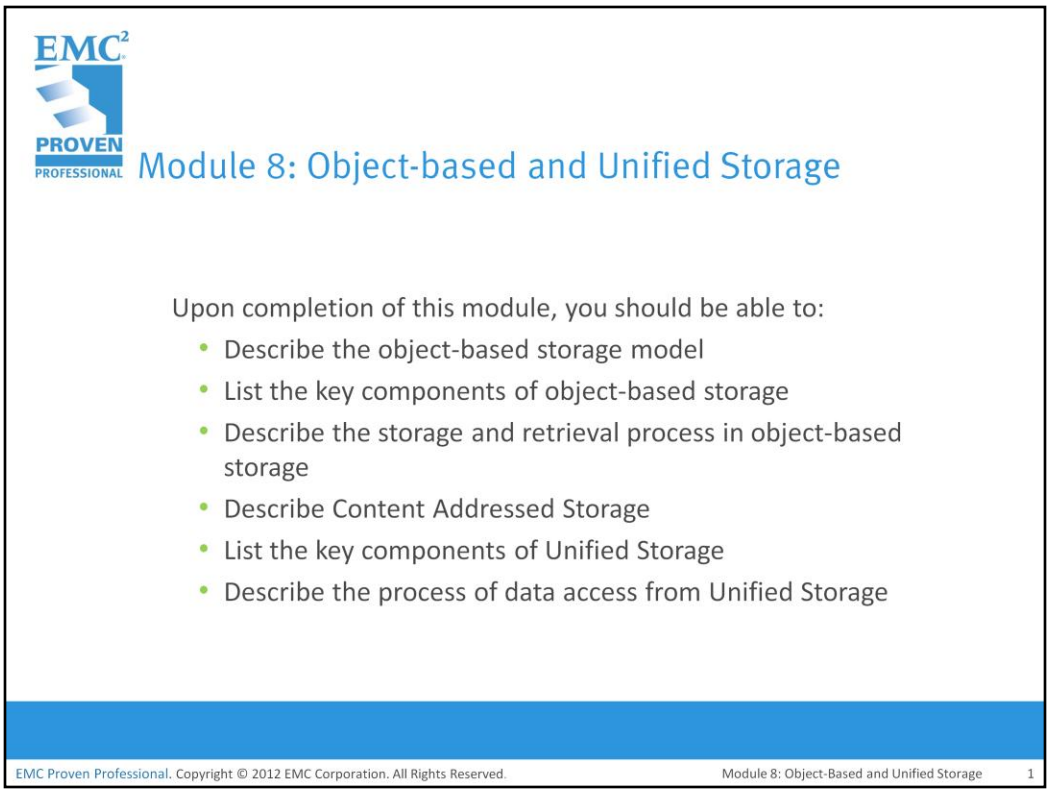
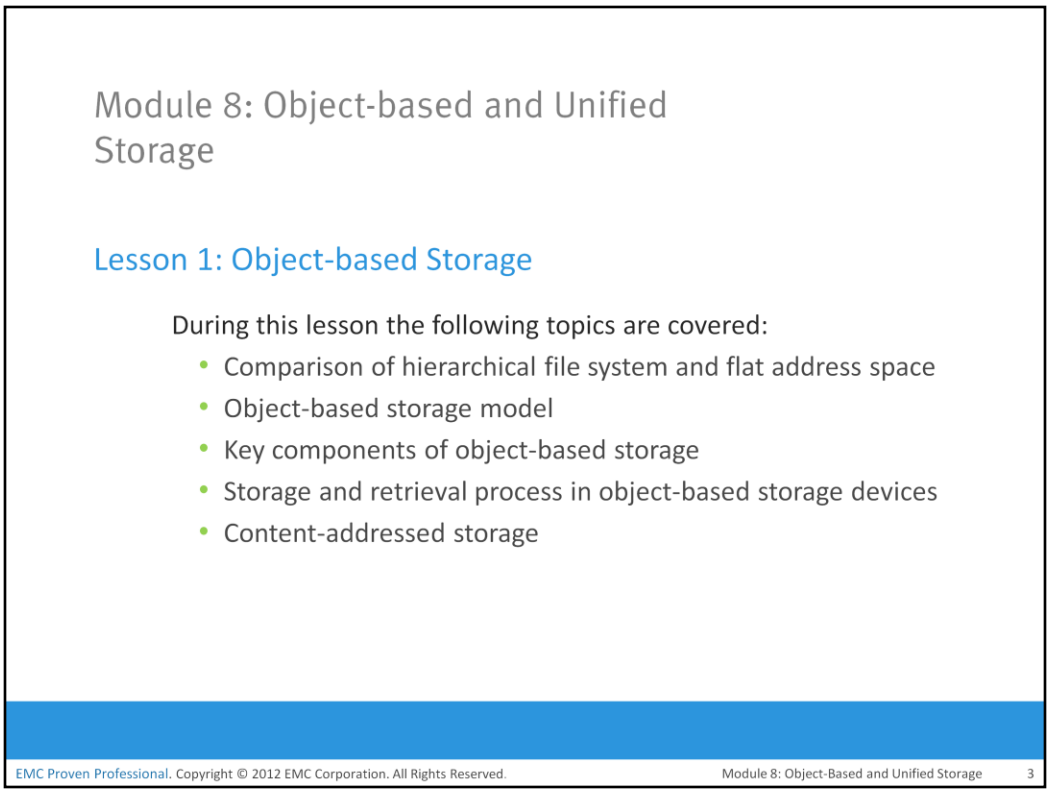
Module – 8

Object-based and Unified Storage

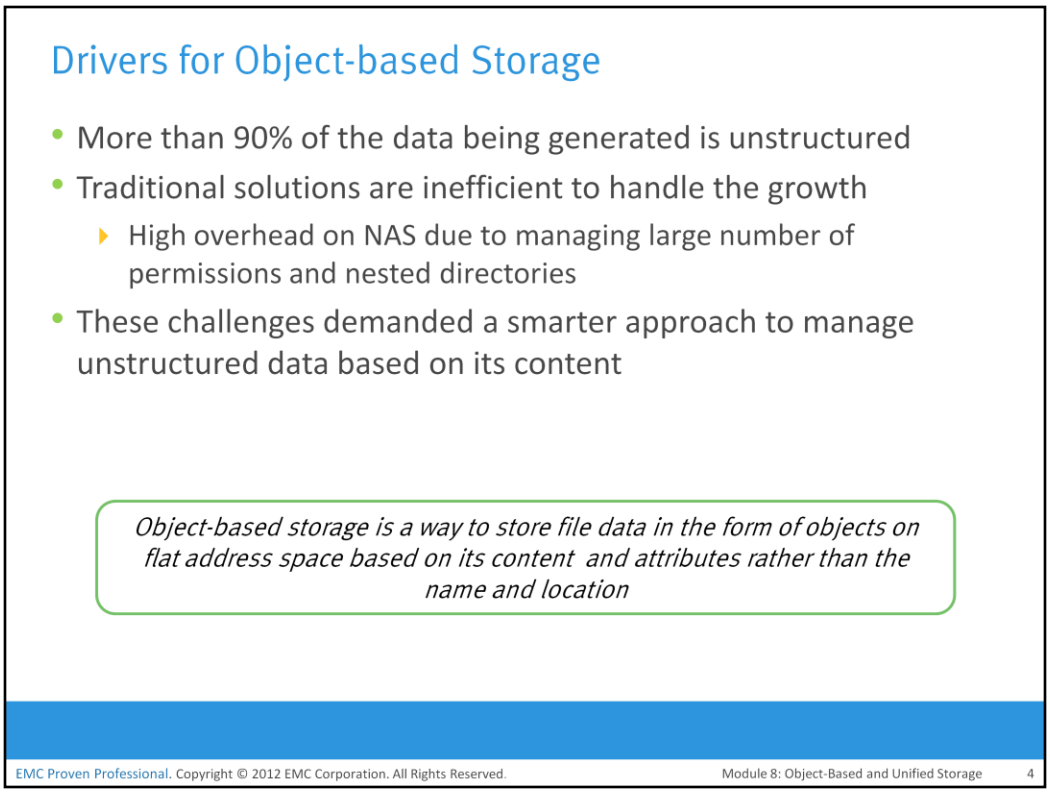


This module focuses on the object-based storage device and the Unified Storage device. It covers the object-based storage model. The module details on the function of the key components and process of storage and retrieval of object-based storage. This module also covers Content Addressed Storage, which is a special type of object-based storage and key features it supports.

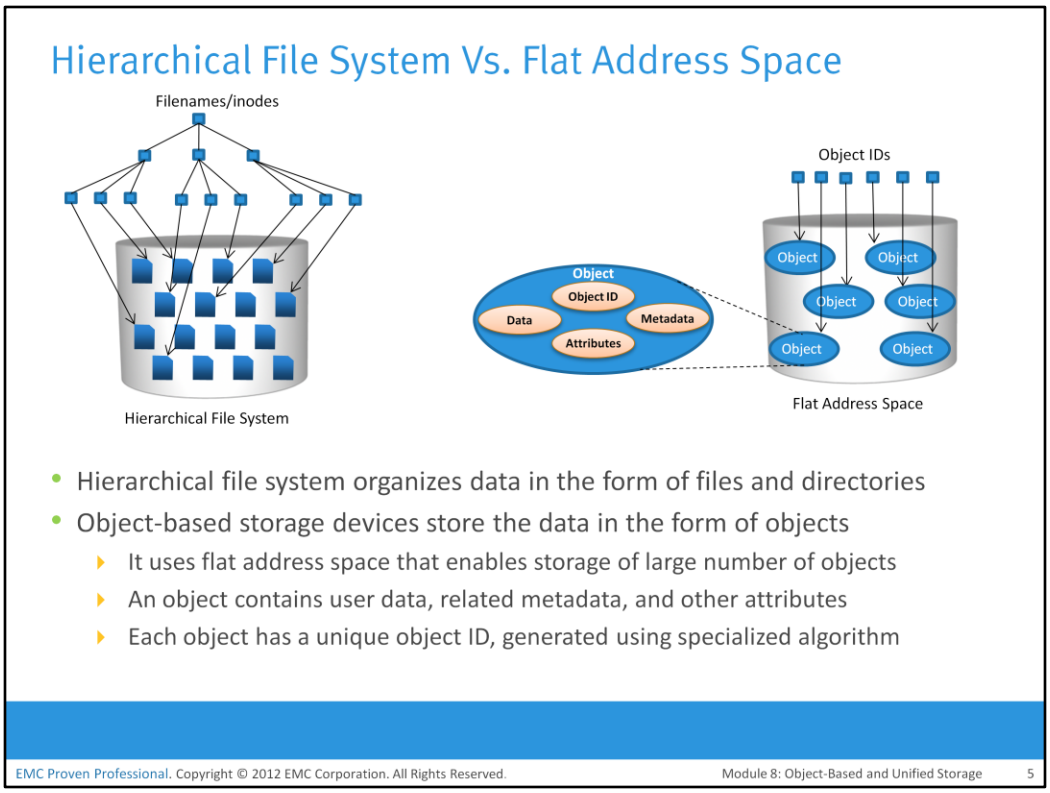
This module also focus on the function of the key components of Unified Storage. Finally, this module covers the process of data access from Unified Storage.



This lesson covers a comparison of hierarchical file system with flat address space. The lesson describes the model and key components of object-based storage. This lesson also covers storage and retrieval process in the object-based storage. Finally, the lesson covers content-addressed storage.

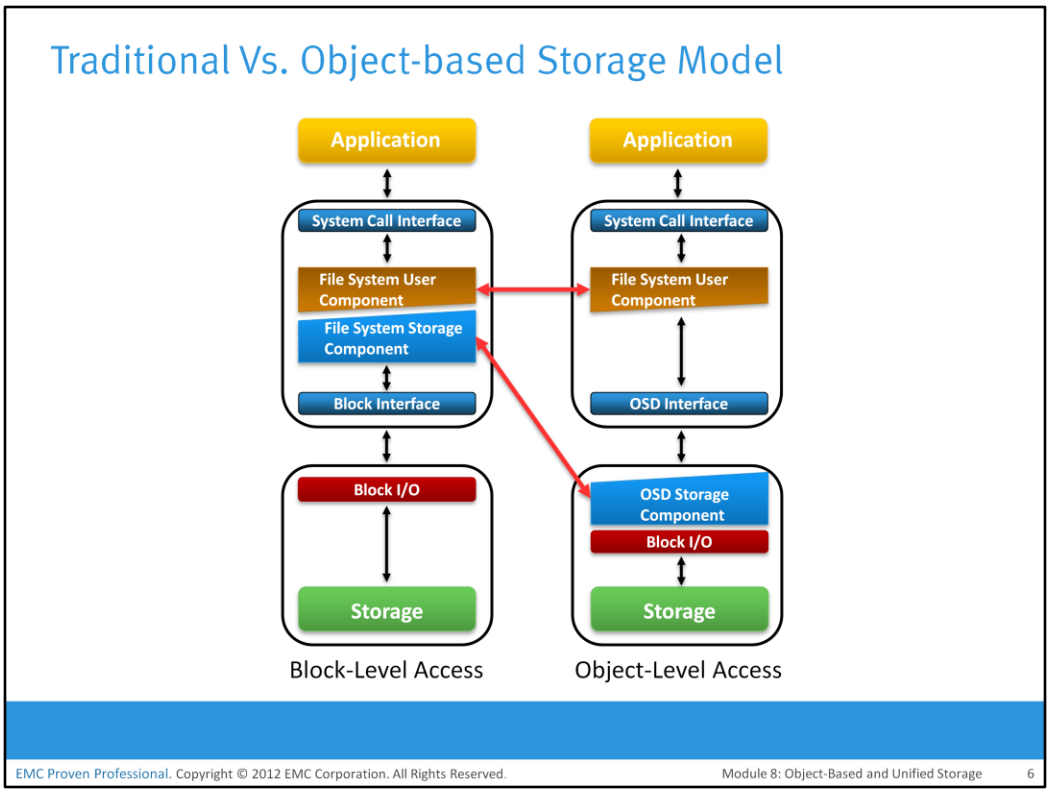


Recent studies have shown that more than 90 percent of data generated is unstructured. This growth of unstructured data has posed new challenges to IT administrators and storage managers. With this growth, traditional NAS, which is a dominant solution for storing unstructured data, has become inefficient. Data growth adds high overhead to the network-attached storage (NAS) in terms of managing large number of permission and nested directories. In an enterprise environment, NAS also manages large amounts of metadata generated by hosts, storage systems, and individual applications. Typically this metadata is stored as part of the file and distributed throughout the environment. This adds to the complexity and latency in searching and retrieving files. These challenges demand a smarter approach to manage unstructured data based on its content rather than metadata about its name, location, and so on. *Object-based storage* is a way to store file data in the form of objects based on its content and other attributes rather than the name and location.



An OSD is a device that organizes and stores unstructured data, such as movies, office documents, and graphics, as objects. Object-based storage provides a scalable, self-managed, protected, and shared storage option. OSD stores data in the form of *objects*. OSD uses flat address space to store data. Therefore, there is no hierarchy of directories and files; as a result, a large number of objects can be stored in an OSD system .

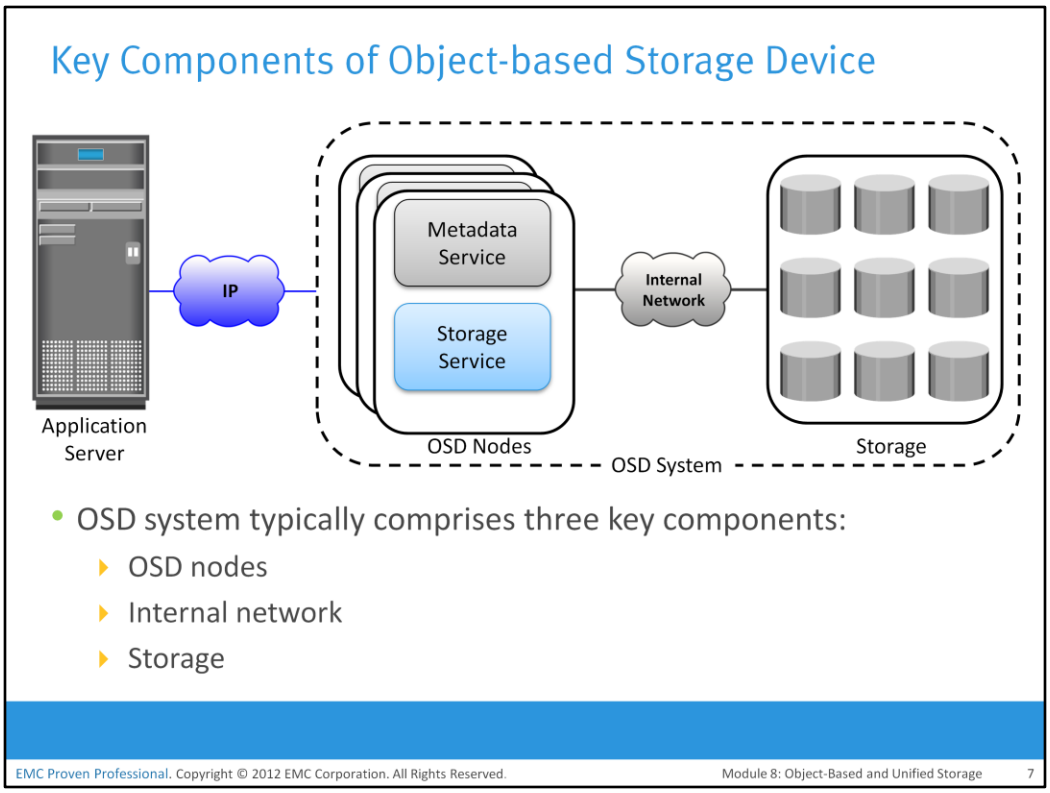
An object might contain user data, related metadata (size, date, ownership, and so on), and other attributes of data (retention, access pattern, and so on). Each object stored in the system is identified by a unique ID called the *object ID*. The object ID is generated using specialized algorithms such as hash function on the data and guarantees that every object is uniquely identified.



An I/O in the traditional block access method passes through various layers in the I/O path. The I/O generated by an application passes through the file system, the channel, or network and reaches the disk drive. When the file system receives the I/O from an application, the file system maps the incoming I/O to the disk blocks. The block interface is used for sending the I/O over the channel or network to the storage device. The I/O is then written to the block allocated on the disk drive.

The file system has two components: user component and storage component. The user component of the file system performs functions such as hierarchy management, naming, and user access control. The storage component maps the files to the physical location on the disk drive.

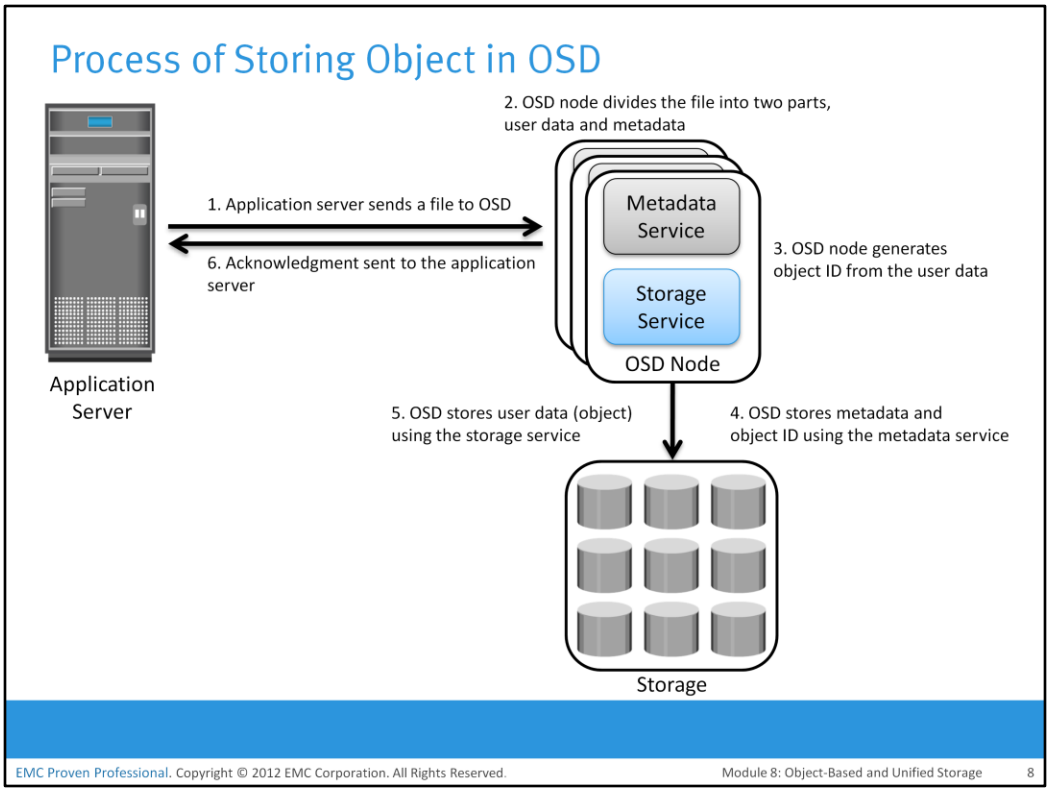
When an application accesses data stored in OSD, the request is sent to the file system user component. The file system user component communicates to the OSD interface, which in turn sends the request to the storage device. The storage device has the OSD storage component responsible for managing the access to the object on a storage device. After the object is stored, the OSD sends an acknowledgment to the application server. The OSD storage component manages all the required low-level storage and space management functions. It also manages security and access control functions for the objects.



The OSD system is typically composed of three key components: nodes, private network, and storage.

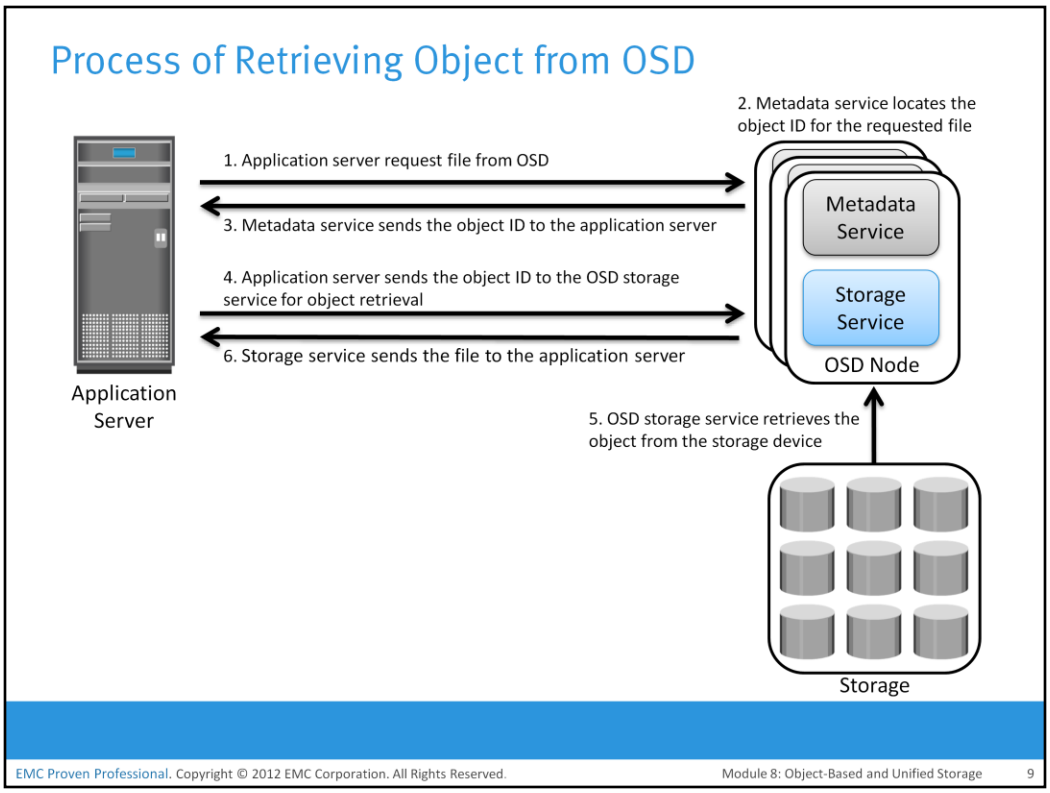
The OSD system is composed of one or more *nodes*. A node is a server that runs the OSD operating environment and provides services to store, retrieve, and manage data in the system. The OSD node has two key services: metadata service and storage service. The metadata service is responsible for generating the object ID from the contents (may also include other attributes of data) of a file. It also maintains the mapping of the object IDs and the file system namespace. The storage service manages a set of disks on which the user data is stored. The OSD nodes connect to the storage via an internal network. The internal network provides node-to-node connectivity and node-to-storage connectivity. The application server accesses the node to store and retrieve data over an external network. In some implementations, such as CAS, the metadata service might reside on the application server or on a separate server.

OSD typically uses low-cost and high-density disk drives to store the objects. As more capacity is required, more disk drives can be added to the system.



The process of storing objects in OSD is illustrated on the slide. The data storage process in an OSD system is as follows:

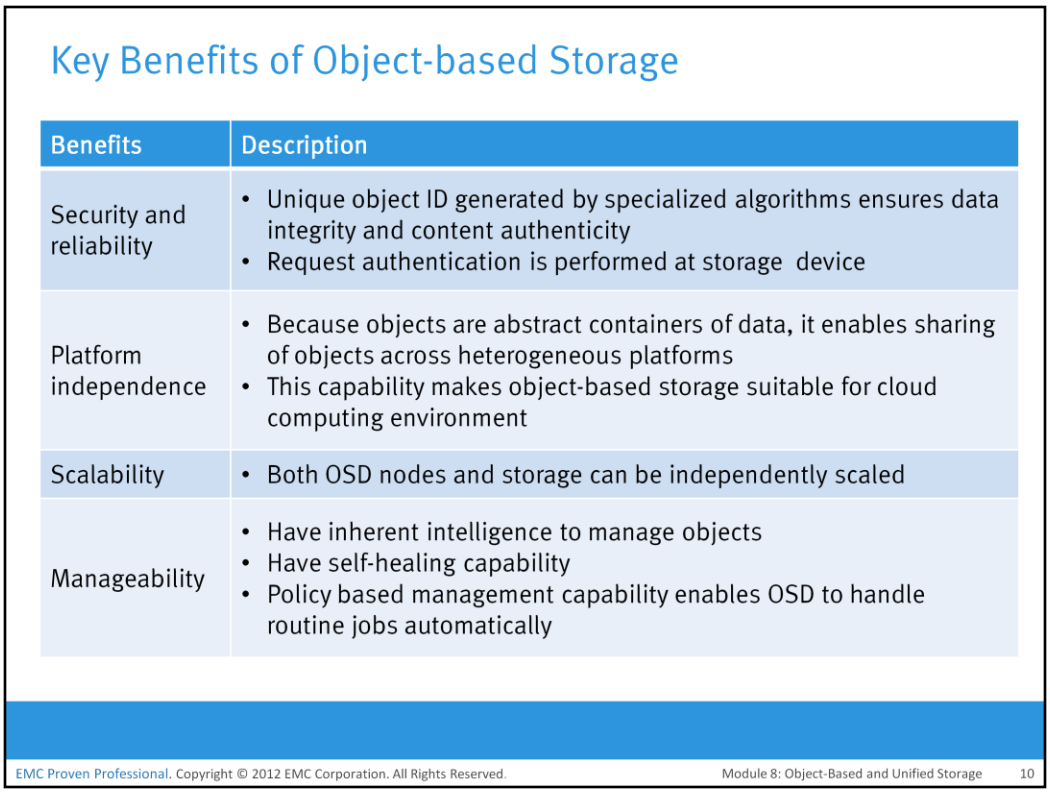
1. The application server presents the file to be stored to the OSD node.
2. The OSD node divides the file into two parts: user data and metadata.
3. The OSD node generates the object ID using a specialized algorithm. The algorithm is executed against the contents of the user data to derive an ID unique to this data.
4. For future access, the OSD node stores the metadata and object ID using the metadata service.
5. The OSD node stores the user data (objects) in the storage device using the storage service.
6. An acknowledgment is sent to the application server stating that the object is stored.



After an object is stored successfully, it is available for retrieval. A user accesses the data stored on OSD by the same filename. The application server retrieves the stored content using the object ID. This process is transparent to the user.

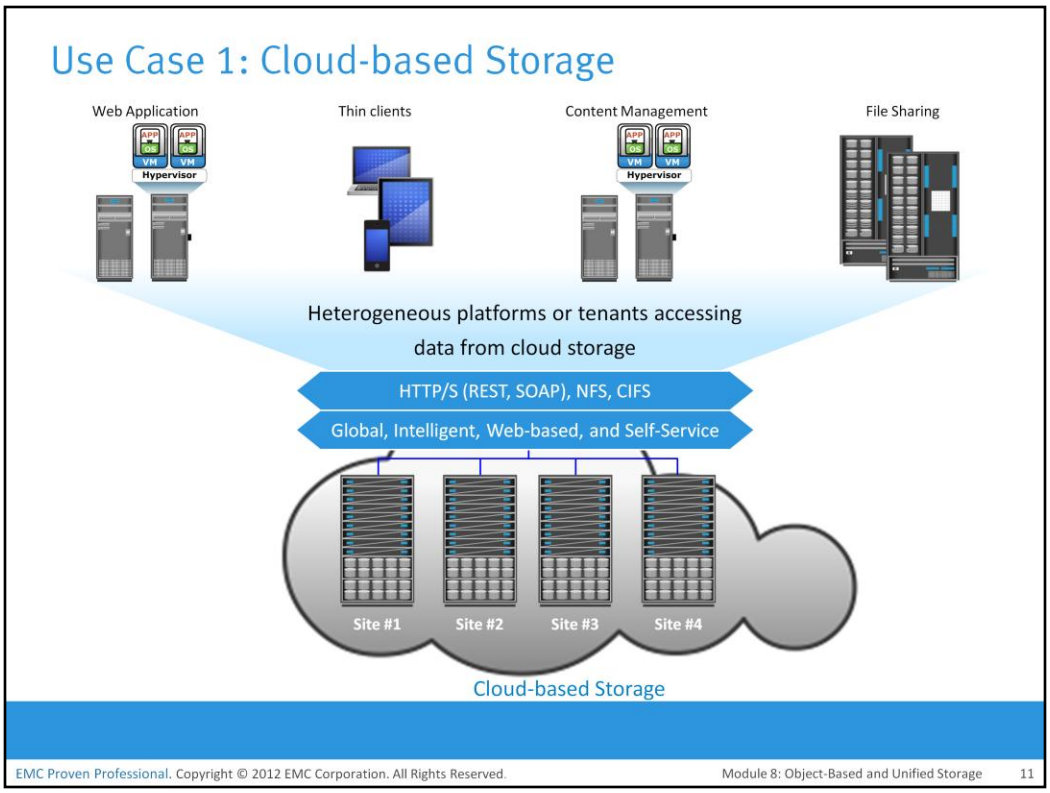
The process of retrieving objects in OSD is illustrated on the slide. The process of data retrieval from OSD is as follows:

1. The application server sends a read request to the OSD system.
2. The metadata service retrieves the object ID for the requested file.
3. The metadata service sends the object ID to the application server.
4. The application server sends the object ID to the OSD storage service for object retrieval.
5. The OSD storage service retrieves the object from the storage device.
6. The OSD storage service sends the file to the application server.



For unstructured data, object-based storage devices provide numerous benefits over traditional storage solutions. An ideal storage architecture should provide performance, scalability, security, and data sharing across multiple platforms. Traditional storage solutions, such as SAN, and NAS, do not offer all these benefits as a single solution. Object-based storage combines benefits of both the worlds. It provides platform and location independence, and at the same time, provides scalability, security and data-sharing capabilities. The key benefits of object-based storage are as follows:

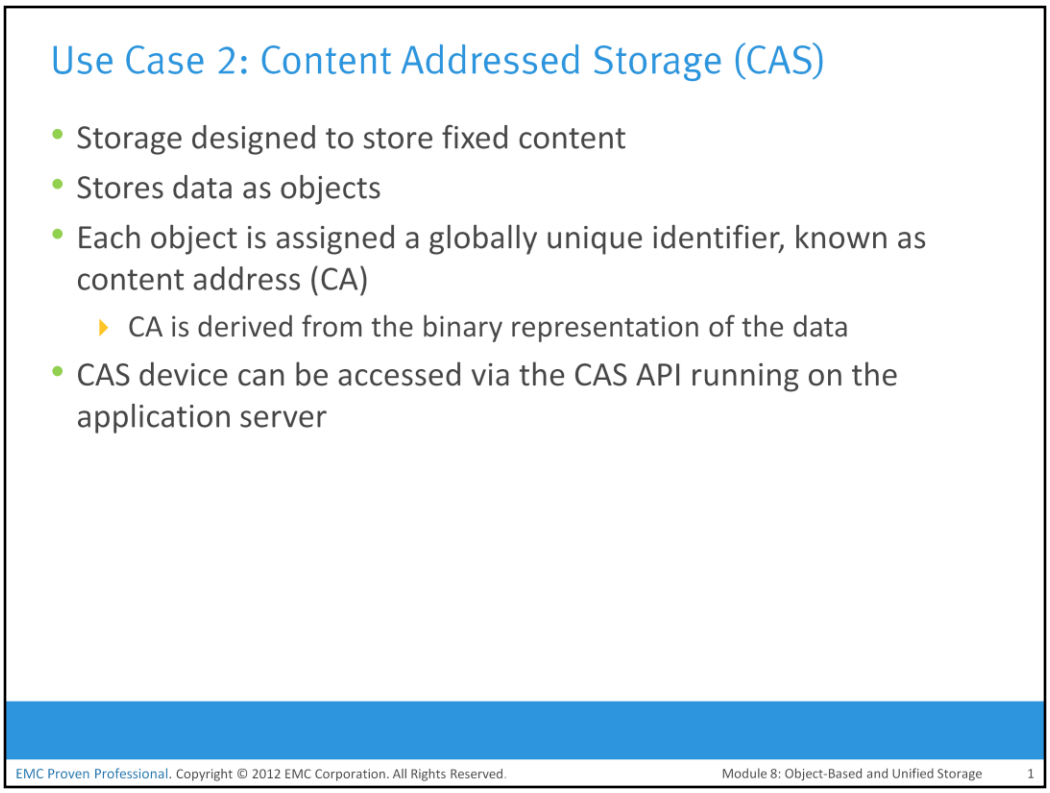
- *Security and reliability*: Data integrity and content authenticity are the key features of object-based storage devices. OSD uses specialized algorithms to create objects that provide strong data encryption capability. In OSD, request authentication is performed at the storage device rather than with an external authentication mechanism.
- *Platform independence*: Objects are abstract containers of data, including metadata and attributes. This feature allows objects to be shared across heterogeneous platforms locally or remotely. This platform-independence capability makes object-based storage the best candidate for cloud computing environments.
- *Scalability*: Due to the use of flat address space, object-based storage can handle large amounts of data without impacting performance. Both storage and OSD nodes can be scaled independently in terms of performance and capacity.
- *Manageability*: Object-based storage has an inherent intelligence to manage and protect objects. It uses self-healing capability to protect and replicate objects. Policy-based management capability helps OSD to handle routine jobs automatically.



Cloud-based storage is a promising use case of OSD. OSD uses a web interface to access storage resources. OSD provides inherent security, scalability, and automated data management. It also enables data sharing across heterogeneous platforms or tenants while ensuring integrity of data. These capabilities make OSD a strong option for cloud-based storage. Cloud service providers can leverage OSD to offer storage-as-a-service. OSD supports web service access via *representational state transfer* (REST) and *simple object access protocol* (SOAP). REST and SOAP APIs can be easily integrated with business applications that access OSD over the web.

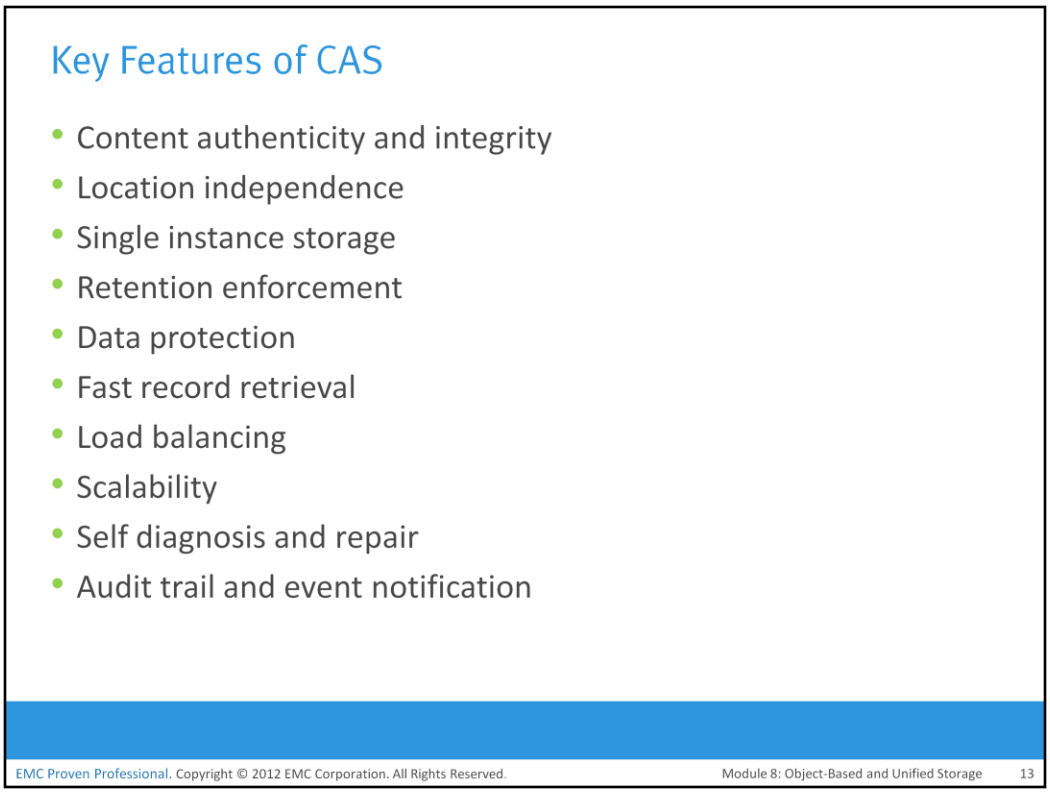
Representational State Transfer or REST is an architectural style developed for modern web applications. REST provides lightweight web services to access resources (for example, documents, blogs, and so on) on which a few basic operations can be performed, such as retrieving, modifying, creating, and deleting resources. REST-style web services are resource-oriented services. Resources can be uniquely located and identified by a Universal Resource Identifier (URI), and operations can be performed on those resources using an HTTP specification. For example, if a user accesses a blog using REST via a unique identifier, the request returns the representation of the blog in a particular format (XML or HTML). *Simple Object Access Protocol or* SOAP is a XML based protocol that enables communication between the web applications running on different OS and based on different programming languages. SOAP provides process to encode HTTP header and XML file to enable and pass information between different computers.

Cloud based storage is further discussed in module 13 Cloud Computing.



A data archival solution is a promising use case for OSD. Data integrity and protection is the primary requirement for any data archiving solution. Traditional archival solutions—CD and DVD-ROM—do not provide scalability and performance. OSD stores data in the form of objects, associates them with a unique object ID, and ensures high data integrity. Along with integrity, it provides scalability and data protection. These capabilities make OSD a viable option for long term data archiving for fixed content. Content Addressed storage (CAS) is a special type of object-based storage device purposely built for storing fixed content.

CAS is an object-based storage device designed for secure online storage and retrieval of fixed content. CAS stores user data and its attributes as an object. The stored object is assigned a globally unique address, known as a *content address* (CA). This address is derived from the object's binary representation. CAS provides an optimized and centrally managed storage solution. Data access in CAS differs from other OSD devices. In CAS, the application server can access the CAS device only via the CAS API running on the application server. However, the way CAS stores data is similar to the other OSD systems.

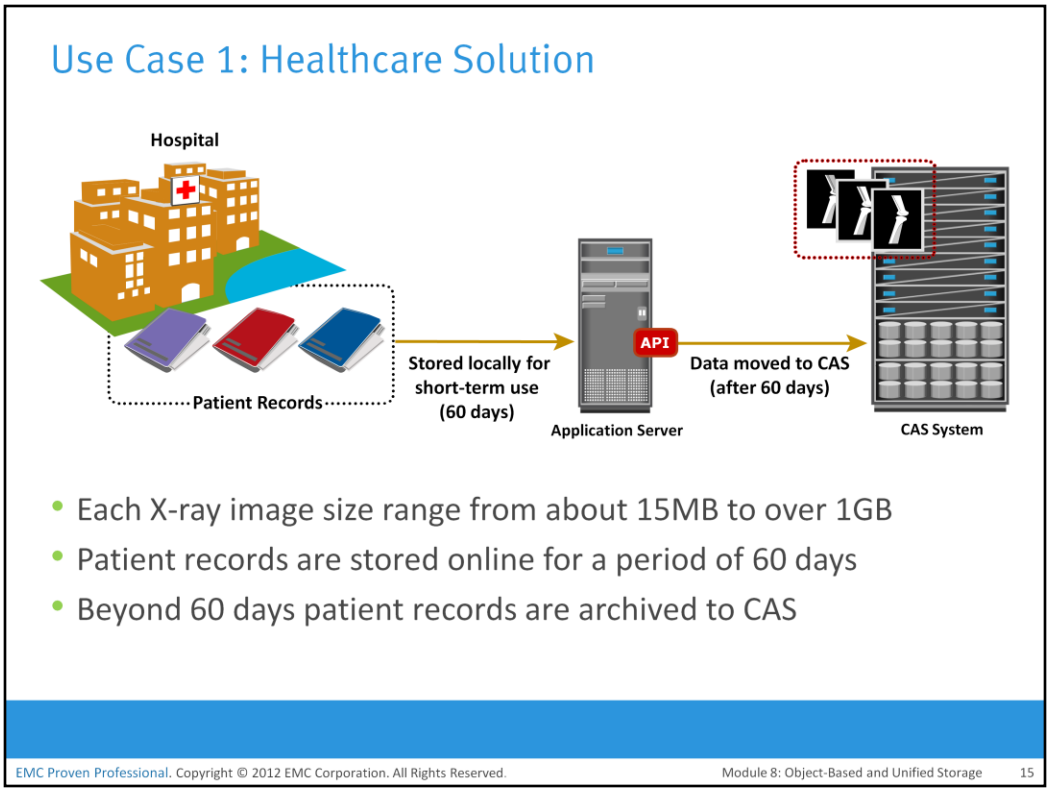


CAS provides all the features required for storing fixed content. The key features of CAS are as follows:

- *Content authenticity*: It assures the genuineness of stored content. This is achieved by generating a unique content address for each object and validating the content address for stored objects at regular intervals. Content authenticity is assured because the address assigned to each object is as unique as a fingerprint. Every time an object is read, CAS uses a hashing algorithm to recalculate the object's content address as a validation step and compares the result to its original content address. If the object fails validation, CAS rebuilds the object using a mirror or parity protection scheme.
- *Content integrity*: It provides assurance that the stored content has not been altered. CAS uses a hashing algorithm for content authenticity and integrity. If the fixed content is altered, CAS generates a new address for the altered content, rather than overwrite the original fixed content.
- *Location independence*: CAS uses a unique content address, rather than directory path names or URLs, to retrieve data. This makes the physical location of the stored data irrelevant to the application that requests the data.

Cont..

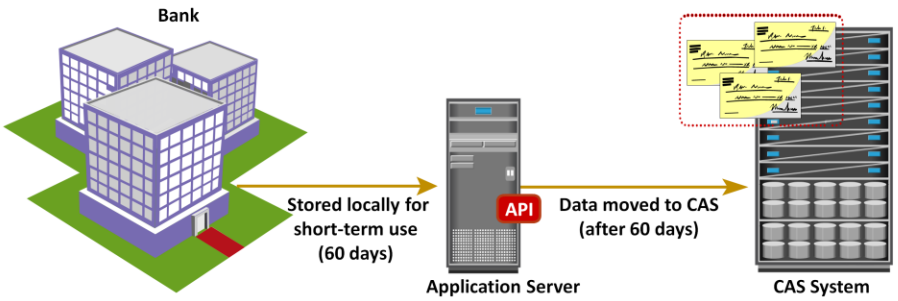
- *Single-instance storage (SIS)*: CAS uses a unique content address to guarantee the storage of only a single instance of an object. When a new object is written, the CAS system is polled to see whether an object is already available with the same content address. If the object is available in the system, it is not stored; instead, only a pointer to that object is created.
- *Retention enforcement*: Protecting and retaining objects is a core requirement of an archive storage system. After an object is stored in the CAS system and the retention policy is defined, CAS does not make the object available for deletion until the policy expires.
- *Data protection*: CAS ensures that the content stored on the CAS system is available even if a disk or a node fails. CAS provides both local and remote protection to the data objects stored on it. In the local protection option, data objects are either mirrored or parity protected. In mirror protection, two copies of the data object are stored on two different nodes in the same cluster. This decreases the total available capacity by 50 percent. In parity protection, the data object is split in multiple parts and parity is generated from them. Each part of the data and its parity are stored on a different node. This method consumes less capacity to protect the stored data, but takes slightly longer to regenerate the data if corruption of data occurs. In the remote replication option, data objects are copied to a secondary CAS at the remote location. In this case, the objects remain accessible from the secondary CAS if the primary CAS system fails.
- *Fast record retrieval*: CAS stores all objects on disks, which provides faster access to the objects compared to tapes and optical discs.
- *Load balancing*: CAS distributes objects across multiple nodes to provide maximum throughput and availability.
- *Scalability*: CAS allows the addition of more nodes to the cluster without any interruption to data access and with minimum administrative overhead.
- *Event notification*: CAS continuously monitors the state of the system and raises an alert for any event that requires the administrator's attention. The event notification is communicated to the administrator through SNMP, SMTP, or e-mail.
- *Self diagnosis and repair*: CAS automatically detects and repairs corrupted objects and alerts the administrator about the potential problem. CAS systems can be configured to alert remote support teams who can diagnose and repair the system remotely.
- *Audit trails*: CAS keeps track of management activities and any access or disposition of data. Audit trails are mandated by compliance requirements.



Large healthcare centers examine hundreds of patients every day and generate large volumes of medical records. Each record might be composed of one or more images that range in size from approximately 15 MB for a standard digital X-ray to more than 1 GB for oncology studies. The patient records are stored online for a specific period of time for immediate use by the attending physicians. Even if a patient's record is no longer needed, compliance requirements might stipulate that the records be kept in the original format for several years.

Medical image solution providers offer hospitals the capability to view medical records, such as X-ray images, with acceptable response times and resolution to enable rapid assessments of patients. Patients' records are retained on the primary storage for 60 days after which they are moved to the CAS system. CAS facilitates long-term storage and at the same time, provides immediate access to data, when needed.

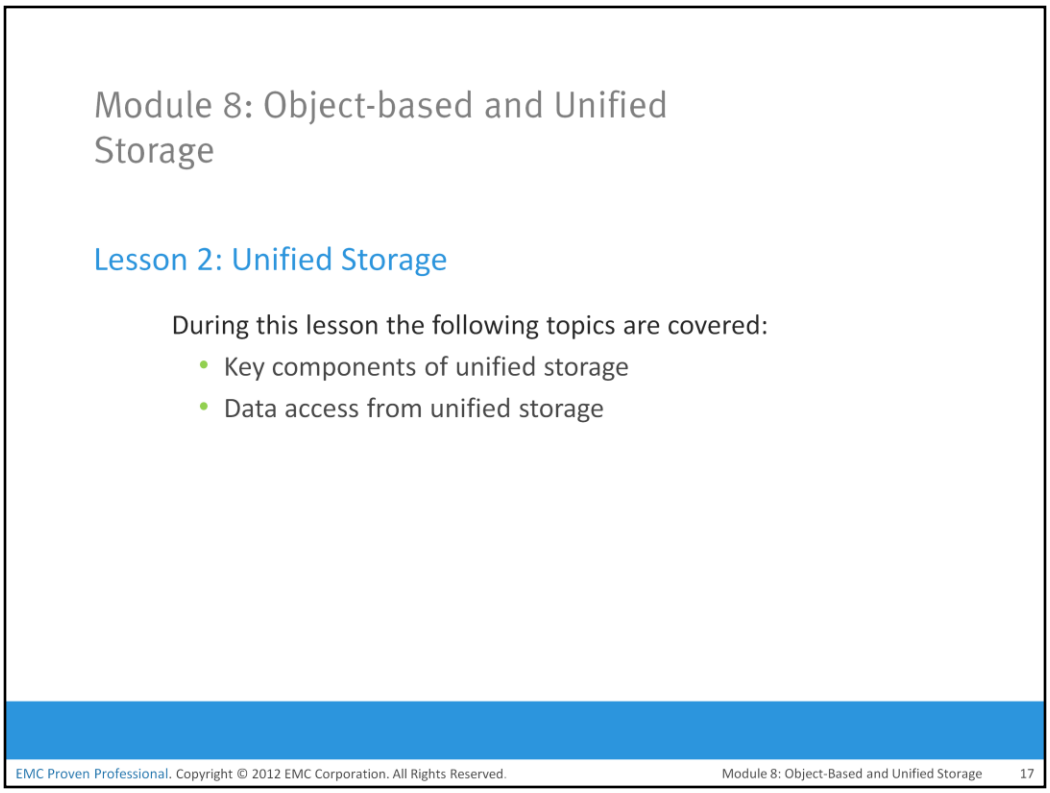
Use Case 2: Financial Solution



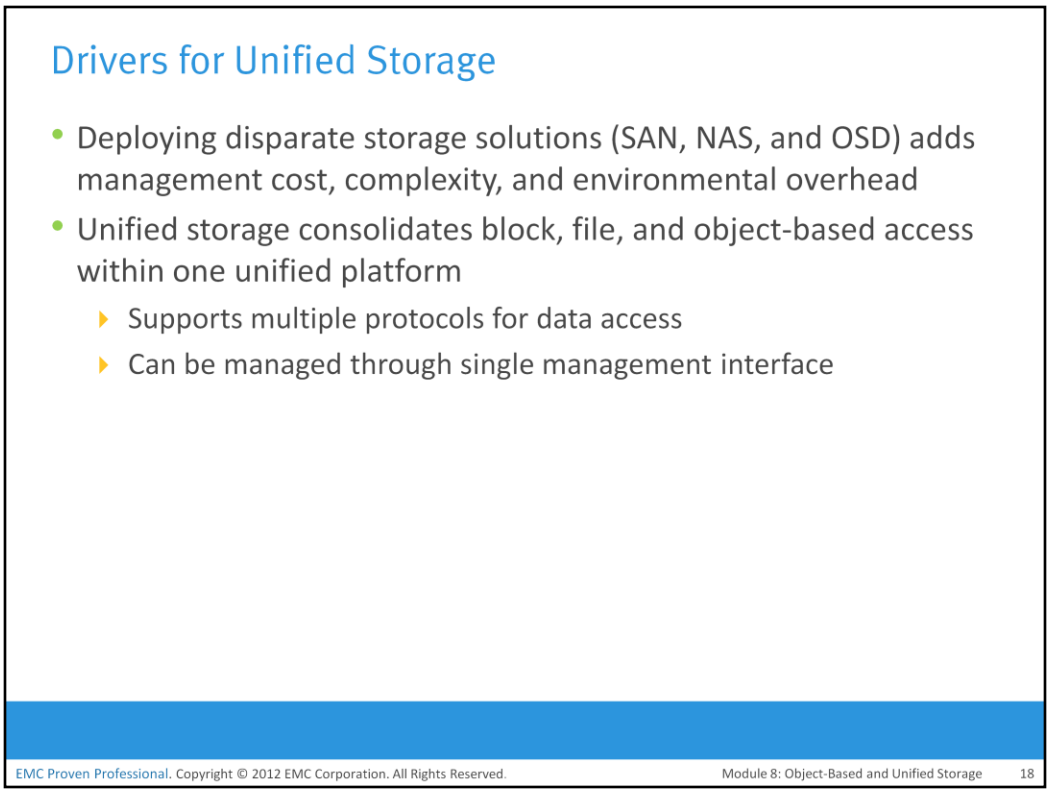
- Each check image size is about 25KB
- Check imaging service provider might process around 90 million check images per month
- Checks are stored online for a period of 60 days
- Beyond 60 days data is archived to CAS

In a typical banking scenario, images of checks, each approximately 25 KB in size, are created and sent to archive services over an IP network. A check imaging service provider might process approximately 90 million check images per month. Typically, check images are actively processed in transactional systems for about 5 days.

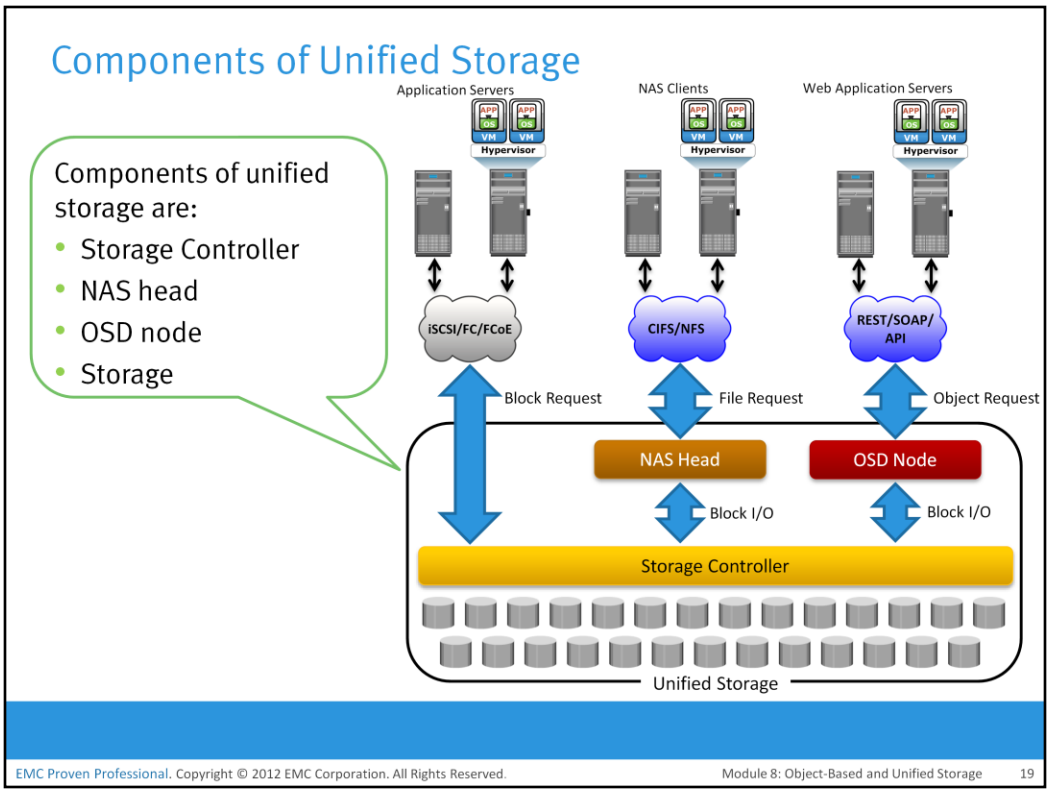
For the next 60 days, check images may be requested by banks or individual consumers for verification purposes and beyond 60 days, access requirements drop drastically. The check images are moved from the primary storage to the CAS system after 60 days, and can be held there for long term based on retention policy. Check imaging is one example of a financial service application that is best serviced with CAS. Customer transactions initiated by e-mail, contracts, and security transaction records might need to be kept online for 30 years; CAS is the preferred storage solution in such cases.



This lesson describes the function of key components of unified storage. This lesson also describes the process of data access from unified storage.

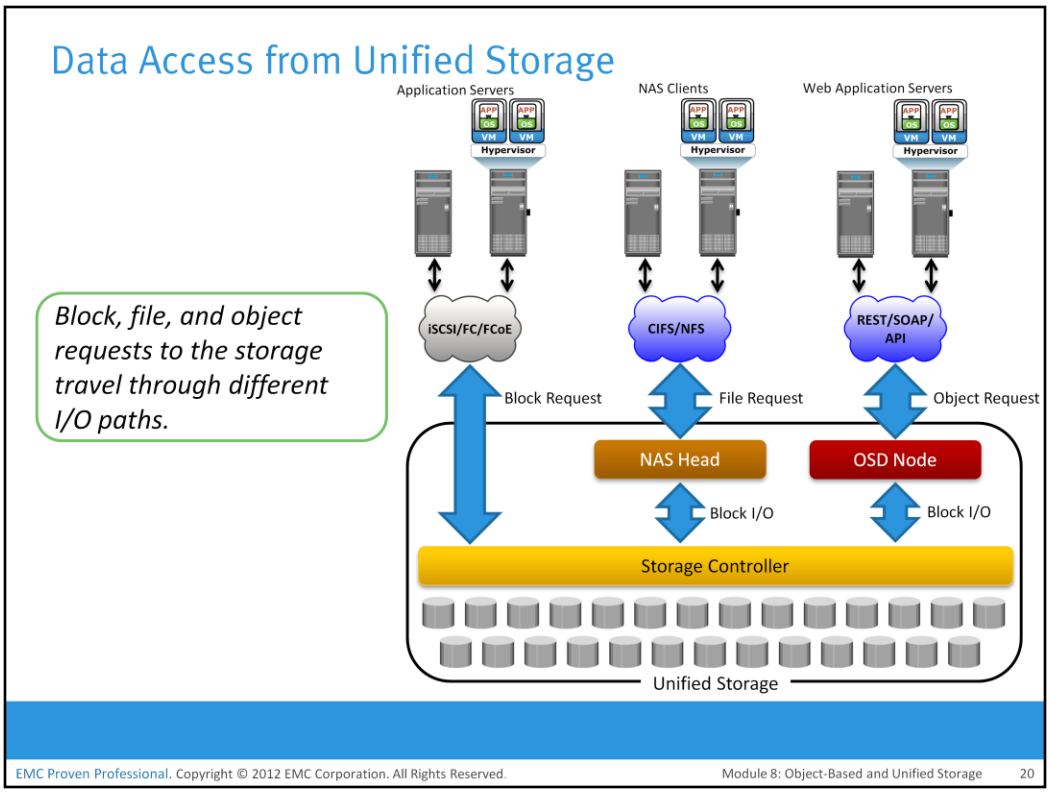


Due to varied application requirements, organizations have been deploying storage area networks (SANs), NAS, and object-based storage devices (OSDs) in their data centers. Deploying these disparate storage solutions adds management complexity, cost and environmental overheads. An ideal solution would be to have an integrated storage solution that supports block, file, and object access. Unified storage has emerged as a solution that consolidates block, file, and object-based access within one unified platform. It supports multiple protocols for data access and can be managed using a single management interface.



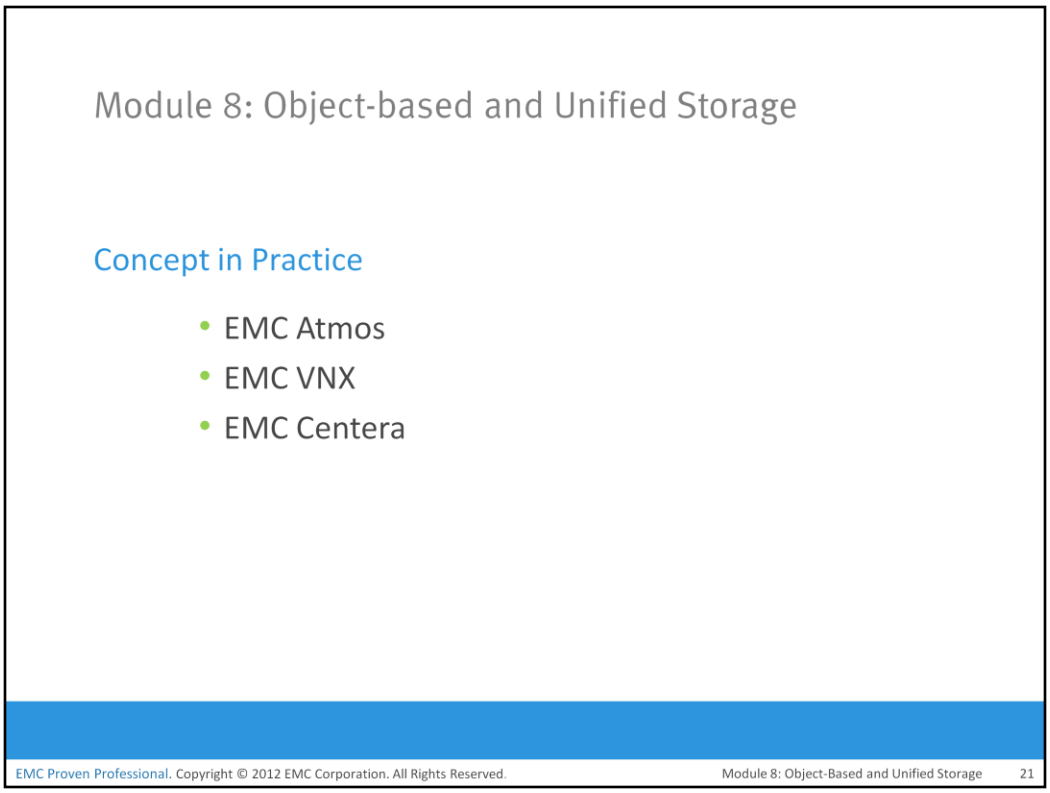
A unified storage system consists of following key components: storage controller, NAS head, OSD node, and storage.

- The *storage controller* provides block-level access to application servers through iSCSI, FC, or FCoE protocols. It contains iSCSI, FC, and FCoE front-end ports for direct block access. The storage controller is also responsible for managing the back-end storage pool in the storage system. The controller configures LUNs and presents them to application servers, NAS heads, and OSD nodes. The LUNs presented to the application server appear as local physical disks. A file system is configured on these LUNs and is made available to applications for storing data.
- A *NAS head* is a dedicated file server that provides file access to NAS clients. The NAS head is connected to the storage via the storage controller typically using a FC or FCoE connection. The system typically has two or more NAS heads for redundancy. The LUNs presented to the NAS head appear as physical disks. The NAS head configures the file systems on these disks, creates a NFS, CIFS, or mixed share, and exports the share to the NAS clients.
- The *OSD node* accesses the storage through the storage controller using a FC or FCoE connection. The LUNs assigned to the OSD node appear as physical disks. These disks are configured by the OSD nodes, enabling them to store the data from the web application servers.

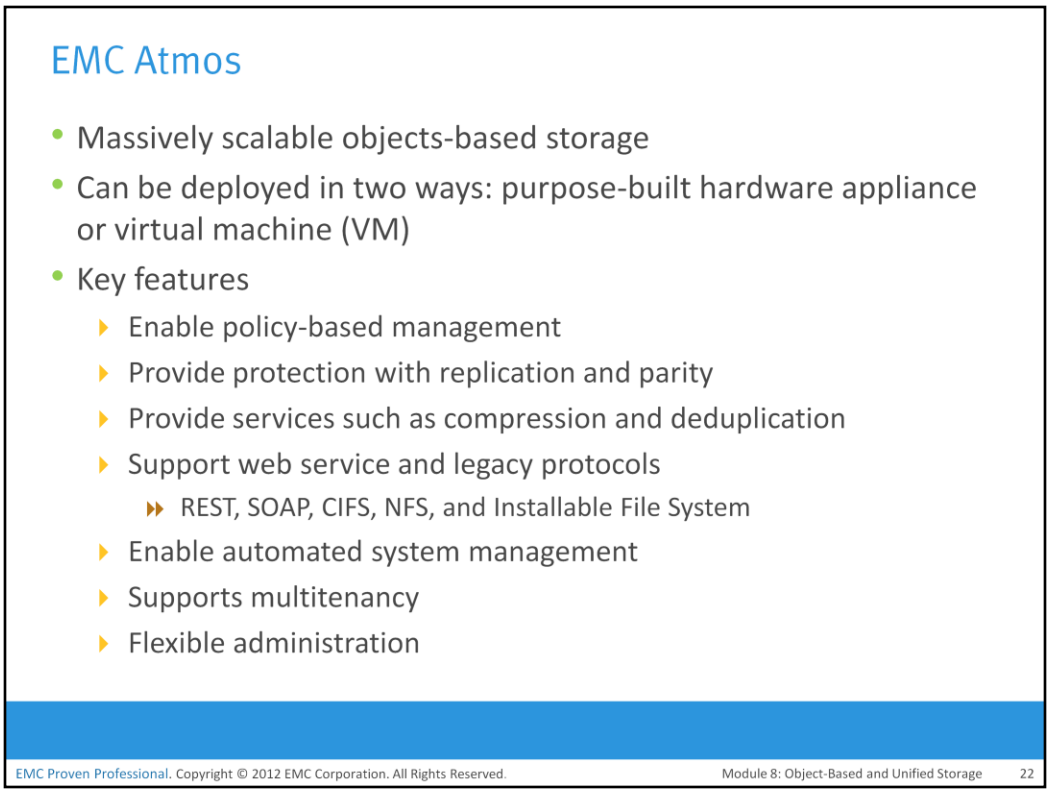


In a unified storage system, block, file, and object requests to the storage travel through different I/O paths.

- *Block I/O request*: The application servers are connected to an FC, iSCSI, or FCoE port on the storage controller. The server sends a block request over an FC, iSCSI, or FCoE connection. The storage controller processes the I/O and responds to the application server.
- *File I/O request*: The NAS clients (where the NAS share is mounted or mapped) sends a file request to the NAS head using the NFS or CIFS protocol. The NAS head receives the request, converts it into a block request, and forwards it to the storage controller. Upon receiving the block data from the storage controller, the NAS head again converts the block request back to the file request and sends it to the clients.
- *Object I/O request*: The web application servers send an object request, typically using REST or SOAP protocols, to the OSD node. The OSD node receives the request, converts it into a block request, and sends it to the disk through the storage controller. The controller in turn processes the block request and responds back to the OSD node, which in turn provides the requested object to the web application server.



The Concept in Practice covers the product example of object-based and unified storage. It covers three products: EMC Atmos, EMC VNX, and EMC Centera.



EMC Atmos supports object-based storage for unstructured data, such as pictures and videos. Atmos combines massive scalability with specialized intelligence to address the cost, distribution, and management challenges associated with vast amounts of unstructured data.

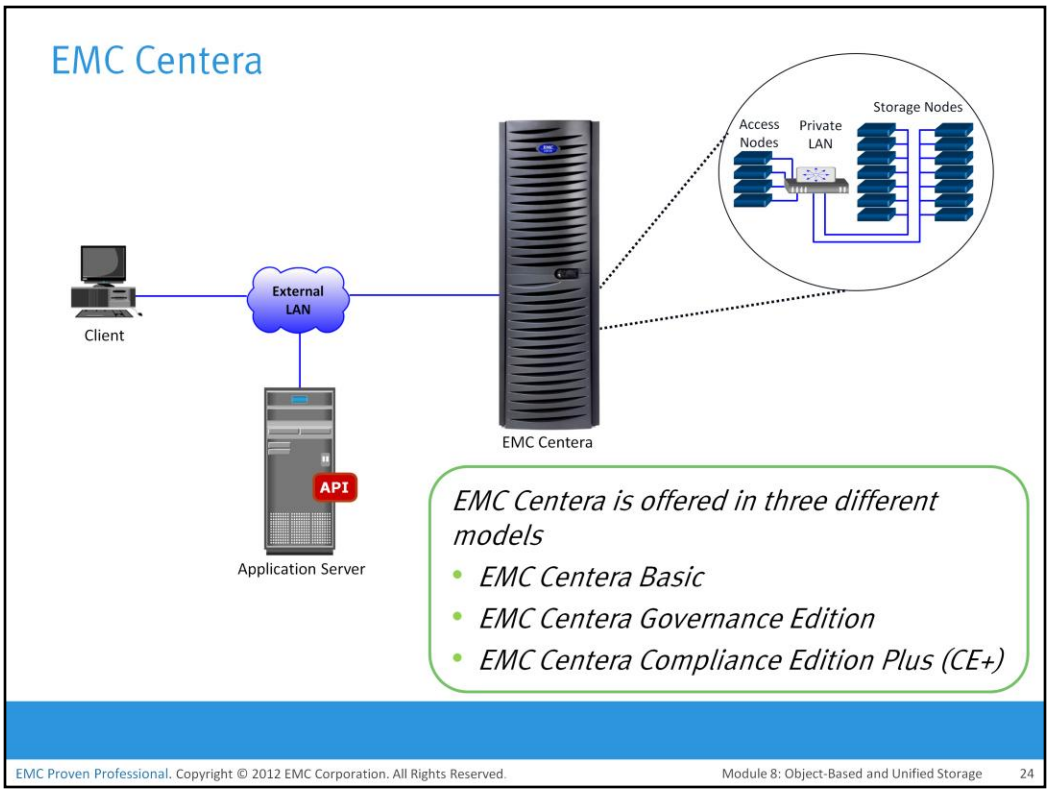
Atmos can be deployed in two ways: as a purpose-built hardware appliance or as software in VMware environments, where Atmos VE can leverage the existing servers and storage. The hardware appliance is comprised of servers (nodes) connected to standard disk enclosures. The rack includes a 24-port Gigabit Ethernet switch to provide internode communication. The Atmos software is installed on each node.

Atmos VE enables users to exploit the power of Atmos in a virtualized environment. It can be deployed on a virtual machine in VMware ESXi hosts and configured with the VMware certified back-end storage.

Cont..

Following are the key features offered by EMC Atmos:

- *Policy-based management*: EMC Atmos improves operational efficiency by automatically distributing content based on business policy. The administrator-defined policies dictate how, when, and where the information resides.
- *Protection*: Atmos offers two options to protect the objects, replication and Geo Parity:
  - *Replication* ensures that the content is available and accessible by creating redundant copies of an object at multiple designated locations.
  - *Geo Parity* ensures that the content is available and accessible by dividing objects into multiple segments plus parity segments and distributing them to one or more designated locations.
- *Data services*: EMC Atmos includes the data services, such as, compression, and deduplication. These features are native to Atmos and can be managed and accessed via a policy.
- *Web services and legacy protocols*: EMC Atmos provides flexible web services access (REST/SOAP) for web-scale applications and file access (CIFS/NFS/Installable File System/Centera API) for traditional applications.
- *Automated system management*: EMC Atmos provides auto-configuring, auto-managing, and auto-healing capabilities to reduce administration and downtime.
- *Multitenancy*: EMC Atmos enables multiple applications to be served from the same infrastructure. Each application is securely partitioned and cannot access the other application's data. Multitenancy is ideal for service providers or large enterprises that want to provide cloud computing services to multiple customers or departments allowing logical and secure separation within a single infrastructure.
- *Flexible administration*: EMC Atmos can be managed via a graphical user interface (GUI) or command line interface (CLI).



EMC Centera is a simple, affordable, and secure repository for information archiving. EMC Centera is designed and optimized specifically to deal with the storage and retrieval of fixed content by meeting performance, compliance, and regulatory requirements. Compared to traditional archive storage, EMC Centera provides faster record retrieval, Single instance storage (SIS), guaranteed content authenticity, self-healing, and support for numerous industry and regulatory standards.

EMC Centera is offered in three different models to meet different types of user requirements—EMC Centera Basic, EMC Centera Governance Edition, and EMC Centera Compliance Edition Plus (CE+):

- *EMC Centera Basic*: Provides all functionalities without the enforcement of retention periods.
- *EMC Centera Governance Edition*: Provides the retention capabilities required by organizations to manage digital records in addition to the features provided by EMC Centera Basic.
- *EMC Centera Compliance Edition Plus*: Provides extensive compliance capabilities. CE+ is designed to meet the requirements of the most stringent regulated business environments for electronic storage media, as established by regulations from the Securities and Exchange Commission (SEC), or other national and international regulatory groups.

Cont..

A client accesses the Centera over a LAN. The client can access Centera only through the server that runs the Centera API (application programming interface). The Centera API is responsible for performing functions that enable an application to store and retrieve the data.

Centera architecture is a *Redundant Array of Independent Nodes* (RAIN). It contains storage nodes and access nodes that are networked as a cluster by using a private LAN. The internal LAN reconfigures automatically when it detects configuration changes, such as the addition of storage or access nodes. The application server accesses the Centera via an external LAN.

The nodes are configured with low-cost, high-capacity SATA disk drives. These nodes run CentraStar, the operating environment for Centera, which provides the features and functionalities required in a Centera system.

When the nodes are installed, they are configured with a “role” that defines the functionality provided by the node . A node can be configured as a storage node, an access node, or a dual-role node.

Storage nodes store and protect data objects. They are sometimes referred to as *back-end nodes*.

Access nodes provide connectivity to application servers through an external LAN. They establish connectivity with the storage nodes in the cluster through a private LAN. The number of access nodes is determined by the amount of throughput required from the cluster. If a node is configured solely as an “access node,” its disk space cannot be used to store data objects. Storage and retrieval requests are sent to the access node via the external LAN.

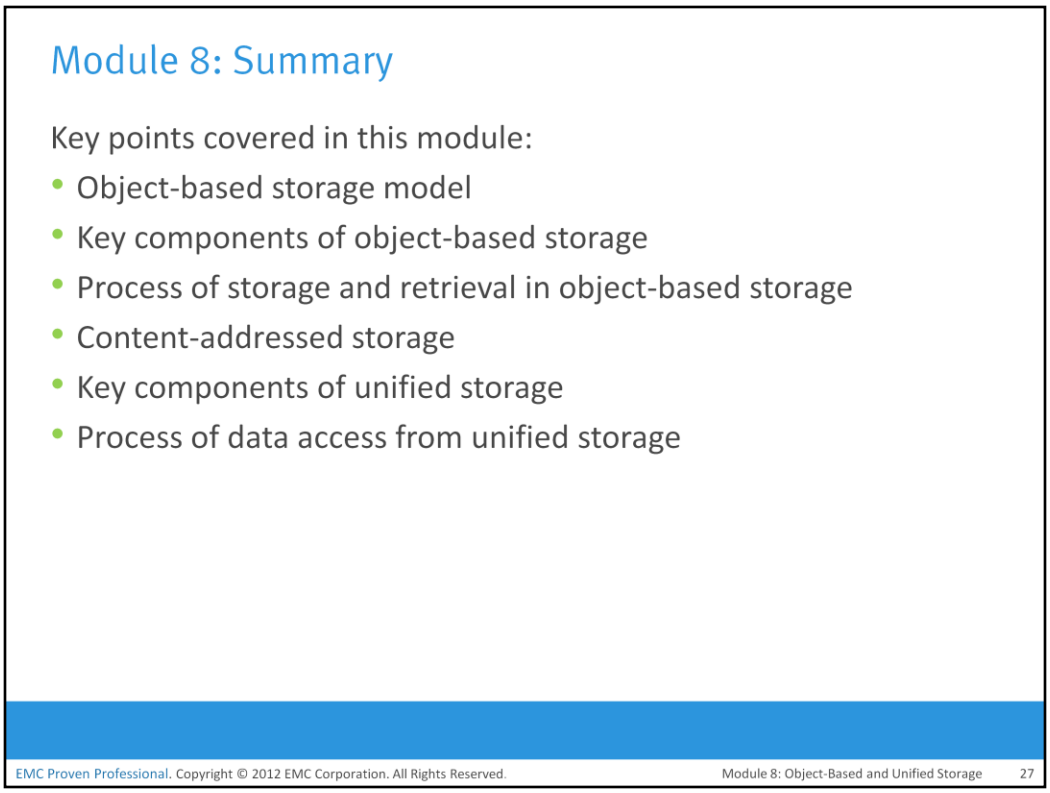
Dual-role nodes provide both storage and access-node capabilities. This configuration is more common than a pure access-node configuration.

EMC VNX

- Unified storage platform that consolidates block, file, and object accesses in one solution
  - Supports block access via storage processors
  - File access via X-Blade
  - Object access via EMC Atmos VE
- Components of VNX are:
  - Storage processor
  - X-Blade
  - Control station
  - Disk-array enclosures
  - Standby power supply

EMC VNX is a unified storage platform that consolidates block, file, and object access in one solution. It implements a modular architecture that integrates hardware components for block, file, and object access. EMC VNX delivers file access (NAS) functionality via X-Blades (Data Movers) and block access functionality via storage processors. Optionally it offers object access to the storage using EMC Atmos Virtual Edition (Atmos VE). VNX storage systems include the following components:

- *Storage processors* (SPs) support block I/O access to storage with FC, iSCSI, and FCoE protocols.
- *X-Blades* access data from the back end and provide host access with NFS, CIFS, MPFS, pNFS, and FTP protocols. The X-Blades in each array are scalable and provide redundancy to ensure no single point of failure.
- *Control Stations* provide management functions to the X-Blades. The Control Station is also responsible for X-Blade failover. The Control Station may optionally be configured with a matching secondary Control Station to ensure management redundancy on the VNX array.
- *Disk-array enclosures* (DAEs) house the drives used in the array. Different sized DAEs are available that can each hold a maximum of 15, 25, or 60 drives. More DAEs can be added to meet growing storage demands.
- *Standby power supplies* provide enough power to each storage processor and first DAE to ensure that any data in flight is stored in the vault area if a power failure occurs. This ensures that no writes are lost.



This module covered the object-based storage model. Object-based storage has three key components such as OSD nodes, internal network, and storage. This module also covered the process of storage and retrieval in object-based storage and content-addressed storage.

Further, this module covered the key components of unified storage; storage processor, NAS head, OSD node, and storage. Finally, this module covered the process of data access from unified storage.

Check Your Knowledge – 1

- What is an advantage of a flat address space over a hierarchical address space?
  - A. Highly scalable with minimal impact on performance
  - B. Provides access to data, based on retention policies
  - C. Provides access to block, file, and object with same interface
  - D. Consumes less bandwidth on network while accessing data

- What is a role of metadata service in an OSD node?
  - A. Responsible for storing data in the form of objects
  - B. Stores unique IDs generated for objects
  - C. Stores both objects and object IDs
  - D. Controls functioning of storage devices

Check Your Knowledge – 2

- What is used to generate an object ID in a CAS system?
  - A. File metadata
  - B. Source and destination address
  - C. Binary representation of data
  - D. File system type and ownership

- What accurately describes block I/O access in a unified storage?
  - A. I/O traverse NAS head and storage controller to disk
  - B. I/O traverse OSD node and storage controller to disk
  - C. I/O traverse storage controller to disk
  - D. I/O is directly sent to the disk

Check Your Knowledge – 3

- What accurately describes unified storage?
  - A. Provides block, file, and object-based access within one platform
  - B. Provides block and file storage access using objects
  - C. Supports block and file access using flat address space
  - D. Specialized storage device purposely built for archiving

