Server Overview Guide for Cisco Unified Communications Domain Manager 8.1.4

First Published: September 30, 2014
Last Modified: September 30, 2014

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Server Overview Guide for Cisco Unified Communications Domain Manager 8.1.4

Contents

Overview

This document provides a technical overview of the CUCDM Server. It is aimed at Technical and Operational personnel responsible for the deployment, management and configuration of a CUCDM-Server (platform).

Note

- This document covers highly technical tasks that can have drastic consequences if not followed correctly. This content is aimed at advanced administrators and system engineers.
- This document is aimed at Virtual environments and UCS based versions of CUCDM Server. For Gentoo based CUCDM-Server versions, please refer to documentation specific to CUCDM-Server 2.1.7 and below.

This system supports various deployments/solutions including HCS and Large Enterprise (LE). This document describes the product in general and is not specific to a particular deployment/solution. Information may vary slightly depending on the installation environment.

Typographic Conventions

The following typographic conventions are used in this document:

Item	Character format	Example
Buttons	**Bold**	Click the **Enter** button.
Checkboxes	*italic*	Select the *Country* checkbox.
Dialog boxes menu items, tab names, radio buttons	*italic*	Select the *Configuration* option, or select the *Parameters* tab.

CHAPTER 1

Introduction

CUCDM-Server Version 4

CUCDM-Server Version 4 is an Ubuntu 10.04 LTSm (Lucid Lynx) server that has been designed to run in a virtual environment, using off-the-shelf VMWare tools

Setup and configuration is managed via a centralized command line interface.

Cisco Unified Computing System

Cisco Unified Computing System (UCS) is a data center platform that combines computing, network, storage access and virtualization into a single, easily managed system. UCS operates on Cisco specific hardware that is based on unified network fabric combined with cx86-architecture servers.

CUCDM-Server is designed to be run on the UCS platform.

Benefits of using the Cisco Unified Computing System include:

- A reduced total cost of ownership
- An increase in flexibility and business agility
- Virtualization reduces the number of hardware devices requiring configuration, management, power, cooling and cabling.

For more information on Cisco UCS, please see the Cisco[1] website.

[1] http://www.cisco.com/en/US/netsol/ns944/index.html#~overview

CHAPTER 2

CUCDM Platform Architecture

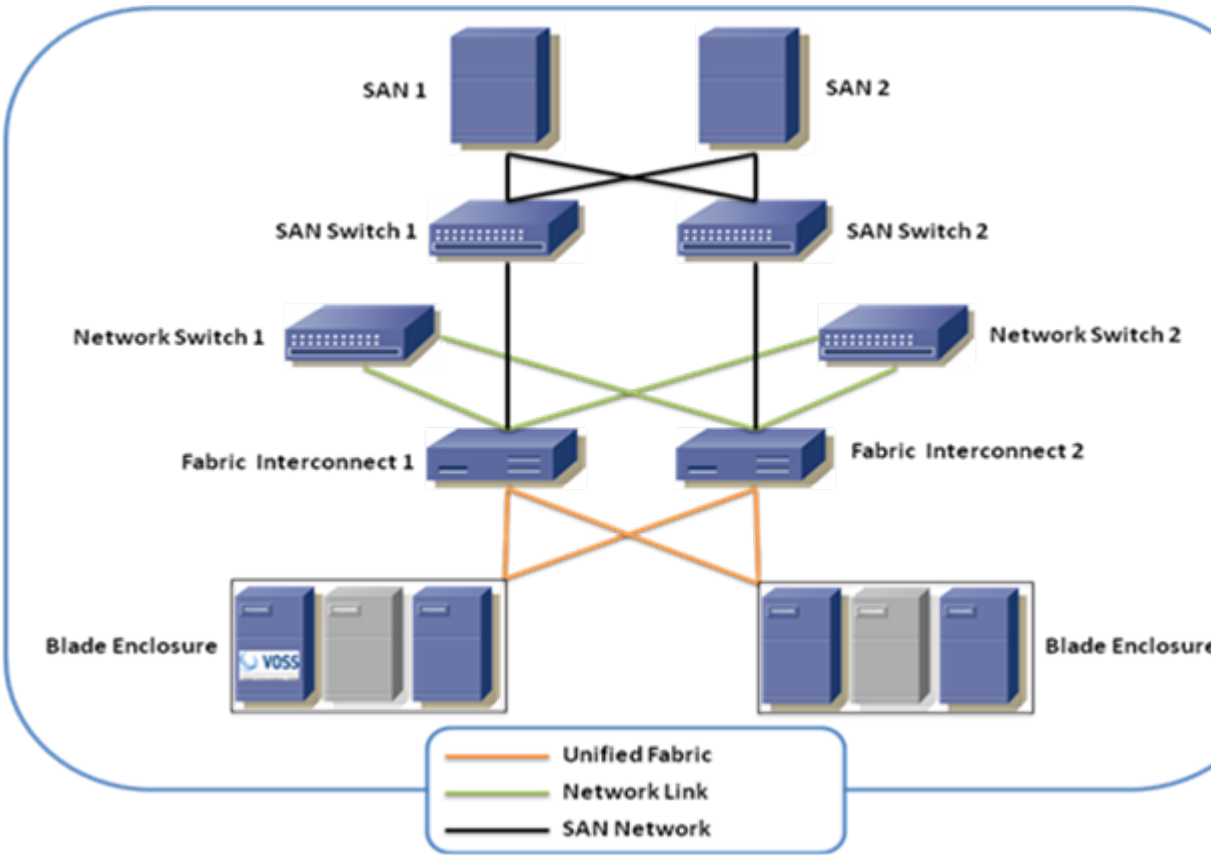
CUCDM-Server version 4 has been designed to run on the Cisco UCS platform. CUCDM runs as a guest within the UCS environment, with UCS itself being managed using Cisco tools such as the Cisco UCS Manager.

Only one instance of CUCDM runs within UCS at any one time. Should an error occur with the active CUCDM Server, UCS will migrate the CUCDM-Server either to another blade on the same enclosure or to another blade on a different enclosure.

The diagram below represents a simplified UCS environment running CUCDM. For more information on the Cisco UCS platform, please see the Cisco[1] website.

[1] http://www.cisco.com/en/US/netsol/ns944/index.html#~overview

Figure 1. CUCDM within a simplified UCS environment



CUCDM Hardware Requirements

Hardware Assumptions

CUCDM makes the following assumption regarding the UCS environment:

- Each site includes a replicated (redundant) SAN
- There are spare (redundant) SAN and Network switches
- There are multiple network routes (redundant) within the data centre
- There is spare (redundant) blade capacity
- Where possible, relevant resources such as power supplies and cooling fans are duplicated (redundant)

It is possible to run CUCDM without meeting the above assumptions; however, this may impact negatively on functionality such as disaster recovery and high availability. For more information on disaster recovery and high availability, please see *High Availability and Disaster Recovery* on page 10.

CUCDM Hardware Requirements

For exact hardware requirements, please see the *CUCDM Hardware Sizing* guide. It is important to note that CUCDM hardware requirements assume that CUCDM is the only application running on the blade, please ensure that hardware is suitably sized to cater for all applications that will be utilizing the resources.

Storage

Storage can be managed via UCS manager as part of a service profile. Storage via a SAN is the primary option.

Note

- CUCDM assumes that the SAN and VMWare tools provide High Availability within a site
- Storage via local SAS drives is currently not supported by CUCDM

Cisco's UCS Hardware Requirements

CUCDM is designed to run on the following UCS hardware:

- Cisco UCS 6100 Fabric Interconnect Switch
- Cisco UCS 5100 Chassis
- Cisco UCS B200M1 Half-width Blade Servers
- Fiber-channel SAN (Raid 10, 2 Controllers) - (provided by the service provider from the SAN vendor of their choice)

CHAPTER 3

High Availability and Disaster Recovery

High availability functionality is provided by the SAN (Storage Area Network), VSphere and UCS. CUCDM no longer provides embedded high availability capabilities as this is now provided by the virtualized hardware environment.

Disaster Recovery (DR) relies on there being geographically separate sites. At any one time, one site is active and the remaining site is in standby. Changes are communicated to the standby site via an asynchronous data connection.

Terminology

The following terminology is used in this section and within this industry:

- High Availability (HA): An approach to IT system design[1] and configuration that ensures a system is operational and accessible during a specified time frame. This is achieved via the use

[1] http://en.wikipedia.org/wiki/System_design

of redundant hardware and resources. If there is a failure, an automatic failover will occur to a second node.

- Disaster Recovery: Disaster Recovery capabilities differ from Automatic Failover. Disaster Recovery provides the capability for a business to recover and continue operating in the event of a major failure or event.

  These events may take various forms, including large scale power failure, fire, flood or any other event that removes a major part of the system from operation.

  Restoring service may require many things to happen, including both technical systems and human intervention and therefore cannot be automated. Major events are difficult to predict and therefore the recovery scenarios need to be planned in advance and operational procedures need to be put in place to deal with the most likely cases. Once the disaster is over, there may be multiple steps to restoring full service depending upon the scale and duration of the event.
- Redundancy: A system of using multiple and redundant resources to ensure that no single point of failure exists within a system.
- **Downtime:** Used to refer to a period when a system is not available to end-users.
- Recovery Time: The time it takes a business to recover from a disruption with IT services.
- **Automatic Failover:** Automatic Failover systems are designed to have multiple nodes where one or more nodes are "active". In the event of a node failing, the system is designed so that other nodes can automatically take over processing without any interruption or system downtime.

UCS High Availability Architecture

Due to the nature of UCS, CUCDM integrates heavily with the UCS HA functionality. The UCS HA functionality is outlined below.

UCS DR Overview

UCS Manager and Cisco fabric interconnects form the backbone of UCS. High Availability (HA) functionality is provided through the use of redundant fabric interconnects that are connected to form a cluster. Two instances of UCS manager are run on the fabric interconnects with the UCS managers communicating via a link between the two interconnects.

The UCS managers rely on the traditional active/standby architecture, in that at any one time, there is an active Manager, called the primary, and a standby manager, called the subordinate. Only the primary manager communicates with external entities. The main UCS configuration database is stored on the primary manager and replicated to the subordinate manager. The primary manager sends updates to the subordinate manager when configuration changes occur.

Both interconnects have private static IP address configured. During a failover scenario, either a DNS entry is updated to point to the subordinate node, or alternatively the IP address of the subordinate is updated. However, such IP address changes should be considered in the greater framework of the CUCDM system.

The two UCS manager instances are aware of each other via heart-beat message exchanges. When both interconnects are running, the configuration database remains in sync.

UCS Failover Scenarios

If the interconnect running as the primary fails, the subordinate manager will assume the role of the primary manager. The subordinate manager will have been in sync with the master manager

so minimal disruption will be experienced. The failed interconnect can now be repaired and then returned to service. After syncing with the currently active manager, the repaired manager can either remain as a spare (Subordinate) or it can resume the active (Primary) role.

UCS Failover Limitations

If the interconnect running as the primary fails, the subordinate manager interconnect will assume the role of the primary manager. When configuration changes are made on the subordinate interconnect, they will not be migrated over to the failed interconnect. If the subordinate interconnect also fails but the first failed fabric interconnect has returned to service but not yet synchronized, it will have the old copy of the configuration database. If this occurs, the active interconnect will check the configuration database version number of the other fabric interconnect.

If the other interconnects configuration database version number is higher it means that the other fabric interconnect has a more recent configuration. The active fabric interconnect will stop loading and will wait for administrator intervention.

When the communication link between two interconnects fail, the interconnects will use the blade chassis to communicate the heart-beat. If the interconnects receive proper heart-beat, the primary will stay as the primary and the subordinate will stay as the subordinate, the system will then enter a *Link-Failed* state. Only heart-beats will be exchanged and any update to the configuration database will not be synchronized to the subordinate.

CUCDM Disaster Recovery and High Availability on UCS

High Availability

To ensure high availability of CUCDM, the CUCDM-Server relies on the Automatic Failover functionality provided by UCS/VMware. Should an error occur, UCS will detect this and automatically move the CUCDM-Server onto a blade (server) that is operating correctly. End users will experience a momentary break in CUCDM functionality as the server is being migrated to an alternate blade (server). On average, failure of the CUCDM blade will be detected in 20 seconds and services will be restored on the new blade within 90 to 120 seconds. For more information on Cisco UCS DR functionality, please see Disaster Recovery on page 17 or the Cisco[2] website.

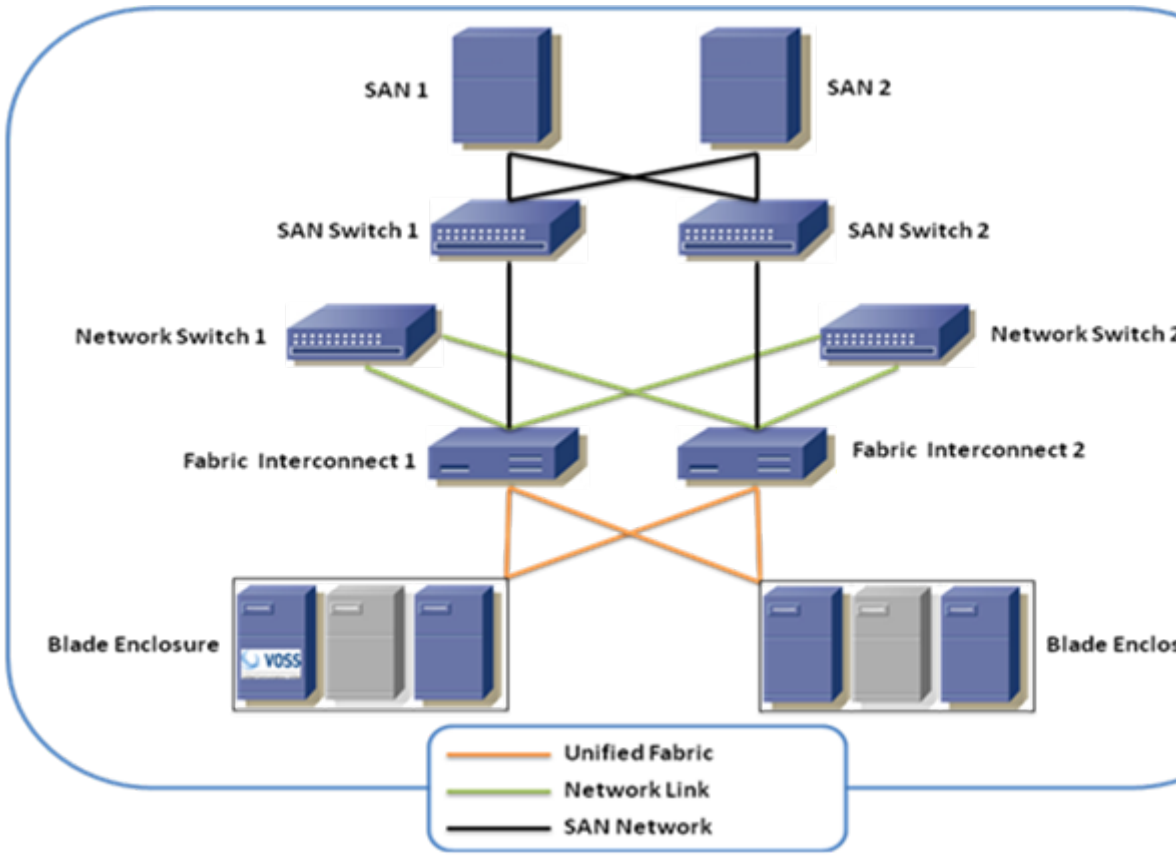
Note

Only a small subset of the CISCO UCS DR functionality is available within the UCS architecture as deployed within UC-UCS. Specifically this is limited to VMWare HA within a site.

An example of a high availability failover is summarized below.

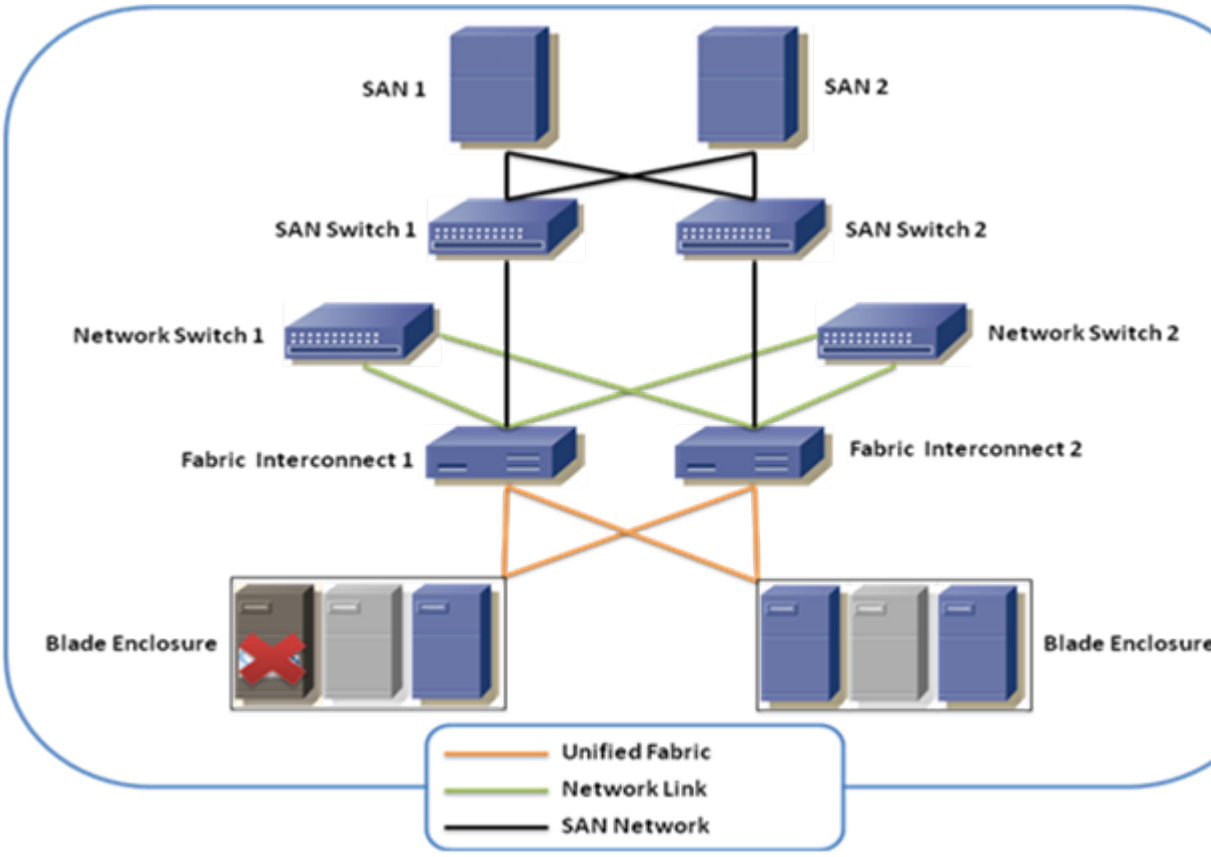
[2] http://www.cisco.com/en/US/netsol/ns944/index.html#~overview

Figure 2. CUCDM operating correctly within a simplified UCS environment.



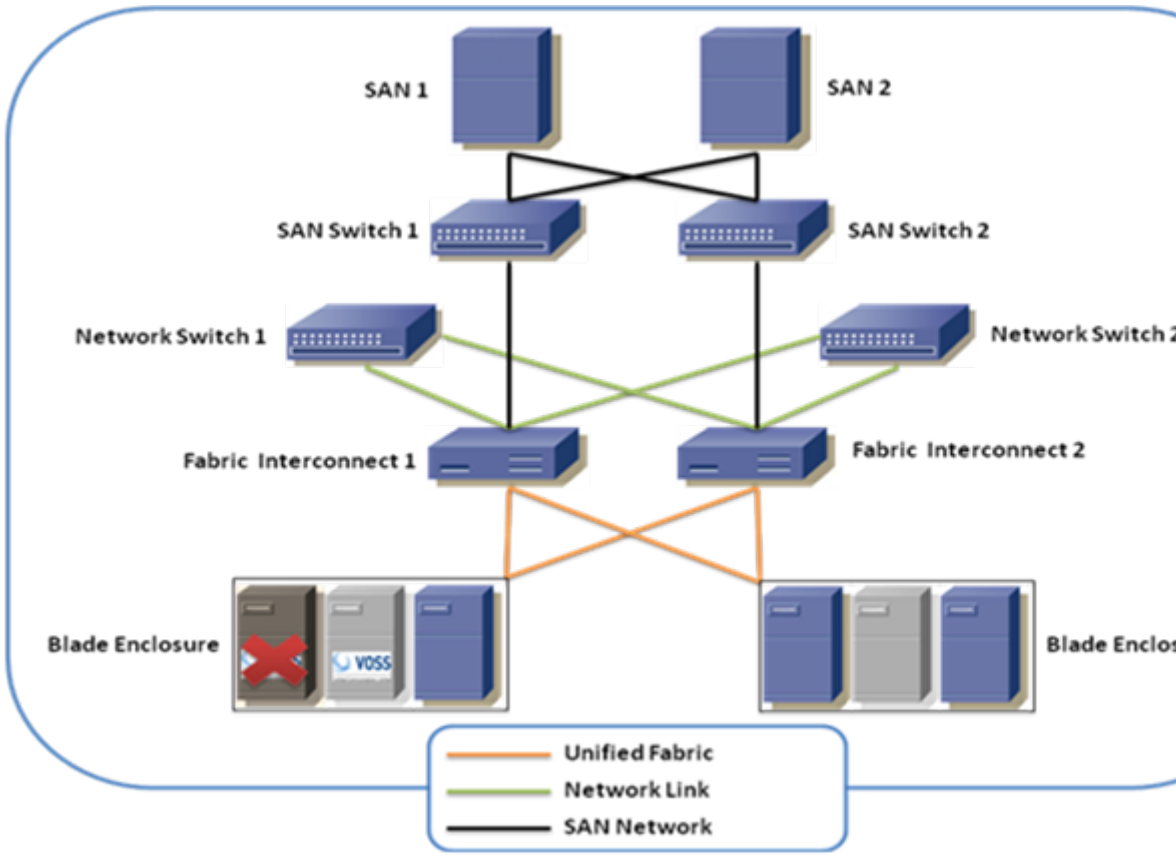
In the figure above, CUCDM is operating correctly and providing provisioning and other functionality to users. The grayed out blade adjacent to the CUCDM-Server is a spare blade currently not running any applications.

Figure 3. The blade running CUCDM has encountered a problem and is now out of service. A momentary lapse in CUCDM functionality will be experienced by end-users.



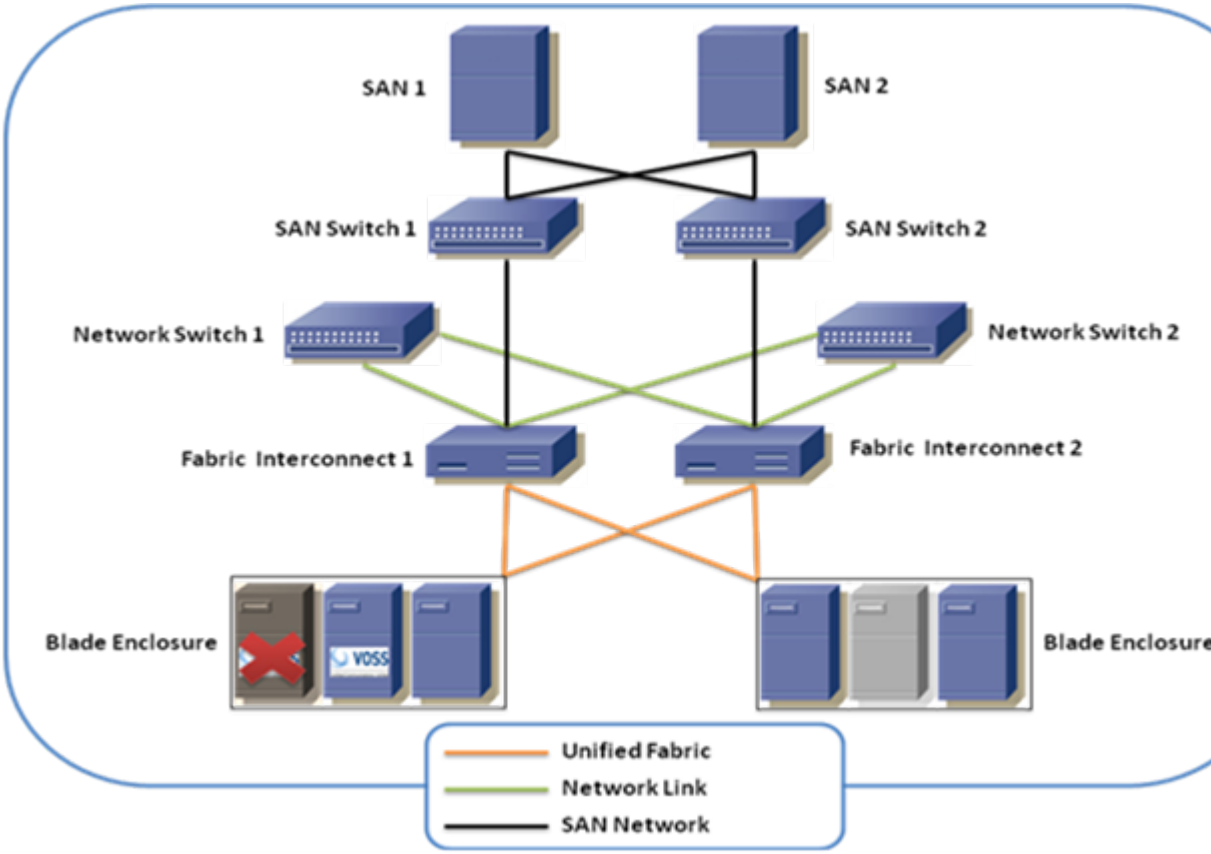
Users will experience a lapse in CUCDM functionality while the blade running CUCDM is out of service.

Figure 4. UCS has detected the failure of the blade and is currently migrating CUCDM over to a new blade



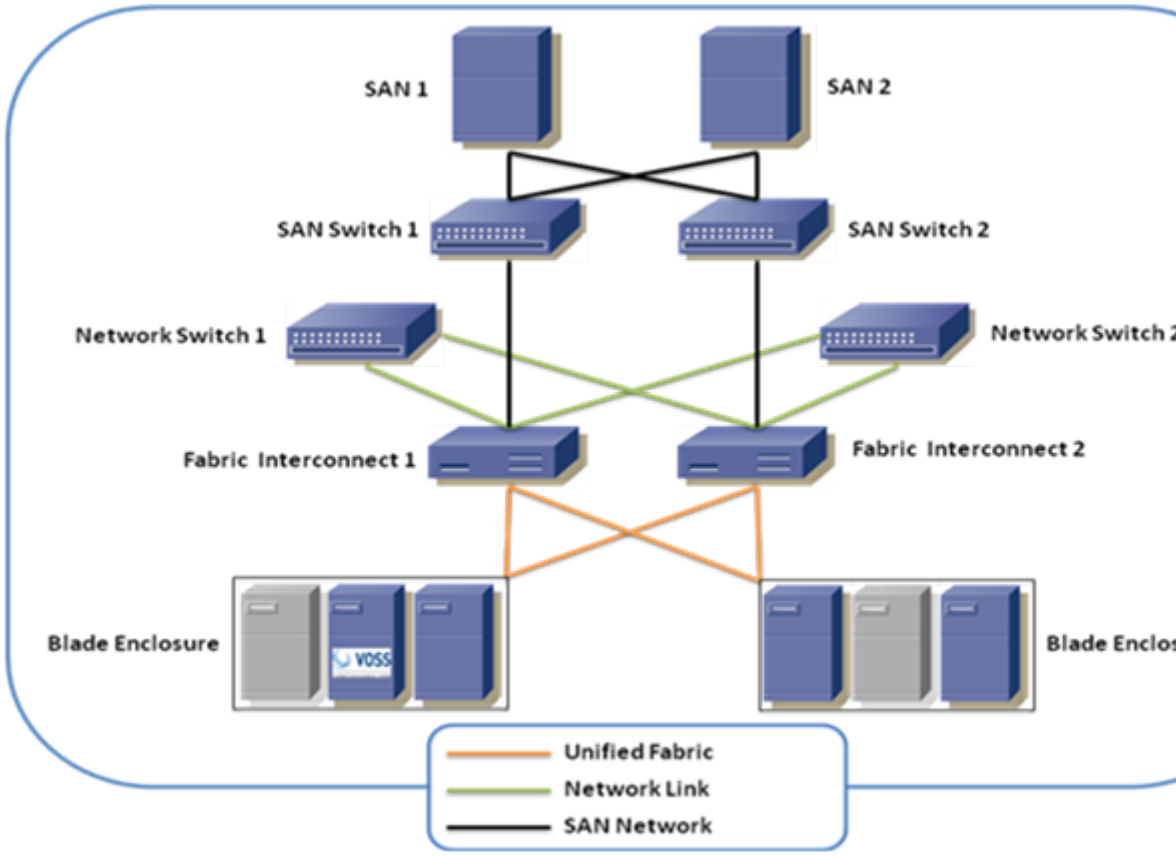
Once UCS has detected a hardware failure, an automatic failover will migrate CUCDM to a healthy blade.

Figure 5. UCS has successfully migrated CUCDM over to a new blade



CUCDM is now operating on a new blade and is back in a healthy state. Service to end-users will resume as usual. Technicians can now replace and/or repair the failed blade.

Figure 6. The failed blade has been repaired and has now been returned to the UCS resource pool as a spare blade



After the failed blade has been replaced or repaired, there is no need to failover CUCDM back to the original blade as the blade can simply be placed back in the UCS resource pool as a spare.

Disaster Recovery

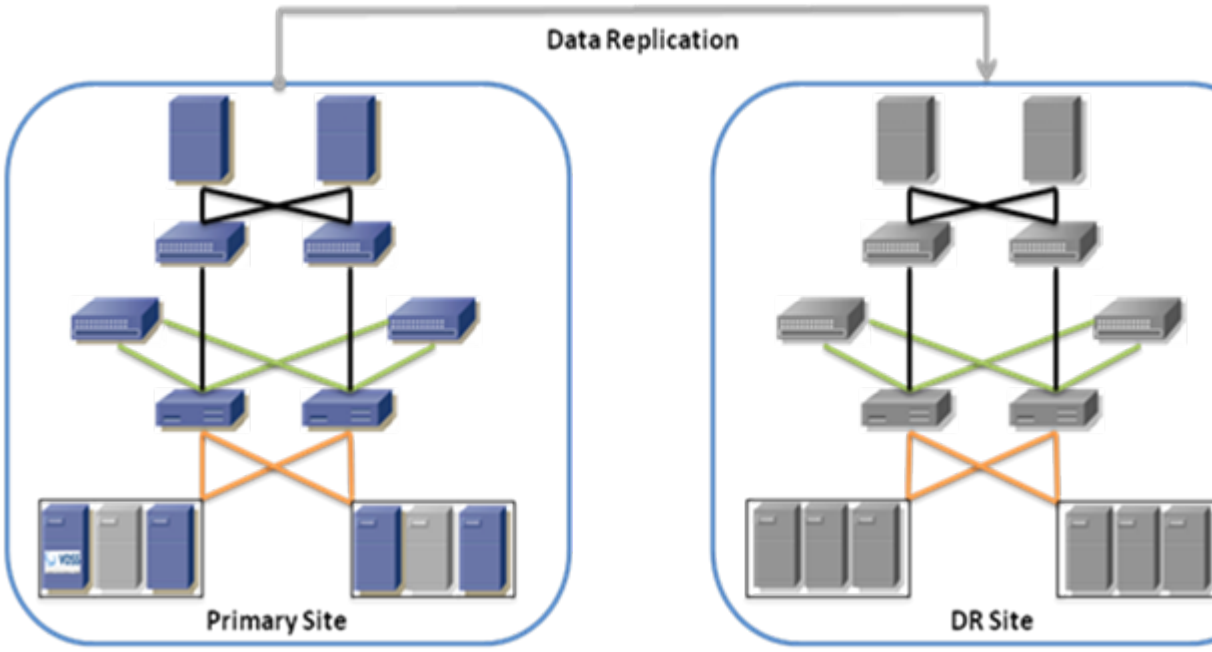
CUCDM-Server relies on multiple, geographically separate, sites for disaster recovery. At any one time, one site is active and the remaining site is in standby. Changes are communicated to the standby site via an asynchronous data connection.

Note

CUCDM disaster recovery functionality is not fully automatic, manual intervention is required during a fail-over.

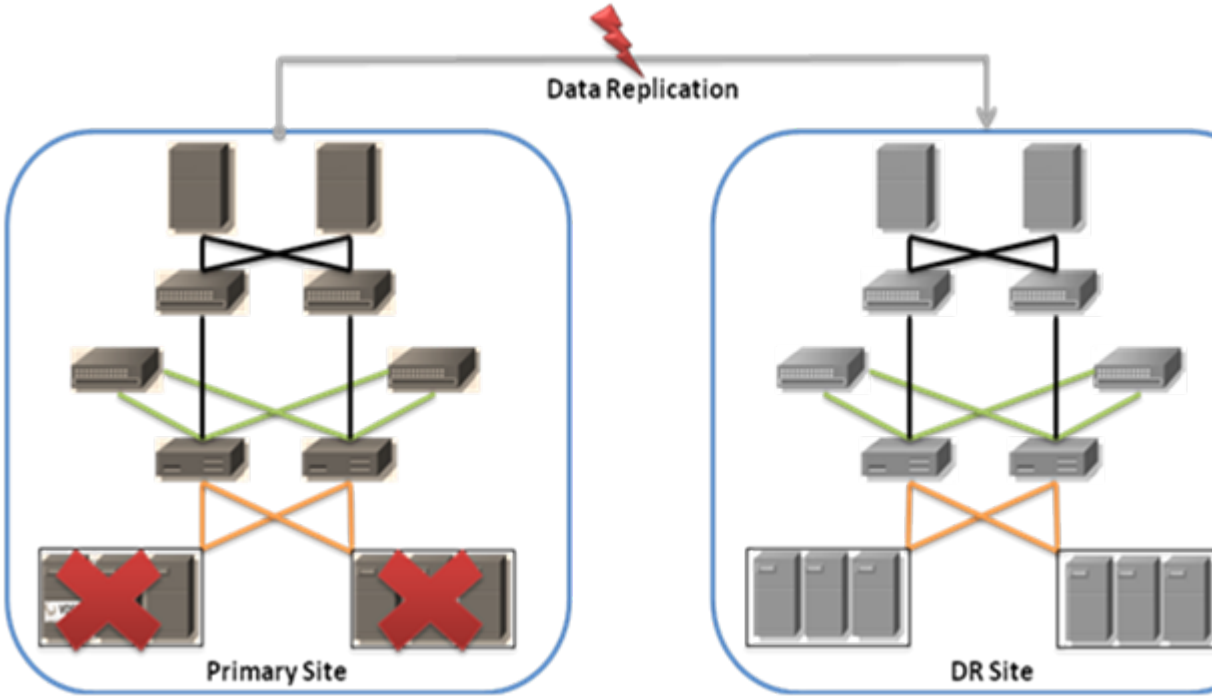
An example of a disaster recovery failover is illustrated below:

Figure 7. CUCDM operating correctly within a simplified UCS environment, changes made to the primary site are being replicated over to the DR site



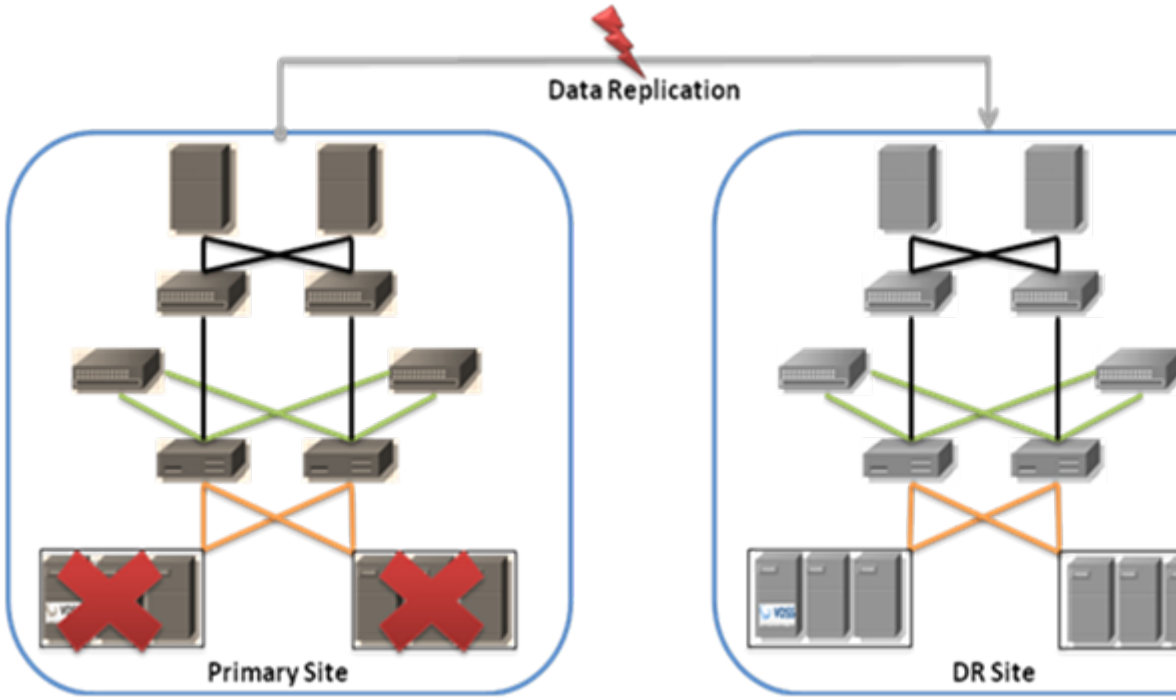
In the figure above, CUCDM is operating correctly and is providing provisioning and other functionality to users. All changes being made to CUCDM on the primary site are being replicated over to the DR site.

Figure 8. A catastrophic failure has occurred within the data center causing site 1 to fail



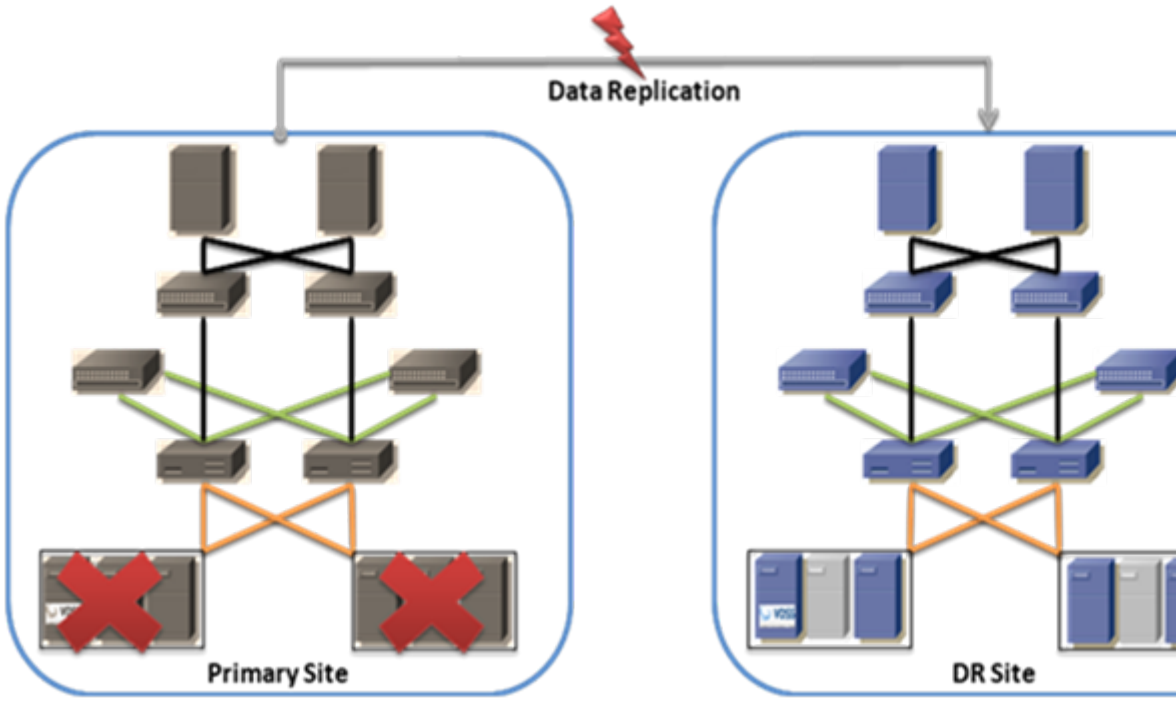
The site currently running CUCDM has encountered a problem and is now out of service. A lapse in CUCDM functionality will be experienced by end-users.

Figure 9. The failure has been detected and the fail-over process has been initiated



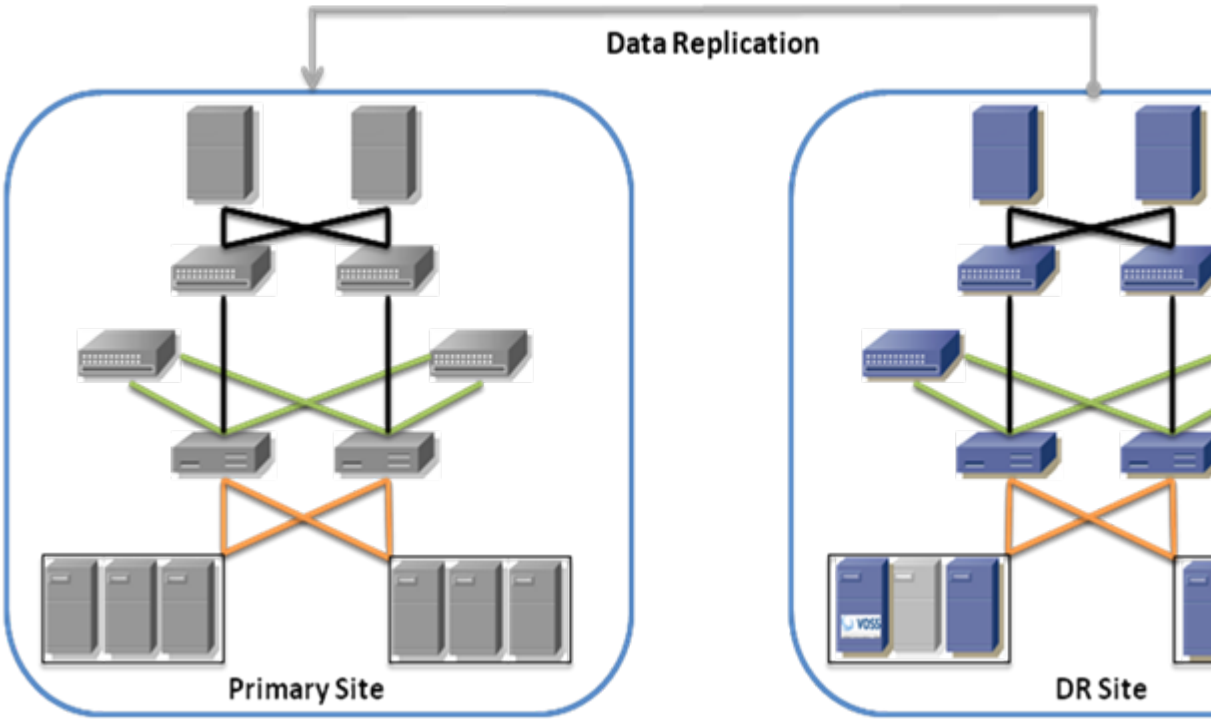
Once an issue has been detected, the fail-over process will begin. During this process, CUCDM will be migrated over to the second site.

Figure 10. CUCDM has successfully been migrated over to the second (DR) site



CUCDM is now operating on a new blade within the second (DR) site and is back in a healthy state. Technicians can now replace and/or repair the failed hardware within the primary site.

Figure 11. The failed hardware has been repaired and has now been returned to service



After the failed hardware has been replaced or repaired, technicians can manually fail-over the DR site back to the primary site.

Limitations and Known Issues

Software Errors

The DR process is not designed to cover situations where an error is present within the CUCDM software or database. This is due to the fact that these errors will also exist within the DR site.

Required Intervention

CUCDM DR requires intervention of the customer and/or CUCDM Support teams. CUCDM Support is required to support the shutdown and re-enablement of data replication.

Directory Services

In most CUCDM implementations, the phone directory is supported by CUCDM, that is, the phones access CUCDM directly to obtain the phone directory.

If configured on the phone using an explicit IP address, then in the event of a failure of the active server, the phone will lose access to the phone directory and the end-user will not be able to use the directory. The preferred mechanism therefore is for the phone directory to be provisioned as a URL rather than an explicit IP address. This mechanism requires a DNS server to exist within the network and be managed correctly so that in the event of disaster, the phones will access directory services via the DR site.

Manual Configuration Changes and Non-Database Related Data

Only changes made to the database are replicated and backed up. Changes such as software upgrades (Service Packs and Emergency Releases), configuration changes on the boxes and

other non-database related changes are not all replicated between the Active Site and the Disaster Recovery Site. Customers are required to have mechanisms in place to ensure any configuration and non-database related changes are effectively migrated between the two sites.

Data to be considered includes:

- DNS Configuration
- Any Manual Changes (i.e. changes to file content)
- Syslogs
- Bulk Loader Sheets

Delay in Synchronization

Data replication between the Disaster Recovery Site and the Active Site is not completed in real-time. At any one time, the lag between the Disaster Recovery Site and the Active site can be up to five minutes. This means that should the Active site fail, it is possible to permanently loose the last five minutes worth of transaction data.

Transactions that are "On the Fly"

Transactions that are being processed when a failure occurs will be lost. Depending on the type and stage of the transaction, certain elements of the transaction may not be rolled-back. When provisioning is re-established using the alternate site, transactions that were on the fly will not be completed, nor will any form of roll-back occur.

Bulk Loading Transactions that are "On the Fly"

Due to the fact that the bulk loading sheets will usually only be present on the active site, if a site fails while bulk loaders are being loaded, the sites may be in an unstable state. This is due to the fact that the Disaster Recovery site will have no way of knowing how far the bulk loader process was and what sheets had already been loaded on the active site.

Syslog Data that is "On the Fly"

Syslog data is not replicated between the two sites, however, due to the nature of syslog, any data that is "on the fly" within a site when a site goes down will be lost.

Autoregister Data

AutoRegister phone data is not replicated between the two sites, so, due to the nature of autoregister, when a site goes down, phones will need to re-establish a connection with the new active site. Autoregistration requests that are "on the fly" at the time of the site failure will be lost and will need to be re-sent once connection to the Disaster Recovery site is available.

Note

If the Disaster Recovery site is using a DHCP server that is configured differently to that of the active site, difficulties may be experienced with autoregistration. This will be due to the fact that phones will be expecting similar IP subnets and IP address ranges.

CHAPTER 4

Overview of Key Features and Functionality

Scalability

Scalability is provided by reconfiguring the Virtual machines' available RAM and CPUs. This enables hardware requirements to be determined based on the required performance and deployment size.

Installation Management

System configuration during installation is managed via a central installation script.

UCS Manager

For all Cisco virtualized customers on UCS, the UCS Manager will create and manage the environment, processors, memory and SAN (service profiles). CUCDM is capable of being deployed, preconfigured and ready to run in the created environment.

UCS manager also provides for inventory, monitoring, configuration, diagnostics, fault detect and statistics relating to the hardware and software.

Messaging and Alerting

Support for messaging and alerting can be handled by UCS Manager for hardware and associated applications. SNMP support is managed in a similar manner as hardware installations, however, CUCDM no longer requires hardware specific MIBs or traps.

Overview of Corosync

The Linux clustering communities have collaborated over the past few years to determine interfaces which allow clustered components from different sources to work together better. A single cluster membership layer is used for a whole cluster rather than having to run two separate "clusters" for node membership and clustered file system membership respectively. OpenAIS[1] is a cluster framework that implements the Service Availability Forum[2] Application Interface Specification. The Corosync cluster engine provides a set of plugins to manage service availability.

[1] http://www.openais.org/
[2] http://www.saforum.org/

On top of this pair of projects, the new architecture uses the Pacemaker[3] cluster resource manager, which is in turn a project started by extracting the original *crm* component form *heartbeat*. While Corosync architecturally sits at the top of the diagram, corosync is actually used to start the cluster (it, in turn, starts corosync as a plug-in). Corosync also starts the AIS checkpoint service, which acts as glue for a distributed lock manager used by clustered file systems.

The primary init script used is the init script which starts corosync. All other process management is carried out using Open Cluster Framework (OCF) Resource Agent (RA) scripts. These RAs live under the `/usr/lib/ocf64/resource.d` directory and are responsible for starting, stopping *and* monitoring applications. Cluster resources are configured in the pacemaker cluster information base (cib) using the *crm* command-line tool. An OCF script can be used to define what's called a *primitive*, which can be thought of as an instance of a service. These primitives can be grouped together into groups, which are implicitly ordered and have implicit linear dependencies within the group. Either primitives or groups can be cloned (run on more than one machine). Primitives, groups and clones can be given location constraints (where they should, must, should not or must not run), colocation constraints (which other resources they should, must, should not, must not run on the same node as) and ordering constraints (what order should services start up or shut down in, how serious the dependency is and whether or not it's a mandatory dependency, which affects service restart).

So, given the above facilities, the new architecture defined the CUCDM solution as a set of primitive services, in some cases grouped together into groups, and in a cluster, in some cases cloned. The appropriate dependencies are set up. At cluster startup, all services are started in the correct order. When services are stopped, any dependent services are first stopped. When the cluster is stopped, an orderly shutdown of all services is performed.

What needs to be explicitly stated is that these dependencies and orderings take place across *the whole cluster*. This means that, unlike the traditional architecture, process management on *engines* (an outdated concept) is performed by the cluster resource manager seamlessly.

So having coordinated start and stop, and cluster-wide process management, the next function that Corosync supports is monitoring. Each primitive has a monitoring schedule configured. If the OCF RA for that service instance reports that the service is not running the service will be restarted. This can be based on complex checks, such as actually logging into a database or message queue middleware layer. If a service fails, all dependent services will be stopped, followed by an orderly startup from that point onwards, ensuring that correct service is restored.

Location constraints were mentioned previously. In a connected cluster, these constraints could take into account connectivity (ping tests) or general node health (memory utilization, SMART disk status, anything that can be measured). This helps to provide optimal use of hardware while maintaining a high level or service.

In a clustered environment, Corosync also has the ability to perform node fencing using a mechanism called *STONITH* (an acronym for *Shoot The Other Node In The Head*). This allows the surviving nodes in a cluster to disable (or fence) any node which is unresponsive. There are a number of mechanisms which could be used for node fencing, such as remote power strips or IPMI.

IPMI is the Intelligent Platform Management Interface, a standards based mechanism for light-out management of hardware. Using IPMI, a cluster can connect to the baseboard management controlled (BMC) or an affected node and power cycle (or switch off) the node, ensuring that the node is in a known state.

The cluster resource manager includes some SNMP capability, where it can raise TRAP and INFORM messages to a monitoring station regarding cluster resource state changes and monitoring activity.

[3] http://www.clusterlabs.org/

To summarize, the platform offers a cluster-wide mechanism for managing and monitoring processes and resources. This mechanism includes self-healing capabilities and node fencing for data integrity. The solution integrates with existing monitoring solutions using SNMP.

CHAPTER 5

Effective User IDs

To enhance system security, a concerted effort is made to not run everything as root. In support of this, a number of built-in accounts have been included and named differently and the *root* account is disabled.

Note

Install, dhcpd, and sftp users can not be used to login. They can also not be edited or managed.

Root Account is Disabled

The platform installation and preparation steps create a user, *CUCDMAdm*, who can *sudo* (execute privileged operations) without a password. The administrator does not have raw access to the underlying platform, nor should any need arise requiring this. Instead, all configuration and monitoring can be performed from the CLI shell.

The web server user is *www-data* and the DHCP user is *dhcpd*.

Separate User Accounts

The following user accounts have been introduced:

Account	UID	Daemon
usmldr	1516	Loader Scheduler, Bulk Loader
usmplt	1517	Phone Login TImeout
iptdevmn	1518	iptdevmn (Device Manager)
iptqueue	1519	iptqueue (Child Queue)
iptparnt	1520	iptparent (Parent Queue)
usmbatch	1521	USM Batch (Python FastCGI Code)
usmsc	1522	USM Selfcare (Python FastCGI Code)

The user IDs listed above enable the isolation of processes in the solution so as to limit the impact of a security breach. It has the added benefit of making accurate process management easier.

CHAPTER 6

File System Layout

The file system layout used by the solution provides all services with their own mount points. This allows the distribution of services across all hosts.

The file system layout is:

Mount point	Service
/srv/VOSS/dhcp	ISC DHCP Daemon
/srv/VOSS/estraier	Full Text Search Engine (HyperEstraier)
/srv/VOSS/imq	OpenMQ Message Queue Broker
/srv/VOSS/pgsql	PostgreSQL RDBMS
/srv/VOSS/shared	Shared Filesystem

CHAPTER 7

No Internal DHCP

There is no internal DHCP. Nodes are statically allocated both an internal and external IP address.

CHAPTER 8

Log Files

The key logs can be found in `/var/log` but there are a few which are not stored in the `/var/log` directory.

Here is a list outlining the location and purpose of each of the log files:

Logfile	Purpose
/var/log/boot.log	Messages from processes that start up at system startup time.
/var/log/daemon.log	Message logged with a facility of daemon. Most software provided by the system logs to this file.
/var/log/dmesg	The kernel startup log.
/var/log/kern.log	Messages logged by the kernel.
/var/log/aptitude /var/log/apt/history /var/log/dpkg.log	Packaging tools logs. Of particular interest is /var/log/apt/history, which records changes to package installations.
/var/log/messages	Most system messages, not including mailer information and a few other noisy sources.
/var/log/syslog	All messages logged via syslog.
/var/log/user.log	All messages logged with a facility of USER.
/var/log/usm_batch/*	Messages logged by the CUCDM Python Batch code.
/var/log/usm_selfcare/*	Messages logged by the CUCDM Selfcare.
/var/log/nginx/access.log	External access to the system via HTTP or HTTPs.
/var/log/nginx/error.log	Errors encountered by the SSL decapsulating reverse proxy.
/srv/VOSS/estraier/data/_log	Full text search engine logs.
/srv/VOSS/imq/instances/CUCDMbkr/log/log.txt	OpenMQ Message Queue broker logs.
/srv/VOSS/pgsql/pgsql/data/pg_ctl.log	PostgreSQL control utility logs.
/srv/VOSS/shared/usm/logs/bulkload/*	Bulk load scheduler logs.
/srv/VOSS/shared/usm/logs/phone_login_timeout/phone_login_timeout.log	Phone login timeout daemon logs.
/srv/VOSS/shared/usm/bulkload/log files/*	Bulk load job log files.
/var/log/usm	Messages logged by USM.
/var/log/cucdm	Messages logged by platform.

CHAPTER 9

Autoregister Changes

The auto registration process has been redeveloped to address all of the issues recently raised against auto register. Auto register has been split into two processes, networked and local auto register.

The networked auto register program listens for syslog messages on UDP port 514. The daemon contains three processes which communicate using shared memory queues. The main process listens to the network socket, which it sets up with a large receive buffer.

It then passes received messages to the analysis and submission process, which extracts the relevant auto registration data from the message and initiates an action against CUCDM based on the contents of the message, or drops the message if needs be. If the message from that MAC address or unique device name has already been seen, the message is ignored. The analysis and submission process then registers the CUCDM request ID with the completion and timeout process, which maintains a list of outstanding request IDs. At a configurable interval, the completion and timeout process polls the outstanding message IDs for their completion statuses and checks for request IDs which have expired (been outstanding for longer than the retry limit timeout). Request IDs which have completed or timed out are sent back to the analysis and submission process so that they can be removed from the ignore caches.

The local auto register process only processes messages which come from the local DHCP server. It monitors syslog for the appropriate DHCP messages and forwards them to the networked auto register process. This process could easily be replaced with a set of syslog filters in the future.

CHAPTER 10

Load Balancing and Web Server Structure

All Apache served content is available only within the cluster, and only on HTTP. The Apache servers are placed behind an (internal) load balancer, called HAProxy, which distributes load across the servers. HAProxy has node weighting dynamically adjusted for it using a daemon (which we wrote) called ldfd-haproxy (Load Directed Feedback Daemon). This daemon has configuration pushed at it from all nodes in the cluster by a process called ldfc-haproxy (Load Directed Feedback Client).

On top of the load balancer, we place Nginx, an extremely scalable lightweight web server. Nginx is configured as an SSL decapsulating reverse proxy. It also serves static content directly from the edge of the cluster, taking that load off of the Apache processes. Since Apache now only interacts with its load balancer, which interacts with Nginx, Apache returns user content at LAN speeds, which is then returned to the user at WAN speeds. The Apache processes are immediately freed up to perform more work, aiding scalability. The use will see large, incrementally rendered, pages appear to load more quickly, as the incremental rendering is buffered by the reverse proxy before being served to the user.

CHAPTER 11

Device Manager, IPT Queue and IPT Parent Changes

While there have been no code changes to the middleware, the introduction of the load balancer discussed above has changed the deployment model of the three middleware processes. The cluster now only runs one instance of each of the middleware daemons. The daemons are configured to call the backend code via the load balancer, which distributes the requests in a weighted round robin fashion. This changes a few notable things. Previously, the device manager and the work it requested only ever ran on the active director, this workload is now handled by the whole cluster. Previously, whichever middleware process consumed a message was responsible for running that request on its own host. This has now changed so that the request is run on the most suitable host.

This means, configuration-wise, that the max_parent_requests parameter now reflects the maximum number of concurrent parent requests that will be executed cluster-wide.

CHAPTER 12

Java Process Overview

Since the cluster resource manager supports complex process health monitoring the Java daemons are simply started using the start-stop-daemon wrapper and monitored by the cluster resource manager.

In line with the above, the automatic restart behavior of the OpenMQ broker allows the cluster resource manager to manage the process.

CHAPTER 13

SNMP

For further information, please see the *CUCDM SNMP Guide*.

CHAPTER 14

Command Line Interface (CLI)

For further information, please see the *CUCDM CLI Guide.*

CHAPTER 15

FAQ

UCS

What is Cisco Unified Computing System?

Cisco Unified Computing System (UCS) combines physical and virtual computing and network resources into a single system. UCS is a set of pre-integrated data center components that comprises blade servers, adapters, fabric interconnects and extenders that are integrated under a common management system.

What are UCS Blade Servers and Rack Servers?

UCS Blade Servers reside in the UCS chassis and are managed using UCS manager. UCS Rack Servers work independently and do not require the full UCS infrastructure. However, rack servers can be integrated into existing UCS infrastructure.

CUCDM

What Base Operating System Does CUCDM Use?

CUCDM-Server version 4.0 is based on Ubuntu 10.04 LTS (Lucid Lynx) and is designed to run in a virtual environment with UCS. UCS is a computing architecture incorporating a next-generation networking and I/O fabric (all from Cisco) running VMware and managed by a lightly modified VSphere installation.

Does CUCDM Download Software or Software Updates from the Internet?

No

In What Ways can CUCDM Software Updates be obtained?

CUCDM support will provide any required updates. These updates will be transferred from a CUCDM repository.

How often does CUCDM need Updating?

CUCDM operates on a quarterly release schedule. Customer deployed CUCDM environments only need to be updated if the customer requires new features/enhancements.

What are the Hardware Requirements for CUCDM?

Hardware requirements will vary based on the type and size of deployment.For detailed platform hardware requirements, please refer to the *CUCDM-Server Hardware Requirements* fact sheet. This document is available from your dedicated CUCDM account manager.

Does CUCDM require the use of Gigabit Ethernet Connections?

For sustained performance, CUCDM requires gigabit connections for the internal database communication. Each server within the cluster will require 2 gigabit connections. Two of the servers will require an additional gigabit Ethernet connection for external connectivity.

What Network Protocols are used by CUCDM?

UDP/IP: Used for DNS lookups, SNMP, NTP and SYSLOG (where appropriate).

TCP/IP: This is used for the following:

- HTTP and HTTPS access both to the CUCDM cluster and for certain provisioning tasks
- SSH, SCP and SFTP to network elements
- Telnet to network elements
- Active/passive nodes for block device replication
- Between sites for database replication

IP multicast: This is used within the cluster for cluster node communications and membership.

What Ports does CUCDM Use?

Telephony UI: 80

Administration and Self Care: 443

Management, monitoring, operations and integration: 22, 67, 123, 161, possibly others depending on deployed network elements and integration requirements.

Does CUCDM Require the Use of a Virtual Private Network (VPN)?

Yes, CUCDM support requires VPN access in order to maintain SLA's, conduct upgrades, and for troubleshooting purposes.

CHAPTER 16

Related Documentation

For further information, please refer to the following documents:

- CUCDM-Server Command Line Interface Guide
- CUCDM SNMP Guide
- CUCDM Redundancy and Disaster Recovery Guide

CHAPTER 17

Support

For support related queries, please contact your dedicated CUCDM support person.