Configuring Resiliency and Management Interface Bonding

To help protect against loss of assets due to network failure, COS Release 3.5.2 supports data resiliency:

- At the node level, enabling recovery of data lost due to drive failure within a node.
- At the cluster level, enabling recovery of lost data due to node failure within a cluster.

In addition, COS supports bonding of two management interface ports to provide redundant connectivity in the event of a network interface card (NIC) failure.

This section describes these features and provides instructions for configuring them through the COS setup file or (for data resiliency only) through the COS Service Manager GUI.

Configuring Resiliency

COS provides resiliency at the node and cluster level using one of two methods: mirroring or erasure coding. Resiliency at the node level is achieved using either local mirroring (LM) or local erasure coding (LEC), which is the default. Similarly, resiliency at the cluster level is achieved using either remote mirroring (RM), which is the default, or distributed erasure coding (DEC).

You can configure resiliency either directly, by updating the COS file /arroyo/test/aftersetupfile, or indirectly, by creating and applying asset redundancy policies through the COS Service Manager GUI. COS also allows for the configuration of mixed resiliency policies through the GUI.

Asset redundancy policies are assigned at the service endpoint level. Because an endpoint can only be associated with a single cluster and policy, asset redundancy policies are always applied at the cluster level, rather than at the node level. When you apply a policy to an endpoint, the COS AIC client writes the appropriate line(s) to the setup file for each node in the cluster.

Note We recommend using either remote mirroring or DEC, but not both. COS 3.5.2 does not support migration from one scheme to another while preserving stored content.

An endpoint can be configured in the GUI, but it must remain in Disabled state until it is associated with both a cluster and an asset redundancy policy. There is no default policy, policy type, or rule associated with an endpoint in the GUI. After an asset redundancy policy is applied to an endpoint, you can change to a different policy simply by applying it to the endpoint instead.

About Mirroring

Mirroring configures the disks in a node, or the nodes in a cluster, to hold a specified number of exact copies of the object data. The number of copies can be set from 1 to 4, with 1 representing the original object data only, and 4 representing the object data plus three copies. Thus, for example, a value of 2 specifies one copy plus the original object data.

To configure mirroring in the GUI, you specify the number of desired copies differently for local and remote mirroring:

- For local mirroring, the number you specify includes the original plus the number of local copies. For example, setting the number to 3 in the GUI specifies the original and two local copies.
- For remote mirroring, the number you specify includes only the number of remote copies. For example, setting the number to 3 in the GUI specifies three remote copies in addition to the (local) original.

About Erasure Coding

Erasure coding, or software RAID, is a method of data protection in which cluster data is redundantly encoded, divided into blocks, and distributed or *striped* across different locations or storage media. The goal is to enable any data lost due to a drive or node failure in the cluster to be reconstructed using data stored in a part of the cluster that was not affected by the failure.

COS 3.5.2 supports both local erasure coding (LEC), in which data is striped across the disks in a node, and distributed erasure coding (DEC), in which data is striped across nodes in a cluster.

For both LEC and DEC, data recovery is performed as a low-priority background task to avoid possible impact on performance. Even so, erasure coding provides faster data reconstruction than hardware RAID. Speed is important in maintaining resiliency, because a failed node cannot help to recover other failed nodes until it has fully recovered.

Defining Resiliency

Two key factors define the degree of resiliency of a given cluster configuration:

- *Resiliency ratio* (RR) functionally divides the disks or nodes in a cluster into data blocks and parity blocks. The RR of a cluster indicates how much of its total storage is devoted to protection against data loss. Conventionally, the RR is expressed as an *N:M* value, where N represents the number of data blocks and M is the number of parity blocks.
- *Resiliency factor* (RF) is the number of data (N) blocks that can fail at one time before losing the ability to fully recover any lost data. The RF is equivalent to the parity (M) value in the RR for the cluster, and can be selected in the GUI as described in About Mirroring, page C-2.

For LEC, COS assigns a default RR of 12:2, but you can choose another RR up to 18:4 maximum. For DEC, the cos-aic-client calculates the RR for a chosen cluster size and RF, but you can choose another RR up to 18:18 maximum. As you consider whether to use the defaults or assign new values, you must weigh a number of factors specific to your deployment, such as:

- The amount of storage that can be devoted to resiliency
- The degree of redundancy (RF) that must be achieved
- How quickly failed data blocks can be recovered

The following examples illustrate how these factors can be used to determine the best resiliency scheme for a particular COS node or cluster.

Example 1: LEC at Node Level

To see how the LEC resiliency ratio affects individual node behavior, consider a cluster in which each node has 4 TB disks, runs at 80% full, and can maintain a data read rate of 20 Gbps (2.5 GB/s).

If each node in the cluster has an assigned LEC RR of 10:1:

- Each node can recover from a single disk failure on its own, but if more than one concurrent disk failure occurs, DEC must be used to rebuild the data.
- 1 MB of parity data is created for every 10 MB of data written to each node, representing a storage overhead of 1/10 or 10%. Also, in the event of a lost disk, the rebuild logic must read in 10 MB of data for every 1 MB recovered, representing a rebuild overhead of 10:1.
- If a node loses a single disk when running at 80% full, then (4 TB x 0.8 =) 3.2 TB of data must be rebuilt. At a 20 Gbps data read rate and a 10:1 rebuild overhead, the data rebuild rate is (20 Gbps / 10 =) 2 Gb (256 MB) per second, or roughly 0.9 TB per hour. This means that 3.2 TB of data can be rebuilt in under 4 hours.

By comparison, if the same cluster were assigned the default LEC RR of 12:2 instead of 10:1:

- Each node can recover from two concurrent disk failures on its own, before using DEC.
- Each node stores 2 MB of parity for every 12 MB of data, representing a 17% a storage overhead and a 12:1 rebuild overhead.
- At a data rebuild rate of (20 Gbps / 12 =) 0.75 TB per hour, recovery takes about 4.3 hours for one lost disk (3.2 TB), or 8.6 hours for two lost disks.

Note

- Actual data rebuild times are highly dependent on the capacity of the hardware components in the server. Servers that can deliver significantly more throughput reduce the rebuild time, while servers with less capacity increase the rebuild time. LEC uses local disk reads when recovering data, so the speed of the disk channel directly impacts recovery time.
- Rebuild activity runs at a lower priority than regular object reads and writes, in an effort to maintain normal performance during rebuild periods. However, because normal reads and writes take priority over repair activity, a busy server takes longer to rebuild data from a lost disk than an idle server, which can devote more bandwidth to repair.
- An exception to the lower priority for rebuild activity is made for *on-demand* repair work. This occurs when a client requests data that is waiting to be rebuilt due to a disk failure. In this case, the requested data is rebuilt immediately so that the repair work is transparent to the clients.

Example 2: DEC at Cluster Level

To see how the DEC resiliency ratio affects cluster configuration and behavior, consider a cluster of 11 nodes, with each node running 80% full, maintaining a data read rate of 16 Gbps (2 GB/s), and having a total storage capacity of 100 TB.

If the cluster has an assigned DEC RR of 8:2:

- The cluster requires a minimum of 10 active nodes (8 data plus 2 parity) to continue writing data at the assigned 8:2 resiliency ratio. With 11 nodes, this cluster can lose one node and still continue to provide the same resiliency for newly added objects.

- The cluster can recover from two concurrent node failures, but to do so, must have enough free disk space to hold the recovered data for two nodes. For an 11-node cluster, this means that 2/11, or just under 20%, of the total disk space must be kept free. This can be reduced to 10% if it is safe to assume that at least one replacement node can be brought online.
- 2 MB of DEC parity data is created for every 8 MB of data written to the cluster, representing a storage overhead of 2/8 or 25%. This compares favorably to mirroring, whose storage overhead is four times greater (100%) but only enables recovery from one node failure, rather than two. Also, in the event of a lost node, the rebuild logic must read in 8 MB of data for every 1 MB recovered, representing a rebuild overhead of 8:1.
- If one node is lost when running at 80% full, then (100 x 0.8 =) 80 TB of data must be rebuilt by the 10 remaining nodes in the cluster. Dividing the task equally among the remaining 10 servers, each must rebuild 8 TB of data. At a 16 Gbps data read rate and an 8:1 rebuild overhead, the data rebuild rate is (16 Gbps / 8 =) 2 Gb (256 MB) per second, or 0.9 TB per hour. At this rate, each node can rebuild its 8 TB share of the total lost data in 8.9 hours.

By comparison, if the same cluster were assigned a DEC RR of 8:1 instead of 8:2:

- The cluster can fully recover data from only a single failed node, but requires only half the free disk space (10%) of the 8:2 RR case to support this recovery.
- Each node stores 1 MB of parity for every 8 MB of data, representing a 12.5% a storage overhead and an 8:1 rebuild overhead.
- Because the rebuild rate is determined by the N value (8), it is the same for 8:1 as for 8:2. At a data rebuild rate of (16 Gbps / 8 =) 0.9 TB per hour, recovery of one lost node (8 TB) takes 8.9 hours.

Note

- Actual data rebuild times are directly affected by the number of nodes in the cluster. If there are fewer nodes in the cluster, the work is divided among fewer servers, resulting in longer rebuild times. DEC uses remote data reads across the network when recovering data, so the speed of the network channel directly impacts the recovery time.
- Actual data rebuild times are highly dependent on the capacity of the hardware components in the server. Servers that can deliver significantly more throughput reduce the rebuild time, while servers with less capacity increase the rebuild time.
- Rebuild activity runs at a lower priority than regular object reads and writes, in an effort to maintain normal performance during rebuild periods. However, because normal reads and writes take priority over repair activity, a busy cluster takes longer to rebuild data from a lost node than an idle cluster, which can devote more bandwidth to repair.
- An exception to the lower priority for rebuild activity is made for *on-demand* repair work. This occurs when a client requests data that is waiting to be rebuilt due to a disk failure. In this case, the requested data is rebuilt immediately so that the repair work is transparent to the clients.

Example 3: Using LEC and DEC Together

To see how LEC and DEC work together, consider the cluster configuration described in the previous examples, but with a LEC RR of 10:1 for each node *and* a DEC RR of 8:2 for the cluster as a whole.

With this configuration:

- The cluster can recover from two concurrent node failures as well as a single disk failure per node at the same time. A LEC RR of 10:1 means that a node must lose at least two disks concurrently before DEC is required to recover the lost data.

- Because the DEC parity data is distributed across the nodes in the cluster, it is further protected by LEC at the node level. This allows recovery from disk failure to be performed locally, including the DEC parity data stored on the node.
- The DEC RR of 8:2 means that every 10 MB block of data includes 8 MB for content storage and 2 MB for DEC parity. In addition, the LEC RR of 10:1 means that every 10 MB block of (content + DEC parity) data requires another 1 MB of parity data. So in total, every 8 MB of content storage requires (2 DEC + 1 LEC =) 3 MB of parity data, giving a total parity overhead of (3 / 8 =) 37.5%.

  For any cluster configuration using both DEC and LEC, the total parity overhead is found by first applying the DEC overhead and then applying the LEC overhead, as follows:

  Total Parity Overhead = (M1/N1) + ((1 + M1/N1) X (M2/N2))

  where N1:M1 represents DEC parity and N2:M2 represents LEC parity.

  Using the values from this example to illustrate:

  Total Parity Overhead = (2/8) + ((1 + 2/8) X (1/10) = 37.5%

Note The parity calculations just described do not account for the additional free disk space needed for storage of recovered data. Be sure to include this requirement in your total overhead calculations. For example, a DEC RR of 8:2 requires free disk space for up to two failed nodes.

Configuring Resiliency Using the GUI

To configure resiliency through the COS Service Manager GUI:

Step 1 Log in to the GUI as described in Using the COS Service Manager GUI, page B-1.

Step 2 On the COS Home page, click to select the service to be configured in the Service Summary section.

The Cloud Object Stores page opens displaying the service definition and its service endpoints.

Step 3 Click the check box to the left of a service endpoint and click the **Edit** icon to enable it for editing.

Step 4 Choose the desired resiliency policy from the Asset Redundancy Policy drop-down list for the endpoint.

Note

- If the desired policy does not appear, choose **Service Domain Objects > Asset Redundancy Policies** to confirm the policy exists. If not, click **Add Row** and create a new policy. Enter a profile name and expected cluster size, select a model (appropriate interfaces and names auto-populate), assign the profile to a cluster, configure a DNS server, and assign IP pools to each interface.
- Beginning with COS 3.5.2, the GUI supports entering an expected cluster size when creating the COS node initialization profile. For DEC, the cos-aic-client uses this cluster size to calculate suitable N:M values given the expected cluster size and the selected resiliency factor. If necessary, these values can be overridden as described in Finding N:M Values, page C-8 by editing or creating the **/arroyo/test/aftersetupfile** COS file to hold them.

Caution When manually setting N:M (or any other) values that must persist, be sure to use aftersetupfile and not setupfile. The settings in setupfile can be overwritten by changes made via the GUI. Also, you must configure the aftersetupfile before registering the new node to the PAM. Otherwise, DEC settings within the cluster will be inconsistent, requiring at least one service-disrupting reboot to correct.

Configuring Local Mirroring Manually

To configure local mirroring on a node manually:

Step 1 Open (or if not present, create) the COS file **/arroyo/test/aftersetupfile** for editing.

Step 2 Include the line **vault local copy count** in the file and set the value to **2**, **3**, or **4** as appropriate.

Note Setting the value to **1** simply maintains the original data and creates no additional copies.

Step 3 Disable local erasure coding by setting **allow vault raid** to **0** (or simply omit or remove this line).

Example

```
# CServer core configuration. Changes to this file require a server reboot.
    serverid 1
    groupid  3333
    arrayid  6666
    . . . .
    allow vault raid 0
    vault local copy count 2
    vault mirror copies 2
    allow server raid 0
    allow tcp traffic 1
    . . . .
    er_enable 0
    rtp_enable 0
```

Configuring Local Erasure Coding Manually

To enable local erasure coding manually:

Step 1 Open (or if not present, create) the COS file **/arroyo/test/aftersetupfile** for editing.

Step 2 Set **allow vault raid** to **1** to enable LEC.

Step 3 Disable local mirroring by setting **vault local copy count** to **1** (or simply omit or remove this line).

Example

```
# CServer core configuration. Changes to this file require a server reboot.
    serverid 1
    groupid  3333
    arrayid  6666
    . . . .
    allow vault raid 1
    vault local copy count 1
    vault mirror copies 2
    allow server raid 0
    allow tcp traffic 1
    . . . .
    er_enable 0
    rtp_enable 0
```

Migrating from LM to LEC Manually

To migrate a service endpoint from local mirroring to local erasure coding:

Step 1 Temporarily leave local mirroring enabled for the service endpoint.

Step 2 Enable local erasure coding for the service endpoint and allow it to establish the needed parity for each data object.

Step 3 When parity is established, disable local mirroring.

Configuring Remote Mirroring Manually

To enable and configure remote mirroring manually:

Step 1 Open (or if not present, create) the COS file **/arroyo/test/aftersetupfile** for editing.

Step 2 Set **vault mirror copies** to the value **2**, **3**, or **4** as appropriate to enable remote mirroring. The value you enter specifies the object data plus the number of exact copies desired.

Note Setting the value to **1** simply maintains the original data and creates no additional copies.

Step 3 Disable distributed erasure coding by setting **allow server raid** to **0** (or simply omit or remove this line).

Example

```
# CServer core configuration. Changes to this file require a server reboot.
    serverid 1
    groupid  3333
    arrayid  6666
    . . . .
    allow vault raid 0
    vault local copy count 2
    vault mirror copies 2
    allow server raid 0
    allow tcp traffic 1
    . . . .
    er_enable 0
    rtp_enable 0
```

Configuring Distributed Erasure Coding Manually

To enable and configure distributed erasure coding manually:

Step 1 Open (or if not present, create) the COS file **/arroyo/test/aftersetupfile** for editing.

Step 2 Set **allow server raid** to **1** and add the following lines immediately below:

- **target server raid data blocks <value>**

  This controls the number of data blocks used. The default <value> is 8, and the valid range is 1-18.

- **target server raid parity blocks <value>**

  This controls the number of parity blocks used. The default <value> is 1, and the valid range is 1-18.

Note See Finding N:M Values, page C-8 to determine appropriate data block and parity block values.

Step 3 Disable remote mirroring by setting **vault mirror copies** to **0** (or simply omit or remove this line).

Example

```
# CServer core configuration. Changes to this file require a server reboot.
    serverid 1
    groupid  3333
    arrayid  6666
    . . . .
    allow vault raid 0
    vault local copy count 2
    vault mirror copies 1
    allow server raid 1
    target server raid data blocks 8
    target server raid parity blocks 1
    allow tcp traffic 1
    . . . .
    er_enable 0
    rtp_enable 0
```

Finding N:M Values

To configure DEC, you must specify the number of data blocks (N) and parity blocks (M) used for data encoding. Table C-1 shows the corresponding data-to-parity block (N:M) values for a given number of nodes in a cluster and for a given degree of resiliency desired for the cluster. For details, see Defining Resiliency, page C-2.

Note COS does not currently support configuration of new N:M (data:parity) block values through the COS Service Manager GUI. If you need to configure new N:M values, you must do so in the **aftersetup** file.

In this table:

- **Nodes** is the number of nodes in the cluster.
- **RF** is the desired *resiliency factor*, or number of nodes that can fail without data loss.
- **Min** is the minimum number of nodes required to achieve a given resiliency factor.

The ratios appearing in the cells of the table are N:M values, where N is the number of data blocks and M is the number of parity blocks needed to achieve the desired resiliency factor for a given node count.

To use the table to find the N:M values for a cluster:

Step 1 In the Nodes column, locate the **row** corresponding to the number of nodes in the cluster.

Note For COS 3.5.2, you must select the N:M configuration based upon the initial nodes in the cluster. COS does not currently support adding nodes to a cluster after DEC is configured for the cluster.

Step 2 Locate the **column** in the table whose header represents the desired RF value for the cluster.

Step 3 Find the corresponding **N:M value** at the intersection of the row and column just located.

Step 4 Configure DEC using **N** as the number of data blocks and **M** as the number of parity blocks.

Table C-1 Possible N:M Values for DEC

Nodes	Min = 1 RF = 0	Min = 3 RF = 1	Min = 5 RF = 2	Min = 7 RF = 3	Min = 9 RF = 4
1	1:0	—	—	—	—
2	1:0	—	—	—	—
3	1:0	1:1	—	—	—
4	1:0	2:1	—	—	—
5	1:0	3:1	2:2	—	—
6	1:0	4:1	3:2	—	—
7	1:0	5:1	4:2	3:3	—
8	1:0	6:1	5:2	4:3	—
9	1:0	7:1	6:2	5:3	4:4
10	1:0	8:1	7:2	6:3	5:4
11	1:0	8:1	8:2	7:3	6:4
12	1:0	8:1	8:2	8:3	7:4
13	1:0	8:1	8:2	9:3	8:4
14	1:0	8:1	8:2	10:3	9:4
15	1:0	8:1	8:2	11:3	10:4
16	1:0	8:1	8:2	12:3	11:4
17	1:0	8:1	8:2	12:3	12:4
18	1:0	8:1	8:2	12:3	12:4
19	1:0	8:1	8:2	12:3	12:4
20	1:0	8:1	8:2	12:3	12:4

Configuring Management Interface Bonding

COS supports the ability to bond two NICs so that they appear to the host node as a single logical management interface. With management interface bonding, two ports on a node are defined as a primary-backup pair.

- For the C3260, the designated management ports are eth0 and eth1.
- For the C3160, the designated management ports are eth0 and eth3.
- For the CDE465, the designated management ports are eth0 and eth1.

Note COS 3.5.2 supports bonding of management interfaces for resiliency, but not for improved performance.

If the two NICs in a node are bonded, the management link is maintained if one logical interface or its physical link are lost. The management link is also maintained if one physical link is disconnected and then reconnected, and then the other physical link is disconnected. The management interface bonding feature itself remains enabled if the node is rebooted.

To Configure Bonding Manually

To configure management interface bonding manually:

Step 1 Open (or if not present, create) the COS file **/arroyo/test/aftersetupfile** for editing.

Step 2 Add the line **management bond <value>**, where <value> is **0** to disable or **1** to enable the feature.

When enabled, one NIC serves as the primary management interface and the other as the backup interface.