Deployment and Configuration

# Dell EMC Isilon: Apache Ranger Setup and Operations

Setup, policy configuration, and migration from a direct-attached storage (DAS) Apache Hadoop cluster


## Abstract

This document describes the official support of Apache® Ranger™ as a part of an Apache Hadoop® deployment with a Dell EMC™ Isilon™ cluster. Also discussed are best practices, implementation strategies, and limitations.



August 2019



H17910

# Revisions

| Date | Description |
|---|---|
| August 2019 | Initial release |

# Acknowledgements

This paper was produced by the following:

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

Support: Venkat Vyasam (Venkat.Vyasam@dell.com), Nicholas Ruggiero (Nicholas.Ruggiero@dell.com )

# Table of contents

# Executive summary

Apache® Ranger™ is a centralized management console that enables you to monitor and manage data security across the Hortonworks® distribution system for Apache Hadoop®. A Ranger administrator can define and apply authorization policies across Hadoop components including the Hadoop Distributed File System (HDFS).

HDFS policies that are defined in Ranger are checked before the native file access control is applied. This two-layered authorization model differs in the way the standard Ranger HDFS policies are checked with directed-attached storage (DAS), but the model is suitable for using Dell EMC Isilon OneFS™ as a multiprotocol data lake with Hadoop. The OneFS native file system ACL allows a storage administrator to correctly set up access control for multiple workloads and with multiprotocol access to the HDFS dataset. A Ranger administrator can apply a further restrictive Hadoop user access control to the same HDFS dataset, thus providing the administrators the appropriate control span within their management domains.

In an Isilon OneFS cluster with a Hadoop deployment, Ranger authorization policies serve as a filter before applying the native file access control.

This document explains how to configure Apache Ranger and set up policies and strategies to migrate Ranger policies from a Hadoop cluster on DAS to a Hadoop deployment on an Isilon cluster.

# 1 OneFS Ranger integration

This section discusses security control for multi-protocol workloads. This authorizes specific users to access HDFS files.

The key features and benefits of OneFS and Ranger integration include the following:

- Enables Ranger authorization policies to be executed in OneFS
- Allows OneFS native file access control to be effective
- Provides dual access-control checks to guarantee that file access meets both Hadoop and IT administrator needs
- Helps the Ranger administrator to enforce Hadoop access policies across all Hadoop components consistently
- Allows the IT or data-center administrator to maintain control of multiprotocol data lake access in OneFS

## 1.1 Isilon and Ranger administration

IT and Ranger administration can be compared as follows:

- IT administration:
  - Manages user access for all workloads on Isilon
  - Enables centralized ACL for all protocols
- Ranger administration:
  - Has more restrictive control over data access on HDFS dataset
  - Cannot override IT admin control

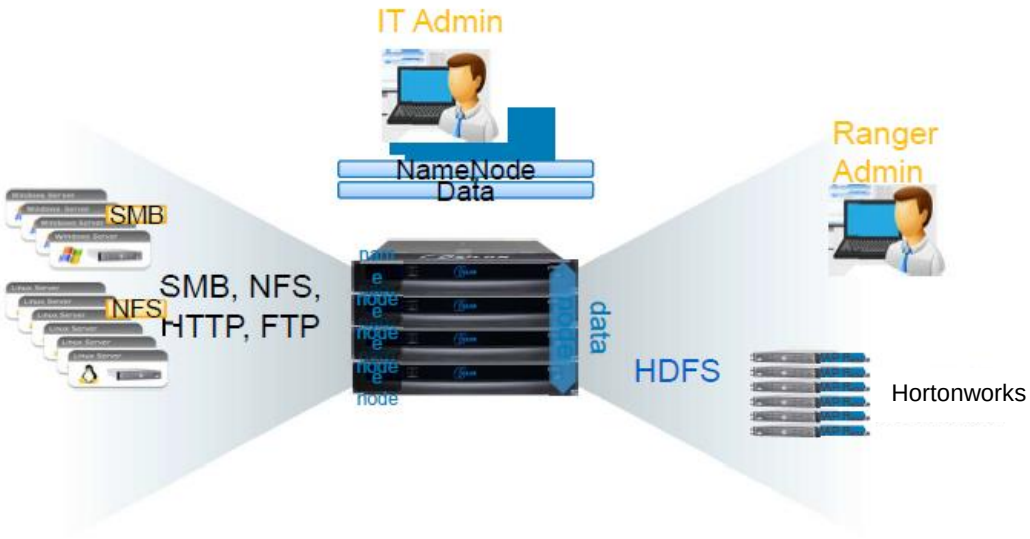



Figure 1 OneFS Ranger intergration

# 2 Apache Ranger support

OneFS supports Apache Ranger as part of a Hadoop deployment with an Isilon cluster.

The Apache Ranger console provides a centralized security framework to manage access control over Hadoop data access components such as Apache Hive™ and others. These policies can be set for both individual users or groups and then enforced consistently on files, folders, and databases.

Ranger HDFS authorization policies with deny conditions are supported by OneFS. This is the simplest form of Ranger integration and the easiest to implement for new installations. Documentation for Apache JIRA RANGER-606 describes how to use deny conditions, which were added to Apache Ranger 0.6.0. For more information on Apache Ranger and specific HDP components, refer to the Apache Ranger pages on the Hortonworks site.

- AD, Kerberos, and local authentication are supported.
- One-way SSL with Ranger policy server is supported with MIT KDC- OneFS 8.1.2.
- One-way SSL with Ranger policy is supported with AD – OneFS 8.2.
- Apache Ranger audit of HDFS access is not currently supported.
- Tag policies are not currently supported.

## 2.1 Editing Apache Ranger HDFS plugin settings

You can enable the Apache Ranger HDFS plugin to allow additional oversight of HDFS protocol authentication using either the OneFS web administration interface or the command-line interface (CLI).

To do this, enable Apache Ranger on Isilon clusters and then check for new authorization policies, receive HDFS requests from clients, and apply authorization policies to the HDFS requests, which can be one of DENY, ALLOW, or UNDETERMINED. Enable the Apache Ranger HDFS plugin using the steps that are outlined in the Hortonworks Security Guide.

Enabling the Apache Ranger plugin allows the authorization policies that are defined in the Ranger HDFS service instance, also called a repository, prior to Apache Ranger 0.6.0. **The policies must first allow users or groups access to resources and then deny specific users or groups from access. If a user is not included in the allow list, they are denied access by default.** For more information about creating a DENY policy, see Apache Ranger deny policies with OneFS 8.0.1.0

**Note**: A poorly formed policy can have an unintended impact, for example, blocking access.

The repository name is a setting within Apache Ranger. The minimum supported version of Apache Ranger is 0.6.0 because the Ranger DENY policy is supported only in 0.6.0 and later versions. In version 0.6.0, Apache Ranger changed the name of this feature to service instance. The service instance is the name of the HDFS service instance within the Apache Ranger Admin UI used as the repository name.

If you have a Kerberos-enabled cluster, follow the instructions in the Hortonworks Security Guide to enable the Ranger HDFS plugin on the cluster.

### 2.1.1 Edit Apache Ranger HDFS plugin settings (Web UI)

Enable the Apache Ranger HDFS plugin using the OneFS web administration interface.

Locate the policy manager URL on the Ambari server at **Ambari > Ranger > Configs** as the **policymgr_external_url**. This URL is created by combining http://, followed by the host name where Ranger Admin is installed, followed by the *ranger.service.http.port* , which is usually 6080, followed by /.

1. Click **Protocols** > **Hadoop (HDFS) > Ranger Plugin Settings**.
2. In the **Ranger Plugin** settings area, select **Enable Ranger Plugin**.
3. In the **Policy manager URL** field, type the URL that points to the location of the Policy Manager.
4. In the **Repository name** field, type the name of the HDFS repository.
5. Click **Save Changes**.

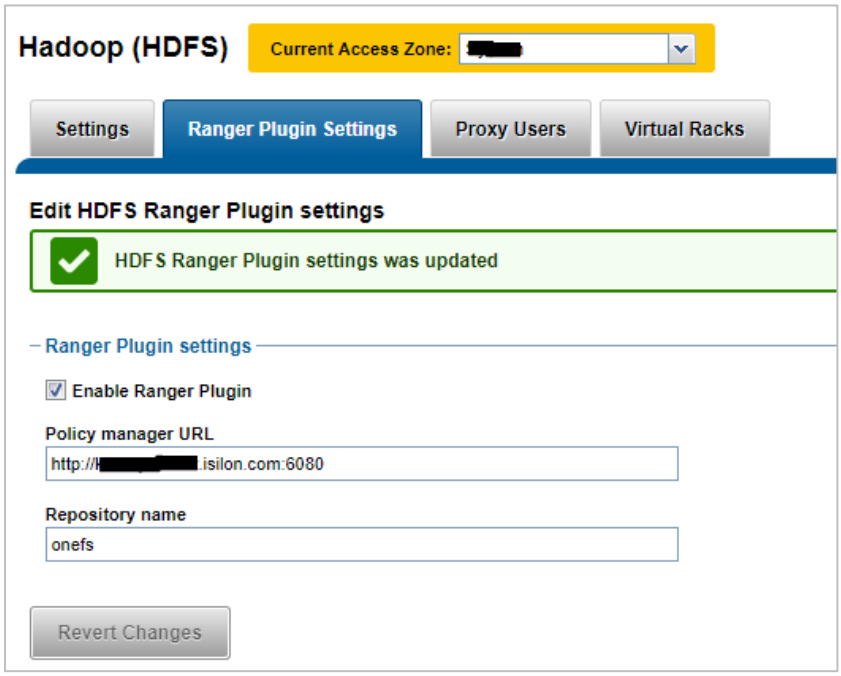



Figure 2 OneFS Web UI Ranger plugin setup

## 2.1.2 Edit Apache Ranger HDFS plugin settings (CLI)

To configure Ranger plugin settings, run the **isi hdfs ranger-plugin settings modify** command.

The **--policy-manager-url** is created by combining **http://,** followed by the hostname where Ranger Admin is installed, followed by the **ranger.service.http.port** , which is usually **6080,** followed by **/**.

The following command configures the Ranger plugin settings:

```
isi hdfs ranger-plugin settings modify --policy-manager-url=http://<ranger sever
FQDN>:6080
isi hdfs ranger-plugin settings modify --repository-name=<ranger service name>
isi hdfs ranger-plugin settings modify --enabled=true
```

```
kbhusan-amtvqmw-1# isi hdfs ranger-plugin settings view
          Enabled: Yes
Policy Manager Url: http://[redacted].isilon.com:6080
   Repository Name: onefs
kbhusan-amtvqmw-1#
```

Figure 3 OneFS CLI Ranger plugin setup

## 2.2 Enabling DENY conditions for policies

Hortonworks HDP 2.6.5 and previous versions come with Ranger 0.7.0 or lower versions which has the DENY condition for policies disabled by default. This can be enabled by setting **ranger.servicedef.enableDenyAndExceptionsInPolicies=true** under the Custom **ranger-admin-site** configuration in the Ambari UI.

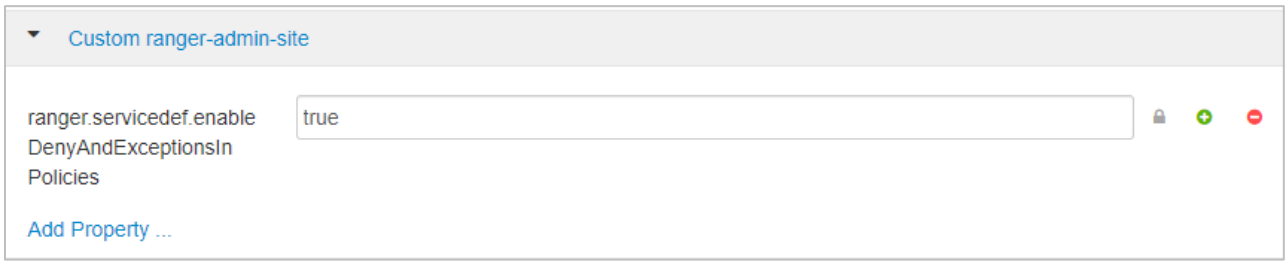



Figure 4 Enable DENY Policy in Ambari UI

**Note**: The Ranger version above (0.7.0) has DENY conditions enabled by default.

## 2.3 Configuring Isilon Ranger SSL

Isilon 8.1.2 implements one-way SSL with Kerberos (MIT KDC). This is accomplished by enabling Kerberos authentication and SPNEGO for Ranger Policy Server. Below are the steps to enable Ranger SSL on Isilon.

1. Add new service principal names for HTTP (SPNEGO/krb ticket) either using the OneFS web administration interface or by running the following commands through an SSH client.

   ```
   isi auth krb5 spn create --provider-name=$REALM --
   spn=HTTP/$RANGER_SERVER_FQDN@$REALM --user=$admin_principal --
   password=$admin_password

   This SPN may have been created by Ambari during the kerberization process
   ```

2. Configure the Ranger policy URL on the Isilon cluster node.

   ```
   isi hdfs ranger-plugin settings modify --policy-manager-
   url=https://<ranger sever FQDN>:6182
   isi hdfs ranger-plugin settings modify --repository-name=<ranger service
   name>
   isi hdfs ranger-plugin settings modify --enabled=true
   ```

Isilon 8.2.0 implements one-way SSL with Kerberos AD this is accomplished by enabling Kerberos authentication and SPNEGO for Ranger Policy Server and configuring AD computer object as an account during the HDFS Service Manager Setting described in the section 2.5.

Perform the following steps during the HDFS service manager setup for implementing SSL with AD.

1. Add the Isilon AD Computer Account object as an account that can download the policy.

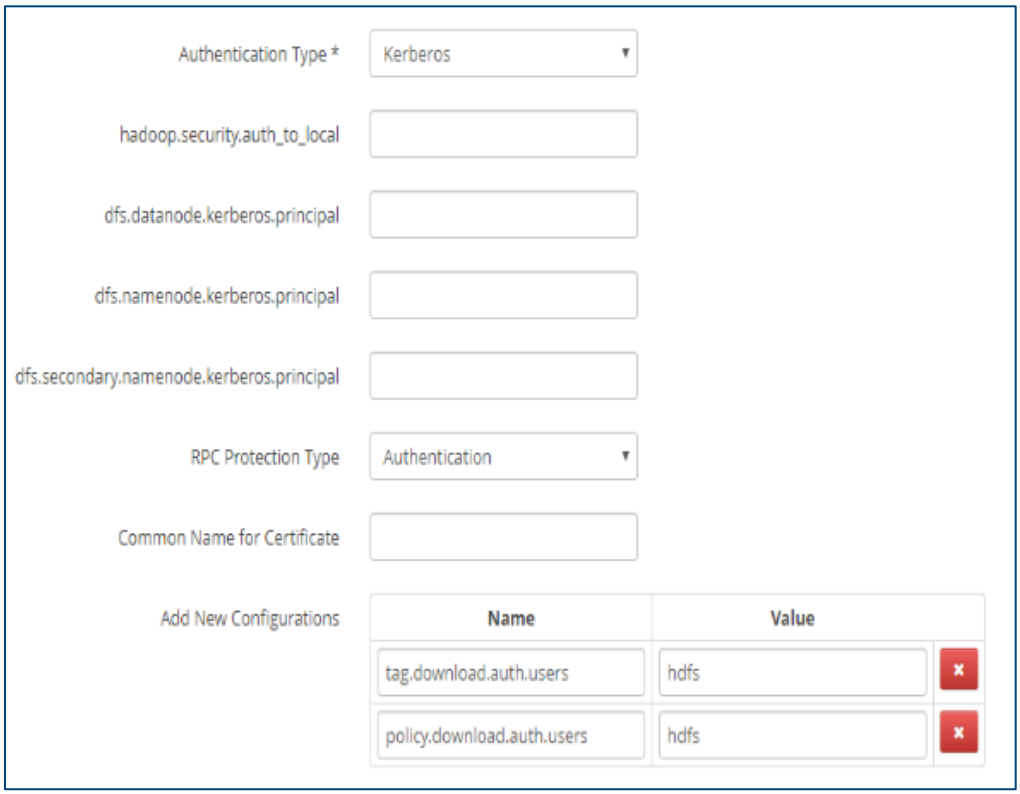



Figure 5 Add AD computer object as an account in the HDFS Service Manager setting

In this example, the AD computer object is FOO-HDP27$, and the $ is needed.

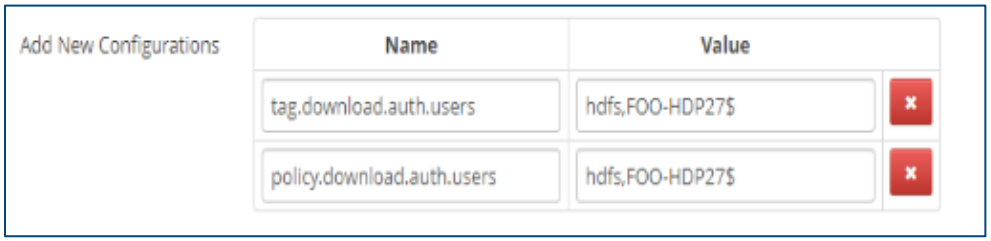



Figure 6 AD computer object added in the Ranger HDFS Service Manager Settings

Figure 7 OneFS HDFS log error if AD computer object is missing

2. If the Isilon system is unable to download the policy with 401 errors, review the Ranger Access Log:

```
cat /var/log/ranger/admin/xa_portal.log
```

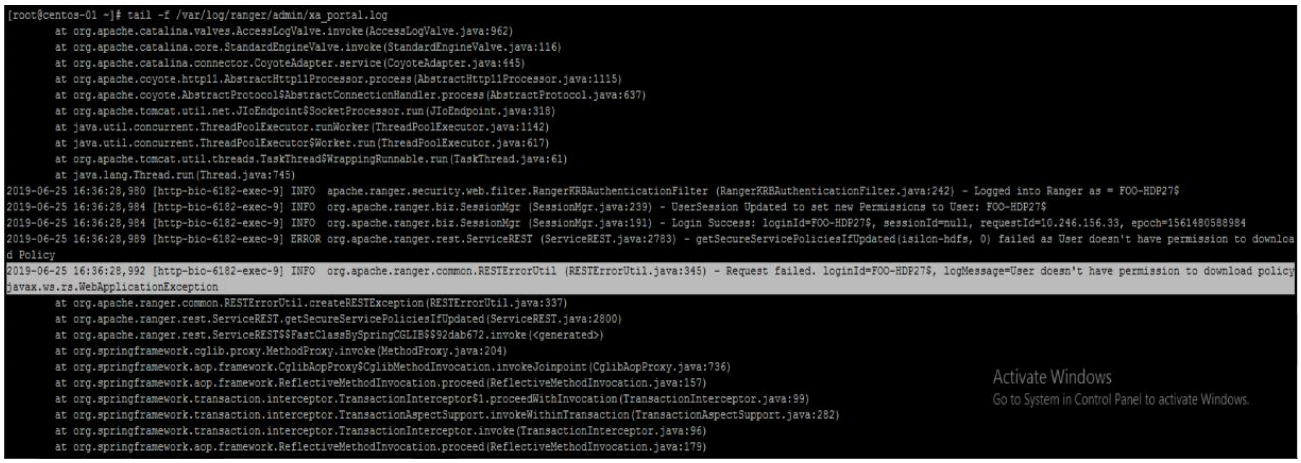



Figure 8 AD Computer Object permission error in Ranger logs

3. Upon granting access to the Isilon computer object to download the policy, the access log will show the following:

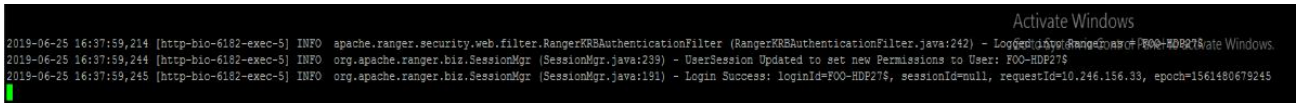



Figure 9 OneFS HDFS log with AD computer object login success

4. The Isilon system will show the following on changes to the Ranger policy.

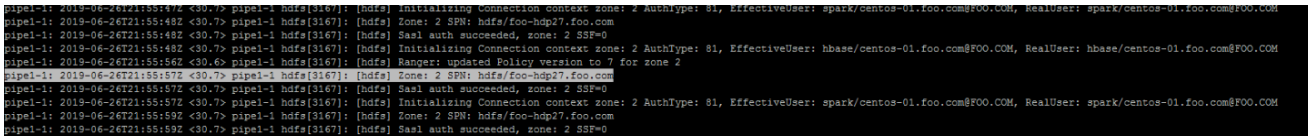



Figure 10 OneFS HDFS log successful Ranger HDFS service setting message

## 2.4 Providing authorization with Apache Ranger

The Service Manager for Resource Based Policies page is displayed when you log in to the Ranger Console. You can use this page to create services for Hadoop resources (such as HDFS, KMS, or Hive) and add access policies to those resources.

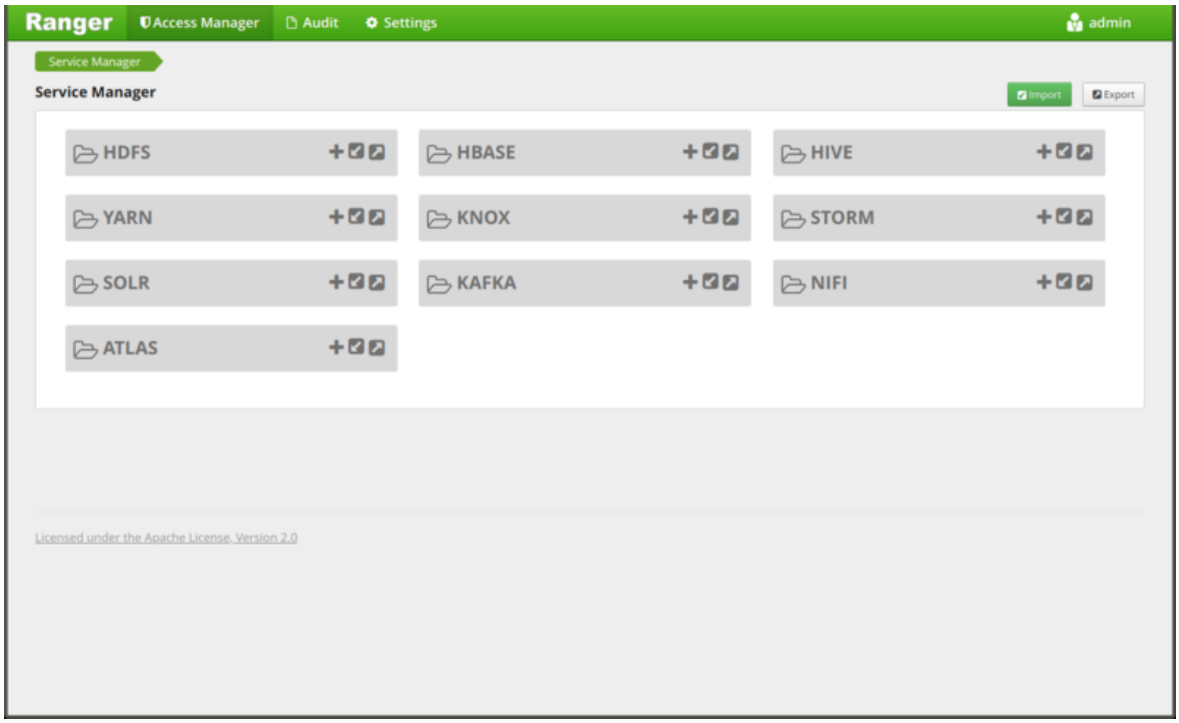



Figure 11 Apache Ranger Web UI

Click **Access Manager** in the top menu to open the **Service Manager for Resource Based Policies** page and display a submenu with links to **Resource Based Policies**, **Tag Based Policies**, and **Reports**. OneFS supports only resource-based policies.

Click **Access Manager > Resource Based Policies** to open the Service Manager for Resource Based Policies page. You can use this page to create services for resources (such as HDFS, KMS, or Hive) and add access policies to those services.

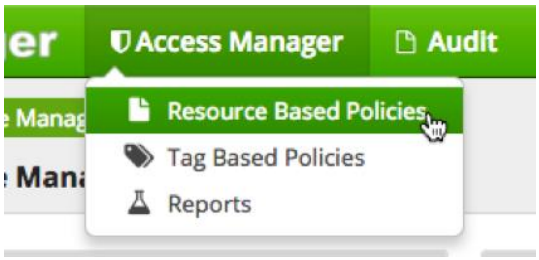



Figure 12 Resource Based Policies

## 2.5 HDFS Service Manager setting

Create a new service instance for OneFS using the **Create Service** page in Ranger. See the Hortonworks Security Guide for details. Specify values in the following fields to create a new service instance for OneFS and make note of the values:

- Specify a value in the Service Name field and make note of it because you must use the same value in OneFS.
- Specify a superuser or admin username and password in the Config Properties section specific to the service instance. The Test Connection option will continue to fail until you have saved and reopened the service instance.

- Specify the Namenode URL as FQDN: hdfs://onefs.smartconnect.name:8020

Once the HDFS service manager is setup, create a default policy all – path or root path - /*.

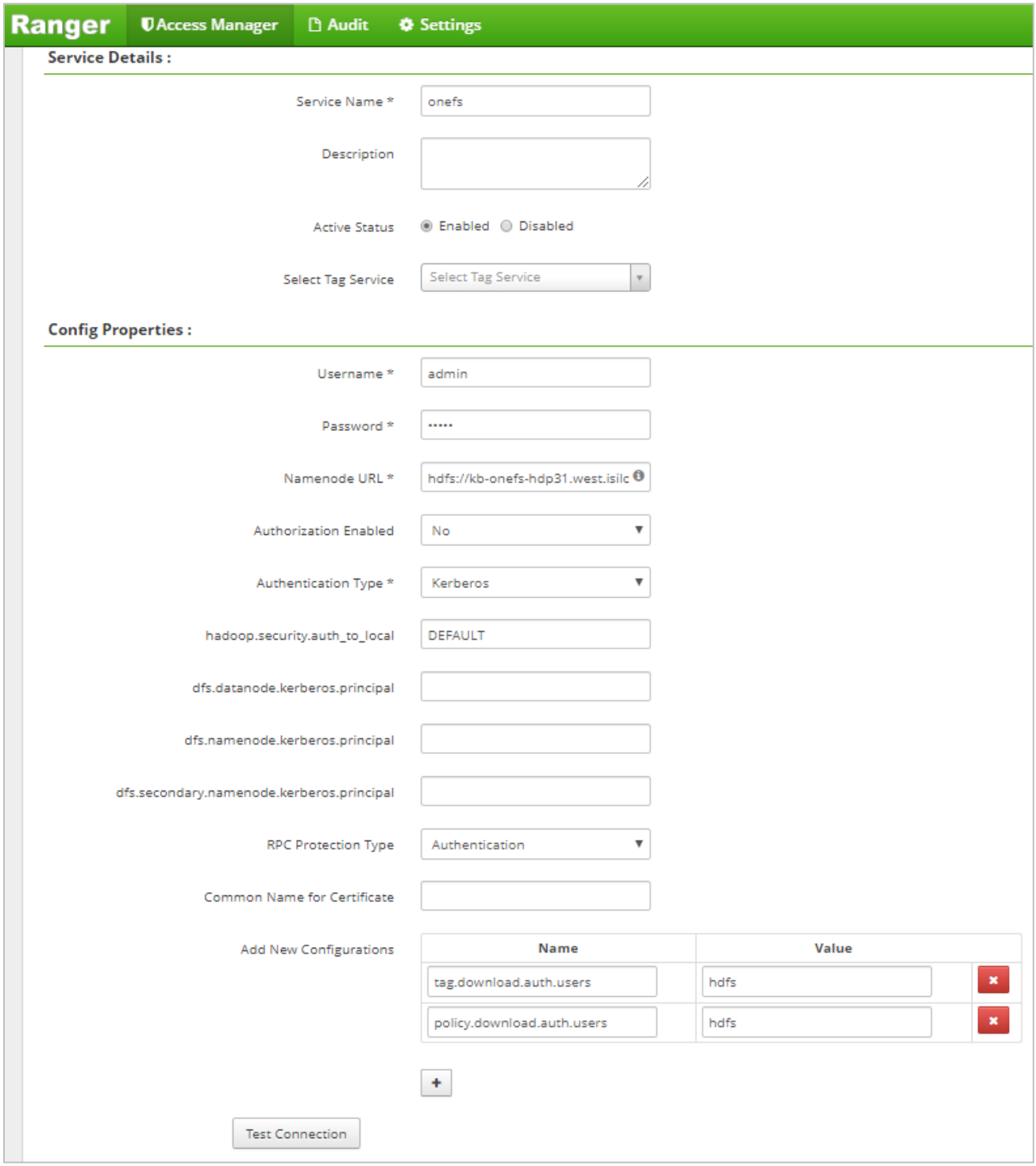



Figure 13 HDFS Service setting

# 3 Isilon Ranger policy setup, prerequisites, and best practices

The Hortonworks website recommends using chmod 000 on all the files to provide a security failsafe if Ranger fails. Performing this on the Isilon system blocks all access. This leads to the other integration options described in detail in this section, and each has their own advantages and limitations that should be considered before policy setup.

## 3.1 Prerequisites

1. Make sure ranger and rangeradmin users are created on all the cluster nodes including Isilon.
2. Set NAMENODE_URL for each Ranger-* plugins.
3. Create default policy all-path if not created already, granting access to all the files for the user that you include in the service details page.
4. Add all groups and individual users associated with an access zone within OneFS to the default policy(all-path) in order to grant access to the groups and users.
5. If you create local users in the OneFS, or use Active Directory, you must change the UserSync settings in Ambari or add the users in the Ranger interface.
6. If Ranger SSL is enabled, make sure necessary spn and AD computer objects are configured.

**Note**: OneFS file system permissions take precedence even if the policy indicates that the user or group can access everything.

## 3.2 Deployment best practices

Ranger with Isilon supports a stacked authorization model. Ranger policies will be checked in conjunction with OneFS file system access control. **Ranger policies for file access are always validated first, and then OneFS file system access control is checked, being the ultimate governing body**. The Ranger policy behavior can be differentiated in the following table.

Table 1 Results of authorization check for Ranger policies

| Implementation | Allow policy | Deny policy | Undetermined policy |
|---|---|---|---|
| Ranger on DAS Hadoop | Allowed | Denied | Check HDFS POSIX |
| Ranger on Isilon Hadoop | Check OneFS ACL | Denied | Denied |

There are three methods to implement the Ranger Policies on Isilon storage. These are Dell EMC recommendations and supported configurations of policies. Evaluate each method and adopt one for your environment, since each have their own advantages and limitations. Below are the methods listed.

- Method 1: Public group X / R-X and ALLOW condition policies
- Method 2: Public group RWX and DENY condition policies

**Note:** In the OneFS cluster with HDFS, permission checking is based on the on-disk OneFS internal permission, either POSIX bits, or the OneFS ACL. In the above method, POSIX RWX bits represent Read, Write, and Execute permission bits.

### 3.2.1 Method 1: Public group X/R-X and ALLOW/DENY condition policies

In this approach, service accounts are allowed access to their respective paths and everyone else can only traverse the tree but cannot read unless allowed by the Ranger policy. All policies are created and managed in Ranger with appropriate POSIX permission to the folder. **By default, Isilon storage starts with a deny policy, meaning there is already no access to anyone unless explicitly provided by POSIX and Ranger Policies.** To assign and manage privileges, the Hadoop folders are given 777 permissions and set ALLOW and DENY condition policies.

**Note:** With the Hadoop on DAS implementation, local files are set to 000 permissions as a Ranger failsafe. On Isilon storage, there is no Ranger failsafe so extra precaution needs to be taken for multi-protocol access.

#### 3.2.1.1 Steps to set up new Ranger Policy

1. Enable the HDFS Plugin and other plugins required.
2. To ensure that the root user on Isilon is part of the Hadoop group, run the below add command. Alternately, add the root user to the all-path policy and provide RWX access described in step 3.

   ```
   isi auth groups modify hadoop --add-user=root
   ```

3. Update the root Ranger policy with ALL GROUPS or PUBLIC to have X access - /* recursive, as show in Figure 14.

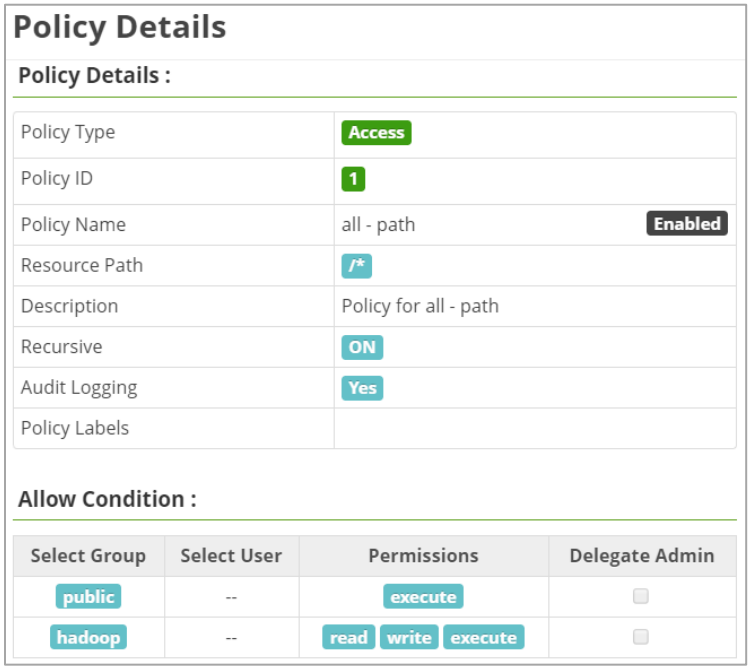



Figure 14 Root Ranger policy setup

4. Grant the local Hadoop super group to have RWX access - /* recursive, as shown in Figure 14.
5. Identify generic service folders (such as /tmp, /mr-history, or /app-logs) and create a new policy to these paths for PUBLIC to have RWX access. This makes service jobs run and everything defaults to file permissions as shown in the Figure 15.

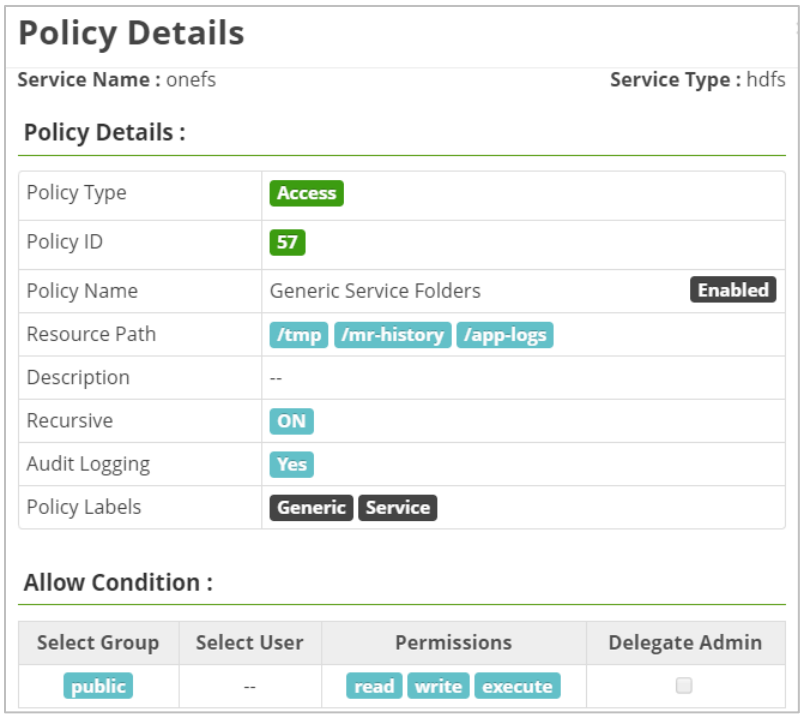
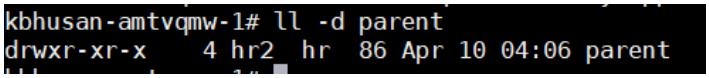



Figure 15 Ranger Policy set on generic service folders

6. Identify standalone folders and create new ranger policies equivalent to their POSIX permissions to mimic no ranger policies. Alternately, standalone folders are controlled by the underlying posix permissions and the user and group that own the folder. A blanket RWX for this folder should work if needed, based on the root policy defined (execute only for public).



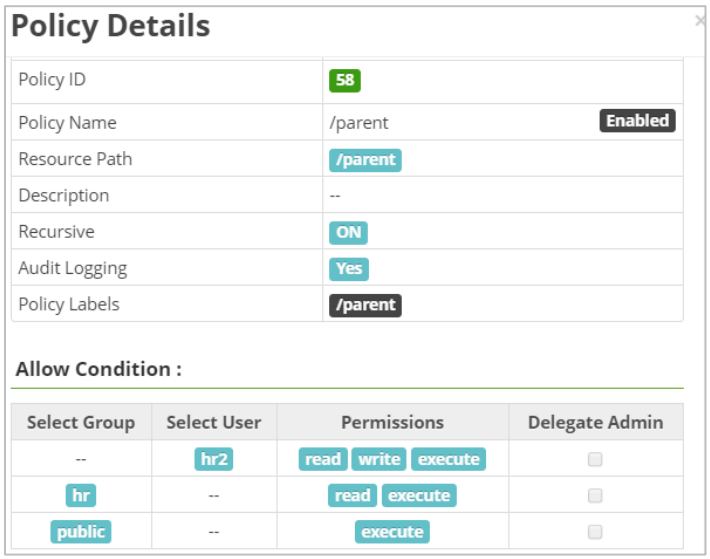




Figure 16 Policy setup for standalone folders

7. Identify the folders which need multi group or users and change POSIX permissions to 777.

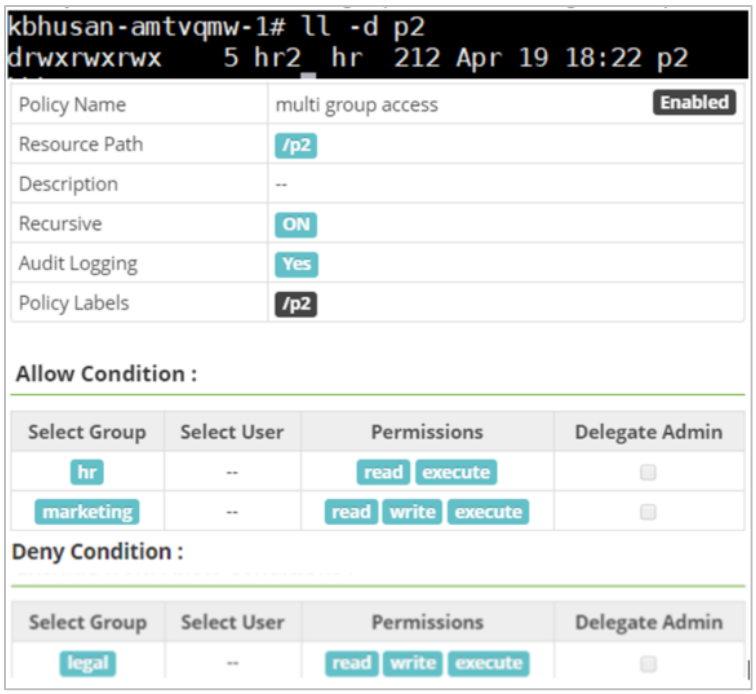



Figure 17 Policy setup for multiple groups or users

8. Update Ranger policies to include ALLOW from ALLOW conditions, which the Isilon plugin reads and applies.
9. Update Ranger policies to include DENY and DENY EXCLUDE from DENY conditions, which the Isilon plugin reads and applies.

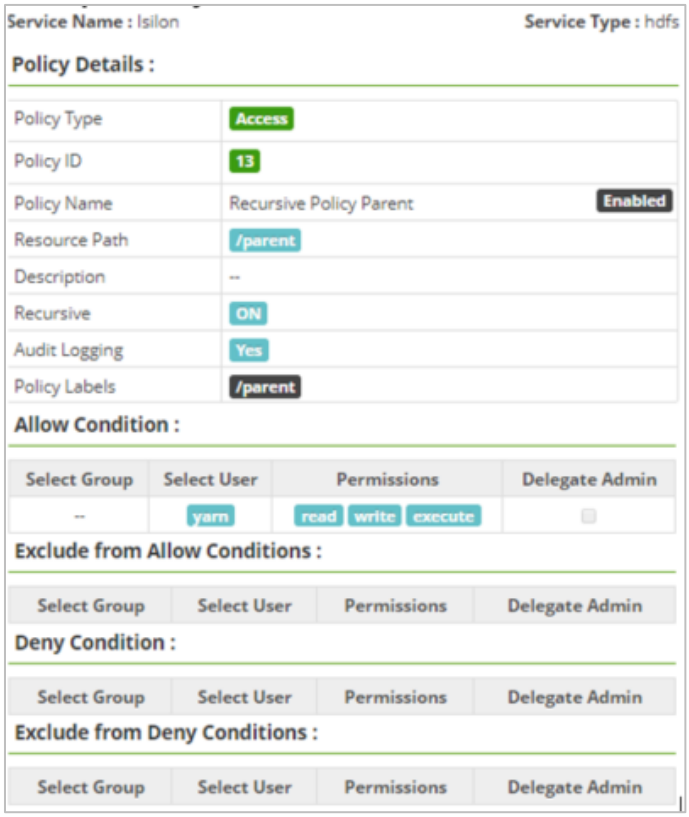
   - ALLOW Conditions: Should include only users or groups that need any of the RWX access.
   - DENY Conditions: Should include only users or groups to be denied access. This takes precedence over ALLOW conditions because it falls under DENY Ranger Access (check case).
   - EXCLUDE from DENY: This should include only users or groups to be excluded from the deny access. This takes precedence over the DENY condition. This would be the same since you would have assigned permissions in ALLOW conditions.

#### 3.2.1.2 Limitations

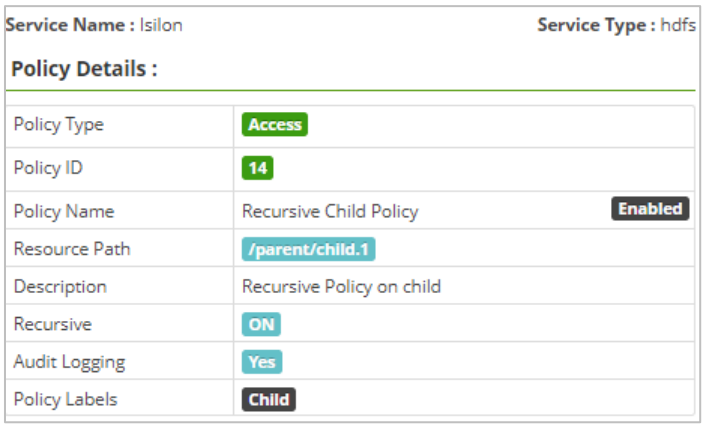
The following are limitations to this method:

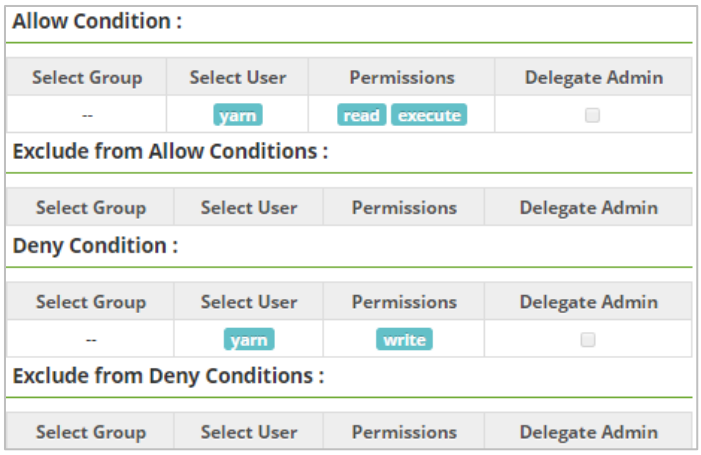
- There is POSIX 777 (everyone RWX) access for the custom folders. Multi-protocol access needs to be restricted.
- There is no Ranger failsafe like in Hadoop on DAS Implementation by setting 000 to permissions.
- Ranger policies need to be created for the complete file system
- Manual file administration is required on the file system to set POSIX permissions.
- Recursive policy implementation with parent and child folders including the group and users from parent policies.

The following is an example policy on the parent folder:




The following is an example policy on the child folder:






- Admin intervention is needed to resolve file-permission conflicts in case of recursive ranger policies.
- Folders with no policies cannot fall back to POSIX unless POSIX-equivalent Ranger policies are set on them. Alternately, annual POSIX permissions are handled and a blanket RWX permission is set on them.

## 3.2.2 Method 2: Public group RWX and DENY condition policies

In this approach, all polices are created and managed in Ranger with appropriate POSIX permissions to the folders. By default, Isilon storage starts with a deny policy meaning there is no access to anyone unless explicitly provided by POSIX with Ranger Policies in the case of Hadoop. To assign and manage privileges ,the Hadoop folders are given 777 permissions.

### 3.2.2.1 Set up a new Ranger Policy

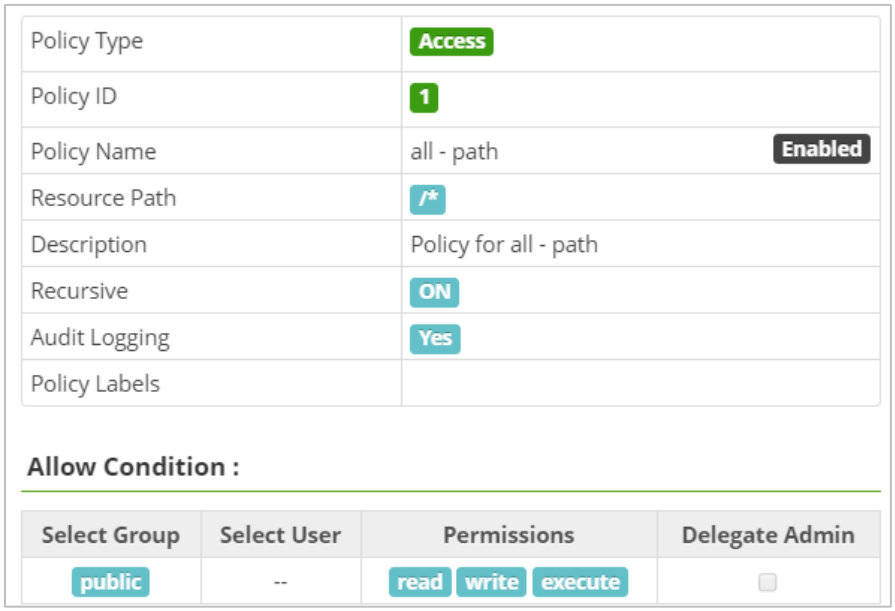
1. To set up umask to 000 in the cluster, add **fs.permissions.umask-mode=000** in the custom hdfs-site.xml config file.


2. Restart the Hadoop cluster.

**Note:** After above step, the new objects created will have 777 POSIX permissions. This is only required for directory resources which have multi-group accesses. The remaining directory resources can be changed back to 755 POSIX permissions.

3. Enable the HDFS plugin and other plugins required.

4. Update the root Ranger policy with ALL GROUPS or PUBLIC to have RWX access.



| | |
|---|---|
| Policy Type | Access |
| Policy ID | 1 |
| Policy Name | all - path Enabled |
| Resource Path | /* |
| Description | Policy for all - path |
| Recursive | ON |
| Audit Logging | Yes |
| Policy Labels | |

**Allow Condition :**

| Select Group | Select User | Permissions | Delegate Admin |
|---|---|---|---|
| public | -- | read write execute | |

5. Update the Ranger policies to include DENY and DENY Exclude from DENY conditions. The Isilon plugin only reads these in apply.

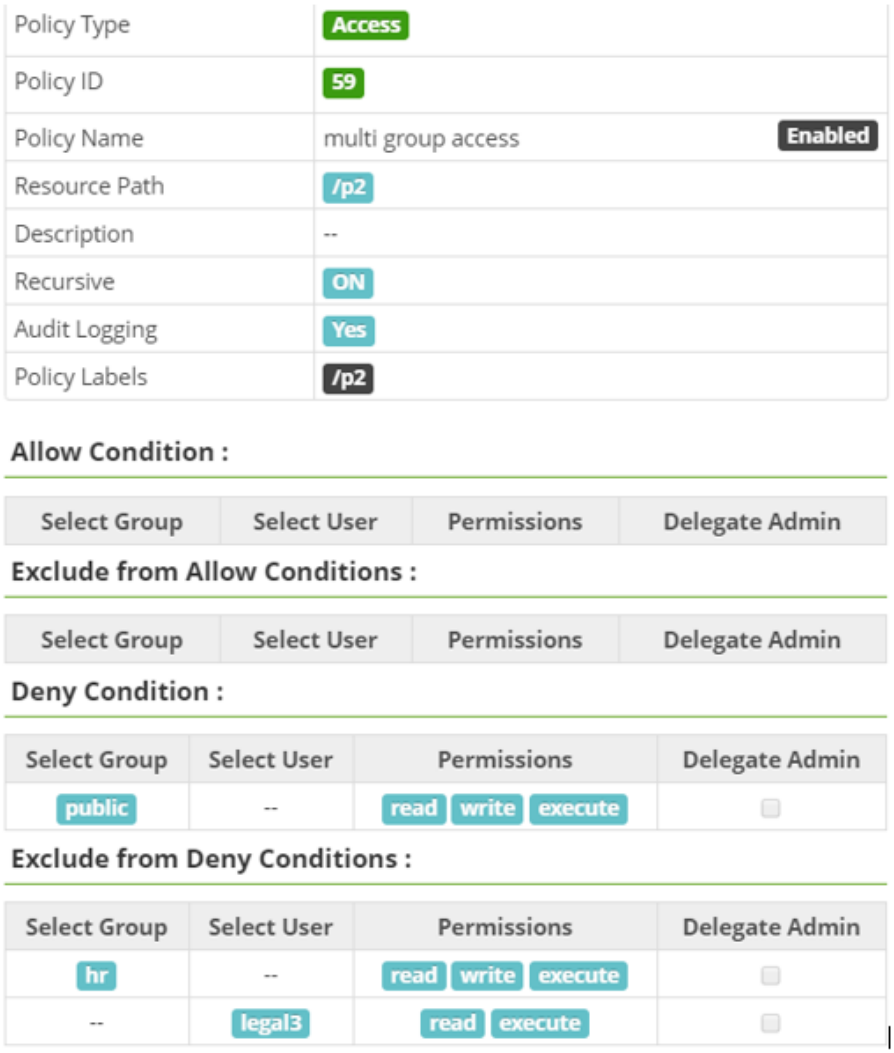



Figure 18 Ranger policy setup with public RWX and DENY conditions

- DENY Conditions: This should include all Groups Accessing Hadoop Cluster or Public group if require, as shown in Figure 18.
- EXCLUDE from DENY: This should include all the groups and Privileges as intended to be in DAS ALLOW conditions.
- EXCLUDE from DENY: This should also include Hadoop groups, parent policy groups that are set with recursive “ON”.

**Note:** Ranger Polices on a subfolder cannot override the restrictive policies set on the parent folder. This is because in Ranger, DENY conditions take precedence over ALLOW when Recursion is set ON.

- EXCLUDE from DENY: The Folder group owner and user should also be carried into the Ranger policy to simulate DAS behavior in case the folder POSIX is set other than **000.**

#### 3.2.2.2 Limitations

This method has the following limitations:

- This has POSIX 777 (everyone RWX) access for the custom folders. Multi-protocol access needs to be restricted.
- There is no Ranger failsafe like in the Hadoop-on-DAS implementation by setting 000 to permissions.
- DENY policies override ALLOW policies, so the Ranger Policies defined for a group restricting access to a subfolder and providing access to the same group in its child folder with Recursion ON cannot be simulated in Isilon ranger policy.
- When a new group requires access, ensure that the group has access from the root folder though the folder requiring access by adding the group in all parent folder policies.
- Umask is set to 000, and there is 777 access for all the custom folders.

## 3.3 Deployment summary and use cases

As discussed in the previous section, you can opt to deploy Ranger on Isilon storage using two methods. The choice of deployment method is focused on the two important aspects of the Ranger plugin behavior on Isilon storage:

- Isilon OneFS implements only a deny policy in the Ranger plugin.
- The Ranger policy is for the Public group on the root path (all-path).

Table 2 Deployment summary and use case

| Deploy method | Root path | POSIX file permission | Recommendation | Use cases |
|---|---|---|---|---|
| Public group X / R-X and ALLOW / DENY Condition | • Hadoop Group RXW<br>• Public Group X | • Root Path ➔ Default file permissions<br>• Folders with no policy ➔ Default file permissions<br>• Custom folders with Policies ➔ 777 file permissions | Global namespace update of ranger policies to make them compatible with Isilon | • Small cluster with few ranger policies planned<br>• Permission conflicts can be handled manually<br>• No Isilon ACL overhead |
| Public group RWX and DENY Condition | • Hadoop Group RXW<br>• Public Group RWX | • Root Path ➔ Default file permissions<br>• Folders with no policy ➔ Default file permissions<br>• Custom folders with Policies ➔ 777 file permissions<br>• New objects created with 777 file permissions. | Folders with Policies➔ Update of policy to sum the privileges for each policy from higher level in hierarchy and add folder level ownership and permissions to each policy<br><br>Folders with no policies ➔ Falls back to POSIX. | • Larger cluster<br>• Fewer folder with policies<br>• Subfolder not requiring less restrictive privileges then the parent<br>• Access groups not renamed or added frequently<br>• No Isilon ACL overhead |

# 4 Ranger policy migration from DAS to Isilon

There are two options to implement the Ranger Policies. Migrating from DAS to Hadoop on Isilon storage are detailed as follows, along with their limitations.

**Ranger Only:** All policies are created and managed in Ranger with appropriate POSIX permissions to the folders as illustrated above.

**Authoritative Isilon ACE (ACLs) or POSIX and Ranger Policies:** This approach leverages Isilon ACEs instead of HDFS Ranger Policies this would be covered with a blanket root-level policy while the privileges are entirely defined by the explicit OneFS ACEs or POSIX defined for each folder or file.

The following table illustrates the high-level steps and differences in creating and managing Ranger Policies with Hadoop on Isilon.

Table 3 Ranger policy migration

| Steps | Ranger on DAS | Ranger on Isilon |
|---|---|---|
| Permissions | The DAS servers start with open permissions, however once we implement the Ranger policies, we revoke all the permissions setting the folder level permission to 000- (no permissions for RWX for anyone). | Isilon by default starts with a deny policy, meaning there is no access to anyone unless explicitly provided by POSIX + ACLs with Ranger Policies in case of Hadoop. In order to assign and manage privileges, the Hadoop folders are given 777 permissions.<br><br>http://doc.isilon.com/onefs/hdfs/ifs_c_ranger.html |
| Policy | Ranger Policies are created for each folder, adding the permissions as needed for multiple groups or users internal or external. This is done in Ranger UI with Allow Conditions for the groups and users require. | Ranger Policies are created for each folder restricting the permissions as required for multiple groups or users internal or external. This is done in Ranger UI with DENY and Exclude from DENY conditions for the required groups and users (*ALLOW conditions cannot be used to restrict permissions as Isilon Ranger plugin reads on DENY Conditions). |
| **Migration** | | |
| Step 1<br><br>Option 1 (Ranger only) | **Allow conditions** apply the strict permissions in case of DAS and DENY and Exclude from DENY Conditions apply strict permissions in case of Isilon on Hadoop<br><br>Migrating existing DAS policies involves the following steps and updates to DAS Policies.<br><br>1. Enable HDFS Plugin and other plugins as required.<br>2. Update root Ranger policy with ALL GROUPS or PUBLIC to have RWX access along with Hadoop group.<br>3. Update Ranger policies to include DENY and Exclude from DENY Conditions. This is because Isilon plugin only reads these in apply.<br><br>• DENY Conditions: This should include all Groups Accessing Hadoop Cluster or Public group if require.<br>• Exclude from DENY: This should include all the groups and privileges as assigned in DAS ALLOW conditions<br>• Exclude from DENY: Should also include Hadoop group, parent policy groups that are set with recursive “ON” (This may be a bug in ranger plugin as recursive should implicitly allow the parent groups access) | |

| | |
|---|---|
| | • EXCLUDE from DENY: The Folder group owner and user should also be carried into the Ranger policy to simulate the DAS behavior in case the folder POSIX is set other than '000'.<br><br>**Limitations:**<br><br>• Ranger Policies defined in DAS for a group restricting access to a subfolder and providing access to the same group in its child folder with Recursive ON cannot be simulated in the Isilon Ranger policy. (It may be argued here that the DAS Ranger is contradicting and relaxing its rules by letting the user override the previous recursive option).<br>• A new group can be introduced anywhere in the tree structure in DAS; the same is not possible with the Isilon Ranger plugin enabled. This is due to the fact that the group would need to have read access to parent folders to enable this have to update the parent policies to let the groups have access to execute or other permissions.<br>• Umask is set to 000 and 777 access is set for all custom folders<br>• No Ranger failsafe. |
| Bulk update of Ranger policy | For the above setup to be in place, export existing DAS Ranger policies in the Ranger UI in JSON format and update the file, moving the Allow Conditions to **Exclude from DENY conditions** and import this (apply the requirement mentioned in option-1 above, step 3). |
| Step 2<br><br>Migration<br><br>Option 2<br>Isilon ACE (ACLs) or POSIX + Ranger policies | **Allow Conditions** apply the strict permissions in case of DAS and **DENY** and **Exclude from DENY Conditions** apply strict permissions in case of Isilon on Hadoop.<br><br>Migrating existing DAS policies involves following steps and updates to DAS policies.<br><br>1. Enable the HDFS Plugin and other plugins as required.<br>2. Update the root Ranger policy for HDFS with ALL GROUPS or PUBLIC to have RWX access along with Hadoop group. This will be the only policy required for HDFS.<br>3. Convert all DAS Ranger Policies to Isilon ACE (ACLs) mimicking the recursive and non-recursive policies with corresponding container inheritance/object inheritance accordingly.<br>4. Hive/HBase policies should work as in DAS, while the HDFS privileges fall back to POSIX + Isilon ACE permissions.<br>5. This option overcomes the umask setting and 777 requirements for all of Hadoop namespace (security concerns are less).<br><br>**Limitations:**<br><br>• Isilon ACE + POSIX mode summation is not guaranteed with consistent results if an end user updates the POSIX mode bits.<br>• Reporting on the folder- and file-level privileges is not possible as in DAS.<br>• There is a learning curve involved in creation and maintenance of Isilon ACEs.<br>• HBase policies have issues accessing the data. |

# A Detailed policy behaviors on DAS and Isilon

## A.1 Ranger on DAS

If there are no policies for a path or user, Apache falls back to using the filesystem permissions.

### A.1.1 Test case 1: Zero Ranger polices

In this case, no Ranger policies are set up.

```
* uid=7020(mktg1) gid=7020(marketing) groups=7020(marketing)
```

Start by listing the root directory. You can see that mktg1 user falls into the **other** group and is allowed to list.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls -d /
drwxr-xr-x   - hdfs hdfs          0 2019-04-11 16:15 /
```

Next, look at the permission on the **user** directory. This does not allow the group or other to do anything.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /
drwx------   - hdfs   hdfs            0 2019-04-19 17:25 /user
```

So, listing the user directory fails with READ_EXECUTE when listing as user **mktg1**.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /user
ls: Permission denied: user=mktg1, access=READ_EXECUTE,
inode="/user":hdfs:hdfs:drwx------
```

Finally, try to list a directory inside the user directory. This directory does exist, and the permission is denied again but with EXECUTE only.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /user/mktg1
ls: Permission denied: user=mktg1, access=EXECUTE,
inode="/user/mktg1":hdfs:hdfs:drwx------
```

### A.1.2 Test case 2: Recursive policy public DENY, and group DENY exceptions

This case adds a **recursive** policy that denies RWX group:public access to /user, but puts group:marketing RX in the deny exceptions list.

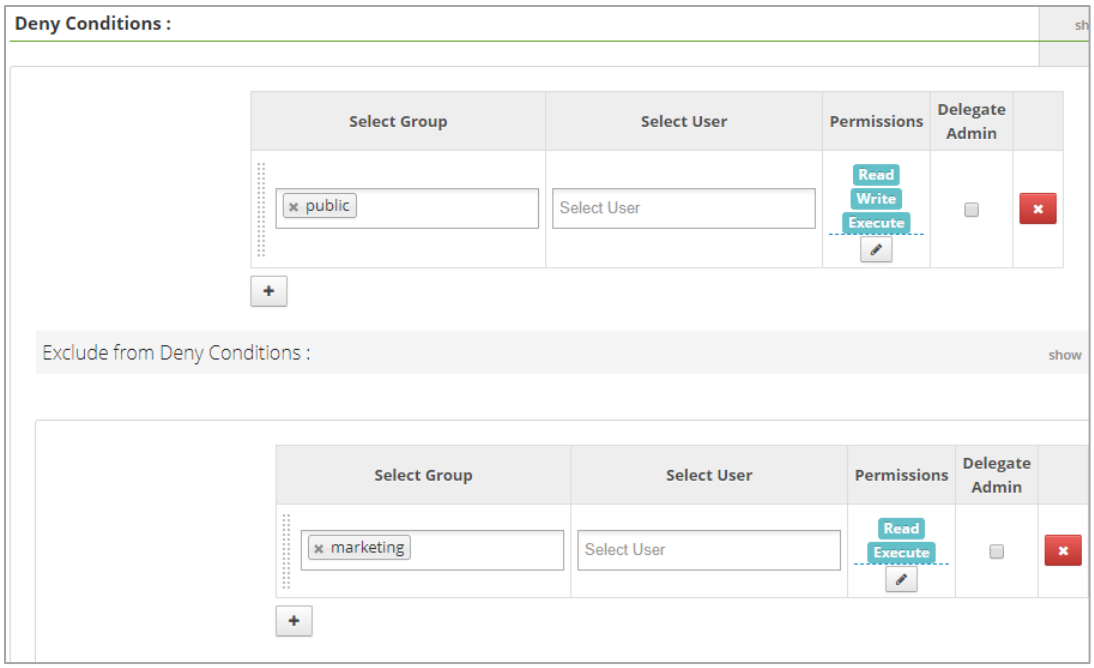



Figure 19 Ranger DAS recursive deny condition

Creating the deny exception for a group does not actually cause any permission behavior to change. In the following, you can see we get the same error as we did when we tried this without any policies. Deny exceptions do not grant any permissions, they just exclude from the deny list.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /user
ls: Permission denied: user=mktg1, access=READ_EXECUTE,
inode="/user":hdfs:hdfs:drwx------
mktg1@kb-hdp1:~ $ hadoop fs -ls /user/mktg1
ls: Permission denied: user=mktg1, access=EXECUTE,
inode="/user/mktg1":hdfs:hdfs:drwx------
```

### A.1.3 Test case 3: Policies with wildcards

This case creates a policy with /us* as the path name and grants RWX to the group:marketing which user:mktg1 is a member of. You can list the content of /us and /user directories.

Before the policy was created:

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /use
ls: Permission denied due to undetermined access: user=mktg1 ,
access=READ_EXECUTE, path=/use
```

After the policy was created, you can list anything that starts with /us*:

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /use
Found 1 items
-rw-r--r--   3 root hadoop          0 2019-04-22 20:48 /use/file1
mktg1@kb-hdp1:~ $ hadoop fs -ls /user
Found 2 items
drwxr-xr-x   - hdfs hdfs                0 2019-04-04 21:59 /user/hdfs
```

```
drwx------   - mktg1 marketing          0 2019-04-19 16:31 /user/mktg1
```

### A.1.4 Test case 4: Trailing forward slash '/'

With DAS, if you include a trailing slash, this causes the policy to only be applied to subdirectories of this path and not include the actual parent directory. The matching system does not account for the fact that /user/ and /user are the same directory. The reason it works on sub-directories is because the policy was recursive. If you remove the recursive setting, all child directories become unusable.

Create a recursive policy for /user/ and grant user:mktg1 RWX access to this path. Here is the result of that policy.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls  /user
ls: Permission denied: user=mktg1, access=READ_EXECUTE,
inode="/user":hdfs:hdfs:drwx------
mktg1@kb-hdp1:~ $ hadoop fs -ls  /user/mktg1
Found 1 items
drwxr-xr-x   - mktg1 hdfs          0 2019-04-19 20:39 /user/mktg1/dir1
```

Now, switch the policy to be /user path and exclude the trailing slash. You can now list the contents of /user because the policy applies to that path and not just sub directories.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls  /user
Found 5 items
drwxrwx---   - ambari-qa hdfs          0 2019-04-10 15:31 /user/ambari-qa
drwxr-xr-x   - hcat      hdfs          0 2019-04-10 15:31 /user/hcat
drwxr-xr-x   - hive      hdfs          0 2019-04-10 15:31 /user/hive
drwxr-xr-x   - mktg1     marketing          0 2019-04-19 20:39 /user/mktg1
drwx------   - yarn      hdfs          0 2019-04-11 16:07 /user/yarn
mktg1@kb-hdp1:~ $ hadoop fs -ls  /user/mktg1
Found 1 items
drwxr-xr-x   - mktg1 hdfs          0 2019-04-19 20:39 /user/mktg1/dir1
```

### A.1.5 Test case 5: Remove execute bit from parent directory

This case resembles the way Isilon works and shows how Apache only works for some cases. If you look at the Isilon solution, you will notice that there needs to be a root policy that allows RX on the entire tree otherwise a user cannot do anything. This is not the case for DAS if you do a default setup. DAS has a normal 755-type directory permission that allows it to fall back to filesystem permissions to grant execute and read. And the best practice states, in DAS set umask 077 which is RW only to the owner.

Here is a listing when a user does not have filesystem permission on / for execute.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls  /user
ls: Permission denied: user=mktg1, access=EXECUTE,
inode="/user":hdfs:hdfs:drwxr-x---
```

Take note that the permission listed above in the stderr are the same as the permissions on /.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls  -d /
drwxr-x---   - hdfs hdfs          0 2019-04-22 22:17 /
```

### A.1.6 Test case 6: Public ALLOW and recursive DENY conditions

This test case discusses some of the key elements. If you deny a particular permission on a parent directory with the recursive flag of true, and then deny a lesser set of permissions down the tree, this shows if the user granted is access or not. Another important policy in this scenario is that there is a root policy that grants everyone ('public' group) full access to everything. This layout has the goal of granting certain users access to a certain directory in a tree but nothing above that directory. In the end, what you see is that policies basically stack on top of each other and if take away something higher in the tree recursively then you do not get it back lower in the tree.

Policy setup:
---

```
user:mktg1     group:marketing,legal,hr
```

The public group means **all groups** and is a special case.

```
path: /
group: public ALLOW rwx
Recursive: True

path: /data1
group: public DENY rwx
group: marketing DENY EXCEPTION RWX
group: legal,hr DENY EXCEPTION RX
Recursive: False

path: /data1/data1.2
group: public DENY RWX
group: marketing,legal,hr DENY EXCEPTION X or RX
Recursive: True

path: /data1/data1.2/marketing-dir
group: public DENY RWX
group: marketing DENY EXCEPTION RWX
Recursive: True
```

DAS example:
---

```
root@kb-hdp1:~ # id mktg1
uid=7020(mktg1) gid=7020(marketing) groups=7020(marketing)
root@kb-hdp1:~ # su - mktg1
mktg1@kb-hdp1:~ $ hadoop fs -ls /data1
Found 1 items
drwxr-xr-x   - hdfs hdfs          0 2019-04-25 17:23 /data1/data1.2
mktg1@kb-hdp1:~ $ hadoop fs -ls /data1/data1.2
Found 1 items
drwxr-xr-x   - hdfs hdfs          0 2019-04-25 17:23 /data1/data1.2/marketing-
dir
```

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /data1/data1.2/marketing-dir
mktg1@kb-hdp1:~ $ hadoop fs -touchz /data1/data1.2/marketing-dir/mktg1-only
touchz: Permission denied: user=mktg1, access=WRITE,
inode="/data1/data1.2/marketing-dir"
```

## A.2 Ranger on Isilon cluster

These test cases below should match up to the same test case as the Apache DAS above. Behavior is different though. Specifically, when Isilon does not have a policy that matches the path it results in undetermined access which results in a 'deny'.

### A.2.1 Test case 1: Zero Ranger Polices

This case has zero policies.

- Isilon ID: `uid=7020(mktg1) gid=7020(marketing) groups=7020(marketing)`
- Hadoop Client ID: `uid=7020(mktg1) gid=7020(marketing) groups=7020(marketing)`

In these examples, you can see Isilon returns **undetermined access**. These exceptions are all **RangerAccessControlException**. They come from the Ranger checks in the Isilon HDFS service, not the filesystem.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls -d /
ls: Permission denied due to undetermined access: user=mktg1 , access=EXECUTE,
path=/
mktg1@kb-hdp1:~ $ hadoop fs -ls /
ls: Permission denied due to undetermined access: user=mktg1 , access=EXECUTE,
path=/
mktg1@kb-hdp1:~ $ hadoop fs -ls /user
ls: Permission denied due to undetermined access: user=mktg1 , access=EXECUTE,
path=/
mktg1@kb-hdp1:~ $ hadoop fs -ls /user/mktg1
ls: Permission denied due to undetermined access: user=mktg1 , access=EXECUTE,
path=/user
```

### A.2.2 Test case 2: Recursive policy public DENY, and group DENY exceptions

This case adds a **recursive** policy that **denies RWX group:public access to /user**, but puts **group:marketing RX in the deny exceptions** list. These are failing because there is no policy for /. So even though a policy was created for /user, that does not matter if the user cannot traverse (execute) to the path that was used for the policy.

This gets exactly the same results as without any policies. This is because setting someone as a deny exception does not grant them any permissions.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls -d /
ls: Permission denied due to undetermined access: user=mktg1 , access=EXECUTE,
path=/
```

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /user
ls: Permission denied due to undetermined access: user=mktg1 , access=EXECUTE,
path=/
mktg1@kb-hdp1:~ $ hadoop fs -ls /user/mktg1
ls: Permission denied due to undetermined access: user=mktg1 , access=EXECUTE,
path=/user
```

### A.2.3 Test case 3: Recursive policy group DENY exception

This case adds an execute recursive policy for group:marketing which user:mktg1 is a member of.

Isilon always needs a root-level policy that grants all users and groups an execute permission, so they can walk the tree and access anything. Without this, they will always get execute exception on /.

### A.2.4 Test case 4: Policies with wildcards

This case has a similar setup as above, but if you create a policy with /us* as the path name and grant RWX to the group:marketing which user:mktg1 is a member of, you can list the contents of /us and /user directories.

Before the policy was created:

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /use
ls: Permission denied due to undetermined access: user=mktg1 ,
access=READ_EXECUTE, path=/use
```

After the policy was created, you can list anything that starts with /us*:

```
mktg1@kb-hdp1:~ $ hadoop fs -ls /use
Found 1 items
-rw-r--r--   3 root hadoop          0 2019-04-22 20:48 /use/file1
mktg1@kb-hdp1:~ $ hadoop fs -ls /user
Found 2 items
drwxr-xr-x   - hdfs hdfs                  0 2019-04-04 21:59 /user/hdfs
drwx------   - mktg1 marketing          0 2019-04-19 16:31 /user/mktg1
```

### A.2.5 Test case 5: Trailing forward slash ‘/’

This case behaves differently to the Apache DAS. With Isilon, if you include a trailing slash, this causes the policy to be applied to both parent directory and subdirectories.

Create a recursive policy for /user/ and grant user:mktg1 RWX access to this path(DENY Exception). Here is the result of that policy:

```
mktg1@kb-hdp1:~ $ hadoop fs -ls  /user
Found 5 items
drwxrwx---   - ambari-qa hdfs          0 2019-04-10 15:31 /user/ambari-qa
drwxr-xr-x   - hcat      hdfs          0 2019-04-10 15:31 /user/hcat
drwxr-xr-x   - hive      hdfs          0 2019-04-10 15:31 /user/hive
drwxr-xr-x   - mktg1      hdfs          0 2019-04-19 20:39 /user/mktg1
drwx------   - yarn      hdfs          0 2019-04-11 16:07 /user/yarn
mktg1@kb-hdp1:~ $ hadoop fs -ls  /user/mktg1
```

```
Found 1 items
drwxr-xr-x   - mktg1 hdfs          0 2019-04-19 20:39 /user/mktg1/dir1
```

Now, switch the policy to be /user path and exclude the trailing slash.

```
mktg1@kb-hdp1:~ $ hadoop fs -ls  /user
Found 5 items
drwxrwx---   - ambari-qa hdfs          0 2019-04-10 15:31 /user/ambari-qa
drwxr-xr-x   - hcat      hdfs          0 2019-04-10 15:31 /user/hcat
drwxr-xr-x   - hive      hdfs          0 2019-04-10 15:31 /user/hive
drwxr-xr-x   - mktg1      hdfs          0 2019-04-19 20:39 /user/mktg1
drwx------   - yarn      hdfs          0 2019-04-11 16:07 /user/yarn
mktg1@kb-hdp1:~ $ hadoop fs -ls  /user/mktg1
Found 1 items
drwxr-xr-x   - mktg1 hdfs          0 2019-04-19 20:39 /user/mktg1/dir1
```

## A.2.6 Test case 6: Public ALLOW and recursive DENY conditions

This test case discusses some of the key elements. If you deny a particular permission on a parent directory with a recursive flag of true, and then deny a lesser set of permissions, this looks down the tree to see if the user is granted access or not. Another important policy in this scenario is that there is a root policy that grants everyone ('public' group) full access to everything. This layout has the goal of granting certain users access to a certain directory in a tree but nothing above that directory. In the end, what you see is that policies basically stack on top of each other and if you take away something higher in the tree recursively, you do not get it back lower in the tree.

Policy setup:
---

```
user:mktg1, group:marketing,legal,hr
```

The Public group means **all groups** and is a special case.

```
path: /
group: public ALLOW rwx
Recursive: True

path: /data1
group: public DENY rwx
group: marketing DENY EXCEPTION RWX
group: legal,hr DENY EXCEPTION RX
Recursive: False

path: /data1/data1.2
group: public DENY RWX
group: marketing,legal,hr DENY EXCEPTION X or RX
Recursive: True

path: /data1/data1.2/marketing-dir
group: public DENY RWX
group: marketing DENY EXCEPTION RWX
Recursive: True
```

Isilon example:

---

```
mktg1@kb-hdp1:~ $ kinit
Password for mktg1@WEST.ISILON.COM:
mktg1@kb-hdp1:~ $ hadoop fs -ls /data1
Found 1 items
drwxr-xr-x   - hdfs hdfs          0 2019-06-10 09:03 /data1/data1.2
mktg1@kb-hdp1:~ $ hadoop fs -ls -R /data1
drwxr-xr-x   - hdfs hdfs          0 2019-06-10 09:03 /data1/data1.2
drwxr-xr-x   - hdfs hdfs          0 2019-06-10 09:03 /data1/data1.2/marketing-
dir
mktg1@kb-hdp1:~ $ hadoop fs -touchz /data1/data1.2/marketing-dir/mktg1-only
touchz: Permission denied: user=mktg1@WEST.ISILON.COM, access=WRITE,
path="/data1/data1.2/marketing-dir/mktg1-only"
mktg1@kb-hdp1:~ $
```

Isilon HDFS log:

---

```
hdfs[2495]: [hdfs] RPC V9 create user: mktg1@WEST.ISILON.COM exception:
org.apache.ranger.authorization.hadoop.exceptions.RangerAccessControlException
cause: Permission denied: user=mktg1@WEST.ISILON.COM, access=WRITE,
path="/data1/data1.2/marketing-dir/mktg1-only"
```

The behavior is exactly same as the Apache Ranger DAS implementation.

### A.2.7 Test case 7: Multiple groups/users access control policy

On the DAS implementation of Ranger, the filesystem permission check is omitted when the Ranger policy is set. However, on Isilon you need to handle the filesystem permission check either by setting RWX to everyone or Isilon ACE (ACLs) for the multiple groups/users on the directories which will have Ranger policy set.

**Ranger only:** All policies are created and managed in Ranger with appropriate POSIX permissions to the folders.

**Ranger Only method:**

1. Create new directory posix_777 and set permissions to 777.
2. Create a recursive policy for /posix_777/ and grant group:marketing and legal RWX access to this path(DENY Exception) and deny group:public.

Policy setup:

---

```
user:mktg1,legal1,hr2, group:marketing,legal,hr
```

The Public group means **all groups** and is a special case.

```
path: /
group: public ALLOW rwx
Recursive: True

path: /posix_777
group: public DENY rwx
group: marketing, legal DENY EXCEPTION RWX
Recursive: True
```

Isilon example:

---

```
mktg1@kb-hdp1:~ $ hadoop fs -ls -d /posix_777
drwxrwxrwx   - hdfs hadoop          0 2019-06-11 07:22 /posix_777
mktg1@kb-hdp1:~ $ whoami
mktg1
mktg1@kb-hdp1:~ $ hadoop fs -touchz /posix_777/mktg1-only
mktg1@kb-hdp1:~ $ hadoop fs -ls -R /posix_777
-rw-r--r--   3 mktg1 hadoop          0 2019-06-11 07:34 /posix_777/mktg1-only
mktg1@kb-hdp1:~ $ exit
exit
root@kb-hdp1:~ # su legal1
legal1@kb-hdp1:~ $ whoami
legal1
legal1@kb-hdp1:~ $ hadoop fs -touchz /posix_777/legal1-only
legal1@kb-hdp1:~ $ hadoop fs -ls -R /posix_777
-rw-r--r--   3 legal1 hadoop          0 2019-06-11 07:35 /posix_777/legal1-only
-rw-r--r--   3 mktg1  hadoop          0 2019-06-11 07:34 /posix_777/mktg1-only
legal1@kb-hdp1:~ $ exit
exit
root@kb-hdp1:~ # su hr2
hr2@kb-hdp1:~ $ hadoop fs -touchz /posix_777/hr2-only
touchz: Permission denied: user=hr2@WEST.ISILON.COM, access=EXECUTE,
path="/posix_777"
hr2@kb-hdp1:~ $ hadoop fs -ls -R /posix_777
ls: Permission denied: user=hr2@WEST.ISILON.COM, access=READ_EXECUTE,
path="/posix_777"
hr2@kb-hdp1:~ $
```

Isilon HDFS log:

---

```
2019-06-11T07:35:49Z <30.6> kbhusan-amtvqmw-1 hdfs[2495]: [hdfs] RPC V9
getFileInfo user: hr2@WEST.ISILON.COM exception:
org.apache.ranger.authorization.hadoop.exceptions.RangerAccessControlException
```

```
cause: Permission denied: user=hr2@WEST.ISILON.COM, access=EXECUTE,
path="/posix_777"
2019-06-11T07:36:02Z <30.6> kbhusan-amtvqmw-1 hdfs[2495]: [hdfs] RPC V9
getListing user: hr2@WEST.ISILON.COM exception:
org.apache.ranger.authorization.hadoop.exceptions.RangerAccessControlException
cause: Permission denied: user=hr2@WEST.ISILON.COM, access=READ_EXECUTE,
path="/posix_777"
```

## A.3 Hive Service Manager

### A.3.1 Hive Managed Tables

Hive policies set on managed tables for the Hadoop-deployed-with-Isilon cluster work in a similar way to the Hadoop-on-DAS implementation.

### A.3.2 Hive External Tables

Hive policies set on external tables for the Hadoop-deployed-with-Isilon cluster work in a similar way to the Hadoop-on-DAS implementation. However, the external table data location path needs to be handled for multiple groups/ users access either by setting file permissions to RWX for everyone or using the Isilon ACE (ACLs) method described in section A.2.7.

# 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.

## B.1 Related resources

- OneFS HDFS Admin Guide
- Apache Ranger DENY Policies with OneFS 8.0.1.0
- Providing Authorization with Apache Ranger
- HDFS Authorization with Apache Ranger Best Practice
- Access Control Lists on Dell EMC Isilon OneFS
- OneFS CLI Administration Guide
- Hortonworks HDP3.0.1 Install on Isilon OneFS8.1.2