write permission for the ambari-qa user, read permission for any member in the hadoop group, and no permissions for anyone else. This is true for both the encryption zone directory and the file itself. This means any user that is a member of the hadoop group should be able to decrypt and read the test file transparently if the KMS policy also permits it. This can be tested easily by accessing the ambari-qa test file as a different user that is a member of the hadoop group.

```
[root]# su - yarn
```

```
Last login: Sun May 26 03:04:50 UTC 2019
```

```
yarn>$ hdfs dfs -cat /A/test.txt
```

```
Can you read this?
```

```
yarn>$
```

The above example shows that the user **yarn**, who is a member of the group **hadoop**, can read the ambari-qa created test.txt file. This is because the encryption zone permissions allow hadoop group members read access and there is a KMS policy that allows the **yarn** user to “**DECRYPT\_EEK**” as shown earlier with the example KMS policy in this paper.

However, if the user **yarn** tried to write a file into the /A encryption zone, the Isilon POSIX level security would restrict the writing of the file as shown in the example below:

```
[root]# su - yarn
```

```
Last login: Tue May 28 01:00:06 UTC 2019 on pts/0
```

```
yarn>$ echo "Yarn Test File" > yarn.txt
```

```
yarn>$ hdfs dfs -put yarn.txt /A
```

```
put: Permission denied: user=yarn, access=WRITE, path="/A/yarn.txt._COPYING_"
```



As another example, trying to read the ambari-qa test.txt file with the user **mapred**, who is also a member of the group **hadoop**, will fail since there is no KMS policy that allows the **mapred** user to “**DECRYPT\_EEK**” as shown in the example output below:

```
[root]# su - mapred
```

```
Last login: Wed May 15 03:07:24 UTC 2019
```

```
mapred>$ hdfs dfs -cat /A/test.txt
```

```
cat: User:mapred not allowed to do 'DECRYPT_EEK' on 'keyA'
```

```
mapred>$
```

As a result, HDFS Transparent Data Encryption (TDE) with Isilon provides an additional layer of security for Hadoop clusters - the POSIX level security inherent with Isilon OneFS and the KMS level security provided with integrating TDE with Isilon HDFS. Both POSIX level security and KMS policies need to have correct permissions before files within encryption zones can be read or written to.


## 5 Enabling Ranger KMS Audit with Isilon

Ranger KMS supports audit to Isilon HDFS. Isilon HDFS is good for archival KMS auditing. To enable Ranger KMS auditing to Isilon HDFS, follow these steps:

### Steps



1. Go to the Ambari UI: <http://<gateway>:8080>.
2. Select ranger-kms from the service.
3. Click the Configs tab and then select Advance tab.
4. In the Advanced ranger-kms-audit list, set `xasecure.audit.is.enabled` to true and place a check next to Audit provider summary enabled as shown below:



`xasecure.audit.is.enabled`   



Audit provider summary     
enabled



5. Select "Audit to HDFS" as shown below:

▼ **Advanced ranger-kms-audit**

`ranger.plugin.kms.ambari.cluster.name`   

Audit to HDFS   

`xasecure.audit.destination.hdfs.batch.filespool.dir`   

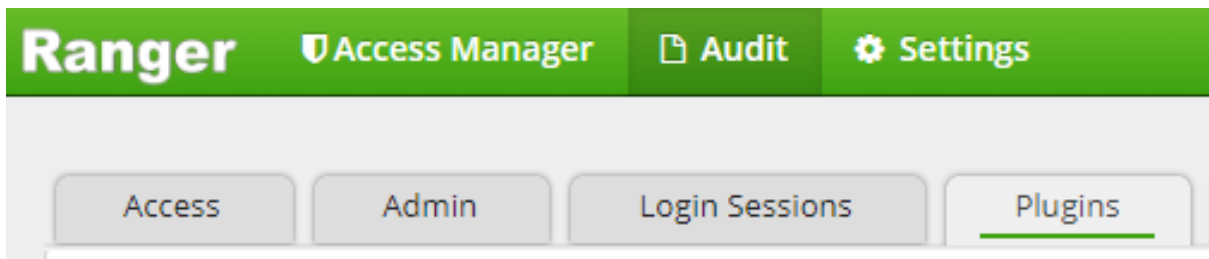
`xasecure.audit.destination.hdfs.dir`   

---

**Note:** The `xasecure.audit.destination.hdfs.dir` path shown above is an example only, specify the fully-qualified-domain-name of the Isilon HDFS SmartConnect zone for the `xasecure.audit.destination.hdfs.dir` in the Advanced ranger-kms-audit section.

---

6. Save the configuration and restart the Ranger KMS service.
7. Check to see if the Ranger KMS Plugin is enabled:
  - a. Go to the Ranger UI: <http://<gateway>:6080>
  - b. Choose the Audit > Plugin Tab as shown below:



- a. Check whether plugins are communicating. The UI should display Http Response code 200 for the respective plugin as shown below:

Service Name	Plugin Id	Plugin IP	Cluster Name	Http Response Code
hdp_kms	kms@n105.solarch.lab.emc.com-hdp_kms	10.246.21.105	hdp	200
hdp_kms	kms@n105.solarch.lab.emc.com-hdp_kms	10.246.21.105	hdp	200
hdp_kms	kms@n105.solarch.lab.emc.com-hdp_kms	10.246.21.105	hdp	200

- c. Make sure that the plugin's root user (kms) has permission to access HDFS  
Path `hdfs://ISILON_HDFS_SMARTCONNECT_FQDN:8020/ranger/audit`
  - d. Restart Ranger KMS.
8. Under `custom core-site.xml`, set `hadoop.proxyuser.kms.groups` to "\*" or to the service user.
  9. In the custom `kms-site` file, add `hadoop.kms.proxyuser.keyadmin.users` and set its value to "\*". (If you are not using keyadmin to access Ranger KMS Admin, replace "keyadmin" with the user account used for authentication.)
  10. In the custom `kms-site` file, add `hadoop.kms.proxyuser.keyadmin.users` and set its value to "\*". (If you are not using keyadmin to access Ranger KMS Admin, replace "keyadmin" with the user account used for authentication.)
  11. In the custom `kms-site` file, add `hadoop.kms.proxyuser.keyadmin.hosts` and set its value to "\*". (If you are not using keyadmin to access Ranger KMS Admin, replace "keyadmin" with the user account used for authentication.)
  12. Create a symbolic link to `/etc/hadoop/conf/core-site.xml` for `/etc/ranger/kms/conf` (`ln -s /etc/hadoop/conf/core-site.xml /etc/ranger/kms/conf/core-site.xml`)
  13. Save the configuration changes and restart all needed services in Ambari.

The KMS audit data is stored on Isilon under the `/ranger/audit/kms` directory. Check for audit data with the following command:

```
kms$ hdfs dfs -ls /ranger/audit/kms
```

Found 6 items

```
drwxrwxr-x - kms kms 0 2019-05-10 02:32 /ranger/audit/kms/20190510
drwxrwxr-x - kms kms 0 2019-05-11 03:11 /ranger/audit/kms/20190511
drwxr-xr-x - kms kms 0 2019-05-15 03:36 /ranger/audit/kms/20190515
drwxr-xr-x - kms kms 0 2019-05-26 18:16 /ranger/audit/kms/20190526
drwxr-xr-x - kms kms 0 2019-05-27 23:34 /ranger/audit/kms/20190527
drwxr-xr-x - kms kms 0 2019-05-28 05:08 /ranger/audit/kms/20190528
```

Each directory will contain audit data for the specified date, an example log is shown below:

```
kms>$ hdfs dfs -cat
/ranger/audit/kms/20190528/kms_ranger_audit_n105.solarch.lab.emc.com.log

{"repoType":7,"repo":"hdp_kms","reqUser":"yarn","evtTime":"2019-05-28
01:00:26.641","access":"decrypteek","resource":"keyA","resType":"keyname","actio
n":"decrypteek","result":1,"policy":1,"enforcer":"ranger-
acl","cliIP":"10.246.21.105","agentHost":"hdp105.dellemc.com","logType":"RangerA
udit","id":"2a966fa6-3c3e-4112-b037-93e214a2c298-
25","seq_num":51,"event_count":1,"event_dur_ms":1,"tags":[],"cluster_name":"hdp"
}
{"repoType":7,"repo":"hdp_kms","reqUser":"mapred","evtTime":"2019-05-28
01:03:33.473","access":"decrypteek","resource":"keyA","resType":"keyname","actio
n":"decrypteek","result":0,"policy":-1,"enforcer":"ranger-
acl","cliIP":"10.246.21.105","agentHost":"hdp105.dellemc.com","logType":"RangerA
udit","id":"2a966fa6-3c3e-4112-b037-93e214a2c298-
26","seq_num":53,"event_count":1,"event_dur_ms":1,"tags":[],"cluster_name":"hdp"
}
{"repoType":7,"repo":"hdp_kms","reqUser":"hdfs","evtTime":"2019-05-28
01:35:51.210","access":"generateeek","resource":"keyA","resType":"keyname","acti
on":"generateeek","result":1,"policy":1,"enforcer":"ranger-
acl","cliIP":"10.246.157.64","agentHost":"hdp105.dellemc.com","logType":"RangerA
udit","id":"2a966fa6-3c3e-4112-b037-93e214a2c298-
27","seq_num":55,"event_count":1,"event_dur_ms":1,"tags":[],"cluster_name":"hdp"
}
```

The log data shows time stamps, requested user info, resource key, action, result, client IP, host, etc. A result of **1** means the action was successful, a result of **0** means unsuccessful.

The local KMS audit file also shows good audit information. The local audit file for Ranger KMS is located at **/var/log/ranger/kms** by default. The current audit log is called **kms-audit.log**. Older log files will have a date extension added at the end of the file.

An example **kms-audit.log** file is shown below for reference:

```
[root@hdp105 kms]# pwd

/var/log/ranger/kms

[root@hdp105 kms]# tail kms-audit.log

2019-05-28 04:45:52,029 UNAUTHENTICATED RemoteHost:10.246.21.105 Method:OPTIONS
URL:http://hdp105.dellemc.com:9292/kms/v1/keyversion/keyA%400/_eek?eek_op=decryp
t ErrorMessage:'Authentication required'
2019-05-28 04:45:52,090 OK[op=DECRYPT_EEK, key=keyA, user=yarn, accessCount=1,
interval=0ms]
2019-05-28 04:46:02,541 OK[op=DECRYPT_EEK, key=keyA, user=yarn, accessCount=1,
interval=10451ms]
```

```
2019-05-28 04:46:17,886 UNAUTHENTICATED RemoteHost:10.246.21.105 Method:OPTIONS
URL:http://hdp105.dellemc.com:9292/kms/v1/keyversion/keyA%400/_eek?eek_op=decryp
t ErrorMsg:'Authentication required'
2019-05-28 04:46:17,937 UNAUTHORIZED[op=DECRYPT_EEK, key=keyA, user=mapred]
2019-05-28 05:01:10,769 UNAUTHENTICATED RemoteHost:10.246.21.105 Method:OPTIONS
URL:http://hdp105.dellemc.com:9292/kms/v1/keyversion/keyA%400/_eek?eek_op=decryp
t ErrorMsg:'Authentication required'
2019-05-28 05:01:10,936 UNAUTHORIZED[op=DECRYPT_EEK, key=keyA, user=mapred]
2019-05-28 05:01:31,334 UNAUTHENTICATED RemoteHost:10.246.21.105 Method:OPTIONS
URL:http://hdp105.dellemc.com:9292/kms/v1/keyversion/keyA%400/_eek?eek_op=decryp
t ErrorMsg:'Authentication required'
2019-05-28 05:01:31,450 OK[op=DECRYPT_EEK, key=keyA, user=yarn, accessCount=1,
interval=0ms]
2019-05-28 05:01:41,623 OK[op=DECRYPT_EEK, key=keyA, user=yarn, accessCount=1,
interval=10173ms]
```

Anything in the log with an access count constitutes a successful operation for the specified key. Unauthorized messages constitute a violation to the existing KMS policy.

The audit procedures described here are specific to Ranger KMS only. Other KMS products will have different setup requirements. Please refer to vendor specific documentation on how to configure audit services for other KMS products.

## 6 PowerScale SyncIQ for data replication with TDE enabled

Simple, efficient, and scalable, Dell EMC PowerScale SyncIQ data replication software provides data intensive businesses with a multi-threaded, multi-site solution for reliable disaster protection.

In this section we demonstrate the supportability of SyncIQ to replicate data from encryption zones, when using TDE. In this solution Hadoop clusters on source and target PowerScale cluster are subscribed to the same Ranger Key management Service for TDE Key management, Access control polices and Audit.

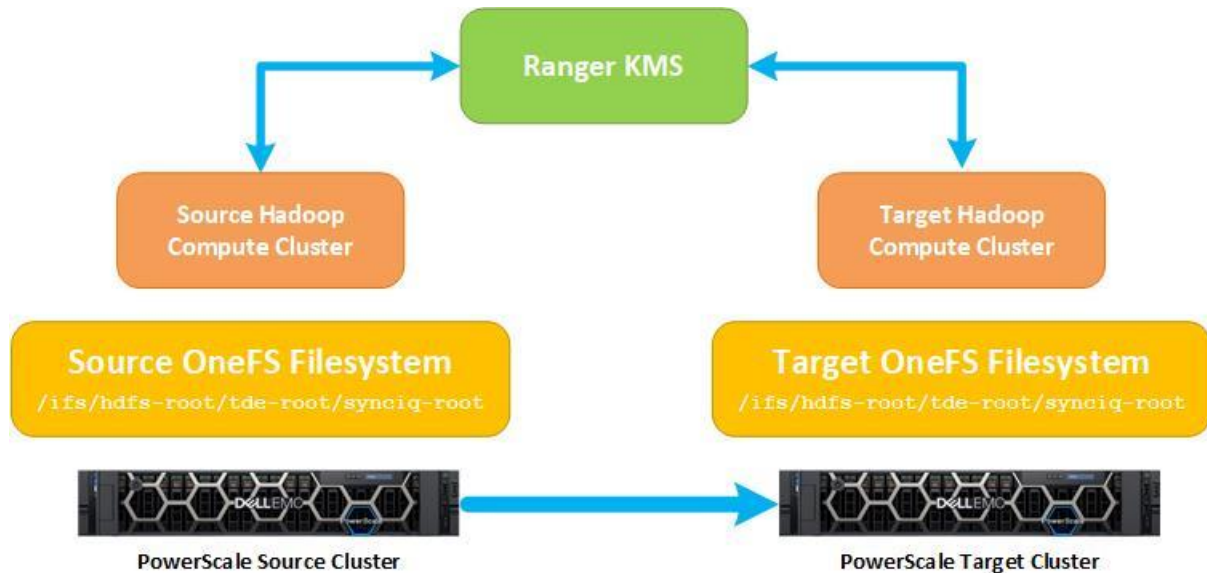


Figure 1 Hadoop TDE synciq one-to-one replication

### 6.1 Solution validation

#### 6.1.1 Setup Hadoop cluster

1. Install Hadoop on source and target PowerScale clusters with a HDFS root directory as expected.  
Example: `/ifs/hdfs-root/`
2. Add Ranger KMS service on both source and target Hadoop clusters.
3. Configure HDFS TDE KMS URL on both source and target clusters as explained in section 3.2 Adding a Key Management Service URL To Isilon
4. Create HDFS TDE Encryption zones of a directory under HDFS root directories in both the clusters **using the same key from the Ranger KMS**. Example: `/ifs/hdfs-root/tde-root`. Refer section .3.3 Configuring an Encryption Zone

#### 6.1.2 PowerScale SyncIQ root directory

On the source Hadoop cluster create a directory under the TDE encryption zone root to be the root of the SyncIQ policy. Example: `/ifs/hdfs-root/tde-root/synciq-root`

#### 6.1.3 PowerScale SyncIQ policy

Create a SyncIQ policy to replicate `/ifs/hdfs-root/tde-root/synciq-root` from the source to target PowerScale cluster.

Figure 2 PowerScale SyncIQ Policy setup

## 6.1.4 Functional testing

### 6.1.4.1 Source Hadoop cluster hdfs client

Connect to the source Hadoop cluster hdfs client and copy a sample file into synciq directory inside the encryption zone.

```
hdfs@source_hdfs_client:~ $ hdfs dfs -copyFromLocal TEXT /tde-source/synciq/
hdfs@source_hdfs_client:~ $ hdfs dfs -cat /tde-source/synciq/TEXT
spam ham test
```

### 6.1.4.2 Source PowerScale cluster

1. Connect to the source PowerScale and make sure the sample file copied into encryption zone is encrypted.

```
source_powerscale# cat /ifs/hdfs/tde-source/synciq/TEXT
c██████████p██████████2#
```

2. Check the OneFS TDE KMS settings

```
source_powerscale# isi hdfs crypto settings
Kms Url: http://10.224.15.207:9292
```

3. Check the encryption zone associated key

```
source_powerscale# isi hdfs crypto encryption-zones list
Path                Key Name
-----
/ifs/hdfs/tde-source key1
-----
Total: 1
```

4. Check the synciq polices

```

source_powerscale# isi sync policies list
Name          Path          Action  Enabled  Target
-----
TDE-policy    /ifs/hdfs/tde-source/synciq copy    Yes     ovsyad-
cl.west.isilon.com
-----
Total: 1

```

### 6.1.4.3 Target Hadoop cluster hdfs client

Connect to target Hadoop cluster hdfs client and check if the file copied into the synciq target directory can be access by the user as expected.

```

hdfs@target_hdfs_client:~ $ hdfs dfs -cat /tde-target/TEXT
spam ham test

```

### 6.1.4.4 Target PowerScale cluster

1. Connect to the target PowerScale and make sure the sample file copied into encryption zone by the SyncIQ from the source PowerScale cluster is encrypted.

```

target_powerscale# cat /ifs/hdfs/tde-target/TEXT
c██████████p██████████2#

```

2. Check the OneFS TDE KMS settings

```

target_powerscale# isi hdfs crypto settings
Kms Url: http://10.224.15.207:9292

```

3. Check the encryption zone associated key

```

target_powerscale# isi hdfs crypto encryption-zones list
Path          Key Name
-----
/ifs/hdfs/tde-target key1
-----
Total: 1

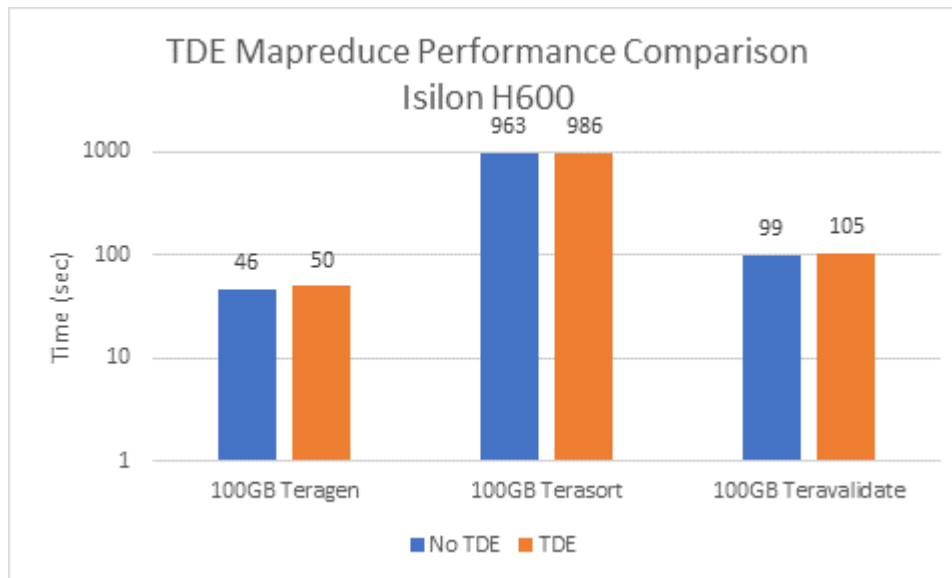
```



## 7 Performance tests

Teragen, Terasort, and Teravalidate are well known Hadoop Mapreduce performance tests. To determine whether encryption zones have an impact on Mapreduce job performance, the tera-suite of performance tests were run against an encryption zone on Isilon and against a regular HDFS directory on Isilon.

The chart below shows the performance results of each test with and without HDFS TDE on Isilon.



Reviewing the performance results, there was an 8.3% performance difference with Teragen, 2.4% difference with Terasort, and 5.9% difference with Teravalidate. Teragen uses Mapreduce to generate data on HDFS, Terasort uses Mapreduce to sort the generated data, and Teravalidate uses Mapreduce to ensure the data was sorted correctly.

The results shown here are specific to a test cluster only. Using HDFS TDE can have an impact on performance, the result will vary by the size and configuration of the Hadoop cluster, the hardware, the network, number of Isilon nodes, and the size of the dataset.

## 8 Conclusion

This document discussed the release of Isilon OneFS v 8.2 and support for HDFS Transparent Data Encryption (TDE). HDFS TDE allows end-to-end data protection for customers using Hadoop clusters with Dell EMC Isilon Scale-out NAS for HDFS storage.

This document covers the steps required for setting up and validating TDE with Isilon HDFS including requirements for the Key Management Service (KMS) needed for HDFS TDE integration. Key management administration, access control settings, and audit configuration steps are also covered in this document.

The overall solution allows for the encryption of sensitive data and protects privileged access. Enterprises can now use HDFS Transparent Data Encryption (TDE) with Isilon OneFS v 8.2 and satisfy compliance sensitive requirements for deployed Hadoop clusters using Dell EMC Isilon Scale-out NAS for HDFS storage.

## A Appendix

### A.1 Using Isilon OneFS with Key Trustee KMS and CDH

The Key Trustee KMS can be used with Cloudera CDH clusters and Isilon OneFS v 8.2. The software should be installed as a parcel, i.e. log into Cloudera Manager and click Hosts > Parcels > Configuration and add the Parcels in Remote Parcel Repository URLs. Note: Key Trustee Server is required by the Key Trustee KMS software. The following parcels were tested with Isilon OneFS 8.2:

- `keytrustee-server-5.15.0-parcels/5.15.0/parcels/`
- `keytrustee-kms-5.15.0-parcels/5.15.0/parcels/`

Download, distribute, and activate the two parcels above.

Before proceeding further, review the entropy level on your designated KMS server with the following command:

```
# cat /proc/sys/kernel/random/entropy_avail
```

Example output:

```
3744
```

A number less the 500 is not enough for cryptographic operations. If the number is too low on your system, install the `rng-tools` package to increase the entropy on your system.

With the entropy at a good level, add the Key Trustee Server Service then the Key Trustee KMS Service per the directions here:

[https://www.cloudera.com/documentation/enterprise/5-14-x/topics/sg\\_hdfs\\_encryption\\_wizard.html#concept\\_ucw\\_lfr\\_wt\\_section\\_ekc\\_kmr\\_wt](https://www.cloudera.com/documentation/enterprise/5-14-x/topics/sg_hdfs_encryption_wizard.html#concept_ucw_lfr_wt_section_ekc_kmr_wt)

Use the existing Key Trustee Server when prompted for a server during the Key Trustee KMS installation. Make sure to restart stale services then redeploy the client configuration.

Setup an authorization secret by accessing the client where the Key Trustee Server is located and run the following command:

```
root@cmclient-4:~ # keytrustee-orgtool add -n cdh-kms -c root@localhost
```

Example output:

```
Dropped privileges to keytrustee
```

Check the privileges by running the following command:

```
root@cmclient-4:~ # keytrustee-orgtool list
```

Example output:

```
Dropped privileges to keytrustee
{
```

```

"cdh-kms": {
  "auth_secret": "2quLbBJooJqy31l1eat7YQ==",
  "contacts": [
    "root@localhost"
  ],
  "creation": "2019-12-06T23:25:24",
  "expiration": "9999-12-31T23:59:59",
  "key_info": null,
  "name": "cdh-kms",
  "state": 0,
  "uuid": "nPJO1IOJKiqGSBQolNhxZdnHsrlQocs4rYM3pQYWFLM"
}
}

```

Use the recommended access control list and set a wildcard \* for the Key Admin User, the Key Admin Group can be left blank. Production deployments should limit the number of Key Admin Users accordingly. TLS was not configured for this setup. TLS KMS configuration instructions are available at the link below:

[https://docs.cloudera.com/documentation/enterprise/latest/topics/cdh\\_sg\\_kms\\_security.html#concept\\_nkz\\_22\\_v\\_mp](https://docs.cloudera.com/documentation/enterprise/latest/topics/cdh_sg_kms_security.html#concept_nkz_22_v_mp)

Finally, set the KMS setting "Authentication Type hadoop.kms.authentication.type" to Kerberos (instead of simple). For details on how to configure Isilon OneFS for Kerberos, see the resources listed below.

## A.2 Configure KMS URL in OneFS

At this point you can configure the KMS URL on Isilon OneFS:

```
isi hdfs crypto settings modify --kms-url=http://<FQDN>:16000
```

Create a key on the KMS, you can use the following curl command:

```
curl -v -X POST -H "Content-Type: application/json" -d '{ "name" : "key1" }'
"http://<FQDN>:16000/kms/v1/keys?user.name=root"
```

Create an encryption zone on OneFS and check the settings:

```
mkdir /ifs/hdfs/ez1
isi hdfs crypto encryption-zones create --key-name=key1 --path=/ifs/hdfs/ez1
isi hdfs crypto encryption-zones list
```

Create a file in the encryption zone and view it:

```
$ hadoop fs -copyFromLocal test.txt /ez1
$ hadoop fs -cat /ez1/test.txt
```

Can you read this?

View the file on the Isilon Cluster:

```
Isilon-h600# cd /ifs/hdfs/ez1
isilon-h600# ls -al
total 15
drwxr-xr-x    3 root  hadoop   47 Jul 17 05:19 .
drwxr-xr-x    8 hdfs  hadoop  134 Jul 16 08:35 ..
drwxr-xr-x    2 root  hadoop    0 Jul 16 09:31 new
-rw-r--r--    1 root  hadoop   14 Jul 17 05:19 test.txt
Isilon-h600# cat test.txt
G?qT7a;n#
```

## B Technical support and resources

[Dell.com/support](https://dell.com/support) is focused on meeting customer needs with proven services and support.

[Storage technical documents and videos](#) provide expertise that helps to ensure customer success on Dell EMC storage platforms.

### B.1 Additional Isilon OneFS Documentation

[Isilon OneFS with Hadoop Kerberos and Identity Management Approaches](#)

[Isilon OneFS Authentication, Identity Management, and Authorization](#)

[Isilon OneFS 8.2.1 Security Configuration Guide](#)

[Isilon OneFS with Hadoop and Cloudera for Kerberos Installation Guide](#)

[Isilon OneFS with Hadoop and Hortonworks For Kerberos Installation Guide](#)