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.

COS has two primary interfaces for content management:

- The OpenStack Swift API, with enhancements to improve quality of service when accessing both large and small media objects
- The Fanout API for efficient storage of unique copies for fair-use compliance

As a managed service of the Cisco Virtualized Video Processing Controller (V2PC), COS is managed through the V2PC graphical web user interface (GUI), which uses REST APIs to simplify COS setup and management. COS also includes a command-line interface (CLI) for management of remote or programmatic content. In addition, COS provides authentication and authorization services using the OpenStack Swauth API.

Through its various management interfaces, COS provides access to large and small media objects, maintains high quality of service, supports cluster management, and coordinates the replication of data across sites to improve resiliency and optimize the physical location of stored data.

NCOS 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.

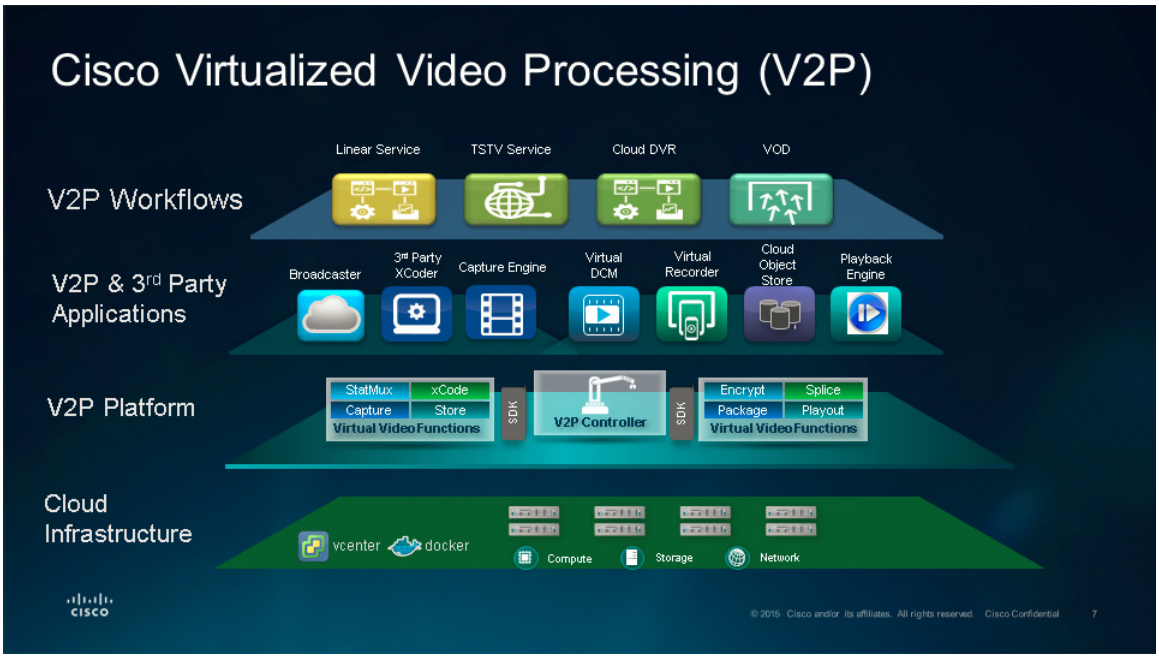
The COS Fanout API includes calls to create, retrieve, and delete fanout objects and to create, retrieve, and delete individual copies of content within a fanout object. The Fanout API also enables interoperability between COS 3.20.1 and Cisco Virtual Media Recorder (VMR) as part of a complete cDVR solution.

COS and V2PC

COS 3.20.1 is installed as a service of Cisco Virtualized Video Processing Controller (V2PC). V2PC provides a common management interface for COS, VMR, and other applications that together form a complete virtualized media processing solution.

V2PC is the control interface for the Cisco Virtualized Video Platform (V2P), an open platform that transforms the way video infrastructure is built, deployed, provisioned, and maintained. V2PC enables a video processing application to run over a cloud or on-premise infrastructure while flexibly orchestrating its media workflows and resources. COS integrates transparently with V2PC, and can be managed through the V2PC graphical user interface (GUI) web application.

Figure 1-1 Cisco Virtualized Video Processing (V2P) Platform



Customers can use V2PC to rapidly create and orchestrate media workflows across video headends and data center environments, and can evolve seamlessly from a hardware based infrastructure to a hybrid or pure virtualized cloud infrastructure. The software centric workflows increase the reachability of content across a variety of content consumption platforms.

This transformation has resulted in flexible user experiences and simplified operations, allowing customers to better manage, modify, and scale media workflows to support services such as Live, VOD, Time Shift, and Cloud DVR (cDVR) to OTT consumers.

V2PC works with a hierarchy of components that includes platforms, application containers, service containers, providers, zones, nodes, and the logical functions they support, which are configured into media workflows.

For more information on V2PC and its components, see the *Cisco Virtualized Video Processing Controller User Guide* for your V2PC release.

Note COS Release 3.20.1 has been tested for compatibility with V2PC Release 3.3.4 build 17417 and cos-app build cisco-cos-bundle-320.1.19. Later releases of COS are expected to be compatible with later versions of V2PC and cos-app. Contact Cisco for updated compatibility information.

COS Components

COS has a number of subsystems.

- **Networks:** Interfaces are grouped into distinct networks to isolate management functions from high-volume data traffic.
- **Nodes, Sites, and Clusters:** A COS installation includes one or more individual COS servers, or nodes. Nodes are configured into sites, which may (but need not) correspond to geographic regions, and also into clusters, which may (but need not) contain multiple sites.
- **Object Metadata Store:** The metadata for the cluster is stored in a high-performance distributed NoSQL database hosted on either the COS nodes in a cluster or a dedicated COS metadata cluster (CMC). Using a separate CMC allows COS object data and metadata to scale independently.
- **Virtualized Video Processing Controller (V2PC):** COS 3.20.1 components are managed using services running on the V2PC.
- **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 these components.

Networks

COS divides network interfaces into two groups: the data network and the management network.

- The management network is used to monitor and manage COS clusters and individual COS nodes. The management network also handles traffic from the COS metadata store.
- 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 for Video on Demand (VoD) services 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. Client applications for Cloud DVR (cDVR) and Time Shift TV (TSTV) services use the Fanout API.

Similarly, the COS installation separates its traffic into data and management traffic, and expects these two types of traffic to be isolated into their own subnets. COS management traffic on 1G management adapters can be combined with other traffic not intended for the COS system. However, COS data traffic on 10G adapters should be on a managed subnet that does not permit traffic not intended for COS. If non-COS traffic is allowed on a COS data subnet, it will degrade system performance and can cause availability issues.

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 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 3.20.1 supports one cluster per V2PC 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 V2PC 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.

Earlier COS releases stored Cassandra metadata on COS nodes along with object data. COS Release 3.20.1 and subsequent releases support the use of a separate COS Metadata Cluster (CMC) for each COS cluster. Moving Cassandra metadata from the COS nodes to one or more dedicated CMC nodes allows Cassandra metadata to grow without encroaching on the space available for data on the COS nodes.

COS Features

Features in Overview

Table 1-1 provides an overview of features supported in COS Release 3.20.1.

Table 1-1 Overview of COS Features

Feature Set	Features
Cisco UCS and CDE Server Support	• COS Release 3.20.1 supports a variety of Cisco UCS and CDE server hardware models and configurations. For details, see Supported Hardware, page 1-10. • For UCS S3260 nodes without COS preinstalled, COS Release 3.20.1 provides a script to enable setup of one or two COS nodes on the S3260. • For CDE6032 nodes with COS already installed, COS provides a script to configure the CDE6032 for service after installation. A script is also provided for CDE6032 nodes that do not yet have COS installed.
Rolling Update Service (RUS)	• Along with the introduction of DEC (Server RAID) comes the need for a way to update COS on each node in a DEC cluster maintaining availability to the application data stored in the cluster. RUS software manages the node-by-node update process so that data within the cluster is always available. For complete details, see the *Cisco COS Rolling Update Service User Guide*.
COS Metadata Cluster (CMC)	• COS Release 3.20.1 allows the option of using a separate CMC for the Cassandra database. Moving the metadata from the COS nodes to a separate platform allows metatdata to grow without reducing space available for object data on the COS nodes.
COS Lock Manager (CLM)	• CLM is an internal service that runs on either the CMC node (if using off-box DB), or COS node (if using on-box DB), but not both. This service ensures that metadata operations occur in the proper sequence, thereby preventing incorrect metadata state.
Automated Node Configuration	• A single configuration file for all COS nodes can be stored 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 V2PC GUI. • COS 3.20.1 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) Support	• PXE can be used to download a network bootstrap program (NBP) to remotely install a COS node over a network.

Table 1-1 Overview of COS Features

Feature Set	Features
Improved TCP Transmission	• COS 3.20.1 includes optimizations to improve TCP transmit performance.
Small Object Support	• For cloud DVR and similar applications, COS 3.20.1 provides Small Object Support to efficiently manage storage of many small files representing media segments.
Fanout Compaction	• COS 3.20.1 includes support for compaction of fanout objects to reclaim disk space when copies of a fanout object are deleted before the entire fanout object has been deleted.
V2PC GUI	• Lets you quickly and easily access many COS 3.20.1 deployment, monitoring, and alarm functions. • Displays storage, network bandwidth, session count, and alarms for individual COS disks, nodes, services, and interfaces. • Includes a graphical display of deployment statistics and trends related to disk, service, and interface status. • Supports configuration of COS node service interface from the GUI. • Supports setting of resiliency policies on a per-cluster basis from the GUI.
V2PC GUI Enhancements	• Support for automated COS Node configuration using either Manual IP assignment, or Automated (IP Pool) IP assignment. • Support for enabling and configuring of the TD-Agent for both COS and CMC Clusters. • Support for additional SLA thresholds for setting Warning and Critical alarm thresholds.
COS Node Telemetry Forwarding	• COS 3.20.1 supports the ability to configure forwarding of log events and statistics either an Elasticsearch, AWS, or Kafka destination using the V2PC GUI. This feature is available for both COS Nodes and CMC Nodes.
High Availability (HA)	• COS supports HA as implemented in V2PC, providing redundancy for V2PC VMs. V2PC uses both Cisco and third-party components to support HA.
Swauth API	• Simple Auth Service API for authentication of Swift operations. • Based on Swauth Open-Source Middleware API. • Used to manage accounts, users and account service endpoints.

Table 1-1 Overview of COS Features

Feature Set	Features
Swift Object Store API	• An implementation of a subset of the continually evolving OpenStack Swift API. • Command executions are authenticated using auth tokens provided by Swauth service. • Used to create and manage containers and objects for persistent storage in a COS cluster. • Supports archiving of content from Cisco or ARRIS recorders using DataDirect Networks (DDN) Web Object Scaler (WOS) archive objects.
Fanout API	• COS 3.20.1 includes support for a Fanout API to enable interactions with other Cisco and third-party applications orchestrated through V2PC.
Object Store Metadata Resiliency	• Metadata resiliency is provided by a distributed and replicated Cassandra document database that optionally can be stored on (or migrated to) a separate CMC. • Manual administrative intervention is required on CMC node failure.
Object Store Data Resiliency	• Data is resilient to both hard drive and COS node failures. • Local Erasure Coding (LEC), or local COS node data resiliency, is provided by local software RAID. LEC is enabled by default and is configured for two drive failures. We recommend this configuration for resiliency. • Distributed erasure coding (DEC) provides data resiliency across nodes, protecting stored content from loss due to node failure. • COS also supports mirroring of local hard drives as an option. However, LEC is enabled by default, and is the generally recommended choice. **Note** When configuring local mirroring for resiliency, we recommend using no more than one local mirror copy. • Supports configuration of mixed resiliency policies (local erasure with remote mirroring) via the GUI. • Alarms are available for loss of storage.

Table 1-1 Overview of COS Features

Feature Set	Features
Resiliency Groups	• COS now defines each COS node as a member of a *resiliency group*, a subdivision of a cluster that provides striping for distributed erasure coding (DEC). Resiliency groups reduce communication overhead between servers by keeping traffic within a resiliency group. Once defined, resiliency groups are maintained as part of cluster and site management, and are transparent to the operator. Each cluster has an associated Cassandra cluster, and a target resiliency that can be met using DEC or mirroring.
Management Interface Bonding	• Supports defining two node management interface ports as a primary-backup pair.
Service Load Balancing	• COS cluster load balancing is provided by DNS round-robin of a FQDN to multiple physical IPv4 addresses hosted by COS nodes. • Optimal load balancing is provided by extensions to the Swift API through the implementation of HTTP redirect. • Remote smoothing facilitates load balancing by moving content to a new node when it is added to a cluster.
VMR-to-COS Load Balancing	• VMR to COS IP discovery and load balancing enables rapid updating of the service directory catalog to reflect changes to VMR-COS interface availability.
Ingest Load Balancing	• Within Ingest Load Balancing allows for a redirect on the original write request. If a redirect occurs then the second request by the caller would be to the new IP address that was returned in the response to the original request.
COS Cluster Support	• Each COS application instance can have one or more clusters created to service that application instance. • Each cluster can have its own asset redundancy policy, shared by all COS nodes that are members of that cluster. • If a cluster is disabled, all member COS nodes will have their interfaces removed from the DNS. Likewise, when a cluster is enabled, all member node interfaces will be added back to the DNS.

Table 1-1 Overview of COS Features

Feature Set	Features
COS AIC Client Management	• The COS application instance controller (AIC) Client process is monitored by the monit process that runs on each COS node, and if not running, is restarted. • The COS AIC Client process creates a PID file that is added to the monit script so it can be monitored and restarted. • Command-line scripts support stopping and restarting the AIC Client process manually, bypassing the normal automatic restart process.
Node Decommissioning Paused for Maintenance Mode	• If a node is in the process of being decommissioned and any node in its cluster is placed in Maintenance mode, the decommissioning process is paused.

Supported Hardware

Table 1-2 lists the Cisco hardware models that fully support installation of COS Release 3.20.1.

Note Note: COS Release 3.20.1 has been tested on the hardware models and firmware listed in Table 1-2. It is recommended that you use the specified firmware with the hardware models listed.

Table 1-2 COS 3.20.1 Supported Hardware

Product Name	Storage Bundle	Configuration Supported	Max HDD Capacity	Max Total Storage	SSDs Used by OS and COS	Intel Xeon CPU	Firmware
Cisco CDE6032 (UCS C3K)	4U6	Single Node	56 x 10 TB	560 TB	2 x 480 GB	E5-2695 v4	HUU 2.0(13e)
Cisco CDE6032 (UCS C3K)	4U5	Dual Node	56 x 10 TB	560 TB	4 x 480 GB	E5-2695 v4	HUU 2.0(13e)
Cisco CDE6032 (UCS C3K)	4U3	Dual Node	56 x 6 TB	336 TB	4 x 480 GB	E5-2620 v4	HUU 2.0(13e)
Cisco CDE6032 (UCS C3K)	4U5	Dual Node	56 x 10 TB	560 TB	4 x 480 GB	E5-2620 v4	HUU 2.0(13e), 3.0(1c)
Cisco CDE6032 (UCS C3K-IOE)	4U8	Single Node	56 x 8 TB	560 TB	2 x 480 GB	E5-2680 v4	HUU 3.0(3a)
Cisco UCS S3260	4U6	Single Node	56 x 10 TB	560 TB	2 x 480 GB	E5-2695 v2	HUU 2.0(13e)
Cisco UCS S3260	4U5	Dual Node	56 x 10 TB	560 TB	4 x 480 GB	E5-2695 v2	HUU 2.0(13e)

Table 1-2 COS 3.20.1 Supported Hardware

Product Name	Storage Bundle	Configuration Supported	Max HDD Capacity	Max Total Storage	SSDs Used by OS and COS	Intel Xeon CPU	Firmware
Cisco UCS S3260	4U4	Single Node	56 x 6 TB	336 TB	2 x 480 GB	E5-2695 v2	HUU 2.0(13e)
Cisco UCS S3260	4U3	Dual Node	56 x 6 TB	336 TB	4 x 480 GB	E5-2695 v2	HUU 2.0(13e)
Cisco UCS S3260	4U3	Single Node	28 x 6 TB	168 TB	2 x 480 GB	E5-2695 v2	HUU 2.0(13e)
Cisco UCS C3160	4U2	Single Node	54 x 6 TB	324 TB	2 x 400 GB	E5-2695 v2	HUU 2.0(6d)
Cisco UCS C3160	4U1	Single Node	54 x 4 TB	216 TB	2 x 400 GB	E5-2695 v2	HUU 2.0(6d)
Cisco CDE465	4R4	Single Node	36 x 6 TB	216 TB	2 x 480 GB	E5-2670 v3	1.0CDEd

For information about installing the hardware, see the following:

- *Cisco UCS S3260 Storage Server Installation and Service Guide*
- *Cisco UCS C3160 Rack Server Installation and Service Guide*
- *Cisco Content Delivery Engine 465 Hardware Installation Guide*

Notes for Cisco UCS C3k-IOE Storage Server

COS 3.16.1 supports the Cisco UCSC-C3K-IOE-4U8 platform, which supports up to 56 storage disks per chassis. The UCSC-C3K-IOE-4U8 is a 4RU chassis that supports a single-node server configuration.

- 2 x 480 GB solid-state drives (SSDs) for operating system and COS installation
- 56 x 8TB hard drives for content storage
- One server node with 12 x 64 GB RAM, providing 768 GB
- 1 system I/O controller with 2 x 40 Gbps QSFP ports

A pre-installation script "preinst_setup_UCSC-C3KIOE_2x10.sh" is provided to properly configure this chassis. You must run this script before installing COS on this server node.

On this COS node, the two 10Gb ports on the IOE, eth0 and eth1, are usually bonded to a bond0 management interface.

Notes for Cisco UCS S3260 Storage Server

COS 3.20.1 supports the Cisco UCS S3260 platform, which supports up to two compute nodes and up to 56 storage disks per chassis.

The UCS S3260 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 16 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 16 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

Note The CDE6032 is a special edition of the UCS S3260 configured specifically for COS and other V2P applications. The CDE6032 comes with COS 3.20.1 preloaded on a boot drive consisting of 4 x 480 GB SSDs in hardware RAID, and ships with the maximum storage currently available (560 TB total).

A pre-installation script is provided to properly configure the S3260 chassis for either single-node or dual-node service. You must run this script before installing COS on any S3260 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 S3260-4U3 (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 S3260-4U4 (56 disks per server node): Select this bundle if you configured a single COS node with 56 x 6 TB hard drives.
- UCS S3260-4U5 (28 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.
- UCS S3260-4U6 (56 disks per server node): Select this bundle if you configured a single COS node setup with 56 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 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.

CMC Supported Hardware

The following CMC hardware is supported in Release 3.14.1 and later.

Product Name	Max DB Capacity	Maximum Total Storage	SSDs Used by OS and CMC	Intel Xeaon CPU
Cisco UCS C220M4 (CMC1)	6 x 1.6TB – Raid10	2.4 TB	2 x 1.6TB – \ Raid1	E5-2695v3
Cisco UCS C220M4* (CMC2)	5 x 480GB – JBOD	2.4 TB	2 x 120GB – \ Raid1	E5-2630v4

* The UCS C220M4 CMC systems - The Cisco UCS C220 M4 Rack Server is the most versatile, high-density, general-purpose enterprise infrastructure and application server in the industry today. It delivers world-record performance for a wide range of enterprise workloads, including virtualization, collaboration, and bare-metal applications.

The C220 M4 (CMC1) is a 1RU server. When configured for COS CMC, the CMC1 includes the following:

- 2 x 1.6TB solid-state drives (SSDs) in RAID1, for operating system and CMC installation.
- 6 x 1.6TB solid-state drives (SSDs) in RAID10, 4.8TB total storage for Database.

- One system I/O controller providing a total of two 10 GbE ports.

The C220 M4 (CMC2) is a 1RU server. When configured for COS CMC, the CMC2 includes the following:

- 2 x 120GB solid-state drives (SSDs) in RAID1, for operating system and CMC \installation.
- 5 x 480GB solid-state drives (SSDs) in JBOD, 2.4TB total storage for Database.
- One system I/O controller providing a total of two 10 GbE ports.

Rolling Update Service

The introduction of DEC (Server RAID) makes it desirable to be able to update COS on each node in a DEC cluster while maintaining availability to the application data stored on the cluster.

Cisco COS Rolling Update Service (RUS) software, an optional companion product to COS, is designed for this purpose. RUS manages the node-by-node update process so that data within the cluster is always available. For details, see the *Cisco COS Rolling Update Service User Guide*.

COS Metadata Cluster

COS Release 3.20.1 allows the option of using a separate COS Metadata Cluster (CMC) for the Cassandra database. Moving the metadata from the COS nodes to a separate platform allows metatdata to grow without reducing space available for object data on the COS nodes. For details, see Create the COS Metadata Cluster, page 2-14.

COS Lock Manager

COS Lock Manager (CLM) is an internal COS service that runs on either CMC node (if using off-box DB), or COS node (if using on-box DB), but not both. Its purpose is to ensure that certain COS metadata operations occur in the correct sequence across various COS nodes so as to prevent incorrect metadata states.

Note CLM requires a minimum of three CMC nodes for operation.

The daemon that provides the locking service executes in one of three states: standby, backup, or primary. Only one daemon instance runs in a backup state, and one instance runs in a primary state. All COS data nodes communicate as clients with the primary deamon. The primary daemon replicates the lock state to the backup daemon, which will become primary if the primary daemon is stopped.

The CLM service is automatically configured and started, and COS data nodes automatically discover and communicate with the CLM primary instance. The COS data nodes communicate with the CLM primary instance via TCP on port 5001, while the CLM primary and backup instances communicate via TCP on port 5002. These ports open automatically in the firewall configuration on CMC nodes.

CLM normally requires no intervention by the user. In the event of problem symptoms such as logged lock timeout errors or failed HTTP client requests, see the *Cisco Cloud Object Storage Release 3.14.1 Troubleshooting Guide* for assistance.

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 GUI or the API. A single configuration file for all COS nodes (or node sets) can be stored on an HTTP server for download by cosinit.

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. See the *Cisco Cloud Object Storage Release 3.8.1 User Guide* for details.

Beginning with COS 3.16.1, the COS Node Profile page of the V2PC GUI can be used to configure a profile template for automated configuration of COS Nodes. 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

Intel Preboot Execution Environment Support

The Intel Preboot Execution Environment (PXE) can be used to download a network bootstrap program (NBP) to remotely install a COS node over a network.

Improved TCP Transmission

COS 3.20.1 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 uses small object support to enable more efficient storage of multiple small files. This technique maps multiple small files to a larger virtual container file which, by virtue of its larger size, makes more efficient use of a disk allocation unit. In this context, the small files are called *small objects* and the container file is called a *container object*, Each small object can be up to 32 MB in size. Each container object can be up to 256 MB in size, and can hold up to 64K (65535) small objects.

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.

Fanout Compaction

In COS Release 3.8.1, as individual copies of a fanout object were deleted, the space allocated to the deleted copies did not become available for reuse until the entire object (that is, all of its copies) were deleted. In solutions that do not guarantee deletion of the entire fanout object within a relatively short period of time, significant unusable disk space could then accumulate, reducing overall storage capacity.

COS 3.20.1 introduces *fanout compaction*, a feature that reclaims the space allocated to deleted copies for use by other object contents before the entire fanout object has been deleted.

Note Enabling this feature requires changes in the way that data is represented on disk. For this reason, all nodes in a cluster in which it is enabled must be running a supporting COS release (currently only 3.20.1) to enable fanout compaction for the cluster.

V2PC GUI

The V2PC GUI enables quick and easy configuration of the COS infrastructure, service domain objects, and services. The GUI also provides valuable monitoring functions, including graphical displays of Node status and alarms and alarm history for individual COS disks, nodes, services, and interfaces. In addition, the GUI displays system and service diagnostics, as well as event logs and log analysis.

The GUI also supports configuration of the COS node service interfaces and management interfaces using Manual IP assignment, in addition to Automated (IP Pool) IP assignment. This feature is part of the Node Profiles page and can be used in conjunction with either the PXE or YAML based automated Node initialization methods.

The V2PC GUI has been enhanced to support additional adjustable SLA reporting thresholds for various Node partition usage levels. These new thresholds determine the level of Warning and Critical Error messages appearing in the V2PC GUI alarms page.

The COS Cluster and COS Metadata Cluster pages have been enhanced to include enabling and configuring the TD-Agent.

For information on using the V2PC GUI to manage COS, see System Monitoring, page 3-1 and Using the V2PC GUI, page A-2.

COS Node Telemetry Forwarding

COS 3.18.1 supports the ability to enable and configure forwarding of log events and statistical information to either an Elasticsearch, AWS, or Kafka destination for both COS and CMC Clusters.

When the administrator chooses to enable the TD-agent for a Cluster (COS or CMC) the GUI will prompt the user for the required information based on the user selected destination type of either Elasticsearch, AWS, or Kafka. For a new configuration, the users input will be applied to the appropriate destination "default" template and a Cluster specific TD-Agent configuration will be saved to the docserver. For existing configurations, the users input will be applied to the existing Cluster specific TD-Agent and stored in the docserver. The current set of information that is forwarded contains:

- A subset of the events from /arroyo/log/http.log.<DATE> and /arroyo/log/cosd.log.<DATE>
- A subset of the statistics from /arroyo/log/protocoltiming.log.<DATE>
- Statistics from /proc/calypso/stats/*_stats

For additional details and instructions for implementing this feature, see Deploying COS, page 2-1.

High Availability

V2PC has two classes of components for High Availability (HA):

- Third-party components such as Consul, MongoDB, and Redis use their own proprietary clustering and redundancy schemes.
- Cisco components, such as the V2PC GUI and DocServer, use ZooKeeper for leader election.

In an HA environment, multiple VMs provide redundancy for the applications. HA requires three VMs because applications such as Consul, 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 a configuration of three V2PC Master 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 3.16.1 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 1-3 COS API Restrictions

Constraint	Value
Maximum # of HTTP Headers per request	90
Maximum length of all HTTP Headers	4096 bytes
Maximum length per HTTP request line	8192 bytes
Maximum length of container name	256 bytes
Maximum length of object name	1024 bytes

Note 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:

- Listing Containers
- Listing Objects
- Creating a Container
- Deleting a Container
- Retrieving an Object
- Retrieving an Archive (DDN WOS) Object
- Creating or Updating an Object
- Creating Unique Object Copies with Variants
- Reserving an Archive (DDN WOS) Object Identifier
- Creating an Archive (DDN WOS) Object
- Deleting an Object

- Deleting an Archive (DDN WOS) Object
- Creating or Updating Container Metadata
- Retrieving Container Metadata
- Deleting Container Metadata
- Retrieving Object Metadata
- Discovering COS Cluster Features

For details, see the *Cisco Cloud Object Storage Release 3.20.1 API Guide*.

Fanout API

COS 3.20.1 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 applications dedicated to different parts of the workflow such as ingest, recording, and storage.

Note 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 3.20.1 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 3.16.1 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

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 3.20.1 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 3.20.1 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 date 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 V2PC 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 3.20.1 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 3.20.1 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.

COS now defines each COS node as a member of a *resiliency group*, a subdivision of a cluster that provides striping for distributed erasure coding (DEC). Resiliency groups reduce communication overhead between servers by keeping traffic within a resiliency group. Once defined, resiliency groups are maintained as part of cluster and site management, and are transparent to the operator. Each cluster has an associated Cassandra cluster, and a target resiliency that can be met using DEC or mirroring.

For additional details, see Configuring Resiliency and Resiliency Groups, page D-1.

Management Interface Port Bonding

Extending resiliency to the network management interface, COS 3.20.1 also supports defining two node ports as a primary-backup pair for management interface bonding. The following table provides a list of designated ports for each device.

Device	Designated Ports
C3K-IOE	eth0 and eth1
CDE465	eth0 and eth1
S3160	eth0 and eth3
S3260	eth0 and eth1

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 3.16.1 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.

Beginning with COS 3.14.1, VMR to COS IP discovery and load balancing enables rapid updating of the service directory catalog to reflect changes to VMR-COS interface availability.

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 E-1.

COS Cluster Support

Each COS application instance can have one or more Clusters created to service that application instance. Each cluster can have its own asset redundancy policy, shared by all COS nodes that are members of that cluster.

If a cluster is disabled, all member COS nodes will have their interfaces removed from the DNS. Likewise, when a cluster is enabled, all member node interfaces will be added back to the DNS.

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 Virtualized Video Processing Controller User Guide* for your V2PC 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 that are 3.12.x and earlier do not support downgrade to any earlier COS release if fanout compaction has been enabled on the node to be downgraded. See Deploying COS, page 2-1 for details.
- There is no COS post-installation script support for C3K-IOE systems with COS 3.16.1 pre-installed.
- See the *Release Notes for COS 3.20.1* for open caveats and other known issues related to this release.