SNIA™
Cloud Storage Initiative

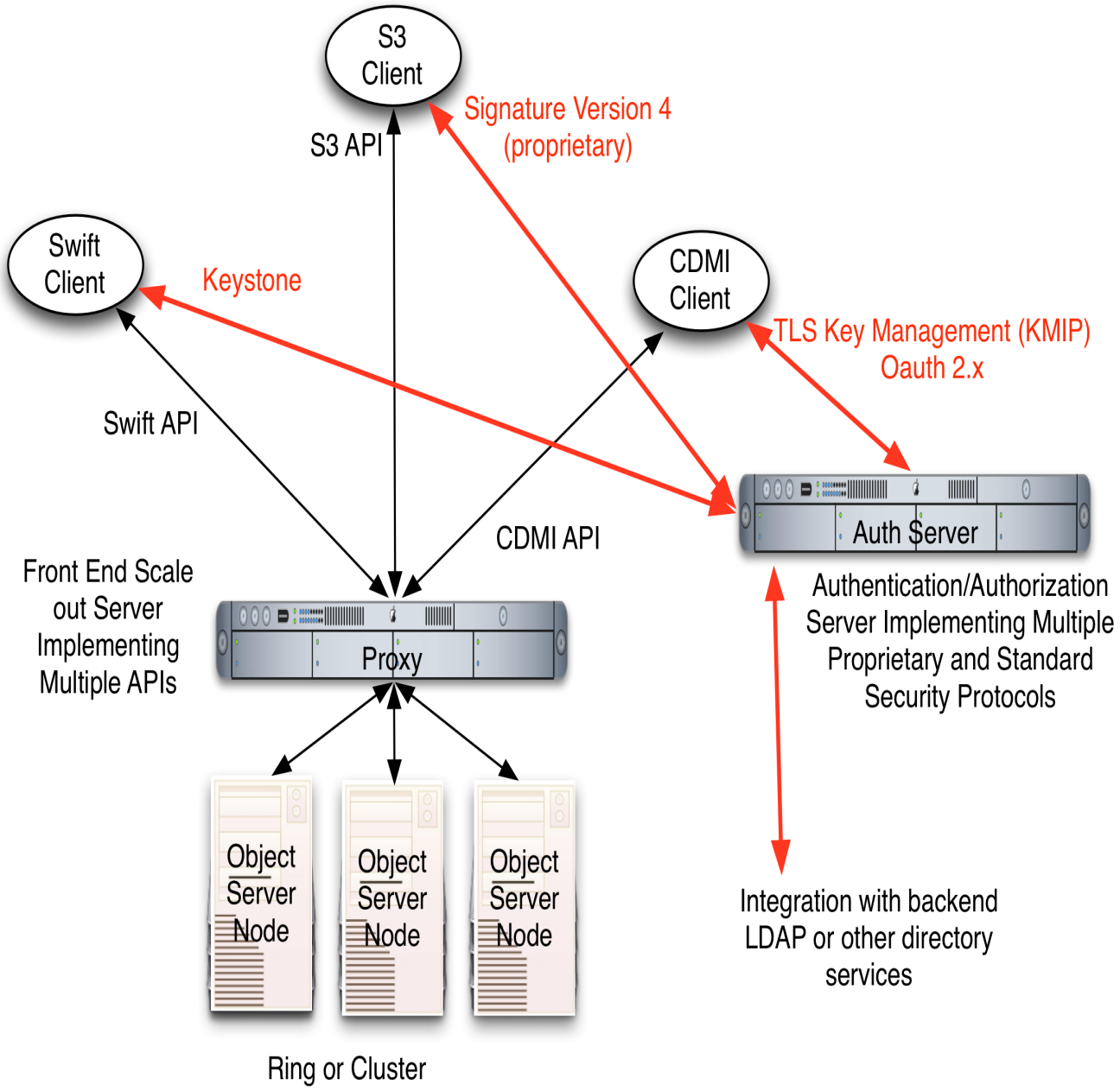
Extending OpenStack Swift with S3 and CDMI Interfaces

What experience have you had?

SNIA™
Cloud Storage Initiative

- When customers ask for support of a given API, can a vendor survive if they ignore these requests?
- A strategy many vendors are taking is to support multiple APIs with a single implementation.
  - Besides the Swift API, many support the S3 defacto and CDMI standard APIs in their implementation.
- What is needed for these APIs to co-exist in an implementation?
- Basic operations are nearly identical between APIs, but what about semantics that have multiple different expressions such as metadata?
- Best practices and tips to implementing multiple protocols in your cloud storage solution

What does this look like?



Breakdown

- Storage Operations
  - CRUD – All pretty much determined by HTTP standard (common code)
  - Headers are API unique however (handle in API specific modules)
- Security Operations
  - Client communication with Auth Server (API unique)
  - Multiple separate services running in Auth Server

What resources are available

- Comparison of S3/Swift functions
- **https://wiki.openstack.org/wiki/Swift/APIFeatureComparison**
  - Somewhat dated – needs updating
- Implementation of CDMI filter driver for Swift
- **https://github.com/osaddon/cdmi**
  - Stagnant for 2 years
  - Implementation of S3 filter driver for Swift
  - **https://github.com/fujita/swift3**
  - Also stagnant

Support

- Latest version of CDMI - http://www.snia.org/sites/default/files/CDMI_Spec_v1.1.1.pdf
- Baseline operations (mostly governed by RFC 2616) now documented in Clause 6 (pgs. 29-36)
- CDMI now uses content type to indicate CDMI-style operations (as opposed to X-CDMI-Specification-Version)
- Spec text that explicitly forbid (in 1.0) functionality required for S3/Swift integration has been removed from the spec.
- HTTP Basic/Digest authentication is no longer mandatory. CDMI implementations can now use S3 or Swift authentication exclusively, if desired.

Discovery of Security Protocol Implementations

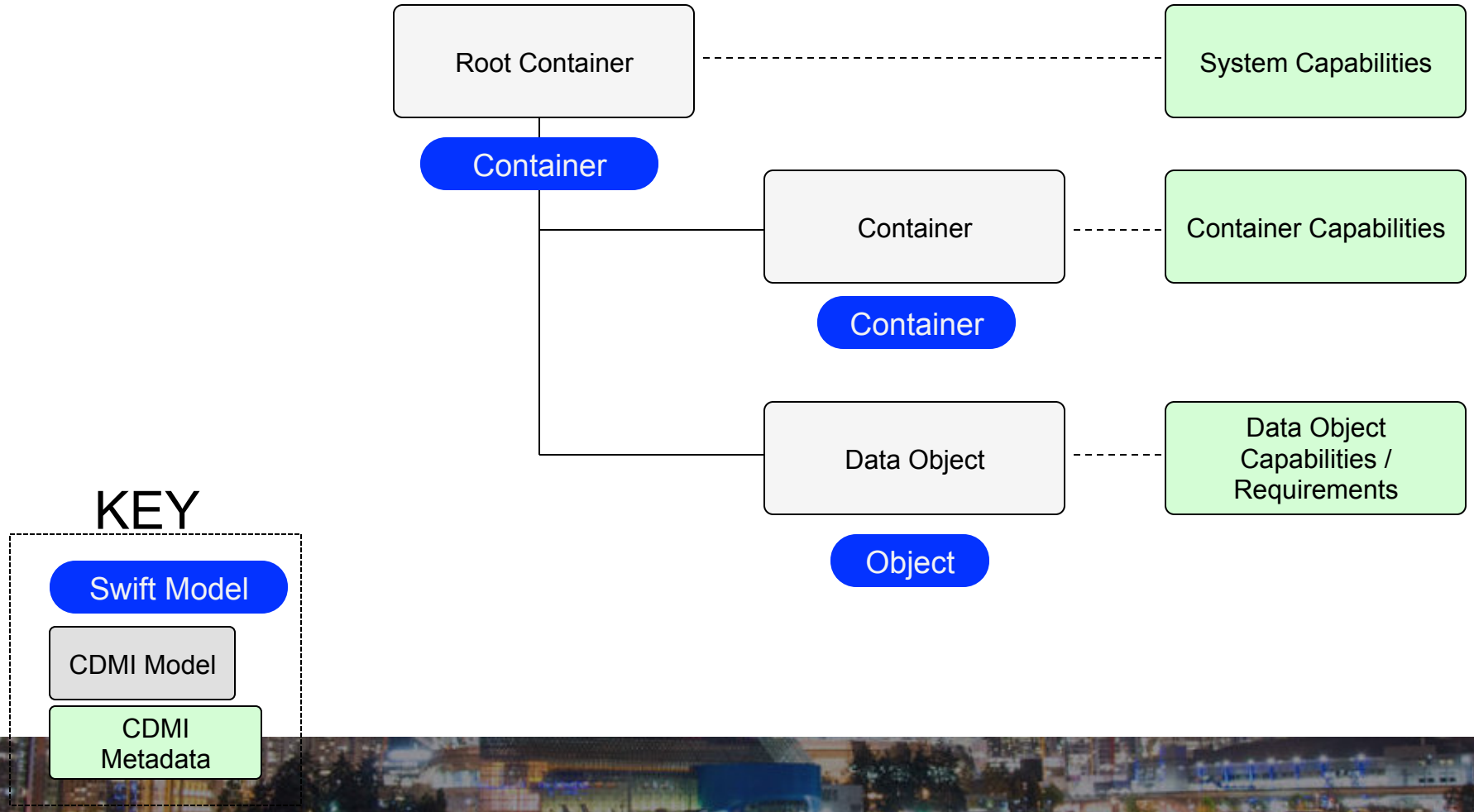
- CDMI 1.1 now includes a standard means of discovering what methods are available:
- cdmi_authentication_methods (Data System Metadata) ***12.1.3***
- If present, this capability contains a list of server-supported authentication methods that are supported by a domain. The following values for authentication method strings are defined:

• "anonymous"-Absence of authentication supported
• "basic"-HTTP basic authentication supported (RFC2617)
• "digest"-HTTP digest authentication supported (RFC2617)
• "krb5"-Kerberos authentication supported, using the Kerberos Domain specified in the CDMI domain (RFC 4559)
• "x509"-certificate-based authentication via TLS (RFC5246)

Extending the standard security types

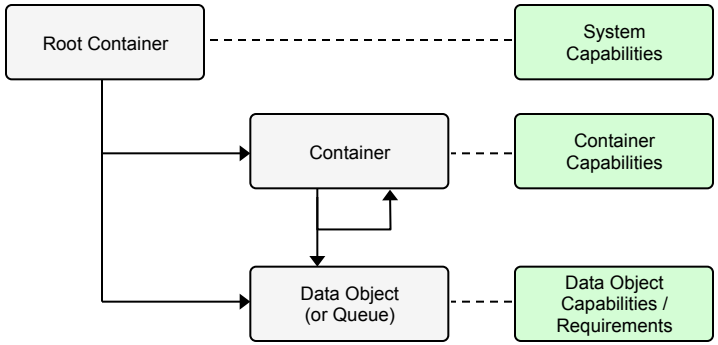
- The following values are examples of other widely used authentication methods that may be supported by a CDMI server:
  - "s3"-S3 API signed header authentication supported
  - "openstack"-OpenStack Identity API header authentication supported
  - Interoperability with these authentication methods are not defined by this international standard.
  - Servers may include other authentication methods not included in the above list. In these cases, it is up to the CDMI client and CDMI server (implementations themselves) to ensure interoperability.
  - When present, the cdmi_authentication_methods data system metadata shall be supported for all domains.

Swift and CDMI Models

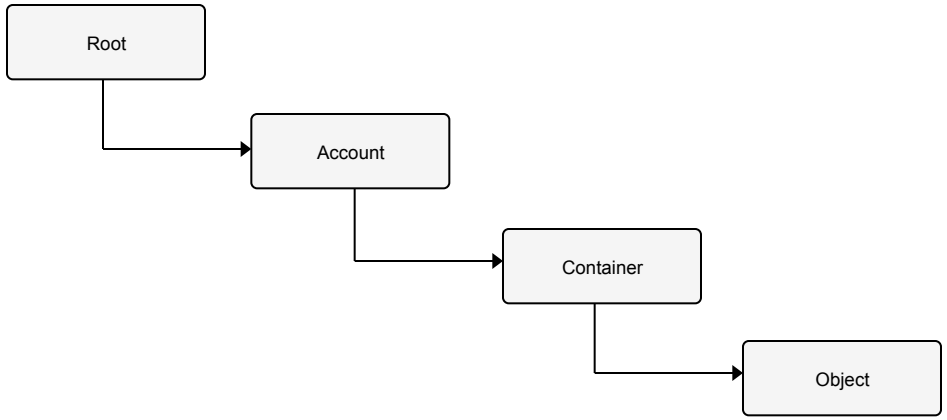


Swift and CDMI Models

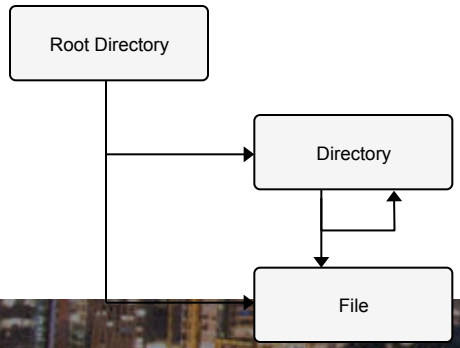
CDMI Object Model



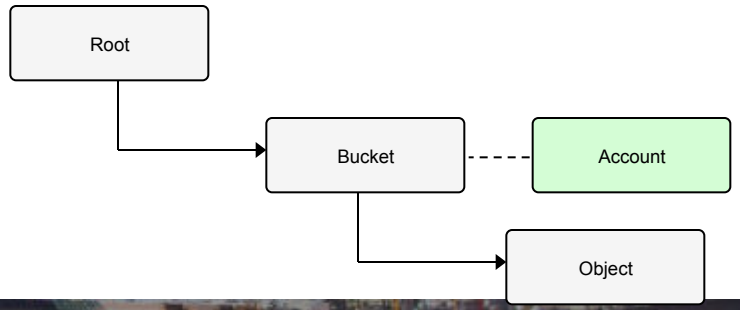
Swift Object Model



File System Model

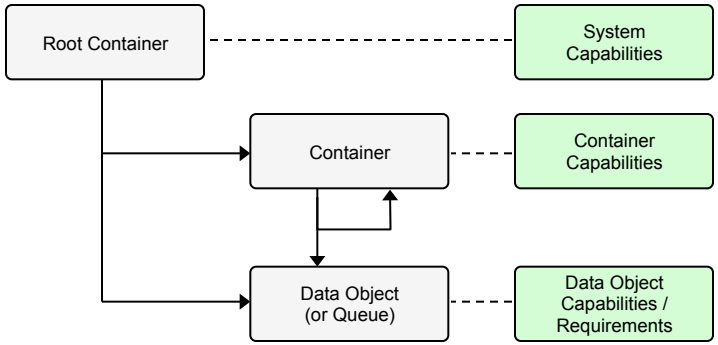


S3 Object Model

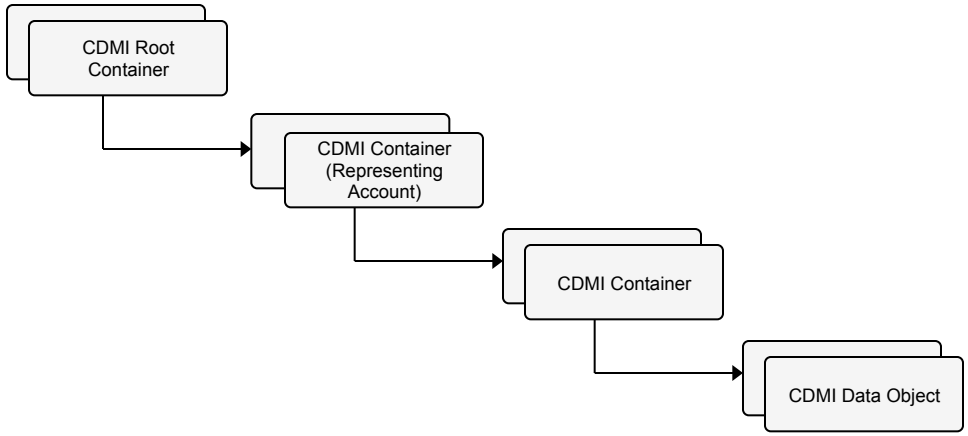


Swift and CDMI Models

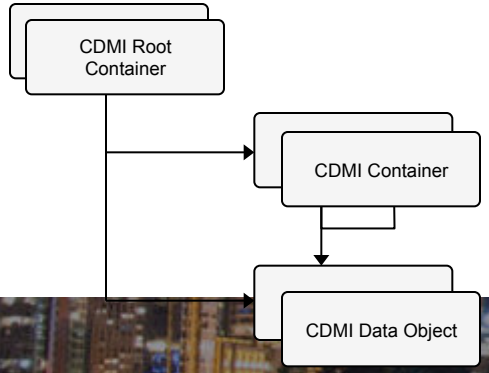
CDMI Object Model



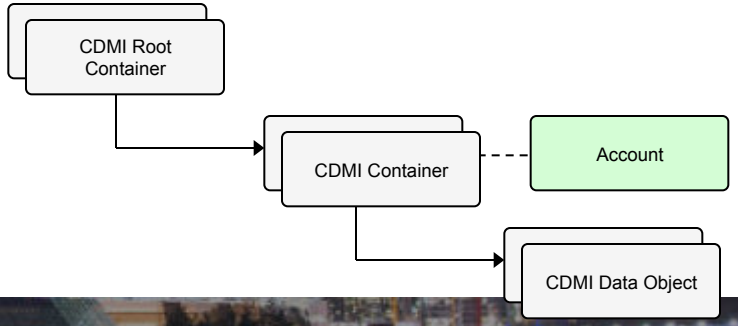
Overlay for Swift Object Model



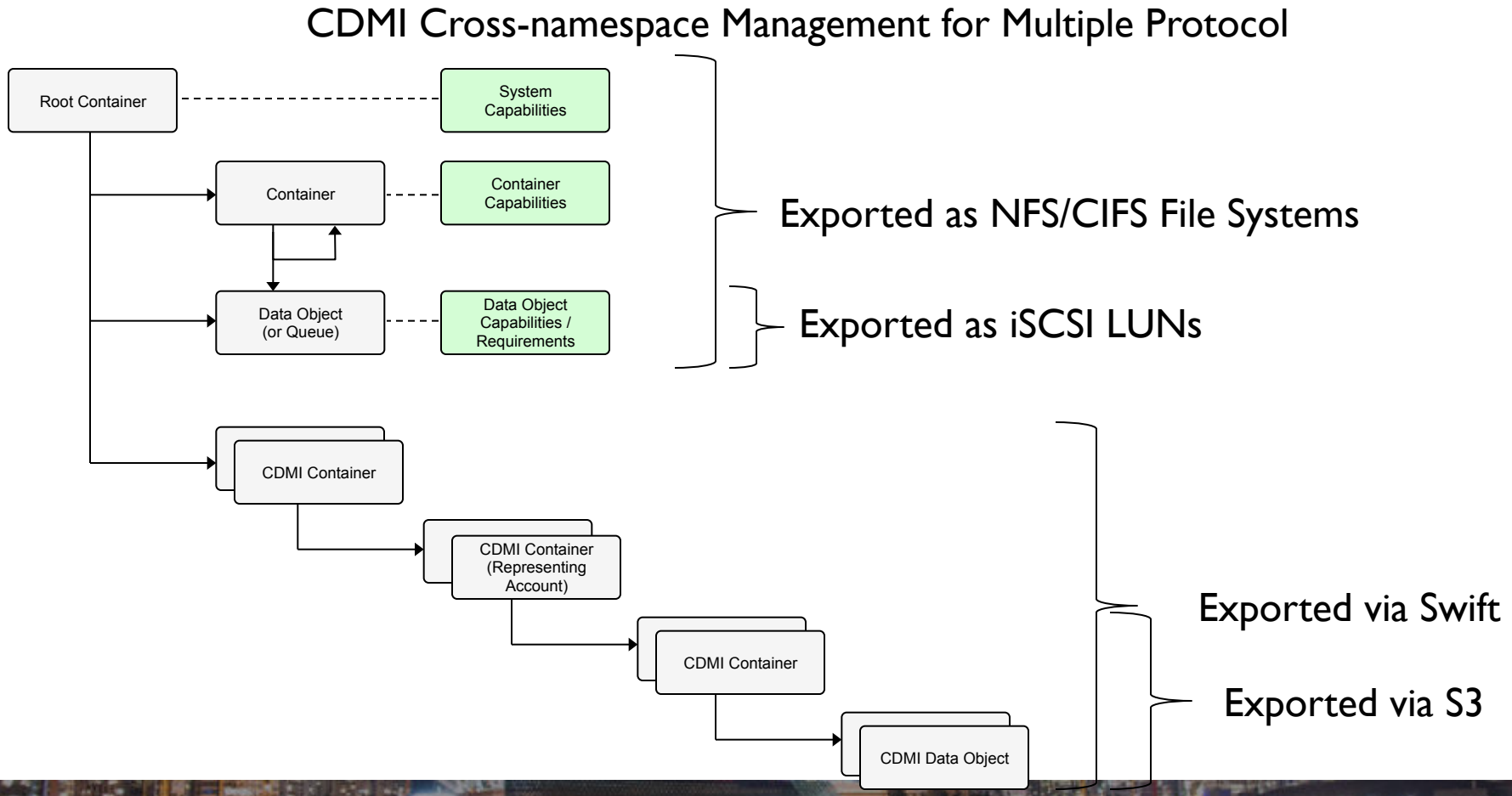
Overlay for File System Model



Overlay for S3 Object Model



Swift and CDMI Models



* File-system-like hierarchies can be emulated on top of S3/Swift, but lack much of the operational expressiveness

CDMI Object Metadata

- 2 major classes – System and User
- SYSTEM Metadata
  - Storage System: Timestamps, Traditional ACLs, Counts etc.
  - Data System: requirements of the object – eg. Retention, Backup, Replication, Performance
- USER Metadata
  - Application and data specific
  - E.g. EXIF data on photos
  - Location data of objects
  - Relationship data
  - SEARCHABLE!

CDMI Capability Metadata

- Describes the capability of a service participating in CDMI cloud environment
  - Performance
  - Retention capability
  - Location information
  - Storage features (Compression, Encryption, Hashes etc.)
- Shrink to fit development and consumption model
  - Advertising capability means that developers only need to implement the standard partially and only advertise what is implemented

CDMI and Swift

- CDMI works alongside Swift and S3 models, not replacing them
- Any of the APIs can be used to access the same data
- CDMI has been implemented for Swift as a filter which allows leverage of the Swift authentication filter
- If Swift and CDMI disagree, then CDMI "faults"
- Swift and CDMI use hierarchical containers but are slightly different in the implementation and language used
  - Swift Folder = CDMI Container
- CDMI Metadata can be stored in Swift metadata storage
  - This means the size limitation is implied for CDMI metadata currently

After This Webcast

- These slides were also given as a webcast and a copy of the slides are posted to the SNIA-CSI website and available on-demand
  - http://www.snia.org/forum/csi/knowledge/webcasts
- A full Q&A from that webcast is posted to the SNIA Cloud blog
  - http://www.sniacloud.com/
- Follow us on Twitter @SNIACloud
- Google Groups:
  - http://groups.google.com/group/snia-cloud

Questions?

- Thank You