

# Dell EMC ECS: High Availability Design

## Abstract

This paper describes the architectural details about how the Dell EMC™ ECS platform provides enterprise availability.

June 2021

## Revisions

Date	Description
July 2017	Initial release
August 2017	Updated to include ECS version 3.1 content
March 2019	Updated to include ECS version 3.3 content
April 2020	Updated 'TSO Behavior with Access during Outage Enabled'
December 2020	Updated the metadata protected method
June 2021	Updated to include ECS version 3.6.1 content

## Acknowledgements

This paper was produced by the following:

Author: [Zhu, Jarvis](#)

The information in this publication is provided "as is." Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any software described in this publication requires an applicable software license.

This document may contain certain words that are not consistent with Dell's current language guidelines. Dell plans to update the document over subsequent future releases to revise these words accordingly.

This document may contain language from third party content that is not under Dell's control and is not consistent with Dell's current guidelines for Dell's own content. When such third-party content is updated by the relevant third parties, this document will be revised accordingly.

Copyright © 2017–2021 Dell Inc. or its subsidiaries. All Rights Reserved. Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners. [6/8/2021] [Technical White Paper] [H16344.6]

# Table of contents

Revisions.....	2
Acknowledgements.....	2
Table of contents .....	3
Executive summary.....	5
Terminology .....	5
<b>1 High availability design overview.....</b>	<b>6</b>
1.1 Chunks.....	6
1.2 ECS metadata .....	6
1.3 Failure domains .....	8
1.4 Advanced data-protection methods.....	9
1.4.1 Triple mirror .....	9
1.4.2 Erasure coding with redundant data segments .....	10
1.4.3 Triple mirror plus in place erasure coding .....	10
1.4.4 Inline erasure coding .....	11
1.5 Erasure coding protection levels .....	12
1.5.1 Default erasure coding scheme (12+4): .....	12
1.5.2 Cold storage erasure coding scheme (10+2): .....	12
1.6 Checksums .....	13
1.7 Writing objects .....	13
1.8 Reading objects.....	14
<b>2 Local site availability.....</b>	<b>16</b>
2.1 Disk failure.....	16
2.2 ECS node failure.....	17
2.2.1 Multiple-node failures .....	18
<b>3 Multi-site design overview .....</b>	<b>22</b>
3.1 Chunk manager tables .....	24
3.2 XOR encoding .....	25
3.3 Replicate to all sites.....	26
3.4 Write data flow in geo-replicated environment .....	27
3.5 Read data flow in geo-replicated environment .....	28
3.6 Update data flow in geo-replicated environment.....	29
<b>4 Multi-site availability .....</b>	<b>31</b>
4.1 Temporary site outage (TSO).....	31
4.1.1 Default TSO behavior .....	32

4.1.2 TSO behavior with access during outage enabled.....	34
4.1.3 Multiple site failures .....	43
4.2 Permanent site outage (PSO) .....	43
4.2.1 PSO with geo-passive replication.....	45
4.2.2 Recoverability from multiple site failures .....	48
5 Conclusion.....	50
A Technical support and resources .....	51
A.1 Related resources.....	51

## Executive summary

Enterprises are storing ever-increasing amounts of data that are critical to keep available. The costs and complexities of having to restore huge amounts of data in the event of system or site failures can be overwhelming to an IT organization.

The Dell EMC™ ECS™ platform has been designed to address both the capacity and availability needs of today's enterprises. ECS provides exabyte scalability with support for a globally distributed object infrastructure. It is architected to provide enterprise availability with automatic failure detection and self-recovery options built in.

This paper describes the architectural details about how ECS provides enterprise availability. It includes specifics such as:

- How the distributed infrastructure provides increased system availability
- Advanced data protection methods that provide data durability
- How data is distributed for optimal availability
- Automatic failure detection
- Built-in self-healing methods
- Disk, node and network failure resolution details
- Disaster recovery:
  - How ECS protects against site wide failures
  - How consistency is maintained in an active-active multi-site configuration
  - How site-wide failures are detected
  - Access options during a site outage
  - How data durability is re-established after a permanent site-wide failure

## Terminology

**Virtual data center (VDC):** In this paper, the term virtual data center (VDC) is used synonymously with site or zone. ECS resources in a single VDC must be part of the same internal management network.

**Geo-federation:** You can deploy ECS software in multiple data centers to create a geo-federation. In a geo-federation, ECS behaves as a loosely coupled federation of autonomous VDCs. Federating sites involves providing replication and management endpoints for communication between sites. Once sites are federated, they can be managed as a single infrastructure from any node in the federation.

**Replication groups:** Replication groups define where data is protected. A local replication group contains a single VDC and protects data within the same VDC against disk or node failures. Global replication groups contain more than one VDC, and protect data against disk, node, and site failures. Replication groups are assigned at the bucket level.

# 1 High availability design overview

High availability can be described in two main areas: system availability and data durability. A system is available when it can respond to a client request. Data durability is provided independent of system availability and provides guarantees for data being stored in the system without loss or corruption. This means that even if the ECS system is down (for example, a network outage) the data is still protected.

The distributed nature of the ECS architecture provides system availability by allowing any node in a virtual data center (VDC)/site to respond to client requests. If a node goes down the client can be redirected either manually or automatically (for example, using DNS or a load balancer) to another node that can service the request.

ECS uses a combination of triple mirroring and erasure coding to write data in a distributed fashion so as to be resilient against disk and node failures. ECS supports replication between sites to increase the availability and resiliency by protecting against site wide failures. ECS also includes regular systematic data integrity checks with self-healing capability.

When looking at high availability it is first important to understand the architecture and how data is distributed for optimal availability and performance within ECS.

## 1.1 Chunks

A chunk is a logical container that ECS uses to store all types of data, including object data, custom client provided metadata, and ECS system metadata. Chunks contain 128 MB of data consisting of one or more objects from a single bucket as shown in Figure 1.



Chunk = 128 MB of data

Figure 1 Logical chunk

ECS uses indexing to keep track of all the data within a chunk, see the 1.2 for more details.

## 1.2 ECS metadata

ECS maintains its own metadata that keeps track of where data exists as well as transaction history. This metadata is maintained in logical tables and journals.

The tables hold key-value pairs to store information relating to the objects. A hash function is used to do fast lookups of values associated with a key. These key-value pairs are stored in a B+ tree for fast indexing of data locations. By storing the key-value pair in a balanced, searched tree format, like a B+ Tree, the location of the data and metadata can be accessed quickly. In addition, to further enhance query performance of these logical tables, ECS implements a two-level log-structured merge (LSM) tree. Thus, there are two tree-like structures where a smaller tree is in memory (memory table) and the main B+ tree resides on disk. So, lookup

of key-value pairs will first query the memory table and if the value is not memory, it will look in the main B+ tree on disk.

Transaction history is recorded in journal logs and these logs are written to disks. The journals keep track of index transactions not yet committed to the B+ tree. After the transaction is logged into a journal, the memory table is updated. Once the table in memory becomes full or after a set period of time, the table is merged sorted or dumped to B+ tree on disk and a checkpoint is recorded in the journal. This process is illustrated in Figure 2.

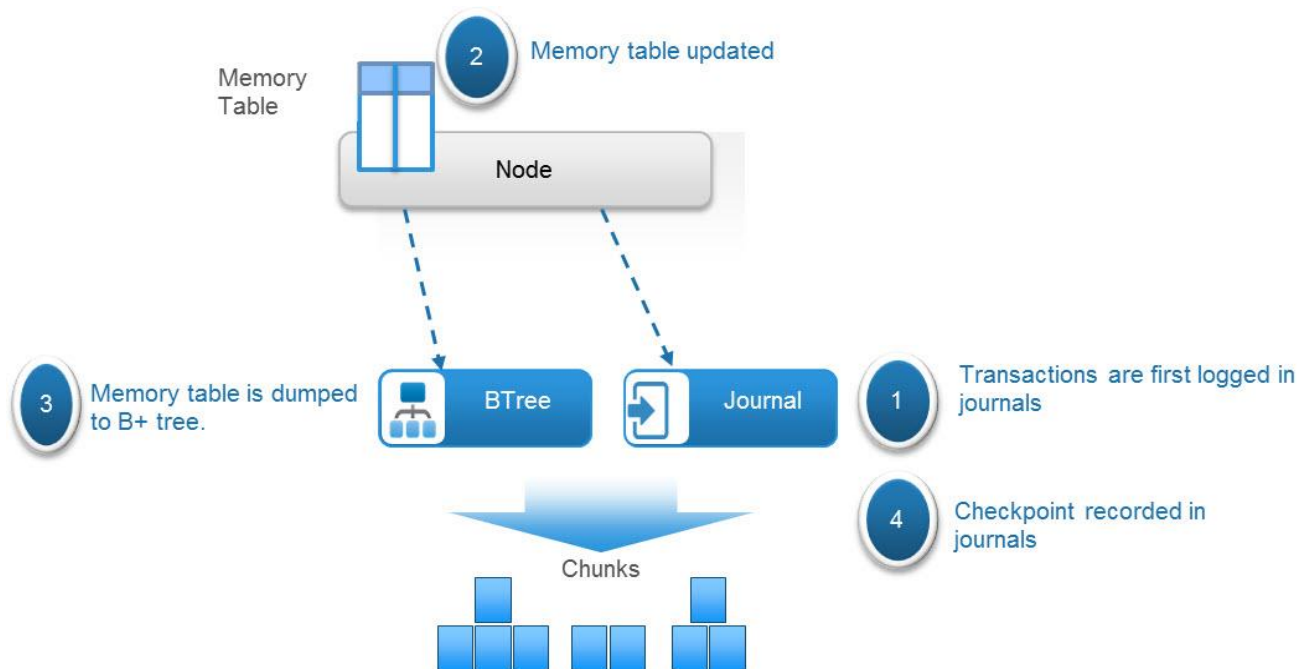


Figure 2 Workflow of transaction updates to ECS tables

Both the journals and B+ trees are written to chunks.

There are several different tables that ECS uses, each of which can get very large. In order to optimize performance of table lookups, each table is divided into partitions that are distributed across the nodes in a VDC/site. The node where the partition is written then becomes the owner/authority of that partition or section of the table.

One such table is a chunk table, which keeps track of the physical location of chunk fragments and replica copies on disk. Table 1 shows a sample of a partition of the chunk table which for each chunk identifies its physical location by listing the disk within the node, the file within the disk, the offset within that file and the length of the data. Here we can see chunk ID C1 is erasure coded and chunk ID C2 is triple mirrored. More details on triple mirroring and erasure coding can be found in section 1.4 of this document.

Table 1 Sample chunk table partition

Chunk ID	Chunk location
C1	Node1:Disk1:file1:offset1:length Node2:Disk1:File1:offset1:length Node3:Disk1:File1:offset1:length Node4:Disk1:File1:offset1:length Node5:Disk1:File1:offset1:length Node6:Disk1:File1:offset1:length Node7:Disk1:File1:offset1:length Node8:Disk1:File1:offset1:length Node1:Disk2:File1:offset1:length Node2:Disk2:File1:offset1:length Node3:Disk2:File1:offset1:length Node4:Disk2:File1:offset1:length Node5:Disk2:File1:offset1:length Node6:Disk2:File1:offset1:length Node7:Disk2:File1:offset1:length Node8:Disk2:File1:offset1:length
C2	Node1:Disk3:File1:offset1:length Node2:Disk3:File1:offset1:length Node3:Disk3:File1:offset1:length

Another example is an object table which is used for object name to chunk mapping. Table 2 shows an example of a partition of an object table which details which chunk(s) and where within the chunk an object resides.

Table 2 Sample object table

Object name	Chunk ID
ImgA	C1:offset:length
FileA	C4:offset:length C6:offset:length

The mapping of table partition owners is maintained by a service, called vnest, which runs on all nodes. Table 3 shows an example of a portion of a vnest mapping table.

Table 3 Sample vnest mapping table

Table ID	Table partition owner
Table 0 P1	Node 1
Table 0 P2	Node 2

## 1.3 Failure domains

In general, failure domains relate to a concept of engineering design that takes into consideration components within a solution that have a potential for failure. ECS software is automatically aware of which disks are in the same node, and which nodes are in the same rack. In order to protect against the most failure scenarios, the



ECS software is designed to utilize this information when writing data. The basic guidelines of failure domains that ECS utilizes include the following:

- ECS never writes fragments from the same chunk to the same disk on a node
- ECS distributes fragments of a chunk equally across nodes
- ECS is rack aware, if a VDC/site contains more than one rack, assuming sufficient space, ECS uses best efforts to equally distribute fragments of a chunk across these racks

## 1.4 Advanced data-protection methods

Within ECS, when an object is created it includes writing data, custom metadata and ECS metadata. ECS metadata include journal chunks and btree chunks. Each is written to a different logical chunk that will contain ~128MB of data from one or more objects. ECS uses a combination of triple mirroring and erasure coding to protect the data within a virtual data center (VDC)/site.

- Triple mirroring ensures three copies of data are written, thus protecting against two node failures.
- Erasure coding provides enhanced data protection from disk and node failures. It utilizes the Reed Solomon erasure coding scheme which breaks up chunks into data and coding fragments which are equally distributed across nodes within a VDC/site.

Depending on the size and type of data it will be written using one of the data protection methods shown in Table 4.

Table 4 Determining what data protection level will be used for different types of data

Type of data	Data protection method used
Journal chunks	Triple mirroring
Btree chunks/custom metadata	Erasure coding with redundant data segments
Object data <128MB	Triple mirroring plus in place erasure coding
Object data >128MB	Inline erasure coding

---

**Note:** In the all flash architecture like EXF900, the btree chunks protection is triple mirroring

---

### 1.4.1 Triple mirror

The triple-mirror write method is applicable to the ECS journal chunks, of which ECS creates three replica copies. Each replica copy is written to a single disk on different nodes across failure domains. This method protects the chunk data against two-node or two-disk failures.

Figure 3 shows an example of triple mirroring whereby a logical chunk, containing 128 MB metadata, has three replica copies, each written to a different node.

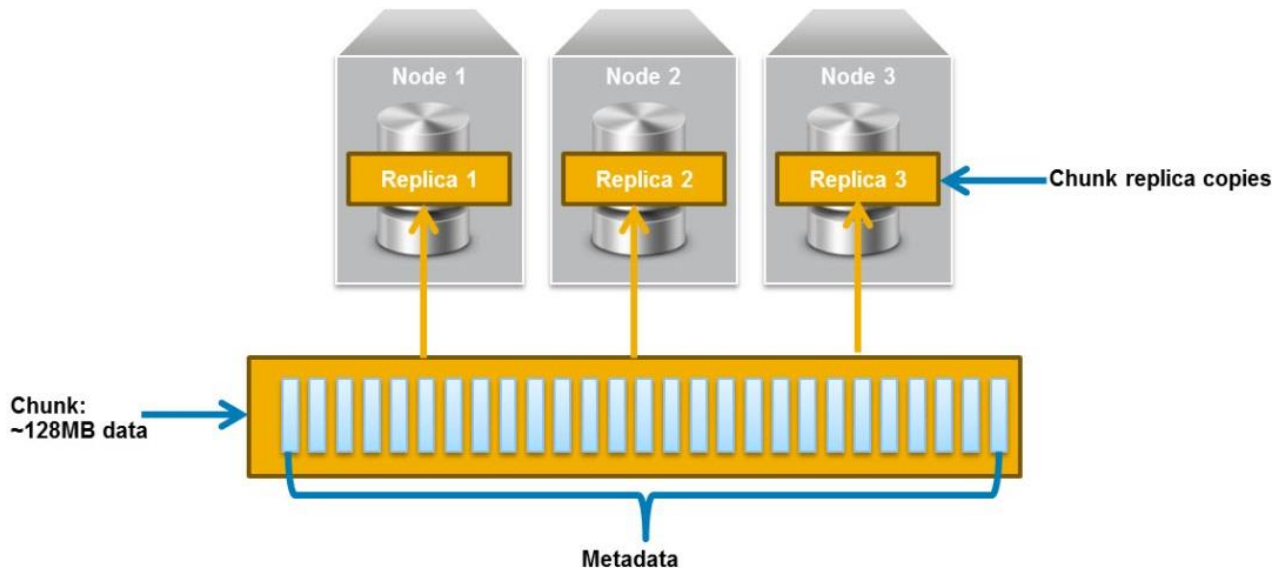


Figure 3 Triple mirroring

#### 1.4.2 Erasure coding with redundant data segments

The erasure coding with redundant data segments write method is applicable to ECS btree chunks and custom object metadata. It includes 12 data segments, 12 replicated data segments and 4 parity segments. The new btree chunk redundant data EC scheme saving metadata protection overhead.

#### 1.4.3 Triple mirror plus in place erasure coding

This write method is applicable to the data from any object that is less than 128 MB in size.

As an object is created it is written to a chunk, which ECS creates three replica copies as follows:

- One copy is written in fragments that are spread across different nodes and disks. The distribution spreads the fragments across as many failure domains as possible. The size that gets written to each disk depends on the erasure coding scheme being used.
  - If the erasure coding scheme is the default (12+4) then each disk would get a maximum of ~10.67 MB
  - If the erasure coding scheme is cold storage (10+2) each disk would get a maximum of ~ 12.8 MB
- A second replica copy of the chunk is written to a single disk on a node.
- A third replica copy of the chunk is written to a single disk on a different node.

This method provides triple mirroring and protects the chunk data against two-node or two-disk failures.

Additional objects will be written to the same chunk until it contains ~128 MB of data or after a predefined time, whichever is less. At this time, the Reed Solomon erasure coding scheme calculates coding (parity) fragments for the chunk and writes these to different disks. This ensures that all fragments within a chunk, including coding fragments will be written to different disks and distributed across failure domains.

Once the coding fragments have been written to disk the second and third replica copies will be deleted from disk. After this completes the chunk is protected by erasure coding, which provides higher levels of availability than triple mirroring.

#### 1.4.4 Inline erasure coding

This write method is applicable to the data from any object that is 128 MB or larger. Objects are broken up into 128 MB chunks. The Reed Solomon erasure coding scheme calculates coding (parity) fragments for each chunk. Each fragment will be written to different disks and distributed across failure domains. The size that gets written to each disk depends on the erasure coding scheme being used.

- If the erasure coding scheme is the default (12+4) then the fragments will be spread across 16 disks, each fragment being ~10.67 MB
- If the erasure coding scheme is cold storage (10+2) then the fragments will be spread across 12 disks, each fragment being ~12.8 MB

Any remaining portion of an object that is less than 128 MB will be written using the triple mirroring plus in place erasure coding scheme mentioned previously. As an example, if an object is 150 MB, 128 MB will be written using inline erasure coding, the remaining 22 MB will be written using triple mirrored plus in place erasure coding.

Figure 4 shows an example of how chunks are distributed across failure domains. This example has a single VDC/site that spans two racks, each rack containing four nodes.

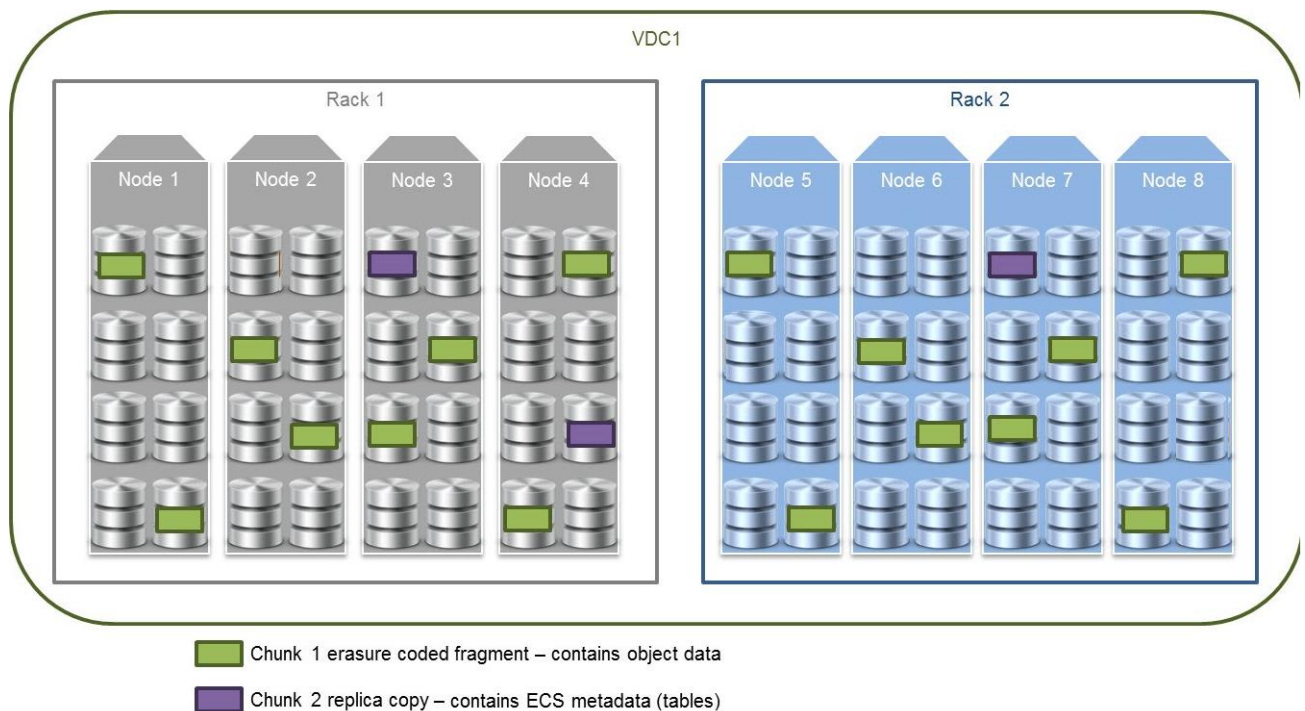


Figure 4 How chunks are distributed across failure domains

- Chunk 1 contains object data that has been erasure coded using an erasure coding of 12+4. Fragments are equally distributed across all 8 nodes, four per rack. Each node contains 2 fragments which it writes to two different disks.

- Chunk 2 contains ECS metadata (tables) and thus is triple mirrored. Each replica copy is written to a different node, each on a single disk. The copies span racks to provide the highest availability.

## 1.5 Erasure coding protection levels

Depending upon the erasure coding scheme that was selected during storage pool creation, erasure coded data is protected against the following failures.

### 1.5.1 Default erasure coding scheme (12+4):

ECS requires a minimum of four nodes to be able to conduct erasure coding using the default erasure coding scheme. Erasure coding will stop if a storage pool contains less than four nodes, meaning that the protection level will be triple mirroring. During this time, the three replica copies will remain and parity will not be calculated on any chunks. Once additional nodes are added to the storage pool and meeting the minimum supported number of nodes, erasure coding will continue on these as well as new chunks.

For each 128 MB chunk the default erasure coding scheme writes twelve data fragments and four coding fragments, each ~10.67 MB in size. It protects the chunk data against the loss of up to four fragments of a chunk which can include the following failure scenarios shown in Table 5.

Table 5 Default erasure coding protection

# Nodes in VDC	Number of chunk fragments per node	Erasure coded data protected against
5 nodes	4	<ul style="list-style-type: none"> <li>• Loss of up to four disks or</li> <li>• Loss of one node</li> </ul>
6 or 7 nodes	3	<ul style="list-style-type: none"> <li>• Loss of up to four disks or</li> <li>• Loss of one node and one disk from a second node</li> </ul>
8 or more nodes	2	<ul style="list-style-type: none"> <li>• Loss of up to four disks or</li> <li>• Loss of two nodes or</li> <li>• Loss of one node and two disks</li> </ul>
16 or more nodes	1	<ul style="list-style-type: none"> <li>• Loss of four nodes or</li> <li>• Loss of three nodes and disks from one additional node or</li> <li>• Loss of two nodes and disks from up to two different nodes or</li> <li>• Loss of one node and disks from up to three different nodes or</li> <li>• Loss of four disks from four different nodes</li> </ul>

**Note:** Table 5 reflects protection levels possible with full distribution of chunk fragments. There may be scenarios where more fragments exist on a node such as if a node has insufficient space available. In this case the protection levels may vary.

### 1.5.2 Cold storage erasure coding scheme (10+2):

ECS requires a minimum of six nodes to be able to conduct erasure coding using the cold storage erasure coding scheme. Erasure coding will stop if a storage pool contains less than six nodes, meaning that the three replica copies will remain and parity will not be calculated on a chunk. Once additional nodes are added to the storage pool, erasure coding will continue on these as well as new chunks.

For each 128 MB chunk the cold storage erasure coding scheme writes ten data fragments and two coding fragments, each ~12.8 MB in size. It protects the chunk data against the loss of up to two fragments of a chunk which can include the following failure scenarios shown in Table 6.

Table 6 Cold storage erasure coding protection

# Nodes in VDC	Number of chunk fragments per node	Erasure coded data protected against
11 or less nodes	2	<ul style="list-style-type: none"> <li>• Loss of up to two disks or</li> <li>• Loss of one node</li> </ul>
12 or more nodes	1	<ul style="list-style-type: none"> <li>• Loss of any number of disks from two different nodes or</li> <li>• Loss of two nodes</li> </ul>

---

**Note:** This table reflects protection levels possible with full distribution of chunk fragments. There may be scenarios where more fragments exist on a node such as if a node has insufficient space available. In this case the protection levels may vary.

---

## 1.6 Checksums

Another mechanism ECS uses to ensure data integrity is to store the checksum for data written. Checksums are done per write-unit, up to 2 MB. So, checksums can occur for one object fragment for large object writes or on a per-object basis for small object writes of less than 2 MB. During write operations, the checksum is calculated in memory and then written to disk. On reads, data is read along with the checksum, and then the checksum is calculated in memory from the data read and compared with the checksum stored in disk to determine data integrity. Also, the storage engine runs a consistency checker periodically in the background and does checksum verification over the entire data set.

## 1.7 Writing objects

When a write operation happens in ECS it starts with a client sending a request to a node. ECS was designed as a distributed architecture, allowing any node in a VDC/site to respond to a read or write request. A write request involves writing the object data, custom object metadata and recording the transaction in a journal log. Once this is complete the client is sent an acknowledgement.

Figure 5 and the steps aforementioned walk through a high-level overview of a write workflow.

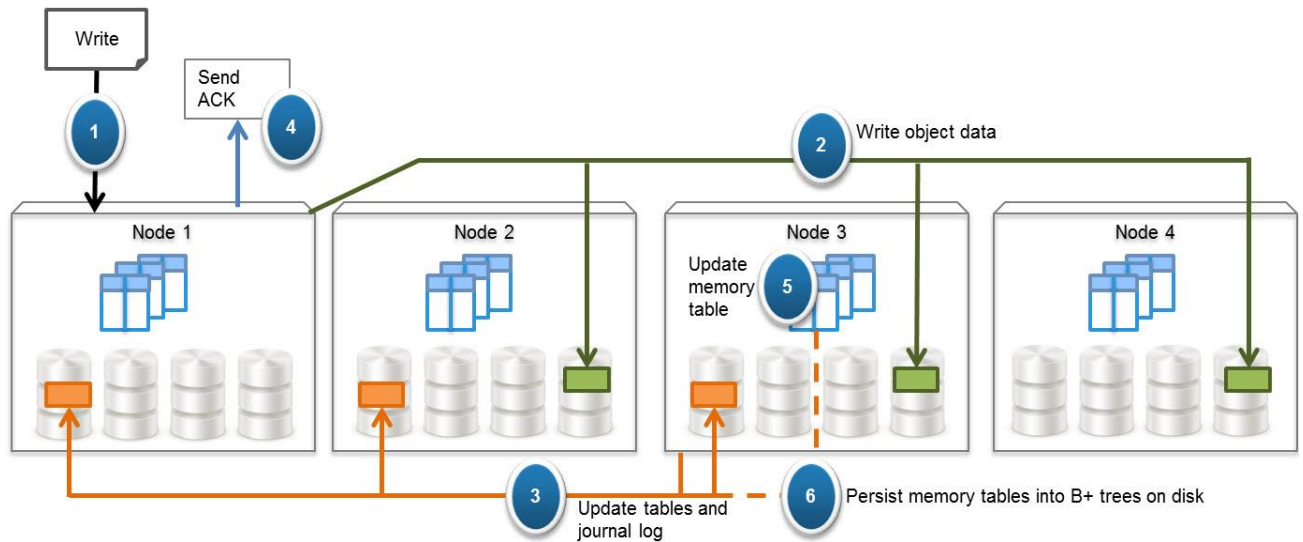


Figure 5 Object write workflow

1. A write object request is received. Any node can respond to this request but in this example Node 1 processes the request.
2. Depending on the size of the object, the data is written to one or more chunks. Each chunk is protected using advanced data protection schemes like triple mirroring and erasure coding. Prior to writing the data to disk ECS runs a checksum function and stores the result.

The data is added to a chunk; since this object is only 10 MB in size it uses the triple mirroring plus in place erasure coding scheme. This results in writes to three disks on three different nodes, in this example Node 2, Node 3, and Node 4. These three nodes send acknowledgements back to Node 1.

3. After the object data is written successfully the object's metadata is stored. In this example, Node 3 owns the partition of the object table this object belongs in. As owner Node 3 writes the object name and chunk ID to this partition of the object table's journal logs. Journal logs are triple mirrored, so Node 3 sends replica copies to three different nodes in parallel, in this example Node 1, Node 2 and Node 3.
4. Acknowledgement is sent to the client.
5. In a background process, the memory table is updated.
6. Once the table in memory becomes full or after a set period of time, the table is merged, sorted or dumped in B+ trees as chunks and a checkpoint is recorded in the journal.

## 1.8 Reading objects

ECS was designed as a distributed architecture, allowing any node in a VDC/site to respond to a read or write request. A read request involves finding the physical location of the data using table lookups from the partition record owner as well as byte offset reads, checksum validation and returning the data to the requesting client.



Figure 6 and the aforementioned steps walk through an overview of the read workflow.

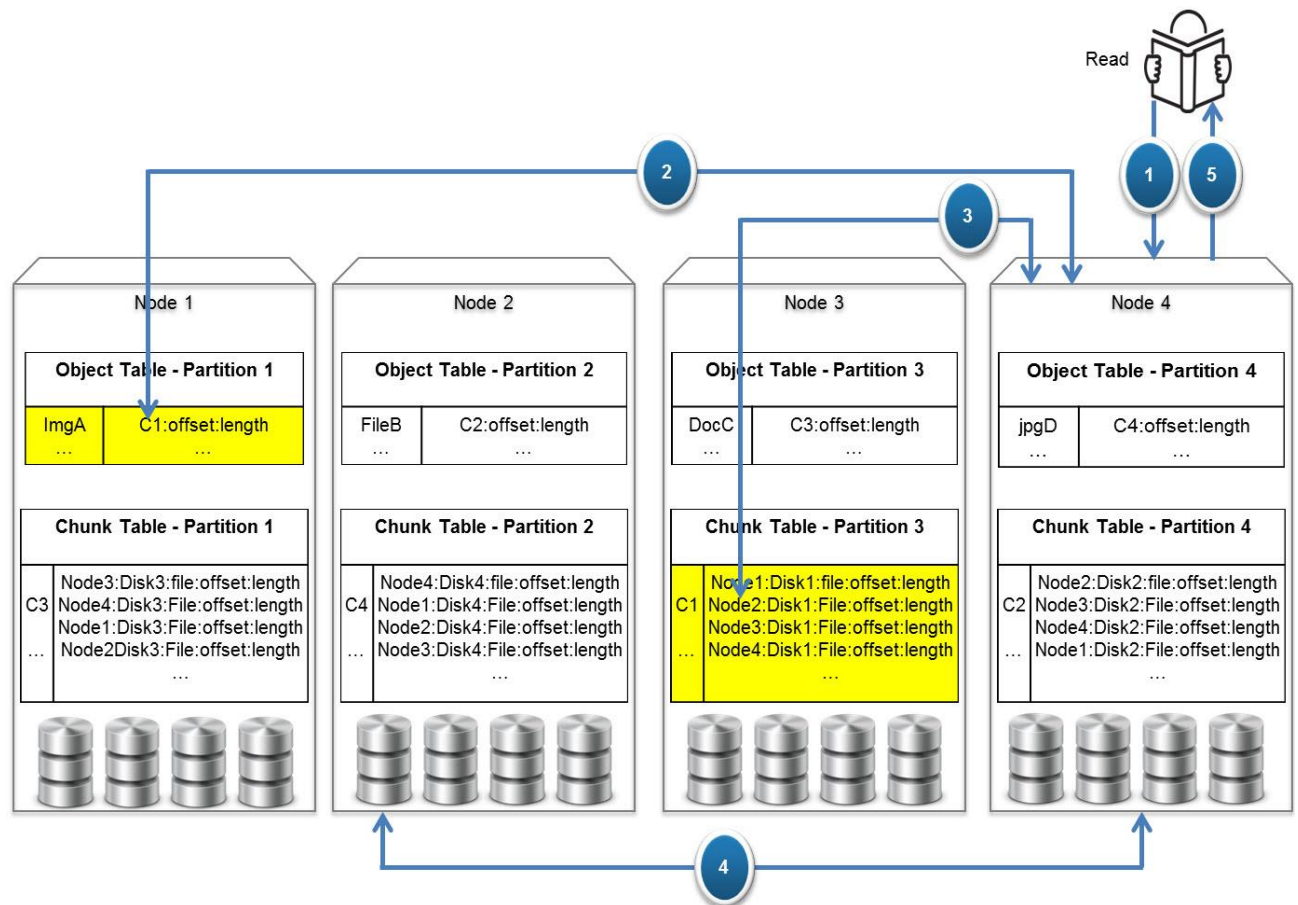


Figure 6 Object read workflow

1. A read request is received for `ImgA`. Any node can respond to this request but in this example Node 4 processes the request.
2. Node 4 requests the chunk information from Node 1 (object table partition owner for `ImgA`).
3. Knowing that `ImgA` is in `C1` at a particular offset and length, Node 4 requests the chunk's physical location from Node 3 (chunk table partition owner for `C1`).
4. Now that Node 4 knows the physical location of `ImgA` it will request that data from the node(s) that contains the data fragment(s) of that file, in this example that is Node 2 Disk 1. The node(s) will perform a byte offset read and return the data to Node 4.
5. Node 4 validates the checksum and then returns the data to the requesting client.

**Note:** In Step 4, for all flash architecture like EXF900, each node can read data from other node directly, other than the hard disk drive architecture like EX300, EX500 and EX3000 that each node can only read the data store in themselves.

## 2 Local site availability

The distributed nature of the ECS architecture provides high availability in the form of both system availability and data durability against a variety of failures. This section focuses on how availability is maintained during local site failures.

### 2.1 Disk failure

The architecture section described how ECS uses a combination of triple mirroring and erasure coding to write data in a distributed fashion to be resilient against various failure scenarios.

To ensure consistency of data checksums are validated upon reads and by a consistency checker. The consistency checker is a background process that periodically performs checksum verification over the entire data set. Read requests also run checksum verification.

If a read request is missing a fragment, due to a drive not responding, or failed checksum verification, a notification will be sent to chunk manager. Chunk manager will initiate a reconstruction of the missing fragment(s) using either the remaining erasure coded data and parity fragments or the replica copies, after which it will update the chunk information. Once the fragments have been re-created any outstanding or new read requests will use the updated chunk information to request the data fragment(s) and service the read request.

ECS nodes are constantly performing health checks on disks directly attached to them. If a disk becomes unresponsive the ECS node will notify chunk manager to stop including it in new write operations. If it remains unresponsive after a period of time (default is 60 minutes) a notification will be sent to chunk manager to re-create the data from the failed drive. The ECS node will identify which chunks have blocks on the failed drive and thus need to be recovered. It will send this information to chunk manager who will start parallel recovery of all chunk fragments stored on the failed drive. The chunk fragments are recovered onto other disks using the remaining erasure coded fragments or replica copies. As new fragments are written the associated chunk tables are updated with this information. If possible, chunk manager will also delete the fragments from the failed drive. If the disk later comes back online and it has been:

- Unresponsive for less than a set period of time (default is 90 min) the remaining recovery operations will be cancelled.
- Unresponsive for a set period of time (default is 90 min) or is reported as failed by hardware manager, ECS will remove the drive. Once a drive is removed the remaining recovery operations will continue until complete. When recovery is complete chunk manager then removes all references to this failed disk in the chunk table.

If this drive comes online after it has been removed it will be added as a new drive and chunk manager will include it in new write operations.



## 2.2 ECS node failure

ECS is constantly performing health checks on nodes. In order to maintain system availability, the ECS distributed architecture allows any node to accept client requests. If one node is down a client can be redirected either manually or automatically (ex. using DNS or a load balancer) to another node that can service the request.

In order not to trigger reconstruction operations for spurious events a full reconstruction operation isn't triggered unless a node fails a set number of sequential health checks, the default is 60 minutes. If an IO request comes in for a node that is not responding but before a full reconstruction is triggered:

- Any request for a partition table hosted on a node that isn't responding, will trigger the requested partition tables' ownership to be redistributed across the remaining nodes in the site. Once this completes the request will complete successfully.
- Any IO requests for data that exists on disks from the unresponsive node will be reconstructed using either the remaining erasure coded data and parity fragments or the replica copies, after which it will update the chunk information. Once the fragments have been re-created any outstanding or new read requests will use the updated chunk information to request the data fragment(s) and service the read request.

After a node fails a set number of sequential health checks, default is 60 minutes, a node is deemed to be down. This automatically triggers a re-creation operation of the partition tables and the chunk fragments on the disks owned by the failed node.

As part of the re-creation operation a notification will be sent to chunk manager which starts a parallel recovery of all chunk fragments stored on the failed nodes' disks. This can include chunks containing object data, custom client provided metadata and ECS metadata. If the failed node comes back online an updated status will be sent to chunk manager and any uncompleted recovery operations will be cancelled. More details about chunk fragment recovery are covered in the disk failure section above.

Besides hardware monitoring ECS also monitors all services and data tables on each node.

- If there is a table failure but the node is still up it will automatically try to reinitialize the table on the same node.
- If it detects a service failure it will first try to restart the service.

If that fails, it will redistribute ownership of the tables owned by the down node or service across all remaining nodes in the VDC/site. The ownership changes involve updating the vnest information and re-creating the memory tables owned by the failed node. The vnest information will be updated on the remaining nodes with new partition table owner information.

The memory tables from the failed node will be re-created by replaying journal entries written after the latest successful journal checkpoint.

## 2.2.1 Multiple-node failures

There are scenarios when multiple nodes can fail within a site. Multiple nodes can fail either one by one or concurrently.

- One-by-one failure:** When nodes fail one by one it means one node fails, all recovery operations complete and then a second node fails. This can occur multiple times and is analogous to a VDC going from something like 4 sites → 3 sites → 2 sites → 1 site. This requires that the remaining nodes have sufficient space to complete recovery operations.
- Concurrent failure:** When nodes fail concurrently it means nodes fail almost at the same time, or a node fails before recovery from a previous failed node completes.

The impact of the failure depends on which nodes go down. Table 7 and Table 8 describe the best-case scenario for failure tolerances of a single site based on erasure coding scheme and number of nodes in a VDC.




















































<b>Legend</b>	 Erasure coding runs	 Reads successful	 Writes successful
	 Erasure coding stops	 Subset of reads fail	 Subset of writes fail
	 Erasure coding stops	 Reads stop	 Writes stop

Table 7 Best-case scenario of multiple node failures within a site based on the default 12+4 erasure coding scheme

# Nodes in VDC at creation	Total # of failed nodes since VDC creation	Status after concurrent failures	Status after most recent one by one failure	Current VDC state after previous one by one failures
5 nodes	1	  	  	5 node VDC with 1 node failing
	2	  	  	VDC previously went from 5 → 4 nodes, now 1 additional node is failing
	3 - 4	  	  	VDC previously went from 5 → 4 → 3 or 5 → 4 → 3 → 2 nodes, now 1 additional node is failing
6 nodes	1	  	  	6 node VDC with 1 node failing
	2	  	  	VDC previously went from 6 → 5 nodes, now 1 additional node is failing
	3	  	  	VDC previously went from 6 → 5 → 4 nodes, now 1 additional node is failing
	4 - 5	  	  	VDC previously went from 6 → 5 → 4 → 3 or 6 → 5 → 4 → 3 → 2 nodes, now 1 additional node is failing









































































# Nodes in VDC at creation	Total # of failed nodes since VDC creation	Status after concurrent failures	Status after most recent one by one failure	Current VDC state after previous one by one failures
8 nodes	1 - 2	EC  	EC  	8 node VDC or VDC that went from 8 → 7 nodes, now 1 node failing
	3 - 4	EC  	EC  	VDC previously went from 8 → 7 → 6 or 8 → 7 → 6 → 5 nodes, now 1 additional node is failing
	5	  	  	VDC previously went from 8 → 7 → 6 → 5 → 4 nodes, now 1 additional node is failing
	6 - 7	  	  	VDC previously went from 8 → 7 → 6 → 5 → 4 → 3 nodes or 8 → 7 → 6 → 5 → 4 → 3 → 2 nodes, now 1 additional node is failing

Table 8 Best-case scenario of multiple node failures within a site based on the cold storage 10+2 erasure coding scheme

# Nodes in VDC at creation	Total # of failed nodes since VDC creation	Status after concurrent failures	Status after most recent one by one failure	Current VDC state after previous one by one failures
6 nodes	1	  	  	6 node VDC with 1 node failing
	2	  	  	VDC previously went from 6 → 5 nodes, now 1 additional node is failing
	3	  	  	VDC previously went from 6 → 5 → 4 nodes, now 1 additional node is failing
	4 - 5	  	  	VDC previously went from 6 → 5 → 4 → 3 or 6 → 5 → 4 → 3 → 2 nodes, now 1 additional node is failing
8 nodes	1	EC  	EC  	8 node VDC with 1 node failing
	2	EC  	EC  	VDC previously went from 8 → 7 nodes, now 1 additional node is failing
	3 - 5	  	  	VDC previously went from 8 → 7 → 6 or 8 → 7 → 6 → 5 or 8 → 7 → 6 → 5 → 4 nodes, now 1 additional node is failing

# Nodes in VDC at creation	Total # of failed nodes since VDC creation	Status after concurrent failures	Status after most recent one by one failure	Current VDC state after previous one by one failures
	6 - 7			VDC previously went from 8 → 7 → 6 → 5 → 4 → 3 nodes or 8 → 7 → 6 → 5 → 4 → 3 → 2 nodes, now 1 additional node is failing
12 nodes	1 - 2	EC  	EC  	12 node VDC or 12 node VDC that went from 12 → 11 nodes, now 1 additional node is failing
	3 - 6	EC  	EC  	VDC previously went from 12 → 11 → 10 or 12 → 11 → 10 → 9 or 12 → 11 → 10 → 9 → 8 or 12 → 11 → 10 → 9 → 8 → 7 nodes, now 1 additional node is failing
	7 - 9			VDC previously went from 12 → 11 → 10 → 9 → 8 → 7 → 6 or 12 → 11 → 10 → 9 → 8 → 7 → 6 → 5 or 12 → 11 → 10 → 9 → 8 → 7 → 6 → 5 → 4 nodes, now 1 additional node is failing
	10 - 11			VDC previously went from 12 → 11 → 10 → 9 → 8 → 7 → 6 → 5 → 4 → 3 or 12 → 11 → 10 → 9 → 8 → 7 → 6 → 5 → 4 → 3 → 2 nodes, now 1 additional node is failing

The basic rules for determining what operations fail in a single site with multiple node failures include:

- If you have three or more concurrent node failures some reads and writes will fail due to the potential loss of all three replica copies of the associated triple mirrored metadata chunks.
- Writes require a minimum of three nodes.
- Erasure coding will stop and erasure coded chunks will be converted to triple mirror protection if the number of nodes is less than the minimum required for each erasure coding scheme. Since the default erasure coding scheme, 12+4 requires 4 nodes, erasure coding will stop if there are less than 4 nodes. For cold storage erasure coding, 10+2, erasure coding will stop if there are less than 6 nodes.
- If the node count goes below the minimum required for the erasure coding scheme, erasure coded chunks will be converted to triple mirror protection. As an example, in a VDC with default erasure coding and 4 nodes, after a node failure the following would happen:
  - Node failure causes 4 fragments to be lost.
  - Missing fragments are rebuilt.
  - Chunk creates 3 replica copies, one on each node.
  - EC copy is deleted.
- One-by-one failures act like single node failures. As an example, if you lose two nodes one-by-one, each failure only consists of recovering data from the single node outage.

As an example, with 6 nodes and default erasure coding:

- First failure: Each of the six nodes has up to three fragments (16 fragments / 6 nodes). The missing three fragments are re-created on the remaining nodes. After recovery completes the VDC ends up with 5 nodes.
- Second failure: Each of the five nodes has up to 4 fragments (16 fragments / 5 nodes). The missing four fragments are re-created on the remaining nodes. After recovery completes the VDC ends up with 4 nodes.
- Third failure: Each of the four nodes has up to 4 fragments (16 fragments / 4 nodes). The missing four fragments are re-created on the remaining nodes. After recovery completes the VDC ends up with 3 nodes and since this is below the minimum for erasure coding, the erasure coded chunk is replaced by three replica copies distributed across the remaining nodes.
- Forth failure: Each of the three nodes has one replica copy. The missing replica copy is re-created on one of the remaining nodes. After recovery completes the VDC ends up with 2 nodes.
- Fifth failure: There are three replica copies, two on one node, and one on the other node. The missing replica copy(s) are re-created on the remaining node. After recovery completes the VDC ends up with 1 node.

### 3 Multi-site design overview

Beyond system availability and data durability designed within a single site, ECS also includes protection against a complete site-wide failure. This is accomplished in a multi-site deployment by federating multiple VDCs/sites together and configuring geo-replication.

Federating sites involves providing replication and management endpoints for communication between sites. Once sites are federated, they can be managed as a single infrastructure.

Replication group policies determine how data is protected and where it can be accessed from. ECS supports both geo active replication and geo passive replication. Geo active replication provides active-active access to data, allowing it to be read and written from any site within its defined replication group.

When replicating all flash series like EXF900 across sites, one should consider the potential performance impacts over the WAN. Large ingest may put high load on the link causing saturation or delayed RPO, plus a user/application may experience higher latency times on remote reads and writes as compared to local requests. The other should consider the partial garbage collection failed, large ingest from both local and replicate site may cause system reach 90% soon which will make system stop writing data and reclaiming data.

---

**Note:** partial garbage collection - When a chunk is 2/3 garbage, reclaim the chunk by merging the valid parts of with other partially filled chunks to a new chunk and then reclaim space.

---

The replication can also be configured as geo-passive which designates two to four source sites and one or two sites to be used only as a replication target. The replication targets are used for recovery purposes only. Replication targets block direct client access for create/update/delete operations.

The benefits of geo-passive replication include:

- It can optimize storage efficiency by increasing the chances of XOR operations running by ensuring writes from both source sites go to the same replication target.
- It allows the administrator to control where the replication copy of data exists, such as in a backup to cloud scenario.

ECS provides geo-replication configuration options at the bucket level, allowing the administrator to configure different levels of replication for different buckets.

Figure 7 shows an example of how an administrator could configure the replication of three buckets:

- Bucket A: Engineering test/dev data — do not replicate, keep locally only
- Bucket B: European sales data — replicate between sites within Europe only
- Bucket C: Companywide training data — replicate to all sites within the company

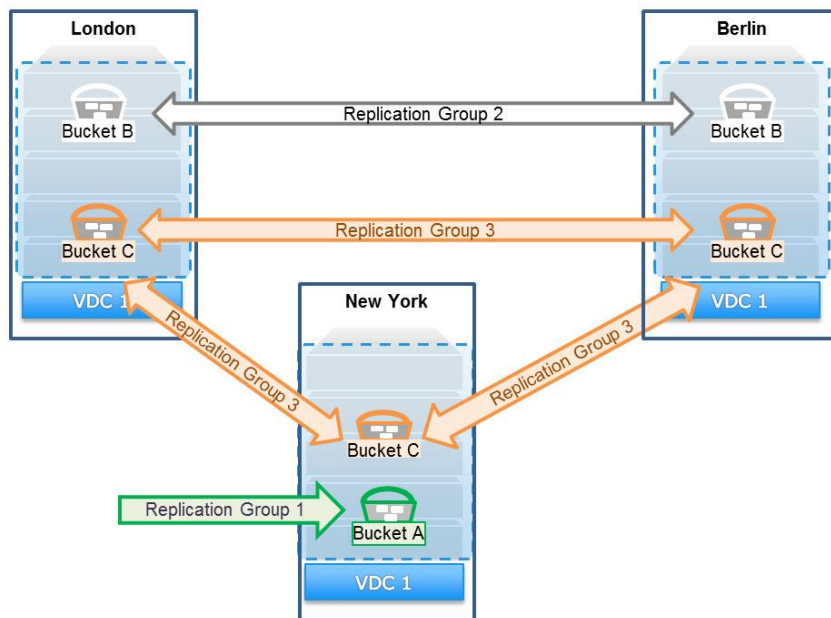


Figure 7 Sample showing how different buckets can use different replication policies

As a best practice, it is recommended to configure replication groups for specific replication paths. As an example, in Figure 7 there is replication group that replicates data between London and Berlin. This should be used for all buckets needing the replication between only London and Berlin.

Data that is geo-replicated is protected by storing a primary copy of the data at the local site and a secondary copy of data at one or more remote sites. The number of copies and the amount of space the secondary copy takes is determined based on the number of sites configured in the replication group, how data is written across sites as well as whether **replicate to all sites** is enabled or not.

Each site is responsible for local data protection meaning that both the local and secondary copies will individually protect the data using erasure coding and/or triple mirroring. The erasure coding schemes at each site does not have to be the same, meaning one site can use the default erasure coding scheme of 12+4 and the other site can use the cold storage erasure coding scheme of 10+2.

Replicated data is encrypted (AES256) and compressed before being sent to the other site via HTTP.

In order to maintain consistency between sites there needs to be an authority that is responsible for maintaining the latest version of metadata. The authority is defined at the site level and determines ownership of namespaces, buckets and objects. Ownership information is first stored on the owner site but is also replicated to the other sites as part of the ECS metadata.

- Authoritative version: The authoritative version is always the owner and that is used to provide strong consistency.
- Replicated version: The replicated version(s) may not be the most recent but are used during failure operations including:
  - If **access during outage** is enabled (eventual consistency).
  - And during permanent site failover operations.

There are authoritative versions of the bucket and object owners.

**Namespace owner:**

- The site that creates the namespace is the namespace owner.
- It is responsible for maintaining the authoritative version of the bucket list.

**Bucket owner:**

- The site that creates the bucket is the bucket owner.
- It is responsible for maintaining the authoritative version of:
  - The bucket listing which includes the most up-to-date version of which objects are in a bucket.
  - The listing of object ownerships for objects within its bucket

**Object owner:**

- Initially the site that first created the object is the object owner. This can change; see the “access during outage” section for details.
- It is responsible for maintaining the authoritative version of the object metadata.

## 3.1 Chunk manager tables

Chunk locations are retained in the chunk manager table which is stored independently on all sites. The location to which a chunk is originally created is known as the primary site; the location that it is replicated to is known as the secondary site.

When a chunk is created, the primary and secondary sites are determined, and the primary site broadcasts the chunks’ site information to the other nodes in the replication group.

In addition, since each site maintains its own chunk manager table it will also include information about the property type for each chunk. Property types include:

- **Local:** on the site in which the chunk was created
- **Copy:** on the site in which the chunk was replicated to
- **Remote:** on the site(s) in which neither the chunk nor its replica is stored locally
- **Parity:** on chunks that contain the result of an XOR operation of other chunks (see XOR section below for more details)
- **Encoded:** on chunks whose data has been replaced locally with XOR data (see XOR section below for more details).

Table 9 through Table 11 show sample portions of chunk manager table listings from each of three sites.

Table 9 Sample chunk manager table from site1

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 2	Local
C2	Site 2	Site 3	Remote
C3	Site 1	Site 3	Local
C4	Site 2	Site 1	Copy



Table 10 Sample chunk manager table from site 2

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 2	Copy
C2	Site 2	Site 3	Local
C3	Site 1	Site 3	Remote
C4	Site 2	Site 1	Local

Table 11 Sample chunk manager table from site 3

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 2	Remote
C2	Site 2	Site 3	Copy
C3	Site 1	Site 3	Copy
C4	Site 2	Site 1	Remote

## 3.2 XOR encoding

In order to maximize storage efficiency of data that is configured with a replication group containing three or more sites, ECS utilizes XOR encoding. As the number of sites in a replication group increases, the ECS algorithm is more efficient in reducing the overhead.

XOR encoding is performed at each site. It scans its chunk manager table and when it finds COPY type chunks that are from each of the other sites in its replication group it can perform XOR encoding on these chunks. For example, in Table 12 it shows site 3 of a three-site configuration that has both chunks **C2** and **C3** that are COPY type each with a different primary site. This allows site 3 to XOR them together and store the result. The result will be a new chunk, **C5** that is an XOR of **C2** and **C3** (mathematically  $C2 \oplus C3$ ) and it will have a type listed as **Parity**, without a secondary site. The chunk IDs of parity chunks are not broadcasted to other sites.

Table 12 shows an example of a chunk manager table on site 3 while it is performing XOR of chunks **C2** and **C3** together into chunk **C5**.

Table 12 Site 3 chunk manager table during XOR operation

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 2	Remote
C2	Site 2	Site 3	Copy
C3	Site 1	Site 3	Copy
C4	Site 2	Site 1	Remote
C5	Site 3		Parity (C2 & C3)

After XOR is complete the data copy for **C2** and **C3** are deleted thus freeing up space on disk, and the chunk manager table type for these chunks changes to type Encoded. The XOR operation is purely a secondary site operation, the primary site is not aware that its chunks were encoded. After XOR encoding completes and the data copy for **C2** and **C3** is deleted, site 3's chunk manager table would be listed as shown in Table 13.

Table 13 Site 3 chunk manager table after completing XOR encoding of C2 and C3

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 2	Remote
C2	Site 2	Site 3	Encoded
C3	Site 1	Site 3	Encoded
C4	Site 2	Site 1	Remote
C5	Site 3		Parity (C2 & C3)

Requests for data in an encoded chunk will be serviced by the site containing the primary copy. For more details on storage efficiency refer to the [ECS Overview and Architecture White Paper](#).

### 3.3 Replicate to all sites

Replicate to all sites is a replication group option that is used when you have three or more sites and you want all chunks to be replicated to all VDCs/sites configured in the replication group. This also prevents XOR operations from running. When the **replicate to all sites** option is:

- **Enabled:** the number of copies of data written is equal to the number of sites in the replication group. As an example, if you have four sites in the replication group you will have a primary copy plus three secondary copies; one in each site.
- **Disabled:** the number of copies of data written is two. You have the primary copy plus one replicated copy in a remote site, regardless of the total number of sites.

---

**Note:** Replicate to all sites cannot be enabled on replication group configured as geo-passive.

---

This setting has no impact on replication groups that contain only two sites since in essence this already replicates all data to both sites. The administrator can choose which buckets utilize this replication group.

Enabling replicate to all sites has the following impact:

- Can improve read performance because after replication completes, subsequent reads are serviced locally.
- Removes performance impact caused by XOR decoding
- Increases data durability.
- Decreases the storage utilization efficiency.
- Increases WAN utilization for geo replication; increase is proportional to the number of VDCs/sites in the replication group.
- Decreases WAN utilization for reads of replicated data.

For these reasons, enabling this option is only recommended on specific buckets in environments that meet the following criteria:

- Whose workload is read intensive for the same data from geographically dispersed sites.
- Whose infrastructure has sufficient WAN bandwidth between sites in the replication group to allow for increased geo replication traffic.
- Who value read performance over the cost of storage utilization efficiency.

### 3.4 Write data flow in geo-replicated environment

Chunks contain 128 MB of data consisting of one or more objects from bucket(s) that share the same replication group settings. Replication is performed asynchronously and initiated at the chunk level by the chunk partition owner. If the chunk is configured with geo-replication data will be added to a replication queue as it is written to the primary site's chunk, it does not wait for the chunk to be sealed. There are worker I/O threads continuously processing the queue.

The write operation first occurs locally, including adding data protection, and then it is replicated and protected at the remote site. Figure 8 shows an example of the write process for a 128 MB object to a geo-replicated bucket.

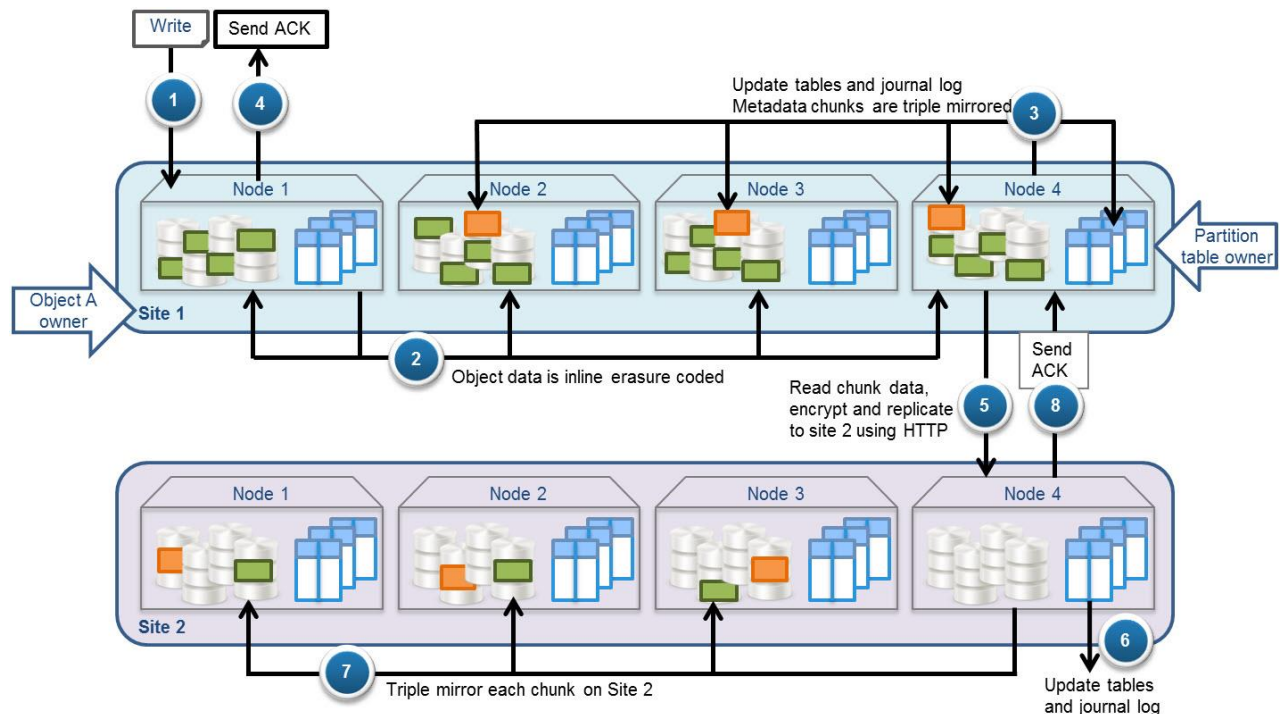


Figure 8 Write data workflow for a 128 MB object to a geo-replicated bucket

1. Write request for Object A is sent to a node, in this example site 1 Node 1. Site 1 becomes the Object A owner.
2. Data is inline erasure coded and written to a chunk in Site 1.
3. Table partition owners, in this example Node 4, update the appropriate tables (ex. chunk, object and bucket listing tables) and write the transactions into its journal logs. This metadata is written to a metadata chunk which is triple mirrored in Site 1.
4. Acknowledgement of the successful write is sent to the client.
5. For each chunk, the chunk partition table owner, in this example Node 4:
  - Reads chunk data, encrypts and replicates to Site 2 using HTTP.
  - Writes the data to Site 2, Node 1.
  - Triple mirrors each chunk on Site 2.
  - Updates tables and journal log.
  - Sends an ACK.

- a. adds the data inside the chunk to the replication queue after it is written locally, it does not wait for the chunk to be sealed.
  - b. reads the data fragments of the chunk (parity fragments are only read if required to re-create a missing data fragment).
  - c. encrypts and replicates the data to site 2 via HTTP.
6. Table partition owners for the replicated chunks, in this example site 2 Node 4, update the appropriate tables and write the transactions into its journal logs which are triple mirrored.
  7. Each replicated chunk is initially written on the second site using triple mirroring.
  8. Acknowledgement is sent back to the primary site chunk partition table owner.

---

**Note:** Data written to the replicated site will be erasure coded after a delay, allowing time for other processes, such as XOR operations, to complete first.

---

### 3.5 Read data flow in geo-replicated environment

Since ECS replicates data asynchronously to multiple VDCs within a replication group, it requires a method to ensure consistency of data across sites / VDCs. ECS ensures strong consistency by fetching the latest copy of the metadata from the site that is the object owner. If the requesting site contains a copy (chunk type = local or copy) of the object it will use that to service the read request, otherwise it will fetch the data from the object owner. An example showing the read data flow is seen in Figure 9 which depicts a read request for Object A from a site other than the object owner site.

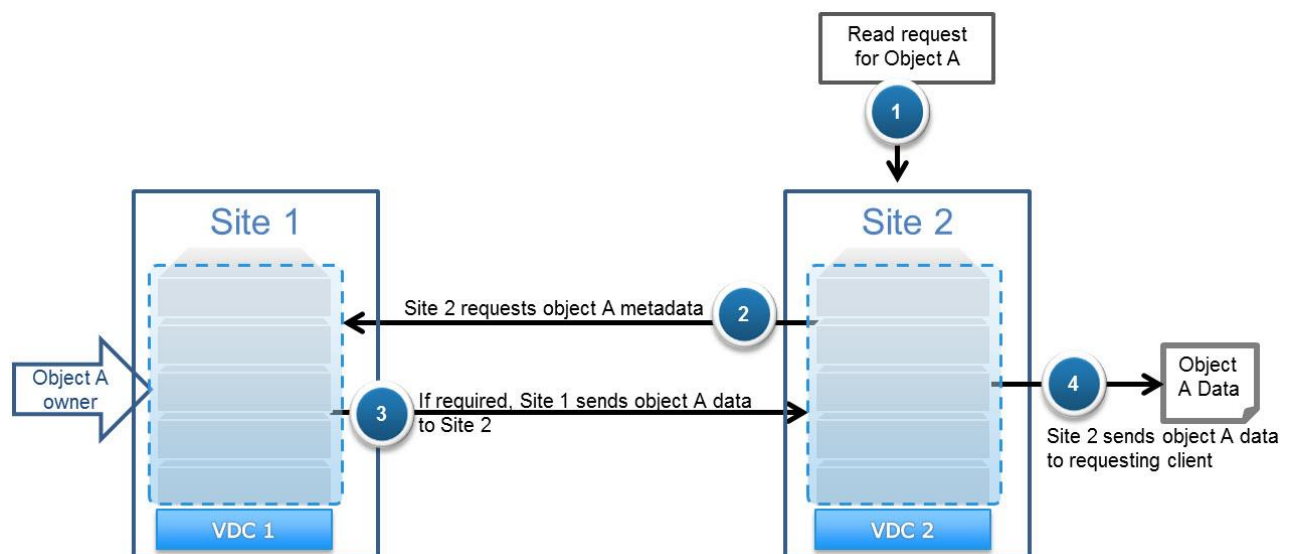


Figure 9 Read data workflow for an geo-replicated object owned by another site

In this example, the flow of the read operation is shown as follows:

1. Site 2 receives a read request for Object A which is owned by Site 1.
2. Site 2 contacts the bucket and object owner, in this example Site 1, to get the latest version of metadata.

Object ownership:

- If **access during outage** is disabled it will check its local information to determine if it is the object owner. If it isn't, it will contact the bucket owner to determine who the object owner is.
  - If **access during outage** is enabled on the bucket the requesting site will check with the bucket owner to see who the current object owner is.
3. If Site 2 does not contain a copy of the object (chunk type = local or copy) then Site 1 will send the Object A data to Site 2.
  4. Site 2 sends the Object A data to requesting client.

### 3.6 Update data flow in geo-replicated environment

ECS is architected to allow active-active updating of data from nodes within the associated buckets replication group. It can achieve this by requiring non-object owner sites to synchronously send information about an object update to the primary owner site and wait for acknowledgement before they can send the acknowledgement back to the client. The data related to the updated object is replicated as part of the normal asynchronous chunk replication activities. If the data is not yet replicated to the owning site and it receives a read request for the data, then it will request the data from the remote site. Figure 10 shows an example depicting an update request for Object A from a site other than the object owner site.

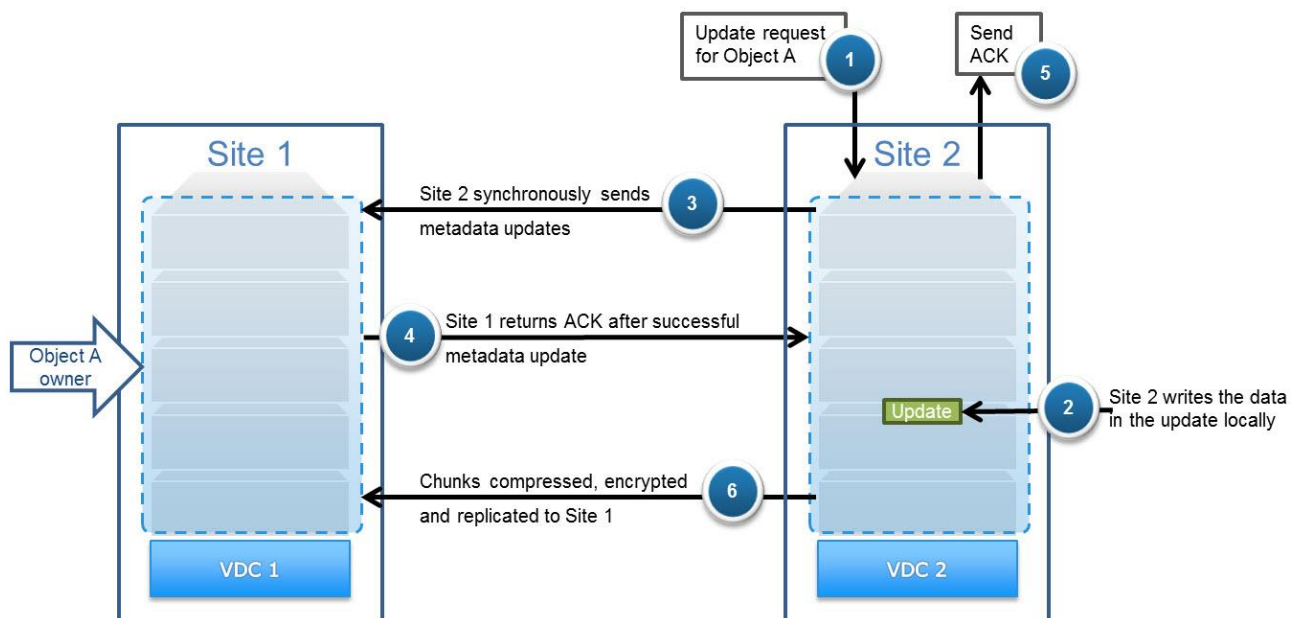


Figure 10 Update object workflow for a geo-replicated object owned by another site

In this example, the flow of the update operation is shown as follows:

1. Site 2 receives an update request for Object A which is owned by Site 1.
2. Site 2 writes the data in the update locally.
3. Site 2 sends metadata updates synchronously to the object owner, in this example Site 1.

---

**Note:** If **access during outage** is enabled on the bucket or Site 2 is not the object owner, it contacts the bucket owner first to determine who the current object owner is.

---

4. Site 1 sends acknowledgement to Site 2 that the metadata update was successful.

5. Site 2 sends the acknowledgement to requesting client that the update was successful.
6. The chunks are added to the replication queue, encrypted and asynchronously replicated to site 1 as usual.

Under normal conditions, the object owner does not change after an update is performed, regardless of which site the update originated from. In this example, the object owner remains site 1 even after the successful update that originated from site 2. The only exception is during a temporary site outage if **access during outage** is enabled; see section 4.1.2 for more details.

## 4 Multi-site availability

ECS provides strong consistency, requiring I/O requests to check with the owner before responding. Because of this, if a site is inaccessible access to some buckets and objects may be temporarily disrupted.

Site outages can occur for different durations and for many different reasons such as:

- Temporarily such as from the loss of network connectivity between federated sites or from the failure of an entire site such as a building power outage.
- Permanently such as from a natural disaster.

To detect temporary site outages federated sites establish a heartbeat between sites. If the heartbeat is lost between sites for a sustained period of time, the default is 15 minutes:

- In a two-site configuration, each will mark the other as failed.
- In a three-or-more-site configuration, a site will only be marked as failed if both:
  - A majority of sites lose heartbeat for the sustained period of time to the same ECS site
  - And if all the remaining sites are currently marked as online

As an example, in a three-site configuration, if sites 2 and 3 both lose network connectivity to site 1 for a sustained period of time, ECS will mark site 1 as temporarily failed.

When a federated site is failed, system availability can be maintained by directing access to other federated systems. During the site-wide failure the geo-replicated data owned by the unavailable site will become temporarily unavailable. The duration that the data remains unavailable is determined by the following:

- Whether or not **access during outage** is enabled
- How long the temporary site outage remains
- The time it takes for a permanent site failover to complete recovery operations

The site failure can either be temporary or permanent. A temporary site outage means the site can be brought back online and is typically caused by power outages or loss of networking between sites. A permanent site outage is when the entire system is unrecoverable, such as from a lab fire. Only an administrator can determine if a site outage is permanent and instigate recovery operations.

### 4.1 Temporary site outage (TSO)

Temporary site outages occur when a site is temporarily inaccessible to other sites in a replication group. ECS allows administrators two configuration options that affect how objects can be accessed during a temporary site outage.

- Disable the **access during outage** (ADO) option which retains strong consistency by:
  - Continuing to allow access to data owned by sites that are accessible.
  - Preventing access to data owned by an inaccessible site.
- Enable the **access during outage** option which allows read and optionally write access to all geo-replicated data including that which is owned by the site marked as failed. During a TSO with **access during outage** enabled, the data in the bucket temporarily switches to eventual consistency; once all sites are back online it will revert back to strong consistency.



The default is for **access during outage** to be disabled.

The **access during outage** option can be set at the bucket level; meaning you can enable this option for some buckets and not for others. This bucket option can be changed at any time so long as all sites are online, it cannot be changed during a site failure.

During a temporary site outage:

- Buckets, namespaces, object users, authentication providers, replication groups and NFS user and group mappings cannot be created, deleted or updated from any site (replication groups can be removed from a VDC during a permanent site failover).
- You cannot list buckets for a namespace when the namespace owner site is not reachable.
- File systems within HDFS/NFS buckets that are owned by the unavailable site are read-only.
- When you copy an object from a bucket owned by the unavailable site, the copy is a full copy of the source object. This means that the same object's data is stored more than once. Under normal non-TSO circumstances, the object copy consists of the data indices of the object, not a full duplicate of the object's data.
- OpenStack Swift users cannot log in to OpenStack during a TSO because ECS cannot authenticate Swift users during the TSO. After the TSO, Swift users must re-authenticate.

### 4.1.1 Default TSO behavior

Since ECS provides strong consistency, IO requests require checking with the owner before responding. If a site is inaccessible to other sites within a replication group, some access to buckets and objects may be disrupted.

Table 14 shows which access is required for an operation to succeed.

Table 14 Access requirements

Operation	Requirements for success
Create object	Requires the bucket owner to be accessible
List objects	Requires the bucket owner and all objects in the bucket be accessible by the requesting node
Read object Update object	Requires the requestor to be: <ul style="list-style-type: none"> <li>• The object owner and bucket owner (bucket ownership is only required if <b>access during outage</b> is enabled on the bucket containing the object</li> <li>• Or that both the object owner and bucket owner are accessible by the requesting node</li> </ul>

- A create object operation includes updating the bucket listing with the new object name. This requires access to the bucket owner and as such will fail if the requesting site doesn't have access to the bucket owner.
- Listing objects in a bucket requires both listing information from the bucket owner and head information for each object in the bucket. Therefore, the following bucket listing requests will fail if **access during outage** is disabled:
  - Requests to list buckets owned by a site that is not accessible to the requestor.
  - Bucket that contain objects owned by a site that is not accessible to the requestor.



- Read object requires first reading the object metadata from the object owner.
  - If the requesting site is the object owner and **access during outage** is disabled, then the request will succeed.
  - If the requesting site is the object and bucket owner, then the request will succeed.
  - If the object owner is not local, the site needs to check with the bucket owner to find the object owner. If either the object owner or bucket owner sites are unavailable to the requestor the read operation will fail.
  
- Updates to objects require successfully updating the object metadata on the object owner.
  - If the requesting site is the object owner and **access during outage** is disabled, then the request will succeed.
  - If the requesting site is the object and bucket owner, then the request will succeed.
  - If the object owner is not local the site needs to check with the bucket owner to find the object owner. If either the object owner or bucket owner sites are unavailable to the requestor the read operation will fail.

Looking at a three-site example, the bucket and object layout are shown in Figure 11.

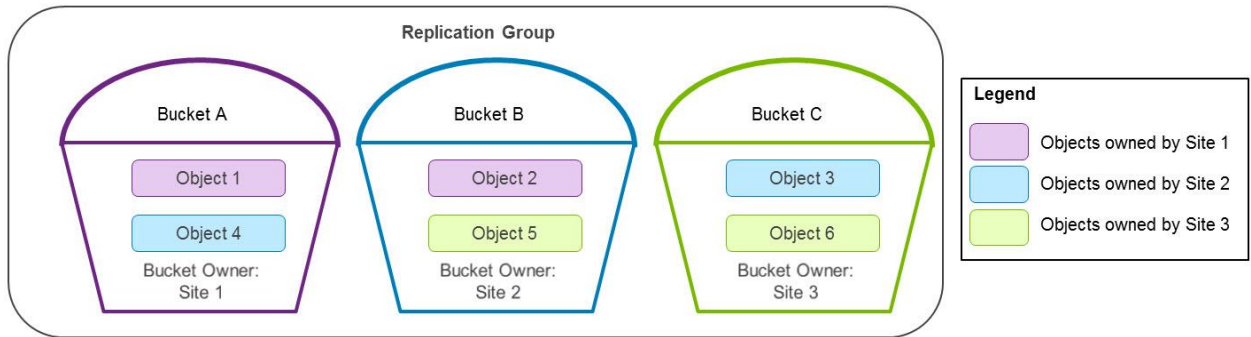


Figure 11 Bucket and object ownership example

Table 15 lists which operations will succeed or fail in the three-site configuration example shown in Figure 11 if site 1 is inaccessible to the other sites in the replication group. To simplify interpretation of the table, the inaccessible site is listed as failed and the other two are listed as online.

Table 15 Operations that succeed or fail if Site 1 is inaccessible to other sites

Operation	Bucket / object	Request sent to		
		Site 1 (failed)	Site 2 (online)	Site 3 (online)
Create objects in	Bucket A	Success Bucket owned locally	Fail Cannot access bucket owner	Fail Cannot access bucket owner
	Bucket B	Fail Cannot access bucket owner	Success Bucket owned locally	Success Bucket owned by online site
	Bucket C	Fail Cannot access bucket owner	Success Bucket owned by online site	Success Bucket owned locally

Operation	Bucket / object	Request sent to		
		Site 1 (failed)	Site 2 (online)	Site 3 (online)
List objects in	Bucket A	Fail Although bucket is owned locally it contains an object that is owned by a site it cannot access	Fail Cannot access bucket owner	Fail Cannot access bucket owner
	Bucket B	Fail Cannot access bucket owner	Fail Although bucket is owned locally it contains an object that is owned by the failed site	Fail Although bucket owner is online the bucket contains an object that is owned by the failed site
	Bucket C	Fail Cannot access bucket owner	Success Bucket owner is online site and all objects are from online sites	Success Bucket owned locally and all objects are from online sites
Read or update object	Object 1	Success Object owned locally	Fail Cannot access object owner	Fail Cannot access object owner
	Object 2	Success Object owned locally	Fail Cannot access object owner	Fail Cannot access object owner
	Object 3	Fail Cannot access object owner	Success Object owned locally	Success Object is not locally owned so gets object owner from bucket owner which is online
	Object 4	Fail Cannot access object owner	Success Object owned locally	Fail Object is not locally owned so requires accessing the bucket owner which is the failed site
	Object 5	Fail Cannot access object owner	Success Object is not locally owned so gets object owner from bucket owner which it is	Success Object owned locally
	Object 6	Fail Cannot access object owner	Success Object is not locally owned so gets object owner from bucket owner which is online	Success Object owned locally

#### 4.1.2 TSO behavior with access during outage enabled

When a site is first inaccessible to other sites within a replication group the behavior is that detailed in the default TSO behavior section. After the heartbeat is lost between sites for a sustained period of time, the default is 15 minutes ECS marks a site as failed. Enabling **access during outage** (ADO) on a bucket changes the TSO behavior after a site is marked as failed, allowing objects in that bucket to utilize eventual consistency. This means that after a site is marked as temporarily failed, any buckets that have the option **access during outage** enabled will support reads and optionally writes from a non-owner site. It

accomplishes this by allowing usage of the replicated metadata when the authoritative copy on the owner site is unavailable. You can change the **access during outage** bucket option at any time except during a site failure.

The benefit of enabling **access during outage** is it allows access to data after a site is marked as failed; the disadvantage is that the data returned may be outdated.

Starting in version 3.1, an additional bucket option was added for **read-only access during outage** which ensures object ownership never changes and removes the chance of conflicts otherwise caused by object updates on both the failed and online sites during a TSO. The disadvantage of **read-only access during outage** is that after a site is marked as failed no new objects can be created and no existing objects in the bucket can be updated until after all sites are back online. The **read-only access during outage** option is available during bucket creation only, it cannot be modified afterwards.

As previously mentioned, a site is marked as failed when the heartbeat is lost between sites for a sustained period of time, the default is 15 minutes. Therefore, if the heartbeat is lost for a sustained period of time:

- In a two-site configuration, each will consider themselves as online and mark the other as failed.
- In a configuration with three or more sites, a site will only be marked as failed if both:
  - A majority of sites lose heartbeat for the sustained period of time to the same ECS site
  - And if all the remaining sites are currently marked as online

A failed site may still be accessible by clients and applications such as when a company's internal network loses connectivity to a single site, but extranet networking remains operational. As an example, in a five-site configuration if sites 2–5 lose network connectivity to site 1 for a sustained period of time ECS will mark site 1 as temporarily failed. If site 1 is still accessible to clients and applications, it will be able to service requests for locally owned buckets and objects because lookups to other sites is not required. However, requests to site 1 for non-owned buckets and objects will fail. Table 16 shows which access is required after a site is marked as failed for an operation to succeed if **access during outage** is set to enabled.

Table 16 Successful operations after a site is marked as failed with access during outage enabled

Operation	Request sent to the failed site (in a federation that contains three or more sites)	Request sent to an online site including either: <ul style="list-style-type: none"> <li>• Any online site in a federation that contains three or more sites</li> <li>• Or either site in a federation that contains only two sites</li> </ul>
Create object	Success for locally owned buckets unless <b>read-only access during outage</b> is enabled on the bucket.  Fail for remotely owned buckets	Successful unless <b>read-only access during outage</b> is enabled on the bucket.
List objects	Only lists objects in its locally owned buckets if all objects are also locally owned	Successful Will not include objects owned by failed site that haven't finished being replicated
Read object	Success for locally owned objects in locally owned buckets (might not be most recent version)  Fail for remotely owned objects	Successful If the object is owned by the failed site it requires that the original object had finished replication before the failure occurred

Operation	Request sent to the failed site (in a federation that contains three or more sites)	Request sent to an online site including either: • Any online site in a federation that contains three or more sites • Or either site in a federation that contains only two sites
Update object	Success for locally owned objects in locally owned buckets unless <b>read-only access during outage</b> is enabled on the bucket.  Fail for remotely owned objects	Successful, unless <b>read-only access during outage</b> is enabled on the bucket. Acquires ownership of object

- Create object

After a site is marked as failed, create objects will not succeed if **read-only access during outage** is enabled on the bucket. If it is disabled:

- In a federation that contains three or more sites, the site marked as failed, if accessible by clients or applications, can create objects only in its locally owned buckets. These new objects can only be accessed from this site, other sites won't be aware of these objects until the failed site is back online and they gain access to the bucket owner.
- The online sites can create objects in any bucket including buckets owned by the site marked as failed. Creating an object requires updating the bucket listing with the new object name. If the bucket owner is down, it will create an object history which will insert the object into the bucket listing table during recovery or rejoin operations of the bucket owner.

- List object

- In a federation that contains three or more sites, the site marked as failed requires local ownership of both the bucket and all objects within the bucket to successfully list objects in a bucket. The listing from the failed site will not include objects created remotely while it is marked as temporarily failed.
- The online sites can list objects in any bucket including a bucket owned by the site marked as failed. It will list the latest version of the bucket listing that it has, it may be slightly outdated.

- Read object

- In a federation that contains three or more sites, the failed site, if accessible by clients or applications, can read only locally owned objects in locally owned buckets.
- The read request to a failed site first needs access to the bucket owner to validate the current object owner. If it can access the bucket owner and the current object owner is local, then the read request will succeed. If either the bucket owner or the current object owner is not accessible the read request will fail.
- The online sites can read any objects including those owned by the site marked as failed so long as the original object has completed replication. It will check the object history and respond with the latest version of the object that is available. If an object was subsequently updated on the site marked as failed and geo-replication of the updated version didn't complete, the older version will be used to service the read request.

---

**Note:** Read requests sent to online sites where the bucket owner is the failed site will use its local bucket listing information and object history to determine the object owner.

---

- Update object
  - After a site is marked as failed, update objects will not succeed if **read-only access during outage** is enabled on the bucket. If it is disabled, in a federation that contains three or more sites, the failed site, if accessible by clients or applications, can update only locally owned objects in locally owned buckets.
  - The update request first needs access to the bucket owner to validate the current object ownership. If it can access the bucket owner and the current object owner is local, then the update request will succeed. If either the bucket owner or the current object owner is not accessible the update request will fail.
  - After rejoin operations complete, this update will not be included in read operations if the remote site also updated the same object during the same TSO.

An online site can update both objects owned by online sites and failed sites. If an object update request is sent to an online site for an object owned by the site marked as failed, it will update the latest version of the object available on a system marked as online.

The site performing the update will become the new object owner and update the object history with the new owner information and sequence number. This will be used in recovery or rejoin operations of the original object owner to update its object history with the new owner.

---

**Note:** Update requests sent to online sites where the bucket owner is the failed site will use its local bucket listing information and object history to determine the object owner.

---

This example shows what would happen with the bucket and object layout for namespace 1 in a three-site configuration, as shown in Figure 12.

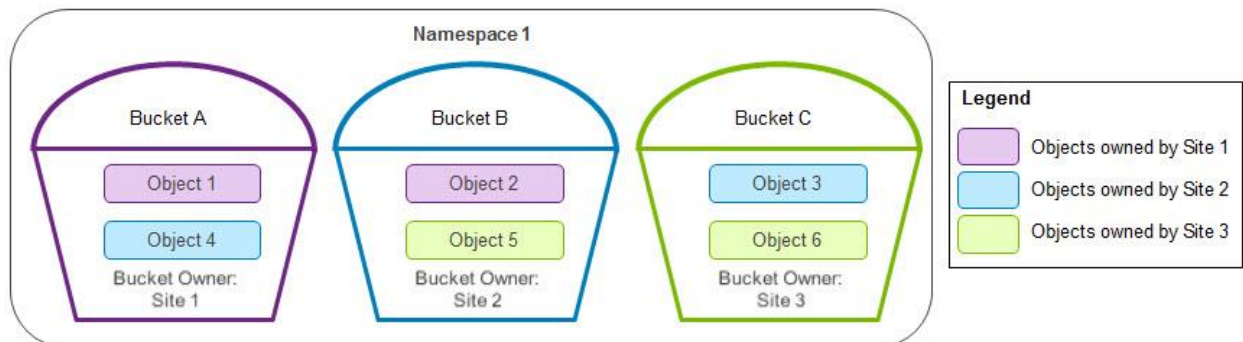


Figure 12 Bucket and object ownership for namespace 1

Table 17 shows an example in this three-site configuration if all three of these are true:

- **Access during outage** is enabled
- And **read-only access during outage** is disabled
- And site 1 is marked as failed

Table 17 Example of operations that succeed or fail with **access during outage** and **read-only access during outage** disabled with Site 1 marked as temporarily failed in a three-site configuration

Operation	Bucket / object	Request sent to	
		Site 1 (marked as failed)	Site 2 or Site 3 (online)
Create objects in	Bucket A	Success	Success
	Bucket B	Fail Failed site can only create objects in locally owned buckets	Success
	Bucket C	Fail Failed site can only create objects in locally owned buckets	Success
List objects in	Bucket A	Fail Although the bucket is owned locally it contains remotely owned objects	Success Will not include objects owned by failed site that haven't been replicated
	Bucket B	Fail Failed site can only list objects in locally owned buckets	Success
	Bucket C	Fail Failed site can only list objects in locally owned buckets	Success
Read or update object	Object 1	Success, both object and bucket are owned locally	Success Requires object to have completed replication prior to TSO Update acquires object ownership
	Object 2	Fail, bucket is not owned locally	
	Object 3 Object 4 Object 5 Object 6	Fail Failed site can only read and update locally owned objects in locally owned buckets	Success

Once the heartbeat is re-established between sites the system marks the site as online and access to this data continues as it did prior to the failure. The rejoin operation will:

- Update the bucket listing tables
- Update object ownerships where necessary
- Resume processing the previously failed sites' replication queue

---

**Note:** ECS only supports access during the temporary failure of a single site.

---

In another two site examples, as shown Figure 13. Both sites will think itself be on line and mark the other site as failed when a TSO happened, All the create, list, read, update operation will be successes.

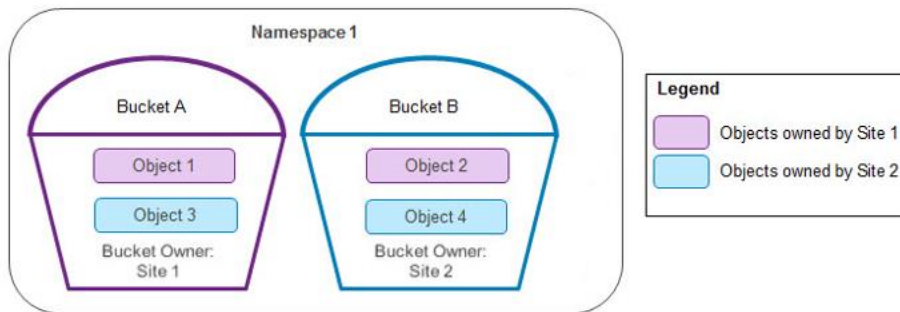


Figure 13 Bucket and object ownership in two sites

During the TSO, all the objects are updated in each site. Table 18 shows the final data in the site once the heartbeat is re-established between sites.

Table 18 Winning site after site re-established

Object	Bucket Name	Bucket Owner	Object Owner	Then "Winning" site is...
Object1	Bucket A	Site 1	Site 1	Site 2
Object2	Bucket B	Site 2	Site 1	The Site has the latest timestamp
Object3	Bucket A	Site 1	Site 2	The Site has the latest timestamp
Object4	Bucket B	Site 2	Site 2	Site 1

**Note:** In this example, the latest timestamp means the object latest updated time in the site.

#### 4.1.2.1 XOR decoding with three or more sites

As we saw in the XOR encoding section, ECS maximizes storage efficiency of data that is configured with a replication group containing three or more sites. The data in secondary copies of chunks may be replaced by data in a parity chunk after an XOR operation. Requests for data in a chunk that has been encoded will be serviced by the site containing the primary copy. If this site is failed the request will go to the site with the secondary copy of the object. However, since this copy was encoded, the secondary site must first retrieve the copy of the chunks that were used for encoding from the online primary sites. Then it will perform an XOR operation to reconstruct the requested object and respond to the request. After the chunks are reconstructed, they are also cached so that the site can respond more quickly to subsequent requests.

Table 19 shows an example of a portion of a chunk manager table on Site 4 in a four-site configuration.

Table 19 Site 4 chunk manager table after completing XOR encoding

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 4	Encoded
C2	Site 2	Site 4	Encoded
C3	Site 3	Site 4	Encoded
C4	Site 4		Parity (C1, C2 & C3)



Figure 14 illustrates the requests involved in re-creating a chunk to service a read request during a TSO.

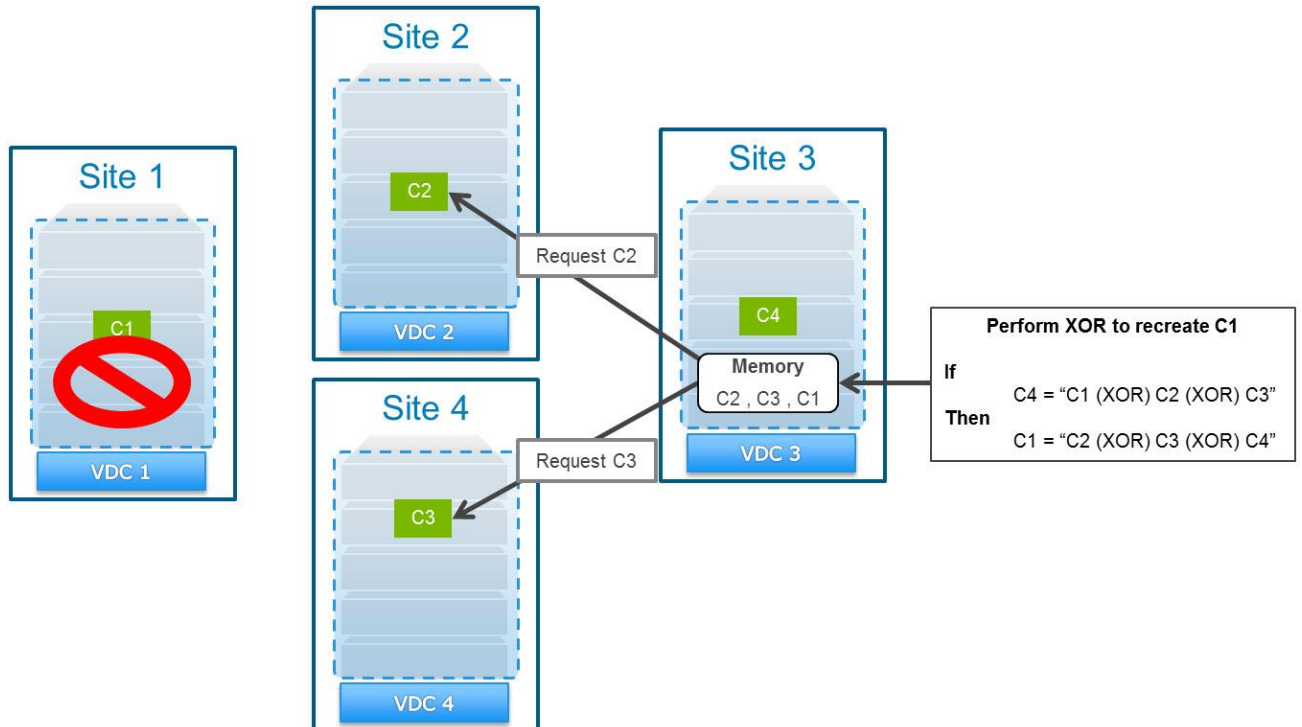


Figure 14 Servicing a read request by reconstructing an XOR'd chunk

In this example if a read came in for an object in chunk **C1** when **site 1** is marked as failed the following would happen:

- Since Site 1 is failed the request would be sent to chunk **C1**'s secondary site, Site 4
- Site 4 has already performed XORs on chunks **C1**, **C2**, and **C3**, meaning it has replaced its local copy of the data from these chunks with data in parity chunk **C4**.
- Site 4 requests a copy of chunk **C2** from its primary site (Site 2) and caches this locally.
- Site 4 requests a copy of chunk **C3** from its primary site (Site 3) and caches this locally.
- Site 4 then performs an XOR operation between the cached chunks **C2** and **C3** with the parity chunk **C4** to re-create chunk **C1** and store it locally in cache.
- Site 4 then responds to the read request for the object in chunk **C1**.

---

**Note:** The time for reconstruction operations to complete increases linearly based on the number of sites in a replication group.

---

#### 4.1.2.2 With geo-passive replication

Any data in a bucket configured with geo-passive replication will have two to four source sites and one or two dedicated replication targets. Data written to the replication targets may be replaced by data in a parity chunk after an XOR operation. Requests for geo-passively replicated data will be serviced by the site containing the primary copy. If this site is inaccessible to the requesting site, the data will need to be recovered from one of the replication target sites.

With geo-passive replication the source sites are always the object and bucket owners. Given this, if a replication target site is marked as temporarily failed all IO operations will continue as usual. The only exception is replication which will continue to queue until the replication target site rejoins the federation.



If one of the source sites fails, requests to the online source site will need to recover non-locally owned data from one of the replication target sites. Let's look at an example where Site 1 and Site 2 are the source sites and Site 3 is the replication target site. In this example an object's primary copy exists in chunk C1 which is owned by Site 1 and the chunk has been replicated to the target destination, Site 3. If Site 1 fails and a request comes into Site 2 to read that object, Site 2 will have to get a copy from Site 3. If the copy was encoded, the secondary site must first retrieve the copy of the other chunk that was used for encoding from the online primary site. Then it will perform an XOR operation to reconstruct the requested object and respond to the request. After the chunks are reconstructed, they are also cached so that the site can respond more quickly to subsequent requests.

Table 20 shows an example of a portion of a chunk manager table on the geo-passive replication target.

Table 20 Geo passive replication target chunk manager table after completing XOR encoding

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 3	Encoded
C2	Site 2	Site 3	Encoded
C3	Site 3		Parity (C1 and C2)

Figure 15 illustrates the requests involved in re-creating a chunk to service a read request during a TSO.

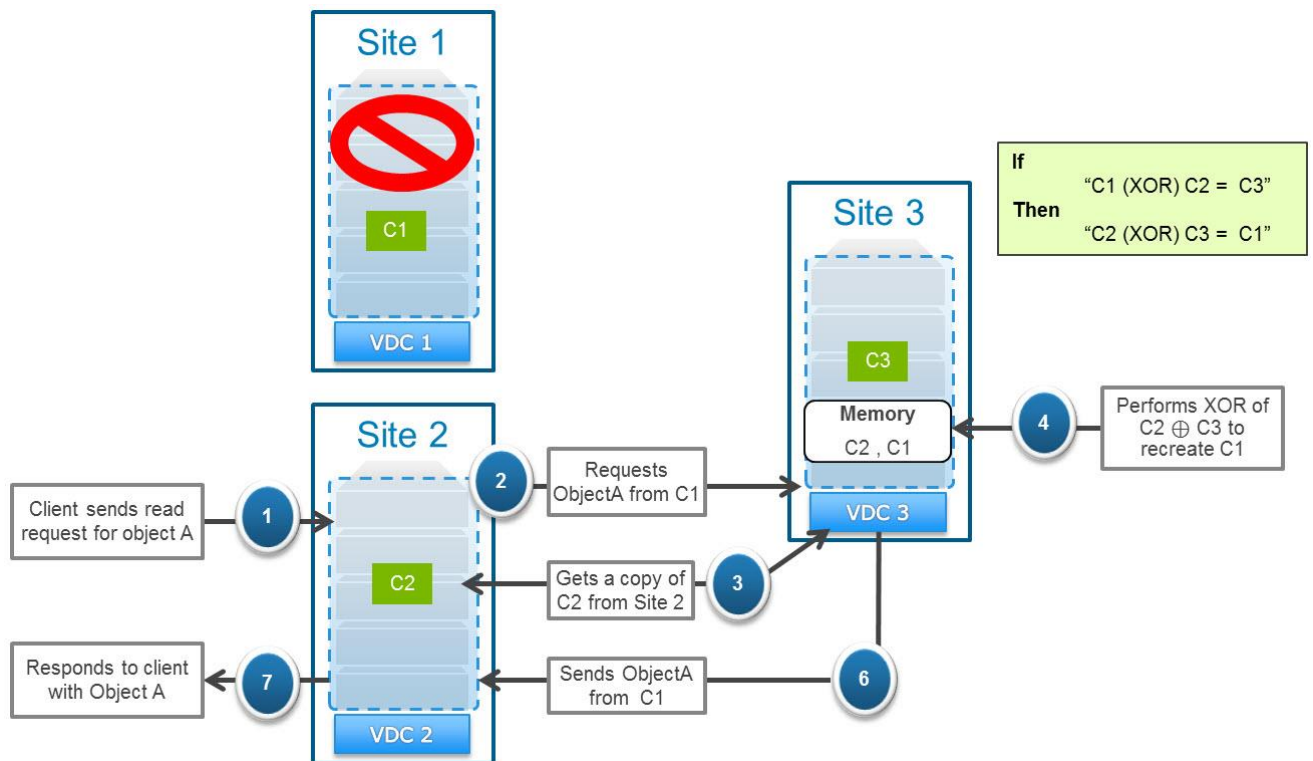


Figure 15 Servicing a read request by reconstructing an XOR'd chunk

In this example, if a read came in for an object in chunk **C1** when **Site 1** is marked as temporarily failed the following would happen:

- Since Site 1 is failed, the request would be sent to chunk C1's secondary site, Site 3.
- Site 3 has already performed XORs on chunks **C1** and **C2**, meaning it has replaced its local copy of the data from these chunks with data in parity chunk **C3**.
- Site 3 requests a copy of chunk **C2** from its primary site (Site 2) and caches this locally.
- Site 3 then performs an XOR operation between the cached chunk **C2** with the parity chunk **C3** to re-create chunk **C1** and store it locally in cache.
- Site 3 then responds to the read request for the object in chunk **C1**.

#### 4.1.2.3 With replicate to all sites enabled

Buckets configured with the options of **replicate to all sites** and **access during outage** can provide faster read performance. The faster read performance comes both during a time when all sites are online, but also during a temporary site outage, since no XOR decoding operation is required and there are more chances that the data will be read locally.

Data in buckets with **replicate to all sites enabled** is replicated to each site. Create and update objects are handled the same as if **replicate to all sites** was disabled. However, read and list objects are handled slightly different because some data might have only completed replication to some but not all sites before the primary site failed.

During a read operation the node servicing the request first checks the latest version of the metadata from the object owner. If the requesting node:

- **Is the object owner:**
  - If it has a local copy of the data being requested, it will use that to service the request.
  - If the object has been updated by another site which failed before it replicated the data, it will return the version that it has locally.
- **Isn't the object owner**
  - If the object owner site is online and replication of the object data:
    - > Has completed to this site then it services the request with its local copy of the data.
    - > Has not completed to this site, it will request a copy from the object owner and it will use that to service the request.
  - If the object owner is down
    - > If replication of the object completed to this site it will use its local copy of the data, this may not be the latest version.
    - > If replication of the object data to the requesting site did not complete the requesting site will request a copy from the secondary site listed first in the chunk manager table. If that site is itself the read operation will fail.

During a list objects in a bucket operation the node requires information from both the bucket owner and head information for each object in the bucket. If the site that is the object owner or the bucket owner is down and **access during outage** is also enabled, it can still service the request if all the remaining sites in the replication group are online. It will list the latest version of the bucket listing that it has, it may be slightly outdated and can vary between sites.

### 4.1.3 Multiple site failures

ECS only supports access during a temporary failure of a single site within a replication group; furthermore, only one site can be marked as failed. This means if more than one site within a replication group is failed concurrently some operations will fail. The first site to be determined to be failed (due to sustained loss of heartbeat) will be marked as failed. Any remaining sites that also have a sustained loss of heartbeat will not be marked as failed and so will be considered online.

As an example, if we have five sites in a replication group and site 1 is identified as having a sustained loss of heartbeat, it is marked as failed. If site 2 is also identified as having a sustained loss of heartbeat it will remain listed as online. The following will occur:

- If **access during outage** is enabled and the bucket owner is Site 2, reads/creates/updates sent to other sites will fail regardless of the object owner. This is because it first checks with the bucket owner to determine the object owner. Unless the bucket owner is marked as failed the requestor will send the request to Site 2, which will fail.
- Read and update requests sent to Site 2 will only succeed if it is the object owner (and bucket owner if **access during outage** is enabled).
- Read and update requests sent to sites other than Site 2 will succeed only if the object owner (and bucket owner if **access during outage** is enabled) is not Site 2.
- Create object will fail if the bucket owner is either Site 1 or Site 2. This is because creating an object requires updating the bucket listing with the new object name. Since it can't succeed doing this on all sites marked as online the create operation will fail.
- Requests to listing objects in a bucket will only succeed if the requesting site can access the bucket owner and all the objects.
  - If the request is sent to Site 2 it will only succeed if it owns the bucket and all the objects in the bucket.
  - If the request is sent to another site it will only succeed if neither the bucket nor any objects within the bucket are owned by Site 2.

## 4.2 Permanent site outage (PSO)

If a disaster occurs at a site and the administrator determines the site is unrecoverable they can initiate a permanent site failover (remove the VDC from the federation). When a permanent site failover is initiated all chunks from the failed site will be recovered on the remaining sites to re-establish data durability.

The recovery process involves the remaining sites scanning their local chunk manager table looking for references to sites that include the failed site. Any that it finds with a chunk type of:

- **Encoded**
  - a. For chunks whose type is Encoded and whose primary site is online, it will re-create the data locally using the data from the primary site. When complete it marks this chunk as a Copy type.
  - b. Next it will re-create the Encoded chunk whose primary site is the failed site by performing an XOR operation of the previously re-created Copy types with the Parity chunk. This site now becomes the chunks' primary site and has a type of Local.
  - c. These chunks are then added to the replication queue to be replicated to other sites listed within the replication group.

- **Copy** and a primary site listed as the failed site, becomes the new primary site. It then adds the chunk to its replication queue to be replicated to a new secondary site.
- **Local** and its secondary site is the failed site, a task is inserted to replicate the chunk to a new secondary site.

Once permanent site failover starts, access to the data owned by the failed site will not be available until after the permanent site failover process is complete. Replication of data is separate from failover operations, and as such it does not have to complete for access to the data owned by the failed site to be restored.

Looking at a three-site example, Site 1 fails and Table 21 and Table 22 are the chunk manager tables of the remaining two sites.

Table 21 Site 2 chunk manager table

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 2	Copy
C2	Site 2	Site 3	Local
C3	Site 1	Site 3	Remote
C4	Site 2	Site 1	Local

Site 2 would perform the following:

- Add chunk **C1** to the replication queue to be replicated. Site 2 will become the new primary site and the site with the new chunk will become the new secondary site.
- Add chunk **C4** to the replication queue to be replicated and update the secondary site in the table

Table 22 Site 3 chunk manager table

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 2	Remote
C2	Site 2	Site 3	Encoded
C3	Site 1	Site 3	Encoded
C4	Site 2	Site 1	Remote
C5	Site 3		Parity (C2 & C3)

Site 3 would perform the following:

1. Re-create chunk **C2** data locally using the data from the primary site (**Site 2**). Change chunk type to Copy.
2. Reconstruct chunk **C3** using **C2** data and the **C5** parity data using the XOR operation  $C2 \oplus C5$ . Site 3 will become the new primary site.
3. Delete chunk **C5**.
4. Add chunk **C3** to the replication queue to be re-replicated, the site with the new chunk will become the new secondary site.

After permanent site failover completes the chunk manager, tables of the two remaining sites would be like that shown in Table 23 and Table 24.

Table 23 Site 2 chunk manager table after PSO completes

Chunk ID	Primary site	Secondary site	Type
C1	Site 2	Site 3	Local
C2	Site 2	Site 3	Local
C3	Site 3	Site 2	Copy
C4	Site 2	Site 3	Local

Table 24 Site 3 chunk manager table after PSO completes

Chunk ID	Primary site	Secondary site	Type
C1	Site 2	Site 3	Copy
C2	Site 2	Site 3	Copy
C3	Site 3	Site 2	Local
C4	Site 2	Site 3	Copy

### 4.2.1 PSO with geo-passive replication

Permanent site outage is handled slightly differently for data replicated using geo-passive replication. Geo-passive replicated data will not re-establish data durability during a PSO; instead it is re-established after a new third site is added to the replication group. The PSO operations differ based on whether the site that has permanently failed is one of the source sites or the replication target site.

The recovery process still involves the remaining sites scanning their local chunk manager table looking for references to sites that include the failed site. Any that it finds with a chunk type of:

- **Encoded** (exists on the replication target site)
  - For chunks whose type is Encoded and whose primary site is online, it will re-create the data locally using the data from the primary site. When complete it marks this chunk as a Copy type.
  - Next it will re-create the Encoded chunk whose primary site is the failed source site by performing an XOR operation of the previously re-created Copy types with the Parity chunk. This site now becomes the chunks' primary site and has a type of Local. No secondary sites will be created until after a third site is added to the replication group.
- **Copy** and a primary site listed as the failed site, becomes the new primary site and its type is changed to Local. No secondary sites will be created until after a third site is added to the replication group.
- **Local** and its secondary site (the replication target) is the failed site. No new secondary sites will be created until after a third site is added to the replication group.

Post-PSO, a third site can be added to re-establish data durability and protect against site-wide failures. After a third site is added to the geo-passive replication group the previous two sites will scan their local chunk manager table looking for chunks with no secondary chunk listed. It will then:

- Local chunks on a source site with no secondary chunk listed will initiate the replication of a secondary chunk to the new replication target site. The chunk manager table will be updated to include the new secondary chunk location.
- Local chunks on the replication target site will initiate a replication of the chunk to a new source site. After replication completes the replication target site type will change from Local to Copy, and the source site type will change from Copy to Local. XOR operations will continue on the destination as normal.

Once PSO starts on a source site, access to the data owned by the failed site will not be available until after the permanent site failover process is complete. Once PSO is complete access to the data is restored. Until a third site is added to the replication group, any new writes to the online source site will be replicated to the replication target but XOR operations will not occur. This is because XOR only runs on chunks from two different source sites, since all new source sites are the same, XOR can't run.

After PSO completes a third site can be added to the replication group to restore data durability and protect against site-wide failures. Also, the replication target can resume running XOR operations on any two chunks from different source sites marked as type Copy.

Let's look at a couple of examples where Site 1 and Site 2 are the source sites, and Site 3 is the target destination site. Table 25 and Table 26 are the chunk manager tables of the two source sites and Table 27 is the chunk manager table of the replication target site.

Table 25 Site 1, source site chunk manager table

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 3	Local
C2	Site 2	Site 3	Remote
C3	Site 1	Site 3	Local

Table 26 Site 2, source site chunk manager table

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 3	Remote
C2	Site 2	Site 3	Local
C3	Site 1	Site 3	Remote

Table 27 Site 3, replication target chunk manager table

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 3	Encoded
C2	Site 2	Site 3	Encoded
C3	Site 1	Site 3	Copy
C4	Site 3		Parity (C1 & C2)

**Example 1:** If Site 3 is removed because of a PSO, the secondary sites would all become empty but the primary sites and types would remain. Until a new replication target is added, any new writes will have a primary site listed but no secondary site.

**Example 2:** If Site1 is removed because of a PSO, the following would happen:

- Site 3 will become the new primary site with a type of Local for chunk “**C3**” and no site will be listed as the secondary site.
- Site 3 will re-create chunk **C2** data using the data from the primary site (Site 2) and change its chunk type to Copy.
- Site 3 will reconstruct chunk **C1** using **C2** data and the **C4** parity data using the XOR operation  $C2 \oplus C4$ . Site 3 will become the new primary site; no secondary site will be listed.
- Site 3 will delete chunk **C4**.

After a permanent site failover of Site 1 completes, the chunk manager tables of the two remaining sites would be like that shown in Table 28 and Table 29.

Table 28 Site 2, source site chunk manager table after PSO completes

Chunk ID	Primary site	Secondary site	Type
C1	Site 3		Remote
C2	Site 2	Site 3	Local
C3	Site 3		Remote

Table 29 Site 3, replication target site chunk manager table after PSO completes

Chunk ID	Primary site	Secondary site	Type
C1	Site 3		Local
C2	Site 2	Site 3	Copy
C3	Site 3		Local

Until a new source site is added, any new writes will have a primary site of Site 2 with a type of Local and a secondary site of Site 3 with a type of Copy.

After a new source site is added to the replication group data durability to protect against site wide failure will be re-established by adding a secondary site and replicating the data to it. XOR operations will also resume on the replication target. The new chunk manager tables would be as shown in Table 30 through Table 32.

- Chunks C1 and C3 will be replicated to the new source site, Site 1. After replication completes the primary site will be listed as Site 1 and the secondary site as Site 3.
- Site 3 will perform XOR encoding on Chunks C1 and C2 resulting in a new C4 chunk with a type of Parity and chunks C1 and C2's type will be change to Encoded.

Table 30 New Site 1 chunk manager table after data durability re-established

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 3	Local
C2	Site 2	Site 3	Remote
C3	Site 1	Site 3	Local

Table 31 Site 2 chunk manager table after new Site 1 added and data durability re-established

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 3	Remote
C2	Site 2	Site 3	Local
C3	Site 1	Site 3	Remote

Table 32 Site 3, replication target site chunk manager table after new Site 1 added and data durability re-established

Chunk ID	Primary site	Secondary site	Type
C1	Site 1	Site 3	Encoded
C2	Site 2	Site 3	Encoded
C3	Site 1	Site 3	Copy
C4	Site 3		Parity (C1 & C2)

## 4.2.2 Recoverability from multiple site failures

ECS only supports recovery from a single site failure at a time. ECS can recover from multiple site failures if both PSO and data recovery operations complete between the site outages. If the second site fails prior to recovery completing:

- For permanent site outage recovery operations to run, all other sites in the system must be online. If there are multiple concurrent site failures, all but one must recover from the TSO before a PSO can be run on a site.
- If there is a second site failure that occurs after PSO completed but before data recovery completes, some data may be lost.



Looking at a four-site scenario, we successfully recover from losing all but one site (assumes there is sufficient space to store all the data in the remaining site):

- Site 4 fails
  - Administrator initiates a PSO operation to remove Site 4
  - Data is recovered on the remaining sites to re-establish data durability

We are now left with a three-site federation containing Site 1, Site 2, and Site 3.

- Second site fails, Site 2:

Sometime after PSO and data recovery complete another site can fail, such as Site 2.

- Administrator initiates a PSO operation to remove Site 2
- Data is recovered on the remaining sites to re-establish data durability

We are now left with a two-site federation containing Site 1 and Site 3.

- Third site fails, Site 1:

Sometime after PSO and data recovery complete another site can fail, such as Site 1.

- Administrator initiates a PSO operation to remove Site 1

We are now left with a single site federation containing Site 3.

This example walked through multiple site failures. This is not a normal scenario; permanent site failures are generally caused by disaster scenarios such as earthquakes and fires and as such are not common to occur in multiple locations in short succession. Typically, after a single site permanently fails, a new site will be added before a subsequent site fails.

## 5 Conclusion

The ECS architecture has been designed from the ground up to provide both system availability and data durability. ECS allows the administrator granularity in how they balance availability requirements with TCO. Features like automatic failure detection and self-healing capabilities minimize IT administrative workloads at the most critical times, when there is an unplanned event such as a site outage.

ECS protects data within a site/VDC against disk failures utilizing a combination of triple mirroring and erasure coding. ECS offers two levels of erasure coding protection, the default for typical use cases and cold storage which is more efficient for infrequently accessed objects. It also distributes the data across failure domains to provide protection against the most failure scenarios.

ECS ensures data integrity by calculating and writing checksums as part of a write operation and validating these checksums during a read operation. The checksum validation is also performed proactively in a background task.

ECS is designed to continue to provide system availability. This is accomplished with the distributed architecture design which allows client requests to be serviced by any node in a site/VDC.

The ECS design extends the system availability and data durability protection by adding optional protection against a complete site-wide failure. It does this by federating sites and allowing the administrator to configure a variety of replication group policy options. These options can be set at the bucket level and determine where to replicate data and how to store the data in the remote site(s) as well as access during outage options.

Additionally, ECS offers customers with an option of "access during outage" which allows read, list and optionally write and update operations sent to an online site when the bucket and/or object is marked as failed.

If an administrator determines a site is unrecoverable they can initiate a permanent site outage. This will remove the VDC/site from the replication group and re-create data as needed to re-establish data durability.

In conclusion, ECS provides an enterprise quality cloud storage solution with built-in resiliency that you can trust.

## A Technical support and resources

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

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

### A.1 Related resources

- [ECS Overview and Architecture White Paper](#)
- [ECS Community](#)
- [ECS Test Drive](#)
- ECS product documentation at [Support site](#) or [Community site](#)
- [SolVe Desktop](#) (Procedure Generator)