Overview

Product Description

Cisco Cloud Object Storage (COS) provides distributed, resilient, high-performance storage and retrieval of binary large object (blob) data. Object storage is distributed across a cluster of hardware systems, or nodes. The storage cluster is resilient against hard drive failure within a node and against node failure within a cluster. Nodes can be added to or removed from the cluster to adjust cluster capacity as needed.

The primary interface for managing COS content is the OpenStack Swift API, with enhancements to improve quality of service when accessing large media objects. COS includes an authentication and authorization service that implements the OpenStack Swauth API. To administer the cluster, COS includes an HTTP-based cluster-management API.

COS and MOS

COS can be installed as a service of Cisco Media Origination System (MOS). COS is designed to integrate transparently with MOS, and can be managed through the MOS Service Manager GUI web application.

MOS is designed for highly optimized ingest and storage. MOS uses a hierarchical storage design that can support huge content libraries while simplifying content storage management. Its distributed architecture can separate ingest and storage from streaming. This allows each function to be scaled independently as needed to dynamically increase network ingest and storage resources.

For details on MOS, see the Cisco Media Origination System User Guide for your MOS release.

Note

COS 3.8.1 has been tested for compatibility with MOS Release 2.7. Later releases of COS are expected to be compatible with later versions of MOS. Contact Cisco for the latest information.

COS and Cloud DVR

Beginning with Release 3.8.1, COS adds support for API calls that enable COS to manage fanout storage operations for applications such as Cloud DVR (cDVR). Fanout storage efficiently supports unique copies for fair-use compliance. A single fanout request can save many copies of an object, thereby saving network resources by optimizing storage compute and disk utilization. The COS Fanout API includes
calls to create, retrieve, and delete fanout objects and to create, retrieve, and delete individual copies of content within a fanout object. The Fanout API also enables interoperability between COS and Cisco Virtual Media Recorder (VMR) as part of a complete cDVR solution under shared MOS management.

Components

COS has a number of subsystems.

- **Networks**: Interfaces are grouped into distinct networks to isolate management functions from high-volume data traffic.
- **Clusters and Nodes**: COS services are provided by a cluster of nodes, with both the cluster and the individual nodes as distinctly manageable components.
- **Object Metadata Store**: The metadata for the cluster is stored in a high-performance distributed NoSQL database hosted on the COS nodes in a cluster.
- **Platform and Applications Manager (PAM)**: COS components are managed using services running on the PAM.
- **Hardware Platforms**: COS software is currently deployed on selected Cisco Content Delivery Engine (CDE) and Cisco UCS server hardware models.

The following sections further describe each of these components.

Networks

COS divides network interfaces into two groups: the management network and the data network. The management network is used for monitoring and managing the COS cluster and individual COS nodes. The data network is used by client applications to interact with the COS authentication and authorization services, and the COS object storage services. Client applications use the Swauth API to interact with the COS authentication and authorization services, and the Swift API to interact with the COS object storage services.

In customer installations, COS management network traffic can be routed with management traffic meant for non-COS systems. To ensure that high-volume data traffic does not congest networks used to manage COS and other systems, each network adapter in a COS node is assigned to either the management network or the data network. Typically, high-bandwidth network adapters are assigned to the data network, with one or two lower-bandwidth adapters assigned to the management network.

Platform and Applications Manager (PAM)

The PAM is a virtual instance dedicated to the installation, configuration, monitoring, and recovery of other COS components. It acts as the management machine for the COS system, and runs the following services and applications:

- The Platform Manager, which facilitates network setup and configuration of external DNS servers, and which hosts the NTP Server
- The Document Server, including MongoDB and Redis
- The COS Service Manager
- The COS Application Instance Controller (AIC), which facilitates management of COS appliances
For more information on the PAM and its components, see the *Cisco Media Origination System User Guide* for your MOS release.

**COS Nodes**

The COS software runs on a collection of computing systems called *nodes*, which are connected via the management and data networks. Currently, there are two types of COS nodes: the cluster controller, which runs the PAM software and the COS AIC software, and the storage nodes.

The storage nodes host software that manages object-store and authentication and authorization service metadata, stores and retrieves object contents, and communicates with the cluster controller. COS storage nodes can be added or removed without disrupting COS service availability. Adding nodes is a way of elastically increasing the storage and bandwidth capacity of the COS cluster.

The COS node software includes a customized Linux distribution, currently based on CentOS 6. This provides the basic framework for the other software applications and modules that run on the node. Each node runs a set of kernel modules and a number of daemons that run in the Linux user-space.

The kernel modules:

- Support real-time management of node hardware resources.
- Provide the distributed, resilient content-store used for object-store data.
- Provide the Swift and Swauth API support via the data network.

The daemons:

- Coordinate service log files.
- Communicate with the cluster controller.
- Provide a distributed database for object-store metadata.
- Communicate with the modules running in the kernel.

While the data-network interfaces communicate directly with the kernel modules, the management network interfaces communicate directly with the user-space daemons.

**COS Cluster**

COS services are provided by software running on a set of nodes called a *COS cluster*. The nodes in the cluster are connected by both data and management networks. COS Release 3.8.1 supports one cluster per PAM deployment. Each cluster has a single fully-qualified domain name (FQDN) that is used by client applications to access COS services.

A COS cluster also has a number of configuration parameters that define the cluster behavior. Some of these parameters include:

- The Swift and Swauth API constraints.
- The IP address pools used to assign IP addresses to individual node network adapters.
- The IP address of the PAM configuration document server.

For a detailed description of the configuration parameters, see *Deploying COS, page 2-1*. 
Object Store Metadata

COS object store metadata and Swauth service data are stored in the high-performance, resilient NoSQL Cassandra database. The cosd daemon running on each COS node acts as the Cassandra client, and implements the schema for Swift and Swauth metadata documents stored in Cassandra. Each COS storage node runs an instance of the Cassandra server, so metadata storage capacity increases linearly along with content storage capacity as COS storage nodes are added to the cluster.

Hardware Platforms

Currently, COS 3.8.1 software can be deployed on the following hardware models:

- Cisco UCSC C3260-4U5 Dual Node Rack Server with 56 x 10 TB hard drives (560 TB total storage), giving 28 drives (280 TB) to each server node
- Cisco UCSC C3260-4U4 Single Node Rack Server with 56 x 6 TB hard drives (336 TB total storage), giving all 56 drives to one server node
- Cisco UCSC C3260-4U3 Dual Node Rack Server with 56 x 6 TB hard drives (336 TB total storage), giving 28 drives (168 TB) to each server node
- Cisco UCSC C3160-4U2 Rack Server with 54 x 6 TB hard drives (324 TB total storage)
- Cisco UCSC C3160-4U1 Rack Server with 54 x 4 TB hard drives (216 TB total storage)
- Cisco Content Delivery Engine CDE465-4R4 with 36 x 6 TB hard drives (216 TB total storage)

For information about installing the hardware, see the following:

- Cisco UCS C3260 Rack Server Installation and Service Guide
- Cisco UCS C3160 Rack Server Installation and Service Guide
- Cisco Content Delivery Engine 465 Hardware Installation Guide

Features

- Overview, page 1-5
- Server Support, page 1-9
- Automated COS Node Configuration, page 1-11
- Intel Preboot Execution Environment (PXE) Support, page 1-11
- Improved TCP Transmission, page 1-11
- Small Object Support, page 1-11
- COS Service Manager GUI, page 1-12
- High Availability (HA), page 1-12
- Swauth API, page 1-13
- Swift Object Store API, page 1-13
- Fanout API, page 1-14
- Object Store Metadata Resiliency, page 1-14
- Object Store Data Resiliency, page 1-15
Overview

The table below provides an overview of the COS features.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Features</th>
</tr>
</thead>
</table>
| Cisco UCS and CDE Server Support     | - Supports installation on the following hardware:  
  - Cisco UCSC C3260-4U5 Dual Node Rack Server with 56 x 10 TB hard drives (560 TB total storage), giving 28 drives (280 TB) to each server node  
  - Cisco UCSC C3260-4U4 Single Node Rack Server with 56 x 6 TB hard drives (336 TB total storage), giving all 56 drives to one server node  
  - Cisco UCSC C3260-4U3 Dual Node Rack Server with 56 x 6 TB hard drives (336 TB total storage), giving 28 drives (168 TB) to each server node  
  - Cisco UCSC C3160-4U2 Rack Server with 54 x 6 TB hard drives (324 TB total storage)  
  - Cisco UCSC C3160-4U1 Rack Server with 54 x 4 TB hard drives (216 TB total storage)  
  - Cisco Content Delivery Engine CDE465-4R4 with 36 x 6 TB hard drives (216 TB total storage)  
  - COS 3.8.1 provides a pre-installation script to configure the C3260 server for single or dual node service.                                                                 |
| Automated Node Configuration         | - A single configuration file for all COS nodes can be stored on the PAM or on an FTP or HTTP server, and then downloaded by the COS initialization routine (cosinit) during installation.  
  - A single downloadable configuration file eliminates the need to configure nodes individually, whether manually or via the COS Service Manager GUI.  
  - COS 3.8.1 lets you specify the URL of a configuration file to be used at installation to automatically configure the node according to a predefined template.                                                                 |
| Intel Preboot Execution Environment  | - PXE can be used to download a network bootstrap program (NBP) to remotely install a COS client over a network.                                                                                                                                               |
| (PXE) Support                        |                                                                                                                                                                                                                                                                  |

Table 1-1  Overview of COS Features
### Table 1-1 Overview of COS Features

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved TCP Transmission</td>
<td>• COS 3.8.1 includes optimizations to improve TCP transmit performance.</td>
</tr>
<tr>
<td>Small Object Support</td>
<td>• For cloud DVR and similar applications, COS 3.8.1 introduces Small Object Support to efficiently manage storage of many small files representing media segments.</td>
</tr>
<tr>
<td>COS Service Manager GUI</td>
<td>• Lets you quickly and easily access many COS deployment, monitoring, and alarm functions.</td>
</tr>
<tr>
<td></td>
<td>• Displays storage, network bandwidth, session count, and alarms for individual COS disks, nodes, services, and interfaces.</td>
</tr>
<tr>
<td></td>
<td>• Includes a graphical display of deployment statistics and trends related to disk, service, and interface status.</td>
</tr>
<tr>
<td></td>
<td>• Supports configuration of COS node service interface from the GUI.</td>
</tr>
<tr>
<td></td>
<td>• Supports setting of resiliency policies on a per-node basis from the GUI.</td>
</tr>
<tr>
<td></td>
<td>• Includes the COS Configuration Wizard, which guides you through the steps for configuring a COS cluster and (optionally) generating a configuration profile.</td>
</tr>
<tr>
<td>High Availability (HA)</td>
<td>• COS supports HA as implemented in MOS, providing redundancy for the PAM VMs. The PAM uses both Cisco and third-party components to support HA.</td>
</tr>
<tr>
<td>Swauth API</td>
<td>• Simple Auth Service API for authentication of Swift operations.</td>
</tr>
<tr>
<td></td>
<td>• Based on Swauth Open-Source Middleware API.</td>
</tr>
<tr>
<td></td>
<td>• Used to manage accounts, users and account service endpoints.</td>
</tr>
</tbody>
</table>
Table 1-1  Overview of COS Features

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swift Object Store API</td>
<td>• An implementation of a subset of the continually evolving OpenStack Swift API.</td>
</tr>
<tr>
<td></td>
<td>• Command executions are authenticated using auth tokens provided by Swauth service.</td>
</tr>
<tr>
<td></td>
<td>• Used to create and manage containers and objects for persistent storage in a COS cluster.</td>
</tr>
<tr>
<td></td>
<td>• Supports archiving of content from Cisco or ARRIS recorders using DataDirect Networks (DDN) Web Object Scaler (WOS) archive objects.</td>
</tr>
<tr>
<td>Fanout API</td>
<td>• COS 3.8.1 includes support for a Fanout API to enable interactions with other Cisco applications in the Virtualized Video Processing (V2P) suite.</td>
</tr>
<tr>
<td>Object Store Metadata Resiliency</td>
<td>• Metadata resiliency is provided by a distributed and replicated Cassandra document database.</td>
</tr>
<tr>
<td></td>
<td>• Each COS node participates in the persistence of a subset of the Cassandra database.</td>
</tr>
<tr>
<td></td>
<td>• Manual administrative intervention is required on node failure.</td>
</tr>
<tr>
<td>Feature Set</td>
<td>Features</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Object Store Data Resiliency</td>
<td>- Data is resilient to both hard drive and COS node failures.</td>
</tr>
<tr>
<td></td>
<td>- Local Erasure Coding (LEC), or local COS node data resiliency, is provided by local software RAID.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>By default, LEC is enabled and is configured for two drive failures. We recommend using this default configuration for resiliency.</td>
</tr>
<tr>
<td></td>
<td>- Distributed erasure coding (DEC) provides data resiliency across nodes, protecting stored content from loss due to node failure.</td>
</tr>
<tr>
<td></td>
<td>- COS cluster data resiliency is provided by object replication (mirroring). The PAM section of the GUI allows for configuration of both local and remote mirror copies.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>When configuring local mirroring for resiliency, we recommend using no more than one local mirror copy.</td>
</tr>
<tr>
<td></td>
<td>- Supports configuration of mixed resiliency policies (local erasure with remote mirroring) via the GUI.</td>
</tr>
<tr>
<td></td>
<td>- Alarms are available for loss of storage.</td>
</tr>
<tr>
<td>Management Interface Bonding</td>
<td>- Supports defining two node management interface ports as a primary-backup pair.</td>
</tr>
<tr>
<td>Service Load Balancing</td>
<td>- COS cluster load balancing is provided by DNS round-robin of a FQDN to multiple physical IPv4 addresses hosted by COS nodes.</td>
</tr>
<tr>
<td></td>
<td>- Optimal load balancing is provided by extensions to the Swift API through the implementation of HTTP redirect.</td>
</tr>
<tr>
<td></td>
<td>- Remote smoothing facilitates load balancing by moving content to a new node when it is added to a cluster.</td>
</tr>
</tbody>
</table>
Server Support

Cisco UCS C3260 Rack Server

COS 3.8.1 supports the Cisco UCS C3260 platform, which supports up to two compute nodes and up to 56 storage disks per chassis.

The C3260 is a 4RU chassis that supports a single-node or dual-node server node configuration. When configured for single-node, the chassis includes the following:

- 2 x 480 GB solid-state drives (SSDs) for operating system and COS installation
- 28 or 56 hard drives for content storage
- One server node with 32 x 16 GB RAM, providing 256 GB
- 1 system I/O controller with 2 x 40 Gbps QSFP ports

When configured for dual-node, the chassis includes the following:

- 4 x 480 GB solid-state drives (SSDs) for operating system and COS installation, 2 per node
- 56 hard drives for content storage, with the drives in slots 1-28 dedicated to server node 1 and the drives in slots 29-56 dedicated to server node 2
- Two server nodes with 32 x 16 GB RAM each, providing 256 GB for each node
- 2 system I/O controllers with 2 x 40 Gbps QSFP ports each, one controller dedicated to each server node

A pre-installation script is provided to properly configure the C3260 chassis for either single-node or dual-node service. You must run this script before installing COS on any C3260 server node.
After COS installation and during the cosinit sequence on each node, you are prompted to select one of three available storage bundles:

- **UCS C3260-4U3 (28 disks per server node):** Select this bundle if you configured a single COS node with 28 hard drives installed, or a dual COS node setup with 28 x 6 TB hard drives per server node.
- **UCS C3260-4U4 (56 disks per server node):** Select this bundle if you configured a single COS node with 56 hard drives.
- **UCS C3260-4U5 (56 disks per server node):** Select this bundle if you configured a dual COS node setup with 28 x 10 TB hard drives per server node.

Knowing which storage bundle is configured allows the system to more accurately report inventory and disk issues, such as bad or missing disk drives, after the node is up and running.

In a dual-node setup, the web GUI displays the status of only those disks assigned to a particular node:

- Node1 will list Cisco Disk 01-28
- Node2 will list Cisco Disk 29-56

On each COS node, eth0 and eth1 are bonded to a bond0 management interface. This differs from the UCS-C3160, where eth0 and eth3 are bonded to a bond0 management interface.

For more information, see Deploying COS, page 2-1.

### Cisco UCS C3160 Rack Server

The Cisco UCS C3160 is a modular, high-density server for service providers, enterprises, and industry-specific environments. The C3160 combines highly scalable computing with high-capacity local storage. Designed for cloud-scale applications, the C3160 is simple to deploy and is well suited for use in unstructured data repositories, media streaming, and content distribution applications.

The C3160 is a 4RU server. When configured for COS, the C3160 includes the following:

- 2 x 400 GB solid-state drives (SSDs) in RAID1, typically located in slots 55 and 56, for operating system and COS installation
- 54 hard drives in JBOD mode for 216 TB (4 TB drives) or 324 TB (6 TB drives) total storage
- One rear SSD
- Two system I/O controllers providing a total of four 10 GbE ports

A pre-installation script is provided to properly configure the C3160 chassis for COS. You must run this script before installing COS on the C3160.

For more information, see Deploying COS, page 2-1.

### Cisco CDE Family Support

The Cisco Content Delivery Engine (CDE) family of rack servers supports ingest, storage, distribution, delivery, and management functions in the context of systems for delivery of entertainment-grade video content to subscribers. Each CDE contributes one or more support functions as determined by the content delivery applications (CDAs) that run on it.

The Cisco CDE465 Rack Server is designed and tested specifically to work with COS and related applications. The CDE465 provides enhanced storage capacity relative to earlier CDE models, with current models offering either 216 or 324 TB total storage.
Automated COS Node Configuration

Beginning with COS 3.5.1, you can automate node configuration by providing a file to cosinit, the COS initialization routine, that includes a cluster name and IP pool reference address for at least one service interface. COS initialization will then configure the node without further intervention through the COS Service Manager GUI or the API. A single configuration file for all COS nodes (or node sets) can be stored on the PAM or an HTTP server for download by cosinit.

Beginning with COS 3.5.2, you can create the configuration file automatically through the COS Service Manager GUI, as the final option presented in the Configuration Wizard. Using a configuration template means that the following files no longer have to be configured for each COS node:

- /arroyo/test/setupfile
- /arroyo/test/SubnetTable
- /arroyo/test/RemoteServers
- /etc/cassandra/conf/cassandra.yaml
- /etc/cosd.conf
- /opt/cisco/cos/config/cos.cql

The COS Configuration Wizard also guides the user through the required UI configuration steps. See Using the COS Configuration Wizard, page B-11 for details.

Beginning with COS 3.8.1, you can specify the URL of a configuration file to be used at installation to automatically configure the node according to a predefined template. Configuration of the node then proceeds automatically using the settings provided in the configuration file. This eliminates the need to configure nodes individually via the GUI or the API. This feature saves time by allowing for fully automated PXE installations as well as reduced effort during manual installation. For instructions, see Automated Configuration at Installation (Optional), page 2-33.

Intel Preboot Execution Environment (PXE) Support

PXE can be used to download a network bootstrap program (NBP) to remotely install a COS client over a network.

Improved TCP Transmission

COS 3.8.1 includes optimizations to improve TCP transmit performance.

Small Object Support

For cloud DVR and similar applications, COS 3.8.1 introduces small object support to efficiently manage storage of many small files representing media segments.

In Cloud DVR (cDVR) applications, the use of segmented media recording has the potential to greatly reduce the amount of duplicate video data stored on disk. However, this potential is limited by the large number of individual files created by media segmentation. Because each of these files is much smaller than a single disk allocation unit, it cannot be stored efficiently on the disk. In addition, having a large number of small files risks using up all available system object IDs (OIDs) before the disk is full. The potential for lost storage efficiency is only compounded when mirroring or erasure coding is applied for data resiliency.
Beginning with Release 3.8.1, COS supports more efficient storage of multiple small files through a new technique called small object support. This technique maps multiple small files to a larger virtual container file which, due to its larger size, makes more efficient use of a disk allocation unit. In this context, the small files are called *small objects* and can be up to 32 MB in size. The container file is called a *container object*, can hold up to 64K (65535) small objects, and can be up to 256 MB in size. COS small object support integrates with distributed erasure coding to maintain parity data and fault tolerance as new files are added to or deleted from the container file. While the individual small files remain differentiated in the object database, they are managed at the object storage level as a single large object that can be stored safely and efficiently. A background "garbage collection" process recycles space in container objects that frees up when small objects within that container are deleted.

**COS Service Manager GUI**

The COS Service Manager GUI enables quick and easy configuration of the COS infrastructure, service domain objects, and services. The GUI also includes the COS Configuration Wizard, which provides guided steps for configuring a COS cluster.

The GUI also provides valuable monitoring functions, including graphical displays of storage, network bandwidth, session count, and alarms and alarm history for individual COS disks, nodes, services, and interfaces. The GUI also displays system and service diagnostics as well as event logs and log analysis.

The GUI supports configuration of the COS node service interface, setting of resiliency policy (erasure coding or mirroring) on a per-node basis, and decommissioning of COS nodes from the GUI, with improved management of node and cluster maintenance.

For information on launching and navigating the GUI, see Using the COS Service Manager GUI, page B-1.

**High Availability (HA)**

The PAM has two classes of components for HA:

- Third party components such as ZooKeeper, MongoDB, and Redis use their own proprietary clustering and redundancy schemes.
- Cisco components, such as the COS Service Manager GUI and DocServer, use ZooKeeper for leader election.

In an HA environment, multiple PAM VMs provide redundancy for the applications that run on the PAM. HA requires at least three PAM VMs because applications such as ZooKeeper, MongoDB, and Redis require at least three components to form a working cluster.

Many of these applications also require a majority in order to form a quorum. That is, a cluster of three components can recover from the failure of a single component, because there are still two components to form a majority. But if two components fail, the single remaining component is not a majority, and the cluster cannot recover until one of the failed components recovers.

Therefore, we recommend configuring more than three PAM VMs to ensure recovery in the event of multiple failures, and to support high performance, especially when sharing databases and other applications.
Swauth API

COS includes a basic authentication service that can be used when COS is not installed along with other OpenStack services, such as the Keystone Identity service. The API for the COS authentication service is derived from the OpenStack Swauth middleware component API.

The authentication service API provides the following functions for managing accounts, users, and service endpoints:

- Listing Accounts
- Retrieving Account Details
- Creating an Account
- Deleting an Account
- Creating or Updating a User
- Retrieving User Details
- Deleting a User
- Creating or Updating Account Service Endpoints
- Getting an Authentication Token

For details, see the Cisco Cloud Object Storage Release 3.8.1 API Guide.

Swift Object Store API

The COS object storage API is based on the OpenStack Swift API. It is implemented as a set of Representational State Transfer (REST) web services. All account, container, and object operations can be performed with standard HTTP calls. The requests are directed to the host and URL described in the X-Storage-Url HTTP header, which is part of the response to a successful request for an authentication token.

The COS object storage API defines restrictions on HTTP requests. The following table lists these restrictions, which are borrowed from the Swift API.

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum # of HTTP Headers per request</td>
<td>90</td>
</tr>
<tr>
<td>Maximum length of all HTTP Headers</td>
<td>4096 bytes</td>
</tr>
<tr>
<td>Maximum length per HTTP request line</td>
<td>8192 bytes</td>
</tr>
<tr>
<td>Maximum length of container name</td>
<td>256 bytes</td>
</tr>
<tr>
<td>Maximum length of object name</td>
<td>1024 bytes</td>
</tr>
</tbody>
</table>

The container and object names must be UTF-8 encoded and then URL-encoded before inclusion in the HTTP request line. All the length restrictions are enforced against the URL-encoded request line.

The COS object store API provides the following functions, some of which provide extended functionality beyond the standard Swift API defined by OpenStack:
Fanout API

COS 3.8.1 includes support for a Fanout API to enable interactions with other Cisco applications in the Virtualized Video Processing (V2P) suite. A typical use case for such interactions is a cloud DVR (cDVR) workflow, which can involve several separate applications each dedicated to a specific part of the workflow such as ingest, recording, and storage.

The Fanout API is supported in production environments only for configurations of three or more nodes.

The Fanout API uses a single request to create, get, or delete multiple copies of an object (hence fanout). Each copy is treated not as an individual object, but instead, is accessed by specifying the object URL and including in the request header a zero-based index to the requested copy. This saves on network resources by optimizing storage, compute, and disk utilization.

Along with Fanout API support, COS 3.8.1 supports basic authentication for password-protected access to Cisco Virtual Media Recorder (VMR). COS manages a single user name and credentials for use with VMR. This VMR “user” can be created, listed, and have its credentials modified using the COS Configuration API. For details, see the Cisco Cloud Object Storage Release 3.8.1 API Guide.

Object Store Metadata Resiliency

COS stores metadata for Swift and Swauth accounts, users, containers, and objects as documents in a Cassandra database instance. Cassandra is a distributed document store. In a typical multi-node Cassandra cluster, no single node persists (saves) a copy of the entire database to local disk. Instead, each
Cassandra cluster node locally persists a subset of the database. To ensure resiliency of the data in case of node failure, Cassandra has configuration options to specify the number of document replicas to maintain on separate cluster nodes.

For metadata resiliency in COS, each COS cluster node participates in the Cassandra cluster, and each COS node locally persists a part of the Cassandra database. The database cluster is automatically configured to create document replicas to be resilient to a single node failure.

⚠️ Caution ⚠️

There is a risk of data loss if a second node fails before full metadata resiliency is restored, or before full content resiliency is restored.

### Object Store Data Resiliency

The COS stores Swift object data to the local drives within the chassis. To maintain data resiliency in the event of a failed local hard drive, COS 3.8.1 enables local erasure coding (LEC) by default. LEC distributes redundant data across local hard drives (two parity blocks for 12 data blocks), enabling full recovery of lost data if any two drives in the set should fail.

🔍 Note

COS 3.8.1 also supports mirroring of local hard drives as an option. However, LEC is enabled by default, and is the generally recommended choice.

If LEC is enabled and a local hard drive fails, the COS system immediately begins to regenerate any lost data due to the drive failure and place it on the surviving hard drives to regain the intended resiliency. Execution of this recovery process is scheduled with low priority, and recovery time depends on the availability of system resources, available storage capacity, and the amount of data lost.

COS cluster data resiliency is provided by object replication, or mirroring. The PAM section of the COS Service Manager GUI allows for configuration of both local and remote mirror copies.

For data resiliency in the event of a COS node failure, the COS cluster can be configured to maintain copies of object data on one or more additional COS nodes. Recommended practice is to configure the COS cluster to maintain at least two copies of object data for resiliency.

When configured for multiple object copies, the COS cluster automatically attempts to create the configured object copy count within the cluster in the event of a COS node failure, without manual intervention. As soon as the COS cluster detects a node failure, the cluster begins to create additional copies of objects stored on the failed node. Upon restoring the failed node, the COS cluster purges unnecessary copies to recover storage space.

🔍 Note

When configuring local mirroring for resiliency, we recommend using no more than one local mirror copy.

As an alternative to mirroring for data resiliency across nodes in a cluster, COS 3.8.1 supports distributed erasure coding (DEC). DEC allows for recovery of corrupted data in the event of loss of up to two nodes in a cluster. If a node fails, COS immediately begins to regenerate the data from the lost node and place the missing data blocks on the surviving nodes. Execution and duration of this recovery process are scheduled with low priority, and recovery time depends on the availability of system resources, network availability, available storage capacity, and the amount of data lost.
COS 3.8.1 also allows for configuration of mixed resiliency policies (local erasure coding with remote mirroring) via the GUI. Additionally, COS notifies the operator if the system gets close to the maximum loss of resiliency as defined by the SLA, and alarms if resiliency is actually lost.

For additional details, see Configuring Resiliency and Management Interface Bonding, page C-1.

Management Interface Port Bonding

Extending resiliency to the network management interface, COS 3.8.1 also supports defining two node ports as a primary-backup pair for management interface bonding. For the C3160, the designated ports are eth0 and eth3, and for the CDE465, the designated ports are eth0 and eth1.

For additional details, see Configuring Resiliency and Management Interface Bonding, page C-1.

Service Load Balancing

The COS cluster is composed of COS nodes, each having limited CPU, network, and disk resources. To ensure best performance and quality of service, the workloads of the Swift and Swauth operations must be distributed effectively among the nodes. The recommended solution for service load balancing is to use a DNS system to round-robin clients to different physical IP addresses hosted by the various nodes. While not perfect, such a DNS round-robin solution should provide sufficient distribution of workloads.

In addition to using DNS to distribute workload, the COS Swift implementation supports intelligently redirecting a Swift client to an optimal location for Swift object create and read operations using standard HTTP redirect semantics. Given a Swift client that supports HTTP redirect semantics, the client can provide an X-Follow-Redirect: true HTTP header in the HTTP PUT and GET requests for Swift object create and read operations. In the event that a more optimal location is used for the operation, the COS node will respond with an HTTP 307 (temporary redirect) status, indicating to the client where the operation should be requested.

For Swift object read operations, COS provides two levels of service and transfer profile: best-effort and committed rate. These levels of service contribute to service load balancing. COS provides extensions to Swift object read that allow the client to request a guaranteed and committed transfer rate as the data is sent from the COS node.

A COS node can reject a read request if the client has requested a committed rate transfer, but the COS node does not have sufficient resources available to satisfy the client request. If a client does not request a committed rate transfer, the COS node attempts to satisfy the request with the system resources available and at a priority lower than that of any in-progress committed rate requests. For more information, see the Cisco Cloud Object Storage Release 3.8.1 API Guide.

Beginning with COS 3.5.1, a remote smoothing feature facilitates load balancing by shifting content to a new node after it has been added to the cluster.

CLI Utilities

COS provides the following command line utilities for use on Linux:
- cos-swift – provides command-line access to the Swift API.
- cos-swauth – provides command-line access to the Swauth API.
Note

These utilities do not work between two COS nodes or between a COS node and a local node, as the HTTP request will be refused.

For more information on the COS command line utilities, see COS Command Line Utilities, page D-1.

Endpoint and Cluster Support

Each COS service instance can have its own endpoint, cluster, and asset redundancy policy. Each COS endpoint can be enabled and disabled individually and dynamically, and each has its own AppStatus message for reporting service level agreement (SLA) status.

If an endpoint is enabled or disabled, only the network interfaces of the COS nodes attached to the endpoint or cluster are added to or removed from the DNS.

COS Release 3.8.1 supports endpoint per service instance, and one service instance per deployment.

COS AIC Client Management

The COS AIC Client process is monitored by the monit process that runs on each COS node. The AIC Client process creates a PID file that is added to the monit script so that it can be monitored and restarted automatically if the monit process discovers the AIC Client process not running.

Command line scripts are also available to stop and restart the AIC Client process manually, bypassing the automatic restart process.

Node Decommissioning Paused for Maintenance Mode

If a COS node is in the process of being decommissioned when it or any other node in its cluster is placed in Maintenance mode, the decommissioning process is paused to preserve the intended cluster resiliency.

Prerequisites

The COS management and configuration operations require specific hardware components for deployment. For more information on the hardware requirements, see the Cisco Media Origination System User Guide for your MOS release.

The COS system is most effective in engineered networks, with separate routes for management and data flow. In designing and provisioning networks, capacity for the high data-network throughput for the expected application of COS must be ensured. Also, the high data traffic generated by the COS systems must not interfere with the management network segment or other important network segments.

Restrictions and Limitations

- COS does not support IPv6.
• The OpenStack Swift and Swauth APIs continue to evolve, and COS does not currently implement all Swift or Swauth API functions. For a list of currently supported functions, see Swift Object Store API and Swauth API in this chapter.

• Secure Sockets Layer (SSL) or other means for providing session security and encryption are not supported with the Swift and Swauth APIs.

• COS Release 3.8.1 does not support upgrade from, or downgrade to, any earlier COS release.

• See the Release Notes for Cisco Cloud Object Storage 3.8.1 for open caveats and known issues related to this release.