Technical Report

NetApp Element software custom protection domains

Feature description and deployment guide

Rafal Sydor, NetApp

June 2022 | TR-4927

Abstract

This document describes custom protection domain supported by NetApp® Element® software. It also provides a general description of system features, the setup process, best practices and configuration scenarios.

TABLE OF CONTENTS

Introduction

This document describes protection site capabilities, which are part of NetApp Element software on NetApp SolidFire storage systems. The objective of this document is to share best practices and recommendations for custom protection domain configurations.

Protection domains

A protection domain is a set of nodes grouped such that any part—or even all of it—might fail while maintaining data availability. Protection domains enable a storage cluster to continue to serve data if part or all of the domain is lost.

Protection domain levels

NetApp Element software offers different types or levels of protection domains.

Type/level	Properties
Node	Cluster default. Primary and secondary metadata and block copies are distributed on the different nodes.
Chassis	At the chassis level, automatically detected. Primary and secondary metadata and block copies are distributed between the chassis.
Custom	At the custom level, the user assigns each node to a specific protection domain. NetApp SolidFire Double Helix data layout ensures that secondary metadata and block copies span domains.

SolidFire has historically provided data protection at the node level using Double Helix data availability, where no two copies of metadata and block data exist on the same node. This protection level is the default for all SolidFire storage clusters.

Element 11.0 introduced a chassis-based protection domain. NetApp H410-X are the only chassis-based storage nodes, bundled as the NetApp HCI offering. Chassis are automatically detected, and the Element Plug-in for VMware vCenter Server configures and monitors chassis-based protection domains.

In the Element 12.0 release, NetApp added the ability to protect sites beyond the node or chassis level as a custom protection domain. This new feature protects against the concurrent failure of one or more nodes defined within a custom protection domain.

Element 12.0 provides a limited release to configure custom data protection using the API only. General availability of this feature was launched with Element 12.5.

Custom protection domain

Use cases

The reality is that most companies face challenges from accelerated virtualization, data growth, or regulatory compliance. IT teams are challenged to come up with architectures that protect data from various scenarios such as:

- Concurrent failure of one or more nodes
- Failure of one or more racks
- Data center halls or full data center unavailability

NetApp Element 12.5 (or later) offers site or domain level protection capabilities that enable you to architect and deploy NetApp SolidFire clusters to mitigate and address such scenarios.

Custom protection domains allow you to protect against the concurrent failure of one or more nodes defined within a custom protection domain. They can be configured by using the API or UI to match specific data protection outcomes.

System requirements

- Hardware: any NetApp SolidFire nodes supported by version 12.5.
- Software: Element 12.5 or later (UI, management, GA release).
- A minimum of 3 domains and 6 nodes, with 2 nodes per domain.
- The protection domain supports clusters with a maximum of 40 nodes.

  Note: There can be a total of 20 custom protection domains within any given cluster (2 nodes per protection domain).
- All nodes need to reside on the same L2 broadcast domain.

How custom protection domains work

An Element cluster provides custom domain protection by assigning block data and metadata replicas to nodes residing in separate protection domains. Additionally, ensemble members need to be distributed among a minimum of three different protection domains. The Element cluster achieves tolerant status if these conditions are met.

There are two levels of protection that can be achieved when a custom protection domain is configured:

- **Tolerance:** ability to continue to serve data after the loss of a protection domain (partial or full domain).
- **Resiliency:** ability to continue to serve data after the loss of a protection domain (partial or full domain) with data auto-healing to protect against subsequent failures.

Tolerance

Custom protection domains are used by four subsystems:

- **Ensemble members:** The ensemble is a function with the cluster to ensure cluster consistency across multiple independent nodes. This subsystem assigns membership across separate custom protection domains to ensure that cluster quorum is maintained in the event of a domain failure. At least three ensemble services must be running across at least three separate custom protection domains to maintain tolerance levels.
- **Block services:** This subsystem assigns its duplicates block data to separate protection domains. The system maintains two copies of all block data and no two copies of a given block reside on the same custom protection domain.
- **Slice services:** This subsystem assigns its replicas and standbys (metadata) to services in separate protection domains. The system maintains two copies of all metadata data, and no two metadata copies reside on the same custom protection domain.
- **VMware vVols protocol endpoint providers:** Assigned primary and secondary protocol endpoints are distributed between different custom protection domains.

These subsystems are independent of each other, and each reaches the highest level of protection that is possible. If any of these subsystems are degraded, the cluster has lost custom protection domain tolerance. If any subsystem tries to use the layout at a level and fails, it then tries the next lower level (for example, chassis, node).

Note: vVols protocol endpoints are only considered for clusters where vVols are enabled. Customers not using vVols do not need to be concerned about vVols as in impact to customer protection domains.

For a tolerance example, see Appendix A.

Protection domain resiliency

Resiliency is determined by the amount of block data that can fit into all but the largest custom protection domain. Achieving resiliency requires tolerance and a sufficient number of surviving nodes with enough available capacity to hold all the data (block and metadata) outside the largest protection domain. In addition, protection domain resiliency is validated at metadata, and ensemble.

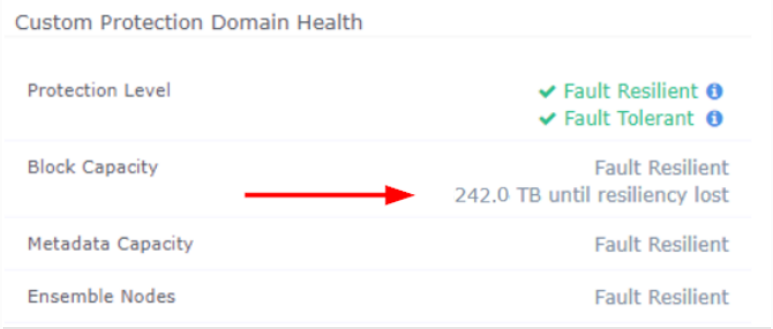
The formula for resiliency threshold for block data:

```
Total Cluster Capacity – Largest Protection Domain Capacity – ( Total Cluster Capacity x 3/100 )
```

Note: 3/100 = `cBinSyncHeadroomPercent`

This threshold is the point at which the cluster wouldn't be able to auto-heal data after the failure of the largest protection domain based on the desired custom protection domain layout. Resiliency threshold is displayed in the Element UI under custom protection domain health reporting.

Note: Element cluster does not enforce capacity reservation; NetApp recommends that you monitor resiliency threshold through API or Active IQ.



Custom Protection Domain Health

Protection Level	✓ Fault Resilient ⓘ ✓ Fault Tolerant ⓘ
Block Capacity	Fault Resilient 242.0 TB until resiliency lost
Metadata Capacity	Fault Resilient
Ensemble Nodes	Fault Resilient

See Appendix B for an example of the block data resiliency threshold.

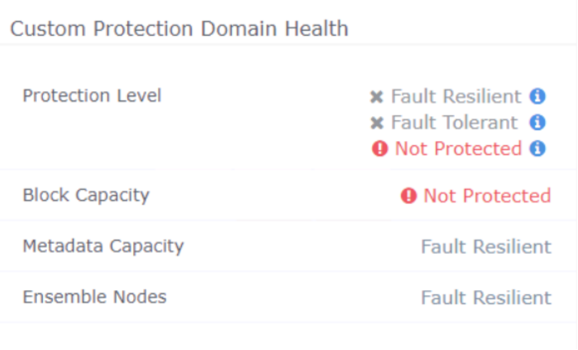
Data protection health, faults, and alerts

When custom protections domain are configured, the Element UI provides health information related to this service.

There are three possible levels of protection:

Not protected. The storage cluster cannot guarantee protection from the failure of one or more of its custom protection domain subsystems (block, metadata, vVols protocol endpoint providers, or ensemble).

If a protection domain is lost, the cluster might not be able to maintain data availability.

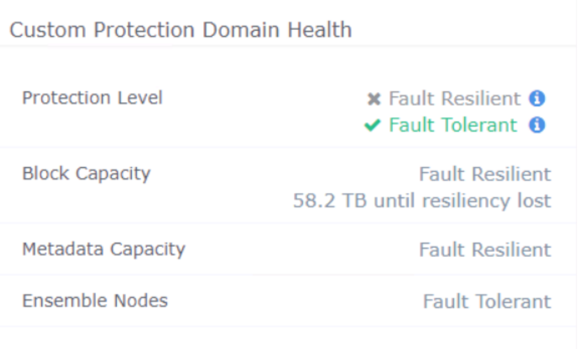


Custom Protection Domain Health

Protection Level	✖ Fault Resilient ⓘ ✖ Fault Tolerant ⓘ ❗ Not Protected ⓘ
Block Capacity	❗ Not Protected
Metadata Capacity	Fault Resilient
Ensemble Nodes	Fault Resilient

The previous screenshot shows that the cluster has lost protection domain tolerance because block data cannot achieve its tolerance. This status refers only to protection at the custom domain level. Tolerance and resiliency are still maintained at the node level in this example.

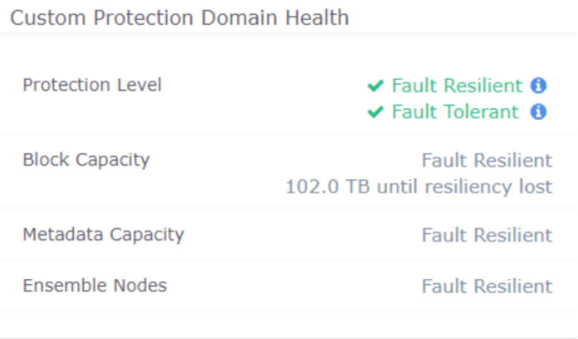
- **Fault tolerant.** The storage cluster has distributed its data, metadata and ensemble members such that it will prevent data unavailability after the failure of one of its custom protection domains. The cluster will continue to serve data after the loss of the custom protection domain.



Custom Protection Domain Health

Protection Level	✖ Fault Resilient ✓ Fault Tolerant
Block Capacity	Fault Resilient 58.2 TB until resiliency lost
Metadata Capacity	Fault Resilient
Ensemble Nodes	Fault Tolerant

- **Fault resilient.** The storage cluster has enough free capacity to self-heal and keep data copies distributed between nodes.

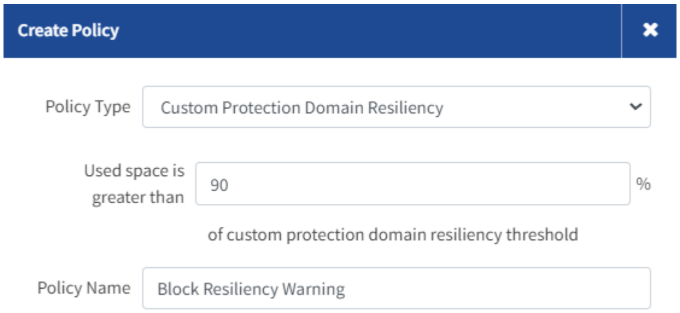


Custom Protection Domain Health

Protection Level	✓ Fault Resilient ✓ Fault Tolerant
Block Capacity	Fault Resilient 102.0 TB until resiliency lost
Metadata Capacity	Fault Resilient
Ensemble Nodes	Fault Resilient

Custom protection domain alerts

The NetApp SolidFire Active IQ® tool allows monitoring and alerting for all clusters. You can set up policy email alerts so that you are notified if the used space reaches a specific percentage of the custom protection domain resiliency threshold.



Create Policy

Policy Type	Custom Protection Domain Resiliency
Used space is greater than	90 % of custom protection domain resiliency threshold
Policy Name	Block Resiliency Warning

Design considerations

Tolerance only versus tolerance with resiliency

In implementing a custom protection domain, you can aim to accomplish tolerance only or tolerance and resiliency. Before implementing a custom protection domain, carefully consider the trade-off between these two design options:

- **Tolerance only**. Reserving capacity to auto-heal at the custom domain level is not required. This only guarantees that data availability is protected through the failure of any custom protection domain. It makes no guarantees of auto-healing from that situation.
- **Tolerance with resiliency.** The system auto-heals after the failure of any custom protection domain, but only guarantees that it can reach node level tolerance. This option requires maintaining block capacity below resiliency threshold.

Users might want to consider the duration of protection they require when deciding whether to design for resiliency. If for example, you are implementing a rack level domain scheme, and you have an SLA of 4 hours to recover a rack from a failure, you might decide that resiliency (auto-heal) is not required since the time to rebuild the data might be longer than the 4-hour SLA to recover a rack. If the goal is to be able to withstand extended outages of any domain, resiliency should always be a requirement. Cluster with more protection domains uses fewer nodes per domain and always auto-heal faster as capacity per domain is smaller (example: 24 node cluster with six vs four domains)

Stretch cluster using custom protection domains

Deployment of custom domains enables a NetApp Element cluster to be stretched between multiple data centers. Such deployment introduces extra complexity, so NetApp recommends that you follow these guidelines:

- All nodes must share the same layer 2 broadcast domain. NetApp recommends using dedicated networks.
- Storage node network interfaces should be allocated 10Gbps links (or more) per node. For example, a 15-node cluster with three domains (5 nodes per domain) would require 50Gbps links between all three data center locations.
- Latencies should not exceed 2ms the round-trip time (RTT) with packet loss less than 2%. Network bandwidth and latency should be tested and validated between all data center sites in scope. These tests should be performed before any Element cluster is configured.

  Note: The distance between the custom domains in multiple data centers determines the network's latency and packet loss.
- iSCSI traffic is not governed by any domain affinity. Clients will need to access nodes in other domains for read/write traffic on a regular basis. Ensure adequate bandwidth is available to support this traffic.
- Network port requirements can be found in Appendix E.

Custom protection domain layouts

Mapping storage nodes to the correct custom domain plays an essential role in ensuring that domains are correctly balanced and maintaining system availability if a domain is lost.

Here are some best practices:

- Keep capacity between all domains the same or as close as possible. This approach prevents stranded capacity.

- For clusters where capacity is not homogeneous across domains ensure that the largest domains do not exceed 33% of cluster capacity - this best practice ensures resiliency is *possible* in all configs. In addition, this safeguard enables the largest domains to retain a buffer for NetApp Element to maintain custom protection domain tolerance if drives or nodes fail in the smaller domains. (For an example, see the “Highly unbalanced configuration—scenario 2” section.)
- When you design for resiliency*,* usable capacity should always be allocated below the resiliency threshold for block data (refer to the “Protection domain resiliency” section).
- Define the desired redundancy for the protection domain. For example, a user could implement a domain solution with four domains and reserve enough capacity such that one domain could fail, and the cluster could auto heal back to full domain level resiliency. (N+1, N+2).

  Note: A cluster with only three custom protection domains cannot regain tolerance if one of the domains fails, since at that point, it will only have two custom protection domains.

- When you are configuring a custom domain layout (using the UI or API), the system does not check that the resulting cluster will be properly balanced. To minimize block data movement, NetApp recommends that you validate that the correct group of nodes belongs to the right custom domain.

  Note: Data redistribution starts immediately, and any mistakenly applied domain layout configuration is queued until the initial synchronizing completes. Depending on cluster capacity and its fullness, this could take minutes, hours, or days to complete.

For an example of the proper layout and balance of custom domains, see Appendix C.

Partially defined custom layout

When some nodes are assigned a custom protection domain, and others are not, we say that the custom layout is partially defined. This situation is problematic because the cluster does not know which custom protection domain should be associated with the unassigned nodes.

This situation temporarily occurs when new nodes join a cluster with an existing custom protection domains. New nodes have unsigned status, and the cluster avoids using the newly added nodes. This approach maintains custom-level tolerance and prevents unnecessary data synchronization.

Growing and shrinking a cluster with custom protection domains

To add new nodes to the cluster with the existing custom protection domains, follow these steps:

1. Configure new nodes by using the terminal user interface (TUI).
2. On the local cluster, go to Cluster > Nodes > Pending.
3. Select the new nodes that will join the cluster.
4. Go to Cluster > Nodes and select Configure Protection Domain.
5. If nodes need to be assigned to a new custom domain, enter the new custom protection domain.
6. Assign each new node to the correct custom domain.
7. Go to Cluster > Drives > Available.
8. Add block drives.

   Note: To minimize data movement, maintain balance between the custom domains by adding equal capacity to each domain simultaneously.

9. Wait for block drives to finish synchronizing (Reporting > Running Tasks).
10. Add metadata drives.

    Note: To minimize data movement, add metadata drives that belong in the same custom domain.

11. Wait for metadata drives to finish synchronizing (Reporting > Running Tasks).
12. Validate the custom protection domain health (Reporting page).

To remove nodes from the cluster with the existing custom protection domains defined, follow these steps:

1. Move primaries away from this node.

```
https://<MVIP>/json-
rpc/12.0?method=MovePrimariesAwayFromNode&nodeID=<nodeID>&perMinutePrimarySwapLimit=30
```

2. Go to Cluster > Drives > Active.
3. Remove metadata drives on that node.

   Note: To minimize data movement, remove metadata drives that belong in the same custom domain.

4. Wait for metadata to finish synchronizing (Reporting > Running Tasks).
5. Remove block drives.

   Note: To minimize data movement, maintain the balance between the custom domains by removing equal capacity from each domain simultaneously.

6. Wait for all blocks to finish synchronizing (Reporting > Running Tasks).
7. Go to Cluster > Nodes > Active.
8. Remove nodes.

   Note: Nodes should not have active drives assigned.

Replacing cluster nodes with an in-use custom protection domain

When all nodes or partial nodes need to be replaced (for example, in a cluster refresh), and the existing cluster is running a custom protection domain, follow these steps:

1. Configure new nodes by using the TUI.
2. On the local cluster, go to Cluster > Nodes > Pending.
3. Select new nodes that will join the cluster.
4. Go to Cluster > Nodes and select Configure Protection Domain.
5. Assign all new nodes to the right custom domains.
6. If possible, to minimize data movement, add and remove nodes in a way that preserves the balance of capacity between all custom protection domains.

   Note: To minimize data movement, the next steps are done for subsets of nodes (new and existing nodes) across all custom protection domains. NetApp recommends that you add or remove subset of drives from all protection domains simultaneously. Refer to following KB article

https://kb.netapp.com/Advice_and_Troubleshooting/Data_Storage_Software/Element_Software/How_to_add_or_remove_drives_from_a_cluster_on_Element

   a. Choose the first set of drives (new nodes) across of domains to be added.
   b. Go to Cluster > Drives > Available.
   c. Add block drives.
   d. Wait for block drives to finish synchronizing (Reporting > Running Tasks).
   e. Add metadata drives.
   f. Wait for metadata drives to finish synchronizing (Reporting > Running Tasks).
   g. Go to Cluster > Nodes > Active and validate that the any new node was promoted as cluster master. If not, promote it by issuing the following command using the correct node ID.

```
https:// <MVIP>/json-rpc/12.0?method=PromoteClusterMaster&nodeID=<nodeID>
```

   h. Choose the first set of drives (existing nodes) to be removed.

i. Move primaries away from this node.

```
https://<MVIP>/json-
rpc/12.0?method=MovePrimariesAwayFromNode&nodeID=<nodeID>&perMinutePrimarySwapLimit=30
```

j. Go to Cluster > Drives > Active.

k. Remove the metadata drives.

l. Wait for metadata to finish synchronizing (Reporting > Running Tasks).

m. Remove block drives.

n. Wait for all blocks to finish synchronizing (Reporting > Running Tasks).

o. Go to Cluster > Nodes > Active and remove the node.

p. Repeat steps a to o for other subset of drives across all domains. Skip step g (which is only performed once on the first new node).

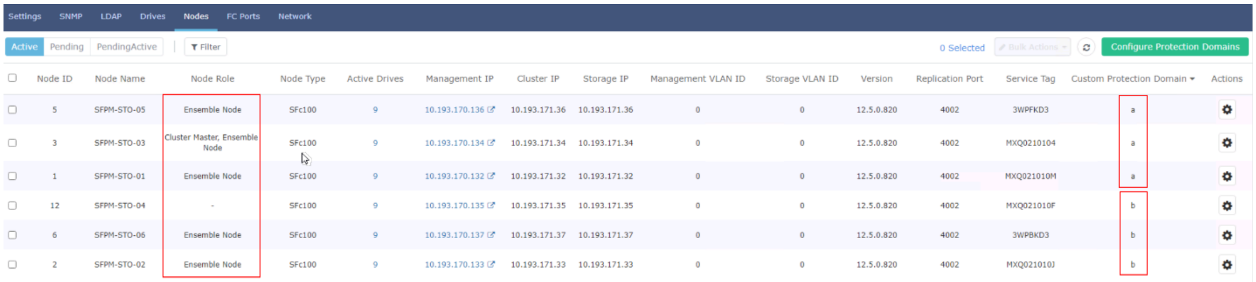
7. When you finish, validate the custom protection domain health status (Reporting UI).

Preventing an unbalanced or invalid cluster layout

It is possible to configure invalid custom domain layouts or highly unbalanced configurations. The following sections cover a few of the scenarios that you should avoid.

Two protection domains

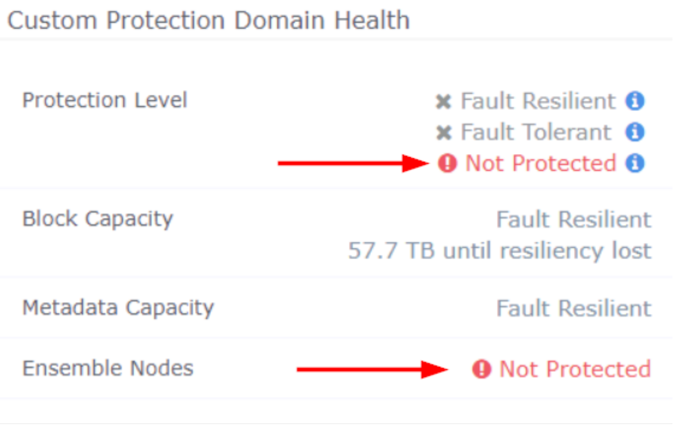
In this scenario, two custom domains (a and b) are implemented with half the nodes in the cluster assigned to each custom domain.



□	Node ID	Node Name	Node Role	Node Type	Active Drives	Management IP	Cluster IP	Storage IP	Management VLAN ID	Storage VLAN ID	Version	Replication Port	Service Tag	Custom Protection Domain	Actions
□	5	SFPM-STO-05	Ensemble Node	SFc100	9	10.193.170.136	10.193.171.36	10.193.171.36	0	0	12.5.0.820	4002	3WPFKD3	a	⚙
□	3	SFPM-STO-03	Cluster Master, Ensemble Node	SFc100	9	10.193.170.134	10.193.171.34	10.193.171.34	0	0	12.5.0.820	4002	MXQ0210104	a	⚙
□	1	SFPM-STO-01	Ensemble Node	SFc100	9	10.193.170.132	10.193.171.32	10.193.171.32	0	0	12.5.0.820	4002	MXQ021010M	a	⚙
□	12	SFPM-STO-04	-	SFc100	9	10.193.170.135	10.193.171.35	10.193.171.35	0	0	12.5.0.820	4002	MXQ021010F	b	⚙
□	6	SFPM-STO-06	Ensemble Node	SFc100	9	10.193.170.137	10.193.171.37	10.193.171.37	0	0	12.5.0.820	4002	3WPBKD3	b	⚙
□	2	SFPM-STO-02	Ensemble Node	SFc100	9	10.193.170.133	10.193.171.33	10.193.171.33	0	0	12.5.0.820	4002	MXQ021010J	b	⚙

Tolerance can never be achieved due to ensemble requirements (three domains minimum).

Ensemble nodes are not protected, which is displayed in the custom protection domain health UI.



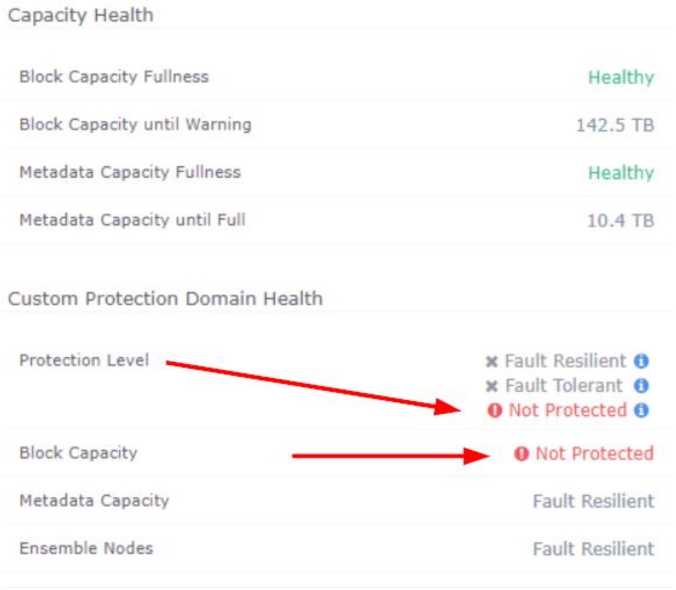
Custom Protection Domain Health

Protection Level	✖ Fault Resilient ℹ ✖ Fault Tolerant ℹ ❗ Not Protected ℹ
Block Capacity	Fault Resilient 57.7 TB until resiliency lost
Metadata Capacity	Fault Resilient
Ensemble Nodes	❗ Not Protected

Highly unbalanced configuration—scenario 1

In this scenario, an end user configures a custom protection domain on a new cluster in a highly unbalanced configuration such as PD1=2 node, PD2=3 nodes, PD3=10 nodes (all nodes have equal capacity).

These three domains are used properly by all subsystems (metadata, vVols, ensemble) except for block data. However, the largest (PD3) exceeds 50% capacity of the cluster and custom-level tolerance cannot be achieved. Block data allocation would be undermined by this imbalanced layout and would strand the capacity of five nodes. Block data would only reach node-level tolerance.



Capacity Health

Block Capacity Fullness	Healthy
Block Capacity until Warning	142.5 TB
Metadata Capacity Fullness	Healthy
Metadata Capacity until Full	10.4 TB

Custom Protection Domain Health

Protection Level	✖ Fault Resilient ⓘ ✖ Fault Tolerant ⓘ ❗ Not Protected ⓘ
Block Capacity	❗ Not Protected
Metadata Capacity	Fault Resilient
Ensemble Nodes	Fault Resilient

Highly unbalanced configuration—scenario 2

In this scenario, an end user configures a custom protection domain on a new cluster in a highly unbalanced configuration such as PD1=2 node, PD2=3 nodes, PD3=5 nodes (all nodes have equal capacity).

In this example, these three domains are used properly by all four subsystems. The largest (PD3) provides 50% capacity of the cluster, which is valid but not recommended. If one of the drives or a node fails on PD1 or PD2, the 50% capacity would be exceeded, meaning that just as in scenario 1, only node-level tolerance could be achieved.

This situation illustrates why NetApp recommends that the largest domain never exceed 33% of the entire cluster capacity before any custom domains are defined.

Note: Use the custom protection domain health UI section to verify whether the protection domain reaches a tolerance level.

APIs

Custom protection domains use three different API commands.

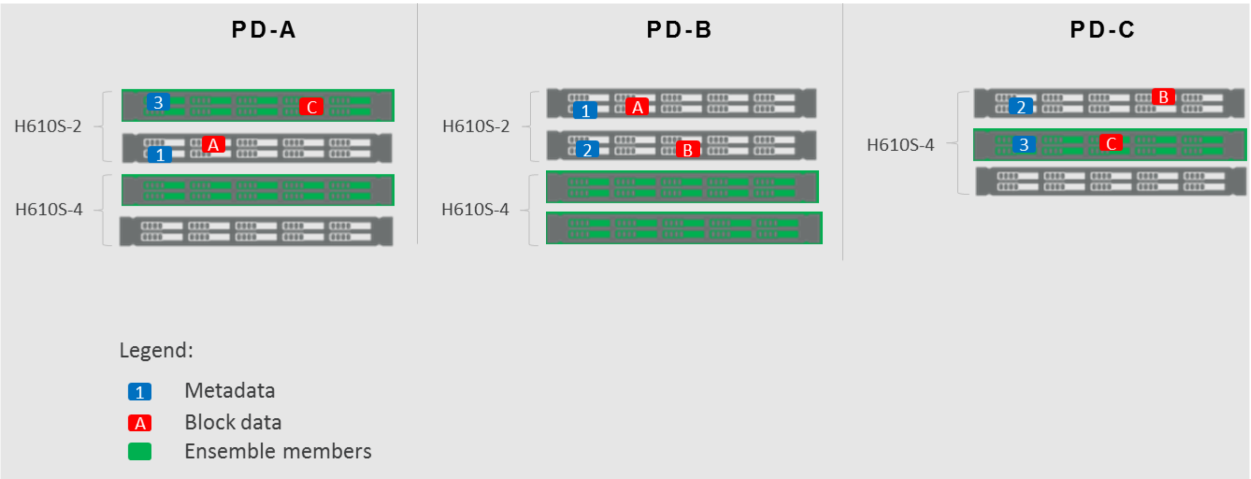
- `SetProtectionDomainLayout`: defines the protection domain layout to the cluster.
- `GetProtectionDomainLayout`: provides protection domain layout information between all levels (node, chassis, custom).

- `ListprotectionDomainLevels`: provides the current tolerance and resiliency of the cluster for the different subsystems at each protection domain level.

For API examples, see Appendix D.

Appendix A: Custom domain tolerance

This example examines a cluster with 11 nodes (4xH610S-2 and 7H610S-4) and compare the effects of three versus four custom domains.

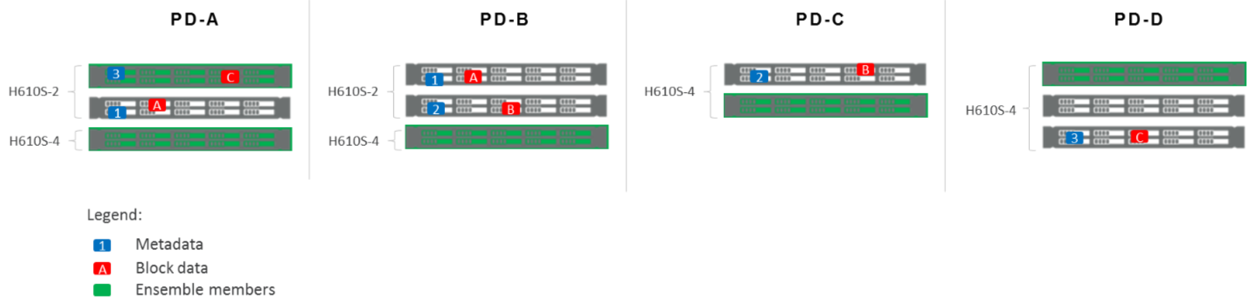


Node #	Node type	Block capacity	Custom domain	Custom domain capacity		
				A	B	C
Node 1	H610S-2	21120	A	21120		
Node 2	H610S-2	21120	A	21120		
Node 3	H610S-2	21120	B		21120	
Node 4	H610S-2	21120	B		21120	
Node 5	H610S-4	42240	A	42240		
Node 6	H610S-4	42240	A	42240		
Node 7	H610S-4	42240	B		42240	
Node 8	H610S-4	42240	B		42240	
Node 9	H610S-4	42240	C			42240
Node 10	H610S-4	42240	C			42240
Node 11	H610S-4	42240	C			42240
Domain capacity (GB)				126720	126720	126720
Domain capacity (%)				33.3%	33.3%	33.3%
Cluster capacity (GB)				**380160**		

Allocating all nodes among three custom domains provides a balance capacity (126TB) for each custom domain. Our capacity threshold per domain is within the NetApp recommended 33.3% capacity allocation. We can sustain two concurrent node failures in PD-C and maintain custom protection tolerance. However, if the goal is to retain tolerance during an entire domain failure (for example, PD-C), using a three-custom-domain layout is not ideal, because ensemble members would not achieve its tolerance with only two remaining custom domains.

Using the same cluster, a four-custom-domain layout strands capacity in PD-D (unbalance). The largest PD-D capacity is at 126TB, which represents 33% of total cluster capacity and is also within the NetApp recommended best practice threshold.

During any custom domain failure, we can sustain custom-level tolerance (by all subsystems including ensemble) and can handle two-node (PD-C) or three-node failures.



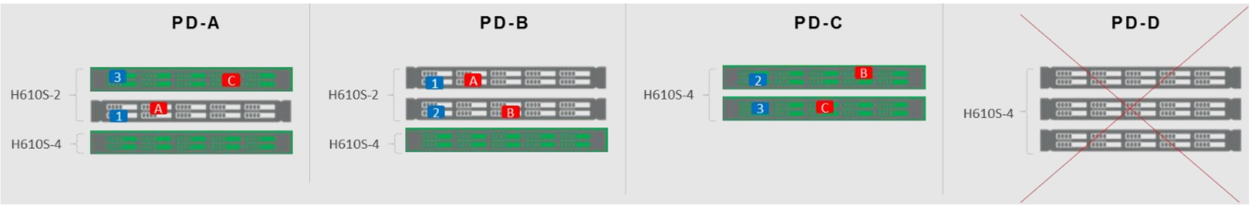
Node #	Node type	Block capacity	Custom domain	Custom domain capacity			
				A	B	C	D
Node 1	H610S-2	21120	A	21120			
Node 2	H610S-2	21120	A	21120			
Node 3	H610S-2	21120	B		21120		
Node 4	H610S-2	21120	B		21120		
Node 5	H610S-4	42240	A	42240			
Node 6	H610S-4	42240	B		42240		
Node 7	H610S-4	42240	C			42240	
Node 8	H610S-4	42240	C			42240	
Node 9	H610S-4	42240	D				42240
Node 10	H610S-4	42240	D				42240
Node 11	H610S-4	42240	D				42240
Domain capacity (GB)				84480	84480	84480	**126720**
Domain capacity (%)				22.2%	22.2%	22.2%	**33.4%**
Cluster capacity (GB)				**380160**			

This illustrates that having four custom domains in an unbalanced layout is more beneficial, because we can tolerate the same number of node failures or an entire domain failure while retaining the tolerance level.

Appendix B: Custom domain resiliency

Using the same example from Appendix A (a four-domain layout), let’s examine resiliency.

If the largest domain (PD-D) fails, would there be enough domains and capacity to auto-heal at the desired custom layout?



Using the formula from the “Protection domain resiliency” section, we can establish or validate the resiliency threshold.

Node #	Node type	Block capacity	Custom domain	Custom domain capacity			
				A	**B**	**C**	**D**
Node 1	H610S-2	21120	A	21120			
Node 2	H610S-2	21120	A	21120			
Node 3	H610S-2	21120	B		21120		
Node 4	H610S-2	21120	B		21120		
Node 5	H610S-4	42240	A	42240			
Node 6	H610S-4	42240	B		42240		
Node 7	H610S-4	42240	C			42240	
Node 8	H610S-4	42240	C			42240	
Node 9	H610S-4	42240	D				42240
Node 10	H610S-4	42240	D				42240
Node 11	H610S-4	42240	D				42240
Capacity at each domain (GB)				84480	84480	84480	**126720**
Cluster capacity (GB)				**380160**			

Block Capacity Resiliency Threshold: 380160GB – 126720GB – (380160GB x 3/100) = 242TB.

Data auto-healing can be maintained at the custom domain layout up to 242TB.

Note: Any data allocated beyond this point would be auto-healed at the node level during any domain failure.

Ensemble members are resilient (five members) and remain active on PD-A, PD-B, PD-C (quorum is satisfied).

Note: A three-custom-domain layout is not considered, because resiliency would not be possible if any domain were to fail.

Appendix C: Evenly balanced custom domains

This example illustrates four custom domains that are correctly balanced with the ability to auto-heal up to 70.4TB at the custom domain level.

Node #	Node type	Block capacity	Custom domain	Custom domain capacity			
				A	**B**	**C**	**D**
Node 1	H610S-2	21120	A	21120			
Node 2	H610S-2	21120	A	21120			

Node #	Node type	Block capacity	Custom domain	Custom domain capacity			
				A	B	C	D
Node 3	H610S-2	21120	B		21120		
Node 4	H610S-2	21120	B		21120		
Node 5	H610S-2	21120	C			21120	
Node 6	H610S-2	21120	C			21120	
Node 7	H610S-2	21120	D				21120
Node 8	H610S-2	21120	D				21120
Capacity at each domain (GB)				42240	42240	42240	42240
Domain capacity (%)				**25%**	**25%**	**25%**	**25%**
Cluster capacity (GB)				**168960**			

Block Capacity Resiliency Threshold: 168960GB – 42240GB – (168960GB x 3/100) = 70.4TB

Block Capacity Tolerance Threshold: 42240GB / 168960GB = 25%

Appendix D: API

The following layout examples involve three protection domains and six nodes.

This example demonstrates the `SetProtectionDomainLayout` method:

```
{
"method": "SetProtectionDomainLayout",
    "params": {
        "protectionDomainLayout": [
            {
                "nodeID": 1,
                "protectionDomains": [
                    {
                        "protectionDomainName": "Site A",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 2,
                "protectionDomains": [
                    {
                        "protectionDomainName": "Site A",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 3,
                "protectionDomains": [
                    {
                        "protectionDomainName": "Site B",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 4,
                "protectionDomains": [
                    {
                        "protectionDomainName": "Site B",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 5,
                "protectionDomains": [
                    {
                        "protectionDomainName": "Site C",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 6,
                "protectionDomains": [
                    {
                        "protectionDomainName": "Site C",
                        "protectionDomainType": "custom"
                    }
                ]
            }
        ]
    },
    "id": 1
}
```

This example demonstrates the `GetProtectionDomainLayout` method:

```
{
    "id": 1,
    "result": {
        "protectionDomainLayout": [
            {
                "nodeID": 1,
                "protectionDomains": [
                    {
                        "protectionDomainName": "85L3JB2",
                        "protectionDomainType": "chassis"
                    },
                    {
                        "protectionDomainName": "Site A",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 2,
                "protectionDomains": [
                    {
                        "protectionDomainName": "H0WHJB2",
                        "protectionDomainType": "chassis"
                    },
                    {
                        "protectionDomainName": "Site A",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 3,
                "protectionDomains": [
                    {
                        "protectionDomainName": "85N7JB2",
                        "protectionDomainType": "chassis"
                    },
                    {
                        "protectionDomainName": "Site B",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 4,
                "protectionDomains": [
                    {
                        "protectionDomainName": "CDXJFB2",
                        "protectionDomainType": "chassis"
                    },
                    {
                        "protectionDomainName": "Site B",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 5,
                "protectionDomains": [
                    {
                        "protectionDomainName": "CGXJFB2",
                        "protectionDomainType": "chassis"
                    },
                    {
                        "protectionDomainName": "Site C",
                        "protectionDomainType": "custom"
                    }
                ]
            },
            {
                "nodeID": 6,
                "protectionDomains": [
                    {
                        "protectionDomainName": "4HJJFB2",
                        "protectionDomainType": "chassis"
                    },
                    {
                        "protectionDomainName": "Site C",
                        "protectionDomainType": "custom"
                    }
                ]
            }
        ]
    }
}
```

This example demonstrates the `ListprotectionDomainLevels` method:

```
{
    "id": 1,
    "result": {
        "protectionDomainLevels": [
            {
                "protectionDomainType": "node",
                "resiliency": {
                    "protectionSchemeResiliencies": [
                        {
                            "protectionScheme": "doubleHelix",
                            "sustainableFailuresForBlockData": 1,
                            "sustainableFailuresForMetadata": 1
                        }
                    ],
                    "singleFailureThresholdBytesForBlockData": 40712384020480,
                    "sustainableFailuresForEnsemble": 2
                },
                "tolerance": {
                    "protectionSchemeTolerances": [
                        {
                            "protectionScheme": "doubleHelix",
                            "sustainableFailuresForBlockData": 1,
                            "sustainableFailuresForMetadata": 1
                        }
                    ],
                    "sustainableFailuresForEnsemble": 2
                }
            },
            {
                "protectionDomainType": "chassis",
                "resiliency": {
                    "protectionSchemeResiliencies": [
                        {
                            "protectionScheme": "doubleHelix",
                            "sustainableFailuresForBlockData": 1,
                            "sustainableFailuresForMetadata": 1
                        }
                    ],
                    "singleFailureThresholdBytesForBlockData": 40712384020480,
                    "sustainableFailuresForEnsemble": 2
                },
                "tolerance": {
                    "protectionSchemeTolerances": [
                        {
                            "protectionScheme": "doubleHelix",
                            "sustainableFailuresForBlockData": 1,
                            "sustainableFailuresForMetadata": 1
                        }
                    ],
                    "sustainableFailuresForEnsemble": 2
                }
            },
            {
                "protectionDomainType": "custom",
                "resiliency": {
                    "protectionSchemeResiliencies": [
                        {
                            "protectionScheme": "doubleHelix",
                            "sustainableFailuresForBlockData": 1,
                            "sustainableFailuresForMetadata": 1
                        }
                    ],
                    "singleFailureThresholdBytesForBlockData": 32072642068480,
                    "sustainableFailuresForEnsemble": 1
                },
                "tolerance": {
                    "protectionSchemeTolerances": [
                        {
                            "protectionScheme": "doubleHelix",
                            "sustainableFailuresForBlockData": 1,
                            "sustainableFailuresForMetadata": 1
                        }
                    ],
                    "sustainableFailuresForEnsemble": 1
                }
            }
        ]
    }
}
```

Appendix E: TCP port requirements

Port type	Port number	Usage of port
ICMP		Cluster-to-cluster latency
TCP-HTTP	2181	Ensemble nodes communication
TCP-RPC	4000–4020	Data communications from node to node
TCP-HTTPS	442	Node access to cluster
TCP-HTTPS	443	Cluster administration, API communication.

Where to find additional information

To learn more about the information described in this technical report, refer to the following documents:

- NetApp Element Software User Guide
  https://docs.netapp.com/us-en/element-software/index.html

Version history

Version	Date	Document version history
Version 1.0	June 2022	Initial release

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

Copyright information

Trademark information

TR-4927-0622

NetApp