Overview

Product Description

The Cisco Cloud Object Storage (COS) provides distributed, resilient, high-performance storage and retrieval of binary large object (blob) data. The primary interface for managing COS content is the OpenStack Swift API, with enhancements that improve the quality of service when accessing large media objects.

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 the cluster. Nodes may be added to or removed from the cluster as needed to provide for changes in cluster capacity. To administer the cluster, COS includes an HTTP-based cluster-management API.

COS also includes an authentication and authorization service that implements the OpenStack Swauth API.

COS and MOS

COS is designed to integrate transparently with the Cisco Media Origination System (MOS), which is designed for highly optimized ingest and storage. MOS uses a hierarchical storage design that supports huge content libraries while simplifying content storage management. Its distributed architecture can separate ingest and storage from streaming, allowing each function to be scaled independently as needed to dynamically increase network ingest and storage resources. For more information, see the *Cisco Media Origination System Release 2.3 User Guide*.

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 Platform and Applications Management system (PAM).

- Hardware Platforms — COS software is deployed on Cisco Content Delivery Engine (CDE) platforms.

Each of these components and subsystems are described further in the following sections.

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. The client applications use 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. When routing COS data network traffic, however, it must be ensured that high-volume data traffic does not congest networks used to manage COS and other systems.

For this reason, 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. One or two lower-bandwidth adapters are usually configured for use on 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 – It facilitates setting up of networks and configuration of external DNS servers, and 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 PAM and its components, see the *Cisco Media Origination System Release 2.3 User Guide*.

COS Nodes

The COS software runs on a collection of computing systems, called nodes, that are connected via the management and data networks. Currently, there are two types of COS nodes: the cluster controller that 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 may 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 number of daemons in the Linux user-space. These include daemons to coordinate service log files, to communicate with the cluster controller, to provide a distributed database for object-store metadata, and to communicate with the modules running in the kernel.

Kernel modules are used to provide real-time management of node hardware resources, to provide the distributed, resilient content-store used for object-store data, and to provide the Swift and Swauth API support via the data network. While the management-network interfaces communicate directly with the user-space daemons, the data-network interfaces communicate directly with the kernel modules.

COS Cluster

COS services are provided by software running on a set of nodes called a *COS cluster* that is connected by data and management networks. Each COS 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's 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, and the IP address of the PAM configuration document server. For a detailed description of the configuration parameters, see Chapter 2, “Deploying COS”.

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

COS software can be deployed on the Cisco Content Delivery Engine (CDE) platforms CDE460 and CDE470. For information about the CDE460 and CDE470 platforms, see the *Cisco Content Delivery Engine 205/220/250/420/460/470 Hardware Installation Guide.*

Features

- CLI Utilities, page 1-8

Overview

The table below provides an overview of the COS features.

Table 1-1 Overview of COS Features

Feature Set	Features
COS Service Manager GUI	• Lets you quickly and easily access many COS deployment and monitoring functions
High Availability (HA)	• COS now supports HA as implemented in MOS
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
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
Object Store Metadata Resiliency	• Metadata resiliency is provided by a distributed and replicated Cassandra document database • Each COS node participates in the persistence of a subset of the Cassandra database • Manual administrative intervention required upon node failure
Object Store Data Resiliency	• Data is resilient to both hard drive and COS node failures • Local COS node data resiliency provided by local software RAID • COS cluster data resiliency provided by object replication
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

COS Service Manager GUI

The COS Service Manager GUI enables you to quickly and easily configure the COS infrastructure, service domain objects, and services. The GUI also provides valuable monitoring functions, including alarm and alarm history display, system and service diagnostics, event display, log analysis and display, and configuration display.

For information on launching an navigating the GUI, the COS Service Manager GUI section of Appendix A, "Reference Information".

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 that you configure more than three PAM VMs to ensure recovery in the event of multiple failures and to support high performance, especially the sharing of 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 a detailed description of these functions, see the *Cisco Cloud Object Storage Release 2.1.1 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 (ReSTful) 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 that is part of the response to a successful request for an authentication token.

The COS object storage API defines restrictions on HTTP requests. These restrictions, borrowed from the Swift API, are listed in the table below.

Table 1-2 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

Also, the container and object names must be UTF-8 encoded and then URL-encoded before inclusion in the HTTP request line.

Note 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
- Creating or Updating An Object
- Deleting an Object
- Creating or Updating Container Metadata
- Retrieving Container Metadata
- Deleting Container Metadata
- Retrieving Object Metadata

For a detailed description of these functions, see the *Cisco Cloud Object Storage Release 2.1.1 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) to local disk a copy of the entire database. 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.

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

Object Data Resiliency

The COS stores Swift object data to local disk or to external facing hard drives. To maintain data resiliency in the event of a loss of a local hard drive, the COS system, by default, uses erasure coding techniques (software RAID) to stripe the object data across multiple local drives. The default scheme for erasure coding is to maintain two parity blocks for twelve data blocks. This ensures that data is resilient to the loss of two local hard drives. In the event of a hard drive loss, the COS system immediately begins to recover the data from the lost drive and replace the missing data blocks on surviving hard drives. The execution of this recovery process, including the duration, is scheduled with low priority, and depends on the availability of system resources and available disk space.

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 administrative action. As soon as the COS cluster detects a node failure, the cluster begins to create additional copies of objects that were maintained on the failed node. Upon restoring the failed COS node, the COS cluster purges unnecessary copies to recover disk space.

Service Load Balancing

The COS cluster is composed of COS nodes. Each node has limited CPU, network, and disk resources. To ensure the best performance and quality of service, the workloads of the Swift and Swauth operations must be distributed among the COS 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 COS nodes. While not a perfect solution, such a DNS round-robin should provide sufficient distribution of workloads across the COS nodes.

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 should be 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 allows 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, and the COS node does not have sufficient resources available to satisfy the client's request. If a client does not request a committed rate transfer, the COS node will attempt 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 2.1.1 API Guide*.

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 the COS Command Line Utilities section of Appendix A, "Reference Information".

Prerequisites

The COS management and configuration operations require specific hardware components for deployment. For more information on the hardware requirements, see the MOS 2.3 User Guide.

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

- The COS does not support IPv6.
- The OpenStack Swift and Swauth APIs continue to evolve. COS does not currently implement all the Swift or Swauth API functions. For a list of 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.
- See the *Release Notes for Cisco Cloud Object Storage Release 2.1.1* for open caveats and known issues related to this release of the COS software.