Dell PowerScale OneFS Permission Repair Job

December 2025

H18442.10

White Paper

Abstract

This white paper examines the OneFS Permission Repair Job feature available on Dell PowerScale clusters.

Contents

Executive summary

Overview

This white paper focuses on the OneFS Job Engine Permission Repair job on a Dell PowerScale cluster, offering access control configuration recommendations for a range of data access scenarios. This paper does not provide a comprehensive background of the OneFS architecture.

See the Dell PowerScale OneFS: Technical Overview white paper for further details about the OneFS architecture.

Audience

The target audience for this white paper is anyone designing and deploying a PowerScale clustered storage environment. It is assumed that the reader has an understanding and working knowledge of the OneFS components, architecture, commands, and features.

More information about OneFS commands and feature configuration is available in the OneFS Administration Guide.

Revisions

Date	Part number/ revision	Description
June 2020		Updated for OneFS 9.0
September 2020		Updated for OneFS 9.1
April 2021		Updated for OneFS 9.2
September 2021		Updated for OneFS 9.3
April 2022		Updated for OneFS 9.4
January 2023		Updated for OneFS 9.5
January 2024		Updated for OneFS 9.7
April 2024	H18442.5	Updated for OneFS 9.8
August 2024	H18442.6	Updated for OneFS 9.9
December 2024	H18422.7	Updated for OneFS 9.10
April 2025	H18422.8	Updated for OneFS 9.11
August 2025	H18422.9	Updated for OneFS 9.12
December 2025	H18422.10	Updated for OneFS 9.13

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on this document. Contact the Dell Technologies team by email.

Author: Nick Trimbee

Note: For links to other documentation for this topic, see the PowerScale Info Hub.

Architecture

The OneFS PermissionRepair job provides an automated way to fix access controls across a dataset. This remediation option can be invaluable for several scenarios including access controls breaking due to inheritable permissions being set incorrectly, user changes, identity management changes, and so on.

The job contains three execution options, or modes, depending on the resolution required:

Mode	Usage
Clone	Used when a directory tree with a large file count requires a new set of permissions, such as switching from POSIX mode bits to Windows access control lists (ACLs)
Convert	Used to modify the on-disk identity type and permission settings, such as converting a directory path to a UNIX identity type
Inherit	Typically used whenever an inherited access control entry (ACE) is added to an existing directory tree

The PermissionRepair job is run against a target that can be a file or directory under `/ifs`. Depending on the job mode, Permission Repair enables:

- Permissions to be copied from a template to the target
- Acquisition of inheritable permissions from a template
- Changes to the on-disk identity type stored in files and directories

In contrast to the UNIX `chmod` command, Permission Repair is considerably more efficient. It performs its operations in parallel across all nodes in the cluster (although using the same system calls that the `chmod` command uses). As such, Permission Repair is a faster way to effect large-scale permissions changes across a sizable directory tree.

It changes permissions by using the OneFS Job Engine as its execution framework. The Job Engine runs across all nodes of the cluster and is responsible for dividing and conquering large storage management and protection tasks. To achieve this goal, it reduces a task into smaller work items and then allocates, or maps, these portions of the overall job to multiple worker threads on each node. Progress is tracked and reported throughout job execution, and a detailed report and status is presented upon completion or termination.

Job Engine

The Job Engine includes a comprehensive check-pointing system that allows jobs to be paused and resumed, in addition to stopped and started. The Job Engine framework also includes an adaptive impact management system, drive-sensitive impact control, and the ability to run multiple jobs at once.

The PermissionRepair job consists of a single phase, which is composed of several work chunks, or tasks. The tasks, which consist of multiple individual work items, are divided and load balanced across the nodes in the cluster. Successful execution of a work item produces an item result that might contain a count of the number of retries required to repair a file, plus any errors that occur during processing.

PermissionRepair performs a directory tree walk. It works similarly to common UNIX utilities, such as the find utility –in a far more efficient, distributed way. For parallel execution, the various job tasks are each assigned a separate subdirectory tree. Unlike LIN scans, tree walks might prove to be heavily unbalanced, due to varying subdirectory depths and file counts.

The Job Engine is based on a delegation hierarchy consisting of coordinator, director, manager, and worker processes.

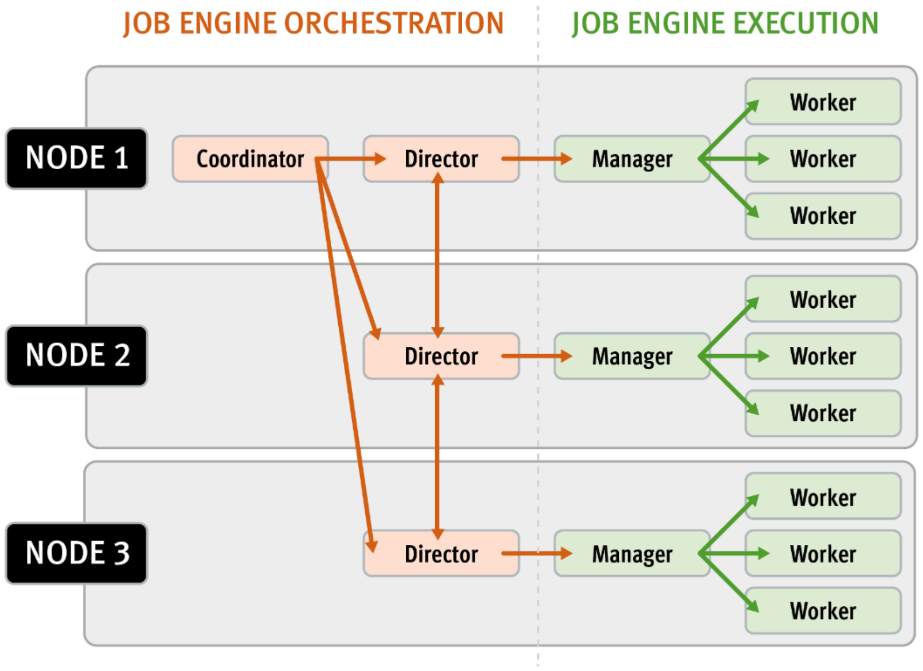


Figure 1. OneFS Job Engine distributed work allocation model

Once the work is initially allocated, the Job Engine uses a shared work distribution model to perform the work. The coordinator process, which runs on one of the nodes in a cluster, handles the Job Engine’s orchestration.

While the individual nodes manage the work item allocation, the coordinator node takes control, divides the job, and evenly distributes the resulting tasks across the nodes in the cluster. The coordinator is also responsible for starting and stopping jobs, and for processing work results as they are returned during the execution of a job.

Each node in the cluster has a Job Engine director process, which runs continuously and independently in the background. The director process is responsible for monitoring, governing, and overseeing all Job Engine activity on a particular node, constantly waiting for instruction from the coordinator to start a new job. The director process serves as a central point of contact for all the manager processes running on a node, and as a liaison with the coordinator process across nodes.

The manager process is responsible for arranging the flow of tasks and task results throughout the duration of a job. The manager processes both request and exchange work, and supervises the worker threads assigned to them. At any point in time, each node in a cluster can have up to three manager processes, one for each job currently running. These managers are responsible for overseeing the flow of tasks and task results.

Each manager controls and assigns work items to multiple worker threads working on items for the designated job. Every worker thread is given a task, if available, which it processes item by item until the task is complete or the manager unassigns the task. The `isi job statistics view` CLI command can provide the status of the nodes' workers. In addition to the number of current worker threads per node, a sleep to work (STW) ratio average is also provided, giving an indication of the worker thread activity level on the node.

The Job Engine allocates a specific number of threads to each node by default, controlling the impact of a workload on the cluster. If little client activity occurs, more worker threads are spun up to allow more work, up to a predefined worker limit. For example, the worker limit for a:

- Low-impact job might allow one or two threads per node to be allocated.
- Medium-impact job might allow from four to six threads to be allocated.
- High-impact job might allow a dozen or more threads to be allocated.

When this worker limit is reached (or before, if client load triggers impact management thresholds first), worker threads are throttled back or terminated.

As jobs are processed, the coordinator consolidates the task status from the constituent nodes, and periodically writes the results to checkpoint files. These checkpoint files allow jobs to be paused and resumed, either proactively, or if there is a cluster outage. For example, if the node on which the Job Engine coordinator was running goes offline for any reason, a new coordinator is automatically started on another node. This new coordinator reads the last consistency checkpoint file. Job control and task processing resume across the cluster from where it left off, and no work is lost.

The Job Engine resource monitoring and execution framework allows jobs to be throttled based on both CPU and disk I/O metrics. The granularity of the resource utilization monitoring data provides the coordinator process with visibility into exactly what is generating IOPS on any drive across the cluster. This level of insight allows the coordinator to make precise determinations about exactly where and how impact control is best applied. The coordinator itself does not communicate directly with the worker threads, but rather with the director process. The director process then instructs a node's manager process for a specific job to cut back threads.

Certain jobs, if left unchecked, can consume vast quantities of a cluster's resources, contending with and impacting client I/O. To counteract this impact, the Job Engine employs a comprehensive work throttling mechanism that can limit the rate at which individual jobs can run. Throttling is employed at a per-manager process level, so job impact can be managed both granularly and gracefully.

Further information is available in the Dell PowerScale OneFS Job Engine White Paper.

Multiprotocol permissions overview

For OneFS to support concurrent, multiprotocol data access natively, it maps the POSIX mode bits from NFS to the access control model of the Windows SMB protocol, and conversely. To achieve this step, OneFS provides:

- Transparent mapping of an on-disk identity to the correct requesting protocol
- Equivalency method to transform authoritative permissions into a form that the requesting access protocol can comprehend
- Permissions representation of where one permission style (that is, ACL) is authoritative and the other an approximation (that is, mode bits)
- The ability to configure how OneFS manages permissions for different environments
- A default access control policy that optimally merges new permissions with existing ones
- A 'Synthetic ACL' that approximates the mode bits of a UNIX file for an SMB client

In OneFS, each ACE in a security descriptor is displayed as a single line prefaced by an index number and containing the following properties:

ACE property	Description
Identity	The identity to which the ACE applies
Allow or Deny	Whether the ACE allows or denies the permissions listed as part of the ACE
Permissions	A list of one or more permissions that the ACE allows or denies
Permissions words	Indication of flags that affect inheritance behavior, if present in the ACE

The identity can be one of these types:

- User
- Group
- Everyone

For example:

```
-rw-r--r--    1 root  wheel   6 Mar 7 10:05 file1
 OWNER: user:root
 GROUP: group:wheel
 SYNTHETIC ACL
 0: user:root allow file_gen_read,file_gen_write,std_write_dac
 1: group:wheel allow file_gen_read
 2: everyone allow file_gen_read
```

Directories can also possess two additional special identities:

- Creator_owner
- Creator_group

For example:

```
drwxrws--- +  1 root  wheel  42 Mar  7 10:41 dir1
 OWNER: user:root
 GROUP: group:wheel
 0: group:wheel allow dir_gen_read,dir_gen_execute,add_file,add_subdir,container_inherit
 1: user:root allow dir_gen_read,dir_gen_write,dir_gen_execute,std_required
 2: creator_owner allow
dir_gen_read,dir_gen_write,dir_gen_execute,std_required,object_inherit,container_inherit,in
herit_only
 3: creator_group allow
dir_gen_read,dir_gen_execute,object_inherit,container_inherit,inherit_only
```

An ACE can optionally contain flags that specify whether it is inherited by subdirectories or files. Inheritance takes place when files and subdirectories are created. Modifying an inherited rule affects only new files and subdirectories, not existing files and subdirectories.

The following flags specify the types of inheritance for permissions in the ACE:

Inheritance type	Description
Object_inherit	Only files in this directory and its descendants inherit the ACE.
Container_inherit	Only directories in this directory and its descendants inherit the ACE,
No_prop_inherit	This ACE does not propagate to descendants (applies to object_inherit and container_inherit ACEs),
Inherit_only	The ACE does not apply for permissions to this object, but does apply to descendants when inherited,
Inherited_ace	The ACE was inherited,

More information about OneFS permissions management is available in the Dell PowerScale OneFS: Authentication, Identity Management, and Authorization white paper.

Permissions inspection and configuration

To help support multiprotocol permissions mapping, OneFS has extended the `chmod` and `ls` CLI commands to provide more fine-grained access control configuration and reporting.

In OneFS, the `ls` command has an `-e` flag extension, which reports on any Windows-style ACEs that the security descriptor contains, and the traditional POSIX mode bits.

For example, consider the following `/ifs/data/file1.txt` file.

A regular long listing (`ls -l`) on this file shows the POSIX mode bits (644) as expected:

```
# ls -l /ifs/data/file1
-rw-r--r-- +  1 root  wheel  6 Feb  12 10:49 /ifs/data/file2
```

The plus (+) sign follows the mode bit representations. It indicates that either the file has an NTFS ACL, or a security descriptor, and that the ACL is NULL. A space (` '`) indicates there is no security descriptor. Every file has a default synthetic ACL that is created automatically, based on the file's mode bits. A file with a synthetic ACL does not have a plus (+) sign next to it, although the ACL is still viewable if the `-e` option is used. The +o' output indicates that a file is wide-open in terms of permissions.

Adding the `e` flag to the command returns both the mode bits, plus the ACL contents:

```
# ls -le /ifs/data/file1
-rw-r--r--    1 root  wheel  6 Feb  12 10:49 /ifs/data/file1
 OWNER: user:root
 GROUP: group:wheel
 SYNTHETIC ACL
 0: user:root allow file_gen_read,file_gen_write,std_write_dac
 1: group:wheel allow file_gen_read
 2: everyone allow file_gen_read
```

The `-e` option for the `ls` command (or the ability to view ACL information in addition to mode bits) is not available from any NFS clients that remotely mount a OneFS export.

Similarly, the `chmod` command in OneFS has an `'a'` mode extension that allows for ACL and ACE configuration and reconfiguration. This extension includes the following options:

`Chmod 'a'` mode	**Description**
+a	Inserts a new ACE into the canonical location in the ACL. If the supplied entry refers to an identity already listed, the two entries are combined.
-a	Deletes ACL entries. All entries exactly matching the supplied entry are deleted. If the entry lists a subset of rights granted by an entry, only the rights listed are removed. Generic rights (generic_all, generic_read, and so on) cannot be removed, they can only be added.
+a#	When a specific ordering is required, `+a#` mode specifies the exact location where an entry is inserted.
=a#	Individual entries are rewritten using the `=a#` mode. **Note**: Some shells require `=` to be escaped with the `\` character.

For example, the following command adds a `deny file read` ACE for the `Administrators` group:

```
# chmod +a group administrators deny file_read /ifs/data/file1

# ls -le /ifs/data/file1
-rw-r--r-- +  1 root  wheel  6 Feb  12 10:49 /ifs/data/file1
 OWNER: user:root
 GROUP: group:wheel
 0: group:Administrators deny file_read
 1: user:root allow file_gen_read,file_gen_write,std_write_dac
 2: group:wheel allow file_gen_read
 3: everyone allow file_gen_read
```

Because POSIX mode bits are a subset of the more comprehensive Windows ACL model, mapping mode bits to ACLs is straightforward. When a Windows client changes the permissions of a file with an ACL, no information is lost because OneFS stores the original ACL and replaces it. Similarly, when a Windows client changes the permissions of a file with mode bits, OneFS replaces the file's synthetic ACL with an actual ACL that is equivalent to the mode bits. However, things are more complex when `chmod` modifies the permissions of a file that are protected by an ACL. OneFS must map the permission changes between two noncorresponding security models by merging the ACL with a patch derived from the change in mode bits. OneFS must be in its default setting.

For example, consider the `file2`.txt file. A Windows client created this file, and the permissions listed are the generic access rights for files and directories. OneFS correctly approximates the mode bits as `764`:

```
# ls -le file2.txt
-rwxrw-r-- + 1 WIND1\nick WIND1\pdm 2056 Feb 12 10:18
file2.txt
OWNER: user:WIND1\nick
GROUP: group:WIND1\pdm
0: user:WIND1\nick allow
file_gen_read,file_gen_write,file_gen_execute,std_write_dac
1: group:WIND1\pdm allow file_gen_read,file_gen_write
2: everyone allow file_gen_read
```

If the mode bits are changed from `764` to `744`, OneFS removes the write permission of the primary group, while preserving the other permissions:

```
# chmod 744 file2.txt
# ls -le file2.txt
-rwxr--r-- + 1 WIND1\nick WIND1\pdm 2056 Feb 12 10:18
file2.txt
OWNER: user:WIND1\nick
GROUP: group:WIND1\pdm
0: user:WIND1\nick allow
file_gen_read,file_gen_write,file_gen_execute,std_write_dac
1: group:WIND1\pdm allow file_gen_read
2: everyone allow file_gen_read
```

Usually, this result preserves the ACL information and minimizes conflicts between actual and expected behavior. However, there are some anomalies of which to be aware.

In the following example, the mode bits and ACEs do not map directly. If the mode bits on the file (`file3.txt`) are changed from `750` (`rwxr-x---`) to `650` (`rw-r-x---`), the resulting merge removes the right to modify the owner in the object's security descriptor. It leaves the user with the standard right to modify the discretionary access control list in the object's security descriptor.

```
# chmod 650 file3.txt
# ls -le file3.txt
-rwxr-x--- + 1 WIND1\nick WIND1\pdm 807 Feb 12 10:19
print.css
OWNER: user: WIND1\nick
GROUP: group: WIND1\pdm
0: user: WIND1\nick allow
```

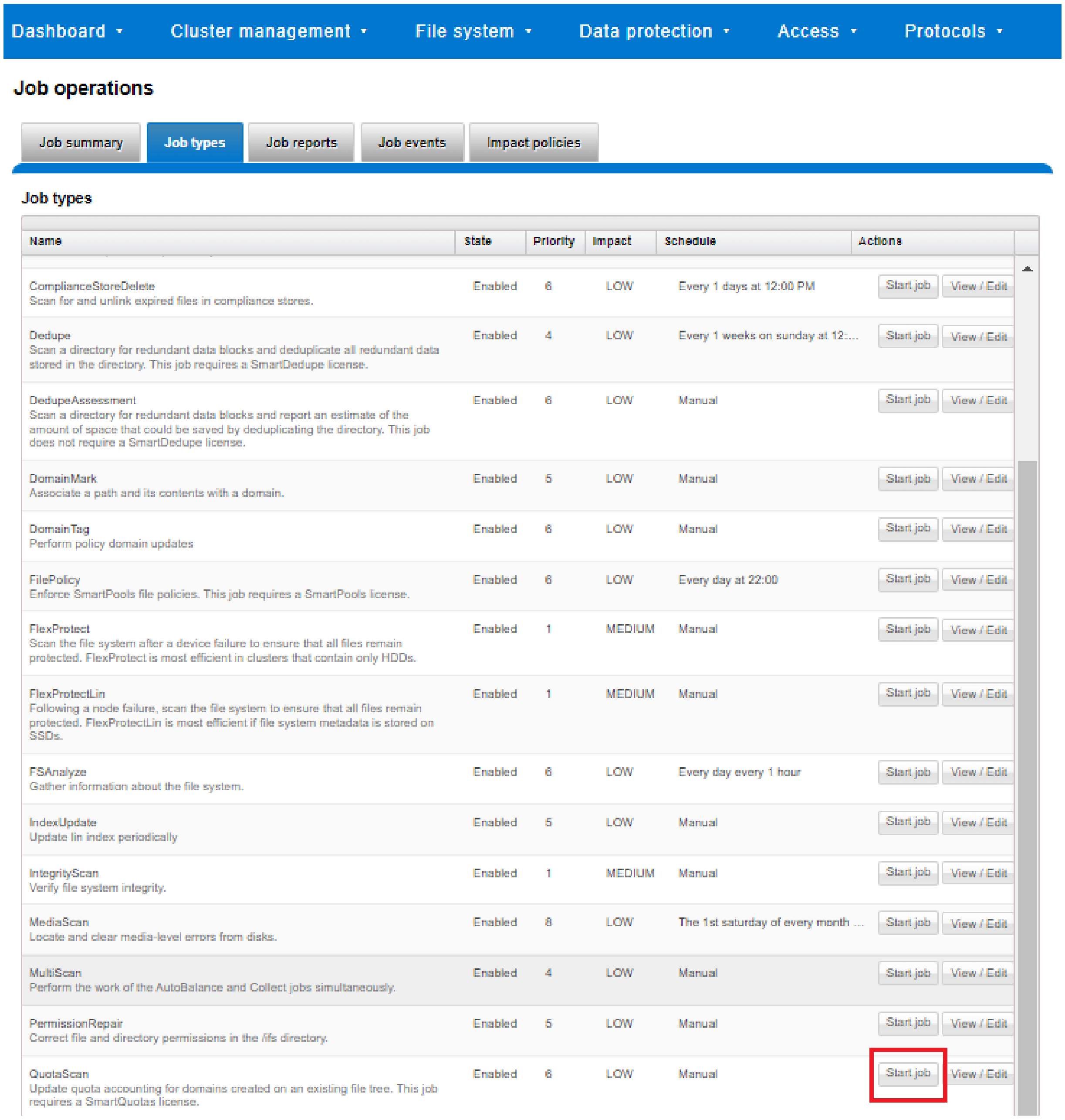
```
file_gen_read,file_gen_write,std_write_dac
1: group: WIND1\pdm allow file_gen_read,file_gen_execute
2: everyone allow std_read_dac,std_synchronize,file_read_attr
```

The following table maps an equivalency of permissions entities between Windows access rights, POSIX mode bits, and OneFS permissions.

POSIX mode bits (approximation)	OneFS representation	Windows ACE
d-w	add_file	FILE_ADD_FILE
d-w	add_subdir	FILE_ADD_SUBDIRECTORY
drwx or -rwx	dir_gen_all, file_gen_all	FILE_ALL_ACCESS
--w- or d-w	append, add_subdir	FILE_APPEND_DATA
d-w	delete_child	FILE_DELETE_CHILD
---x	execute	FILE_EXECUTE
dr--	list	FILE_LIST_DIRECTORY
-r--	file_read_attr	FILE_READ_ATTRIBUTES
-r--	file_read	FILE_READ_DATA
-r--	file_read_ext_attr	FILE_READ_EA
d--x	traverse	FILE_TRAVERSE
--w	file_write_attr	FILE_WRITE_ATTRIBUTES
--w	file_write	FILE_WRITE_DATA
--w	file_write_ext_attr	FILE_WRITE_EA
d-w- or --w	std_delete	DELETE
dr-- or -r--	std_read_dac	READ_CONTROL
drwx or -rwx	std_write_dac	WRITE_DAC
drwx or -rwx	std_write_owner	WRITE_OWNER
N/A	std_synchronize	SYNCHRONIZE

Job execution

The PermissionRepair job can be initiated using the OneFS WebUI by going to **Cluster Management** > **Job Operations** > **Job Types** > **PermissionRepair** and selecting **Start Job**.



Name	State	Priority	Impact	Schedule	Actions
ComplianceStoreDelete Scan for and unlink expired files in compliance stores.	Enabled	6	LOW	Every 1 days at 12:00 PM	Start job View / Edit
Dedupe Scan a directory for redundant data blocks and deduplicate all redundant data stored in the directory. This job requires a SmartDedupe license.	Enabled	4	LOW	Every 1 weeks on sunday at 12:...	Start job View / Edit
DedupeAssessment Scan a directory for redundant data blocks and report an estimate of the amount of space that could be saved by deduplicating the directory. This job does not require a SmartDedupe license.	Enabled	6	LOW	Manual	Start job View / Edit
DomainMark Associate a path and its contents with a domain.	Enabled	5	LOW	Manual	Start job View / Edit
DomainTag Perform policy domain updates	Enabled	6	LOW	Manual	Start job View / Edit
FilePolicy Enforce SmartPools file policies. This job requires a SmartPools license.	Enabled	6	LOW	Every day at 22:00	Start job View / Edit
FlexProtect Scan the file system after a device failure to ensure that all files remain protected. FlexProtect is most efficient in clusters that contain only HDDs.	Enabled	1	MEDIUM	Manual	Start job View / Edit
FlexProtectLin Following a node failure, scan the file system to ensure that all files remain protected. FlexProtectLin is most efficient if file system metadata is stored on SSDs.	Enabled	1	MEDIUM	Manual	Start job View / Edit
FSAnalyze Gather information about the file system.	Enabled	6	LOW	Every day every 1 hour	Start job View / Edit
IndexUpdate Update lin index periodically	Enabled	5	LOW	Manual	Start job View / Edit
IntegrityScan Verify file system integrity.	Enabled	1	MEDIUM	Manual	Start job View / Edit
MediaScan Locate and clear media-level errors from disks.	Enabled	8	LOW	The 1st saturday of every month ...	Start job View / Edit
MultiScan Perform the work of the AutoBalance and Collect jobs simultaneously.	Enabled	4	LOW	Manual	Start job View / Edit
PermissionRepair Correct file and directory permissions in the /ifs directory.	Enabled	5	LOW	Manual	Start job View / Edit
QuotaScan Update quota accounting for domains created on an existing file tree. This job requires a SmartQuotas license.	Enabled	6	LOW	Manual	Start job View / Edit

Figure 2. Starting a PermissionRepair job from the WebUI

Alternatively, you can run the `isi job jobs start` CLI command, with the following available options:

Command option	Description
--path=<*ifs-directory*>	Path to repair permissions on
--mode=<*clone* | *inherit* | *convert*>	Mode for Permission Repair (clone, inherit, or convert)
--type=<*system* | *sid* | *unix* | *native*>	Identity type for Permission Repair (system, native, UNIX, or SID)

Permission Repair job modes

The PermissionRepair job can run in these different modes:

Mode	Description
Clone	Applies the permissions settings for the directory specified by the Template File or Directory setting to the directory you set in the Paths fields
Convert	For each file and directory in the specified Paths fields, converts the owner, group, and ACL to the target on-disk identity based on the Mapping Type setting
Inherit	Recursively applies the ACL of the directory that the Template File or Directory setting specifies to each file and subdirectory in the specified Paths fields, according to standard inheritance rules

Caution: Running the PermissionRepair job in any of its modes affects the current access controls for the specified directory path. A best practice is to test and validate any new permissions configurations on a temporary dataset before making changes that affect the production environment.

Clone mode

In Clone mode, Permission Repair replicates the permissions from a template file or directory on the `/ifs` directory to the target.

Note: Clone mode does not append to the target's permissions but writes the template's permissions over the target's permissions.

Clone mode is preferred when a directory tree with a large file count requires a new set of permissions, such as switching from POSIX mode bits to Windows ACLs.

The following example shows clone mode using the CLI commands:

1. Create files and subdirectories under the target:

```
# touch file1 file2
# mkdir dir1
# touch dir1/file3
```

2. Verify their permissions (POSIX mode bits and Windows ACLs) using the `-e` flag on the `ls` CLI command:

```
# ls -lze /ifs/data/target
total 53
drwxr-xr-x    2 root  wheel  21 Feb 7 10:05 dir1
```

```
 OWNER: user:root
 GROUP: group:wheel
 SYNTHETIC ACL
0: user:root allow
dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,dele
te_child
 1: group:wheel allow dir_gen_read,dir_gen_execute
 2: everyone allow dir_gen_read,dir_gen_execute
-rw-r--r--    1 root  wheel   6 Feb 7 10:05 file1
 OWNER: user:root
 GROUP: group:wheel
 SYNTHETIC ACL
 0: user:root allow
file_gen_read,file_gen_write,std_write_dac
 1: group:wheel allow file_gen_read
 2: everyone allow file_gen_read
-rw-r--r--    1 root  wheel   6 Feb 7 10:05 file2
 OWNER: user:root
 GROUP: group:wheel
 SYNTHETIC ACL
 0: user:root allow
file_gen_read,file_gen_write,std_write_dac
 1: group:wheel allow file_gen_read
 2: everyone allow file_gen_read
```

3. Modify the template using the `chmod` command's `+a` flag to add an ACE of user `admin` with `delete child` rights (the ACE with index number `0` below):

```
# chmod 777 /ifs/data/template
# chmod +a user admin allow delete_child /ifs/data/template
# ls -lze /ifs/data
...
drwxrwxrwx +  2 root  wheel   0 Feb 7 08:45 template
 OWNER: user:root
 GROUP: group:wheel
 0: user:admin allow delete_child
 1: user:root allow
dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,dele
te_child
 2: group:wheel allow
dir_gen_read,dir_gen_write,dir_gen_execute,delete_child
 3: everyone allow
dir_gen_read,dir_gen_write,dir_gen_execute,delete_child
```

4. Start Permission Repair in clone mode to copy the template's permissions to the target's permissions:

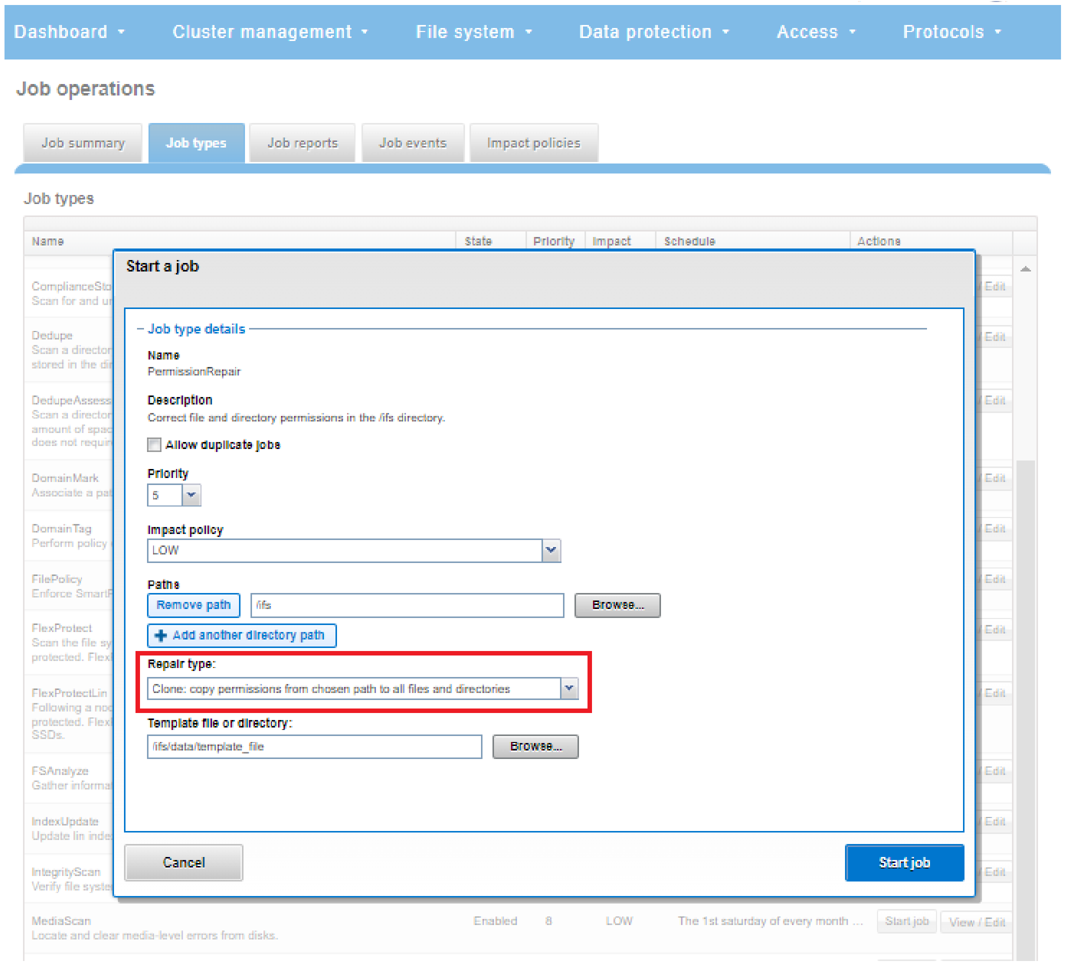
```
# isi job jobs start permissionrepair --mode=clone --
template=/ifs/data/template --path=/ifs/data/target
```

5. Confirm that the `admin` user permissions were copied from the template to the target directory and contents:

```
# ls -lze /ifs/data/target
total 53
drwxrwxrwx +  2 root  wheel  21 Feb 7 10:08 dir1
OWNER: user:root
 GROUP: group:wheel
 0: user:admin allow delete_child
 1: user:root allow
dir_gen_read,dir_gen_write,dir_gen_execute,std_write_dac,dele
te_child
 2: group:wheel allow
dir_gen_read,dir_gen_write,dir_gen_execute,delete_child
 3: everyone allow
dir_gen_read,dir_gen_write,dir_gen_execute,delete_child
-rwxrwxrwx +  1 root  wheel   6 Feb 7 10:08 file1
 OWNER: user:root
 GROUP: group:wheel
 0: user:admin allow delete_child
 1: user:root allow
file_gen_read,file_gen_write,file_gen_execute,std_write_dac,d
elete_child
 2: group:wheel allow
file_gen_read,file_gen_write,file_gen_execute,delete_child
 3: everyone allow
file_gen_read,file_gen_write,file_gen_execute,delete_child
-rwxrwxrwx +  1 root  wheel   6 Feb 7 10:08 file2
 OWNER: user:root
 GROUP: group:wheel
 0: user:admin allow delete_child
 1: user:root allow
file_gen_read,file_gen_write,file_gen_execute,std_write_dac,d
elete_child
 2: group:wheel allow
file_gen_read,file_gen_write,file_gen_execute,delete_child
 3: everyone allow
file_gen_read,file_gen_write,file_gen_execute,delete_child
```

Note: Changing the modes and ACLs of the template and running Permission Repair in clone mode copies the template directory's permissions to the target.

PermissionRepair can also be started in clone mode using the WebUI, by going to **Job Operations** > **Job Types** > **PermissionRepair** > **Start Job**, and selecting **Clone** from the **Repair Type** drop-down menu:



Start a job

Job type details

Name
PermissionRepair

Description
Correct file and directory permissions in the /ifs directory.

Allow duplicate jobs

Priority
5

Impact policy
LOW

Paths
Remove path /ifs Browse...
Add another directory path

Repair type:
Clone: copy permissions from chosen path to all files and directories

Template file or directory:
/ifs/data/template_file Browse...

Cancel Start job

MediaScan — Locate and clear media-level errors from disks. — Enabled — 8 — LOW — The 1st saturday of every month ... — Start job — View / Edit

Figure 3. Permission Repair clone mode

Convert mode

Convert mode changes all the file and directory on-disk identities under the target path to the identity type specified by the Mapping Type field. The following table lists the on-disk identity type options:

Identity type	Description
Global	Applies the system's default identity
SID (Windows)	Applies the Windows identity
UNIX	Applies the UNIX identity
Native	If a user or group does not have an authoritative UNIX identifier (UID or GID), applies the Windows identity (SID)

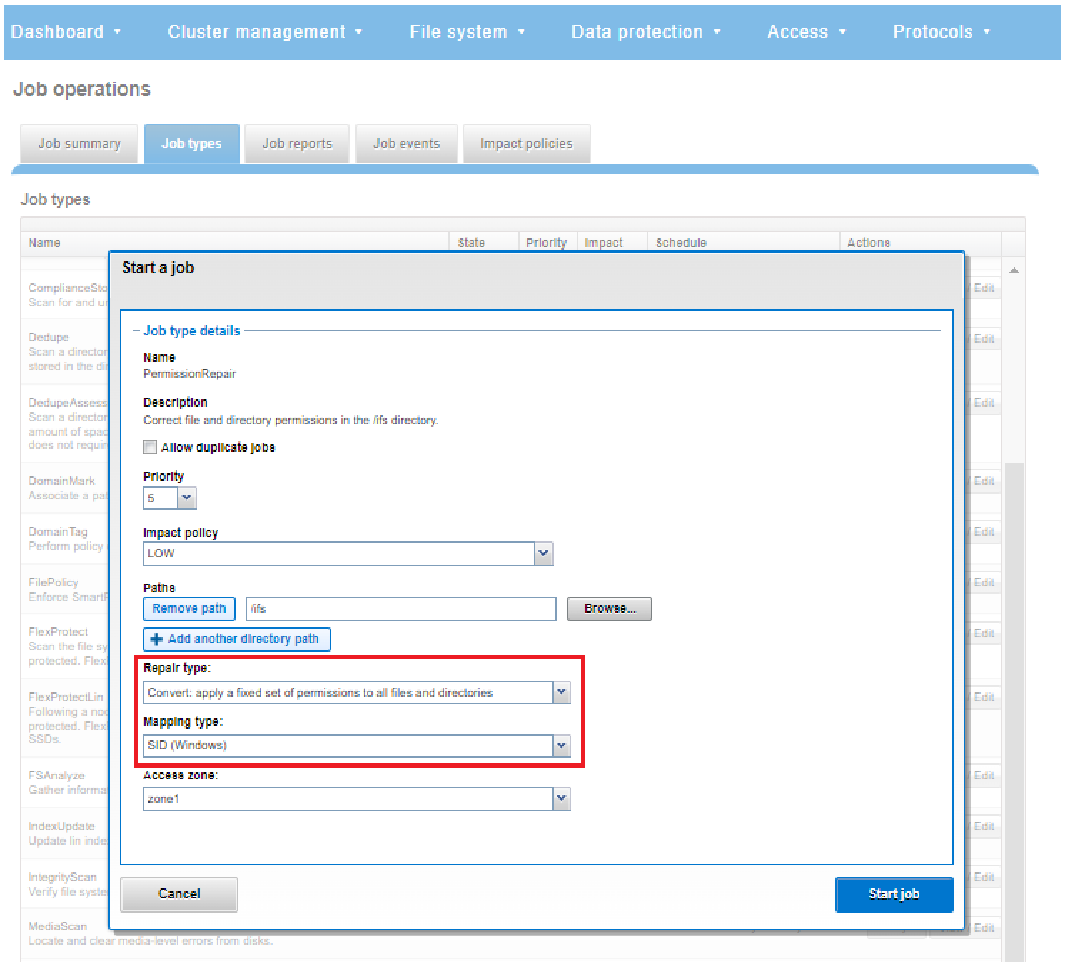
Convert mode is typically used for modifying the on-disk identity type and wanted permission settings.

For example, the following command converts the `/ifs/data/target` directory path to UNIX identity type:

```
# isi job jobs start permissionrepair --mode=convert --mapping-
type=UNIX --path=/ifs/data/target
```

PermissionRepair can also be started in convert mode by using the WebUI. Go to **Job Operations** > **Job Types** > **PermissionRepair** > **Start Job**, select **Convert** from the **Repair Type** drop-down menu, and choose the wanted **Mapping Type**.

For example, to convert `/ifs/data/test1` to the Windows SID mapping type:



Start a job

Job type details

Name
PermissionRepair

Description
Correct file and directory permissions in the /ifs directory.

Allow duplicate jobs

Priority: 5

Impact policy: LOW

Paths: /ifs

Repair type: Convert: apply a fixed set of permissions to all files and directories

Mapping type: SID (Windows)

Access zone: zone1

Figure 4. Permission Repair convert mode

A template directory is not required for the Permission Repair job when it is run in convert mode as nothing is being cloned or inherited.

Note: If you change the on-disk identity type, run the PermissionRepair job with the Convert repair type selected to ensure that the disk representation of all files is consistent with the changed setting.

As for the general job itself, PermissionRepair must be started manually (that is, not scheduled) and it runs by default with a `low` impact policy and priority value of `6` –

although these default settings can be changed if wanted. If configuring a target directory, it must be below the `/ifs` directory in the file system hierarchy.

Inherit mode

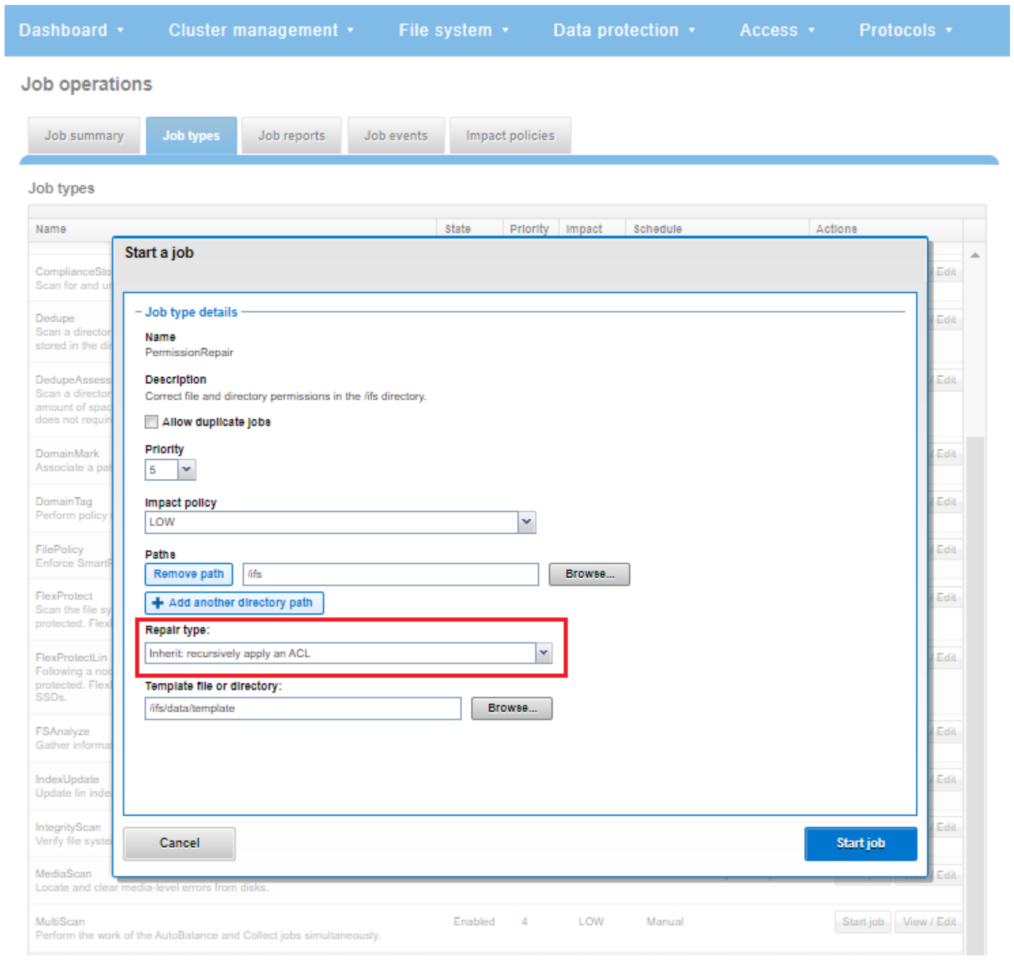
When Permission Repair is run in inherit mode, the current permissions of directories or files under the target directory are overwritten (not appended) by only the inheritable permissions of the template directory.

Inherit mode is typically used whenever an inherited ACE is added to an existing directory tree. Unlike Windows, which automatically traces all parent directories looking for inherited ACEs and adds them to the access check, OneFS manages ACL inheritance by explicitly adding the inherited ACE onto the security descriptor of the file or directory. This method means that OneFS can only respect inherited ACLs on newly created files and directories. As such, the Inherit mode PermissionRepair job needs to be run to update any objects in a tree that were present before an inherited ACL was added to the parent directory.

For example, the following CLI command runs the inherit-mode Permission Repair job on the `/ifs/data/test1` file using the `/ifs/data/template` directory as the template directory.

```
# isi job jobs start permissionrepair --mode=inherit --
template=/ifs/data/template --path=/ifs/data/test1
```

The PermissionRepair job can also be started in inherit mode using the WebUI, by going to **Job Operations** > **Job Types** > **PermissionRepair** > **Start Job**, and selecting **Inherit** from the **Repair Type** drop-down menu:



Start a job

Job type details

Name
PermissionRepair

Description
Correct file and directory permissions in the /ifs directory.

☐ Allow duplicate jobs

Priority
5

Impact policy
LOW

Paths
Remove path /ifs Browse...
Add another directory path

Repair type:
Inherit: recursively apply an ACL

Template file or directory:
/ifs/data/template Browse...

Figure 5. Permission Repair inherit mode

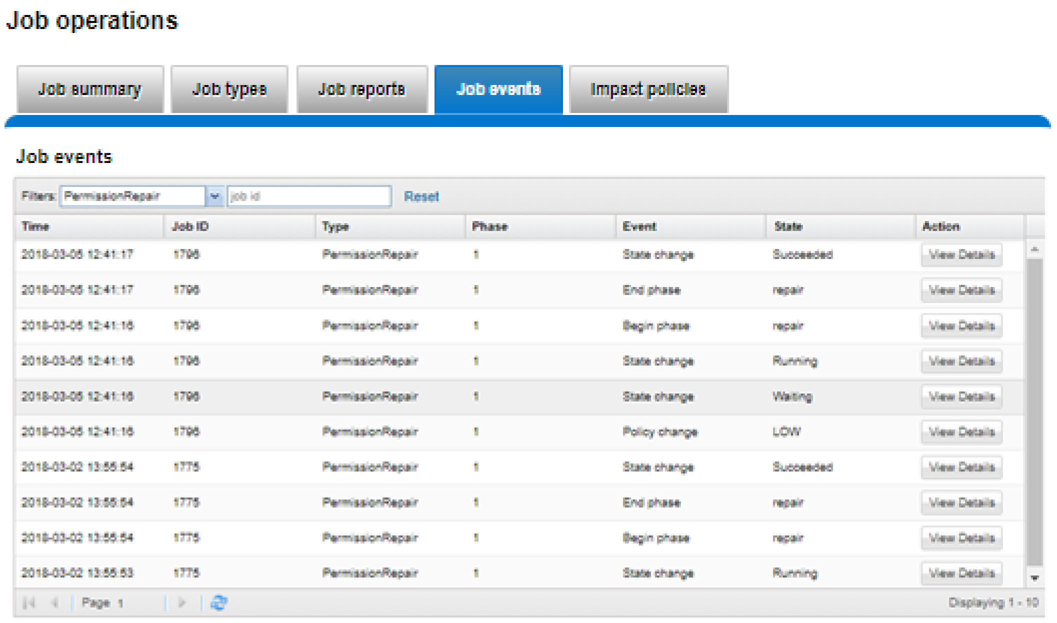
Note: With inherit mode, the template must be a directory because a file does not possess inheritable permissions.

Job monitoring and reporting

The OneFS Job Engine provides detailed monitoring and statistics gathering, with insight into jobs and the Job Engine. Various Job Engine specific metrics are available using the OneFS CLI, including per job disk usage and so on. For example, worker statistics and job level resource usage can be viewed with the `isi job statistics list` CLI command. Also, the status of the Job Engine workers is available using the OneFS `isi job statistics view` CLI command.

Job events

PermissionRepair job events, including pause/resume, waiting, job success, failure, and so on, are reported under the **Job Events** tab of the WebUI. Additional information is available by clicking **View Details** for the appropriate job events entry in the WebUI. These events are accessed by going to **Cluster Management** > **Job Operations** > **Job Events**. A filter can be added to display all PermissionRepair job events or a specific job ID:

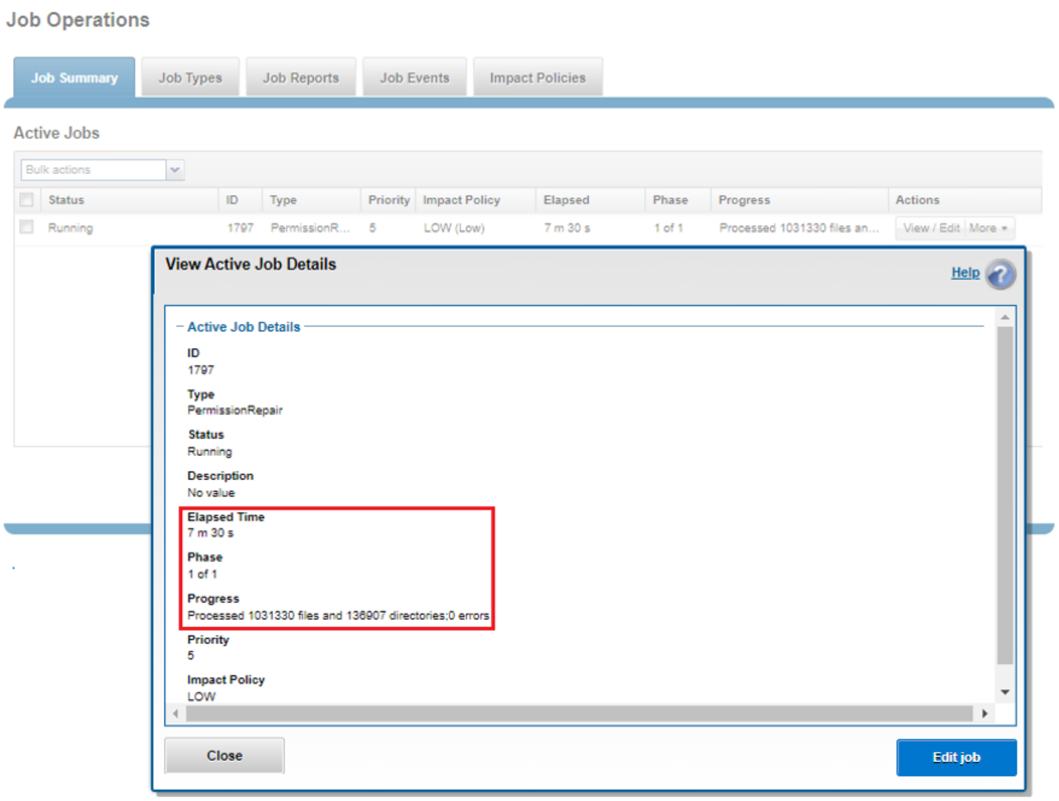


Time	Job ID	Type	Phase	Event	State	Action
2018-03-05 12:41:17	1796	PermissionRepair	1	State change	Succeeded	View Details
2018-03-05 12:41:17	1796	PermissionRepair	1	End phase	repair	View Details
2018-03-05 12:41:16	1796	PermissionRepair	1	Begin phase	repair	View Details
2018-03-05 12:41:16	1796	PermissionRepair	1	State change	Running	View Details
2018-03-05 12:41:16	1796	PermissionRepair	1	State change	Waiting	View Details
2018-03-05 12:41:16	1796	PermissionRepair	1	Policy change	LOW	View Details
2018-03-02 13:55:54	1775	PermissionRepair	1	State change	Succeeded	View Details
2018-03-02 13:55:54	1775	PermissionRepair	1	End phase	repair	View Details
2018-03-02 13:55:54	1775	PermissionRepair	1	Begin phase	repair	View Details
2018-03-02 13:55:53	1775	PermissionRepair	1	State change	Running	View Details

Figure 6. PermissionRepair job events

Active job details

While the PermissionRepair job is running, an Active Job Details report is also available. This report provides contextual information, including elapsed time, current job phase, job progress status, and so on.



Status	ID	Type	Priority	Impact Policy	Elapsed	Phase	Progress	Actions
Running	1797	PermissionR...	5	LOW (Low)	7 m 30 s	1 of 1	Processed 1031330 files an...	View / Edit More

ID
1797

Type
PermissionRepair

Status
Running

Description
No value

Elapsed Time
7 m 30 s

Phase
1 of 1

Progress
Processed 1031330 files and 136907 directories;0 errors

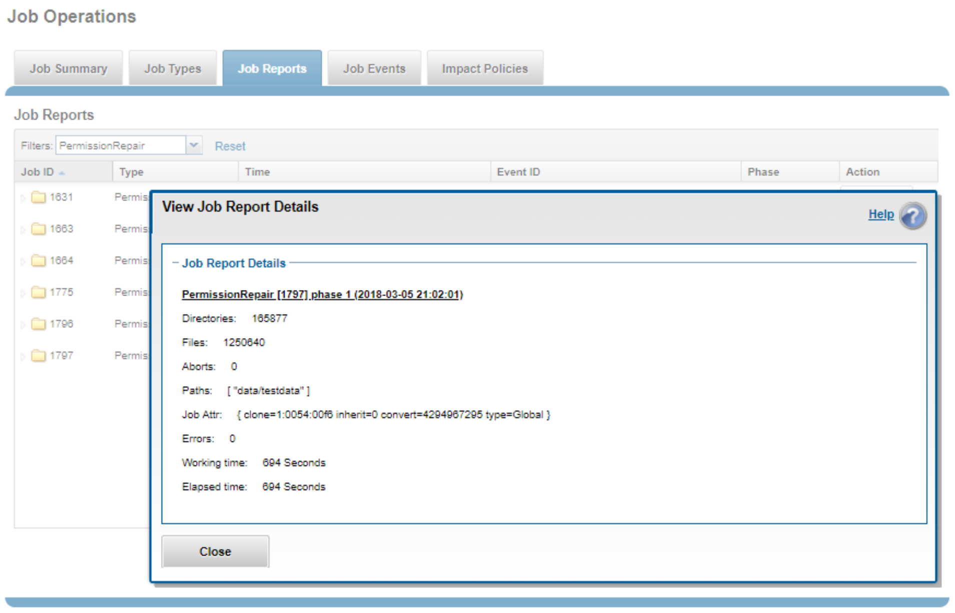
Priority
5

Impact Policy
LOW

Figure 7. Permission Repair active job details

Job reports

A report is also provided for each PermissionRepair job. This report contains detailed information about runtime, the number of objects scanned, and other work details or errors.



PermissionRepair [1797] phase 1 (2018-03-05 21:02:01)

Directories: 165877

Files: 1250640

Aborts: 0

Paths: ["data/testdata"]

Job Attr: { clone=1:0054:00f6 inherit=0 convert=4294967295 type=Global }

Errors: 0

Working time: 694 Seconds

Elapsed time: 694 Seconds

Figure 8. PermissionRepair job report

Similar active job details and job report information are available from the CLI using the following syntax:

```
# isi job jobs view <job ID>
```

OneFS performance resource management provides statistics for the resources used by jobs - both cluster-wide and per-node. This information is provided by using the `isi statistics workload` CLI command. Available in a `top` format, this command displays the top jobs and processes, and periodically updates the information.

For example, the following syntax shows, and indefinitely refreshes, the top five processes on a cluster:

```
# isi statistics workload --limit 5 --format=top
last update: 2018-03-05T16:45:25 (s)ort: default

    CPU Reads Writes   L2  L3 Node          SystemName            JobType
377.7ms   0.0   5.9k 25.8 0.0    3          Job: 1798 PermissionRepair[0]
368.6ms   0.0    0.0  0.0 0.0    3              system                  -
328.9ms   0.0   4.9k 18.4 0.0    1          Job: 1798 PermissionRepair[0]
326.2ms   0.0    0.0  0.0 0.0    2               celog                  -
262.3ms   0.0   3.4k 52.1 0.0    2          Job: 1798 PermissionRepair[0]
```

The resource statistics tracked per job, per job phase, and per node include CPU, reads, writes, and L2 and L3 cache hits. Unlike the output from the `top` command, this information makes it easier to diagnose individual job resource issues, and so on.

Summary

Dell PowerScale overcomes the problems that undermine traditional NAS systems by combining the three traditional layers of storage architecture—file system, volume manager, and data protection—into a scale-out NAS cluster with a distributed file system. The scale-out architecture of OneFS eliminates controller bottlenecks to reduce wall-clock runtimes for concurrent jobs, accelerates metadata operations, improves storage efficiency to lower capital expenditures, centralizes management to reduce operating expenses, and delivers strategic advantages to increase competitiveness in a global market.

Take the next step

Contact your Dell sales representative or authorized reseller to learn more about how Dell PowerScale scale NAS storage solutions can benefit your organization.

Visit Dell PowerScale to compare features and get more information.