DELL EMC

Technical White Paper

Dell EMC Isilon: Access Control Lists on HDFS and Isilon OneFS

Abstract

This document provides descriptions, comparisons, and migration strategies for access control lists (ACLs) on the Apache® Hadoop® Distributed File System (HDFS) and Dell EMC™ Isilon™ OneFS™.

July 2019

H17850

Revisions

Date	Description
June 2019	Initial release

Acknowledgements

Author: Kirankumar Bhusanurmath (Kirankumar.bhusanurmath@dell.com)

Table of contents

Executive summary

Access control lists (ACLs) provide the ability to specify fine-grained file permissions for specific named users or named groups, an ability that is not limited to just the file owner and group.

This document addresses three main topics that include usage and examples:

Isilon OneFS ACLs: This section provides a brief introduction to the Dell EMC™ Isilon™ OneFS™ ACL technology implementation on the Linux® platform.

Isilon OneFS ACL consideration for HDFS cluster: This section discusses key considerations to set up Apache® Hadoop® Distributed File System (HDFS) ACLs on Isilon OneFS.

Apache HDFS ACLs: This section introduces the Apache HDFS ACL technology.

1 Isilon OneFS ACLs

1.1 OneFS ACLs

OneFS provides a single namespace for multi-protocol access and it has its own internal OneFS ACL representation to perform access control when ACLs are in use. For additional information, refer to the document Access Control Lists on Dell EMC Isilon OneFS. When connecting to an Isilon cluster with SSH, you can manage not only POSIX mode bits but also OneFS ACLs with standard UNIX® tools such as the augmented Isilon OneFS chmod commands.

1.2 OneFS access control entries

The OneFS ACL access control entries (ACEs) contain following information:

- Identity name: The name of a user or group
- ACE type: The type of the ACE (allow or deny)
- ACE permissions and inheritance flags: A list of permissions and inheritance flags separated with commas.

1.3 OneFS ACE permissions

OneFS divides permissions into the following three types:

- Standard ACE permissions: These apply to any object in the file system (see Table 1).
- Generic ACE permissions: These map to a bundle of specific permissions (see Table 2).
- Constant ACE permissions: These are specific permissions for file-system objects (see Table 3).

The standard ACE permissions that can appear for a file-system object are shown in Table 1.

Table 1 OneFS standard ACE Permissions

ACE permission	Applies to	Description
std_delete	Directory or file	The right to delete the object
std_read_dac	Directory or file	The right to read the security descriptor, not including the SACL
std_write_dac	Directory or file	The right to modify the DACL in the object's security descriptor
std_write_owner	Directory or file	The right to change the owner in the object's security descriptor
std_synchronize	Directory or file	The right to use the object as a thread synchronization primitive
std_required	Directory or file	Maps to std_delete, std_read_dac, std_write_dac, and std_write_owner

The generic ACE permissions that can appear for a file-system object are shown in Table 2.

Table 2 OneFS generic ACE permissions

ACE permission	Applies to	Description
generic_all	Directory or file	Read, write, and execute access. Maps to file_gen_all or dir_gen_all.
generic_read	Directory or file	Read access. Maps to file_gen_read or dir_gen_read.
generic_write	Directory or file	Write access. Maps to file_gen_write or dir_gen_write..
generic_exec	Directory or file	Execute access. Maps to file_gen_execute or dir_gen_execute
dir_gen_all	Directory	Maps to dir_gen_read, dir_gen_write, dir_gen_execute, delete_child, and std_write_owner.
dir_gen_read	Directory	Maps to list, dir_read_attr, dir_read_ext_attr, std_read_dac, and std_synchronize.
dir_gen_write	Directory	Maps to add_file, add_subdir, dir_write_attr, dir_write_ext_attr, std_read_dac, and std_synchronize.
dir_gen_execute	Directory	Maps to traverse, std_read_dac, and std_synchronize.
file_gen_all	File	Maps to file_gen_read, file_gen_write, file_gen_execute, delete_child, and std_write_owner.
file_gen_read	File	Maps to file_read, file_read_attr, file_read_ext_attr, std_read_dac, and std_synchronize.
file_gen_write	File	Maps to file_write, file_write_attr, file_write_ext_attr, append, std_read_dac, and std_synchronize.
file_gen_execute	File	Maps to execute, std_read_dac, and std_synchronize.

The constant ACE permissions that can appear for a file-system object are shown in Table 3.

Table 3 OneFS constant ACE permissions

ACE permission	Applies to	Description
modify	File	Maps to file_write, append, file_write_ext_attr, file_write_attr, delete_child, std_delete, std_write_dac, and std_write_owner
file_read	File	The right to read file data
file_write	File	The right to write file data
append	File	The right to append to a file
execute	File	The right to execute a file
file_read_attr	File	The right to read file attributes
file_write_attr	File	The right to write file attributes
file_read_ext_attr	File	The right to read extended file attributes

ACE permission	Applies to	Description
file_write_ext_attr	File	The right to write extended file attributes
delete_child	Directory or file	The right to delete children, including read-only files within a directory; this is currently not used for a file, but can still be set for Windows compatibility
list	Directory	List entries
add_file	Directory	The right to create a file in the directory
add_subdir	Directory	The right to create a subdirectory
traverse	Directory	The right to traverse the directory
dir_read_attr	Directory	The right to read directory attributes
dir_write_attr	Directory	The right to write directory attributes
dir_read_ext_attr	Directory	The right to read extended directory attributes
dir_write_ext_attr	Directory	The right to write extended directory attributes

1.4 OneFS ACL inheritance

Inheritance allows permissions to be layered or overridden as needed in an object hierarchy and allows for simplified permissions management. The semantics of OneFS ACL inheritance meanings are the same as Microsoft® Windows® ACL inheritance, and they are easy to understand if you are already familiar with Windows NTFS ACL inheritance. Table 4 shows the ACE inheritance flags defined in OneFS.

Table 4 OneFS ACE inheritance

ACE permission	Applies to	Description
object_inherit	Directory only	Indicates than a ACE applies to the current directory and files within the directory
container_inherit	Directory only	Indicates than a ACE applies to the current directory and subdirectories within the directory
inherit_only	Directory only	Indicates than a ACE applies to subdirectories only, files only, or both within the directory.
no_prop_inherit	Directory only	Indicates than a ACE applies to the current directory or only the first-level contents of the directory, not the second-level or subsequent contents
inherited_ace	File or directory	Indicates that an ACE is inherited from the parent directory

1.5 OneFS ACL examples

The following examples show how ACLs can be used with OneFS.

1.5.1 Set, modify, and view a OneFS ACL

To manage and manipulate ACLs directly in OneFS, traditional **ls** and **chmod** command tools are retrofitted. They can be used to view and manage ACLs on OneFS, including adding, modifying, and deleting ACL entries. They also provide options to set ACL inheritance flags. This example shows how to use the retrofitted **ls** and **chmod** command tools to manage OneFS ACL.

To view the ACL of a file in OneFS, run the command **ls –le** or **ls –len**. To view the ACL of a directory in OneFS, run the command **ls –led** or **ls –lend**. The **–n** option in the command is used to display user and group IDs numerically rather than converting to a user or group name string.

```
kbhusan-amtvqmw-1# ls -led parent-dir
drwxr-xr-x +  3 hdfs  hadoop  25 Jun 17 09:41 parent-dir
 OWNER: user:hdfs
 GROUP: group:hadoop
 0: user:hdfs allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: user:mktg1 allow dir_gen_read,dir_gen_execute
 2: group:hadoop allow dir_gen_read,dir_gen_execute
 3: everyone allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 1 ls -led example showing an ACL

Figure 1 shows the ACL of a directory. In the output, the **+** sign is added after the POSIX mode bits, indicating that a file or directory contains a OneFS real ACL.

The following breaks down a chmod to add an ACL:

```
#chmod +a# <#> group|user <group|user_name> allow|deny <permissions> <folder>
```

The parameter details are as follows:

```
\+a#                 To add the ACE at position number # in the ACL
Group|user           group and user being added
Group|user_name      group name or````````` user name to be added
Allow|deny           Allow or Deny permission
Permissions          the ACE itself
Folder               the folder o file to be applied to
```

This example adds an ACL entry to the directory using the **chmod** command with the **+a#** option. As Figure 2 shows, the ACL entry is placed in the ACL with index **1** and allows **mktg1** to have the permissions of **dir_gen_read and dir_gen_write** for the parent-dir directory.

```
kbhusan-amtvqmw-1# chmod +a# 1 user mktg1 allow dir_gen_read,dir_gen_execute parent-dir
kbhusan-amtvqmw-1# ls -led parent-dir
drwxr-xr-x +  3 hdfs  hadoop  25 Jun 17 09:41 parent-dir
 OWNER: user:hdfs
 GROUP: group:hadoop
 0: user:hdfs allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: user:mktg1 allow dir_gen_read,dir_gen_execute
 2: group:hadoop allow dir_gen_read,dir_gen_execute
 3: everyone allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 2 Add an ACL entry

To modify the ACL entry added previously, use the **chmod** command with the **=a#** option. Some shells require **=** to be escaped with the **** character. As Figure 3 shows, the ACL entry is modified only to grant the permission of **dir_gen_read** to **mktg1**.

```
kbhusan-amtvqmw-1# chmod \=a# 1 user mktg1 allow dir_gen_read parent-dir
kbhusan-amtvqmw-1# ls -led parent-dir
drwxr-xr-x +  3 hdfs  hadoop  25 Jun 17 09:41 parent-dir
 OWNER: user:hdfs
 GROUP: group:hadoop
 0: user:hdfs allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: user:mktg1 allow dir_gen_read
 2: group:hadoop allow dir_gen_read,dir_gen_execute
 3: everyone allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 3 Modify an ACL entry

To delete the ACL entry modified previously, use the **chmod** command with the **–a#** option, followed by an index of the ACE, shown in Figure 4.

```
kbhusan-amtvqmw-1# chmod -a# 1 parent-dir
kbhusan-amtvqmw-1# ls -led parent-dir
drwxr-xr-x +  3 hdfs  hadoop  25 Jun 17 09:41 parent-dir
 OWNER: user:hdfs
 GROUP: group:hadoop
 0: user:hdfs allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: group:hadoop allow dir_gen_read,dir_gen_execute
 2: everyone allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 4 Delete an ACL entry

1.5.2 Use ACE Inheritance

This example adds an ACL entry with the **object_inherit** and **container_inherit** inheritance flags specified. This applies the ACE to the current directory and propagates it to the subdirectories and files within the parent-dir directory.

```
kbhusan-amtvqmw-1# chmod +a# 1 user mktg1 allow dir_gen_read,dir_gen_execute,object_inherit,container_inherit parent-dir
kbhusan-amtvqmw-1# ls -led parent-dir
drwxr-xr-x +  3 hdfs  hadoop  25 Jun 17 09:41 parent-dir
 OWNER: user:hdfs
 GROUP: group:hadoop
 0: user:hdfs allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: user:mktg1 allow dir_gen_read,dir_gen_execute,object_inherit,container_inherit
 2: group:hadoop allow dir_gen_read,dir_gen_execute
 3: everyone allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 5 ACL with inheritance flags

The next step creates a new directory under parent-dir from Hadoop client. Figure 6 shows the ACE with the flags of **object_inherit** and **container_inherit** specified in the parent directory, which are propagated to the new directory. There is also an **inherited_ace** flag that indicates the ACE is inherited from parent directory and not explicit specified.

```
kbhusan-amtvqmw-1# ls -led parent-dir/sub-dir2
d---r-x--- +  2 hdfs  hadoop  0 Jun 17 09:50 parent-dir/sub-dir2
 OWNER: user:hdfs
 GROUP: group:hadoop
 CONTROL:dacl_auto_inherited,sacl_auto_inherited
 0: user:mktg1 allow inherited dir_gen_read,dir_gen_execute,object_inherit,container_inherit,inherited_ace
kbhusan-amtvqmw-1#
```

Figure 6 Inherited ACL on a new directory

2 Isilon ACL considerations for HDFS cluster during migration

HDFS ACLs are 100% compliant with POSIX ACLs. This section describes how to support POSIX features in OneFS.

When a client connects to a OneFS cluster with HDFS, permission checking is based on the on-disk OneFS internal permission, either POSIX bits, or the OneFS ACL. In OneFS, it is required to present a protocol-specific view of every file to clients. OneFS will map its internal permission to a protocol-specific view while the permission checking is still based on its explicit internal permission representation. The following subsections show the permission-inheritance flag mapping and permission mapping between HDFS protocols and the OneFS ACL.

When a file contains an authoritative OneFS real ACL, the POSIX mode bits are only for representation and are not expressive enough to represent the actual OneFS ACL permissions on disk. When an HDFS client checks the POSIX mode bits of a file, if the file contains a OneFS real ACL, it is not possible to see the actual permission of the file from the client side.

Note: Refer appendix A to understand Apache Hadoop ACL usage and examples.

2.1 ACL migration motive

The OneFS HDFS protocol does not support HDFS ACLs, and this requires migrating HDFS ACLs to Isilon OneFS ACLs to maintain a granular ACL. Isilon OneFS does not support POSIX ACLs, so POSIX ACLs need to be migrated to OneFS ACLs; a detailed HDFS ACLs/POSIX ACLs mapping to OneFS ACL is described in section 2.2.

ACL migration is required when customers plan to migrate DAS Hadoop data onto Isilon storage and retain rich ACLs. Migration can be done manually using Windows Explorer or through the Isilon CLI.

2.1.1 Windows Explorer to manage Isilon ACLs

A simple method of managing OneFS ACLs is to make use of Windows SMB share to administratively manage the native OneFS ACL's graphically.

Create a new SMB share on the Isilon HDFS Access zone and provide the Hadoop Admin full control or access to change the security settings.

```
isi smb shares create --name=hdfs-share --path=/ifs/hdfs-root
isi smb shares permission modify hdfs-share --permission-type=allow --permission=full --wellknown=everyone
```

Figure 7 Isilon commands to create HDFS access zone SMB share

From Windows, connect to the Isilon HDFS access zone SMB share

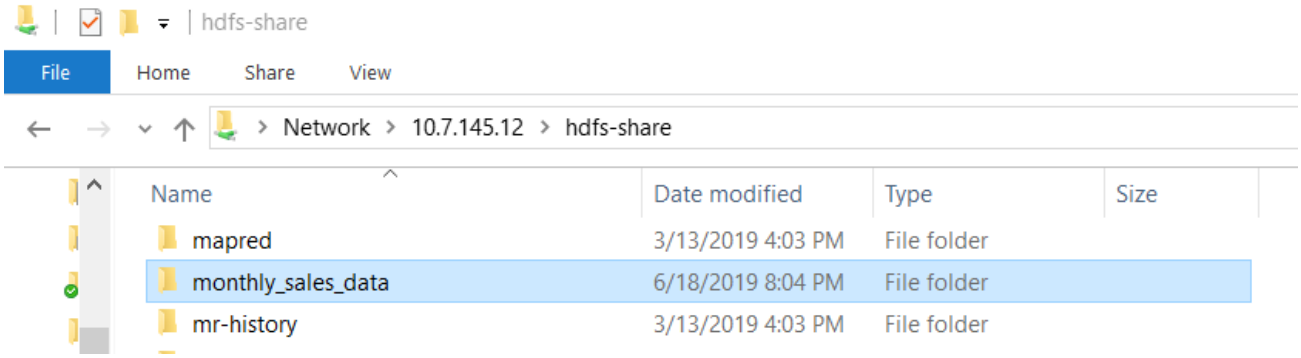

Figure 8 Windows Explorer

Right-click the folder and click **Properties** > **Security** > **Advanced** to add or update Isilon ACLs for the user or groups of folders.

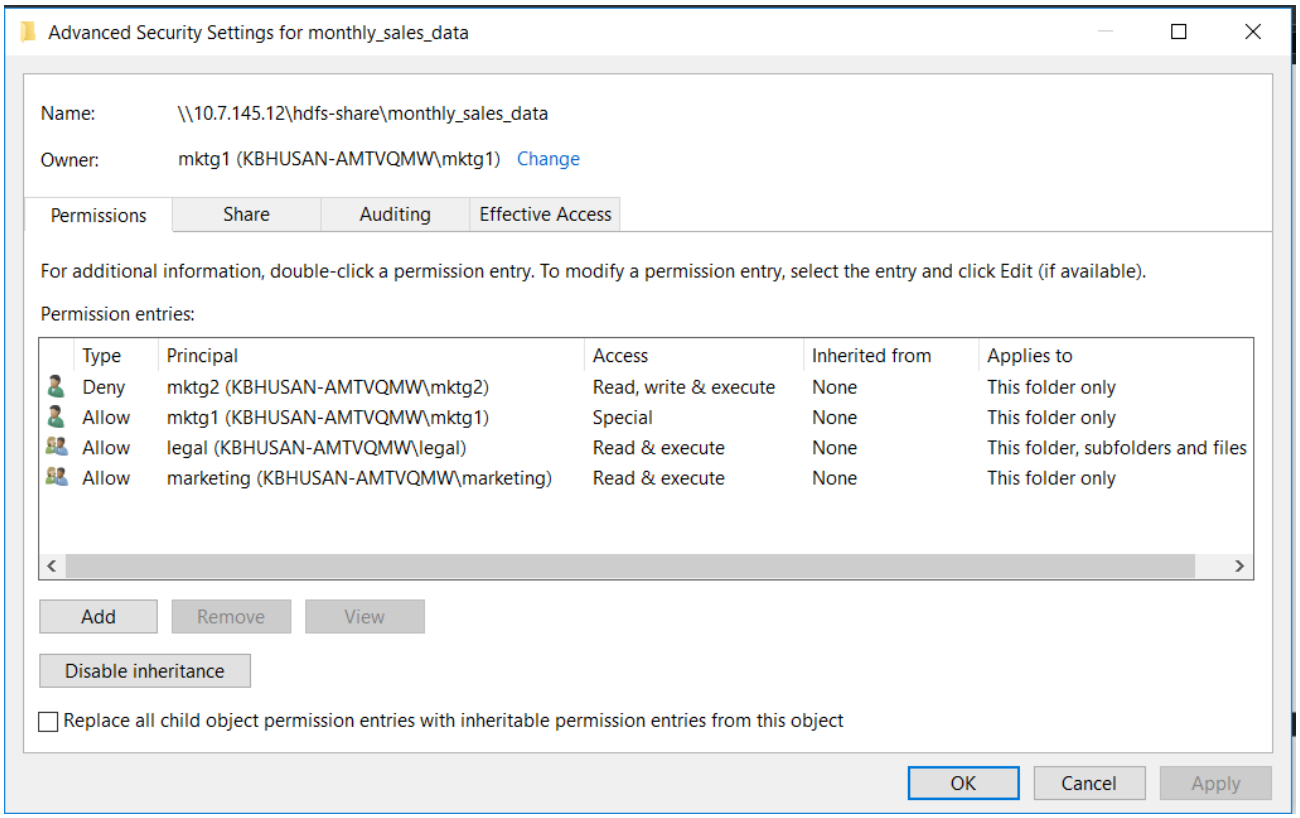

Figure 9 Isilon ACL update from Windows Explorer

Note: Advanced options can be used to set up ACLs. Refer to the document Access Control Lists on Dell EMC Isilon OneFS, section Mapping Windows permission to OneFS internal and POSIX section.

2.2 Mapping POSIX mode bits to OneFS internal ACE

Mapping POSIX mode bits to ACLs is simpler because the mode bits are a subset of the rich ACL model, so no security information is lost. The below table shows how OneFS processes the POSIX mode bits to be mapped to OneFS synthetic ACL permissions.

The following table represents the **rich** ACL of OneFS ACL inheritance flags.

Table 5 POSIX mode bits permission mapping

POSIX mode bits	OneFS internal ACE permissions	Windows Explorer option (reference)
r	dir_gen_read	Read
	file_gen_read	
w	dir_gen_write, delete_child, dir_read_attr	Write, Read attributes, Delete subfolders and files, Read permissions
	file_gen_write, file_read_attr	Write, Read attributes, Read permissions
x	dir_gen_execute, dir_read_attr	Traverse folder/execute file, Read attributes, Read permissions
	file_gen_execute, file_read_attr	

2.3 Isilon ACL usage and example

ACLs on Isilon help address access-related issues better than Permission Bits.
Consider an example in which /customer_data is created by the mktg1 user of the marketing group, and chmod 750 sets authoritative POSIX permissions on the directory so that only the mktg1 user controls all modification of customer_data, and other members of the marketing department can only view the customer_data; everyone else in the company outside marketing department cannot view the data.

Note: The examples shown in this section are similar to the Apache Hadoop ACLs examples described in appendix A, but on Isilon HDFS.

>drwxr-x--- - mktg1 marketing 0 2019-06-05 00:04 /customer_data

```
kbhusan-amtvqmw-1# ls -led customer_data
drwxr-x---    2 mktg1  marketing  0 Jun 18 11:42 customer_data
 OWNER: user:mktg1
 GROUP: group:marketing
 SYNTHETIC ACL
 0: user:mktg1 allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: group:marketing allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 10 Example customer_data POSIX permissions on Isilon

```
mktg1@kb-hdp1:~ $ hadoop fs -ls -d /customer_data
drwxr-x---   - mktg1 marketing          0 2019-06-18 11:21 /customer_data
mktg1@kb-hdp1:~ $
```

Figure 11 HDFS Client sees the same POSIX permissions on customer_data folder

As described in the section 2.1.1, Hadoop administrators who have full access to the SMB share on the Isilon HDFS Access Zone can use Windows Explorer to set the ACLs.

Figure 12 shows the customer_data folder Isilon OneFS ACL permission mapped to windows permission from Windows Explorer.

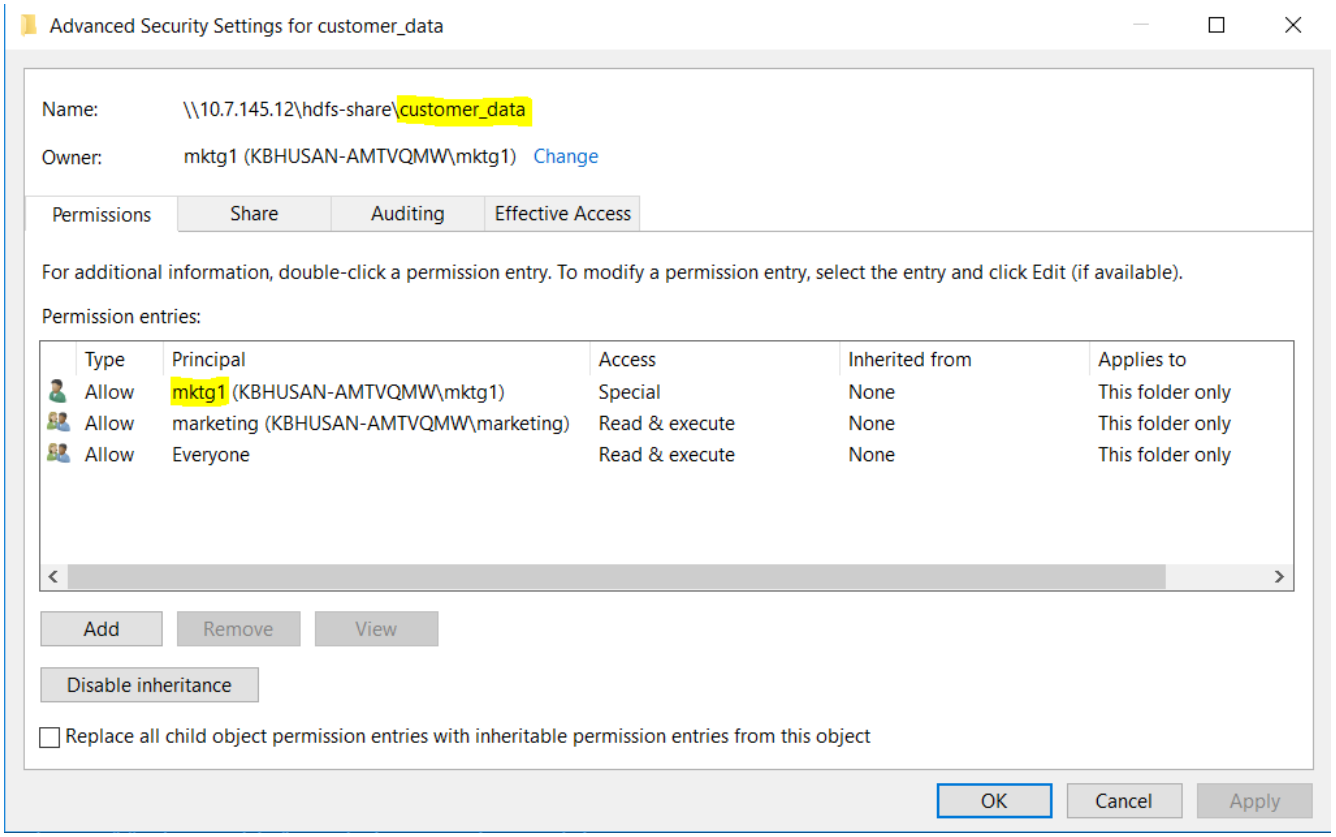

Figure 12 Isilon ACL mapped to Windows Permissions for customer_data

2.3.1 Example 1: Granting access to another named group

This example sets an Isilon ACL that grants read access to customer_data for members of the legal group.

2.3.1.1 Set the ACL: On Isilon CLI

```
kbhusan-amtvqmw-1# chmod +a# 1 group legal allow dir_gen_read,dir_gen_execute customer_data
kbhusan-amtvqmw-1# ls -led customer_data
drwxr-x--- +  2 mktg1  marketing  0 Jun 18 11:42 customer_data
 OWNER: user:mktg1
 GROUP: group:marketing
 0: user:mktg1 allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: group:legal allow dir_gen_read,dir_gen_execute
 2: group:marketing allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 13 Isilon ACL add named group

Additionally, the output of ls has been modified to append '+' to the permissions of a file or directory that has an ACL. The directory now has an authoritative OneFS ACL, not a POSIX permission.

From HDFS client

```
legal1@kb-hdp1:~ $ whoami
legal1
legal1@kb-hdp1:~ $ hadoop fs -ls -d /customer_data
drwxr-x---   - mktg1 marketing          0 2019-06-18 11:42 /customer_data
legal1@kb-hdp1:~ $
```

Figure 14 HDFS client access the Isilon ACL set permission

The new Isilon ACL entry is added to the existing permissions defined by the permission bits. User mktg1 has full control as the file owner. Members of either the marketing group or the legal group have read access. All others have no access from the HDFS protocol. The HDFS client has no visibility to the full OneFS ACL, but the ACL fully defines permissions on the directory and not the representative POSIX mode bits. The ACL can only be viewed from the Isilon CLI or Windows Explorer.

2.3.1.2 Set the ACL: On Windows Explorer

Hadoop administrators can set the similar permission using Windows Explorer. Figure 15 shows the legal group read permissions being set and Figure 16 shows the overall permission on the customer_data folder.

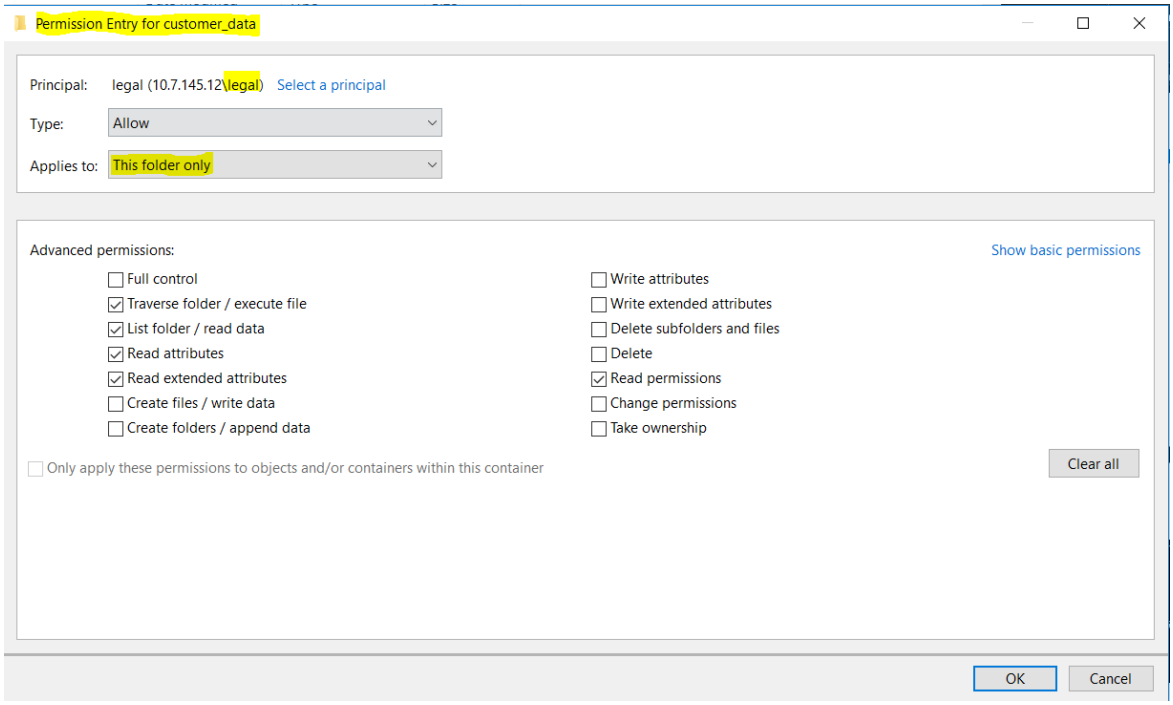

Figure 15 Windows permission mapping OneFS ACL permission for legal group

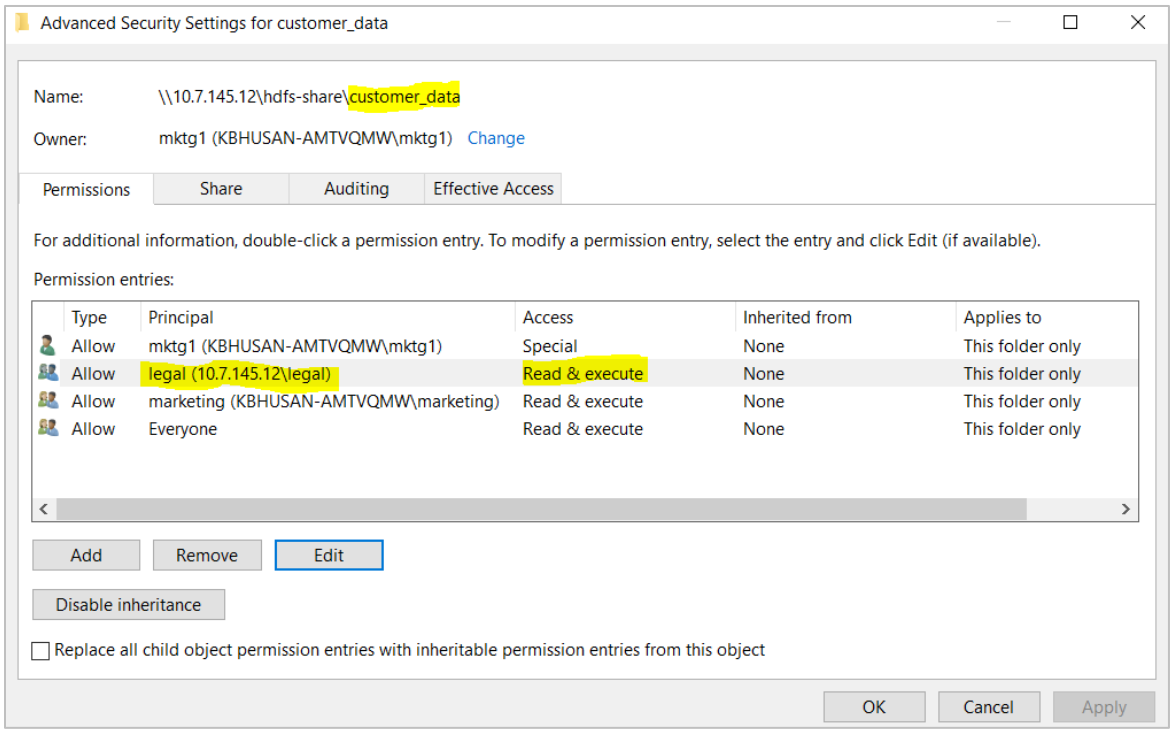

Figure 16 Read permissions set on legal group using Windows permissions

2.3.2 Example 2: Using a default ACL for automatic application to new children

Similar to the default ACL in Hadoop, Isilon has the concept of ACL inheritance described in the section 1.4. An ACL inheritance may be applied only to a directory, not a file. Inherited ACLs have no direct effect on permission checks and instead define the ACL that newly created child files and directories receive automatically.

Suppose we have a monthly-sales-data directory, further sub-divided into separate directories for each month. This example sets an ACL inheritance to guarantee that members of the legal group automatically get access to new sub-directories as they are created for each month.

```
kbhusan-amtvqmw-1# ls -led monthly_sales_data
drwxr-x---    2 mktg1  marketing  0 Jun 18 11:57 monthly_sales_data
 OWNER: user:mktg1
 GROUP: group:marketing
 SYNTHETIC ACL
 0: user:mktg1 allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: group:marketing allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 17 Example monthly_sales_data initial permissions

2.3.2.1 Set ACL Inheritance on parent directory: OneFS CLI

```
kbhusan-amtvqmw-1# chmod +a# 1 group legal allow dir_gen_read,dir_gen_execute,object_inherit,container_inherit monthly_sales_data
kbhusan-amtvqmw-1# ls -led monthly_sales_data
drwxr-x---  +  2 mktg1  marketing  0 Jun 18 11:57 monthly_sales_data
 OWNER: user:mktg1
 GROUP: group:marketing
 0: user:mktg1 allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: group:legal allow dir_gen_read,dir_gen_execute,object_inherit,container_inherit
 2: group:marketing allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 18 Isilon ACL Inheritance on monthly_sales_data

2.3.2.2 Set ACL inheritance on parent directory: Windows Explorer

Isilon ACL inheritance can also be set using Windows Explorer. Figure 19 shows the legal group added with inheritance by setting permission **Applies to: This folder, subfolder and files.** Figure 20 shows the overall permission set on the monthly_sales_data folder after adding the new group legal.

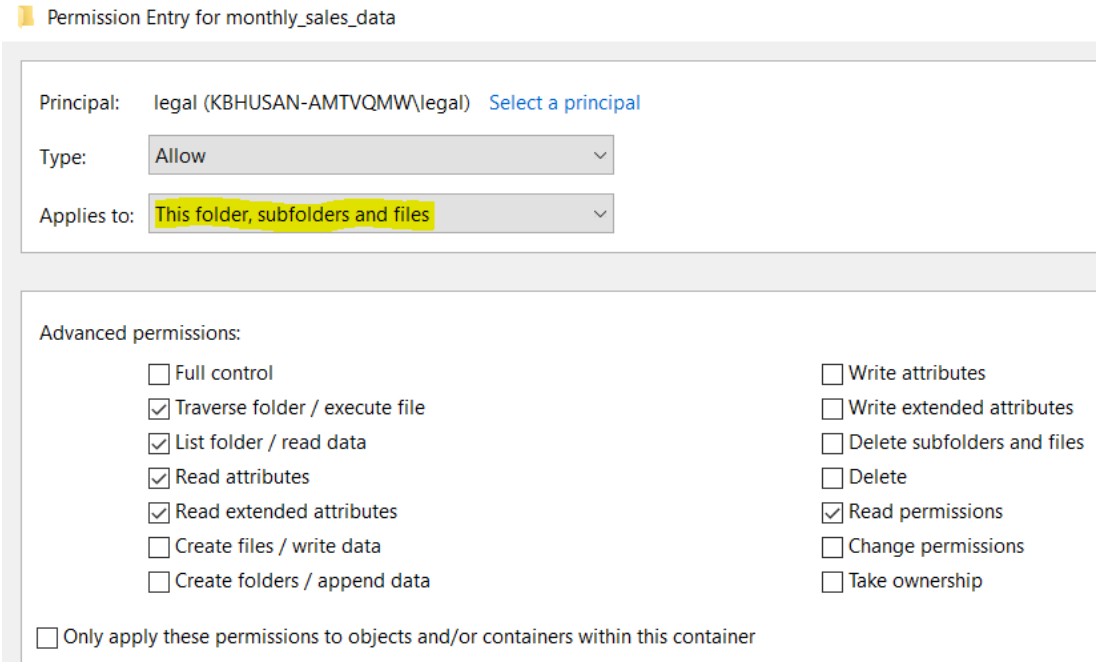

Figure 19 Windows permission mapping OneFS ACL permission for legal group

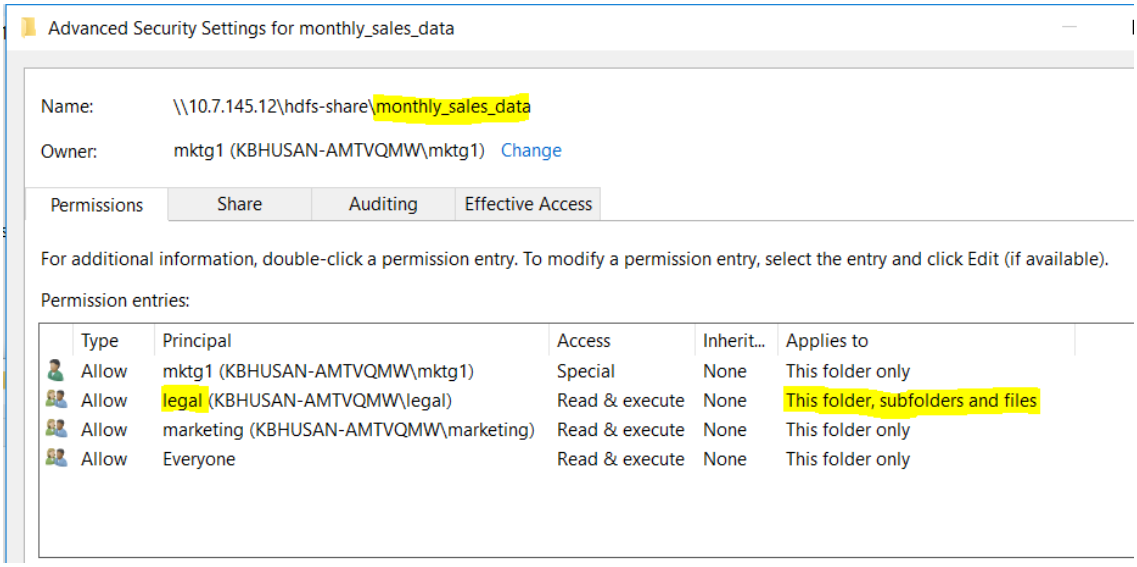

Figure 20 Overall Windows permssion on monthly_sales_data

Make sub-directories from HDFS client

```
mktg1@kb-hdp1:~ $ hadoop fs -mkdir /monthly_sales_data/JAN
mktg1@kb-hdp1:~ $ hadoop fs -mkdir /monthly_sales_data/FEB
mktg1@kb-hdp1:~ $ hadoop fs -ls /monthly_sales_data
Found 2 items
dr-xr-x---   - mktg1 marketing          0 2019-06-18 12:04 /monthly_sales_data/FEB
dr-xr-x---   - mktg1 marketing          0 2019-06-18 12:04 /monthly_sales_data/JAN
mktg1@kb-hdp1:~ $
```

Figure 21 HDFS mkdir commands on Isilon ACL inheritance set directory

Verify HDFS has automatically applied Isilon ACL inheritance to sub-directories

```
kbhusan-amtvqmw-1# ls -led monthly_sales_data/JAN
dr-xr-x--- +  2 mktg1  marketing  0 Jun 18 12:04 monthly_sales_data/JAN
 OWNER: user:mktg1
 GROUP: group:marketing
 CONTROL:dacl_auto_inherited,sacl_auto_inherited
 0: group:legal allow inherited dir_gen_read,dir_gen_execute,object_inherit,container_inherit,inherited_ace
kbhusan-amtvqmw-1# ls -led monthly_sales_data/FEB
dr-xr-x--- +  2 mktg1  marketing  0 Jun 18 12:04 monthly_sales_data/FEB
 OWNER: user:mktg1
 GROUP: group:marketing
 CONTROL:dacl_auto_inherited,sacl_auto_inherited
 0: group:legal allow inherited dir_gen_read,dir_gen_execute,object_inherit,container_inherit,inherited_ace
kbhusan-amtvqmw-1#
```

Figure 22 Isilon ACL inheritance applied to sub-directories

Verify legal group inherited read access to the sub-directories from the HDFS protocol

```
legal1@kb-hdp1:~ $ hadoop fs -ls -d /monthly_sales_data/JAN
dr-xr-x---   - mktg1 marketing          0 2019-06-18 12:04 /monthly_sales_data/JAN
legal1@kb-hdp1:~ $ hadoop fs -ls -d /monthly_sales_data/FEB
dr-xr-x---   - mktg1 marketing          0 2019-06-18 12:04 /monthly_sales_data/FEB
legal1@kb-hdp1:~ $ whoami
legal1
legal1@kb-hdp1:~ $
```

Figure 23 Verify Isilon ACL inheritance from hdfs protocol

The Isilon ACL inheritance is copied from the parent directory to the child file or child directory at time of creation. Subsequent changes to the parent directory's ACL inheritance do not alter the ACLs of existing children.

2.3.3 Example 3: Blocking access to a sub-tree for a specific user

Suppose there is an emergency need to block access to an entire sub-tree for a specific user. Applying a named-user ACL entry to the root of that sub-tree is the fastest way to accomplish this without accidentally revoking permissions for other users.

2.3.3.1 Set ACL deny: OneFS CLI

Add an Isilon ACL entry to deny all access to monthly_sales_data by user mktg2.

```
kbhusan-amtvqmw-1# chmod +a# 1 user mktg2 deny dir_gen_read,dir_gen_write,dir_gen_execute monthly_sales_data
kbhusan-amtvqmw-1# ls -led monthly_sales_data
drwxr-x--- +  4 mktg1  marketing  42 Jun 18 12:04 monthly_sales_data
 OWNER: user:mktg1
 GROUP: group:marketing
 0: user:mktg1 allow dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,delete_child
 1: user:mktg2 deny dir_gen_read,dir_gen_write,dir_gen_execute
 2: group:legal allow dir_gen_read,dir_gen_execute,object_inherit,container_inherit
 3: group:marketing allow dir_gen_read,dir_gen_execute
kbhusan-amtvqmw-1#
```

Figure 24 Isilon ACL deny user

Verify HDFS does not let user mktg2 access monthly_sales_data directory

```
[mktg2@krb-client1 ~]$ whoami
mktg2
[mktg2@krb-client1 ~]$ hadoop fs -ls /monthly_sales_data
ls: Permission denied: user=mktg2, access=READ_EXECUTE, inode="/monthly_sales_data":mktg1:marketing:drwxr-x---
[mktg2@krb-client1 ~]$
```

Figure 25 Verify Isilon ACL deny case from hdfs protocol

2.3.3.2 Set ACL deny: Windows Explorer

Isilon ACL deny can also be set using Windows Explorer. Figure 26 shows the user mktg2 permissions denied in Windows Explorer by setting the permissions **Type: Deny** and **Applies to: This folder only**. Figure 27 shows overall permission set on the monthly_sales_data folder after the mktg2 user is denied access permission.

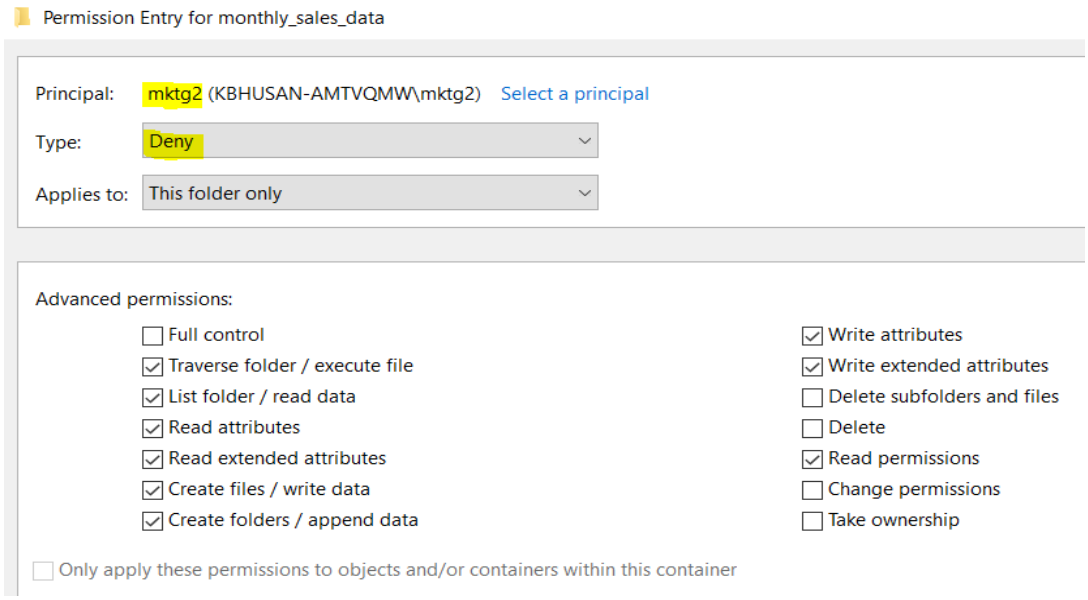

Figure 26 Windows permission mapping OneFS ACL to deny for mktg2 user

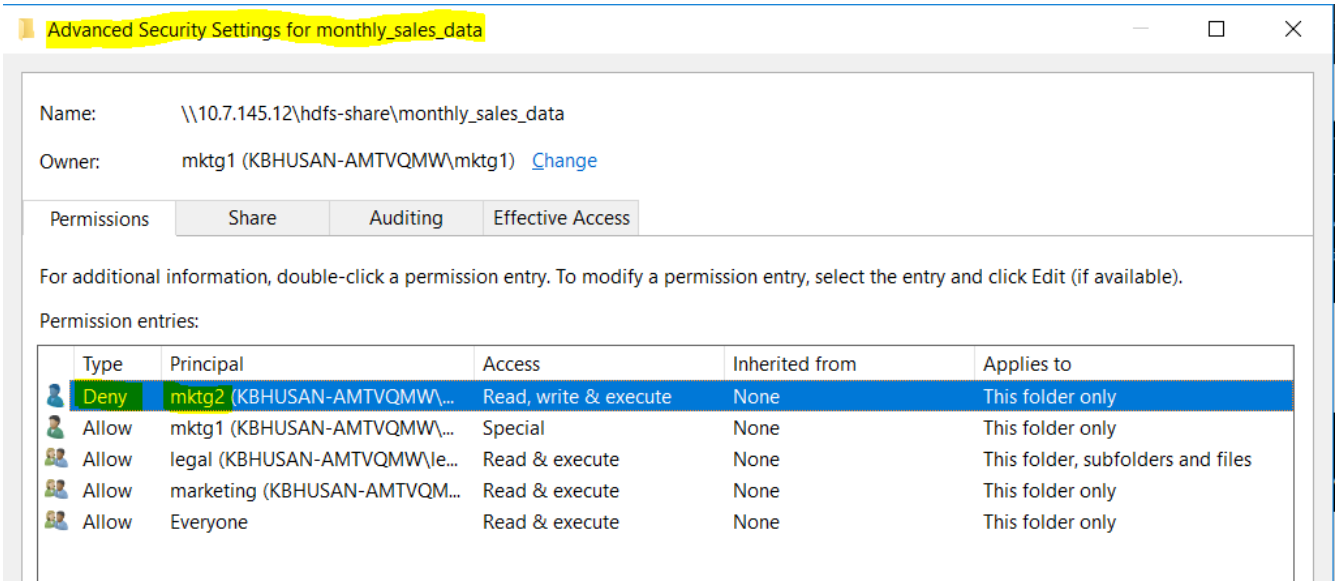

Figure 27 Overall Windows permssion on monthly_sales_data after mktg2 user set to deny

3 Apache HDFS ACLs

3.1 Apache HDFS ACLs

ACLs extend the HDFS permission model to support more granular file access based on combinations of users and groups. This allows for fine-grained permissions for HDFS files in Hadoop.

Securing any system requires implementing layers of protection. ACLs are typically applied to data to restrict access to data to approved entities. Application of ACLs at every layer of access for data is critical to secure a system.

In general, plain Unix permissions are not sufficient when permission requirements do not map cleanly to an enterprise's natural hierarchy of users and groups. The HDFS ACLs feature addresses this shortcoming. HDFS ACLs is available in Apache Hadoop 2.4.0 and later versions.

HDFS ACLs give the ability to specify fine-grained file permissions for specific named users or named groups, not just the file's owner and group. HDFS ACLs are modeled after POSIX ACLs. The best practice is to rely on traditional permission bits to implement most permission requirements and define a smaller number of ACLs to augment the permission bits with a few exceptional rules.

3.2 Configuring ACLs on HDFS

You must configure **dfs.namenode.acls.enabled** in **hdfs-site.xml** to enable ACLs on HDFS.

To use ACLs, this requires enabling ACLs on the NameNode by adding the following configuration property to hdfs-site.xml and restarting the NameNode.

```
<property>
<name>dfs.namenode.acls.enabled</name>
<value>true</value>
</property>

Or via Cloudera Manager or Ambari to set and configure this property
```

3.3 ACL command usage

You can use the sub-commands **setfacl** and **getfacl** to create and list ACLs on HDFS. These commands are modeled after the same Linux shell commands.

3.3.1 setfacl

Sets ACLs for files and directories.

Usage

```
-setfacl [-bkR] {-m|-x} <acl_spec> <path>
-setfacl --set <acl_spec> <path>
```

Table 6 SETACL options

Options	Description
-b	Remove all entries but retain the base ACL entries. The entries for User, Group, and Others are retained for compatibility with Permission Bits.
-k	Remove the default ACL.
-R	Apply operations to all files and directories recursively.
-m	Modify the ACL. New entries are added to the ACL, and existing entries are retained.
-x	Remove the specified ACL entries. All other ACL entries are retained.
--set	Fully replace the ACL and discard all existing entries. The acl_spec must include entries for User, Group, and Others for compatibility with Permission Bits.
<acl_spec>	A comma-separated list of ACL entries.
<path>	The path to the file or directory to modify.

3.3.2 getacl

This displays the ACLs of files and directories. If a directory has a default ACL, getfacl also displays the default ACL.

Usage

```
-getfacl [-R] <path>
```

Table 7 GETACL options

Options	Description
-R	List the ACLs of all files and directories recursively.
<path>	The path to the file or directory to list.

A HDFS ACL examples

ACLs on HDFS help in addressing access-related issues better than permission bits.

Consider an example in which /customer_data is created by the mktg1 user of the marketing group, and chmod 640 is set on the directory so only the mktg1 user controls all modification of customer_data, and other members of the marketing department can only view the customer_data; everyone else in the company outside marketing department cannot view the data.

```
>drw-r-----   - mktg1 marketing          0 2019-06-05 00:04 /customer_data
```

A.1 Example 1: Granting access to another named group

This example sets an ACL that grants read access to customer_data for members of the legal group.

Set the ACL:

```
> hdfs dfs -setfacl -m group:legal:r-- /customer_data
```

Check the results by running getfacl:

```
> hdfs dfs -getfacl /customer_data
# file: /customer_data
# owner: mktg1
# group: marketing
user::rw-
group::r--
group:legal:r--
mask::r--
other::---
```

Additionally, the output of ls has been modified to append '+' to the permissions of a file or directory that has an ACL.

```
> hdfs dfs -ls -d /customer_data
drw-r-----+  - mktg1 marketing          0 2019-06-05 00:04 /customer_data
```

The new ACL entry is added to the existing permissions defined by the permission bits. User mktg1 has full control as the file owner. Members of either the marketing group or the legal group have read access. All others have no access.

A.2 Example 2: Using a default ACL for automatic application to new children

In addition to an ACL being enforced during permission checks, there is also a separate concept of a default ACL. A default ACL may be applied only to a directory, not a file. Default ACLs have no direct effect on permission checks and instead define the ACL that newly created child files and directories receive automatically.

Suppose there is a monthly-sales-data directory, further sub-divided into separate directories for each month. This example sets a default ACL to guarantee that members of the legal group automatically get access to new sub-directories, as they get created for each month.

Set the default ACL on the parent directory:

```
>hdfs dfs -setfacl -m default:group:legal:r-x /monthly_sales_data
```

Make sub-directories:

```
> hdfs dfs -mkdir /monthly-sales-data/JAN
> hdfs dfs -mkdir /monthly-sales-data/FEB
```

Verify that HDFS has automatically applied the default ACL to the sub-directories:

```
> hdfs dfs -getfacl -R /monthly_sales_data
# file: /monthly_sales_data
# owner: mktg1
# group: marketing
user::rwx
group::r-x
other::---
default:user::rwx
default:group::r-x
default:group:legal:r-x
default:mask::r-x
default:other::---

# file: /monthly_sales_data/FEB
# owner: mktg1
# group: marketing
user::rwx
group::r-x
group:legal:r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:legal:r-x
default:mask::r-x
default:other::---
```

```
# file: /monthly_sales_data/JAN
# owner: mktg1
# group: marketing
user::rwx
group::r-x
group:legal:r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:legal:r-x
default:mask::r-x
default:other::---
```

The default ACL is copied from the parent directory to the child file or child directory at time of creation. Subsequent changes to the parent directory's default ACL do not alter the ACLs of existing children.

A.3 Example 3: Blocking access to a sub-tree for a specific user

Suppose there is an emergency need to block access to an entire sub-tree for a specific user. Applying a named user ACL entry to the root of that sub-tree is the fastest way to accomplish this without accidentally revoking permissions for other users.

Add the ACL entry to block all access to monthly_sales_data by the user mktg2:

```
> hdfs dfs -setfacl -m user:mktg2:--- /monthly_sales_data
```

Check the results by running getfacl:

```
> hdfs dfs -getfacl /monthly_sales_data
# file: /monthly_sales_data
# owner: mktg1
# group: marketing
user::rwx
user:mktg2:---
group::r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:legal:r-x
default:mask::r-x
default:other::---
```

It is important to keep in mind the order of evaluation for ACL entries when a user attempts to access a file system object:

- If the user is the file owner, then the owner permission bits are enforced.
- Else, if the user has a named user ACL entry, then those permissions are enforced.
- Else, if the user is a member of the file's group or any named group in an ACL entry, then the union of permissions for all matching entries are enforced. (The user may be a member of multiple groups.)

If none of the above are applicable, then the other permission bits are enforced.

In this example, the named user ACL entry accomplished the goal because the user is not the file owner, and the named user entry takes precedence over all other entries.

B Technical support and resources

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.

See the following additional resources for more information:

- OneFS Technical Overview
- OneFS 8.1.0 CLI Administration Guide
- OneFS 8.1.0 CLI Command Reference
- OneFS 8.1.0 Web Administration Guide
- OneFS 8.1.0 Security Configuration Guide
- Apache HDFS ACLs
- Access Control Lists on Dell EMC Isilon OneFS