Cisco Virtualization Solution for EMC VSPEX with Microsoft Hyper-V 2012 R2 for up to 1000 Virtual Machines
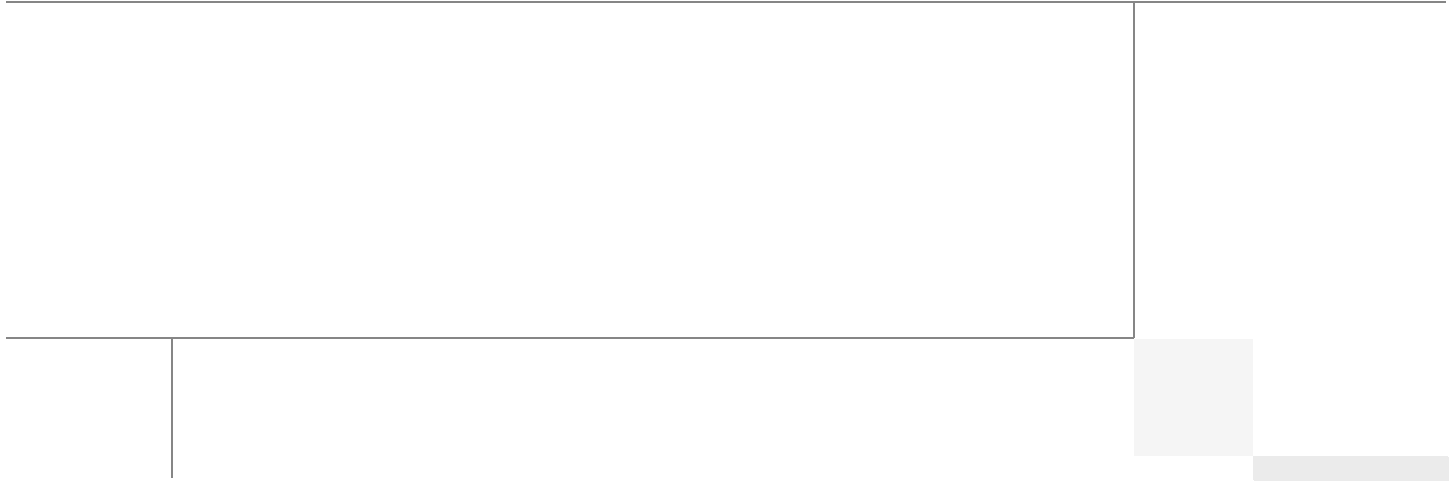
Last Updated: February 3, 2014

Cisco Validated Design

Building Architectures to Solve

About the Authors

Tim Cerling, Technical Marketing Engineer, Cisco

Tim Cerling is a Technical Marketing Engineer with Cisco's Datacenter Group, focusing on delivering customer-driven solutions on Microsoft Hyper-V and System Center products. Tim has been in the IT business since 1979. He started working with Windows NT 3.5 on the DEC Alpha product line during his 19 year tenure with DEC, and he has continued working with Windows Server technologies since then with Compaq, Microsoft, and now Cisco. During his twelve years as a Windows Server specialist at Microsoft, he co-authored a book on Microsoft virtualization technologies - Mastering Microsoft Virtualization. Tim holds a BA in Computer Science from the University of Iowa.

Prashanto Kochavara, Solutions Engineer, EMC

Prashanto has been working for the EMC solutions group for over 3 years. Prashanto is a SME on EMC Storage and Virtualization technologies including VMware and Hyper-V. He has vast amount of experience in end-to-end solution planning and deployments of VSPEX architectures. Prior to joining the solutions group at EMC, Prashanto has interned as a Systems Engineer (EMC) and Software Developer (MBMS Inc.). Prashanto holds a Bachelor's degree in Computer Engineering from SUNY Buffalo and will be graduating with a Master's Degree in Computer Science from North Carolina State University in December 2013.

Acknowledgments

For their support and contribution to the design, validation, and creation of this Cisco Validated Design, we would like to thank:

- Mike Mankovsky—Technical Lead, Cisco
- Mehul Bhatt—Technical Lead, Cisco

About Cisco Validated Design (CVD) Program

The CVD program consists of systems and solutions designed, tested, and documented to facilitate faster, more reliable, and more predictable customer deployments. For more information visit www.cisco.com/go/designzone.

ALL DESIGNS, SPECIFICATIONS, STATEMENTS, INFORMATION, AND RECOMMENDATIONS (COLLECTIVELY, "DESIGNS") IN THIS MANUAL ARE PRESENTED "AS IS," WITH ALL FAULTS. CISCO AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE. IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THE DESIGNS, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

THE DESIGNS ARE SUBJECT TO CHANGE WITHOUT NOTICE. USERS ARE SOLELY RESPONSIBLE FOR THEIR APPLICATION OF THE DESIGNS. THE DESIGNS DO NOT CONSTITUTE THE TECHNICAL OR OTHER PROFESSIONAL ADVICE OF CISCO, ITS SUPPLIERS OR PARTNERS. USERS SHOULD CONSULT THEIR OWN TECHNICAL ADVISORS BEFORE IMPLEMENTING THE DESIGNS. RESULTS MAY VARY DEPENDING ON FACTORS NOT TESTED BY CISCO.

Cisco Virtualization Solution for EMC VSPEX with Microsoft Hyper-V 2012 R2 for up to 300 Virtual Machines

Executive Summary

Cisco solution for EMC VSPEX is a pre-validated and modular architecture built with proven best-of-breed technologies to create and complete an end-to-end virtualization solution. The end-to-end solutions enable you to make an informed decision while choosing the hypervisor, compute, storage, and networking layers. VSPEX minimizes the server virtualization planning and configuration burdens. The VSPEX infrastructures accelerate your IT Transformation by enabling faster deployments, greater flexibility of choice, efficiency, and lower risk. This Cisco Validated Design document focuses on the Microsoft Windows Server 2012 R2 Hyper-V architecture for up to 300 virtual machines with Cisco solution for the EMC VSPEX.

Introduction

Virtualization is a key and critical strategic deployment model for reducing the Total Cost of Ownership (TCO) and achieving better utilization of platform components like hardware, software, network, and storage. However, choosing an appropriate platform for virtualization can be challenging. Virtualization platforms should be flexible, reliable, and cost effective to facilitate the deployment of various enterprise applications. In a virtualization platform for compute, network, and storage resources to be used effectively, the ability to slice and dice the underlying platform is essential to size the application requirements. The Cisco solution for the EMC VSPEX provides a very simplistic yet fully integrated and validated infrastructure to deploy VMs in various sizes to suit various application needs.

Target Audience

The reader of this document is expected to have the necessary training and background to install and configure Microsoft Windows Server 2012 R2, EMC VNX5400, Cisco Nexus 5548UP switch, and Cisco Unified Computing (UCS) B200 M3 Blade Servers. External references are provided wherever applicable and it is recommended that the reader be familiar with these documents.

Readers are also expected to be familiar with the infrastructure and database security policies of the customer installation.

Corporate Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Purpose of This Guide

This document describes the overall design goals of the Cisco solution for EMC VSPEX for Microsoft Hyper-V. The document provides an architectural description of the solution for providing up to 300 virtual machines.

Business Needs

The VSPEX solutions are built with proven best-of-breed technologies to create complete virtualization solutions that enable you to make an informed decision in the hypervisor, server, and networking layers. The VSPEX infrastructures accelerate your IT transformation by enabling faster deployments, greater flexibility of choice, efficiency, and lower risk.

Business applications are moving into the consolidated compute, network, and storage environment. The Cisco solution for the EMC VSPEX using Microsoft Hyper-V reduces the complexity of configuring every component of a traditional deployment model. The complexity of integration management is reduced while maintaining the application design and implementation options. Administration is unified, while process separation can be adequately controlled and monitored. The following are the business needs for the Cisco solution for EMC VSPEX using Microsoft Hyper-V architectures:

- Provide an end-to-end virtualization solution to utilize the capabilities of the unified infrastructure components.
- Provide a Cisco VSPEX for Microsoft Hyper-V ITaaS solution for efficiently virtualizing up to 300 virtual machines for varied customer use cases.
- Provide a reliable, flexible, and scalable reference design.

Solution Overview

The Cisco solution for EMC VSPEX using Microsoft Windows Server 2012 R2 provides an end-to-end architecture with Cisco, EMC, and Microsoft technologies that demonstrate support for up to 300, 600, or 1000 generic virtual machines and provide high availability and server redundancy.

The following are the components used for the design and deployment:

- Cisco UCS B-series Unified Computing System servers
- Cisco UCS 5108 Chassis
- Cisco UCS 2204XP Fabric Extenders
- Cisco UCS 6248UP Fabric Interconnects
- Cisco Nexus 5548UP Switches
- Cisco virtual Port Channels for network load balancing and high availability
- EMC VNX5400 (300 VMs), VNX5600 (600 VMs), or VNX5800 (1000) storage array
- Microsoft Active Directory (provided by customer)
- Microsoft System Center Virtual Machine Manager (provided by customer)

The solution is designed to host scalable and mixed application workloads. The scope of this Cisco Validated Design document is limited to the Cisco solution for EMC VSPEX Microsoft Hyper-V solutions for up to 300 virtual machines only. When the base configuration is in place, it is a simple matter of adding more servers or storage to handle the larger workloads.

Table 1   Components for up to 300, 600, or 1000 Virtual Machines

Solution	300 VMs	600 VMs	1,000 VMs
Software	Windows Server 2012 R2 with Hyper-V		
Compute	1 Cisco UCS 5108 Blade Server Chassis with: - 6 Cisco UCS B200 Blade Servers, each with 1 Cisco UCS Virtual Interface Card (VIC) 1240 and 192 GB RAM	2 Cisco UCS 5108 Blade Server Chassis with: - 12 Cisco UCS B200 Blade Servers, each with 1 Cisco UCS Virtual Interface Card (VIC) 1240 and 192 GB RAM	3 Cisco UCS 5108 Blade Server Chassis with: - 18 Cisco UCS B200 Blade Servers, each with 1 Cisco UCS Virtual Interface Card (VIC) 1240 and 192 GB RAM
Networking	· 2 Cisco UCS 6248UP 48-port Fabric Interconnects · 2 Cisco Nexus 5548UP Switches · 1 Cisco Nexus 1000V (Optional)	· 2 Cisco UCS 6248UP 48-port Fabric Interconnects · 2 Cisco Nexus 5548UP Switches · 1 Cisco Nexus 1000V (Optional)	· 2 Cisco UCS 6248UP 48-port Fabric Interconnects · 2 Cisco Nexus 5548UP Switches · 1 Cisco Nexus 1000V (Optional)
Storage	EMC VNX5400 Storage System with: - 2 storage controllers - Redundant Fibre Channel modules - 6 x 200 GB flash drives (FAST VP) - 1 x 200 GB flash drive (hot spare) - 110 x 600 GB SAS drives (virtual servers) - 4 x 600 GB SAS drives (hot spares)	EMC VNX5600 Storage System with: - 2 storage controllers - Redundant Fibre Channel modules - 10 x 200 GB flash drives (FAST VP) - 1 x 200 GB flash drive (hot spare) - 220 x 600 GB SAS drives (virtual servers) - 8 x 600 GB SAS drives (hot spares)	EMC VNX5800 Storage System with: - 2 storage controllers - Redundant Fibre Channel modules - 16 x 200 GB flash drives (FAST VP) - 1 x 200 GB flash drive (hot spare) - 360 x 600 GB SAS drives (virtual servers) - 12 x 600 GB SAS drives (hot spares)

Technology Overview

Cisco Unified Computing System

The Cisco Unified Computing System is a next-generation data center platform that unites compute, network, and storage access. The platform, optimized for virtual environments, is designed using open industry-standard technologies and aims to reduce total cost of ownership (TCO) and increase business agility. The system integrates a low-latency; lossless 10 Gigabit Ethernet unified network fabric with enterprise-class, x86-architecture servers. It is an integrated, scalable, multi chassis platform in which all resources participate in a unified management domain.

The main components of Cisco Unified Computing System are:

- Computing - the system is based on an entirely new class of computing system that incorporates blade servers based on Intel Xeon E5-2600 Series Processors.

- Network - the system is integrated onto a low-latency, lossless, 10-Gbps unified network fabric. This network foundation consolidates LANs, SANs, and high-performance computing networks which historically have been separate networks. The unified fabric lowers costs by reducing the number of network adapters, switches, and cables, and by decreasing the power and cooling requirements.

- Virtualization - the system unleashes the full potential of virtualization by enhancing the scalability, performance, and operational control of virtual environments. Cisco security, policy enforcement, and diagnostic features are now extended into virtualized environments to better support changing business and IT requirements.

- Storage access - the system provides consolidated access to both SAN storage and Network Attached Storage (NAS) over the unified fabric. By unifying the storage access, the Cisco Unified Computing System can access storage over Ethernet, Fibre Channel, Fibre Channel over Ethernet (FCoE), iSCSI, and SMB 3.0. This provides customers with choice for storage access and investment protection. In addition, the server administrators can pre-assign storage-access policies for system connectivity to storage resources, simplifying storage connectivity and management for increased productivity.

- Management - the system uniquely integrates all system components to enable the entire solution to be managed as a single entity by the Cisco UCS Manager. The Cisco UCS Manager has an intuitive graphical user interface (GUI), a command-line interface (CLI), and a powerful scripting library module for Microsoft PowerShell built on a robust application programming interface (API) to manage all system configuration and operations.

The Cisco Unified Computing System is designed to deliver:

- A reduced Total Cost of Ownership and increased business agility.

- Increased IT staff productivity through just-in-time provisioning and mobility support.

- A cohesive, integrated system which unifies the technology in the data center. The system is managed, serviced, and tested as a whole.

- Scalability through a design for hundreds of discrete servers and thousands of virtual machines and the capability to scale I/O bandwidth to match demand.

- Industry standards supported by a partner ecosystem of industry leaders.

Cisco UCS Manager

Cisco UCS Manager provides unified, embedded management of all software and hardware components of the Cisco Unified Computing System through an intuitive GUI, a command line interface (CLI), a Microsoft PowerShell module, or an XML API. The Cisco UCS Manager provides unified management domain with centralized management capabilities and controls multiple chassis and thousands of virtual machines.

Cisco UCS Fabric Interconnect

The Cisco® UCS 6200 Series Fabric Interconnect is a core part of the Cisco Unified Computing System, providing both network connectivity and management capabilities for the system. The Cisco UCS 6200 Series offers line-rate, low-latency, lossless 10 Gigabit Ethernet, Fibre Channel over Ethernet (FCoE), and Fibre Channel functions.

The Cisco UCS 6200 Series provides the management and communication backbone for the Cisco UCS B-Series Blade Servers and Cisco UCS 5100 Series Blade Server Chassis. All chassis, and therefore all blades, attached to the Cisco UCS 6200 Series Fabric Interconnects become part of a single, highly available management domain. In addition, by supporting unified fabric, the Cisco UCS 6200 Series provides both the LAN and SAN connectivity for all blades within its domain.

From a networking perspective, the Cisco UCS 6200 Series uses a cut-through architecture, supporting deterministic, low-latency, line-rate 10 Gigabit Ethernet on all ports, 1 Tb switching capacity, 160 Gbps bandwidth per chassis, independent of packet size and enabled services. The product family supports Cisco low-latency,

lossless 10 Gigabit Ethernet unified network fabric capabilities, which increase the reliability, efficiency, and scalability of Ethernet networks. The Fabric Interconnect supports multiple traffic classes over a lossless Ethernet fabric from a blade server through an interconnect. Significant TCO savings come from an FCoE-optimized server design in which network interface cards (NICs), host bus adapters (HBAs), cables, and switches can be consolidated.

The Cisco UCS 6248UP 48-Port Fabric Interconnect is a one-rack-unit (1RU) 10 Gigabit Ethernet, FCoE and Fibre Channel switch offering up to 960-Gbps throughput and up to 48 ports. The switch has 32 1/10-Gbps fixed Ethernet, FCoE, and FC ports and one expansion slot.

Figure 1   Cisco UCS 6248UP Fabric Interconnect

Cisco UCS Fabric Extenders

The Cisco UCS 2200 Series Fabric Extenders multiplex and forward all traffic from blade servers in a chassis to a parent Cisco UCS fabric interconnect over from 10-Gbps unified fabric links. All traffic, even traffic between blades on the same chassis or virtual machines on the same blade, is forwarded to the parent interconnect, where network profiles are managed efficiently and effectively by the fabric interconnect. At the core of the Cisco UCS fabric extender are application-specific integrated circuit (ASIC) processors developed by Cisco that multiplex all traffic.

The Cisco UCS 2204XP Fabric Extender has four 10 Gigabit Ethernet, FCoE-capable, SFP+ ports that connect the blade chassis to the fabric interconnect. Each Cisco UCS 2204XP has sixteen 10 Gigabit Ethernet ports connected through the midplane to each half-width slot in the chassis. Typically configured in pairs for redundancy, two fabric extenders provide up to 80 Gbps of I/O to the chassis.

Figure 2   Cisco UCS 2204XP Fabric Extender


Cisco UCS Blade Chassis

The Cisco UCS 5100 Series Blade Server Chassis is a crucial building block of the Cisco Unified Computing System, delivering a scalable and flexible blade server chassis.

The Cisco UCS 5108 Blade Server Chassis is six rack units (6RU) high and can mount in an industry-standard 19-inch rack. A single chassis can house up to eight half-width Cisco UCS B-Series Blade Servers and can accommodate both half-width and full-width blade form factors.

Four single-phase, hot-swappable power supplies are accessible from the front of the chassis. These power supplies are 92 per cent efficient and can be configured to support non-redundant, N+ 1 redundant, and grid-redundant configurations. The rear of the chassis contains eight hot-swappable fans, four power connectors (one per power supply), and two I/O bays for Cisco UCS 2204XP Fabric Extenders.

A passive mid-plane provides up to 40 Gbps of I/O bandwidth per server slot and up to 80 Gbps of I/O bandwidth for two slots. The chassis is capable of supporting future 40 Gigabit Ethernet standards. The Cisco UCS Blade Server Chassis is shown in Figure 3.

Figure 3   Cisco UCS 5108 Blade Server Chassis (back and front)
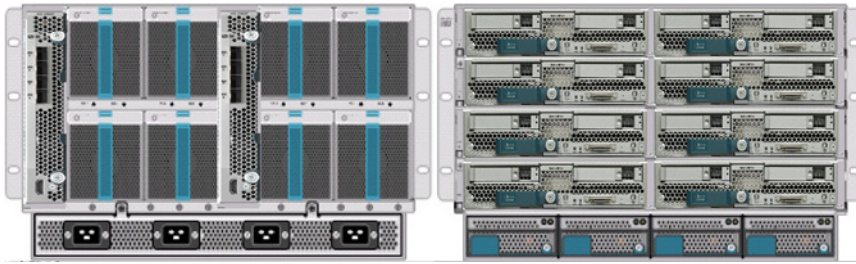
Cisco UCS Blade Servers

Delivering performance, versatility, and density without compromise, the Cisco UCS B200 M3 Blade Server addresses the broadest set of workloads, from IT and Web Infrastructure through distributed database to virtualization.

Building on the success of the Cisco UCS B200 M2 blade servers, the enterprise-class Cisco UCS B200 M3 server further extends the capabilities of Cisco's Unified Computing System portfolio in a half blade form factor. The Cisco UCS B200 M3 server harnesses the power and efficiency of the Intel Xeon E5-2600 processor product family, up to 768 GB of RAM, 2 drives or SSDs and up to 2 x 20 GE to deliver exceptional levels of performance, memory expandability, and I/O throughput for nearly all applications. In addition, the Cisco UCS B200 M3 blade server offers a modern design that removes the need for redundant switching components in every chassis in favor of a simplified top of rack design, allowing more space for server resources, providing a density, power, and performance advantage over previous generation servers. The Cisco UCS B200 M3 Server is shown in Figure 4.

Figure 4   Cisco UCS B200 M3 Blade Server

Cisco Nexus 5548UP Switch

The Cisco Nexus 5548UP is a 1RU 1 Gigabit and 10 Gigabit Ethernet switch offering up to 960 gigabits per second throughput and scaling up to 48 ports. It offers 32 1/10 Gigabit Ethernet fixed enhanced Small Form-Factor Pluggable (SFP+) Ethernet/FCoE or 1/2/4/8-Gbps native FC unified ports and three expansion slots. These slots have a combination of Ethernet/FCoE and native FC ports. The Cisco Nexus 5548UP switch is shown in Figure 5.

Figure 5   Cisco Nexus 5548UP Switch

Cisco I/O Adapters

Cisco UCS Blade Servers support various Converged Network Adapter (CNA) options. Cisco UCS Virtual Interface Card (VIC) 1240 is used in this EMC VSPEX solution.

The Cisco UCS Virtual Interface Card 1240 is a 4-port 10 Gigabit Ethernet, Fibre Channel over Ethernet (FCoE)-capable modular LAN on motherboard (mLOM) designed exclusively for the M3 generation of Cisco UCS B-Series Blade Servers. When used in combination with an optional Port Expander, the Cisco UCS VIC 1240 capabilities can be expanded to eight ports of 10 Gigabit Ethernet.

The Cisco UCS VIC 1240 enables a policy-based, stateless, agile server infrastructure that can present up to 256 PCIe standards-compliant interfaces to the host that can be dynamically configured as either network interface cards (NICs) or host bus adapters (HBAs). In addition, the Cisco UCS VIC 1240 supports Cisco Data Center Virtual Machine Fabric Extender (VM-FEX) technology, which extends the Cisco UCS fabric interconnect ports to virtual machines, simplifying server virtualization deployment.

Figure 6   Cisco UCS VIC 1240
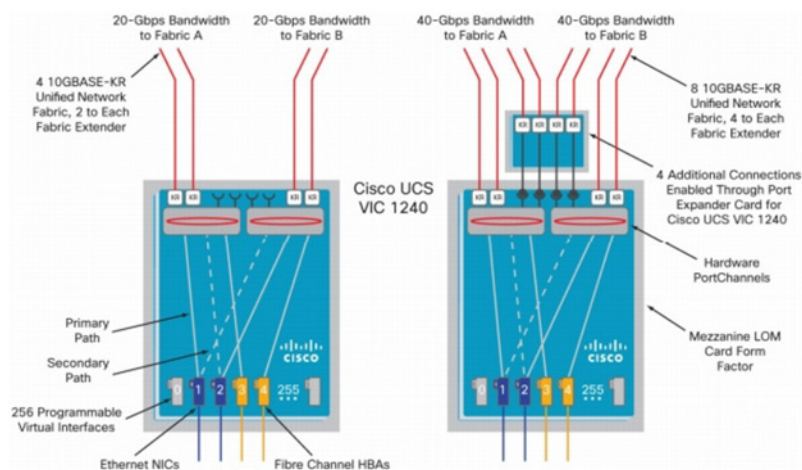
Cisco UCS Differentiators

Cisco's Unified Computing System is revolutionizing the way servers are managed in data centers.

Following are the unique differentiators of Cisco UCS and Cisco UCS Manager.

- Embedded management—In Cisco Unified Computing System, the servers are managed by the embedded firmware in the Fabric Interconnects, eliminating need for any external physical or virtual devices to manage the servers. Also, a pair of FIs can manage up to 40 chassis, each containing up 8 blade servers, to a total of 160 servers with fully redundant connectivity. This gives enormous scaling on the management plane.

- Unified fabric—In Cisco Unified Computing System, from blade server chassis or rack server fabric-extender to FI, there is a single Ethernet cable used for LAN, SAN, and management traffic. This converged I/O results in reduced cables, SFPs, and adapters - reducing capital and operational expenses of overall solution.

- Auto Discovery—By simply inserting the blade server in the chassis, discovery and inventory of compute resource occurs automatically without any management intervention. The combination of unified fabric and auto-discovery enables the wire-once architecture of Cisco Unified Computing System, where compute capability of Cisco Unified Computing System can be extended easily while keeping the existing external connectivity to LAN, SAN, and management networks.

- Policy based resource classification—When a compute resource is discovered by Cisco UCS Manager, it can be automatically classified to a given resource pool based on policies defined. This capability is useful in multi-tenant cloud computing.

- Combined Rack and Blade server management—Cisco UCS Manager can manage Cisco UCS B-series blade servers and Cisco UCS C-series rack server under the same Cisco UCS domain. This feature, along with stateless computing makes compute resources truly hardware form factor agnostic.

- Model based management architecture—Cisco UCS Manager architecture and management database is model based and data driven. An open, standard based XML API is provided to operate on the management model. This enables easy and scalable integration of Cisco UCS Manager with other management system, such as VMware vCloud director, Microsoft System Center, and Citrix Cloud Platform.

- Policies, Pools, Templates—The management approach in Cisco UCS Manager is based on defining policies, pools and templates, instead of cluttered configuration, which enables a simple, loosely coupled, data driven approach in managing compute, network and storage resources.

- Loose referential integrity—In Cisco UCS Manager, a service profile, port profile, or policies can refer to other policies or logical resources with loose referential integrity. A referred policy does not have to exist at the time of authoring the referring policy or a referred policy can be deleted even though other policies are referring to it. This provides different subject matter experts to work independently from each other. This provides great flexibility where different experts from different domains, such as network, storage, security, server, and virtualization work together to accomplish a complex task.

- Policy resolution—In Cisco UCS Manager, a tree structure of organizational unit hierarchy can be created that mimics the real life tenants and/or organization relationships. Various policies, pools, and templates can be defined at different levels of organization hierarchy. A policy referring to another policy by name is resolved in the organization hierarchy with closest policy match. If no policy with specific name is found in the hierarchy of the root organization, then special policy named "default" is searched. This policy resolution practice enables automation friendly management APIs and provides great flexibility to owners of different organizations.

- Service profiles and stateless computing—A service profile is a logical representation of a server, carrying its various identities and policies. This logical server can be assigned to any physical compute resource as far as it meets the resource requirements. Stateless computing enables procurement of a server within minutes, which used to take days in legacy server management systems.

- Built-in multi-tenancy support—The combination of policies, pools, templates, loose referential integrity, policy resolution in organization hierarchy, and a service profiles based approach to compute resources makes Cisco UCS Manager inherently friendly to multi-tenant environment typically observed in private and public clouds.

- Extended Memory—The extended memory architecture of Cisco Unified Computing System servers allows up to 760 GB RAM per server - allowing huge VM to physical server ratio required in many deployments, or allowing large memory operations required by certain architectures like Big-Data.

- Virtualization aware network—VM-FEX technology makes access layer of network aware about host virtualization. This prevents domain pollution of compute and network domains with virtualization when virtual network is managed by port-profiles defined by the network administrators' team. VM-FEX also off loads hypervisor CPU by performing switching in the hardware, thus allowing hypervisor CPU to do more virtualization related tasks. VM-FEX technology is well integrated with VMware vCenter, Linux KVM, and Hyper-V SR-IOV to simplify cloud management.

- Simplified QoS—When the Fibre Channel and Ethernet are converged in Cisco Unified Computing System fabric, built-in support for QoS and lossless Ethernet makes it seamless. Network Quality of Service (QoS) is simplified in Cisco UCS Manager by representing all system classes in one GUI panel.

Microsoft Hyper-V 2012 R2

Microsoft Hyper-V 2012 R2 is a next-generation virtualization solution from Microsoft that builds upon previous releases and provides greater levels of scalability, security, and availability to virtualized environments. Hyper-V 2012 R2 offers improvements in performance and utilization of CPU, memory, and I/O. It also offers users the option to assign up to 64 virtual CPU to a virtual machine—giving system administrators more flexibility in their virtual server farms as processor-intensive workloads continue to increase. Table 2 illustrates the increase in scale from the previous major release.

Table 2   Hyper-V Scale

System	Resource	Maximum Number	
		Windows Server 2008 R2	Windows Server 2012 R2
Host	Logical processors on hardware	64	320
	Physical memory	1 TB	4 TB
	Virtual processors per host	512	1,024
Virtual Machine	Virtual processor per VM	4	64
	Memory per VM	64 GB	1 TB
	Active virtual machines	384	1,024
	Virtual disk size	2 TB	64 TB
Cluster	Nodes	16	64
	Virtual machines	1,000	8,000

Microsoft provides System Center 2012 R2 to provide additional management capabilities to a virtualized and physical environment. This Design document assumes the existence of a System Center Virtual Machine Manager server, but Hyper-V provides significant management capabilities without the addition of System Center. Included in the base capabilities of Hyper-V are:

- High availability—up to 64 Hyper-V hosts can be formed into a single cluster hosting up to 8,000 virtual machines.

- Disaster recovery—virtual machine replicas can be made to other locations for rapid recovery and restart in case of a disaster.

- Live migration—virtual machines can be live migrated (moved from one host to another with no service downtime) between any two Hyper-V hosts, whether they are clustered or not, without the need for any shared storage.

- Live storage migration—as with live machine migration, storage for a virtual machine can be migrated without any service downtime.

- Dynamic memory—virtual machines defined with dynamic memory can release unused memory for use by other virtual machines that require it.

- Cluster Shared Volumes—storage volumes in a failover cluster environment allow any virtual machine executing on any cluster host full read/write access to its virtual hard drives from any node in the cluster. This also provides additional high availability by ensuring access to the volume and uninterrupted virtual machine execution even if the Hyper-V host loses physical connection to the volume.

- Clustering of virtual machines—virtual machine clusters can use storage from many locations: iSCSI Targets, Virtual HBA to directly access Fibre Channel storage, SMB 3.0 file shares, or virtual hard drives residing on Cluster Shared Volumes.

- PowerShell—a complete PowerShell module enables management of virtual machines and their resources via scripting so repetitive tasks can be easily executed.

- NIC teaming—teaming at the host level enables up to 32 physical NICs to form a single team. Within a virtual machine, two virtual NICs can form a team.

- Data deduplication—automatically deduplicate common data on the disk, including operating system files on virtual hard drives.

EMC Storage Technology and Benefits

The VNX storage series provides both file and block access with a broad feature set, which makes it an ideal choice for any private cloud implementation.

VNX storage includes the following components, sized for the stated reference architecture workload:

- Host adapter ports (For block)—Provide host connectivity through fabric to the array
- Storage processors—The compute components of the storage array, which are used for all aspects of data moving into, out of, and between arrays
- Disk drives—Disk spindles and solid state drives (SSDs) that contain the host or application data and their enclosures
- Data Movers (For file)—Front-end appliances that provide file services to hosts (optional if CIFS services are provided)

> **Note** The term Data Mover refers to a VNX hardware component, which has a CPU, memory, and I/O ports. It enables Common Internet File System (CIFS-SMB) and Network File System (NFS) protocols on the VNX.

The Microsoft Hyper-V private cloud solutions for 300, 600, and 1,000 virtual machines described in this document are based on the EMC VNX5400, EMC VNX5600, and the EMC VNX5800 storage arrays, respectively. The VNX5400 array can support a maximum of 250 drives, the VNX5600 can host up to 500 drives, and the VNX5800 can host up to 750 drives.

The VNX series supports a wide range of business-class features that are ideal for the private cloud environment, including:

- EMC Fully Automated Storage Tiering for Virtual Pools (FAST VP™)
- EMC FAST Cache
- File-level data deduplication and compression
- Block deduplication
- Thin provisioning
- Replication
- Snapshots or checkpoints
- File-level retention
- Quota management

Features and Enhancements

The EMC VNX flash-optimized unified storage platform delivers innovation and enterprise capabilities for file, block, and object storage in a single, scalable, and easy-to-use solution. Ideal for mixed workloads in physical or virtual environments, VNX combines powerful and flexible hardware with advanced efficiency, management, and protection software to meet the demanding needs of today's virtualized application environments.

VNX includes many features and enhancements designed and built upon the first generation's success. These features and enhancements include:

- More capacity with multicore optimization with Multicore Cache, Multicore RAID, and Multicore FAST Cache (MCx)
- Greater efficiency with a flash-optimized hybrid array
- Better protection by increasing application availability with active/active storage processors
- Easier administration and deployment by increasing productivity with a new Unisphere Management Suite

VSPEX is built with the next generation of VNX to deliver even greater efficiency, performance, and scale than ever before.

Flash-Optimized Hybrid Array

VNX is a flash-optimized hybrid array that provides automated tiering to deliver the best performance to your critical data, while intelligently moving less frequently accessed data to lower-cost disks.

In this hybrid approach, a small percentage of flash drives in the overall system can provide a high percentage of the overall IOPS. A flash-optimized VNX takes full advantage of the low latency of flash to deliver cost-saving optimization and high performance scalability. The EMC Fully Automated Storage Tiering Suite (FAST Cache and FAST VP) tiers both block and file data across heterogeneous drives and boosts the most active data to the cache, ensuring that customers never have to make concessions for cost or performance.

FAST VP dynamically absorbs unpredicted spikes in system workloads. As that data ages and becomes less active over time, FAST VP tiers the data from high-performance to high-capacity drives automatically, based on customer-defined policies. This functionality has been enhanced with four times better granularity and with new FAST VP solid-state disks (SSDs) based on enterprise multi-level cell (eMLC) technology to lower the cost per gigabyte. All VSPEX use cases benefit from the increased efficiency.

VSPEX Proven Infrastructures deliver private cloud, end-user computing, and virtualized application solutions. With VNX, customers can realize an even greater return on their investment. VNX provides out-of-band, block-based deduplication that can dramatically lower the costs of the flash tier.

VNX Intel MCx Code Path Optimization

The advent of flash technology has been a catalyst in totally changing the requirements of midrange storage systems. EMC redesigned the midrange storage platform to efficiently optimize multicore CPUs to provide the highest performing storage system at the lowest cost in the market.

MCx distributes all VNX data services across all cores—up to 32, as shown in Figure 7. The VNX series with MCx has dramatically improved the file performance for transactional applications like databases or virtual machines over network-attached storage (NAS).

Figure 7   *Next-Generation VNX with multicore optimization*
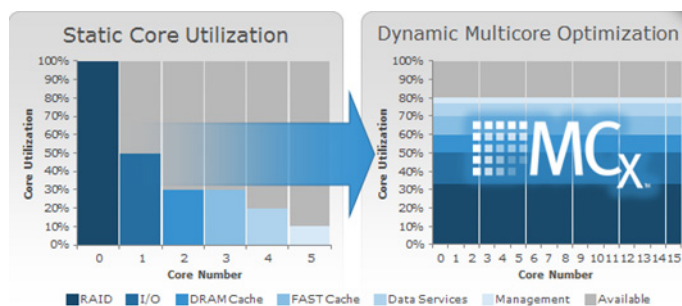
Multicore Cache

The cache is the most valuable asset in the storage subsystem; its efficient use is key to the overall efficiency of the platform in handling variable and changing workloads. The cache engine has been modularized to take advantage of all the cores available in the system.

Multicore RAID

Another important part of the MCx redesign is the handling of I/O to the permanent back-end storage-hard disk drives (HDDs) and SSDs. Greatly increased performance improvements in VNX come from the modularization of the back-end data management processing, which enables MCx to seamlessly scale across all processors.

VNX Performance

Performance enhancements

VNX storage, enabled with the MCx architecture, is optimized for FLASH 1st and provides unprecedented overall performance, optimizing for transaction performance (cost per IOPS), bandwidth performance (cost per GB/s) with low latency, and providing optimal capacity efficiency (cost per GB).

VNX provides the following performance improvements:

- Up to four times more file transactions when compared with dual controller arrays
- Increased file performance for transactional applications (for example, Microsoft Exchange on VMware over NFS) by up to three times with a 60 percent better response time
- Up to four times more Oracle and Microsoft SQL Server OLTP transactions
- Up to four times more virtual machines, a greater than three times improvement

Active/Active Array Storage Processors

The new VNX architecture provides active/active array storage processors, as shown in the following figure, which eliminate application timeouts during path failover since both paths are actively serving I/O.

Load balancing is also improved and applications can achieve an up to two times improvement in performance. Active/Active for block is ideal for applications that require the highest levels of availability and performance, but do not require tiering or efficiency services like compression, deduplication, or snapshot.

With this VNX release, VSPEX customers can use virtual Data Movers (VDMs) and VNX Replicator to perform automated and high-speed file system migrations between systems. This process migrates all snaps and settings automatically, and enables the clients to continue operation during the migration.

Figure 8   Active/active processors increase performance, resiliency, and efficiency
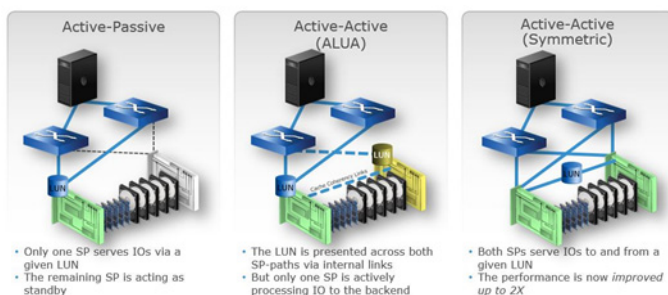
Unisphere Management Suite

The new Unisphere Management Suite extends Unisphere's easy-to-use, interface to include VNX Monitoring and Reporting for validating performance and anticipating capacity requirements. As shown in the following figure, the suite also includes Unisphere Remote for centrally managing up to thousands of VNX and VNXe systems with new support for XtremSW Cache.

Figure 9   Unisphere Management Suite

Virtualization Management

EMC Storage Integrator

EMC Storage Integrator (ESI) is targeted towards the Windows and Application administrator. ESI is easy to use, delivers end-to end monitoring, and is hypervisor agnostic. Administrators can provision in both virtual and physical environments for a Windows platform, and troubleshoot by viewing the topology of an application from the underlying hypervisor to the storage.

Microsoft Hyper-V

With Windows Server 2012, Microsoft provides Hyper-V 3.0, an enhanced hypervisor for private cloud that can run on NAS protocols for simplified connectivity.

Offloaded Data Transfer

The Offloaded Data Transfer (ODX) feature of Microsoft Hyper-V enables data transfers during copy operations to be offloaded to the storage array, freeing up host cycles. For example, using ODX for a live migration of a SQL Server virtual machine doubled performance, decreased migration time by 50 percent, reduced CPU on the Hyper-V sever by 20 percent, and eliminated network traffic.

EMC Avamar

EMC's Avamar® data deduplication technology seamlessly integrates into virtual environments, providing rapid backup and restoration capabilities. Avamar's deduplication results in vastly less data traversing the network, and greatly reduces the amount of data being backed up and stored; resulting in storage, bandwidth and operational savings.

The following are the two most common recovery requests used in backup and recovery:

- File-level recovery—Object-level recoveries account for the vast majority of user support requests. Common actions requiring file-level recovery are-individual users deleting files, applications requiring recoveries, and batch process-related erasures.

- System recovery—Although complete system recovery requests are less frequent in number than those for file-level recovery, this bare metal restore capability is vital to the enterprise. Some of the common root causes for full system recovery requests are viral infestation, registry corruption, or unidentifiable unrecoverable issues.

The Avamar System State protection functionality adds backup and recovery capabilities in both of these scenarios.

Architectural Overview

This Cisco Design discusses the deployment model of the Microsoft Hyper-V 2012 R2 solution for up to 300 virtual machines. Understanding the virtual machine workloads vary from customer to customer, Cisco server components can be easily added in single server increments to address the exact workload of the customer.

Table 3 lists the hardware requirements of the designed solution.

Table 3   Hardware Requirements

Component	Hardware Required
Servers	Six Cisco B200 M3 Servers with 192 GB of memory
Adapters	Six Cisco VIC 1240 adapters; one per server
Chassis	One Cisco UCS 5108 Blade Server Chassis
Fabric extenders	Two 2204XP fabric extenders; two per chassis
Fabric interconnects	Two Cisco UCS 6248UP Fabric Interconnects
Network switches	Two Cisco Nexus 5548UP Switches
Storage	One EMC VNX5400 Storage array

Table 4 lists the various firmware and software components for this VSPEX design.

Table 4   Firmware and Software Components

Vendor	Name	Version	Description
Cisco	Cisco UCS B200 M3 servers	2.1(2a) – CIMC B200M3.2.1.1a.0.12172 0121447	Cisco UCS B200 M3 blade server firmware
Cisco	Cisco VIC 1240	2.1(2a)	Cisco Virtual Interface card firmware
Cisco	Cisco UCS 2204XP Fabric Extender	2.1(2a)	Cisco UCS fabric extender firmware
Cisco	Cisco UCS 6248UP Fabric Interconnect	2.1(2a)	Cisco UCS fabric interconnect firmware
Cisco	Cisco UCS Manager	2.1(2a)	Cisco UCS Manager software
Cisco	Cisco Nexus 5548UP Switch	6.0(2)N1(2a)	Cisco NX-OS
EMC	EMC VNX5400	VNX Block OE 05.33	EMC Storage array firmware
EMC	EMC Avamar	6.1 SP1	EMC data backup software
EMC	Data Domain OS	5.2	EMC data domain operating system
Microsoft	Microsoft Windows Server	2012 R2	Operating system
Microsoft	System Center Virtual Machine Manager	2012 R2	Virtual Machine management software (customer provided)

Table 5 outlines the Cisco UCS B200 M3 server configuration details.

Table 5   Server Configuration Details

Component	Capacity
Memory (RAM)	192 GB (12 x 16 GB)
Processor	2 x Intel E2650 CPUs, 2.6 GHz, 8 cores, 16 threads
Adapters	Cisco VIC 1240

This architecture assumes there is an existing infrastructure / management network available in which a virtual machine hosting Microsoft System Center Virtual Machine Manager 2012 R2 server and Windows Active Directory/DNS/DHCP server are present. Figure 10 illustrates a high-level Cisco solution for EMC VSPEX Microsoft Hyper-V architecture for up to 300 virtual machines.

Figure 10   Reference Architecture for up to 300 Virtual Machines
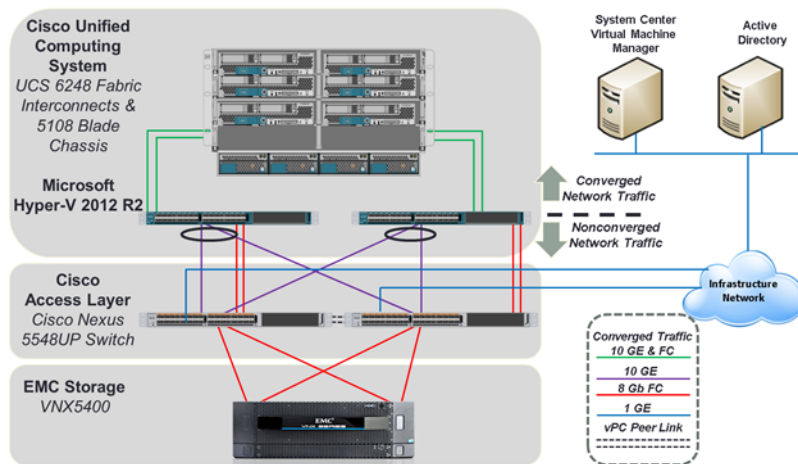
The following are the high-level design points of the architecture:

- Only Ethernet is used as network layer 2 media to access Cisco UCS 6248UP from the Cisco UCS B200 M3 blade servers.
- Infrastructure network is on a separate 1GE network.
- Network redundancy is built in by providing two switches, two storage controllers and redundant connectivity for data, storage, and infrastructure networking.

This design does not recommend or require any specific layout of infrastructure network. The Virtual Machine Manager server and AD/DNS/DHCP virtual machines are hosted on the infrastructure network. However, design does require accessibility of certain VLANs from the infrastructure network to reach the servers.

Hyper-V 2012 R2 is used as the hypervisor on each server and is installed on fibre channel SAN. The defined load is 60 virtual machines per Cisco UCS B200 M3 server blade.

Memory Configuration Guidelines

This section provides guidelines for allocating memory to the virtual machines. The guidelines outlined here take into account Hyper-V memory overhead and the virtual machine memory settings.

Hyper-V Memory Management Concepts

Microsoft Hyper-V has a number of advanced features to maximize performance, and overall resource utilization. The most important features relate to memory management. This section describes some of these features, and the items to consider when using these features in the VSPEX environment.

Dynamic Memory

Dynamic Memory was introduced in Windows Server 2008 R2 SP1 to increase physical memory efficiency by treating memory as a shared resource, and dynamically allocating it to virtual machines. The amount of memory used by each virtual machine is adjustable at any time. Dynamic Memory reclaims unused memory from idle virtual machines, which allows more virtual machines to run at any given time. In Windows Server 2012, Dynamic Memory enables administrators to dynamically increase the maximum memory available to virtual machines.

Dynamic memory pools all the physical memory available on a physical host and dynamically distributes it to virtual machines running on that host as the virtual machines need it. As workloads change, virtual machines will be able to dynamically ask for memory if it is needed or dynamically release memory if it is no longer needed without service interruptions.

Dynamic Memory requires that a virtual machine have a minimum and maximum size of virtual memory assigned. The virtual machine will never have less memory assigned to it than what is specified by the minimum and never ask for more than what is specified as the maximum. When a virtual machine is initialized, it is given the minimum amount specified. As processes are loaded, they create a demand for a specific amount of memory. If the virtual machine does not yet have enough physical memory assigned to it based on its demand, the hypervisor and SLAT will allocate more physical memory to ensure optimal performance. If the virtual machine is no longer using memory, it uses a ballooning technique to free up unused physical memory to return to Hyper-V to be allocated to other virtual machines that need the memory.

One of the things that often happens is that when a machine, physical or virtual, is first starting, it may require more physical memory than it would require while it is running its normal tasks. This happens due to the many startup processes that are run to get the system running but then stop once they have performed their function. Therefore, Hyper-V also has a startup RAM setting that can be set larger than the minimum, thereby ensuring more memory at startup for a faster startup time. After the virtual machine has started and has it normal processes running, the ballooning technology will free any excess memory for use by other virtual machines.

In addition to the startup, minimum, and maximum settings offered by Hyper-V, it also allows two other settings to optimize memory usage. One setting is to specify the percentage of memory that Hyper-V should try to reserve as a buffer for the virtual machine. Then when the virtual machine has a demand for more physical memory, it can draw from this buffer instead going through the more intensive route of asking for new physical memory. This ensures a quicker response to memory demands within the virtual machine. The second setting is a memory weight that specifies how to prioritize memory demands for one virtual machine in relationship to the demands of other virtual machines. This allows for high-priority virtual machines to have their memory demands satisfied before lower priority virtual machines.

Smart Paging

Even with Dynamic Memory, Hyper-V allows more virtual machines than the available physical memory can support. In most cases, there is a memory gap between minimum memory and startup memory. Smart Paging is a memory management technique that uses disk resources as temporary memory replacement. It swaps out less-used memory to disk storage, and swaps in when needed. Performance degradation is a potential drawback of Smart Paging. Hyper-V continues to use the guest paging when the host memory is oversubscribed because it is more efficient than Smart Paging.

Non-Uniform Memory Access

Non-Uniform Memory Access (NUMA) is a multi-node computer technology that enables a CPU to access remote-node memory. This type of memory access degrades performance, so Windows Server 2012 employs a process known as processor affinity, which pins threads to a single CPU to avoid remote-node memory access. In previous versions of Windows, this feature is only available to the host. Windows Server 2012 extends this functionality to the virtual machines, which provides improved performance in symmetrical multiprocessor (SMP) environments.

Allocating Memory to Virtual Machines

Memory sizing for a virtual machine in VSPEX architectures is based on many factors. With the number of application services and use cases available determining a suitable configuration for an environment requires creating a baseline configuration, testing, and making adjustments, as discussed later in this paper. Table 6 outlines the resources used by a single virtual machine.

Table 6   Resources for a Single Virtual Machine

Characteristics	Value
Virtual processor per VM (vCPU)	1
RAM per VM	2 GB
Available storage capacity per VM	100 GB
I/O operations per second (IOPS) per VM	25
I/O pattern	Random
I/O read/write ration	2:1

The following are some recommended best practices for memory allocation:

- Account for memory overhead - Virtual machines require memory beyond the amount allocated, and this memory overhead is per virtual machine. Memory overhead includes space reserved for integration services and other virtualization-related processes. The amount of overhead is somewhat trivial, but it still needs to be factored in. VMs with 1 GB or less of RAM only use about 32 MB of memory for virtualization-related overhead. You should add 8 MB for every gigabyte of additional RAM. For example, a VM with 2 GB of RAM would use 40 MB (32 MB plus 8 MB) of memory for virtualization-related overhead. Likewise, a VM with 4 GB of memory would have 56 MB of memory overhead. Each running virtual machine

also has an associated virtual machine worker process to coordinate management tasks for the virtual machine. This process uses a little less than 7 MB of memory. This memory and process overhead is in addition to the memory allocated to the virtual machine and must be available on the Hyper-V host.

- "Right-size" memory allocations - Over-allocating memory to virtual machines can waste memory unnecessarily, but it can also increase the amount of memory overhead required to run the virtual machine, thus reducing the overall memory available for other virtual machines. Fine-tuning the memory for a virtual machine is done easily and quickly by adjusting the virtual machine properties. Using Dynamic Memory helps ease the administration of this right-sizing.

- Do not overcommit - as the term 'overcommit' implies, this is trying to use more than is available. Just as you cannot pump 10 gigabits of data through a 1 Gpbs network in one second, you cannot use more memory than what is physically available. Again, Dynamic Memory helps ease the administration of memory on a physical system that is running close to its memory capacity.

- Monitor usage - Hyper-V provides performance monitoring statistics for resources used by virtual machines. These statistics can be monitored from the host environment without the need to go into every virtual machine individually. Perfmon, a performance monitoring utility that is part of the operating system, provides these statistics in counters prefixed with the string 'Hyper-V'. For more information about monitoring performance on Hyper-V see
http://technet.microsoft.com/en-US/library/cc768535(v=BTS.10).aspx.

In addition to accounting for the memory used by the virtual machines, you should also allow the host to have at least 2 GB for the parent partition. This partition includes processes for monitoring and managing the environment and features such as the built-in failover clustering capability. Additionally control information for the running of the virtual machines is also stored in this memory.

Storage Guidelines

VSPEX architecture for Microsoft Hyper-V up to 300 VMs uses Fibre Channel to access storage arrays. The Hyper-V hosts boot from the SAN storage, ensuring stateless configuration for the individual Hyper-V hosts. If one of the Cisco UCS B200 M3 servers becomes unavailable for any reason, the service profile defining that server's configuration can be associated to another Cisco UCS B200 M3 server and boot from the same operating system image on the SAN without any reconfiguration to bring it into service. The shared storage for storing VMs and their data uses the same storage array and protocol, minimizing the management overhead associated with managing VM storage. VMs can access Fibre Channel LUNs directly (optional) using virtual HBAs in the VMs. Highly available failover clusters built with VMs can use either the virtual HBAs or simply shared virtual hard disks (VHDX) stored on the array.

Virtual Server Configuration

Storage for Hyper-V can be categorized into three layers of storage technology:

- The storage array is the bottom layer consisting for physical disk spindles. These spindles are aggregated into RAID sets and then LUNs are defined on the RAID sets to present to the Hyper-V hosts.
- Storage array LUNs presented to the Hyper-V hosts are used for two purposes.
  - Boot volumes - operating system image used for booting the Hyper-V host
  - Cluster Shared Volumes - shared storage that is read/write accessible by all Hyper-V hosts configured in a Microsoft Failover Cluster
- Virtual disk files (VHDX) are created on the Cluster Shared Volumes and are used for multiple purposes.
  - Boot volumes - guest operating system image used for booting a virtual machine
  - Data volumes - data volumes required by applications

- Shared storage - shared volumes used by virtual machine guest clusters

Figure 11 illustrates the above explanation.

Figure 11   Hyper-V Storage Virtualization Stack
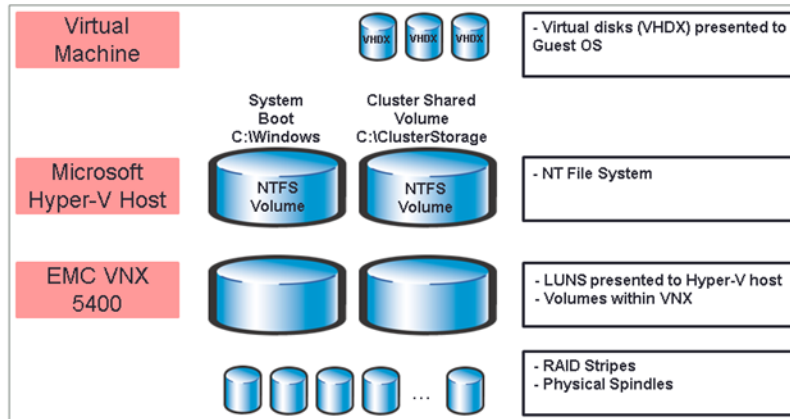
Storage Protocol Capabilities

The EMC VNX5400 provides Hyper-V and storage administrators with the flexibility to use the storage protocol that meets the requirements or standards of the business. This can be a single protocol datacenter-wide or multiple protocols for tiered scenarios. The EMC VNX5400 can support Fibre Channel, FCoE, iSCSI, and SMB protocols.

The Cisco solution for EMC VSPEX with Microsoft Hyper-V recommends a single protocol, Fibre Channel, throughout in order to simplify the design.

Storage Best Practices

It is recommended that storage administrators become familiar with Microsoft's suggestions for performance tuning. They have published a document that can be found at:

http://msdn.microsoft.com/en-us/library/windows/hardware/jj248719.aspx.

- Multi-path—having a redundant set of paths to the EMC VNX is critical to protecting the availability of the environment as well as ensuring the best performance. Microsoft provides built-in multi-path IO support as part of its operating system. EMC builds on top of this built-in capability to provided added functions with its PowerPath software.

- Partition alignment—in the past, it was often recommended to manually align partitions on disk volumes to ensure optimal performance. Since Windows Server 2008, Microsoft has taken the guess-work out of partition alignment. Microsoft Windows Server 2012 and later natively support the 4K sector disks that are commonly available today and automatically aligns the partitions for optimal performance.

- Shared storage—Cluster Shared Volumes provide a high level of availability to the virtual machine environment. VHDX files (virtual hard disks) can be accessed from any node of the cluster (up to 64 nodes) with full read/write capability. If a node loses its physical connection to a CSV, it can still access the VHDX files via the network, ensuring uninterrupted service from that VM owning those VHDX files.

- Calculate total VM storage requirements—each virtual machine may require more space than the total of its VHDX files. One of the settings for a virtual machine is to take an automatic stop action; that is action to be performed if the host machine is gracefully shut down. One of the automatic stop actions often used is to save the virtual machine state. In order to ensure that enough space is available on disk when an automatic

stop action is initiated, Hyper-V will create file on disk with enough space to save the memory contents of the virtual machine. For example, assume that you have a virtual machine with 8 GB or RAM allocated and a 40 GB VHDX system volume and a 50 GB VHDX data volume. If the automatic stop action for a virtual machine is to save the virtual machine state (the default setting), you will need to ensure that you have at least 98 GB of disk space available for this VM.

- Understand I/O requirements—Under-provisioned storage can significantly slow responsiveness and performance for applications. In a multi-tier application, you can expect each tier of application to have different I/O requirements. As a general recommendation, pay close attention to the amount of virtual machine disk files hosted on a single CSV. Over-subscription of the I/O resources can go unnoticed at first and slowly begin to degrade performance if not monitored proactively

Storage Layout

Figure 12 shows the physical disk layout. Disk provisioning on the VNX5400 storage array is simplified through the use of wizards, so that administrators do not choose which disks belong to a given storage pool. The wizard may choose any available disk of the proper type, regardless of where the disk physically resides in the array.

Figure 12   Storage Architecture for up to 300 Virtual Machines
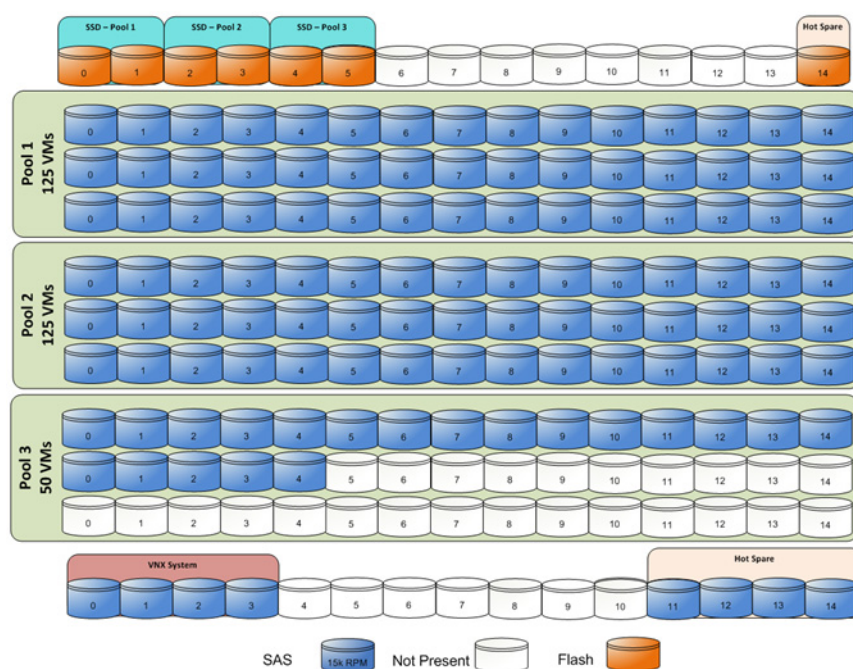
The reference architecture uses the following configuration:

- One hundred ten 600 GB SAS disks are allocated to three block based storage pools.
- Four 600 GB SAS disks are configured as hot spares.

> **Note**   System drives are specifically excluded from the pools and are not used for additional storage.

- Six 200 GB flash drives are configured in the array FAST VP.

- A single 200 GB flash drive is configured as a hot spare.
- Three pools are configured to map to the three SAS disk pools.

The VNX family storage array is designed for five 9s availability (99.999% uptime) by using redundant components throughout the array. All of the array components are capable of continued operation in case of hardware failure. The RAID disk configuration on the array provides protection against data loss due to individual disk failures, and the available hot spare drives can be dynamically allocated to replace a failing disk.

Storage Building Blocks

VSPEX uses a building block approach to reduce this complexity. A building block is a set of disk spindles that can support a certain number of virtual servers in the VSPEX architecture. Each building block combines several disk spindles to create a storage pool that supports the needs of the private cloud environment. Each building block storage pool, regardless of the size, contains two flash drives with FAST VP storage tiering to enhance metadata operations and performance.

VSPEX solutions have been engineered to provide a variety of sizing configurations which afford flexibility when designing the solution. Customers can start out by deploying smaller configurations and scale up as their needs grow. At the same time, customers can avoid over-purchasing by choosing a configuration that closely meets their needs. To accomplish this, VSPEX solutions can be deployed using one or both of the scale-points below to obtain the ideal configuration while guaranteeing a given performance level.

Building Block for 13 Virtual Servers

The first building block can contain up to 13 virtual servers. It has two flash drives and five SAS drives in a storage pool.

Figure 13   13 Virtual Server Building Block
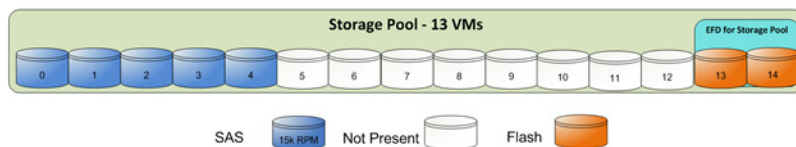
This is the smallest building block qualified for the VSPEX architecture. This building block can be expanded by adding five SAS drives and allowing the pool to restripe to add support for 13 more virtual servers.

Building Block for 125 Virtual Servers

The second building block can contain up to 125 virtual servers. It contains two flash drives, and 45 SAS drives. The preceding sections outline an approach to grow from 13 virtual machines in a pool to 125 virtual machines in a pool. However, after reaching 125 virtual machines in a pool, do not go to 138. Create a new pool and start the scaling sequence again.

Figure 14　　　　125 Virtual Server Building Block
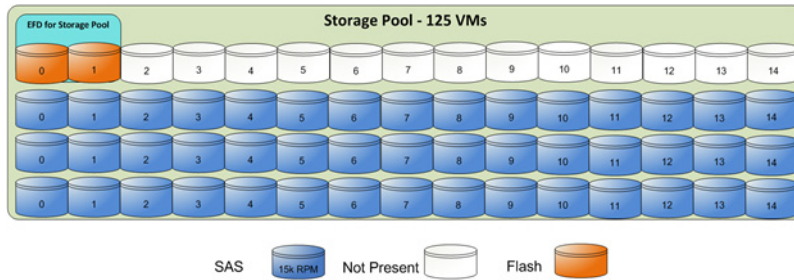
Each building block for 125 VMs would be presented to the host as two LUNs to be used as CSV for storing 62 or 63 virtual machines.

Implement this building block with all of the resources in the pool initially, or expand the pool over time as the environment grows. Table 7 lists the flash and SAS requirements in a pool for different numbers of virtual servers.

Table 7   Number of disks required for different numbers of virtual machines

Virtual Servers	FlasH Drives	SAS Drives
13	3	5
26	2	10
39	2	15
52	2	20
65	2	25
78	2	30
104	2	35
117	2	45
125	2	45*

Note Due to increased efficiency with larger stripes, the building block with 45 SAS drives can support up to 125 virtual servers.

To grow the environment beyond 125 virtual servers, create another storage pool using the building block method described here.

Using this building block approach to scalability, the following two diagrams picture the storage layout for 600 and 1000 virtual machines.

Figure 15   VNX5600 Storage for 600 VMs
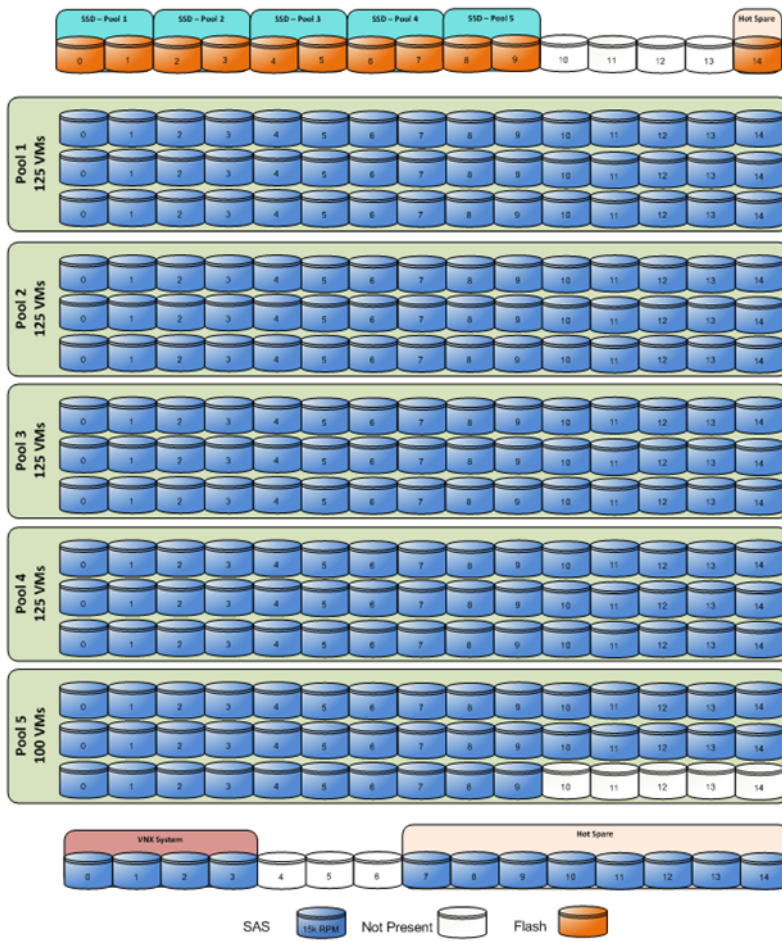
Figure 16   VNX5800 Storage for 1000 VMs
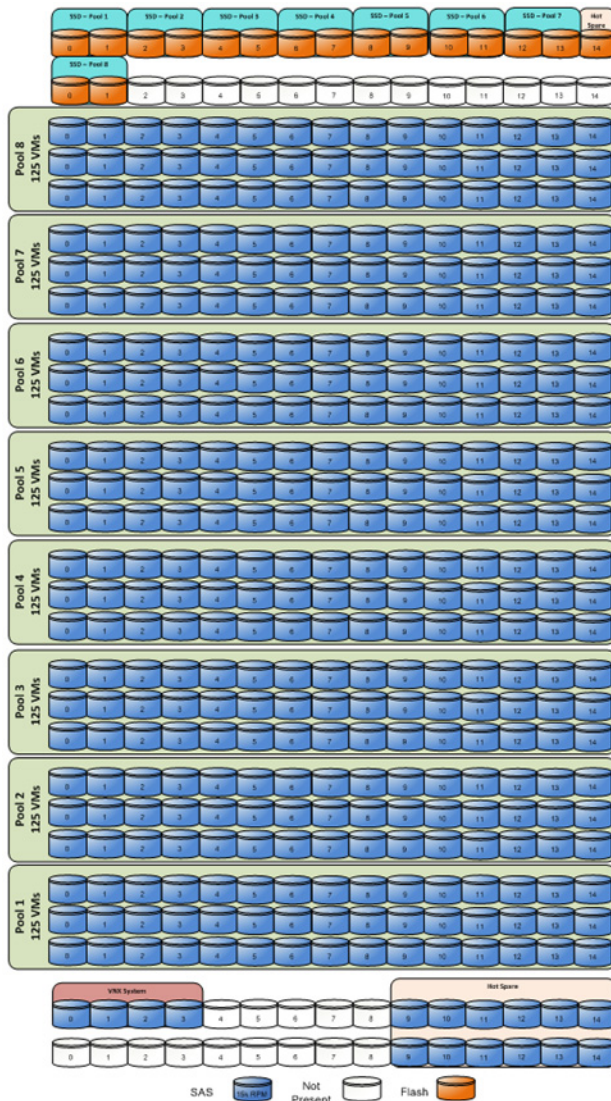
Networking Guidelines

The following are some recommended best practices for networking with Hyper-V:

- Fabric failover—always use the fabric failover feature of Cisco UCS VIC adapters for high-availability of network access.
- Separate virtual machine and host traffic—keep virtual machine and host traffic separate. This can be accomplished physically using separate virtual switches for VM networks that are defined on separate physical NICs or virtually using VLAN segmentation.

Recommended NICs

The recommended and suggested networking for a Microsoft Hyper-V solution have the following NICs:

- Host (or infrastructure) management—used by the Hyper-V host for management functions. If using the Nexus 1000V (optional), it is recommended that its Virtual Supervisor Module network run on this management network.
- Live Migration—used by Hyper-V host for live migrating virtual machines. This is not a required network, but it is highly recommended to separate this traffic
- Cluster Shared Volume—used by the Hyper-V host for managing the cluster shared volumes. This is not a required network, but it is highly recommended.
  - Normal operations—this network sees very little traffic. Each node in the cluster is directly accessing the CSV to read/write to the VHDX files in use by the virtual machines running on that node. The only traffic that passes over this network is what is known as 'metadata updates'. Metadata updates comprise file and directory creation/deletion/extension at the Hyper-V level. Most I/O is directly to the contents of the VHDX files in use by the VMs, and none of that is considered a metadata update. Metadata updates are handled by the node in the cluster that owns the LUN presented as a CSV.
  - Redirected mode—should a node of the cluster lose physical connectivity to a CSV, the CSV is set in redirected mode for that host. This means that read/write operations that would normally go directly to the CSV from the VM are redirected over the network designated as the CSV network to the node in the cluster that owns the LUN. This mode also may be initiated by backup programs.
- Virtual machine access—this network should be a separate network for accessing the resources of the virtual machines running on Hyper-V host. It is recommended that this network be defined as not available for use by the host. There should be a minimum of one virtual machine access network. Depending on your needs, you may require more.

Optional NICs

Depending on your business needs, you might be requirements to have additional networks as follows:

- Virtual Ethernet Module—when using the Nexus 1000V (optional), it is necessary to define a network for each subnet you want managed by the Nexus 1000V.
- iSCSI—if using iSCSI storage, it is recommended to deploy two networks on different subnets
- SMB—if using SMB storage, it is recommended to deploy two networks on different subnets

Quality of Service

It is recommended to define a quality of service for the Live Migration network to ensure optimal performance. If iSCSI and/or SMB networks are defined, it is recommended to define a quality of service for their use. In general, the QoS defined for storage will be different from the QoS defined for live migration.

Solution for up to 300 Hyper-V Virtual Machines

The key aspects of the Cisco UCS with EMC VNX5400 for up to 300 Hyper-V virtual machines solution are as follows:

- The solution is built with redundancy at every level—compute, networking, and storage.
- Six Cisco UCS B200 M3 servers are configured into a failover cluster. An average of 60 VMs per server allows for a single server to be available as a spare for failure or maintenance.
- Each Cisco UCS B200 M3 server is configured with 192 GB of RAM. This is slightly more RAM than is needed for the 60 reference VMs, ensuring that slight changes in the customer's configuration can be accommodated.

- Windows Server Hyper-V 2012 R2 is booted from SAN disk. FCoE is used from the servers to the fabric interconnects. Native FC is used between the Nexus 5248UP switches and the VNX5400.
- SAN boot and Cisco UCS Manager service profiles provide a stateless computing architecture. A Cisco UCS B200 M3 server can be replaced with very little, if any, downtime.
- The entire solution is built using a building block approach so the configuration can easily grow as the needs increase beyond the initial 300 virtual machines.

Stateless Computing

Cisco UCS Manager (UCSM) provides the concept of a Service Profile for a server running on a physical hardware. A service profile is a logical entity associated to a physical server. Among other things, service profile includes various identities of the server or server components, such as:

- BIOS UUID
- MAC address of virtual NIC of the server
- Node WWN (WWNN) for Fibre Channel SAN access
- Port WWN (WWPN) of the virtual HBA of the server
- IQN ID, if iSCSI protocol is used for storage access
- Management IP address for the KVM access

All these identities can be assigned to any physical server managed by the Cisco UCS Manager. All other configuration of the service profile is based on templates, pools, and policies, providing immense flexibility to the administrator. This includes firmware and BIOS versions required by the server. These concepts enable Cisco UCS Manager to provide stateless computing across the entire Cisco UCS Manager managed compute hardware. If remote storage is used to boot operating system of the server (such as SAN boot, PXE boot, iSCSI boot, etc.), a given service profile can be associated to any physical server hardware and downtime for migrating such a server can be reduced to few minutes. The solution presented in this CVD makes use of identity pools and SAN storage to simplify the server procurement and provide stateless computing capability.

Sizing Guidelines

In any discussion about virtual infrastructures, it is important to first define a reference workload. Not all servers perform the same tasks, and it is impractical to build a reference that takes into account every possible combination of workload characteristics.

Defining the Reference Workload

To simplify the discussion, a representative customer reference workload is defined as a virtual machine with specific characteristics. By comparing the actual customer usage to this reference workload, one can extrapolate which reference architecture to choose.

For the VSPEX solutions, the reference workload was defined as a single virtual machine. This virtual machine has the following characteristics (Table 8).

Table 8   Reference Virtual Machine Workload

Characteristic	Value
Virtual machines operating system	Windows Server 2012 R2
Virtual processors per VM (vCPU)	1
RAM per VM	2 GB

Available storage capacity per VM	100 GB
I/O operations per second (IOPS) per VM	25
I/O pattern	Random
I/O read/write ratio	2:1
Logical CPU to virtual CPU ratio	Up to 4:1

This specification for a virtual machine is not intended to represent any specific application. Rather, it represents a single common point of reference to measure other virtual machines.

Applying the Reference Workload

When considering an existing server which will move into a virtual infrastructure, you have the opportunity to gain efficiency by right-sizing the virtual hardware resources assigned to that system.

The reference architecture creates a pool of resources sufficient to host a target number of reference virtual machines as described above. It is entirely probable that customer virtual machines will not exactly match the specifications above. In that case, you can say that a single specific customer virtual machine is the equivalent of some number of reference virtual machines, and assume that number of virtual machines have been used in the pool. You can continue to provision virtual machines from the pool of resources until it is exhausted. Consider these examples:

Example 1 - Customer Built Application

A small custom-built application server will move into this virtual infrastructure. The physical hardware supporting the application is not being fully utilized at present. A careful analysis of the existing application reveals the application uses only one processor and needs 3 GB of memory to run efficiently. The IO workload ranges between 4 IOPS at idle time to 15 IOPS when busy. The entire application is only using about 30 GB on local hard drive storage.

The following resources are needed from the resource pool to virtualize this application:

- CPU resources for 1 VM

- Memory resources for 2 VMs

- Storage capacity for 1 VM

- IOPS for 1 VM

In this example a single virtual machine uses the resources of two reference virtual machines. When this VM is deployed, the solution's new capability would be 298 VMs.

Example 2 - Point of Sale System

The database server for a customer's point-of-sale system will move into this virtual infrastructure. It is currently running on a physical system with four CPUs and 16 GB of memory. It uses 200 GB storage and generates 200 IOPS during an average busy cycle.

The following resources are needed from the resource pool to virtualize this application:

- CPUs of 4 reference VMs

- Memory of 8 reference VMs

- Storage of 2 reference VMs

- IOPS of 8 reference VMs

In this example, the one virtual machine uses the resources of eight reference virtual machines. Once this VM is deployed, the solution's new capability would be 292 VMs.

Example 3 - Web Server

The customer's web server will move into this virtual infrastructure. It is currently running on a physical system with two CPUs and 8GB of memory. It uses 25 GB of storage and generates 50 IOPS during an average busy cycle.

The following resources are needed from the resource pool to virtualize this application:

- CPUs of 2 reference VMs
- Memory of 4 reference VMs
- Storage of 1 reference VMs
- IOPS of 2 reference VMs

In this example the virtual machine would use the resources of four reference virtual machines. Once this VM is deployed, the solution's new capability would be 296 VMs.

Example 4 - Decision Support Database

The database server for a customer's decision support system will move into this virtual infrastructure. It is currently running on a physical system with 10 CPUs and 48 GB of memory. It uses 5 TB of storage and generates 700 IOPS during an average busy cycle.

The following resources are needed from the resource pool to virtualize this application:

- CPUs of ten reference VMs
- Memory of 24 reference VMs
- Storage of 50 reference VMs
- IOPS of 28 reference VMs

In this example the one virtual machine uses the resources of fifty reference virtual machines. When this VM is deployed, the solution's new capability would be 250 VMs.

Summary of Examples

The four examples show the flexibility of the resource pool model. In all four cases the workloads simply reduce the number of available resources in the pool. If all four examples were implemented on the same virtual infrastructure, with an initial capacity of up to 300 virtual machines, they would leave a capacity for 236 reference virtual machines in the resource pool.

In more advanced cases, there may be trade-offs between memory and I/O or other relationships where increasing the amount of one resource, decreases the need for another. In these cases, the interactions between resource allocations become highly complex and are out of the scope of this document. However, when a change in the resource balance is observed, and the new level of requirements is known; these virtual machines can be added to the infrastructure using the method described in the above examples.

If the customer does not have a thorough understanding of the resource needs of their particular environment, Microsoft has a free tool, *Microsoft Assessment and Planning Toolkit*, that can be run against the customer environment to capture the actual characteristics. This tool can be found at www.microsoft.com/map.

Configuration Guidelines

This section provides the procedure to deploy the Cisco solution for EMC VSPEX Hyper-V architecture.

Follow these steps to configure the Cisco solution for EMC VSPEX VMware architectures:

- Pre-deployment tasks
- Physical setup

- Cable connectivity

- Configure Cisco Nexus switches

- Configure Cisco Unified Computing System using Cisco UCS Manager

- Prepare and configure storage array

- Install Initial Microsoft Windows Server 2012 R2

- Install Additional Microsoft Windows Server 2012 R2 and Failover Cluster

- Test the installation

These steps are described in detail in the following sections.

Included in this document are sample PowerShell scripts that can be used to more quickly build out this VSPEX environment. Though the scripts have been tested, no warranty is implied or granted that they do not contain errors. The use of these scripts assumes a 'green field' environment in which nothing else has been previously installed. In any case, each script should be examined before execution in the customer environment. Several of the scripts may have IP addresses hard coded in them that need to be changed to reflect the customer environment.

It is assumed that a person familiar with Microsoft's PowerShell scripting language is available to review these scripts for the customer before execution in the customer environment. In particular, the UcsConfig.xml file contains many variables that should be reviewed with the customer to ensure they reflect the customer environment and naming conventions.

Pre-deployment Tasks

Pre-deployment tasks include procedures that do not directly relate to environment installation and configuration, but whose results will be needed at the time of installation. Examples of pre-deployment tasks are collection of hostnames, IP addresses, VLAN IDs, license keys, installation media, and so on. These tasks should be performed before the customer visit to decrease the time required onsite.

- Gather documents—Gather the vendor product installation documents. These are used throughout the text of this document to provide detail on setup procedures and deployment best practices for the various components of the solution.

- Gather tools—Gather the required and optional tools for the deployment. Use Table 9 to confirm that all equipment, software, and appropriate licenses are available before the deployment process.

- Gather data—Collect the customer-specific configuration data for networking, naming, and required accounts. Enter this information into the Customer Configuration Worksheets found later in this document for reference during the deployment process.

Table 9   Requisite Components

Layer	Description	Version or Release	Reference
Compute	Cisco UCS 6248UP Fabric Interconnect	2.1(2a)	
	Cisco UCS B200 M3	2.1(2a)	
Network	Cisco Nexus 5548UP Network Switch	6.0(2)N1(2a)	
Storage	EMC VNX5400 block	VNX Block OE 05.33	

Software	Cisco UCS Hosts	2012 R2	Microsoft Windows Server Datacenter Edition with Hyper-V Role enabled
	Cisco UCS PowerTool	1.0.0	
	Cisco Nexus 1000V (optional)	1.0	
	Cisco UCS SCVMM Extension (optional)	1.0	
	EMC Storage Integrator (ESI)	2.1.812.5137	
	EMC PowerPath	5.7	
	EMC Unisphere Host Agent	1.2.25.1.0163	

Configuration Workstation

It is recommended to have a Windows 8 or Windows Server 2012 workstation configured with certain pre-requisite software and joined to the same domain as the Hyper-V servers will be joined. Using a properly configured workstation makes the job of installing the solution easier. Here is the recommendation for software to be installed on the workstation.

Windows 8 (8.1) Workstation

- Install .NET Framework 3.5 by issuing the following command from an elevated command prompt: `Enable-WindowsOptionalFeature -Online -FeatureName NetFx3 -Source D:\sources\sxs`. This assumes the drive D: is the location of your Windows distribution media.

- Install the Remote Server Administration Tools. This is found at http://www.microsoft.com/en-us/download/details.aspx?id=28972. This is available in both a 32-bit and 64-bit distribution. Make sure you select the copy that matches your Windows 8 installation.

- After installing the Remote Server Administration Tools, install specific management tools.

  - Hyper-V Management Tools - issue the following command from an elevated command prompt: dism /online /enable-feature /all /featurename:Microsoft-Hyper-V-Tools-All

- Failover Clustering Tools - issue the following command from an elevated command prompt: `dism /online /enable-feature /featurename:RemoteServerAdministrationTools-Features-Clustering`

Windows Server 2012 (2012 R2)

- Install .NET Framework 3.5 by issuing the following command from an elevated command prompt: `Add-WindowsFeature -Name NET-Framework-Core -Source D:\sources\sxs`. This assumes the drive D: is the location of your Windows distribution media.

- Install the Hyper-V Management Tools by issuing this PowerShell cmdlet: `Install-WindowsFeature -Name RSAT-Hyper-V-Tools`

- Install the Windows Failover Clustering Tools by issuing this PowerShell cmdlet: Install-WindowsFeature -Name RSAT-Clustering

Tools on Both Workstations

- Naviseccli - Navisphere Secure Command Line Interface

- ESI (EMC Storage Integrator) - EMC PowerShell library

- Java 7 - required for running UCS Manager. Installed from the web.

- Cisco UCS PowerTool for UCSM, version 1.0.1. Installation instructions are found in section on Cisco Integration Components.

- PuTTY - an SSH and Telnet client helpful in initial configuration of the Cisco UCS 6248UP Fabric Interconnects. This program just needs to be copied to the system.

- PL-2303 USB-to-Serial driver - used to connect to the Cisco UCS 6248UP Fabric Interconnects through a serial cable connected to a USB port on the workstation. The download is a .zip file. Extract the executable from the .zip file and load it on the system.

You can download all the software listed in the revision table to this workstation. Some of the software, such as distribution media, can be placed into a file share for access by other systems.

There are several PowerShell scripts contained in the Sample PowerShell Scripts section of this document. These are sample scripts. They have been tested, but they are not warranted against errors. They are provided as is, and no support is assumed. But they assist greatly in getting the Hyper-V implementation configured properly and quickly. Some of the scripts will require editing to reflect customer-specific configurations. It is best to create a file share on the configuration workstation and place all the PowerShell scripts on that file share. Most of the scripts will run from the configuration workstation, but there may be some that have to be run locally on the server being configured. Having them available on a file share makes it easier to access them.

For each of the PowerShell scripts contained in Sample PowerShell Scripts, do the following.

1. Open Notepad (or Windows PowerShell ISE or your editor of choice)

2. Copy the contents of a section in Sample PowerShell Scripts

3. Paste into Notepad

4. Save the file using as the name of the file the name of the section in Sample PowerShell Scripts. While saving, ensure to set the "Save as type:" field to "All files (*)". For example, section Set-UcsHyperVRemoteMgmt.ps1 should be saved as "Set-UcsHyperVRemoteMgmt.ps1".

Physical Setup

Physical setup includes the following tasks:

1. Unpack and mount all hardware

2. Connect power cords and management connectivity to all hardware

3. Perform the initial setup steps for all hardware involved.

Make use of the appropriate vendor installation guides.

Cisco UCS Components

For information on mounting the hardware, see the Cisco UCS B-Series Hardware Installation Guide.

Care must be taken about efficient cooling and proper airflow while mounting any equipment in the data center. Similarly, you need to pay attention to power requirements of chassis, servers, and fabric interconnects.

Cisco UCS 5108 chassis, including its embedded blade servers and fabric extenders, do not require management connectivity as they are managed by the fabric interconnects. Fabric interconnects are deployed as a pair for high availability. Both the fabric interconnects require 100 Mbps peer connectivity for synchronizing the management plane between them. In addition, both the FIs require 1Gbps out-of-band management connectivity.

Cisco UCS Manager software runs on the Cisco UCS Fabric Interconnects. The Cisco UCS 6000 Series Fabric Interconnects expand the UCS networking portfolio and offer higher capacity, higher port density, and lower power consumption. These interconnects provide the management and communication backbone for the Cisco UCS B-Series Blades and Cisco UCS Blade Server Chassis. All chassis and the blade servers attached to the fabric interconnects are part of a single, highly available management domain. By supporting unified fabric, the Cisco UCS 6000 Series provides the flexibility to support LAN and SAN connectivity for all blade servers within its domain right at the configuration time. Typically deployed in redundant pairs, the Cisco UCS Fabric Interconnect provides uniform access to both network and storage, facilitating a fully virtualized environment.

Initial setup steps of Cisco UCS 6248UP Fabric Interconnects and the Cisco UCS Manager are similar to those of the Nexus 5548UP switches:

1. Connect the RJ-45 connector of the console cable to the primary fabric interconnect console port.

2. Configure the terminal emulator program on the host to match the following default port characteristics: 9600 baud, 8 data bits, 1 stop bit, and no parity.

3. Choose the CLI based initial setup configuration and provide basic information about the fabric interconnect cluster.

4. Connect two fabric interconnects using two 100 MbpsEthernet cables to create management plane cluster.

5. Repeat steps 1, 2 and 3 for the second fabric interconnect. The initial setup for the second fabric interconnect is relatively easier, as it forms a UCS management plane cluster with the pre-configured fabric interconnect, and assumes the role of secondary fabric interconnect.

Cisco UCS 5108 Chassis, Cisco UCS 2204XP Fabric Extenders and Cisco UCS B200 M3 blade servers would be part of the Cisco UCS Manager (UCSM) management domain, so no special configuration is required for them.

Preparing Cisco Nexus Switches

Cisco Nexus 5548UP switches are 1RU top of the rack 10Gbps Ethernet and Fibre Channel switches.

For information on how to deploy these switches, see Nexus 5548UP Product Documentation.

For initial configuration of these switches, follow these steps:

1. Connect the RJ-45 connector of the console cable to the Cisco Nexus 5548UP Switch console port.

2. Configure the terminal emulator program on the host to match the following default port characteristics: 9600 baud, 8 data bits, 1 stop bit, and no parity.

3. Type Setup at the switch prompt and follow the menu driven to configure the IP address on the management port and allow ssh to enable remote configuration of the switch.

4. Using the RJ-45 cable, connect to the upstream switch/router (or to the infrastructure network switch for managing remotely).

Preparing EMC VNX5400

Initial configuration and implementation of an EMC VNX5400 is covered in detail from the EMC documentation library. This is accessible at https://mydocs.emc.com/VNX/ and select Install VNX, using the VNX5400 series as the installation type. Installation documentation covers all areas from unpacking VNX storage components, installing in rack, provisioning power requirements and physical cabling.

When physically installed, the VNX should include the Disk Processing Enclosure (DPE) and two additional Disk Array Enclosures (DAEs), cabled as shown in Figure 17.

Figure 17   Cabling Diagram for VNX5400 with Two DAE
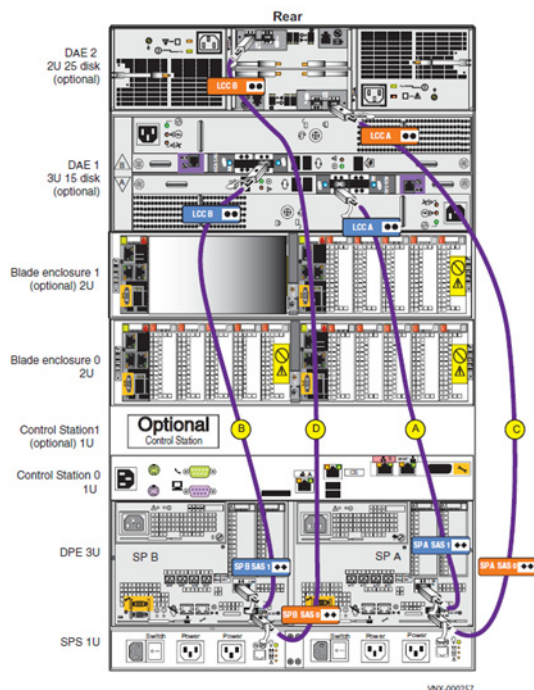
To complete software setup of the VNX array, it will be necessary to configure system connectivity including the creation of an Administrative user for the VNX array. The following worksheets (also found in the Installation documentation) list all required information, and can be used to facilitate the initial installation.

VNX Worksheets

With your network administrator, determine the IP addresses and network parameters you plan to use with the storage system, and record the information on the following worksheet. You must have this information to set up and initialize the system. The VNX5400 array is managed through a dedicated LAN port on the Control Station and each storage processor. These ports must share a subnet with the host you use to initialize the system. After initialization, any host on the same network and with a supported browser can manage the system through the management ports. This information can be recorded in Table 10.

Table 10   IPv4 Management Port Information

	IP Address	Subnet Mask	Gateway
CSO (optional)			
SP A			
SP B			

> **Note**   Do not use 128.221.1.248 through 128.221.1.255, 192.168.1.1, or 192.168.1.2 for an IPv4 IP Address.

While it is possible to implement IPv6 settings for the VNX array, the VSPEX implementation does not require it, and it is not implemented.

It is possible to more fully configure management IP addresses for the VNX5400 array. Table 11 lists some of the addresses you can optionally configure.

Table 11   Optional Control Station LAN Settings

Field	Value	Comments
CSO Primary hostname		
DNS domain		
Primary DNS Server		
Secondary DNS Server		
NTP Server		
Time Zone		

An administrative user account is required to be set for the array, and this account can be later utilized for executing NaviSecCLI commands, as well as for the ESI PowerShell environment used to provision LUNs from storage pools, and map those LUNs to hosts. Information required is outlined in Table 12.

Table 12   Login Information for the Storage System Administrator

Field	Description	Value
Username	sysasadmin (default)	Passwords are default and should be changed during installation or from within Unisphere.
Password	sysadmin (default)	

It is also necessary at this time to install the NaviSecCli command line interface from a supported Windows client environment. The client should have network access to the VNX5400 array for both HTTP/HTTPS access and for remote NaviSecCli command execution.

Installation media for the NaviSecCli utility, as well as ESI, are available by download at http://support.emc.com. The current version of the media should always be utilized. Installation of the utility is implemented through the typical application installation process for Windows-based systems.

After array installation, it will also be possible to connect to the VNX5400 array via the Unisphere graphical user interface at the IP address assigned to either SP-A or SP-B, or the control station in the event that a Unified version of the VNX is being implemented.
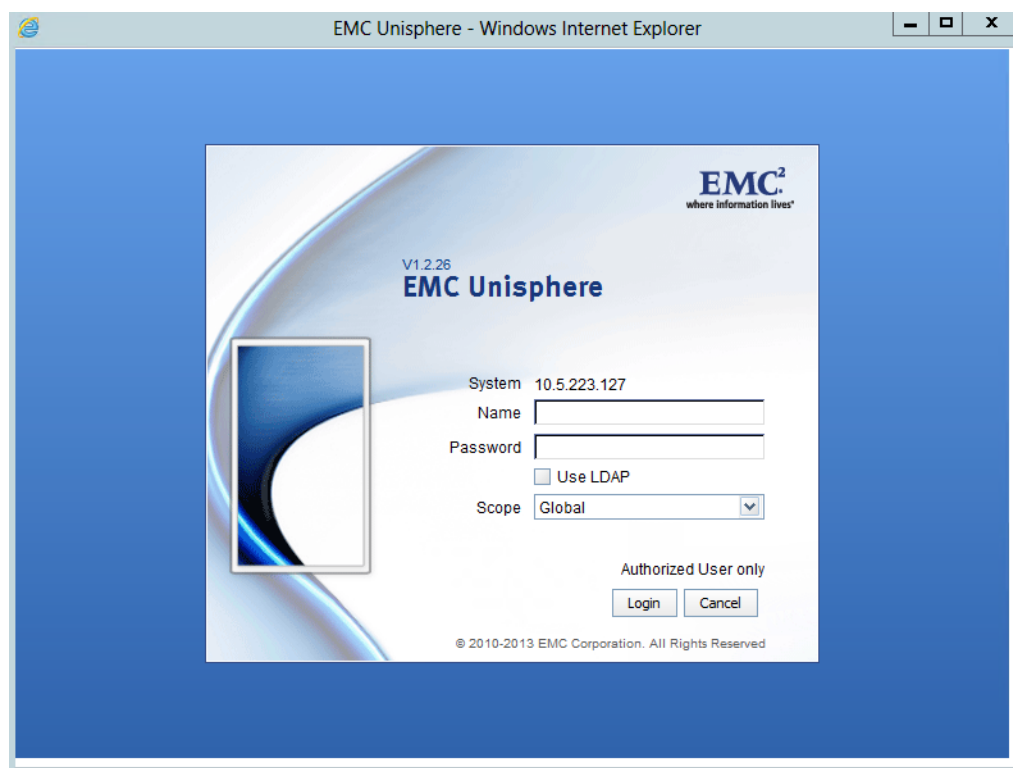
After entering appropriate login credentials, the Unisphere home page will be presented, providing an overview of the VNX5400 storage array. Summary alerts and errors will be visible as well as full management capabilities for all array features.
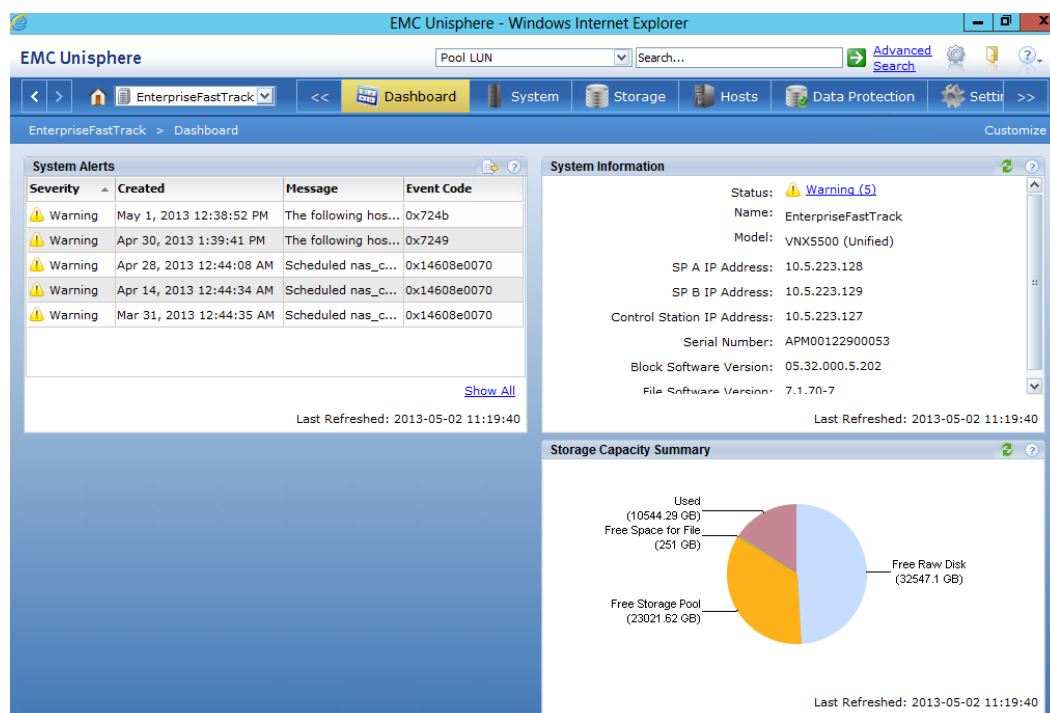
The following configuration also assumes that the array has been configured with:

- DAE - BUS 0 / Enclosure 1 45 drives
- DAE - BUS 0 / Enclosure 2 45 drives
- DAE - BUS 1 / Enclosure 0 20 drives

In the event that the physical configuration of the system differs in regards to the DAE placements, then modifications to the Bus Enclosure naming used subsequently will need to be appropriately altered.

Creating Storage Pools—Unisphere

1. Log into Unisphere and browse to the Storage Pool page.

2. Storage > Storage Configuration > Storage Pools

3. Click Create.

4. Under disks, manually select 45 600GB SAS drives from 3 DAE's and 2 EFD's.

5. Name the pool in Storage Pool Name. This name is used in the sample PowerShell scripts.

6. Click OK.
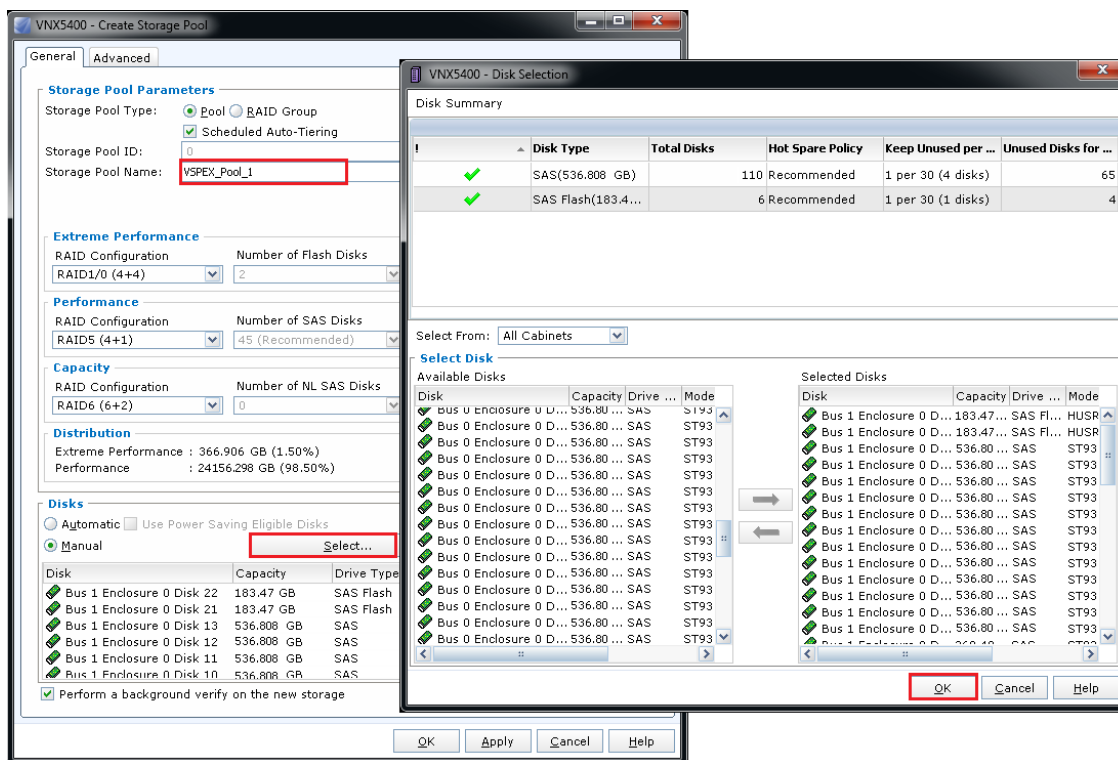
7. Under Extreme Performance, change the RAID configuration to RAID 1/0 (4+4). The number of Flash disks will automatically change to 2.

8. Click OK.

9. There will be a warning message which says that the number of Flash Disks should be in multiples of 8. Ignore the message and click yes to continue.

10. Repeat these steps to create 2 more Storage Pools.

✎

Note   The third Storage Pool will consist of 20 x 600GB SAS drives and 2 EFDs
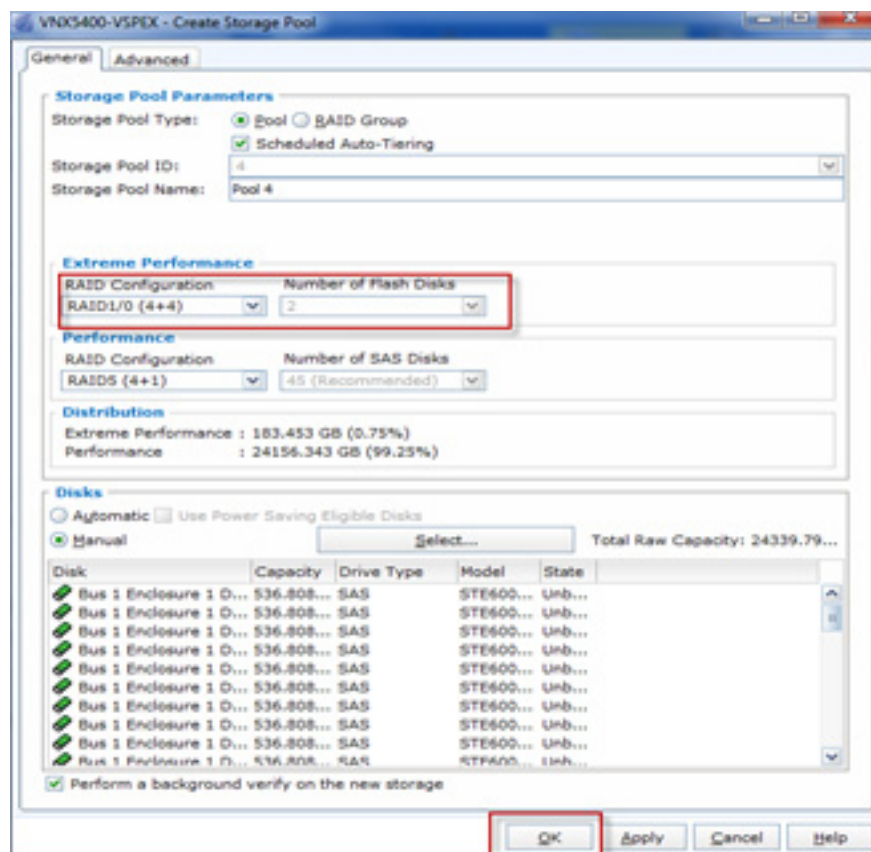
Creating Support for Clone Private LUNs

In the previous step storage pools were defined on the VNX array based on disks within the chassis. Additional RAID Group based LUNs are required to support clone private LUNs in the system. As part of the automation of Virtual Machine deployments, SnapView Clones can be utilized both through scripting and also through the SMI-S integration of System Center Virtual Machine Manager.

The use of clones requires the existence of at least one RAID group on the VNX5400 for storage of the clone private LUNs used in the cloning process. To help ensure proper performance for the number of virtual machines, all available free disks are configured into storage pools. Therefore, it will be necessary to create a RAID group using the system drives and configure just the clone private LUNs, and no others, on this RAID group.

1. Navigate to Storage > Storage Configuration > Storage Pools.

2. Select the RAID Groups tab.

3. Click Create.
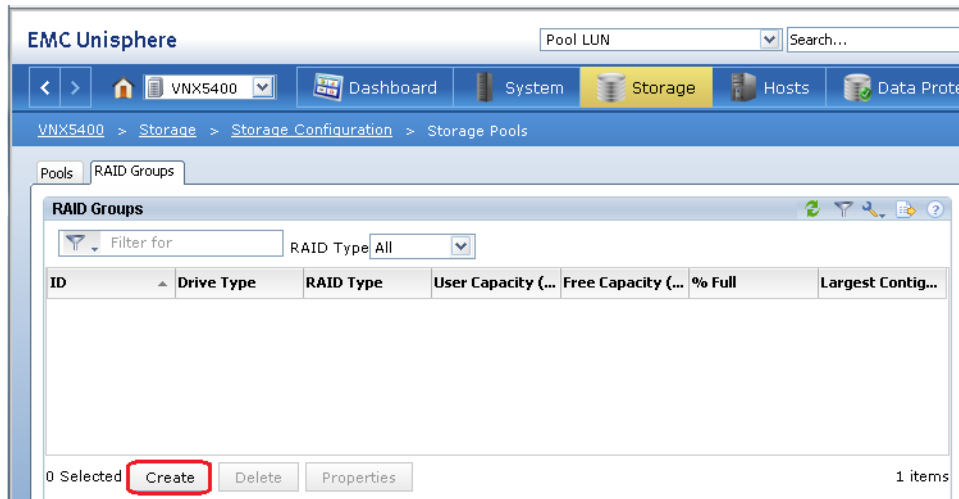
4. On the General tab, select RAID1/0 in the RAID Configuration drop down list.

5. Click the radio button by Manual.

6. Click Select to select the individual drives.
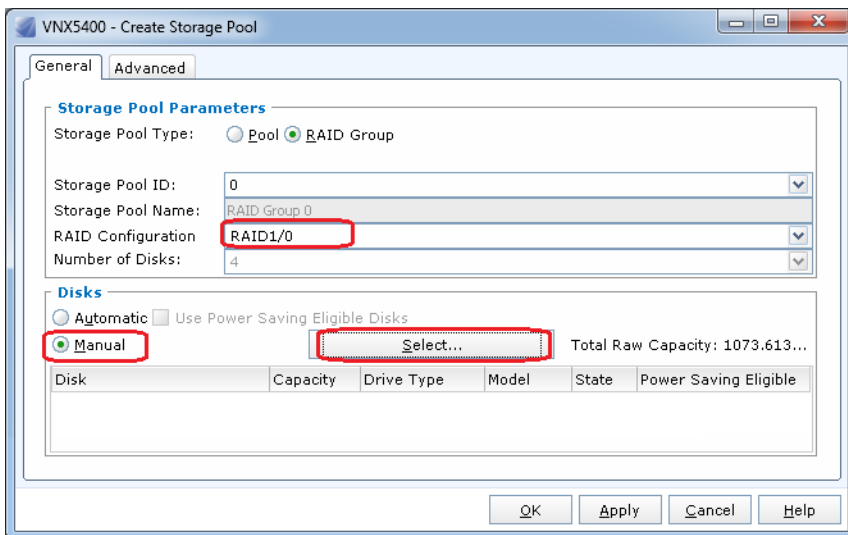
7. From the drop down menu select Bus 0 Enclosure 0.

8. Select the first four drives (0-3) from Bus 0 Enclosure 0 and add to Selected Disks. These should be the only disks selected.

9. Back on the General tab, ensure that RAID1/0 is selected as the RAID Configuration.
10. Click OK.
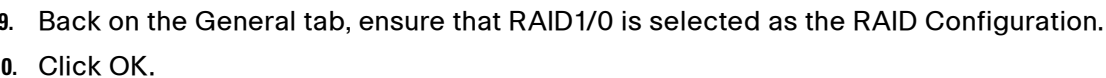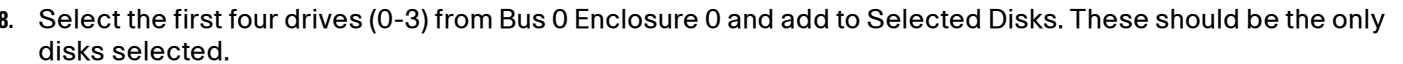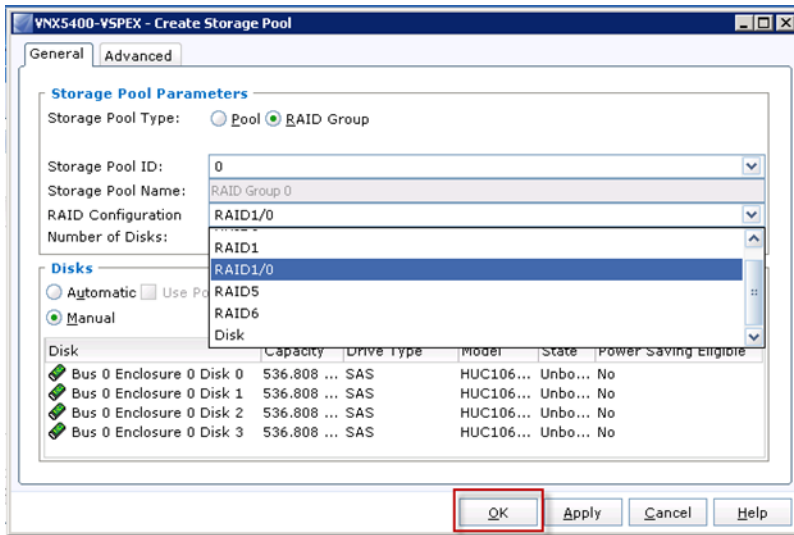
11. Navigate to Storage > LUNs.

12. Click Create.
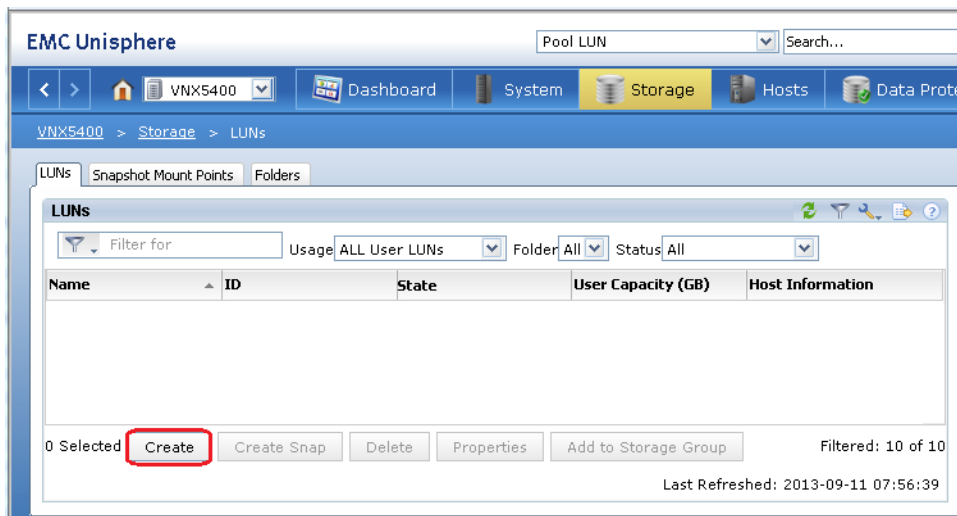
13. Select the RAID Group radio button.

14. Specify 1 GB for the User Capacity.

15. Select the Name radio button and specify a name for the LUN to be created.

> **Note** Two LUNs need to be created for clone private LUNs. One needs to be one SPA and the other on SPB.

16. After entering a value for the Name, click the Advanced tab.
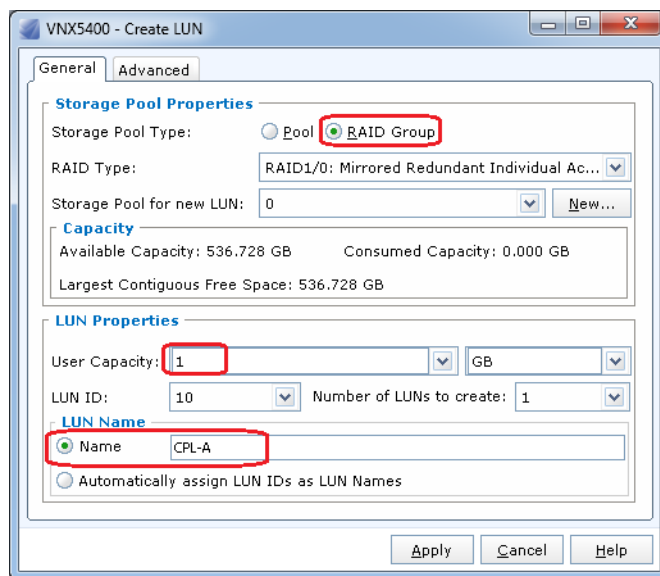
17. Make sure the radio button for one of the SPs is selected.

18. Click Apply.

19. Repeat, creating another LUN on the other SP.
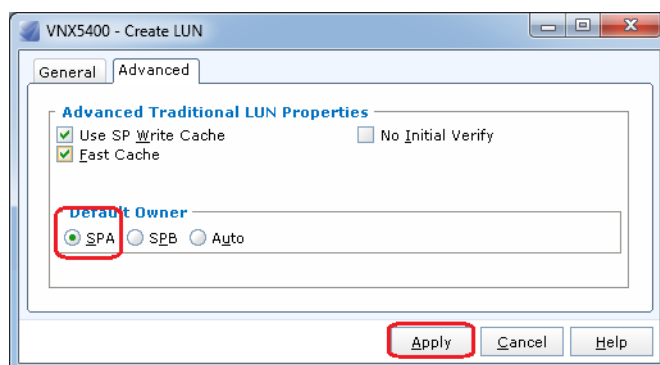
20. Navigate to Data Protection > Clones.

21. Click Configure Clone Settings.
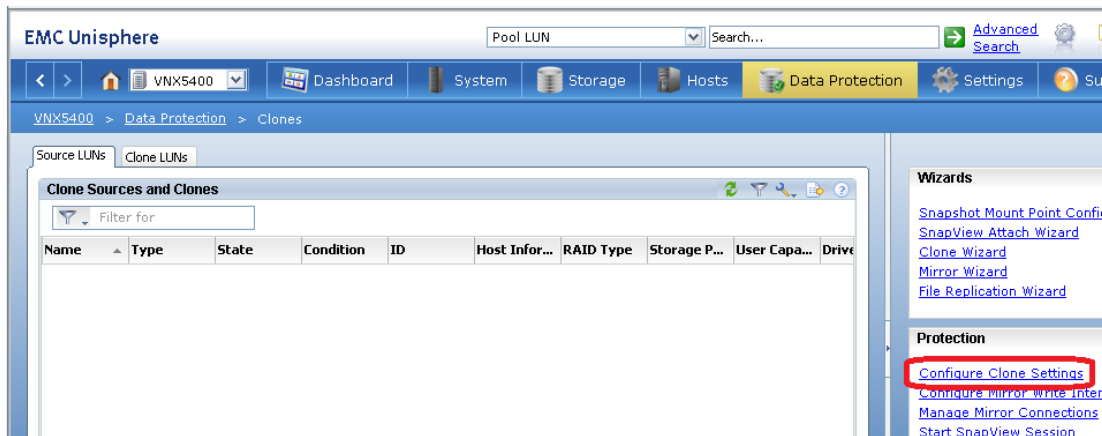
22. Expand each SPA. Select each of the newly created LUNs and click Add to add them to the clone private LUNs.

23. Click OK.
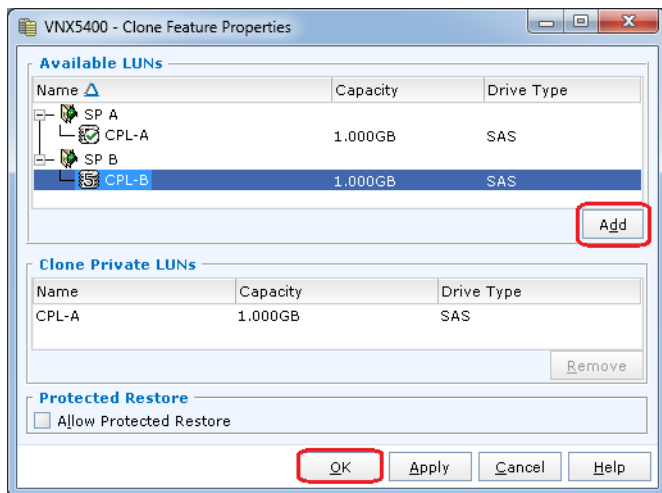
Record WWNN and WWPN Values

The Fibre Channel protocol uses Worldwide Node Names (WWNN) and Worldwide Port Names (WWPN) as addresses to uniquely identify endpoints in the Fibre Channel communication. These values must be recorded to be used in other steps in this configuration. For example, the WWPN values need to be entered into the UcsConfig.xml file used to configure UCS.

Obtain the WWPN information from the EMC VNX5400 by using the NaviSecCli that is installed on your Windows management system and record it in the WWNN/WWPN worksheet found in the Customer Configuration Worksheet section. Following is an example for obtaining the WWPNs from the connections to the VNX5400. It may be necessary to provide additional parameters, for login, password and scope options. The example below returns configuration information for all ports configured within the array. The WWPN for any given Fibre Channel port is derived from the last half of the SP UID entry. The first half of the SP UID is the WWNN entry. As an example, the WWNN of the array is 50:06:01:60:88:60:06:A1 and the WWPN of Port 4 on SP-A Port ID 4 is 50:06:01:64:08:60:06:A1.

```
C:\> navoseccli -address <<IP Address of SP-A or SP-B>> -User <<Admin user>> -Password <<Admin user
password>> -Scope 0 port -list -sp
```

```
SP Name:SP A
SP Port ID:     4
SP UID:         50:06:01:60:88:60:06:A1:50:06:01:64:08:60:06:A1
Link Status:    Up
Port Status:    Online
Switch Present:YES
Switch UID:     20:02:00:05:73:A1:DA:C1:20:02:00:05:73:A1:DA:C1
SP Source ID:   4
…
(report truncated)
```

Alternatively EMC Storage Integrator (ESI) PowerShell Toolkit can be used to obtain WWPN and IQN information like the following examples.

```
$targetports = Get-EmcTargetPort
$targetports | Where {$_.PortLocation -like "*Module 4*"} | fl PortLocation,
@{Expression={$_.Wwn.tostring().substring(0,23)};Label="WWNN"},
@{Expression={$_.Wwn.tostring().substring(24)};Label="WWPN"}

PortLocation : SP A I/O Module 4 Port 4
WWNN         : 50:06:01:60:88:60:06:A1
WWPN         : 50:06:01:64:08:60:06:A1
PortLocation : SP A I/O Module 4 Port 5
WWNN         : 50:06:01:60:88:60:06:A1
WWPN         : 50:06:01:65:08:60:06:A1
PortLocation : SP B I/O Module 4 Port 4
WWNN         : 50:06:01:60:88:60:06:A1
WWPN         : 50:06:01:6C:08:60:06:A1
PortLocation : SP B I/O Module 4 Port 5
WWNN         : 50:06:01:60:88:60:06:A1
WWPN         : 50:06:01:6D:08:60:06:A1
```

Or, the information can be found from EMC Unisphere. You will need to find the SP-Ports that are connected to the Cisco Nexus 5548 switches.

Cable Connectivity

The Customer Configuration Worksheet section of this document contains tables showing the cabling that was used to validate this configuration for up to 300 virtual machines. If a different configuration is implemented, update the worksheets for reference documentation.

Configure Cisco Nexus Switches

The following sections provide a detailed procedure for configuring the Cisco Nexus 5548 switches for use in this solution. Follow these steps precisely because failure to do so could result in an improper configuration. Make use of information captured in the Customer Configuration Worksheets to complete these steps.

In the Sample PowerShell Scripts section is a sample script, UcsConfig.ps1, for configuration UCS to reflect the cabling configuration shown in the Customer Configuration Worksheets. UcsConfig.ps1 reads the information from the UcsConfig.xml file, also contained in the Sample PowerShell Scripts sections. If changes are made to cabling configuration, the XML file will need to be edited to reflect those changes.

Set Up Initial Cisco Nexus 5548 Switch

These steps provide details for the initial Cisco Nexus 5548 Switch setup.

Cisco Nexus 5548 A

On initial boot and connection to the serial or console port of the switch, the NX-OS setup should automatically start.

1. Enter yes to enforce secure password standards.
2. Enter the password for the admin user.
3. Enter the password a second time to commit the password.
4. Enter yes to enter the basic configuration dialog.
5. Create another login account (yes/no) [n]: Enter.
6. Configure read-only SNMP community string (yes/no) [n]: Enter.
7. Configure read-write SNMP community string (yes/no) [n]: Enter.
8. Enter the switch name: <Nexus A Switch name> Enter.
9. Continue with out-of-band (mgmt0) management configuration? (yes/no) [y]: Enter.
10. Mgmt0 IPv4 address: <Nexus A mgmt0 IP> Enter.
11. Mgmt0 IPv4 netmask: <Nexus A mgmt0 netmask> Enter.
12. Configure the default gateway? (yes/no) [y]: Enter.
13. IPv4 address of the default gateway: <Nexus A mgmt0 gateway> Enter.
14. Enable the telnet service? (yes/no) [n]: Enter.
15. Enable the ssh service? (yes/no) [y]: Enter.
16. Type of ssh key you would like to generate (dsa/rsa):rsa.
17. Number of key bits <768-2048> :1024 Enter.
18. Configure the ntp server? (yes/no) [y]: Enter.
19. NTP server IPv4 address: <NTP Server IP> Enter.
20. Enter basic FC configurations (yes/no) [n]: Enter.
21. Would you like to edit the configuration? (yes/no) [n]: Enter.
22. Be sure to review the configuration summary before enabling it.
23. Use this configuration and save it? (yes/no) [y]: Enter.
24. Configuration may be continued from the console or by using SSH. To use SSH, connect to the mgmt0 address of Nexus A.

25. Log in as user admin with the password previously entered.

Cisco Nexus 5548 B

On initial boot and connection to the serial or console port of the switch, the NX-OS setup should automatically start.

1. Enter yes to enforce secure password standards.

2. Enter the password for the admin user.

3. Enter the password a second time to commit the password.

4. Enter yes to enter the basic configuration dialog.

5. Create another login account (yes/no) [n]: Enter.

6. Configure read-only SNMP community string (yes/no) [n]: Enter.

7. Configure read-write SNMP community string (yes/no) [n]: Enter.

8. Enter the switch name: <Nexus B Switch name> Enter.

9. Continue with out-of-band (mgmt0) management configuration? (yes/no) [y]: Enter.

10. Mgmt0 IPv4 address: <Nexus B mgmt0 IP> Enter.

11. Mgmt0 IPv4 netmask: <Nexus B mgmt0 netmask> Enter.

12. Configure the default gateway? (yes/no) [y]: Enter.

13. IPv4 address of the default gateway: <Nexus B mgmt0 gateway> Enter.

14. Enable the telnet service? (yes/no) [n]: Enter.

15. Enable the ssh service? (yes/no) [y]: Enter.

16. Type of ssh key you would like to generate (dsa/rsa):rsa

17. Number of key bits <768-2048> :1024 Enter.

18. Configure the ntp server? (yes/no) [y]: Enter.

19. NTP server IPv4 address: <NTP Server IP> Enter.

20. Enter basic FC configurations (yes/no) [n]: Enter.

21. Would you like to edit the configuration? (yes/no) [n]: Enter.

22. Be sure to review the configuration summary before enabling it.

23. Use this configuration and save it? (yes/no) [y]: Enter.

24. Configuration may be continued from the console or by using SSH. To use SSH, connect to the mgmt0 address of Nexus A.

25. Log in as user admin with the password previously entered.

Enable Appropriate Cisco Nexus Features

These commands enable the appropriate Cisco Nexus features.

Nexus A and Nexus B

```
config t
feature lacp
feature fcoe
feature npiv
feature vpc
feature fport-channel-trunk
feature interface-vlan
```

```
spanning-tree port type network default
spanning-tree port type edge bpduguard default
spanning-tree port type edge bpdufilter default
copy run start
```

Configure Fibre Channel Ports

These commands configure the necessary FC ports on the Nexus switches.

Nexus A and Nexus B

```
config t
slot 1
port 29-32 type fc
copy run start
reload.
```
The Nexus switch will reboot. This will take several minutes.

Create Necessary VLANs

These steps provide details for creating the necessary VLANs. Note that the SMB (or iSCSI) VLANs are not created on the Nexus switches. The SMB (or iSCSI) connections are made directly from the Fabric Interconnects to the EMC VNX array. The Nexus switches do not see this SMB (or iSCSI)-related traffic.

Nexus A and Nexus B

Following the switch reloads, log in with user admin and the password previously entered.These commands define the minimum VLANs used in this configuration.

```
config t
vlan <MGMT VLAN ID>
name Mgmt
exit
vlan <CSV VLAN ID>
name CSV
exit
vlan <Live Migration VLAN ID>
name LiveMigration
exit
vlan <VMaccess VLAN ID>
name VMaccess
exit
copy run start
```

Add Individual Port Descriptions for Troubleshooting

These add individual port descriptions for troubleshooting activity and verification.

Cisco Nexus 5548 A

```
config t
interface Eth1/1
description <Nexus B:Eth1/1>
exit
interface Eth1/2
description <Nexus B:Eth1/2>
exit
interface Eth1/17
description <UCSM A:Eth1/17>
exit
```

```
interface Eth1/18
description <UCSM B:Eth1/17>
exit
copy run start
```

Cisco Nexus 5548 B

```
config t
interface Eth1/1
description <Nexus A:Eth1/1>
exit
interface Eth1/2
description <Nexus A:Eth1/2>
exit
interface Eth1/17
description <UCSM B:Eth1/18>
exit
interface Eth1/18
description <UCSM A:Eth1/18>
exit
copy run start
```

Create Necessary Port Channels

These commands create the necessary port channels between devices.

Cisco Nexus 5548 A

```
config t
interface Po10
description vPC Peer-Link
exit
interface Eth1/1-2
channel-group 10 mode active
no shutdown
exit
interface Po201
description <VSPEX-UCS-A>
exit
interface Eth1/17
channel-group 201 mode active
no shutdown
exit
interface Po202
description <VSPEX-UCS-B>
exit
interface Eth1/18
channel-group 202 mode active
no shutdown
exit
copy run start
```

Cisco Nexus 5548 B

```
config t
interface Po10
description vPC Peer-Link
exit
interface Eth1/1-2
channel-group 10 mode active
no shutdown
exit
interface Po201
```

```
description <VSPEX-UCS-B>
exit
interface Eth1/17
channel-group 201 mode active
no shutdown
exit
interface Po202
description <VSPEX-UCS-A>
exit
interface Eth1/18
channel-group 202 mode active
no shutdown
exit
copy run start
```

Add Port Channel Configurations

These commands add prot channel configurations.

Cisco Nexus 5548 A

```
config t
interface Po10
switchport mode trunk
switchport trunk native vlan <Native VLAN ID>
switchport trunk allowed vlan <MGMT VLAN ID, CSV VLAN ID, LiveMigration VLAN ID, VMaccess VLAN ID, VEM
VLAN ID>
spanning-tree port type network
no shutdown
exit
interface Po201
switchport mode trunk
switchport trunk native vlan <MGMT VLAN ID>
switchport trunk allowed vlan < MGMT VLAN ID, CSV VLAN ID, LiveMigration VLAN ID, VMaccess VLAN ID, VEM
VLAN ID >
spanning-tree port type edge trunk
no shut
exit
interface Po202
switchport mode trunk
switchport trunk native vlan <MGMT VLAN ID>
switchport trunk allowed vlan < MGMT VLAN ID, CSV VLAN ID, LiveMigration VLAN ID, VMaccess VLAN ID, VEM
VLAN ID >
spanning-tree port type edge trunk
no shut
exit
copy run start
```

Cisco Nexus 5548 B

```
config t
interface Po10
switchport mode trunk
switchport trunk native vlan <Native VLAN ID>
switchport trunk allowed vlan < MGMT VLAN ID, CSV VLAN ID, LiveMigration VLAN ID, VMaccess VLAN ID, VEM
VLAN ID >
spanning-tree port type network
no shutdown
exit
interface Po201
switchport mode trunk
switchport trunk native vlan <MGMT VLAN ID>
```

```
switchport trunk allowed vlan MGMT VLAN ID, CSV VLAN ID, LiveMigration VLAN ID, VMaccess VLAN ID, VEM
VLAN ID  >
spanning-tree port type edge trunk
no shut
exit
Interface Po202
switchport mode trunk
switchport trunk native vlan <MGMT VLAN ID>
switchport trunk allowed vlan MGMT VLAN ID, CSV VLAN ID, LiveMigration VLAN ID, VMaccess VLAN ID, VEM
VLAN ID >
spanning-tree port type edge trunk
no shut
exi
copy run start
```

Configure Virtual Port Channels

These commands configure virtual port channels (vPCs)

Cisco Nexus 5548 A

```
config t
vpc domain <Nexus vPC domain ID>
role priority 10
peer-keepalive destination <Nexus B mgmt0 IP> source <Nexus A mgmt0 IP>
exit
interface Po10
vpc peer-link
exit
interface Po201
vpc 201
exit
interface Po202
vpc 202
exit
copy run start
```

Cisco Nexus 5548 B

```
config t
vpc domain <Nexus vPC domain ID>
role priority 20
peer-keepalive destination <Nexus A mgmt0 IP> source <Nexus B mgmt0 IP>
exit
interface Po10
vpc peer-link
exit
interface Po201
vpc 201
exit
interface Po202
vpc 202
exit
copy run start
```

Configure Fibre Channel Ports

Nexus A and Nexus B

```
config t
interface fc1/29
switchport trunk mode off
```

```
no shutdown
exit
interface fc1/30
switchport trunk mode off
no shutdown
exit
interface fc1/31
switchport trunk mode off
no shutdown
exit
interface fc1/32
switchport trunk mode off
no shutdown
exit
copy run start
```

Link Into Existing Network Infrastructure

Depending on the available network infrastructure, several methods and features can be used to uplink the private cloud environment. If an existing Cisco Nexus environment is present, Cisco recommends using vPCs to uplink the Cisco Nexus 5548 switches included in the private cloud environment into the infrastructure. The previously described procedures can be used to create an uplink vPC to the existing environment.

Initial Configuration of Cisco Unified Computing System

The following information provides a detailed procedure for the initial configuration of the Cisco UCS Manager. These steps should be followed precisely because a failure to do so could result in an improper configuration. You will need information captured on the customer configuration worksheets.

Cisco UCS 6248 A

1. Connect to the console port on the first Cisco UCS 6248 fabric interconnect.
2. At the prompt to enter the configuration method, enter console to continue.
3. If asked to either do a new setup or restore from backup, enter setup to continue.
4. Enter y to continue to set up a new fabric interconnect.
5. Enter y to enforce strong passwords.
6. Enter the password for the admin user.
7. Enter the same password again to confirm the password for the admin user.
8. When asked if this fabric interconnect is part of a cluster, answer y to continue.
9. Enter A for the switch fabric.
10. Enter the cluster name for the system name.
11. Enter the Mgmt0 IPv4 address.
12. Enter the Mgmt0 IPv4 netmask.
13. Enter the IPv4 address of the default gateway.
14. Enter the cluster IPv4 address.
15. To configure DNS, answer y.
16. Enter the DNS IPv4 address.
17. Answer y to set up the default domain name.

18. Enter the default domain name.

19. Review the settings that were printed to the console, and if they are correct, answer yes to save the configuration.

20. Wait for the login prompt to make sure the configuration has been saved.

Cisco UCS 6248 B

1. Connect to the console port on the second Cisco UCS 6248 fabric interconnect.

2. When prompted to enter the configuration method, enter console to continue.

3. The installer detects the presence of the partner fabric interconnect and adds this fabric interconnect to the cluster. Enter y to continue the installation.

4. Enter the admin password for the first fabric interconnect.

5. Enter the Mgmt0 IPv4 address.

6. Answer yes to save the configuration.

7. Wait for the login prompt to confirm that the configuration has been saved.

Logging Into Cisco UCS Manager

These steps provide details for logging into the Cisco UCS environment:

1. Open a Web browser and navigate to the Cisco UCS 6248 fabric interconnect cluster address.

2. A warning displays about the security certificate. Click Continue.
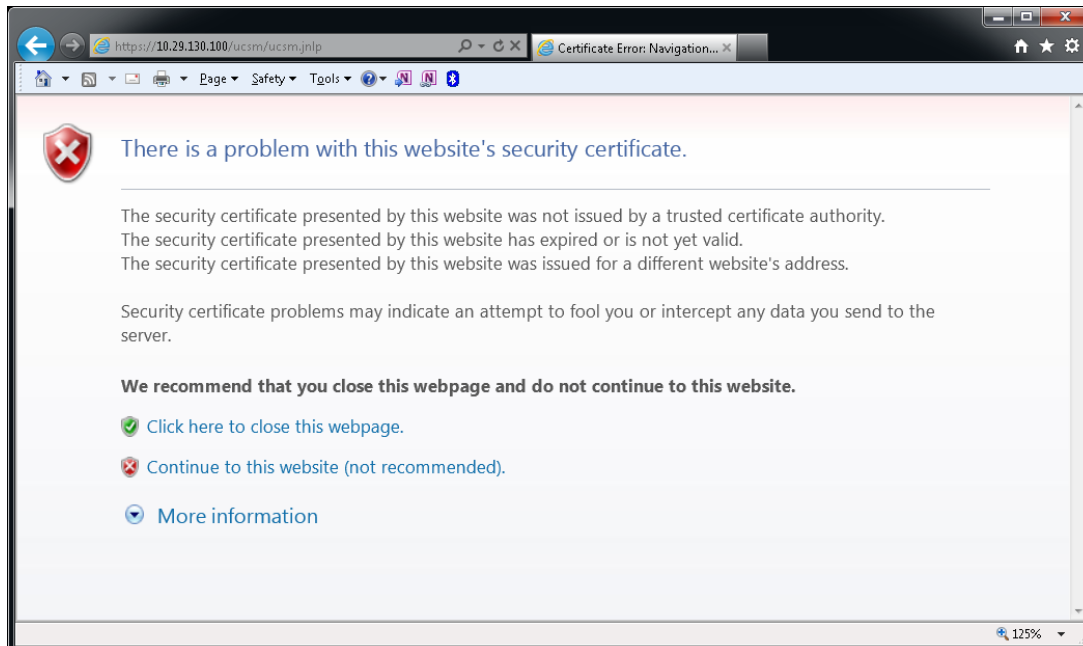
3. Select the Launch link to download the Cisco UCS Manager software.

4. If prompted to accept security certificates, accept as necessary.
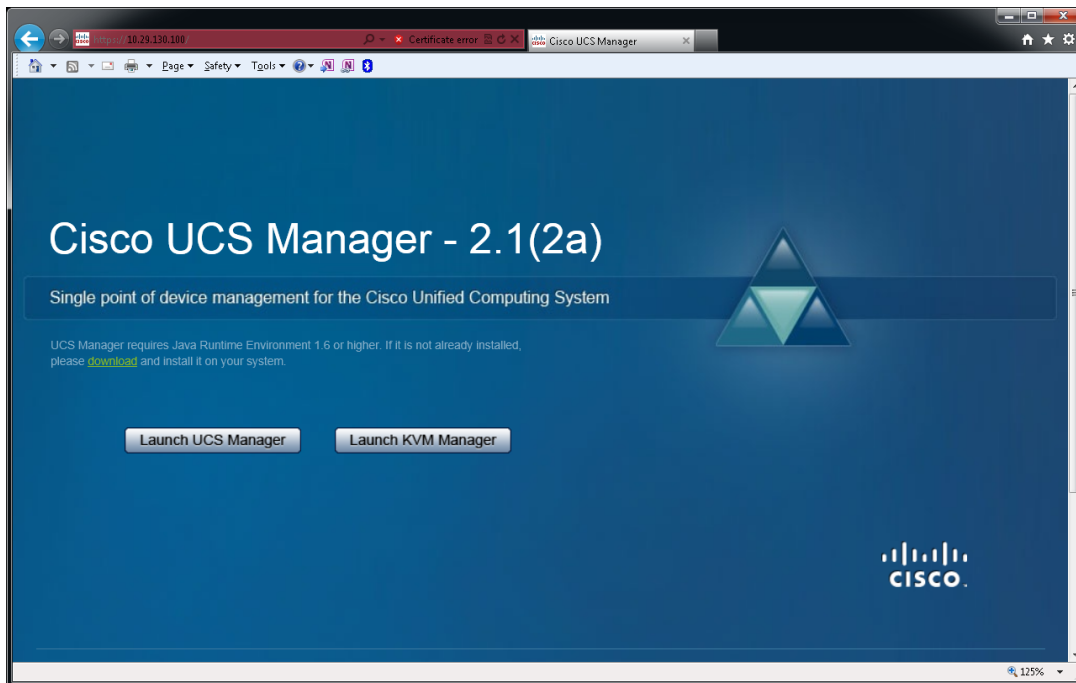
5. When prompted, enter admin for the username and enter the administrative password and click Login to log in to the Cisco UCS Manager software.
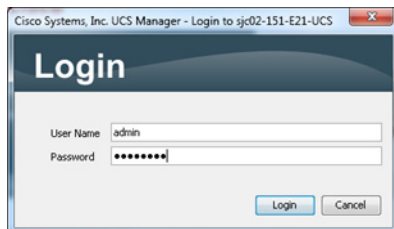
Create the VSPEX Environment with Cisco UCS PowerTool

Contained in the Sample PowerShell Scripts of this document are the following two files:

*   UcsConfig.ps1—this script reads UcsConfig.xml, using the customer information entered in it to define the configuration of UCS cabling and various pools, policies, and templates.
*   UcsConfig.xml—this file contains customer provided information for values and names to assign to pools, policies, and templates. When the Customer Configuration Worksheets are completed, the information should be transferred to the appropriate locations within this XML file. Care should be used when editing this file to ensure that the XML structure defined is not altered. f altered, it will likely cause the UcsConfig.ps1 script to fail.

It is recommended that you review and validate these files before running. The script takes about 30 seconds to run to configure Cisco Unified Computing System according to the VSPEX validated design. This validation can save a significant amount of time.

Create the VSPEX Environment with Cisco UCS Manager

The following steps detail the basics for configuring the VSPEX environment by using the Cisco UCS Manager instead of using the UcsConfig.ps1 PowerShell script. It is assumed some basic knowledge of Cisco UCS Manager. For example, when creating VLANs, it shows the basic procedure to create a VLAN, but it does not step through creating every VLAN.

Synchronize Cisco UCS to NTP

These steps provide details for synchronizing the Cisco UCS environment to the NTP server:

1. Select the Admin tab at the top of the left window.
2. Select All > Timezone Management.
3. In the right pane, select the appropriate timezone in the Timezone drop-down menu.
4. Click Add NTP Server.
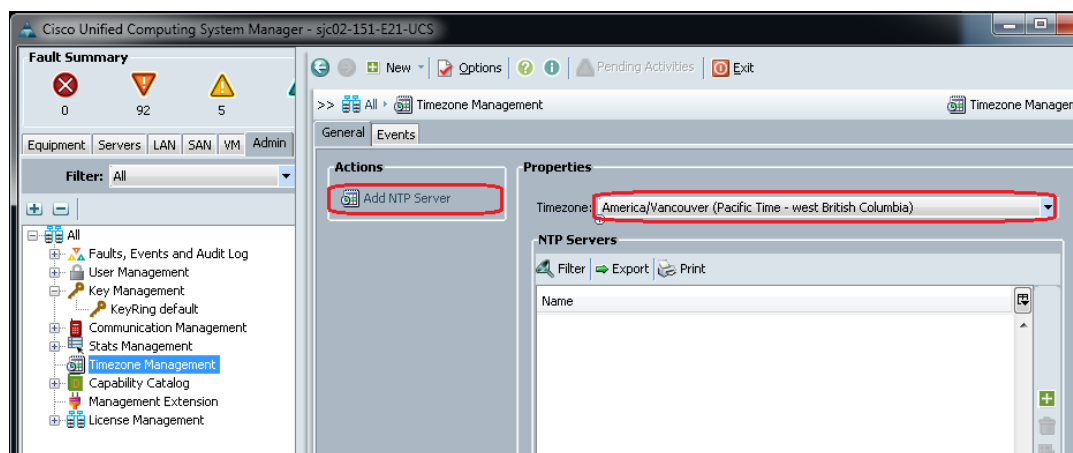
5. Input the NTP server IP and click OK.
6. On the next screen click Save Changes and then OK.
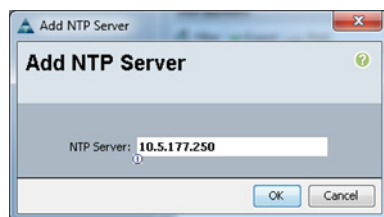
Add a Block of IP Addresses for KVM Access

These steps provide details for creating a block of KVM IP addresses for server access in the Cisco UCS environment:

1. Select the LAN tab at the top of the left window.
2. Select Pools > root > IP Pools > IP Pool ext-mgmt.

3. Select the appropriate radio button for the preferred assignment order.

4. Select Create Block of IP Addresses.
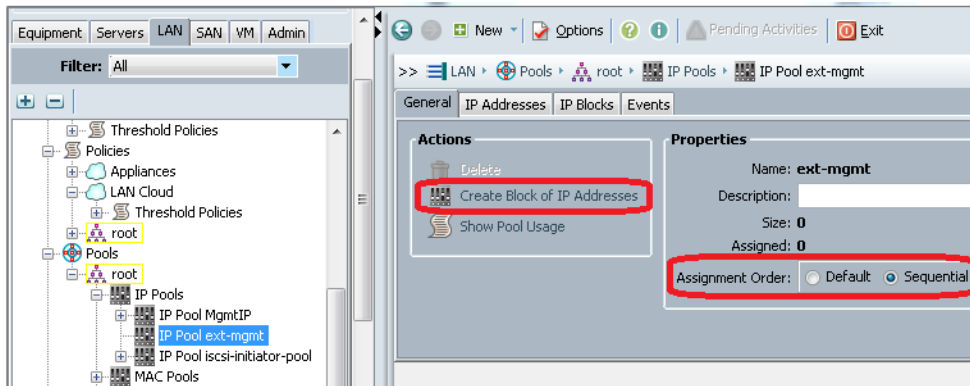
5. Enter the starting IP address of the block and number of IPs needed as well as the subnet and gateway information.

6. Click OK to create the IP block.

7. Click OK in the message box.
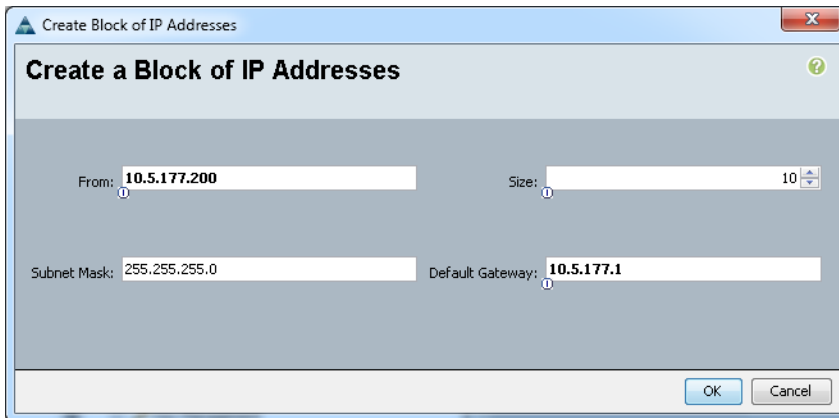
The ext-mgmt pool is the default management pool. By default, IP addresses are assigned to the physical servers as they are recognized. It is also possible to create a separate pool and then assign it Service Profiles (Management IP Address) when they are assigned to a physical server. This is important if you plan on using SMI-S for management of the servers, as then the IP address follows the service profile instead of the physical server.

Edit the Chassis Discovery Policy

These steps provide details for modifying the chassis discovery policy as the base architecture includes two uplinks from each fabric extender installed in the Cisco UCS chassis:

1. Navigate to the Equipment tab in the left pane.

2. In the right pane, click the Policies tab.

3. Under Global Policies, change the Chassis Discovery Policy to 2-link.

4. Select the Port Channel radio button for the Link Grouping Preference.

5. Select the desired settings for the Rack Server Discovery Policy, Power Policy, and MAC Address Table Aging.

6. Click Save Changes.
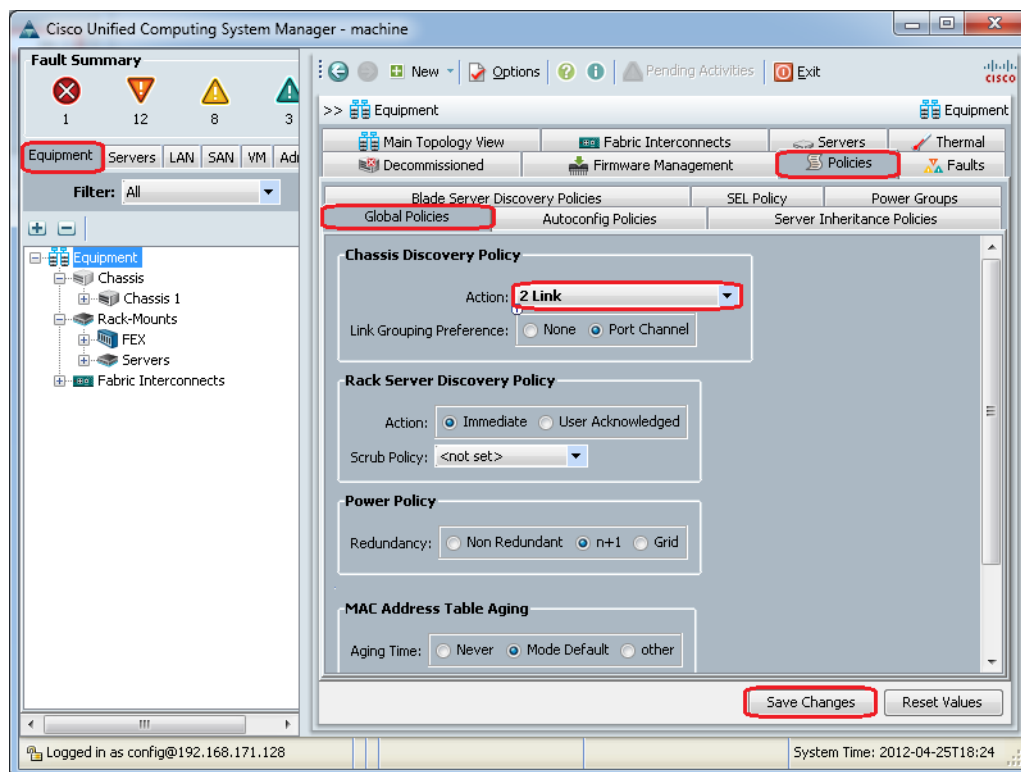
7. After setting the Chassis Discovery Policy, acknowledge each chassis to be managed.

8. On the Equipment tab, select Chassis 1 in the left pane.

9. Click Acknowledge Chassis.
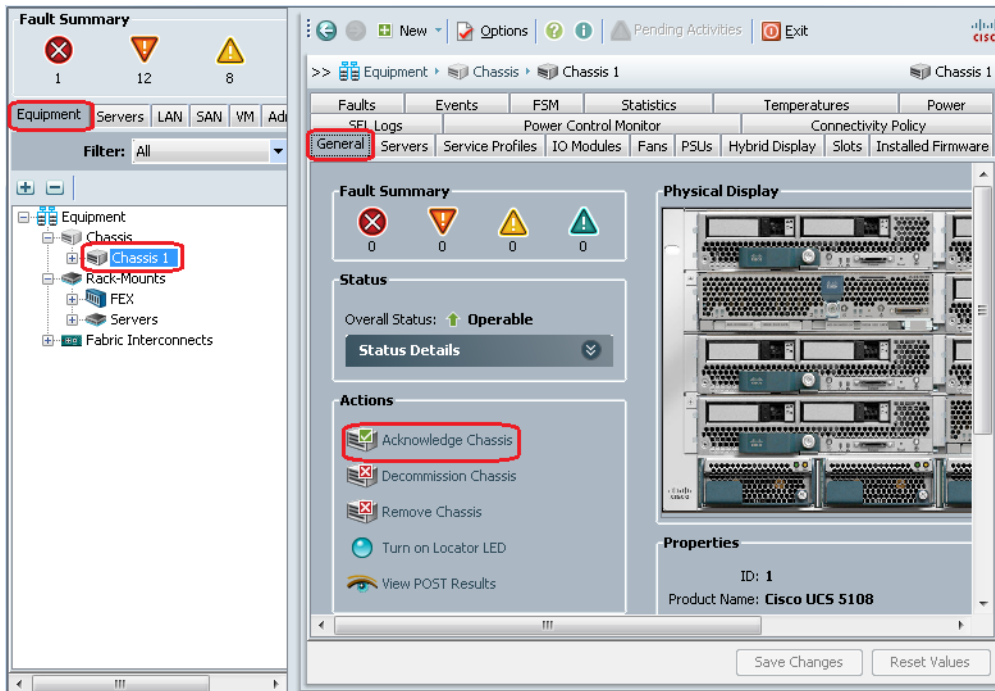
Create an Organization

The following steps provide details for configuring an organization in the Cisco UCS environment. Organizations are used as a means to organize and restrict access to various groups within the IT organization, thereby enabling multi-tenancy of the compute resources.

> ✎
>
> **Note**   This document assumes the use of an Organization for VSPEX and the necessary steps are included below.

From the New... menu at the top of the window, select Create Organization.
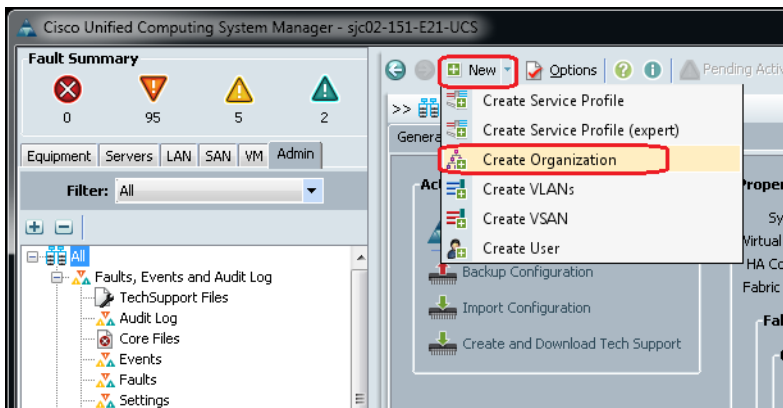
10.   Enter a name for the organization.

11. Enter a description for the organization (optional).

12. Click OK.

13. In the message box that displays, click OK.
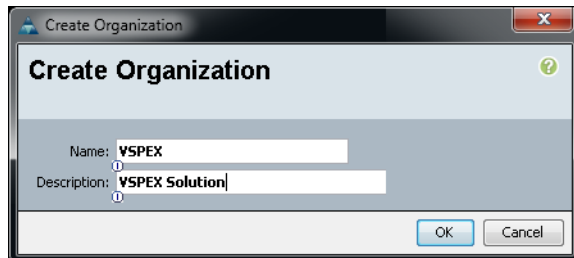
Enable Quality of Service in Cisco UCS Fabric

These steps provide details for enabling the quality of service in the Cisco UCS Fabric and setting Jumbo frames:

1. Select the LAN tab at the top left of the window.

2. Go to LAN Cloud > QoS System Class.

3. In the right pane, click the General tab.

4. On the Platinum, Gold, and Best Effort rows, type 9000 in the MTU boxes.

5. Click Save Changes in the bottom right corner.

6. Click OK to continue.
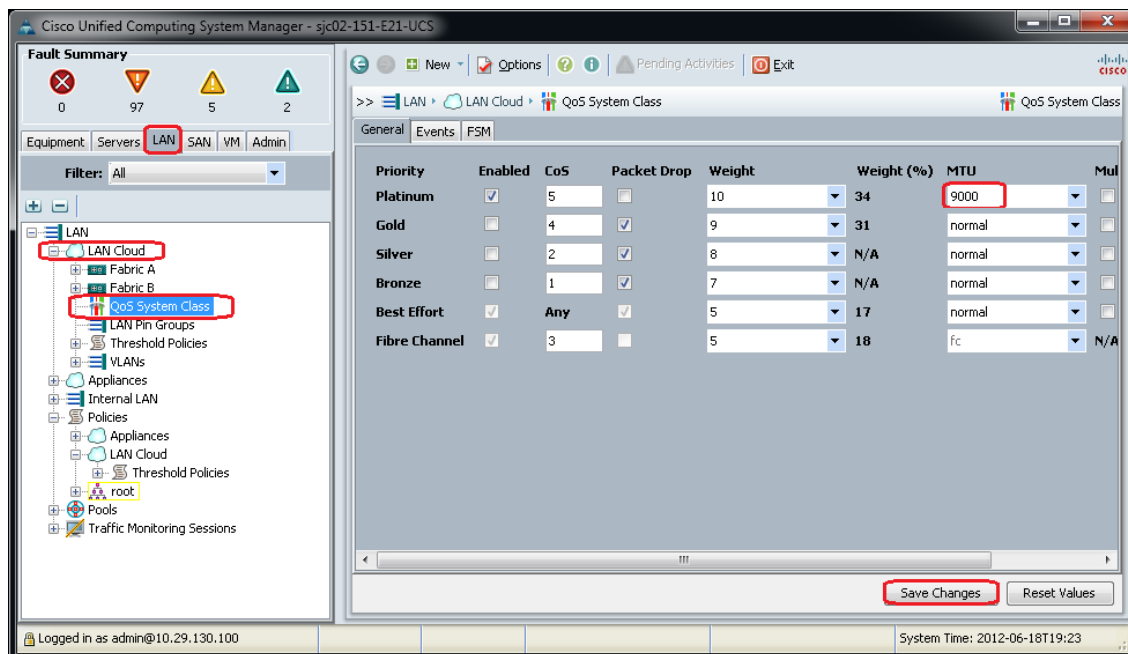
7. Select the LAN tab on the left of the window.

8. Go to LAN > Policies > Root.

9. Right-click QoS Policies.

10. Select Create QoS Policy.
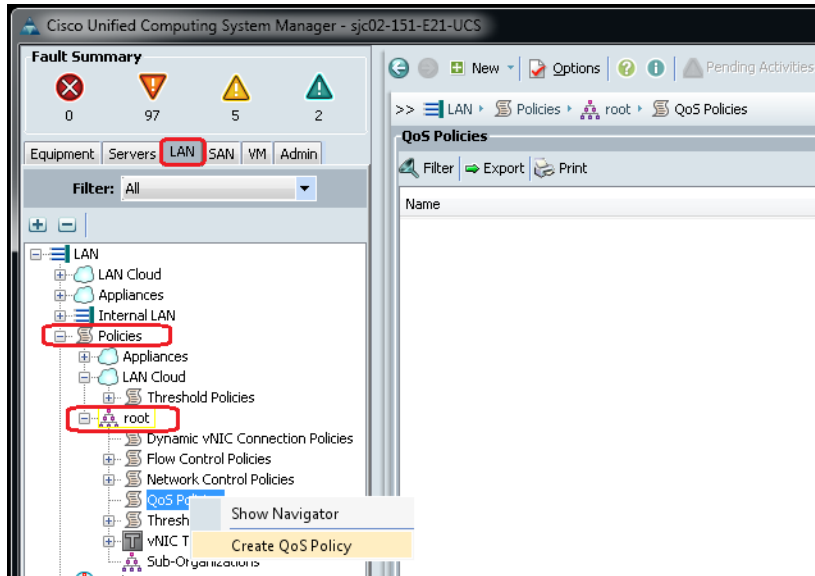
11. Enter <LiveMigration> as the QoS Policy name.

12. Change the Priority to Platinum. Leave Burst (Bytes) set to 10240. Leave Rate (Kbps) set to line-rate.

13. Leave Host Control set to None.

14. Click OK.
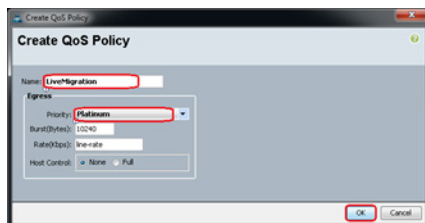
Create a Power Control Policy

These steps provide details for creating a Power Control Policy for the Cisco UCS environment:

1. Select the Servers tab at the top left of the window.

2. Go to Policies > root.

3. Right-click Power Controller Policies.

4. Select Create Power Control Policy
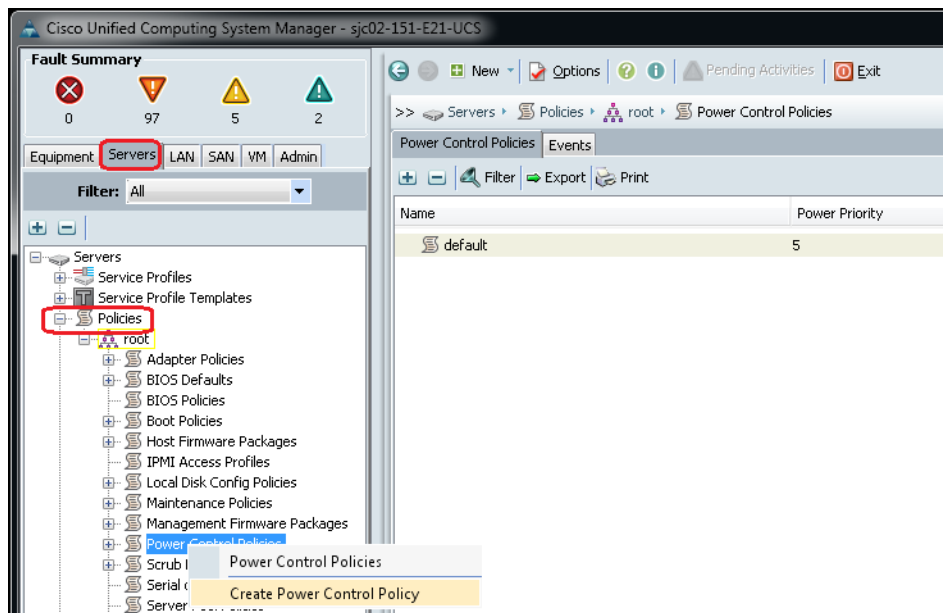
5. Enter <No-Cap> as the power control policy Name.

6. Change the Power Capping to No Cap.

7. Click OK to complete creating the host firmware package.

8. Click OK.
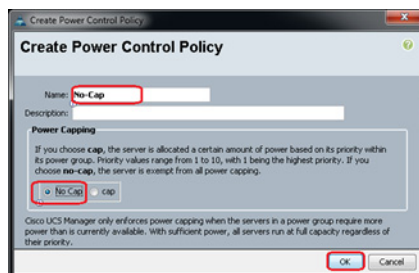
Create a Scrub Policy

Scrub policies define what is erased when a service profile is disassociated from a server. If the disk field is set to Yes, when a service profile containing this scrub policy is disassociated from a server, all data on the server local drives is completely erased. If this field is set to No, the data on the local drives is preserved, including all local storage configuration.

1. Select the Servers tab at the top left of the window.

2. Go to Policies > root.

3. Right-click Scrub Policies.

4. Select Create Scrub Policy.
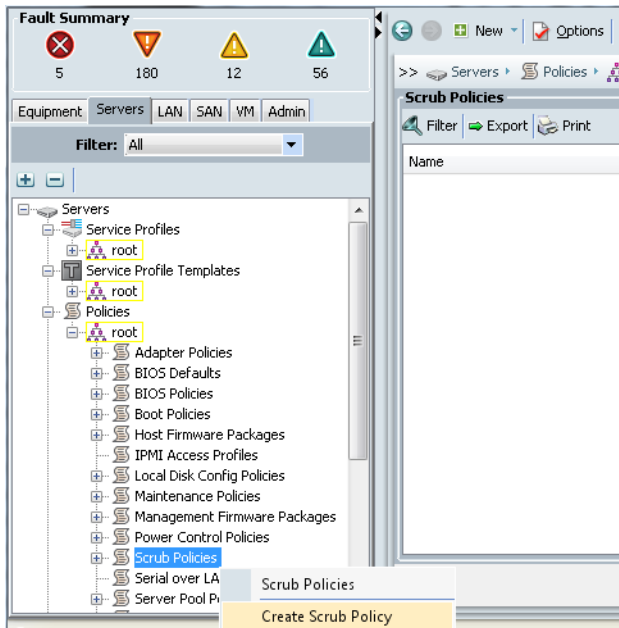
5. Enter <NoScrub> as the disk scrub policy Name.

6. Select the radio button by No.

7. Click OK to complete creating the scrub policy.

8. Click OK.
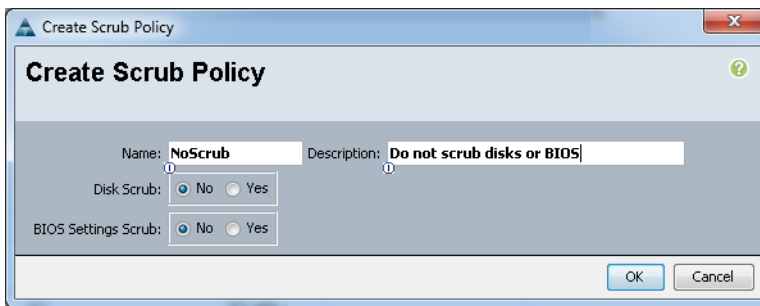
Create Maintenance Policy

When a service profile is associated with a server, or when changes are made to a service profile that is already associated with a server, the server needs to be rebooted to complete the process. The Reboot Policy field of the maintenance policy determines when the reboot occurs for servers associated with any service profiles that include this maintenance policy.

1. Select the Servers tab at the top left of the window.

2. Go to Policies > root.

3. Right-click Maintenance Policies.

4. Select Create Maintenance Policy.
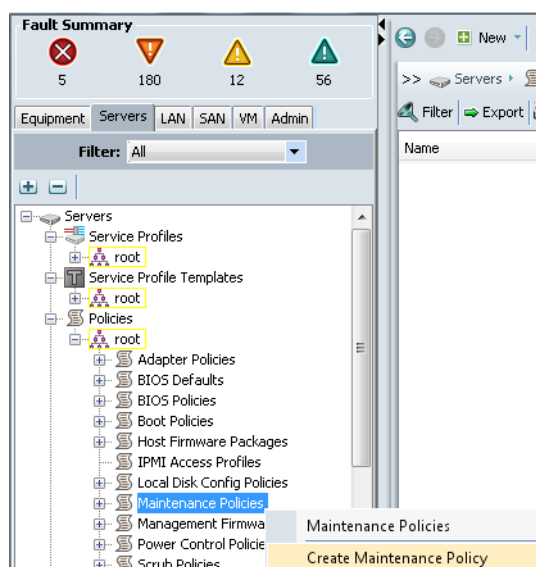
5. Enter <UserAck> as the maintenance policy Name.

6. Optionally, enter a Description.

7. Select the radio button by the desired reboot policy.

8. Click OK to complete creating the scrub policy.

9. Click OK.
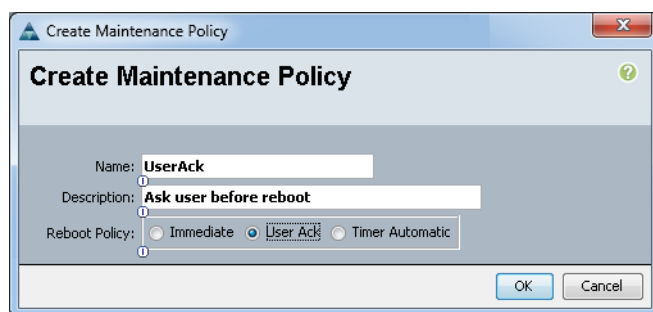
Create a Local Disk Configuration Policy

These steps provide details for creating a local disk configuration for the Cisco UCS environment, which is necessary if the servers in question do not have a local disk.

✎ **Note** This policy is recommended for virtualization servers even if they do have local disks. Flexibility is a key component of virtualization, so it is best to have configurations as loosely tied to physical hardware as possible. By not making provision for local disks and SAN booting, you ensure that moving the profile to another system will not create an environment that will lose something as it moves.

1. Select the Servers tab on the left of the window.

2. Go to Policies > root.

3. Right-click Local Disk Config Policies.

4. Select Create Local Disk Configuration Policy.
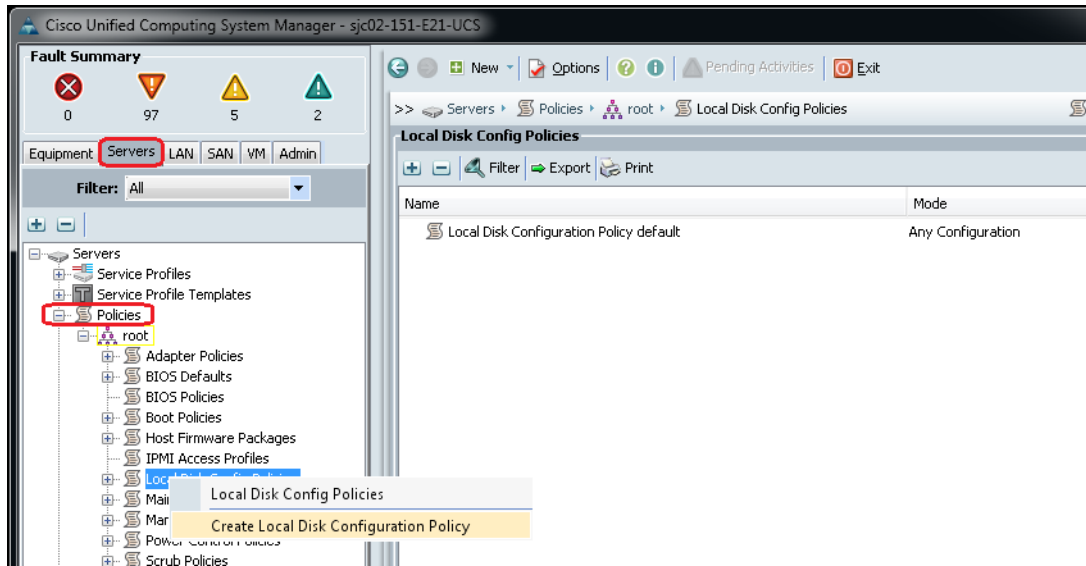
5. Enter <SAN-Boot> as the local disk configuration policy Name.

6. Change the Mode to No Local Storage. Uncheck the Protect Configuration box.

7. Click OK to complete creating the host firmware package.

8. Click OK.
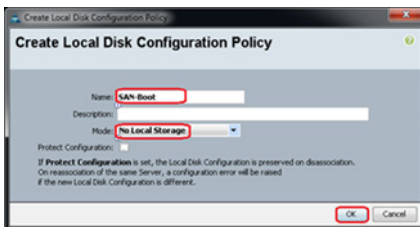
Create a Server BIOS Policy

These steps provide details for creating a server BIOS policy for the Cisco UCS environment. It is recommended to have a BIOS policy available that disables quiet boot in order to assist in debugging boot issues.

1. Select the Servers tab on the left of the window.

2. Go to Policies > root.

3. Right-click BIOS Policies.

4. Select Create BIOS Policy.
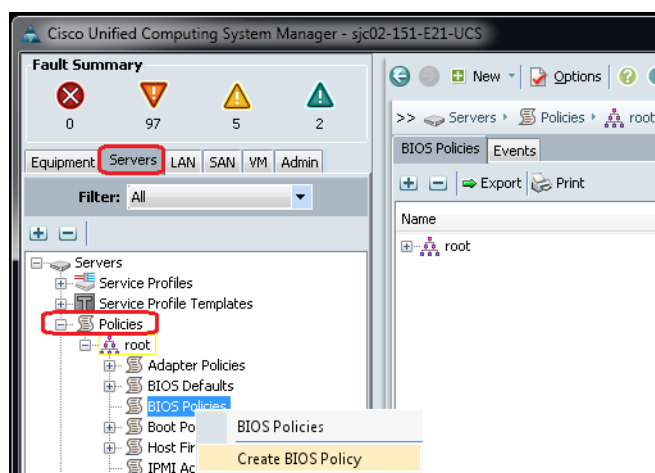
5. Enter <FastTrack3-Host> as the BIOS policy Name.

6. Change the Quiet Boot property to Disabled.

7. Click Finish to complete creating the BIOS policy.

8. Click OK.
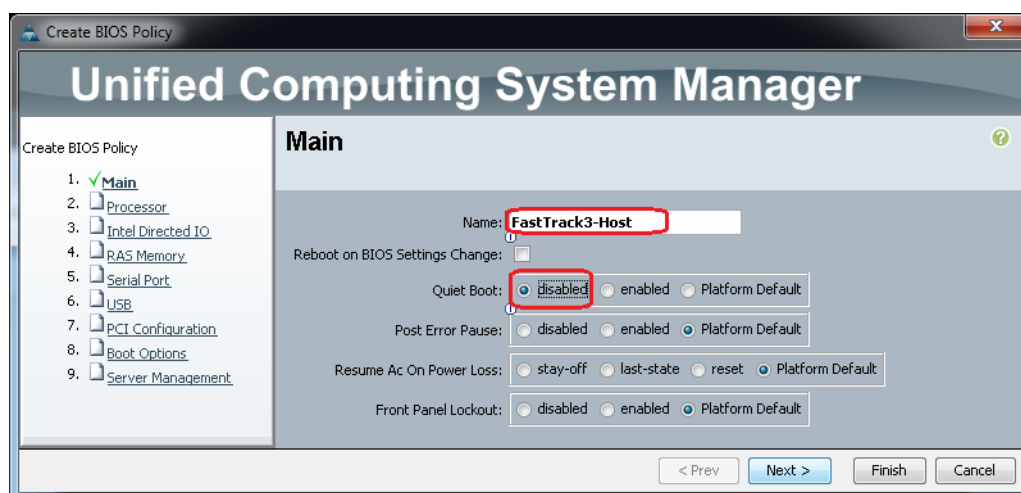
Enable Fabric Interconnect Port Definitions

These steps provide details for enabling Fibre Channel, server, and uplinks ports:

1. Select the Equipment tab on the top left of the window.

2. Select Equipment > Fabric Interconnects > Fabric Interconnect A (primary) > Fixed Module.

3. Expand the Unconfigured Ethernet Ports section.

4. Select the ports that are connected to the Cisco UCS chassis (2 per chassis).

5. Click Reconfigure, then select Configure as Server Port from the drop-down menu.

6. A prompt displays asking if this is what you want to do. Click Yes, then OK to continue.

7. Repeat for other ports, selecting the appropriate configuration from the Customer Configuration Worksheets.

8. Repeat for Fabric Interconnect B.
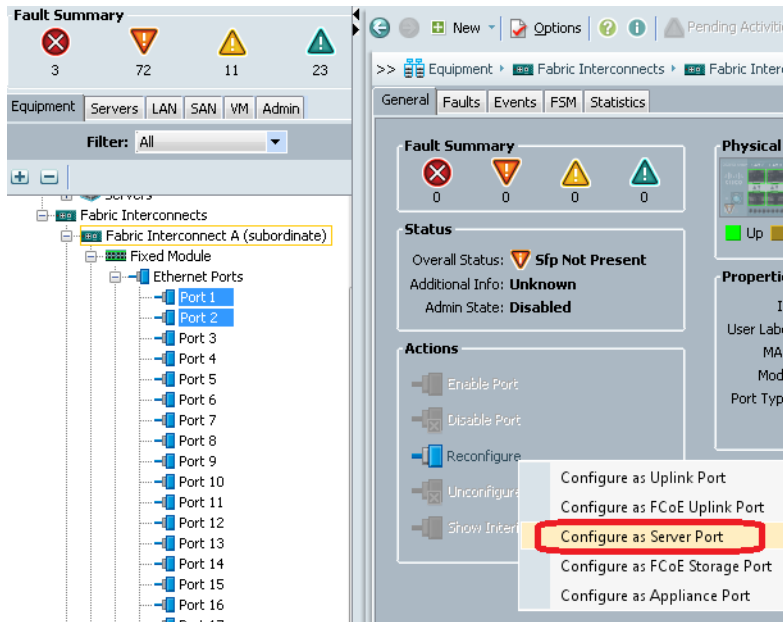
Create Uplink Port Channels to the Cisco Nexus 5548 Switches

These steps provide details for configuring the necessary Port Channels out of the Cisco UCS environment:

1. Select the LAN tab on the left of the window.

✎
Note Two PortChannels are created, one from fabric A to both Cisco Nexus 5548 switches and one from fabric B to both Cisco Nexus 5548 switches.

2. Under LAN Cloud, expand the Fabric A tree.

3. Right-click Port Channels.

4. Select Create Port Channel.
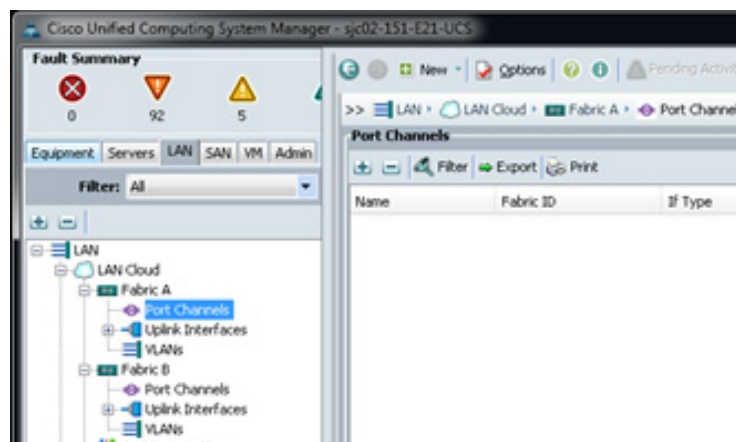
5. Enter <201> as the unique ID of the PortChannel.

6. Enter <vPC-201> as the Name of the PortChannel.

7. Click Next.
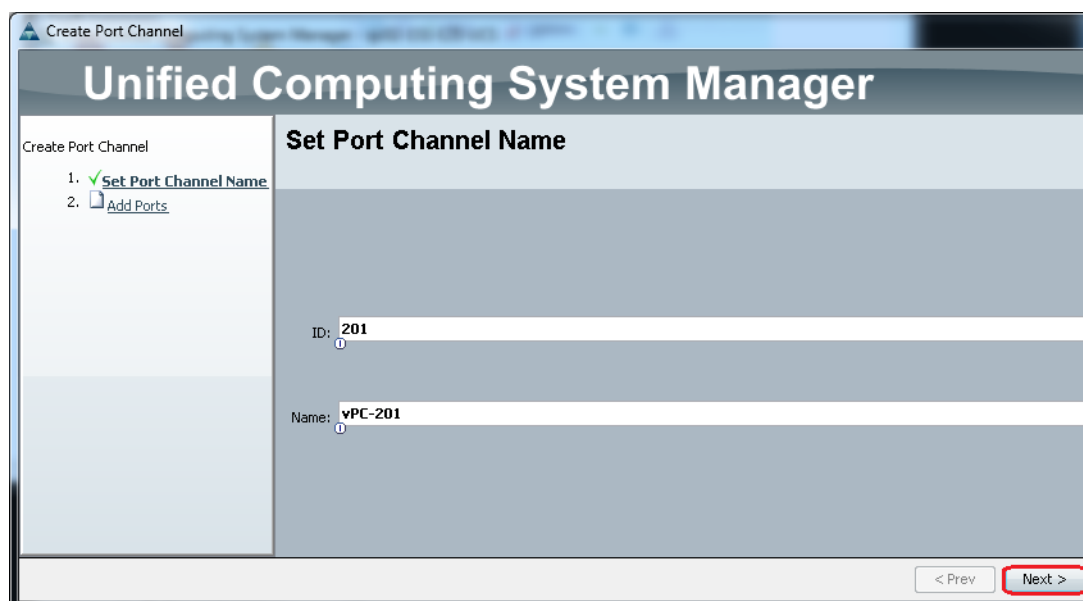
8. Select the port with slot ID 1 and port 17 and also the port with slot ID 1 and port 18 to be added to the PortChannel.

9. Click >> to add the ports to the PortChannel.

10. Click Finish to create the PortChannel.

11. Right-click the newly created port channel and select Show navigator.
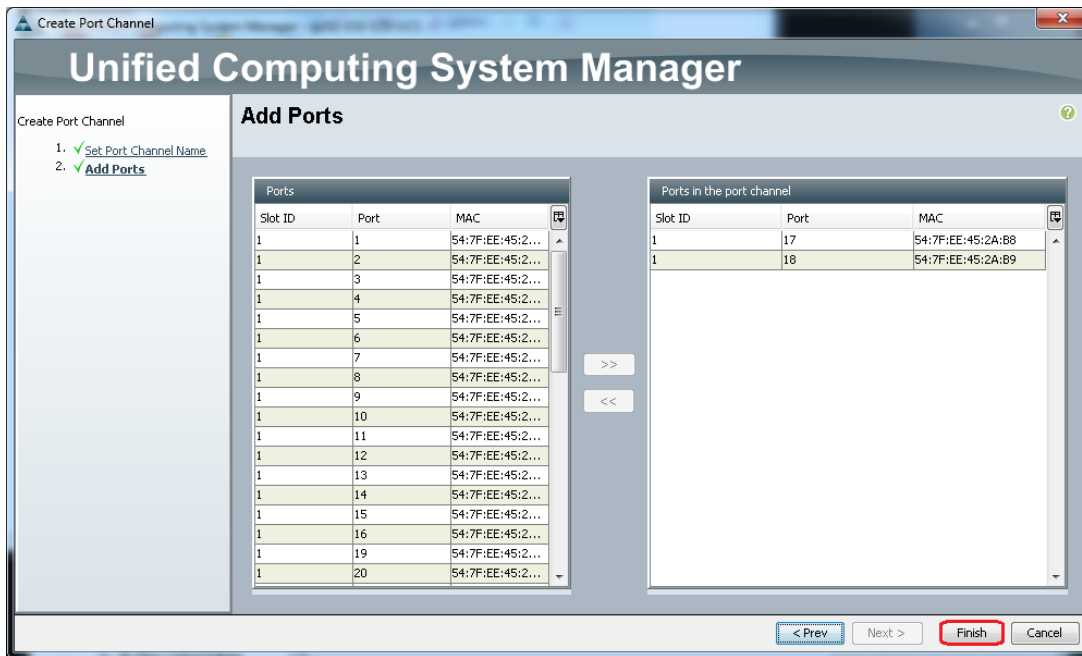
12. Under Actions, select Enable Port Channel.

13. In the pop-up box, click Yes, then OK to enable.

14. Wait until the overall status of the Port Channel is up.

15. Click OK to close the Navigator.

16. Repeat for Fabric B using <202> as the unique ID of the Port Channel and <vpc-202> as the name.
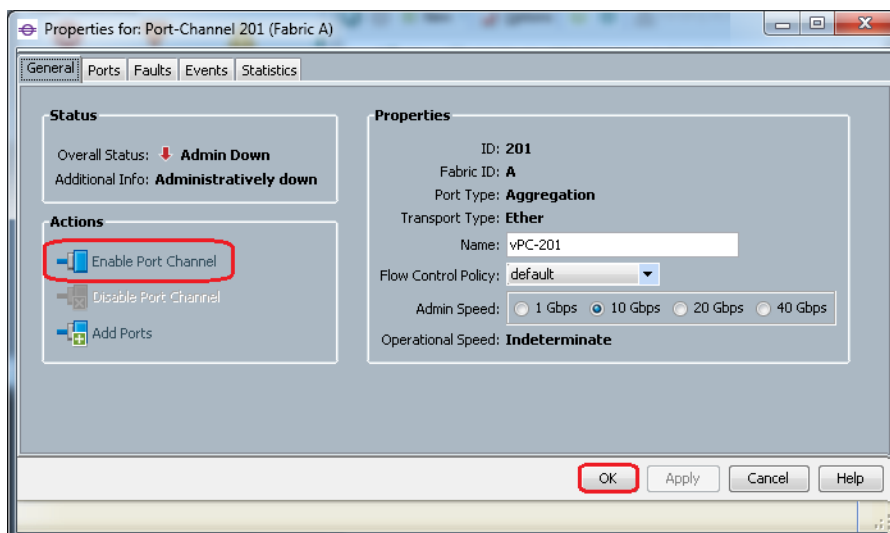
Create vNIC Templates

This environment defines six vNIC templates. Depending upon the needs, more or less can be created. A sample creation is shown. It will need to be repeated for each vNIC template. Table 13 shows the values used for creating the vNIC templates in this CVD. The Native VLAN is needed on vNIC templates that will be accessed by the Hyper-V host. It is not selected for vNIC templates that will be accessed by the virtual machines.

Table 13   Values Used to Create the vNIC Template

Name	Fabric ID	VLAN	Native	MTU	MAC Pool	QoS Policy
Mgmt	A	Mgmt	Yes	1500	VSPEX-99-MAC	
CSV	A	CSV	Yes	9000	VSPEX-99-MAC	
LiveMigration	A	LiveMigration	Yes	9000	VSPEX-99-MAC	LiveMigration
VMaccess	B	VMaccess	No	1500	VSPEX-99-MAC	
ClusComm	B	ClusComm	No	1500	VSPEX-99-MAC	
VEM	B	VEM	No	1500	VSPEX-99-MAC	

To create vNIC templates, do the following:

1. Navigate to the LAN tab.

2. Navigate to LAN > Policies > root > vNIC Templates.

3. Right-click on vNIC Templates and select Create vNIC Template.
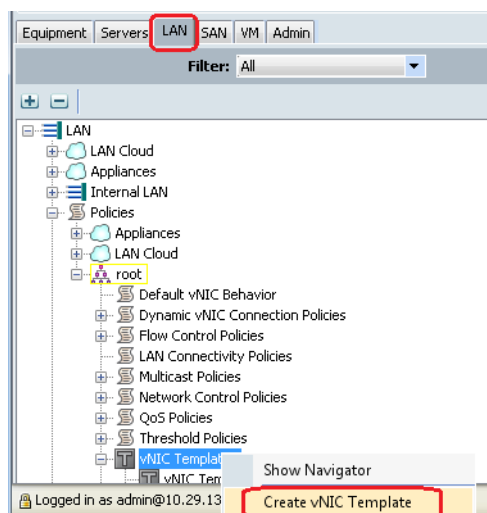
4. Right-click vNIC Templates.

5. Select Create vNIC Template.

6. Enter <LiveMigration> as the vNIC template Name.

7. Check Fabric B.

8. Ensure the Enable Failover box is cleared.

9. Under target, unselect the VM box.

10. Select Updating Template as the Template Type.

11. Under VLANs, select <LiveMigration>. Set Native VLAN.

12. Under MTU, set to 9000.

13. Under MAC Pool, select <VSPEX-99-MAC>.

14. For QoS Policy, select <LiveMigration>.

15. Click OK to complete creating the vNIC template

16. Repeat for each vNIC template.

Create vHBA Templates

To create vHBA templates, do the following:

1. Navigate to the LAN tab.

2. Navigate to SAN > Policies > root > Sub-Organizations > VSPEX > vHBA Templates.

   > **Note** This CVD was created with sub-organizations, but it is not necessary.

3. Right-click on vHBA Templates and select Create vHBA Template.
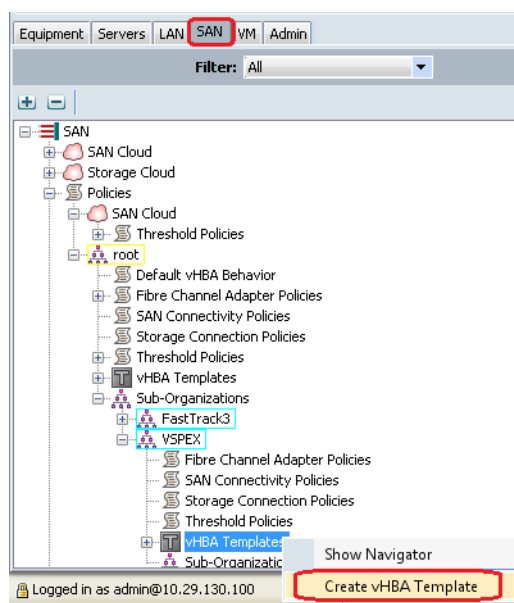
4. Enter a Name.

5. Select A for Fabric ID.

6. Select Updating Template.

7. Select a WWPN Pool.

8. Repeat to create a template for Fabric B.
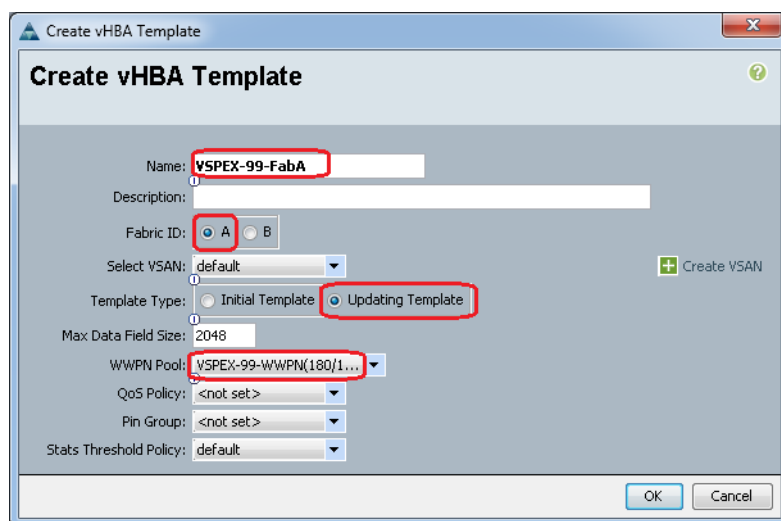
Create Boot Policies

Two boot policies need to be created. The first boot policy will be created to boot from Fabric A and the second to boot from Fabric B. Though not absolutely necessary to have two boot policies, having two options helps spread the load and helps ensure that a total failure does not happen if a disaster happens that removes an entire fabric.

For this sample, the following WWPN values are used for the VNX5400. Your ports may vary depending on the configuration of your VNX5400.

Port	WWPN
SPA A-4	50:06:01:**64**:08:60:06:A1
SPA A-5	50:06:01:**65**:08:60:06:A1
SPB B-4	50:06:01:**6C**:08:60:06:A1
SPB B-5	50:06:01:**6D**:08:60:06:A1

To create boot policies, do the following:

1. Navigate to the Servers tab.

2. Navigate to Servers > Policies > root > Sub-Orgnizations > VSPEX > Boot Policies.

3. Right-click Boot Policies and select Create Boot Policy.
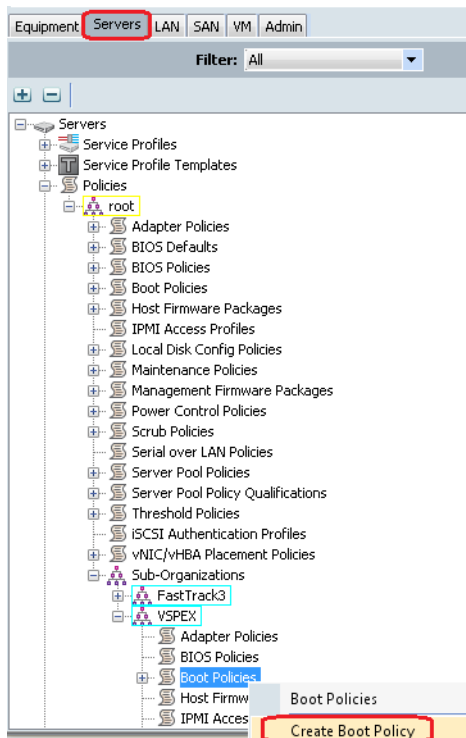
4. Name the boot policy <VSPEX-SAN-A-Boot>.

5. (Optional) Give the boot policy a description.

6. Leave Reboot on Boot Order Change and Enforce vNIC/vHBA Name unchecked.

7. Expand the Local Devices drop-down menu and select Add CD-ROM.

8. Expand the vHBAs drop-down menu and select Add SAN Boot.
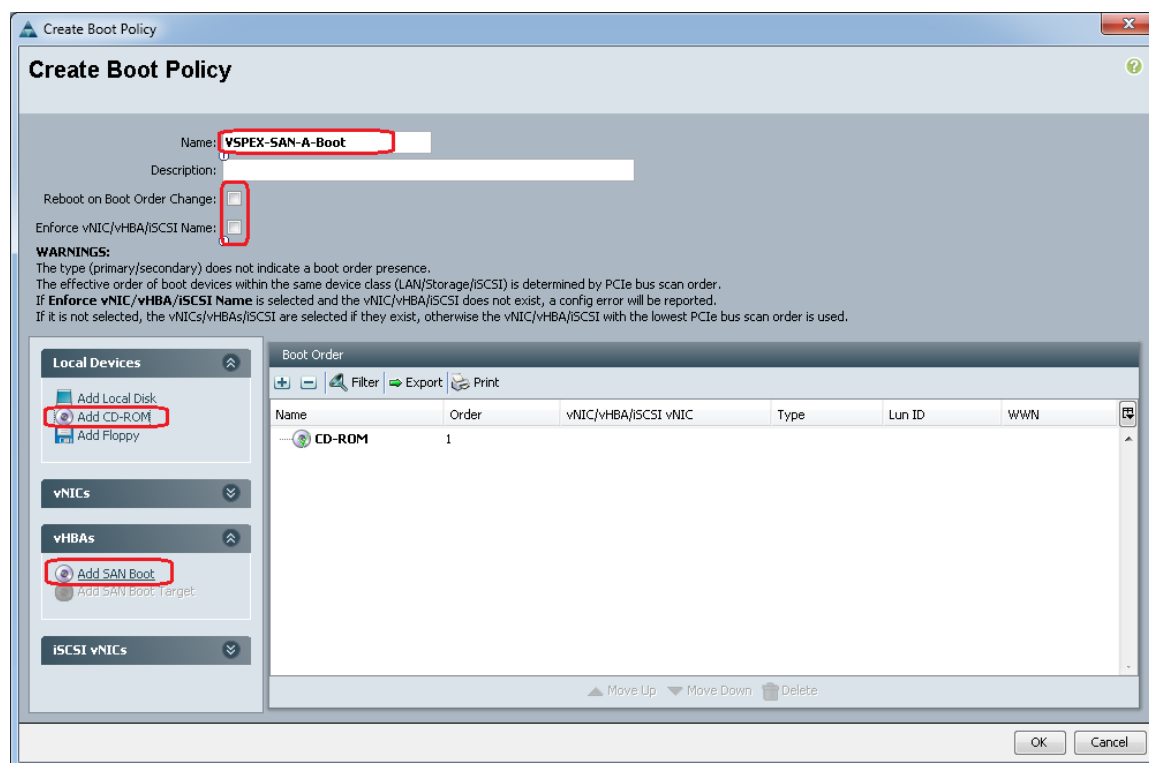
9. Enter <FabChn-A> in the vHBA field in the Add SAN Boot window that displays.

10. Make sure that Primary is selected as the Type.

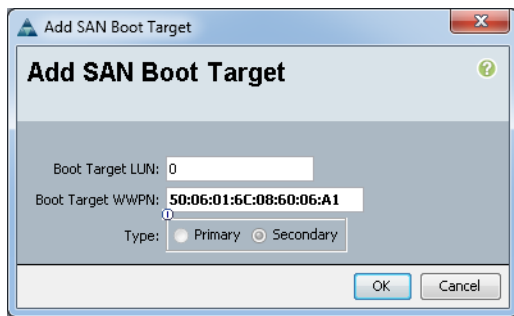11. Click OK to add the SAN boot initiator.
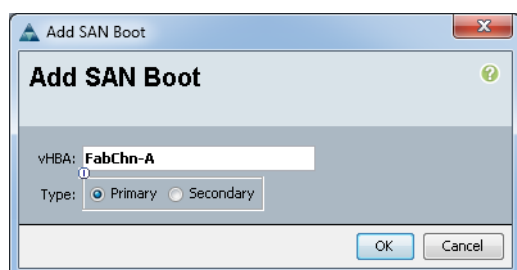
12. Under the vHBA drop-down menu, select Add SAN Boot Target. Keep the value for Boot Target LUN as 0.
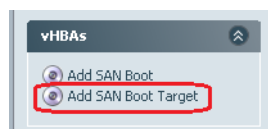
13. Enter the WWPN for the primary FC adapter interface SPA A-5 (remember to use your port number) as the Boot Target WWPN. Keep the Type as Primary.

14. Click OK to add the SAN boot target.
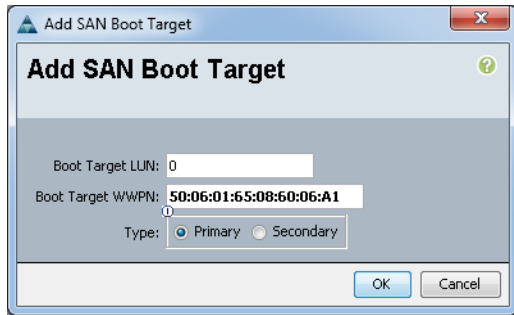
15. Under the vHBA drop-down menu, select Add SAN Boot Target. Keep the value for Boot Target LUN as 0.
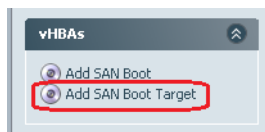
16. Enter the WWPN for the primary FC adapter interface SPB B-5 (remember to use your port number) as the Boot Target WWPN. Select the Type as Secondary; it is the default and cannot be changed on the second entry.

17. Click OK to add the SAN boot target.
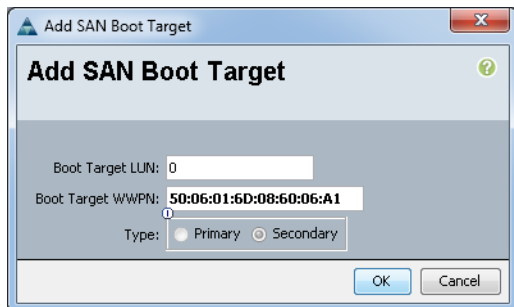
18. Select Add SAN Boot under the vHBA drop-down menu.
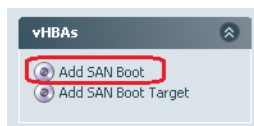
19. Enter <FabChn-B> in the vHBA field in the Add SAN Boot window that displays.

20. The type is automatically set to Secondary and it will be grayed out.

21. Click OK to add the SAN boot target.
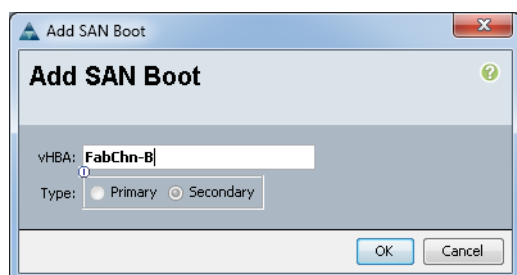
22. Select Add SAN Boot Target under the vHBA drop-down menu.
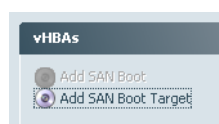
23. The Add SAN Boot Target window displays. Keep the value for Boot Target LUN as 0.

24. Enter the WWPN for the secondary FC adapter interface SPA A-4 as the Boot Target WWPN. Keep the Type as Primary.

25. Click OK to add the SAN boot target.
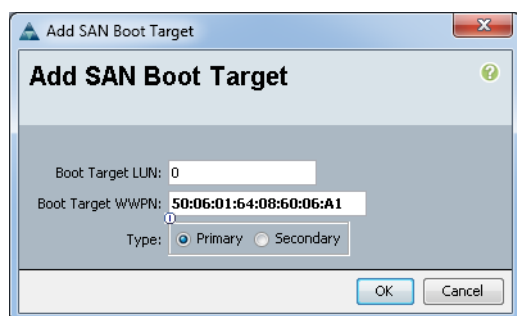
26. Select Add SAN Boot Target under the vHBA drop-down menu.
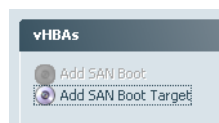
27. Enter the WWPN for the secondary FC adapter interface SPB B-4 as the Boot Target WWPN. Select the Type as Secondary.

28. Click OK to add the SAN boot target.

29. Verify that your configuration looks similar to the screenshot below:

30. Repeat the process to create a boot policy to boot from Fabric B. It should look similar to the screenshot below:

Create Service Profile Templates and Service Profiles

Two Service Profile Templates can now be created. One should be created to use the Fabric A boot policy and the other should be created with the Fabric B boot policy. Otherwise, all the other pool, template, and policy selections should be the same. Use the Create Service Profile (expert) option and select the pools, templates, and policies just created.

When the two Service Profile Templates have been created, create a Service Profile for each server. Assign half the service profiles to the first service profile template and the other half of the service profiles to the other service profile template. Use the Create Service Profiles From Template option.

Create EMC VNX5400 LUNs for VSPEX

The VSPEX cloud environment implements a boot from SAN environment using the concept of a Master Boot LUN. The Master Boot LUN is a storage area that will be used to maintain an image of a Windows Server 2012 R2 image to be used as a Clone source. This image should be configured as a base image to be used for subsequent installations, so all patching and custom configuration steps should be taken. For example, maybe a desired configuration setting is to ensure that all physical servers are able to be remotely managed. When the image is configured according to customer policy, the Microsoft sysprep utility can be run against this image to prepare it for use as a Clone. The steps to configure this Master Boot LUN image are in the following section.

Clones created from the Master Boot LUN will be presented to the physical servers defined by Service Profiles in the UCS environment. This style of deployment allows Service Profiles to be fully transportable between different physical blades as the boot device is external to the chassis, and also allows for multiple Master Boot images to be implemented providing support for different operating system versions or configurations which may need to be implemented over time.

Management of the boot LUN requires special consideration, and needs to ensure that the LUN ID provided to the LUN, as seen from the host is set to 0 (zero). The ESI (EMC Storage Integrator) PowerShell commands do not allow the manipulation of the LUN ID for devices presented to servers, and simply default to the sequential allocation of LUN IDs as implemented by the VNX array. As a result of this behavior, the boot LUN must be the first device that is mapped to the server (Cisco UCS service profile). If this is incorrectly implemented, then the wrong target will be selected for Windows boot operations on server power-up.

As described, the ESI PowerShell commands or Unisphere are utilized for provisioning of the LUNs required within the environment, and assume that the storage pool creation outlined in the previous section has been completed. For the procedure to set up a Master Boot LUN, a single LUN is created, and is used to install a Windows Server 2012 R2 instance. This server instance subsequently will be processed with Windows sysprep and be removed from the server. All compute nodes will then use a clone of the sysprep image to be customized as individual server instances.

Creation of all necessary LUNs within the Private Cloud environment can be executed with the PowerShell script ProcessStorageRequests.ps1 provided Appendix B. The defined XML configuration file is read by the PowerShell script. This XML configuration file contains five parameters. There are two classes that can be repeated multiple times. The XML class <luns> can be repeated multiple times to define multiple LUNs for a server. The <Server> class can be repeated to create multiple server records.

For the purpose of defining and creating the Master Boot LUN, it is recommended to create a unique XML configuration file that defines only this specific device. Later the format of the XML configuration file can be followed for creating multiple LUNs.

- <label> - the name that will be assigned to the LUN that is created
- <pool> - the storage pool from which the LUN will be created
- <size> - the size of the LUN (in GB) to be created
- <ServerName> - the name of the server that will be assigned the LUN that must match the Service Profile name in UCS Manager, including case. This name is also used for management purposes on the VNX array
- <IPAddress> - the management IP address of the server

Here is a sample XML file illustrating the content for creating the Master Boot LUN. It will need to be modified to reflect the customer environment.

```
<StorageParams>
   <Array>VNX5400</Array>
   <UCSAddress>10.29.130.100</UCSAddress>
   <Servers>
     <Server>
       <ServerName>VSPEX-01</ServerName>
       <IPAddress>10.29.130.21</IPAddress>
```

```
          <luns>
             <label>Master_Boot_2012R2</label>
          <pool>VSPEX_Pool_1</pool>
          <size>50GB</size>
          </luns>
       </Server>
     </Servers>
   </StorageParams>
```

In addition to the five parameters listed above that can be repeated, there are two other parameters that are defined only once. The <Array> parameter is the name of the VNX array. The <UCSAddress> parameter is the IP address for accessing the Cisco UCS management console. An example of the contents of a configuration is found in the Sample PowerShell Scripts section in a configuration file called Storage_Luns.xml. This configuration file is used by three different sample scripts.

- PrepMasterBoot-AddViaWWPN.ps1

- ProcessStorageRequests.ps1

- PostClone_AddViaWWPN.ps1

Alternatively to using the provided PowerShell scripts, EMC Unisphere can be used for the purpose of creating LUNs for the boot from SAN deployment.

To create EMC VNX5400 LUNs for VSPEX, do the following:

1. From the Storage > LUNs menu, select Create and create the LUN.

2. Select the RAID Group radio button.

3. Specify a User Capacity of 50 GB (or whatever your standard size is).

4. Provide a meaningful Name for the master boot LUN.
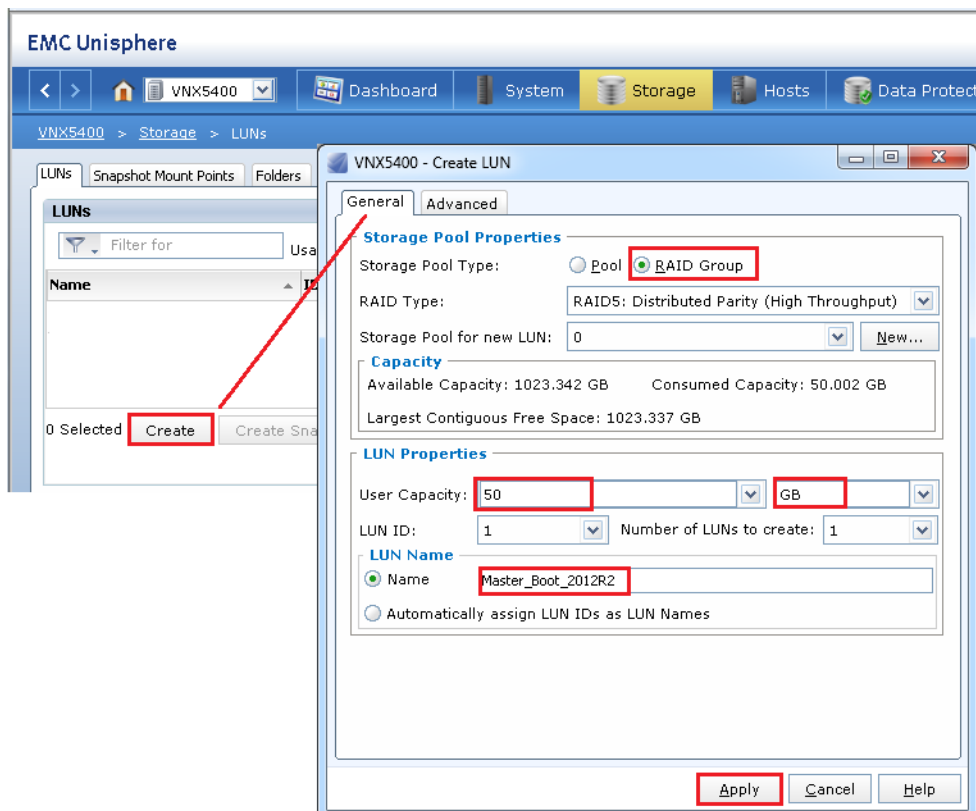
5. Click the Advanced tab.

6. Ensure the Default Owner is SPA by selecting the radio button by SPA.

7. Click Apply to create the LUN.
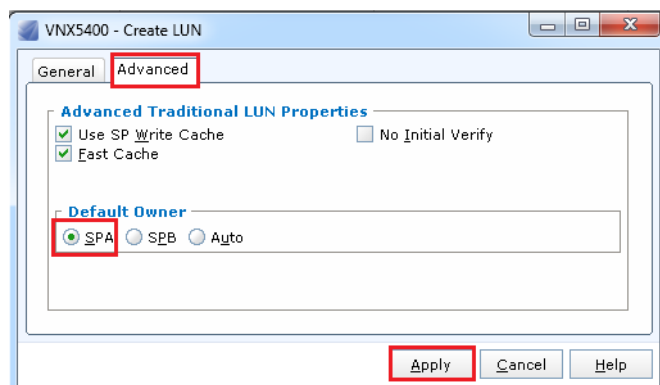
After creation of the required LUN, it is necessary to create a storage group containing the LUN and present the storage group to the Service Profile. The example PowerShell script found in the Sample PowerShell Scripts section, PrepMasterBoot_AddViaWWPN.ps1, utilizes both EMC Storage Integrator and the Cisco UCS PowerTool, and expects that both have been successfully installed. After presentation of the storage group to the WWPNs defined within the Service Profile, it will be possible to proceed with Windows Server installation.

An alternative to using ESI PowerShell would be to manually present storage using Unisphere as in the following steps:

1. Open a browser.

2. Enter the IP address of your EMC VNX5400 SAN with an https:// prefix.

3. Click Continue.
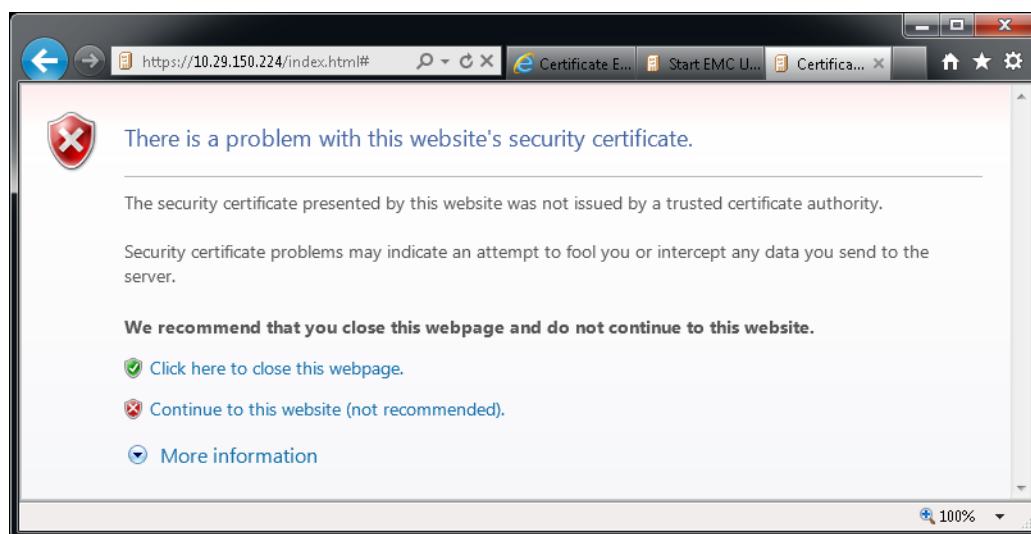
4. Click Accept to accept EMC's licensing agreement.
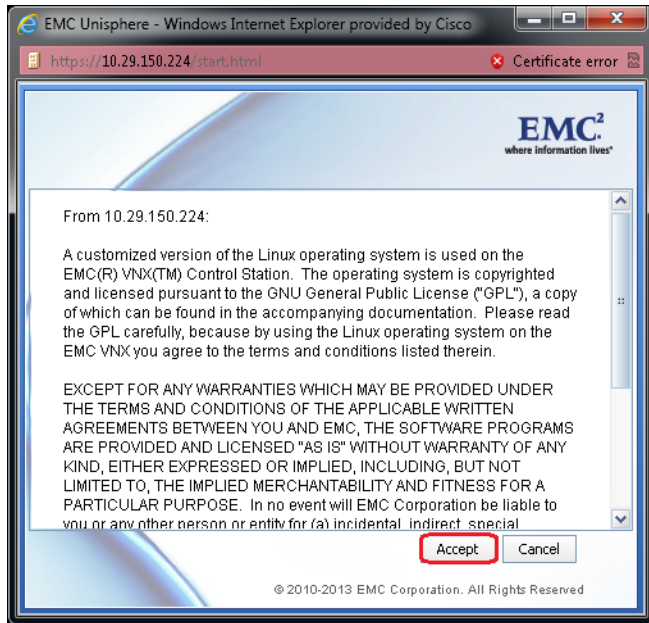
**5.** Enter the Name and Password for your installation.
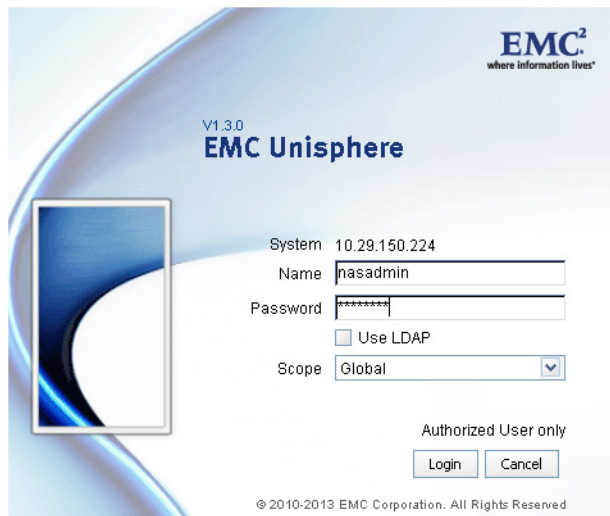
**6.** From the drop-down, select your EMC VNX5400 SAN.
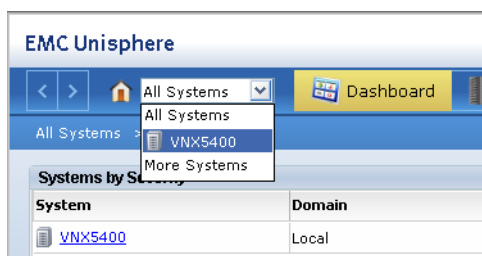
7. Select Initiators from the Hosts tab.

8. Select Create to create a host initiator for accessing the boot LUN.
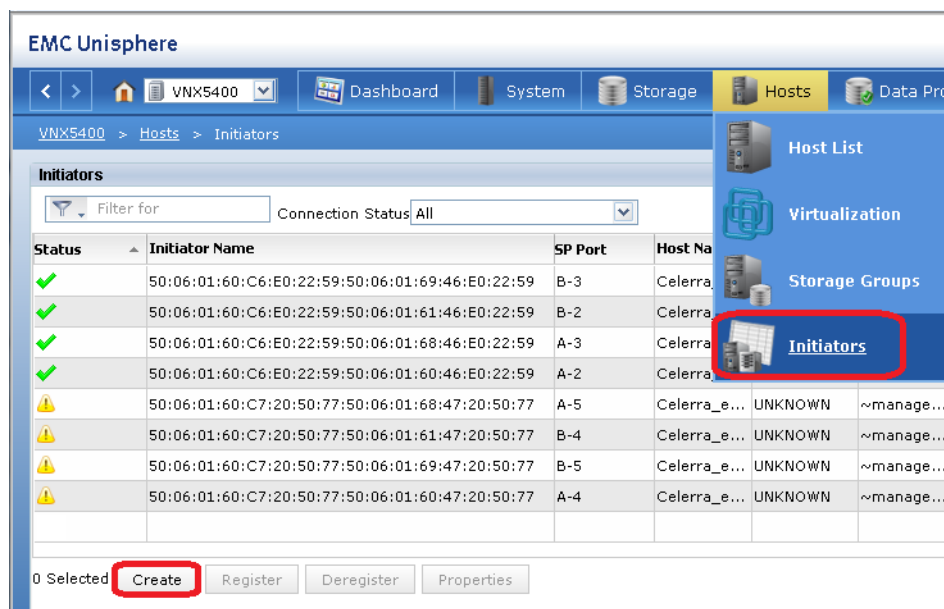

9. Enter the host's WWNN and WWPN in the WWN/IQN field.

10. Select the appropriate port to which this host is connected in the SP-port drop-down list.

11. Select CLARiiON/VNX from the Initiator Type drop-down list.

12. Ensure that Failover Mode is ALUA.

13. Select the radio button for New Host. Enter your Host Name and its IP Address.

14. Click OK.
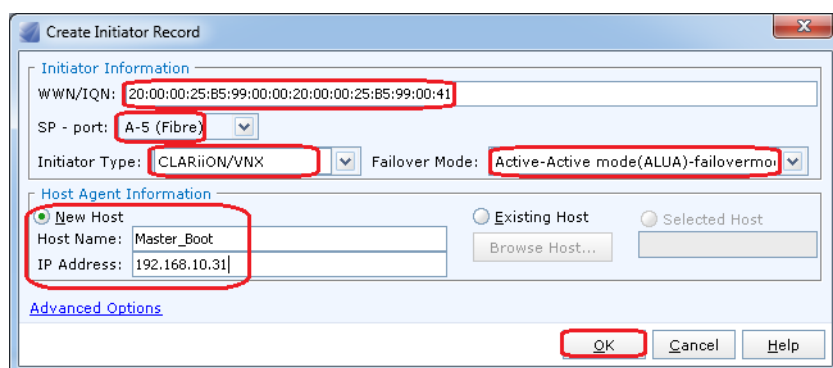
15. Select Storage Groups from the Hosts tab.

16. Click Create.
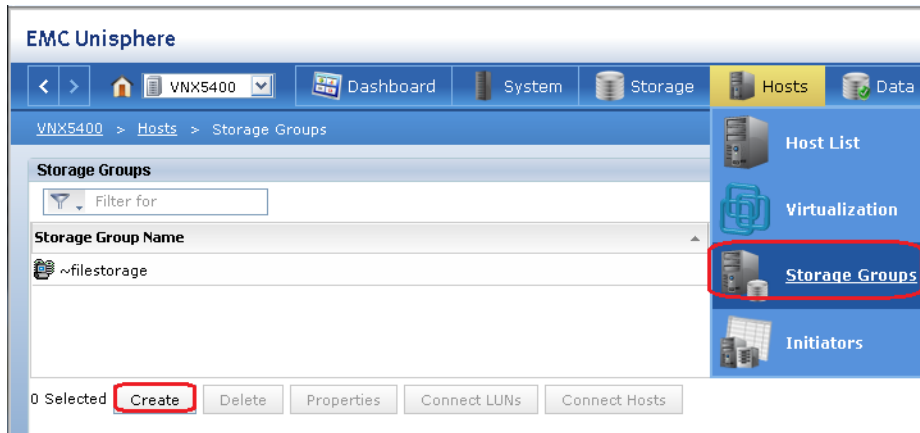
17. Enter a name for a storage group to be assigned to this server in the Storage Group Name field.
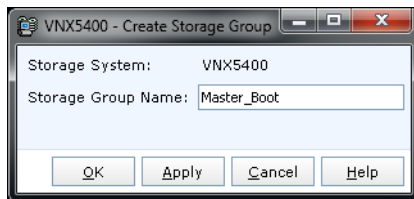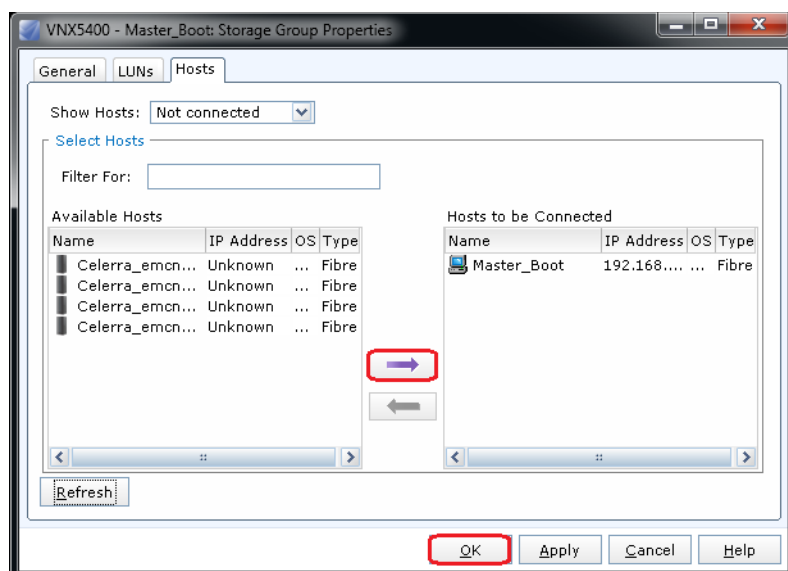

18. On the LUNs tab, select the boot LUN that was created for this server. Click Add and an entry will appear in the Selected LUNs section of the screen.
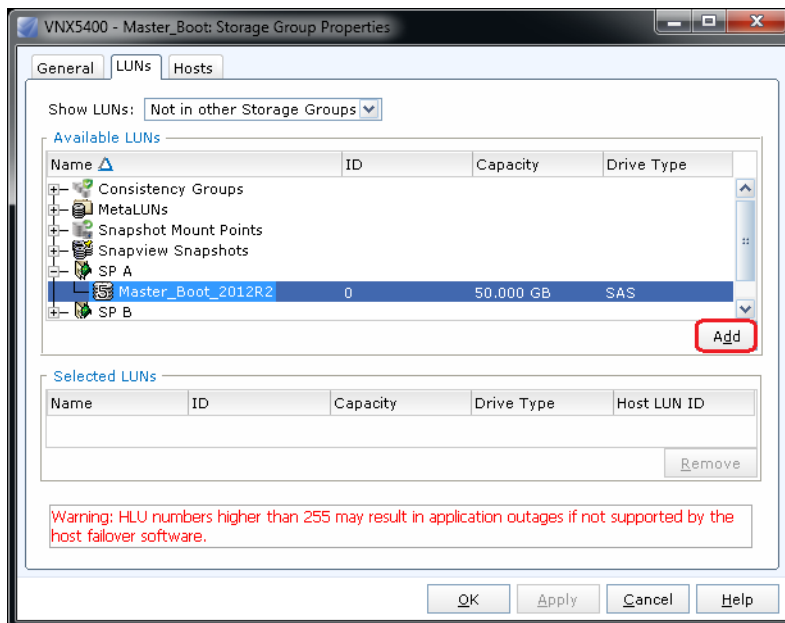
19. Select the Hosts tab.

20. Select the initiator record you created earlier for this server. Click on the right-pointing arrow to move it to the Hosts to be Connected column.

21. Click OK.


Configure Zoning on Cisco Nexus 5548 Switches

The following steps detail the procedure for configuring the Cisco UCS environment to boot the blade servers from the EMC VNX5400 SAN.

Gather Necessary Information

After the Cisco UCS service profiles have been created (earlier section), each infrastructure management blade has a unique configuration. To proceed with the deployment, specific information must be gathered from each Cisco UCS blade to enable SAN booting. Insert the required information in the WWNN/WWPN table for Hyper-V servers in the Customer Configuration Worksheets sections. Both WWNN and WWPN from the Cisco UCS service profiles are needed for masking the LUNs on the VNX5400 SAN.

To gather the information for the Cisco UCS servers, launch the Cisco UCS Manager GUI as follows:

1. Select the Servers tab.

2. Expand Servers > Service Profiles > root.

3. Click each service profile and then click the Storage tab on the right.

4. Record the WWPN for Fabric A and Fabric B for each service profile.

5. Record the WWNN for each service profile.
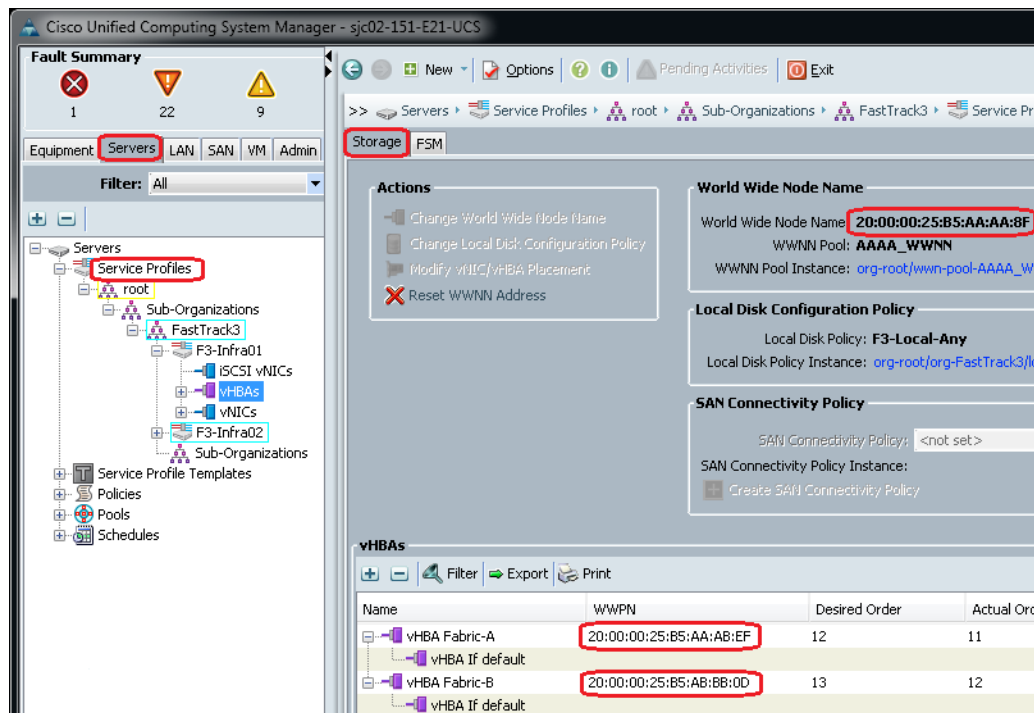
Create Device Aliases and Create Zone for First Server

These steps provide details for configuring device aliases for all devices on both Nexus A and Nexus B. It also creates a zone for the primary boot path for the first server that will be installed and used for creating a 'gold image' or Master Boot image. The initial zoning provides a single path to the SAN. If more than one path is defined to the boot volume, and there is no multipath software available, as is the case for an initial installation of Windows Server 2012 R2, data corruption can occur on the disk. After the operating system is installed and configured for MPIO, the secondary boot path can be defined. This configuration assumes the use of the default VSAN 1.

Cisco Nexus 5548 A

```
device-alias database
device-alias name <VSPEX-01-A> pwwn <VSPEX-01 Fabric-A WWPN>
device-alias name <VNX5400-SPA-A0> pwwn <SPA-A0 WWPN>
device-alias name <VNX5400-SPB-B0> pwwn <SPB-B0 WWPN>
device-alias commit
zone name <VSPEX-01> vsan 1
member device-alias <VSPEX-01-A>
member device-alias <VNX5400-SPA-A0>
exit
zoneset name <VSPEX> vsan 1
member <VSPEX-01>
exit.
zoneset activate name <VSPEX> vsan 1
The Nexus should respond with "Zoneset activation initiated.  Check zone status."
copy run start
```

Cisco Nexus 5548 B

Create the device-alias database on Nexus B at this time. Later in the process the zones and zoneset for Nexus B will be created will be created.

```
device-alias database
device-alias name <VSPEX-01-B> pwwn <VSPEX-01 Fabric-B WWPN>
device-alias name <VNX5400-SPA-A1> pwwn <SPA-A1 WWPN>
device-alias name <VNX5400-SPB-B1> pwwn <SPB-B1 WWPN>
device-alias commit
zoneset name <VSPEX> vsan 1
exit
copy run start
```

Install Initial Microsoft Windows Server 2012 R2

The following steps provide the details necessary to prepare the host for the installation of Windows Server 2012 R2 Datacenter Edition. It assumes that the SAN has been zoned and the EMC V NX5400 has masked the LUN so only a single path to the server is available.

To speed the process of installing Windows Server 2012 R2 across all the physical hosts, a multiple step process is employed:

- Install Windows Server 2012 R2 on a single physical server with the boot volume on the EMC VNX5400.
- Perform some initial configuration tasks that are common for all servers used.
- Update the installation with the latest patches from Microsoft Update.
- Present the boot LUN to both vHBAs and configure MPIO.
- Sysprep the image.
- Remove the boot volume from the server on which it was installed.
- Make clones of the sysprepped volume within the EMC VNX5400 so each physical server will have its own clone to boot from.
- Configure zoning and masking for other servers.
- Start each host and complete the mini-setup to tailor each node with things like name, IP addressing (if fixed IP addresses are used), and join to the domain. (It is possible to configure this sort of information with unattend command files. That is beyond the scope of this document, and many shops already have such procedures in place.)

> ✎
> **Note**   In order for the Windows Installer to recognize the Fibre Channel SAN boot disk for the initial server, the Cisco UCS fnic (storage) driver must be loaded into the Windows installer during installation. Please download the latest Cisco Unified Computing System (UCS) drivers from www.cisco.com under Cisco UCS B-Series Blade Server Software and place the ISO on the same machine with the Windows Server 2012 DVD ISO.
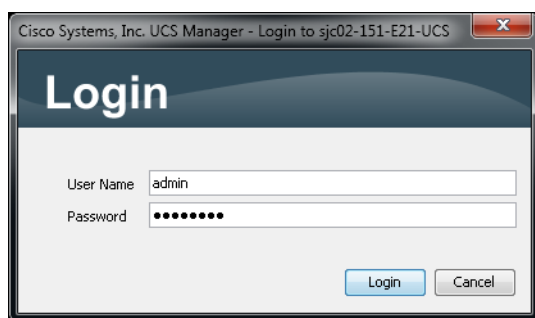
1. Open a browser.
2. Enter the IP address of your fabric interconnect cluster with an https:// prefix.
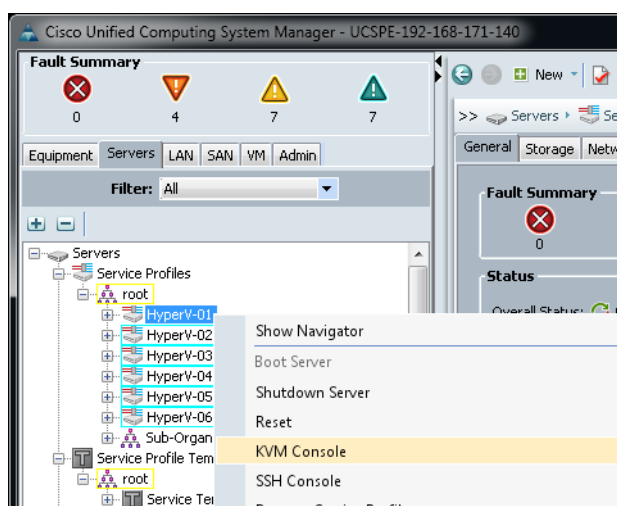3. Click Continue.
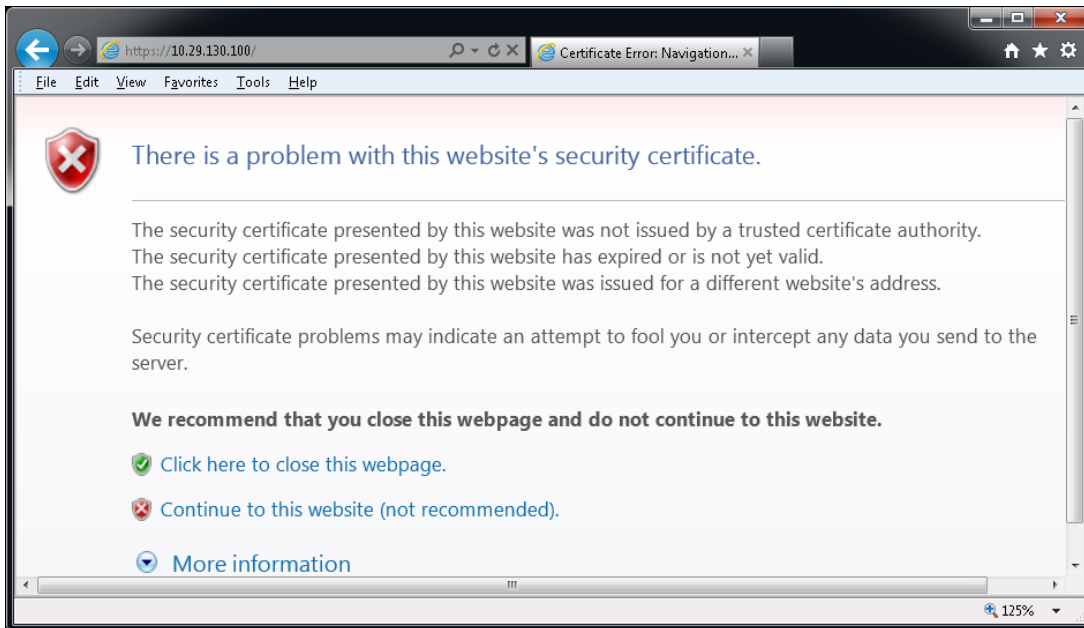
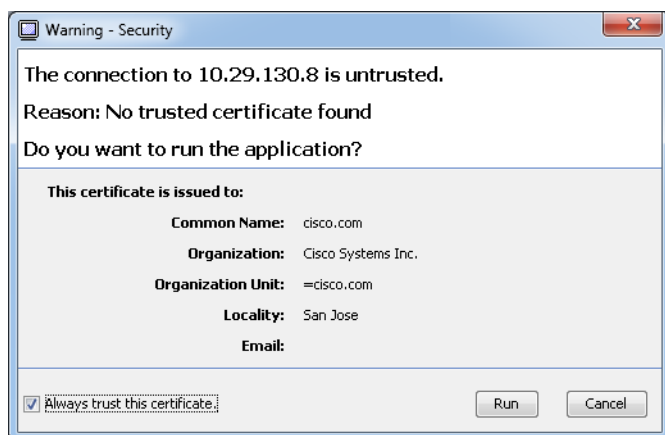**4.** Click Launch UCS Manager.
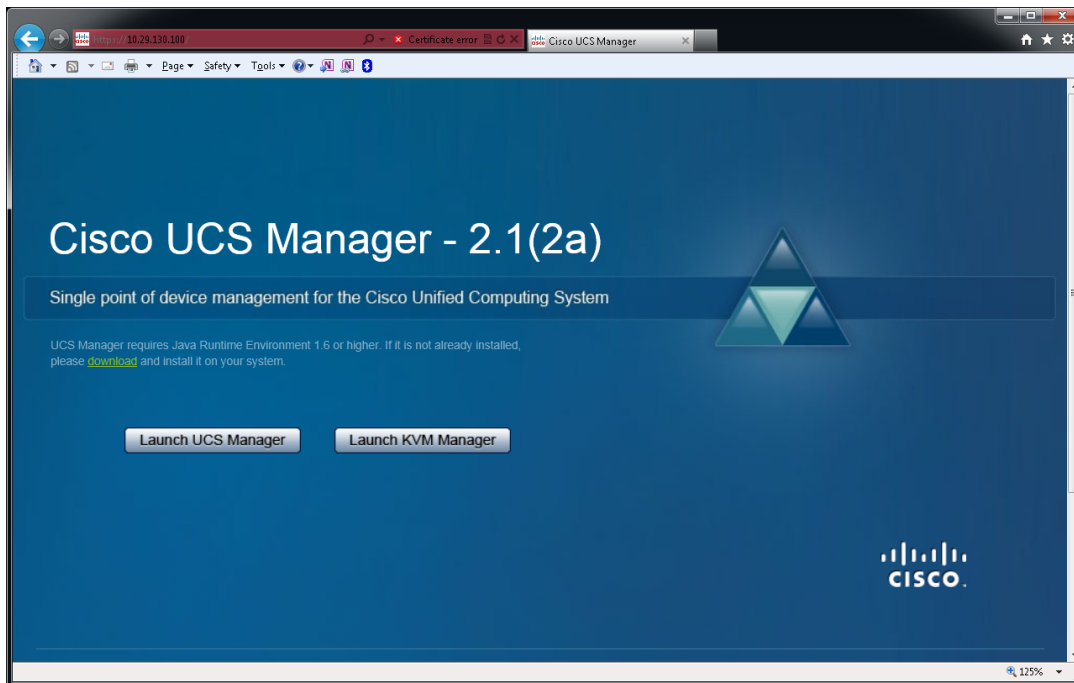
5. Enter admin as the user name.

**6.** Enter the password specified in the initial setup.

7. Select the Servers tab.

8. Navigate the tree Servers > Service Profiles > root > F3-Infra01.

9. Right-click HyperV-01 and select KVM Console.

10. A warning appears regarding certificates. Click the Always trust this certificate check box.

11. Click Run.

12. Another window will display warning of an unencrypted KVM session. Select the radio button for Accept this session. Optionally, select the check box by Remember this configuration for future connections to the server.

13. Click Apply.
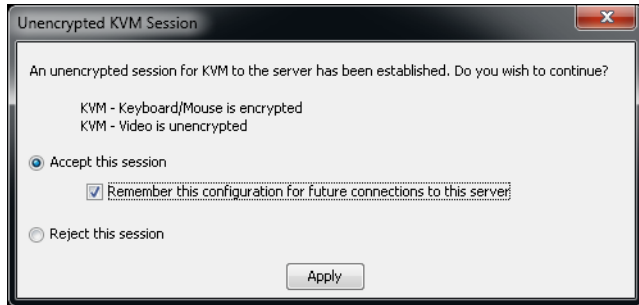
14. Click on the Virtual Media tab of the KVM console. (You will receive another warning window like above).

15. Click the Add Image... button on the right.
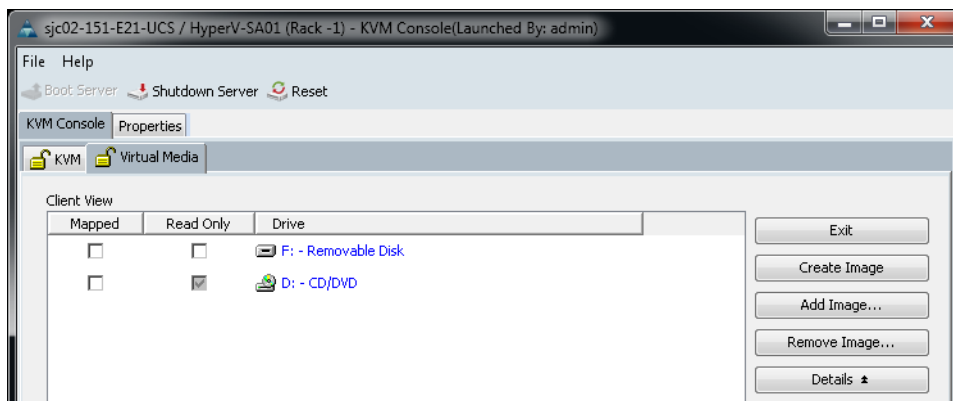

16. Browse to the location on your configuration workstation where you have stored a copy of the Windows Server 2012 R2 installation media.

17. Click Open.
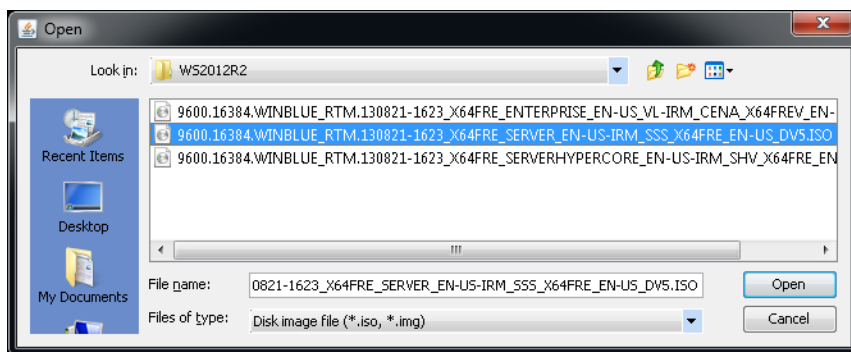
18. Click the Mapped box in the Virtual Media window.

19. Repeat the process to load an .img or .iso file containing the 1240 VIC drivers, except do not click the Mapped box.

20. Click the KVM tab to return to the KVM window.

21. Click Reset to cause the server to boot to the installation media.

The installation will start.
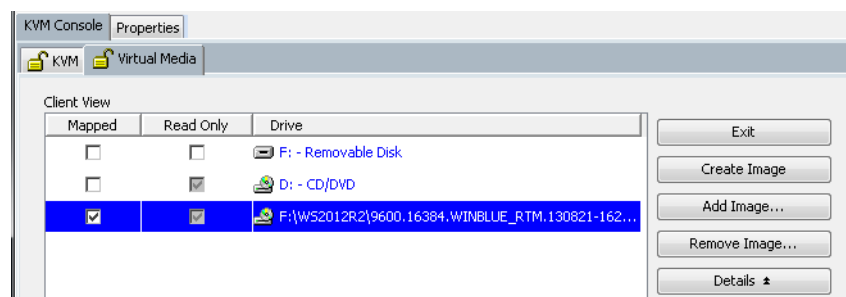
22. Select the appropriate localization features.

23. Click Next.

24. On next screen, click Install Now.
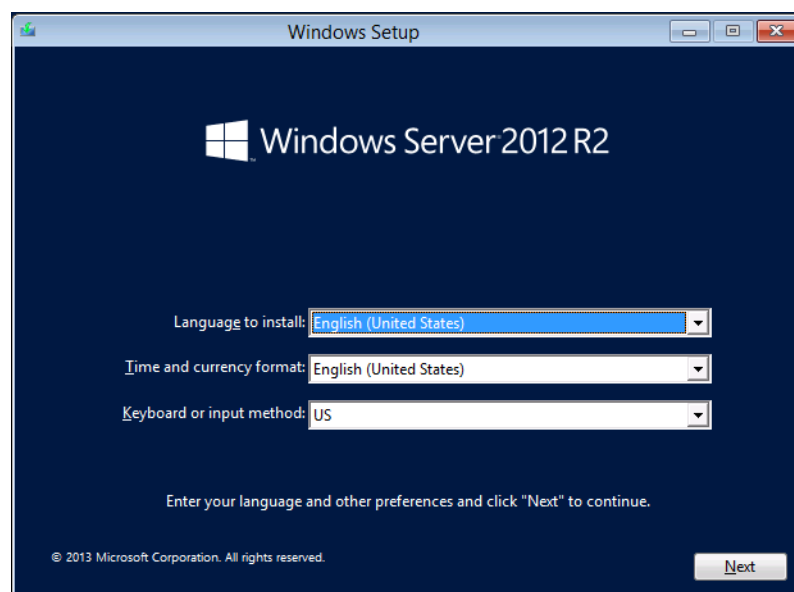
Depending upon the distribution you are using, you may or may not see this window. If you are using a Retail copy, you will see this window. If you are using a volume license copy, you will not see this window.

25. If you are using a Retail copy, enter the 25-character key that came with your software.
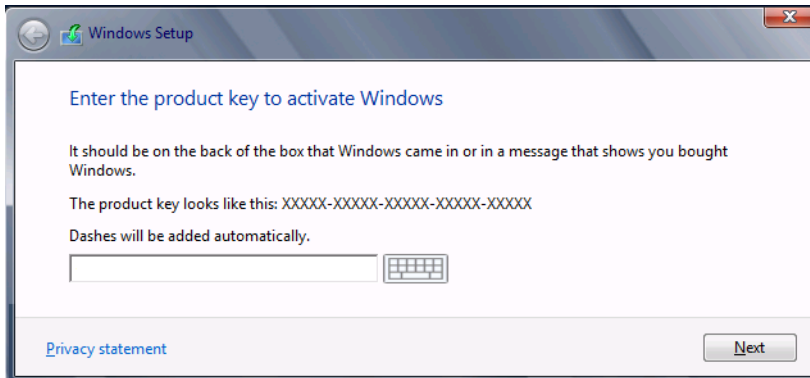
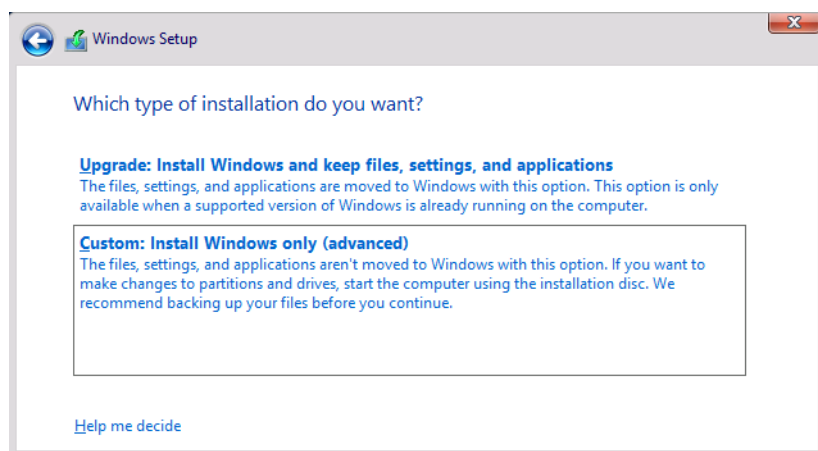26. Select the Windows Server 2012 R2 Datacenter (Server with a GUI) option.
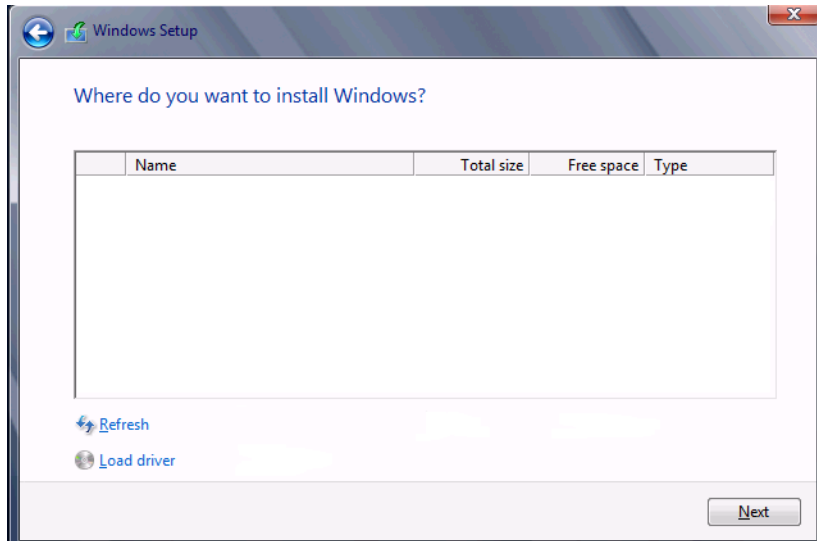
27. Click Next.
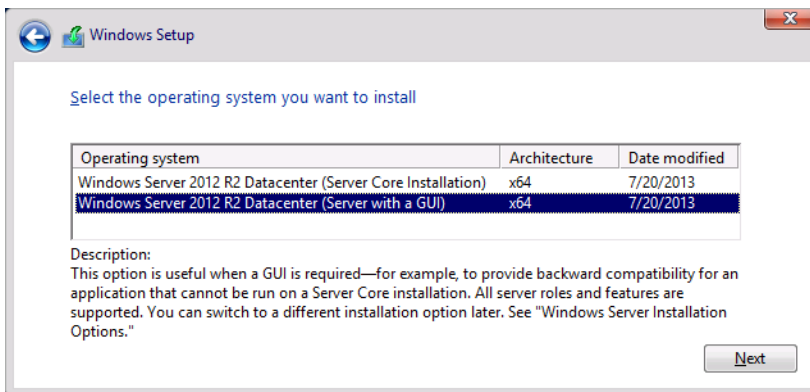
28. Click the check box to accept the license terms.

29. Click Next.

30. Click on Custom: Install Windows only (advanced).

31. You will not see any disks because the 1240 drivers are not included as part of the Windows Server 2012 R2 installation media. You will have to manually load them.

32. Click Load driver.

33. Click on the Virtual Media tab of the KVM, uncheck the box for the Windows media and check the box for the driver media. You will receive a warning about disconnecting in this manner instead of gracefully dismounting in the operating system. Dismount anyway.

34. Switch back to the KVM tab.

35. Click the Browse button to browse to the virtual media containing your Cisco UCS 1240 drivers and install the storage driver for the 1240 fnic.
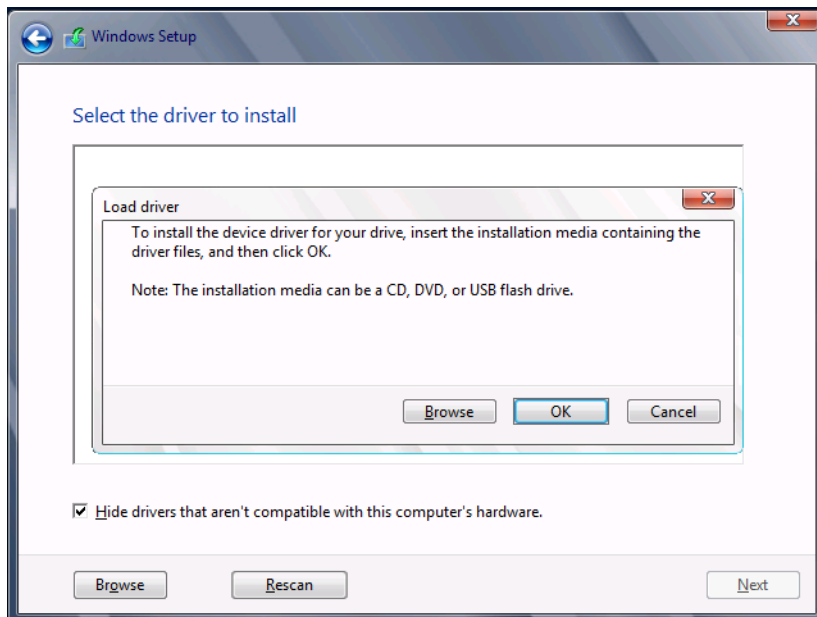
36. Click Next to install the driver.

37. Repeat these steps for loading the enic drivers. If you do not load the NIC drivers at this time, you will need to do it after the system has been installed.
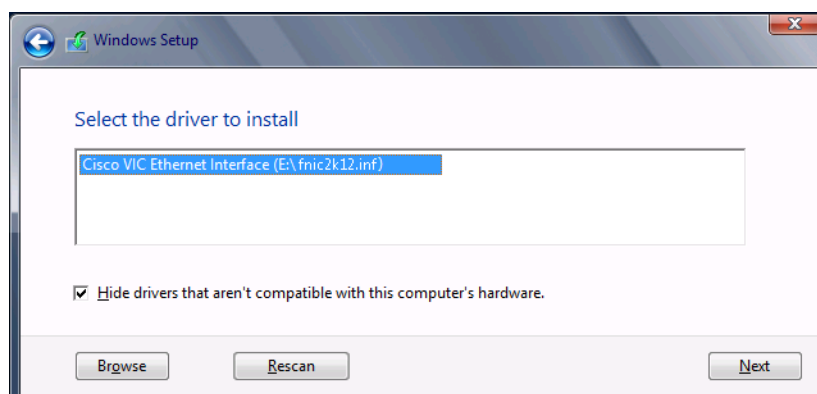
38. When the driver installation is complete, you will be returned to this window. You may have to click Refresh to get the storage to show.

39. Return to the Virtual Media tab and swap the media back to the Windows distribution.

40. Ignore the size warning at the bottom of the window.

41. Click Next.
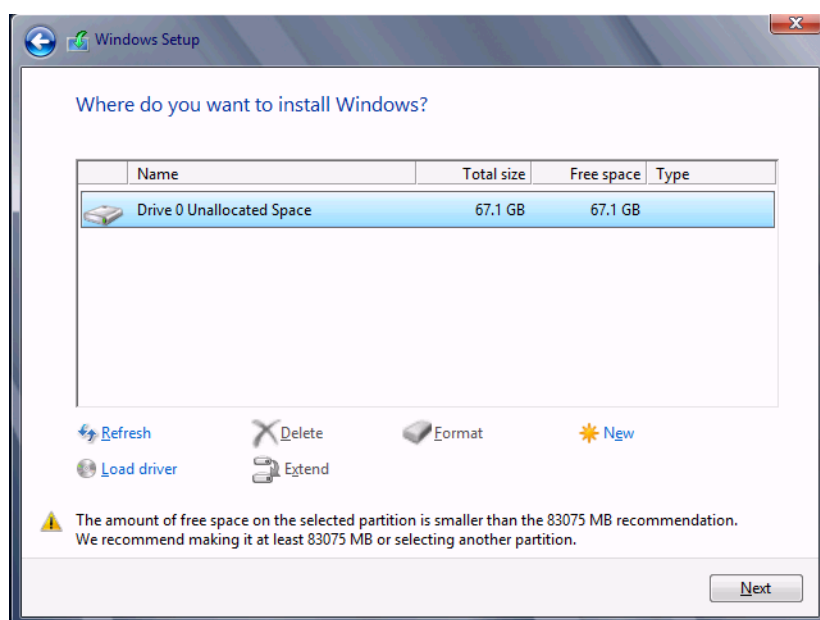
42. Windows will now proceed through its initial setup.

43. As noted, Windows will reboot during this process. You may see a message to Press any key to boot from CD or DVD. Do not enter any key as it will start the installation process from the beginning again. (You can ensure this message does not appear by removing the Windows Server 2012 R2 virtual media.)
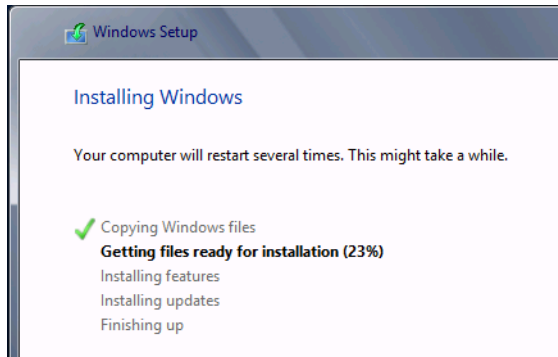
44. Enter password for local administrator account.

45. Re-enter password to validate.
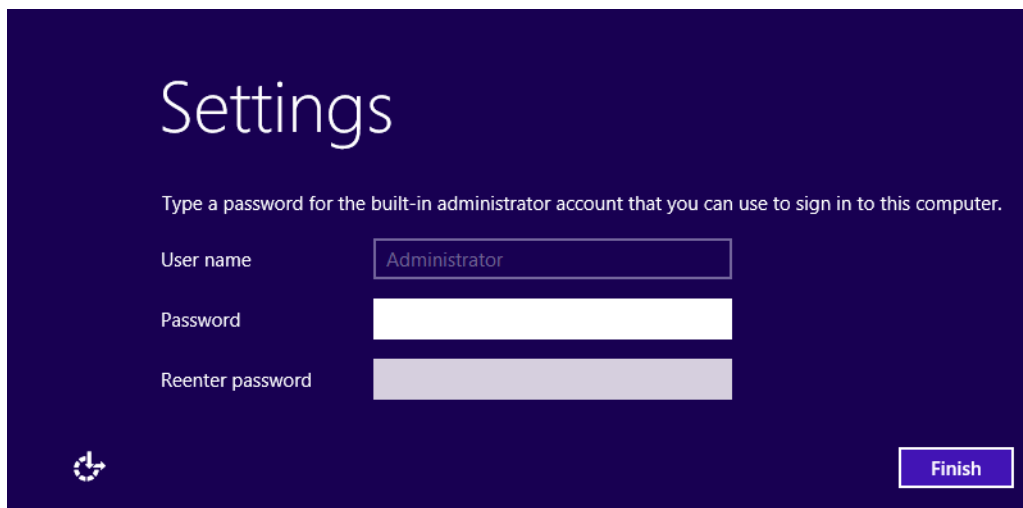
46. Login to the new machine.
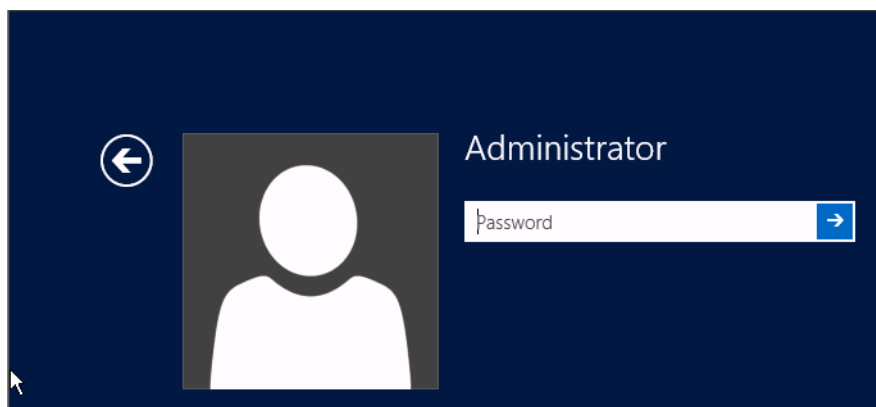
Sysprep the Initial Image

Before you run the sysprep utility against this newly created image, it is a good idea to tailor the image to specific customer needs. This includes adding and/or configuring software that will be common for all systems built from this image. In addition to the tasks listed here, the customer may have their own management software to install. Remember that not all software can be installed before an image is sysprepped, so check with the software vendor before installing.

At this point, if you have a DHCP server installed on your Management Network, the Management Network Interface should come up with an IP address. If you do not have DHCP, use the following steps to determine which Network Interface is on the Management VLAN and configure it with a static IP with connection to the outside world.

Initial Network Configuration

The following sample screen shots may vary significantly from the actual customer environment. This is due to the fact that there are many variables in the potential customer network, and all variations are not covered in these samples. These samples assume that there is no DHCP server (which would make this a little easier, but is beyond the scope of this document). By assuming there is no DHCP server, all NICs will initially be configured with 169.254/16 APIPA addresses. These steps will assign fixed IP addresses to all the NICs.

The first step that is necessary is to find the NIC through which host management is performed. This is not the out-of-band NIC used by UCSM, but the NIC dedicated to host management.

1. Log into the server.

2. Enter the following PowerShell command:

   ```
   Gwmi Win32_NetworkAdapter | Where {$_.MACaddress -ne $Null} | FT NetConnectionID, MACaddress
   ```
   This returns a table of the network names and their associated MAC addresses.

3. Go to the Servers tab in UCSM.

4. Select Servers > Service Profiles > root and the service profile for the machine you are working on.

5. Expand the Service Profile.

6. Click on vNICs.

7. This enables you to see the MAC addresses for the Mgmt vNIC (in this example, Mgmt is the NIC used for host management).

8. Find the MAC address in the table displayed in the previous step, and take note of the assigned name. For example purposes, assume it is "Ethernet"
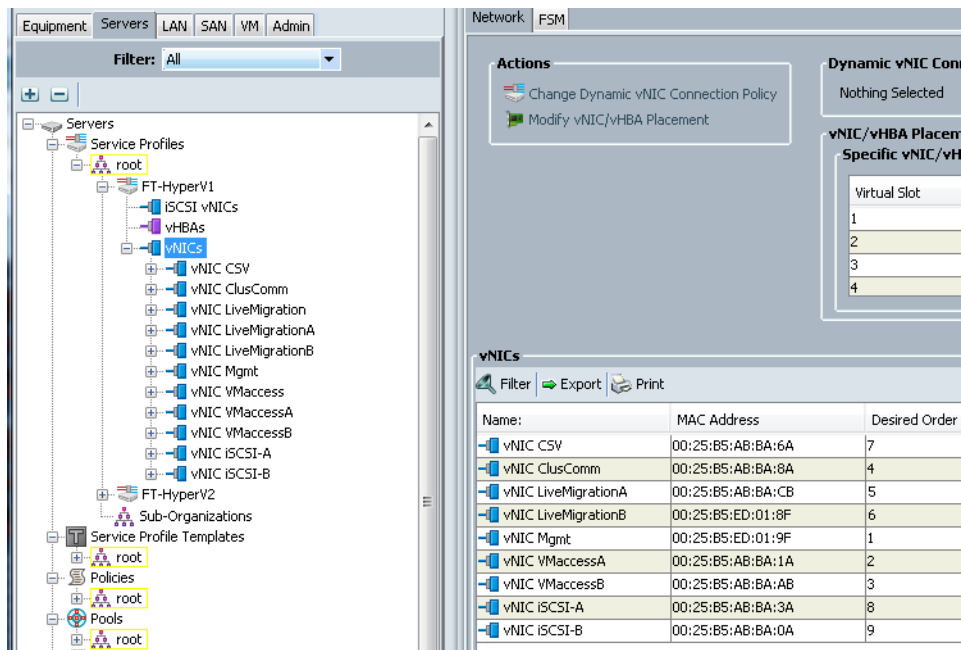
9. In Server Manager click on Local Server.
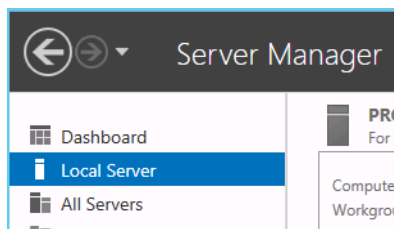
10. Click on any one of the networks. This will bring up the Network Manager window.
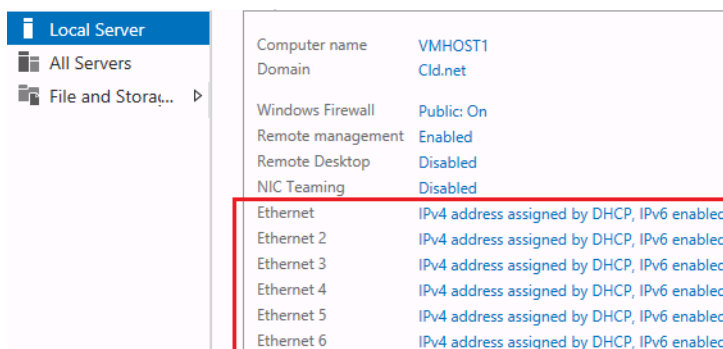
11. Double-click the entry for "Ethernet". This brings up the Status window for the Ethernet NIC.

12. Click Details... to ensure you have the right MAC address.

13. Click Properties.
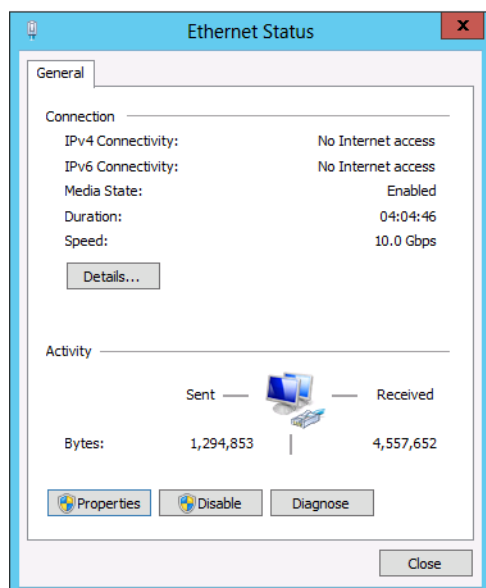
14. Click on the Internet Protocol Version 4 (TCP/IPv4) line. (Leave the check box checked.)

15. Click Properties.
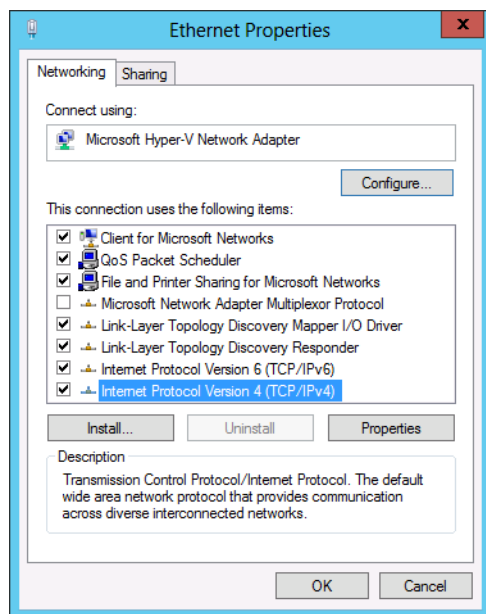
16. Configure the IP settings appropriately for the customer environment.

17. Click OK.

18. Click Close.

19. Click Close.

20. Back in the Windows PowerShell window, ping the Domain Controller by its name to help ensure you have properly configured the network settings.
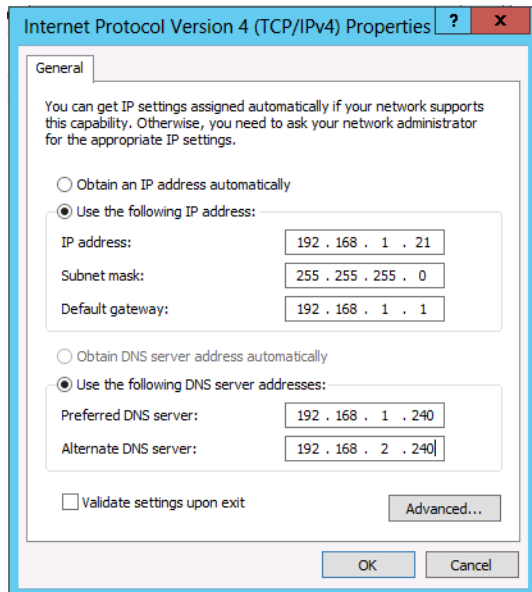
Common Configuration Tasks

There are some tasks that are performed to ensure the ability for the hosts to be remotely managed for the rest of these instructions. In an existing customer environment, the customer may handle some of these tasks via Active Directory group policy objects. Setting up these tasks to be handled by group policies is beyond the scope of this document, so they should be reviewed with the customer. The Sample PowerShell Scripts section contains a sample PowerShell script, Set-UcsHyperVRemoteMgmt.ps1, that sets a number of firewall rules to enable remote management, enables some services to automatically start, and enable remote desktop. Run this script from a PowerShell command window.

.NET Framework 3.5 Feature (optional)

Depending upon the management tools that are deployed in the customer environment, it might make sense to load the .NET Framework 3.5 feature. Doing it once on this image that will be sysprepped will save time in later deployments. This is not necessary in every data center.

Ensure the KVM has the Windows Server 2012 R2 installation media mounted. Assuming the Windows Server installation media is mounted on drive E:, issue the following PowerShell command to add the .NET Framework 3.5 feature.

```
Install-WindowsFeature -Name NET-Framework-Core -Source E:\sources\sxs
```

Run Windows Update

It is highly recommended to fully patch the server at this time from Windows Update. Depending on the patches, it might be necessary to reboot and check for updates multiple times before the server is completely patched.

Install Windows Roles and Features

The Sample PowerShell Scripts section contains a sample PowerShell script, Add-UcsHyperVFeatures.ps1, which installs the MPIO and Failover Cluster features and the Hyper-V role. Run this script from a PowerShell command window. Installation of the Hyper-V role causes a reboot.

Configure Paging File

By default, Windows allocates and manages a portion of the system disk to be used as a paging file based on the amount of physical memory on a server. Since the workload running on Hyper-V servers really runs in the VMs, the majority of paging occurs within the VMs, minimizing the need for a large page file on the physical server. Therefore, it makes sense to minimize the size of the paging file of the Hyper-V host to minimize the amount of storage on the boot volume that is reserved for the paging file.

1. In Server Manager, click on the Computer Name to bring up the System Properties window.

2. Click on the Advanced tab.
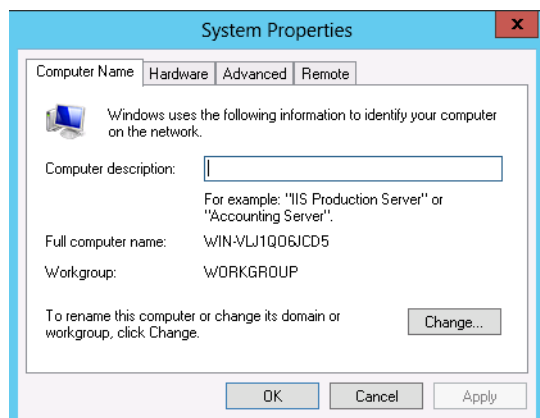
3. Click the Settings... button in the Performance section of the window.
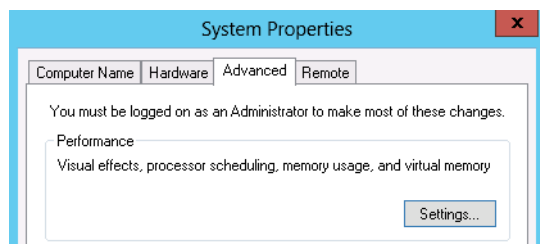
4. Click the Advanced tab.

5. Click on the Change... button.
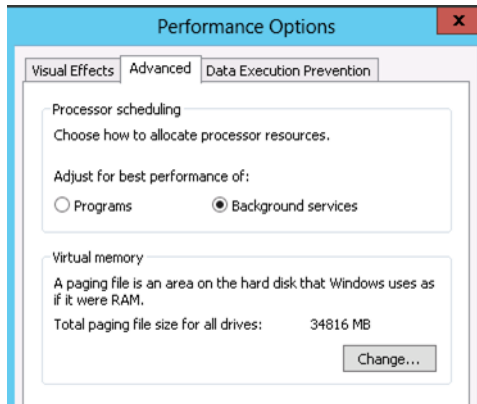
6. Uncheck the Automatically manage paging file size for all drives box.

7. Click the Custom size: radio button.

8. Enter 2048 into the Initial size (MB): field.

9. Enter 4096 into the Maximum size (MB): field.

10. Click the Set button to set the new values.

11. Click OK button four times to accept the change. System must be rebooted to implement the change.

> **Note** The server will be powered down in the next step, so there is no need to reboot it at this point.

> **Note** Though the values shown in this example should be sufficient for most environments, the customer might have their own policies in regards to sizing the paging file.
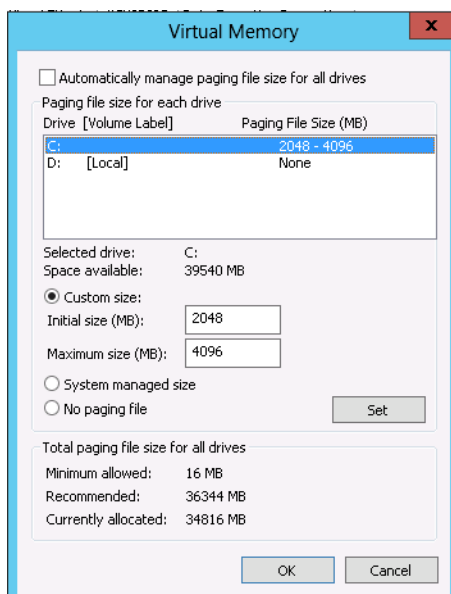
Configure MPIO

After the server has been configured with the MPIO feature, it is necessary to present the additional paths to the boot LUN and configure MPIO. The goal is to sysprep this operating system image and then clone the LUN for use by all other physical servers, this means MPIO has to be configured only once. Then, since the operating system image that will be used for booting the additional blades will already have MPIO configured, it is possible to configure paths through both Nexus switches for initial boot of the sysprepped image.

The Add-UcsHyperVFeatures.ps1 PowerShell script performed the installation and some configuration of Microsoft's MPIO. From an elevated command prompt issue the command mpclaim -s -d 0 to validate MPIO is configured on your system.

```
PS C:\Users\Administrator> mpclaim -s -d 0

MPIO Disk0: 01 Paths, Round Robin with Subset, Implicit and Explicit
    Controlling DSM: Microsoft DSM
    SN: 6061606D703204931EFCC9011E311
    Supported Load Balance Policies: FOO RRWS LQD WP LB

    Path ID            State            SCSI Address      Weight
    ------------------------------------------------------------
    0000000077010000   Active/Optimized  001|000|000|000   0
    * TPG_State : Active/Optimized   , TPG_Id: 1, : 6


PS C:\Users\Administrator>
```

Next, prepare the Cisco Nexus 5548 switches with zones that reflect all paths to the boot LUN.

Cisco Nexus 5548 A

Remember that we had previously configured only a single path on Cisco Nexus 5548 A for the initial installation. Issue the following commands to create the secondary path.

```
zone name <VSPEX-01> vsan 1
member device-alias <VNX5400-SPB-B0>
exit
zoneset activate name <VSPEX> vsan 1
```
The Nexus should respond with "Zoneset activation initiated. Check zone status."

```
copy run start
```

Cisco Nexus 5548 B

```
zone name <VSPEX-01> vsan 1
member device-alias <VSPEX-01-B>
member device-alias <VNX5400-SPB-B1>
member device-alias <VNX5400-SPA-A1>
exit
zoneset name <VSPEX> vsan 1
member <VSPEX-01>
exit.
zoneset activate name <VSPEX> vsan 1
```
The Nexus should respond with "Zoneset activation initiated. Check zone status."

```
copy run start
```
To help ensure you have configured WWN correctly issue the command:

```
Show zoneset active
```
Notice that an asterisk should appear at the beginning of each line indicating the wwn has been detected.

```
sjc2-151-E21-5548-B(config)# show zoneset active
zoneset name VSPEX vsan 1
  zone name VSPEX-01 vsan 1
  * fcid 0xca08ef [pwwn 50:06:01:6c:08:60:06:a1] [VNX5400-SPB-B1]
  * fcid 0xca07ef [pwwn 50:06:01:64:08:60:06:a1] [VNX5400-SPA-A1]
  * fcid 0xca0020 [pwwn 20:00:00:25:b5:99:00:40] [VSPEX-01-B]
sjc2-151-E21-5548-B(config)#
```

Present Boot LUN to Additional Paths

When the zones and zonesets have been updated to reflect the multiple paths to the LUN, it is necessary to configure the EMC VNX5400 SAN to present the boot LUN to the additional paths.

1. Power off the server before starting.

2. In Unisphere, go to Hosts > Initiators and click the Create button to add a new initiator.

3. The goal is to create an initiator to each port on the VNX5400. You will have two initiator records for each WWNN and WWPN combination for the server that match your zone entries on the Nexus switches. Be sure to select the appropriate SP-Port. Also select Existing Host and select the proper host.

4. When all initiators are defined and registered, select Hosts > Storage Groups.

5. Select the storage group for this server.

6. Select the server from the Hosts to be Connected column and move it to the Available Hosts column. Click Apply.
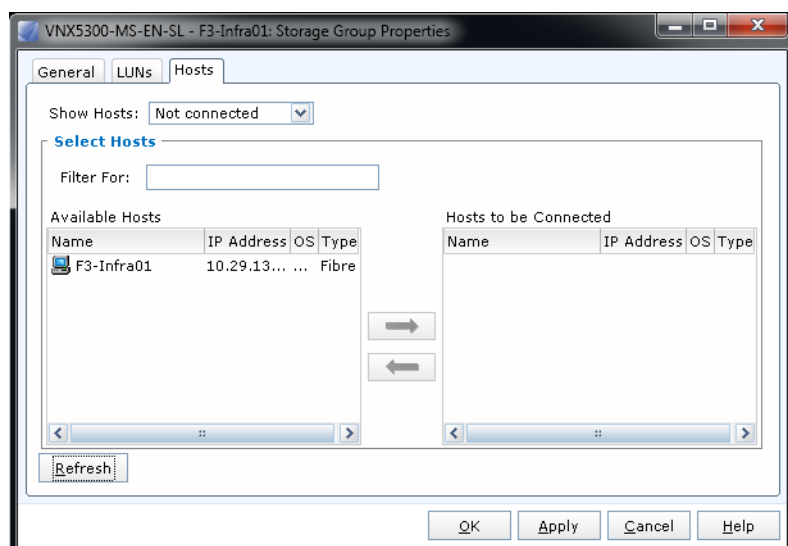
7. Select the server from Available Hosts and move it to Hosts to be Connected.

8. Click OK.

9. This refreshes Unisphere to recognize the fact that this host now has four paths.
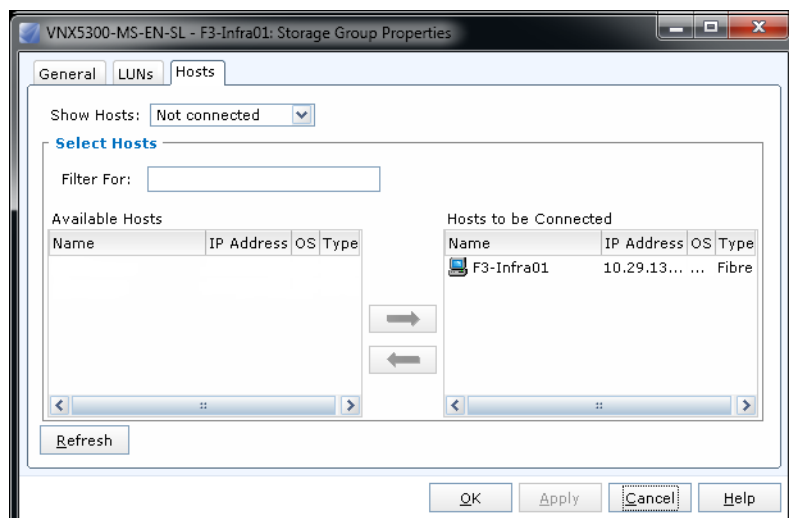
10. Boot the server.

11. During the boot, you will see four paths, one matching each port on the VNX5400.

```
Adapter BIOS Disabled. No Logical Drive Handled by BIOS on HA - 0
0 Virtual Drive(s) handled by BIOS
Press <Ctrl><H> to Enable BIOS


Cisco VIC FC, Boot Driver Version 2.1(2a)
(C) 2010 Cisco Systems, Inc.
    DGC       50060165086006a1:000
    DGC       5006016d086006a1:000
Option ROM installed successfully

Cisco VIC FC, Boot Driver Version 2.1(2a)
(C) 2010 Cisco Systems, Inc.
    DGC       50060164086006a1:000
    DGC       5006016c086006a1:000
Option ROM installed successfully
```

12. Log into the server.

13. From an elevated command prompt or PowerShell window issue the command mpclaim -s -d 0

You should see four entries, similar to what is shown in this screen shot, validating that you have properly configured MPIO.

```
PS C:\Users\Administrator> mpclaim -s -d 0

MPIO Disk0: 04 Paths, Round Robin with Subset, Implicit and Explicit
    Controlling DSM: Microsoft DSM
    SN: 6061606D703204931EFCC9011E311
    Supported Load Balance Policies: FOO RRWS LQD WP LB

    Path ID            State              SCSI Address      Weight
    --------------------------------------------------------------------
    0000000077020001 Active/Optimized    002|000|001|000    0
    * TPG_State : Active/Optimized   , TPG_Id: 1, : 5

    0000000077020000 Active/Optimized    002|000|000|000    0
      TPG_State : Active/Optimized   , TPG_Id: 2, : 15

    0000000077010001 Active/Optimized    001|000|001|000    0
      TPG_State : Active/Optimized   , TPG_Id: 2, : 16

    0000000077010000 Active/Optimized    001|000|000|000    0
    * TPG_State : Active/Optimized   , TPG_Id: 1, : 6


PS C:\Users\Administrator> _
```

Run Sysprep

When the image is properly configured for multipath, Microsoft's sysprep utility can be used to create an image that can be used for cloning to quickly provision any additional physical hosts needed in the environment.

1. From an elevated command window, enter the command c:\Windows\System32\sysprep\sysprep.exe

> ✎
> **Note** The sysprep utility is unique for each version of the operating system. Do not try to use one from another installation.

```
Administrator: Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\>c:\Windows\System32\sysprep\sysprep.exe_
```

2. Select Enter System Out-of-Box Experience (OOBE) from the System Cleanup Action drop-down menu.

3. Select the Generalize box.

4. Select Shutdown from the Shutdown Options drop-down menu.

5. Click OK.

When the KVM console shows the physical server has shut down, clones can be made of the LUN for use by all the physical hosts.
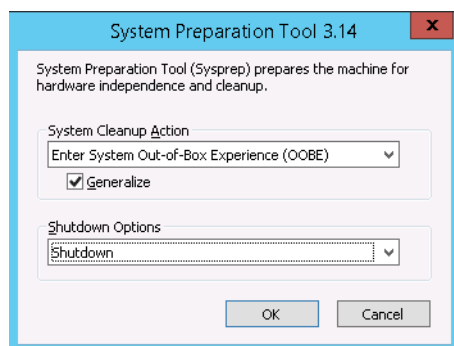
Cloning the Sysprepped Image

Removal of the Source Master Image

After installation of the Windows Server instance and execution of the sysprep process, it is necessary to remove the source LUN from the Service Profile that was used to build the image. To remove the LUN from the Service Profile, use Unisphere to remove the Master_Boot LUN and the assigned host from its storage group. You may also want to remove the host initiator entries for the Master_Boot if you are using a different naming convention for production servers.

With the base sysprep image created and the LUN containing the sysprepped image removed from the service profile, clones can be taken in order to replicate the contents of the master LUN for other servers in the environment. Prior to copying the data, target devices need to be created to be associated with the planned clone sessions. The clones can be created with ESI or through Unisphere.

Create Target LUNs with ESI

Run the ProcessStorageRequests.ps1 script to create the appropriate clone target devices. This script uses the Storage_Luns.xml found in the Sample PowerShell Scripts. It must be modified to reflect the number of hosts for which boot LUNs will be created and the naming conventions used by the customer.

Create Target LUNs with Unisphere

Alternatively, EMC Unisphere can be used to create the target LUNs.
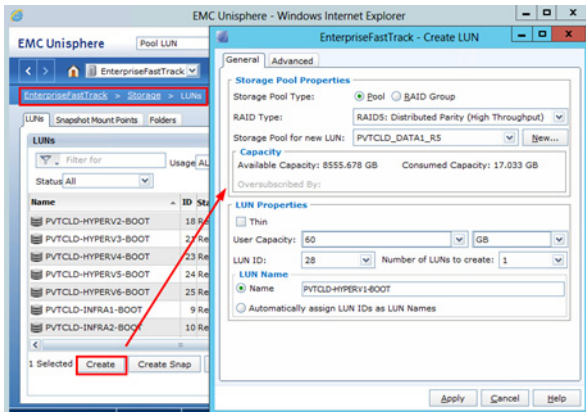
Create Clones with ESI

Now that the clone target LUNs are created, the clone process can be run. To automate the clone process, modify the ProcessClones.xml configuration file contents to reflect the customer environment. Run the ProcessClones.ps1 script found in Sample PowerShell Scripts which reads the XML configuration file using ESI and naviseccli.

The script will create concurrent clone copies and wait for 100% synchronization. When the copies are complete, the script will delete the clone relationship and the target LUNs can be used for deployment.

Create Clones with EMC Unisphere

Alternatively, the following process can be executed from Unisphere to create the clone relationships and copy the data from the master LUN to the boot target LUNs.

1. In Unisphere, go to Data Protection > Clones.
2. Select the Create Clone Group link from the protection side-bar.
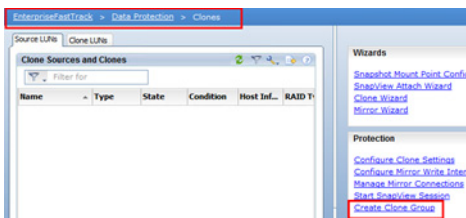
3. Give the Clone Group a name and select the master boot image LUN as the "LUN to be Cloned." Then select OK
4. Select Yes after reviewing the confirmation screen and OK after the group creation returns with success.
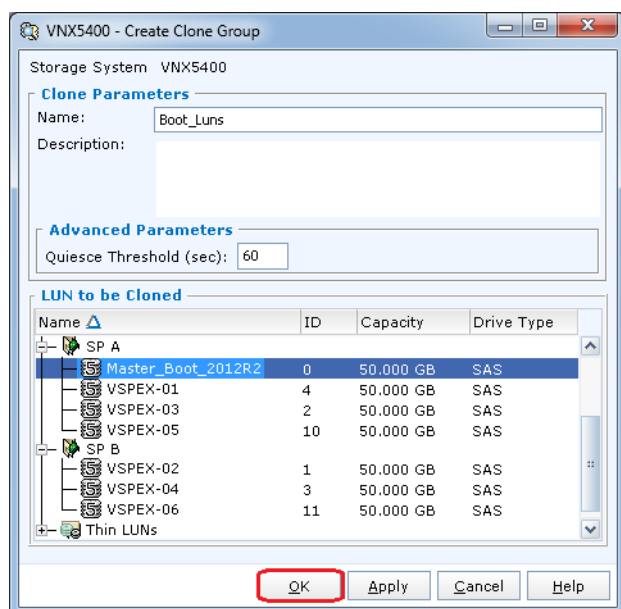
5. Right-click on the newly created Clone Source and select Add Clone.
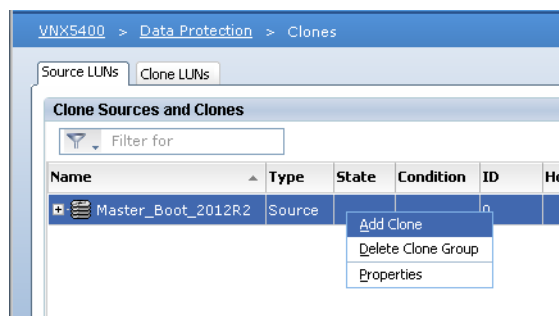
6. Select the appropriate clone target LUN intended for Boot from SAN and select Apply.
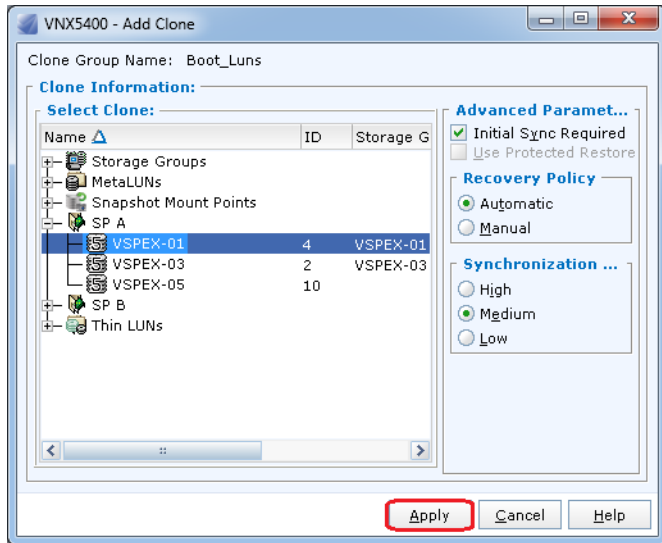
7. When replicating between thin LUNs the following warning will pop up. Select Yes.
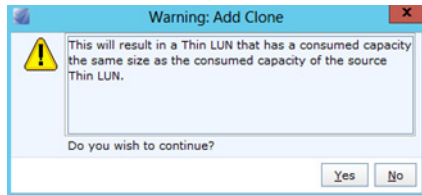
8. Confirm the target LUN will be overwritten by selecting Yes.

9. Select OK after the successful addition of the clone.

10. Repeat the previous steps to add the desired number of clone copies. Up to 8 can be added concurrently.
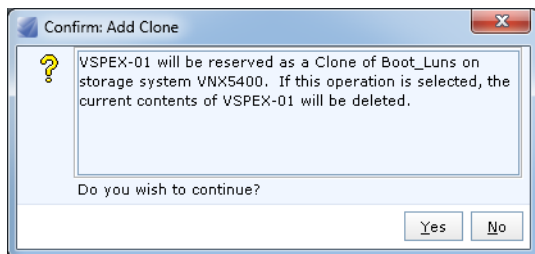
11. Verify the clones are synchronizing from the Clone LUNs tab.
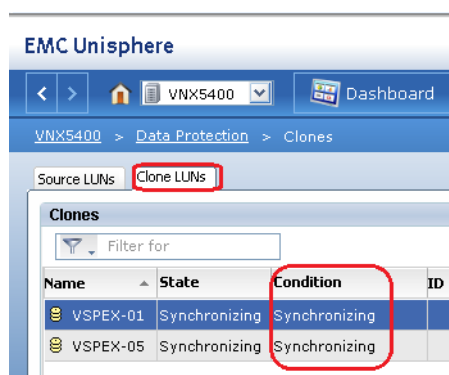
12. To get more detail on synchronization, right-click on a clone LUN and select Properties.

13. Each clone will have its own tab. Within each tab will be a Synchronized percentage.

14. Wait for all clones to get to a "Synchronized" "State" before continuing.
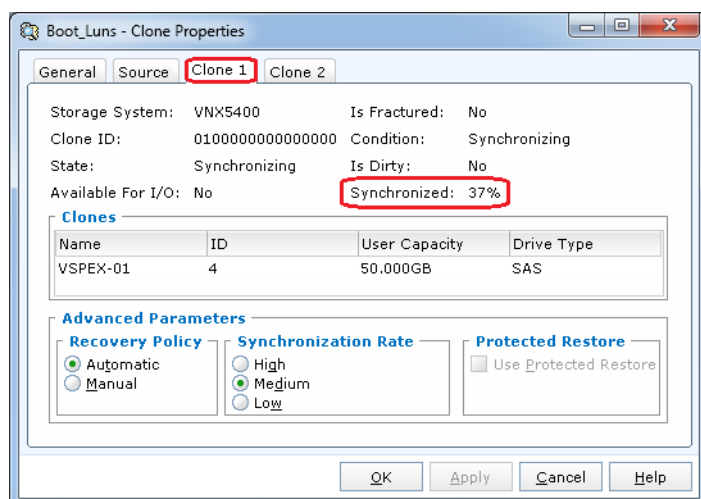
15. Delete each fractured clone. Select one clone at a time, right click, and select Delete.

16. Select OK following the successful delete.
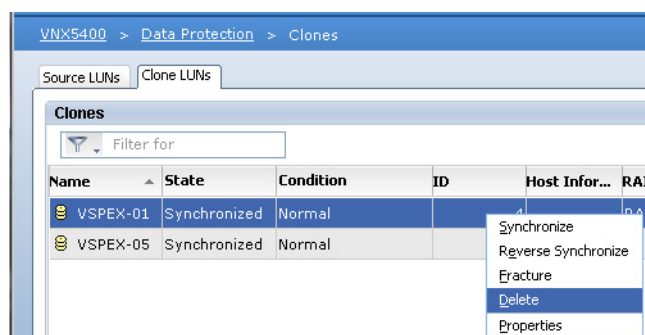
17. Optionally delete the clone group. From the Source LUNs tab, right click on the group and select Delete Clone Group.

18. Confirm the deletion by selecting Yes at the following screen.

This completes the cloning process.

✎

Note   It is recommended to keep this clone group to speed deployment of subsequent servers as your needs expand.
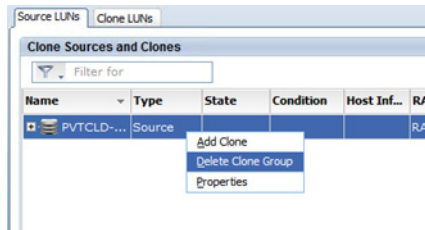
Boot from Sysprepped LUNs

After a clone of the sysprepped LUN has been created for each physical server to be built, you need to zone and mask the LUN before completing a build from the sysprepped image presented.

Zone the Network

Presenting the LUNs to the various hosts is a combination of configuring the zones and zonesets on the Cisco Nexus 5548 switches and masking the LUNs through Unisphere or naviseccli. The detailed steps for this were shown previously, so they will be summarized here.

Zoning

- Create the device alias for each service profile with the value of the fabric A WWPN defined on the A Nexus, and the value of the fabric B WWPN defined on the B Nexus.
- Create a zone for each service profile on each Nexus containing the device alias for appropriate server WWPN and both WWPNs of the associated EMC interfaces.
- Add the created zones to the zoneset and activate it.

Figure 18 shows the end result of this step will provide a listing of the zoneset. (WWPN values will differ for each environment).

Figure 18   Example Zoneset for Cisco Nexus 5548 A

Mask Boot LUNs to Service Profiles

Following the cloning and zoning processes, the boot LUNs can be presented to their respective service profiles. The same XML configuration file used to create the boot LUNs can be used in conjunction with the PostClone_AddViaWWPN.ps1 script to present the boot LUNs to the servers. The script will also register the appropriate initiators with the storage array and create the necessary storage groups along with presenting the LUNs to the appropriate servers.

An alternative to using the script would be to use the Unisphere management GUI as outlined previously in the "Mask Boot LUN with EMC Unisphere" section. Following the masking operations, start each host and complete the mini-setup to tailor each node with things like name, IP addressing (if fixed IP addresses are used), and join to the domain.

Complete Image Builds from Sysprepped Images

When the sysprep image has been cloned and the LUNs are properly masked so the boot volumes only appear to the owning host, every server must complete its installation. Booting from a sysprep image runs what is referred to as a 'mini-setup'.

> **Note**   This document does not describe the use of an unattend file. If your organization makes use of unattended installations of sysprep images, that can be used to replace these steps.

1. Open Cisco UCS Manager.
2. Select the Servers tab.
3. Open Service Profiles.
4. Select <VSPEX-01>.
5. Click on KVM Console to open a window from which you can manage the mini-setup.

The association between the service profile and the blade should have taken effect when you created the service profile, so you should see the first screen of the Windows Server mini-setup. If it is still booting when you connect to it, you may see a series of progress messages display as the system completes the initial setup.
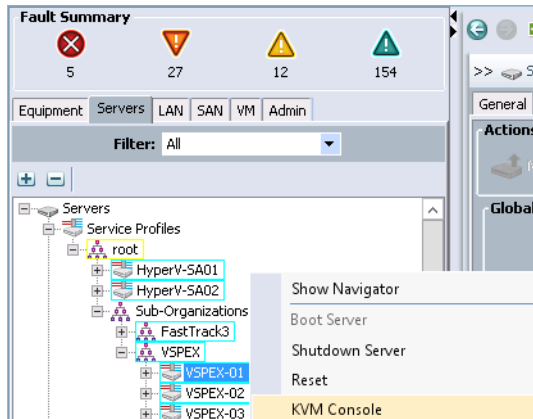
6. Make any necessary changes to the Region and Language settings.

7. Click Next.
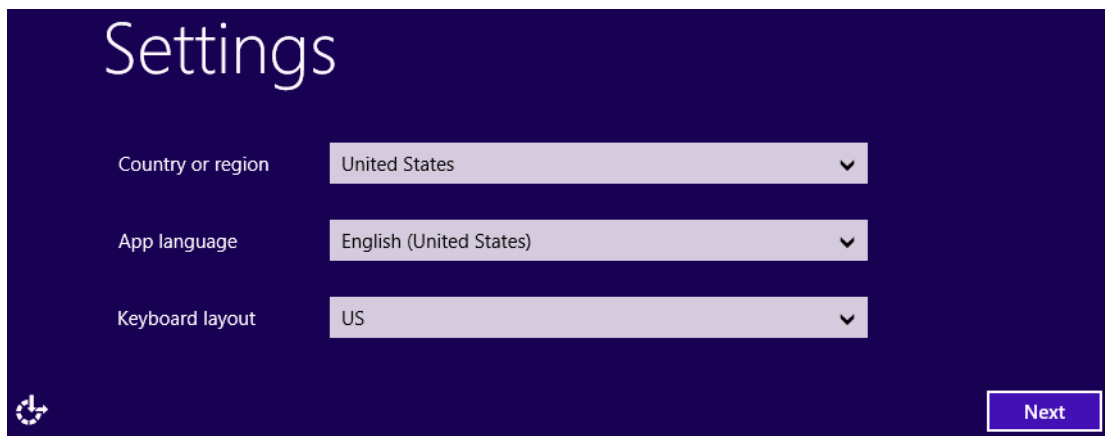
8. Click the box next to I accept the license terms for using Windows.

9. Click Accept.
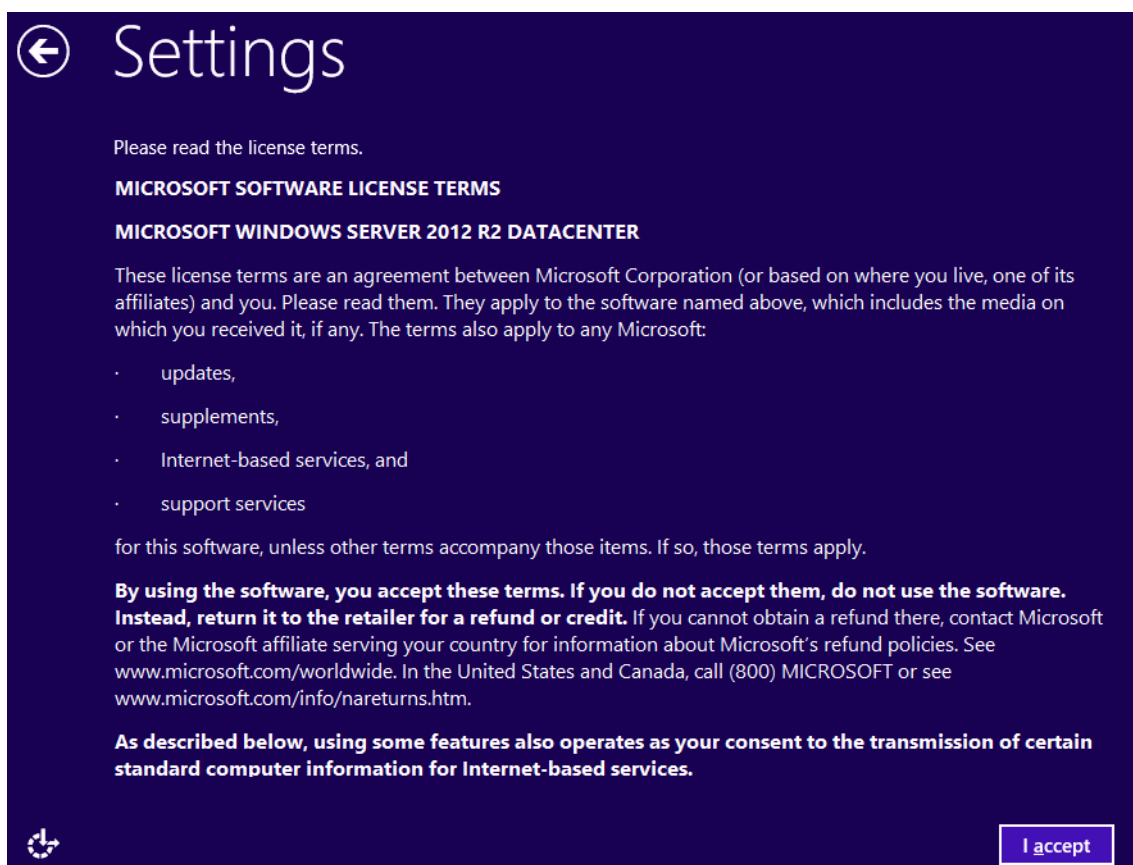
1. Enter a complex password. The password must contain three of the following and be at least eight characters in length.

- Upper case character
- Lower case character
- Digit
- Special character

2. Re-enter the same password.
3. Click Finish.

At this point, you will have a complete base image. This means you will need to activate Windows, change the name of the system, join to the domain, configure your network settings, and complete any other tailoring required to meet your company requirements for Windows Server installation.

Configure Networks

It is recommended that you rename the network adapters from the Windows default values of "Local Area Connect #x" to reflect the actual network from the Cisco UCS Service Profile. You can use the manual procedure defined earlier in the document, or you can use the sample PowerShell script, Set-UcsHyperVAdapters.ps1, found in Sample PowerShell Scripts. This script requires that the machine domain-joined and the script is being run from a workstation that has the Cisco UCS PowerTool installed.

The Set-UcsHyperVAdapters.ps1 script will assign a fixed IP address to each NIC based on the a 192.168.xx.yy notation where xx is the VLAN read in from Cisco UCS and yy is a specific value assigned so that the last octet of each address is the same on for each server. It also sets each adapter, except the excluded (generally the management) adapter so that it does not register itself in DNS. It is best to have only the primary (management) address register in DNS.

Depending upon your configuration, you might have DHCP set up for every network. In which case, it is not recommended to use this script without first modifying it to not alter IP addresses.

Configure Hyper-V Virtual Switches

To configure Hyper-V virtual switches, do the following:

1. From Server Manager > Tools, select Hyper-V Manager.
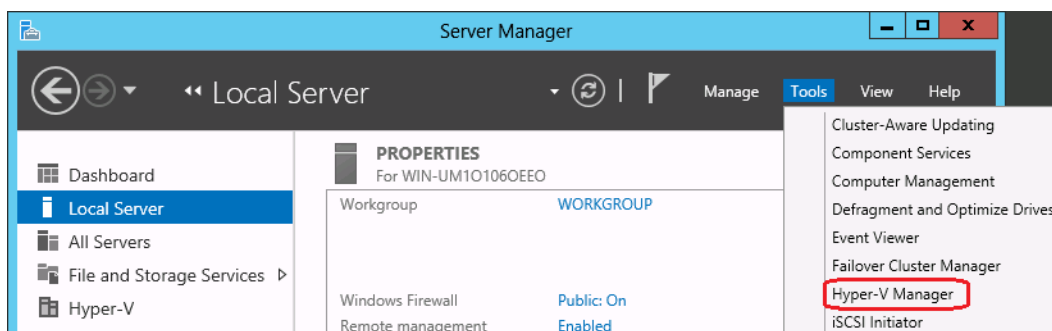
2. From the Hyper-V management console, select Virtual Switch Manager from the right-hand side.
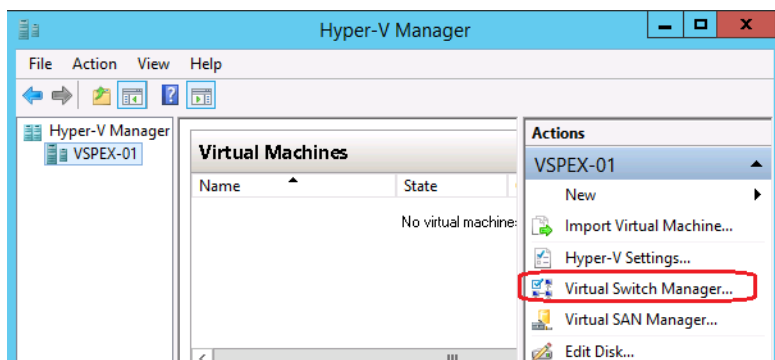
3. Make sure that New virtual network switch is highlighted on the left-hand side and External is highlighted on the right-hand side.

4. Click Create Virtual Switch.
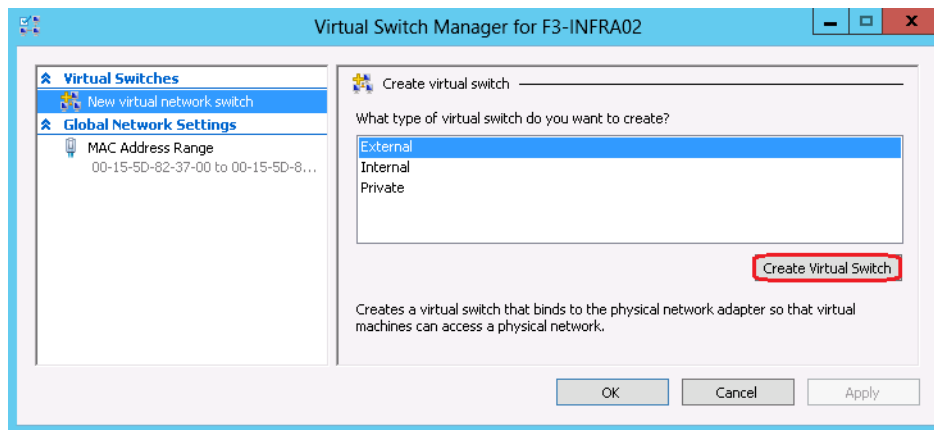
5. Enter an appropriate name in the Name field.

6. Make sure that you select the correct Cisco VIC Ethernet Interface from the drop-down list for External network.

7. Uncheck the Allow management operating system to share this network adapter.

8. Repeat previous step and this step for the VMaccess and ClusComm NICs.

9. Click OK to complete creating these virtual network switches.
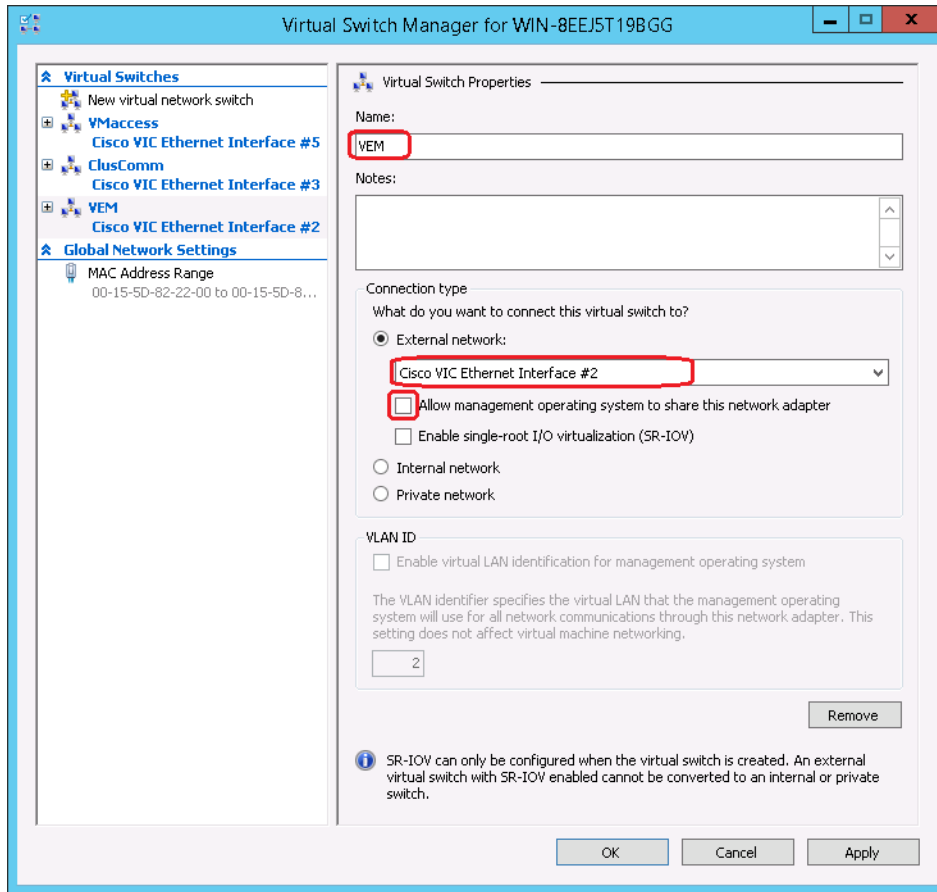
10. A warning window will display cautioning about possible disconnection from the machine. You are not accessing the physical host through any of the network adapters selected; you can click Yes with no issues.
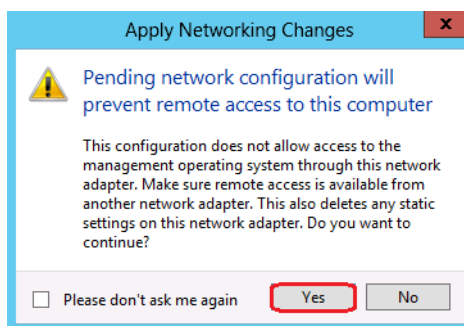
When completed, your Network Connections will like the screenshot below. (Device names will vary)

Network interfaces that have been defined as virtual switches will show Enabled in the Status column.

The two interfaces that were teamed for LiveMigration also show as Enabled.

The Unidentified network entries are networks that are 'private' networks, i.e. not used for accessing the outside network.

For most configurations, you will see only the Mgmt network with internet access.
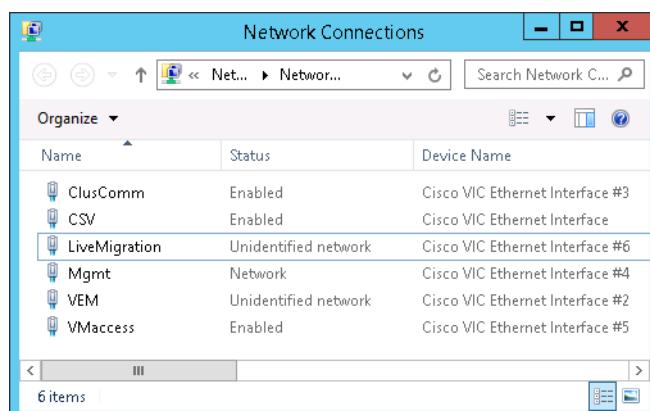
Unconfigure DNS Registration

If you do not use the Set-UcsHyperVAdapters.ps1 script to rename and partially configure the NICs, it is a good practice to remove all but the management NIC from DNS registration.

1. In Network Connections, right-click on a network that is still assigned to the host (e.g. CSV or LiveMigration) and select Properties.
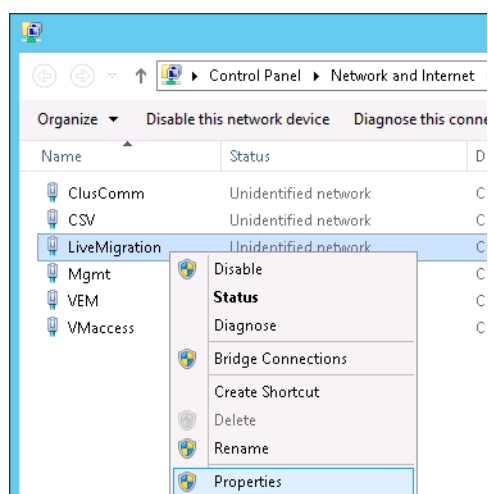
2. In the Properties window, scroll down to the Internet Protocol Version 4 (TCP/IPv4) line and select it.
3. Click on Properties.
4. Click Advanced... in the Properties windows that displays.
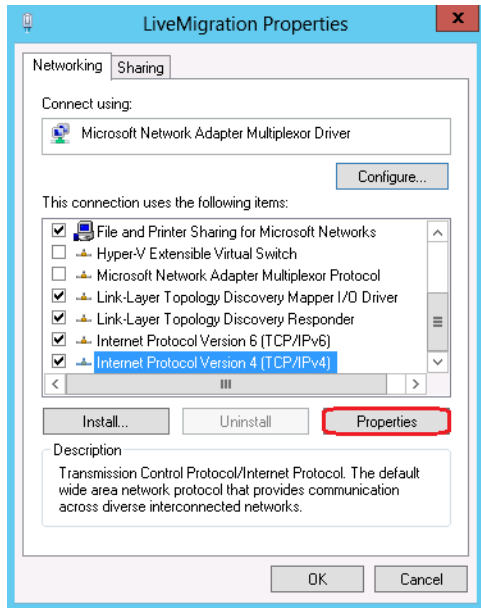
5. Select the DNS tab and uncheck the Register this connection's addresses in DNS box.

6. Click OK twice and then Close.
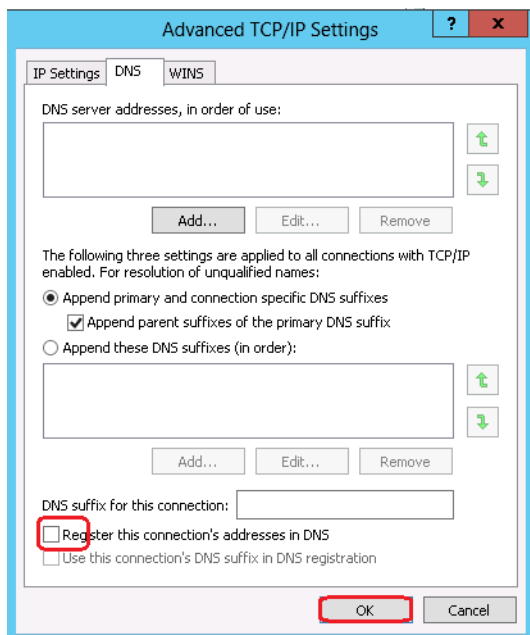
Binding Order

It is a good practice to have a proper binding order of NICs.

7. On the Network Connections window, click the Alt key on the keyboard to display the toolbar for the window.

8. Click on Advanced and select Advanced Settings... from the drop-down menu.
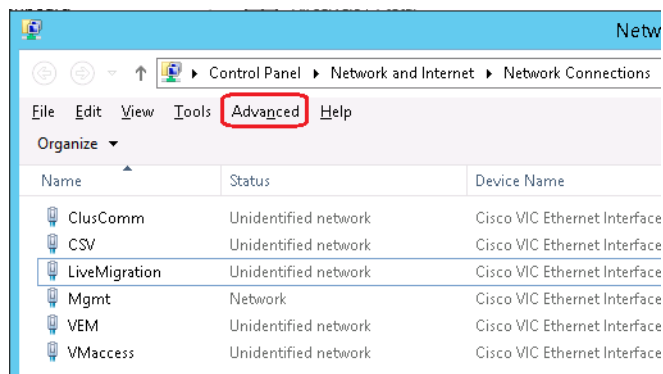
9. Using the up and down arrows on the right-hand side of the screen, select the various connections and arrange them so Management, LiveMigration, and CSV are ordered as first, second, and third.

10. Click OK to continue.
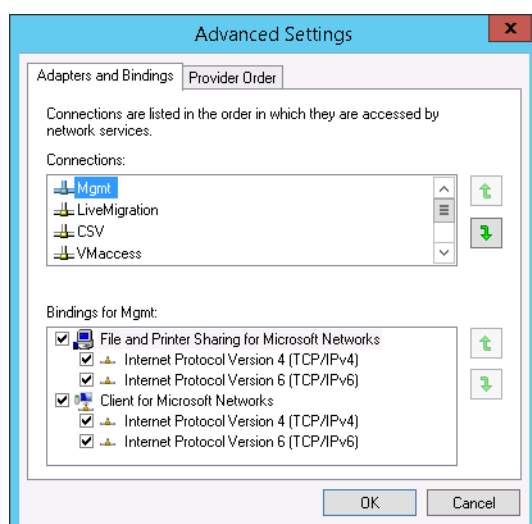
11. Repeat the Configure Network section for each server in your environment.

Install EMC PowerPath

As a part of tailoring each system, EMC PowerPath can be installed for enhanced multi-pathing functionality EMC PowerPath for Windows version 5.7 or higher should be used.

1. Launch the EMC PowerPath installer, EMCPower.X64.signed.5.7.b223.exe

2. Click Next.
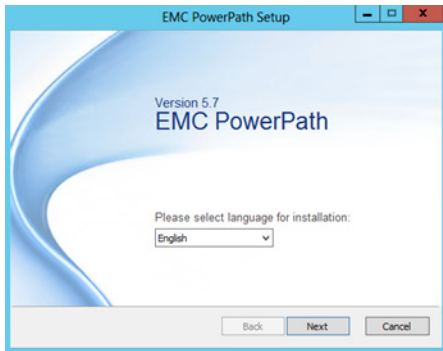
3. Click Next at the copyright information screen.
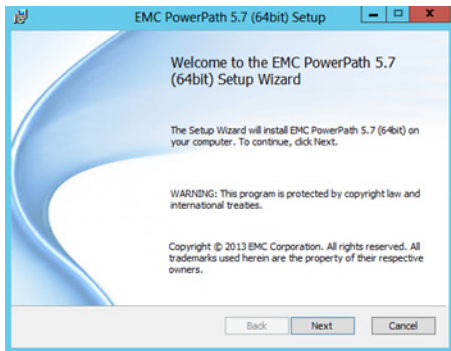
4. Accept the default feature installation options and select Next.
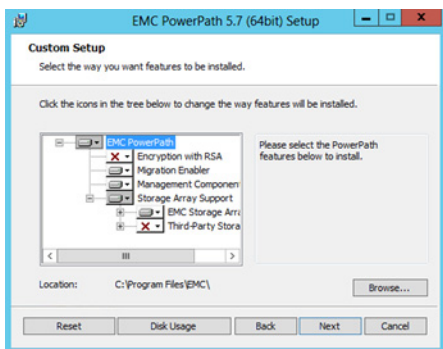
5. Select Install.
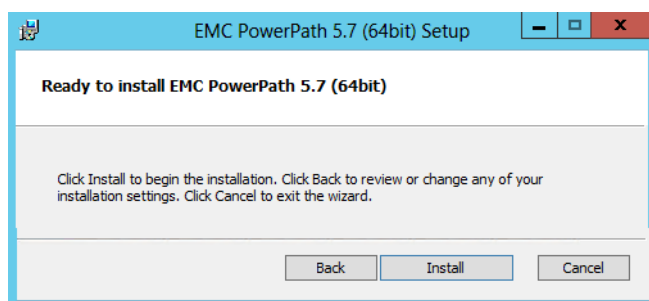
6. When prompted, enter the appropriate license key for your environment and select OK.

> **Note** If no license key is entered, PowerPath will be unlicensed and will run in a "basic failover" mode, which allows two storage port connections to one HBA. The other HBA will be marked as unlicensed.

7. An appropriate license should be obtained, otherwise PowerPath should be uninstalled and native Windows Server 2012 MPIO should be used.
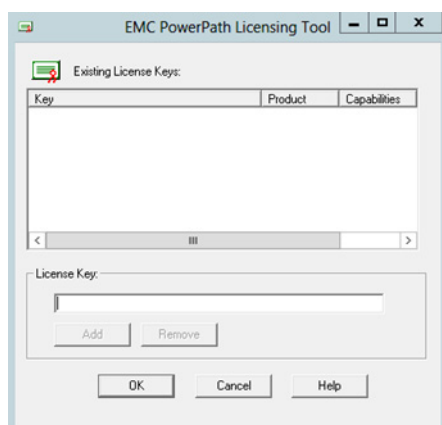
8. Select Finish.
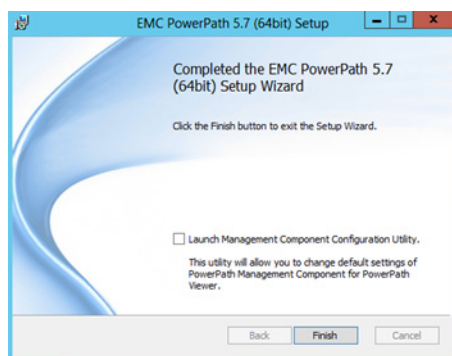
9. Select Yes to reboot the server and complete the installation.
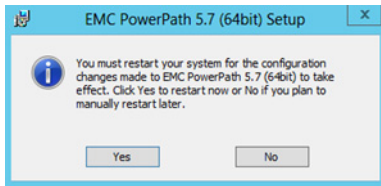
Install EMC Unisphere Host Agent

The Unisphere Host Agent allows for host specific information to be sent to management applications, like Unisphere, for ease of administration. LUN mapping and Operating System information as well as initiator information can be forwarded from a server to the VNX via the agent. Follow the procedure below to install the Unisphere Host Agent on either a physical Windows Server 2012 R2 server or Virtual Machine.

> **Note** For virtual machines with iSCSI access to the array, configure the iSCSI connections first prior to installing the agent. This will allow the agent to discover the configured paths and automatically register the iSCSI initiators with the VNX.

10. Run the following command, from an elevated PowerShell command window, to open the required firewall port for the Unisphere Host Agent:

```
New-NetFirewallRule -Name UniAgent-TCP -DisplayName UniAgent-TCP -Action Allow -Direction Inbound
-Protocol TCP -LocalPort 6389
```
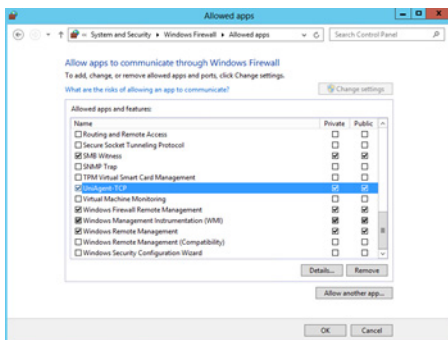

11. Launch the EMC Unisphere Host Agent installer,
    UnisphereHostAgent-Win-32-x86-en_US-1.2.25.1.0163-1.exe

12. Click Next.
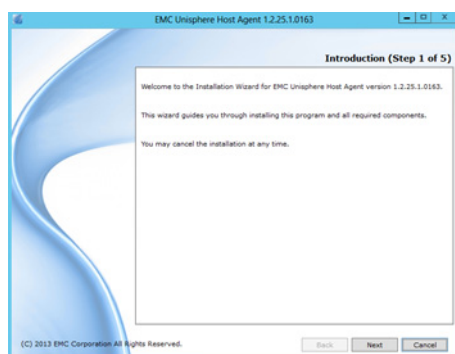
13. Choose installation directory and select Next.
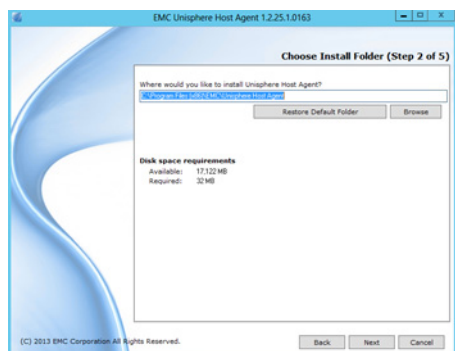
14. Enter the IP Addresses of each block service processor (SPA and SPB) and select Next.
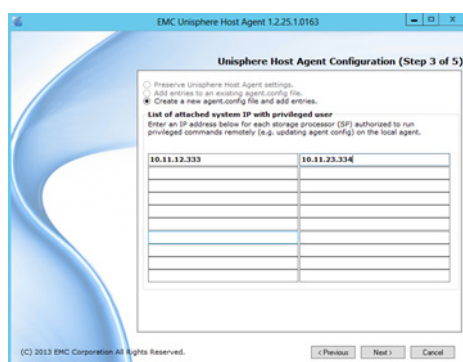
15. Select Next after reviewing the Pre-Installation Summary to begin the installation.
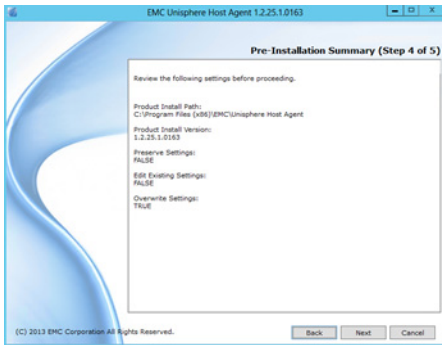
16. Select Finish to complete the install.
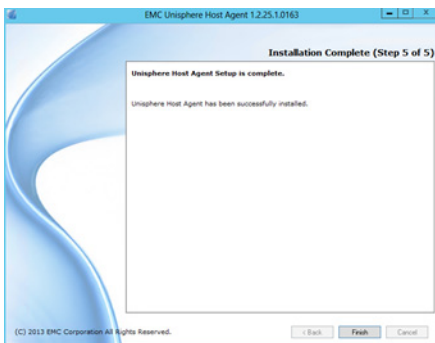
17. The Unisphere Host Agent will bind to the first NIC within the binding order on the host. This needs to be a NIC which can communicate with the VNX SP IP addresses If this ends up being the incorrect NIC, use the agentID.txt to set the correct interface.

18. In the installation directory for the Unisphere Host Agent (default = C:\Program Files (x86)\EMC\Unisphere Host Agent) create a file called agentID.txt. Within the file, place the server name on the first line, press enter, and then place the IP address of the desired management interface on the second line.
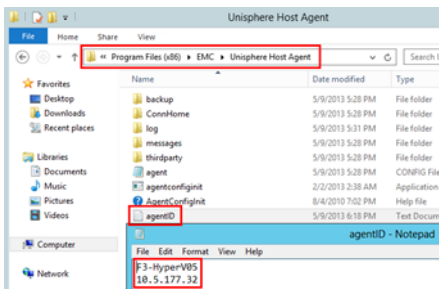
19. From a command window or from the services control panel start the "Navisphere Agent"

```
net start "Navisphere Agent"
```
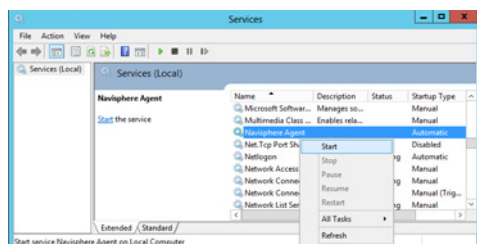
20. Validate the host agent is pushing information to the VNX from the Hosts > Host List menu in Unisphere.
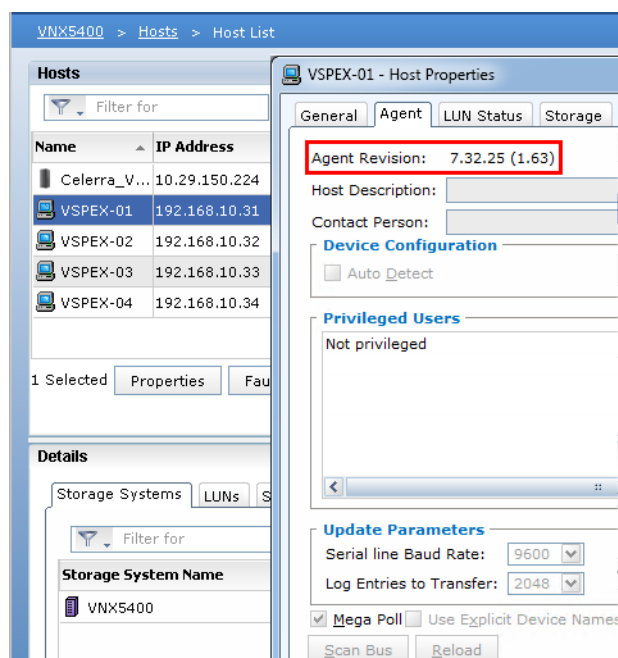
Create Hyper-V Failover Cluster

When you have completed the build of the servers to SAN boot in a multipath IO environment, have all the network adapters configured the same, and the hosts joined to the Active Directory domain, you will create the failover cluster on which all the virtual machines will be deployed. This cluster can be expanded up to a total of 64 hosts for running VMs within the Microsoft virtualized environment.

Create Shared Storage

Microsoft Failover Clusters use shared storage for storing the VMs. For a 300 VM deployment, 3 storage pools are created with 45 SAS drives and 2 EFDs in two pools and 20 SAS drives and 2 EFDs in the third pool. Two 7TB LUNs will be provisioned from each 45 drive pool and two 3TB LUNs will be provisioned from the 20 drive pool. A small LUN of 1 GB in size must be created on any of the 3 pools to act as the witness disk in the cluster. There will be 7 LUNs created in total.

- Witness Disk - 1 GB
- Cluster Shared Volume 1 - 7 TB

- Cluster Shared Volume 2 - 7 TB

- Cluster Shared Volume 3 - 7 TB

- Cluster Shared Volume 4 - 7 TB

- Cluster Shared Volume 5 - 3 TB

- Cluster Shared Volume 6 - 3 TB

When these LUNs are created, they need to be added to each of the storage groups associated with each node in the Hyper-V cluster. When the same LUN is added to multiple storage groups, the VNX will display an error message cautioning about the possibility of corrupting data. The clustering software controls access to the LUNs, so that is acceptable.

It is recommended that at least one CSV is created for each node in the cluster. A new feature of Windows Server 2012 R2 is to distribute the CSV ownership across the cluster nodes to ensure that management functions are not concentrated on any single node. However, to speed the time it takes to run the Cluster Validation Wizard, you should not present these volumes to the cluster until after the cluster is formed. The Cluster Validation Wizard will create multiple combinations of failover scenarios to be tested. The more disks in the test, the longer it takes to complete the validation. Since the storage array will be tested with the initial two disks, subsequent disks can be added later with the knowledge that the initial disks were configured correctly.

You may also want to create larger or smaller LUNs depending upon the practices in place in the customer environment. The VSPEX reference VM allocates 100 GB per VM, though a typical boot VM is only 50 GB. During normal operations, Hyper-V reserves enough space on disk to capture the memory of the running VM in case of certain types of failures. This space is in addition to the space assigned to the virtual hard drive files. To run 300 VMs on the six nodes of a cluster assumes that 60 reference VMs will run on each of five nodes, with a single node for backup when one of the other nodes needs to be taken out of service. In order for a single CSV to contain all 60 VMs, it is recommended to create the CSV to be 7 TB to allow for a little overage.

Before you can test and form the cluster, it is necessary to format the shared LUNs as NTFS volumes. Perform the following steps on only one node of the cluster to format the drives.

1. From Server Manager on one of the hosts to which the storage has been presented, select Tools > Computer Management.

(Alternatively, type compmgmt.msc into a command or PowerShell window.)

2. Right-click on the area under the disk number designation and select Online to bring the volume online.

3. Repeat for each new LUN.
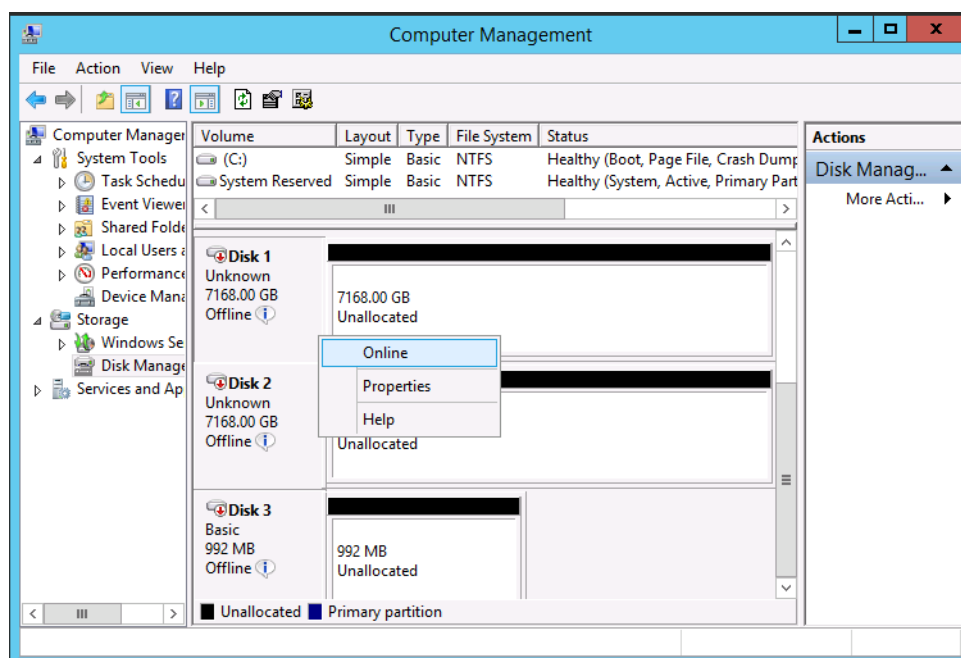
4. After all disks are online, right-click in the same area on one of the disks and select Initialize Disk.
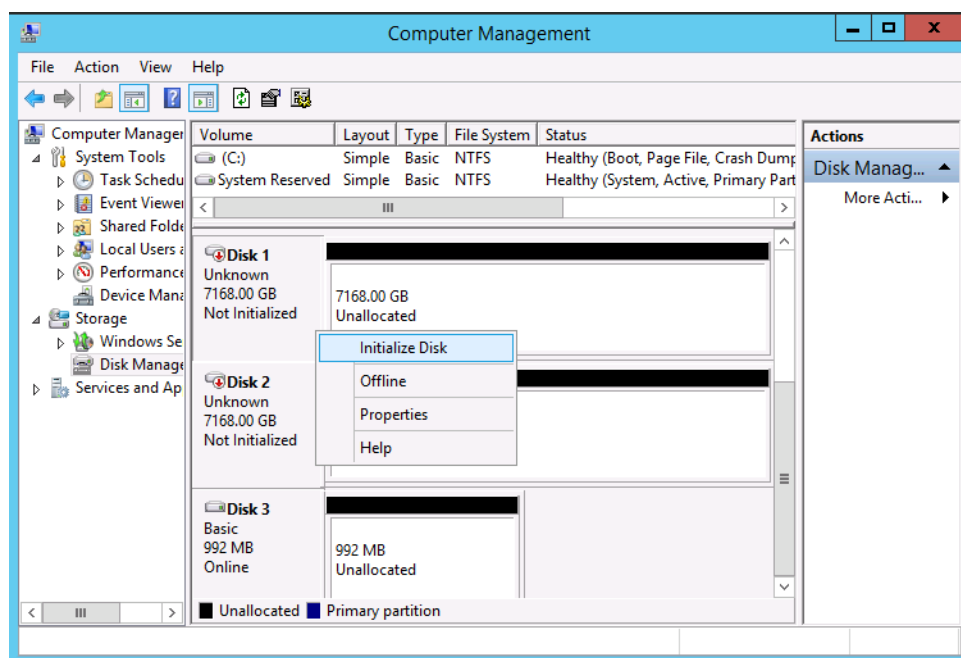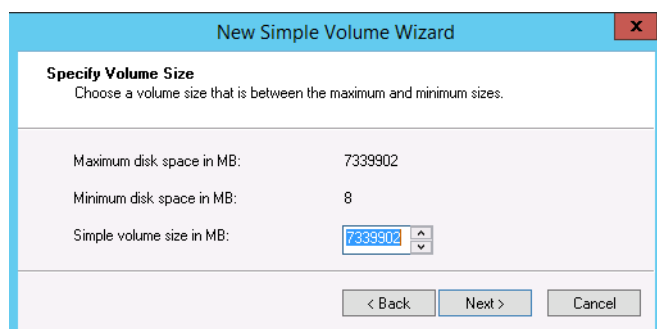
5. All uninitialized disks will be listed. It is recommended to use GPT disks for clustering; select the radio button by GPT (GUID Partition Table).

6. Click OK to start the initialization.
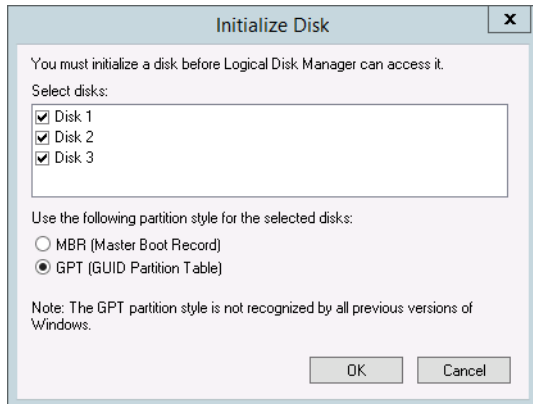
7. Right-click on one of the disks and select New Simple Volume. This brings up the New Simple Volume Wizard window.

8. Click Next to continue.
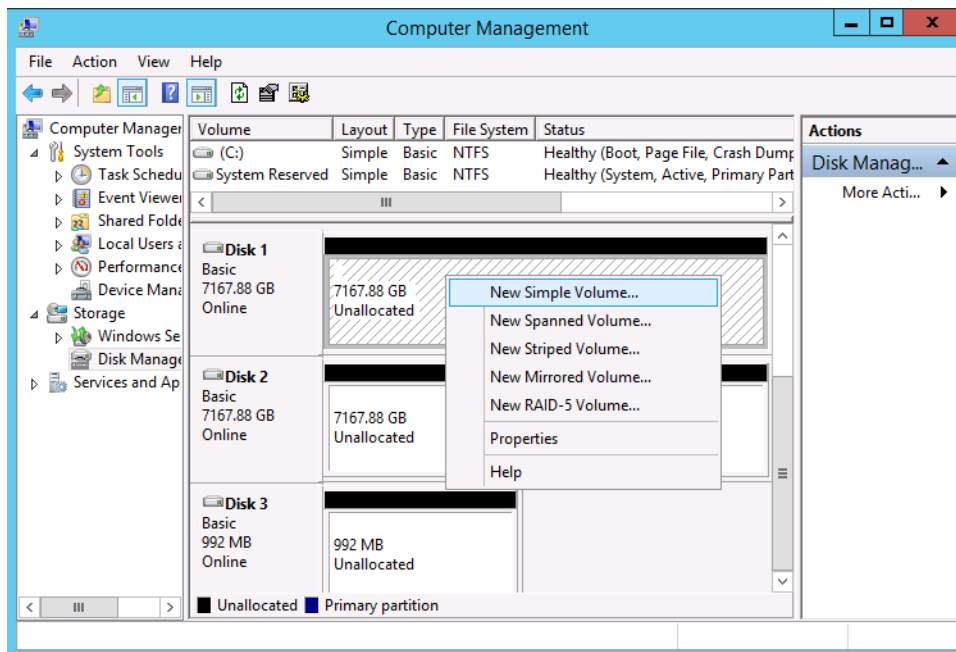
9. Accept the values in the Specify Volume Size window.

10. Click Next to continue.

11. In the Assign Drive Letter or Path window, click the radio button by Do not assign a drive letter or drive path.

> **Note**   Cluster Shared Volumes are accessed from mount points, so no drive letter is needed. Disk Witnesses are not accessed by any user functions, so no drive letter is needed.

12. Click Next to continue.

13. It is a good practice to enter a useful identifier in the Volume label field.

14. Change Allocation unit size to 64K for the large LUNs.

15. Click Next to continue.

16. A summary window displays. Validate what you selected. If any changes are needed, use the Back button to get to the window to correct it. Otherwise, click Finish to complete the formatting process.

17. Repeat the process to create a new simple volume on each LUN.
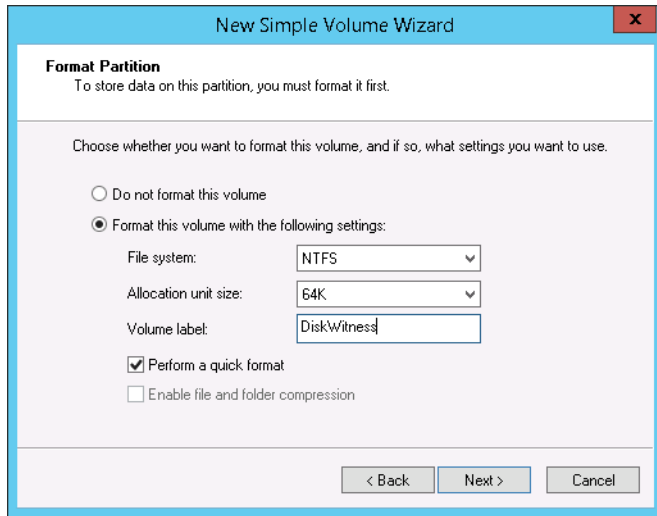
18. After all disks have been formatted and volumes created, place the disks offline.

19. Right-click on the disk and select the Offline option.
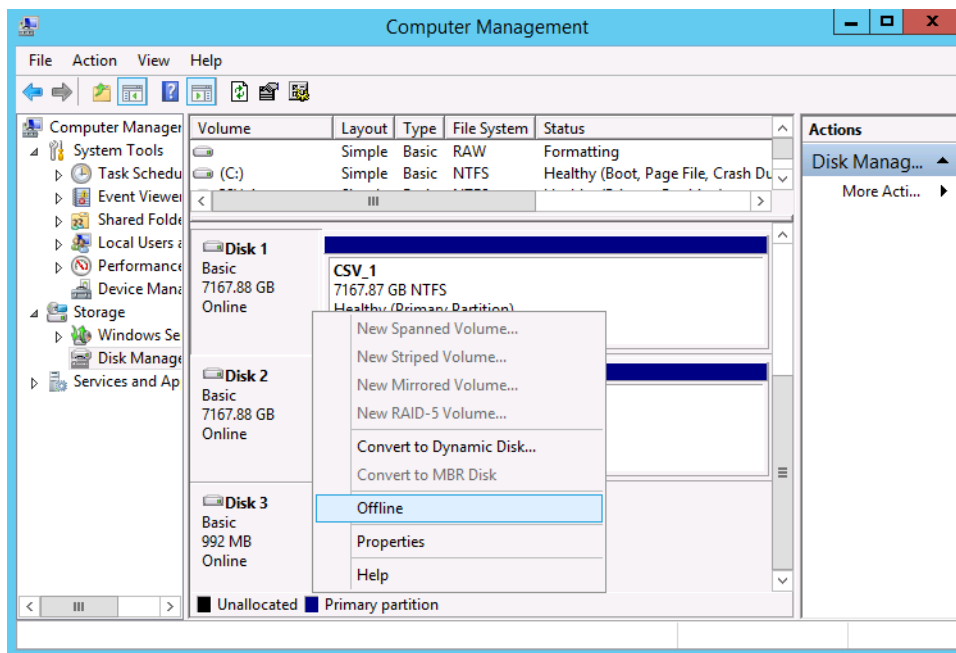
Before running the Cluster Validation Wizard, it is a good practice to bring the disks online and offline on the other node(s) of the cluster. The Cluster Validation Wizard will do this, too, but checking beforehand will save the time it takes to run the wizard if you need to do some troubleshooting.

Run Cluster Validation Wizard

1. The easiest way to run the Cluster Validation Wizard is to execute from a PowerShell window.

```
Test-Cluster F3-Infra01,F3-Infra02
```

It is not uncommon to have errors or warnings. The first run in the screen shot shows a message of HadFailures. Failures must be fixed before creating the cluster.

The second run shows a test run with no failures, but there were some warnings. Upon investigation, it was determined that the warnings were expected and the cluster can be created.

The last line in yellow gives the location of the report file detailing the test results.

Create Cluster

1. From Server Manager, launch the Failover Cluster Manager from Tools > Failover Cluster Manager.

2. In the Management section of the Failover Cluster Manager, select Create Cluster....

3. This launches the Create Cluster Wizard. On the Before You Begin window, click Next to continue.
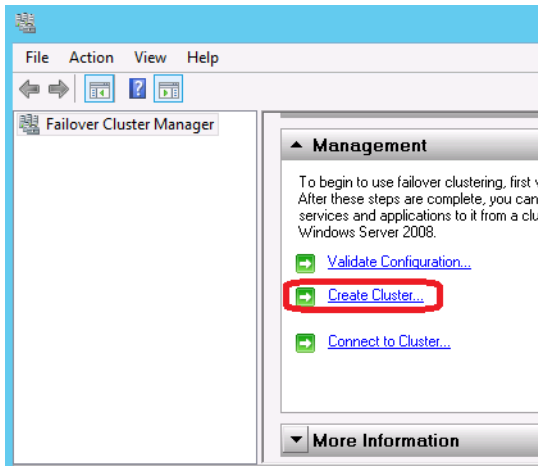
4. In the Select Servers window, browse Active Directory, enter the FQDN or NetBIOS names individually, or enter them in a comma separated list.

5. Click Next to continue after the nodes have been selected.
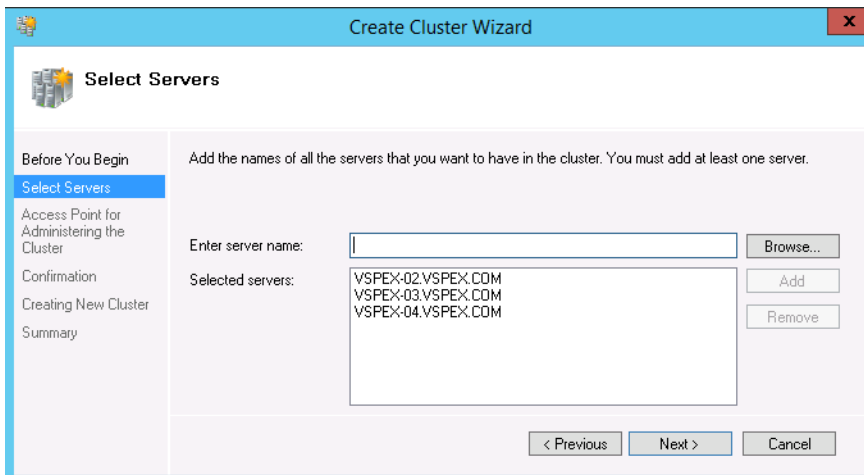
6. In the Access Point for Administering the Cluster window, enter a name in the Cluster Name field. This name will be added to Active Directory as a Cluster Name Object.

7. If you are not using DHCP for address assignment, you will be prompted to enter an IP address.

8. The Cluster Name and IP address will be registered in DNS.

9. Check your answers on the Confirmation window.

10. Click Next to create the cluster.

11. Click Finish on the Summary window. If any errors occurred, they would be listed on the summary window. They would need to be resolved before continuing.

12. The cluster can also be created with the following PowerShell command:

13. New-Cluster -Node <Node1>, <Node2> -Name <ClusterName> -StaticAddress <ClusterIPAddress>

If you are running from one of the nodes, the Failover Cluster Manager will show the cluster. If you are running from the workstation, you will need to use the option to Connect to Cluster and enter the cluster name.
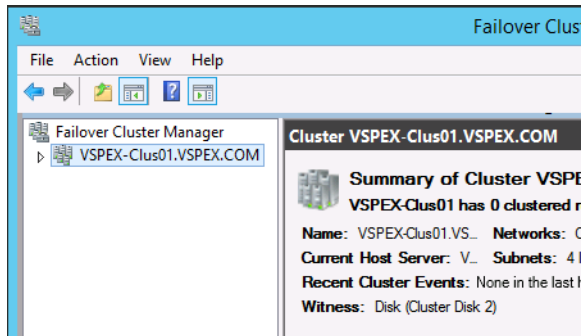
14. Expand the cluster name, expand the Storage, and click on Disks to expose the disks. By default, the create cluster process will automatically choose the smallest disk for use as the disk witness.
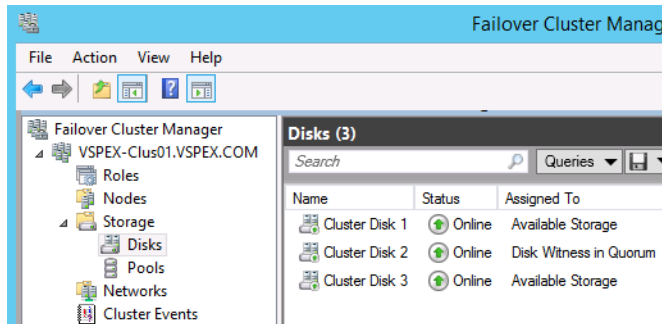
15. It is a good practice to change the name of the disks to be the same as the volume name.

16. Click on any disk and you can see the volume ID of the disk in the disk properties at the bottom of the window.

17. Right-click on the disk at the top of the window and select Properties.
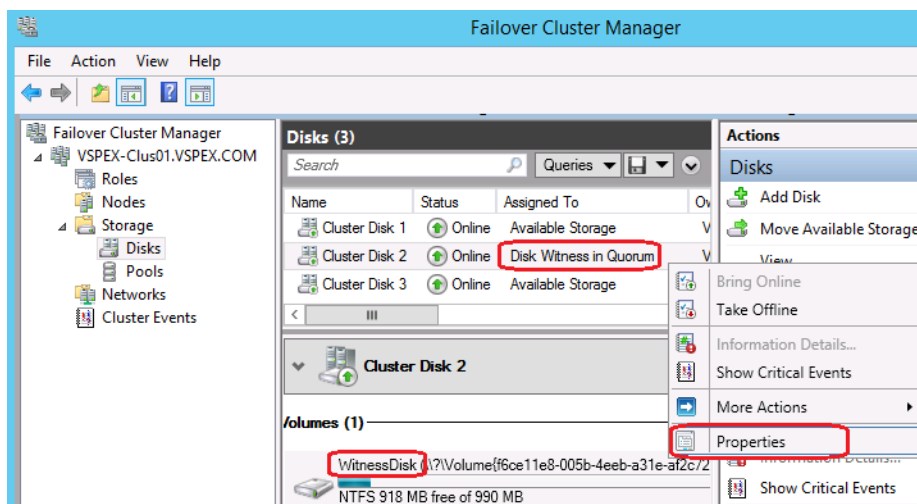
18. In the properties window, change the Name of the disk to be the same as the volume name.

19. Repeat for all disks.
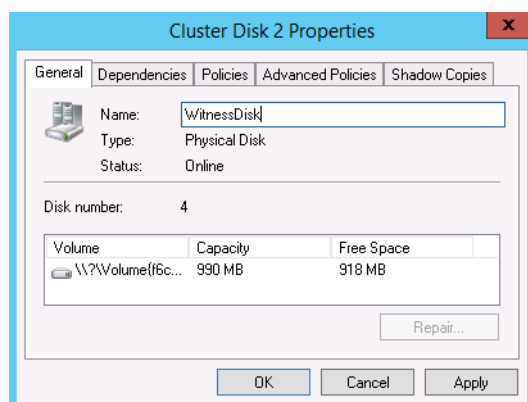
20. Right-click on the first disk that you want to be a Cluster Shared Volume. Select the Add to Cluster Shared Volumes from the drop-down menu.

21. Repeat for other disks that will be CSVs.

> ✎
> **Note** It is a good practice to ensure the disks are added in a sequence that is meaningful.   As disks are added, mount points for referencing the disks are created sequentially.
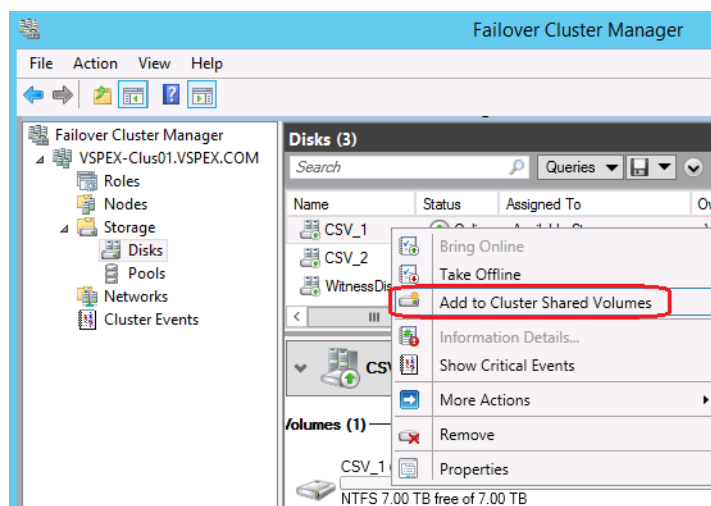
22. Select Networks and click on one of the networks. Select the Network Connections tab and ensure the networks are named the same on all nodes.

> ✎
> **Note** Same name is not required, but it greatly assists in troubleshooting.

23. Right-click and select Properties.
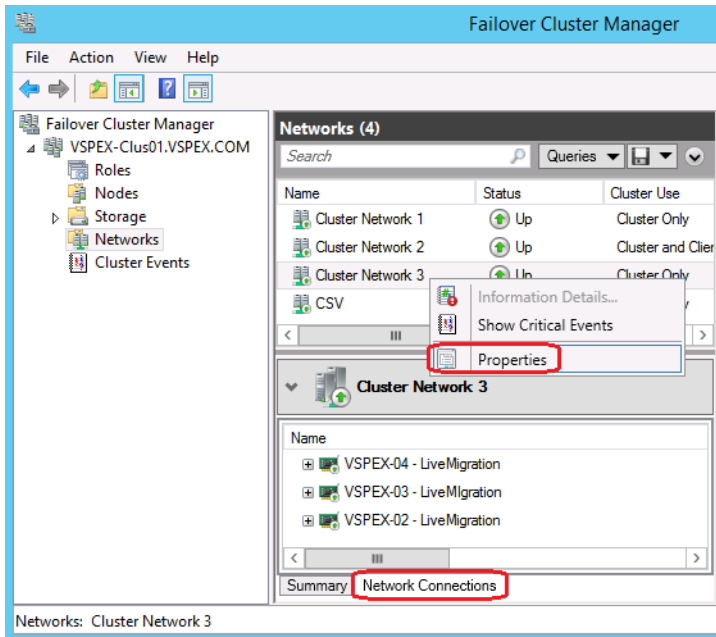
24. Rename the network names according to the names they are known to by the operating system.

25. Management network should have Allow cluster network communication on this network and Allow clients to connect through this network selected.
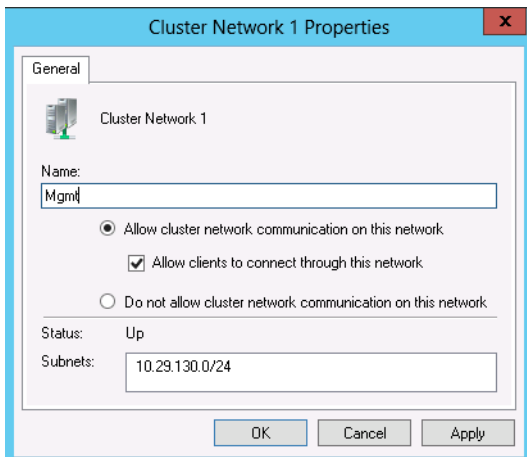
26. For the CSV network, ensure just the Allow cluster communication on this network is selected.
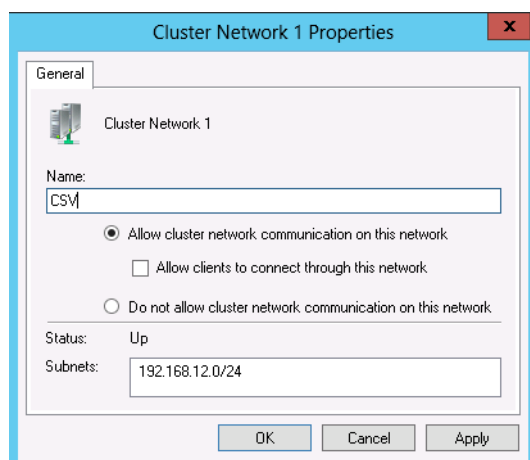
27. On the LiveMigration network, ensure just the Do not allow cluster network communication on this network is selected.
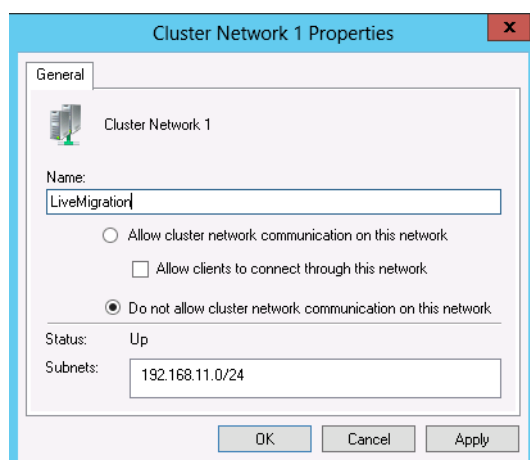
28. Click on Networks.

29. From the Actions menu on the right-hand side of the window, click on Live Migration Settings.
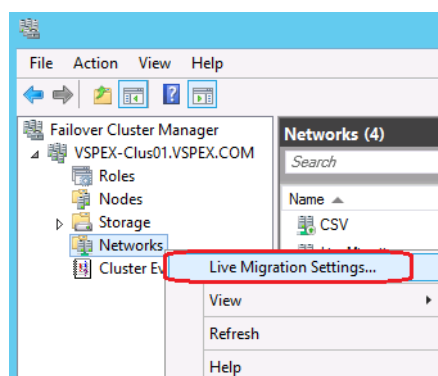
30. In the Live Migrations Settings window, ensure that the box by the LiveMigration and Mgmt networks are checked.
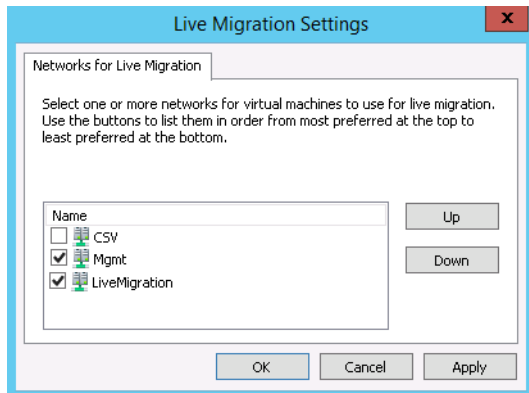
An alternate method to rename the generic network names and assign the proper function to each is to use a PowerShell script with these commands (modified for your environment).

```
(Get-ClusterNetwork -Cluster VSPEX-Clus01 | ? {$_.Address -like "10.29.130.*" }).Name = "Mgmt"
(Get-ClusterNetwork -Cluster VSPEX-Clus01 | ? {$_.Address -like "192.168.12.*" }).Name = "CSV"
(Get-ClusterNetwork -Cluster VSPEX-Clus01 | ? {$_.Address -like "192.168.11.*" }).Name = "LiveMigration"

(Get-ClusterNetwork -Cluster VSPEX-Clus01 -Name Mgmt).Role = 3
(Get-ClusterNetwork -Cluster VSPEX-Clus01 -Name CSV).Role = 1
(Get-ClusterNetwork -Cluster VSPEX-Clus01 -Name LiveMigration).Role = 0
```

The cluster is complete.

By default, Hyper-V will store the virtual hard drives for created virtual machines on the system drive. It is easy to set up Hyper-V to default to the Cluster Shared Volumes for storage. This is not an absolute requirement, but it does make management easier. A good practice is to have the same number of Cluster Shared Volumes as you have nodes in the Hyper-V cluster. Each node in the cluster would have a default storage location of one of the Cluster Shared Volumes.

1. Within the Hyper-V Management console, select Hyper-V Settings… from the Actions pane.
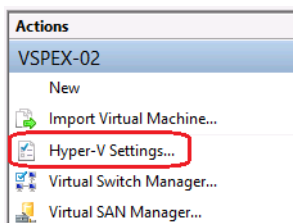
2. From the Server column, select Virtual Hard Disks.

3. In the right column, browse to the C:\ClusterStorage\Volumex (x is a sequence number) location and select it as the default.

4. Repeat for Virtual Machines.

5. Click OK to accept the change.

6. Repeat for each node of the cluster.
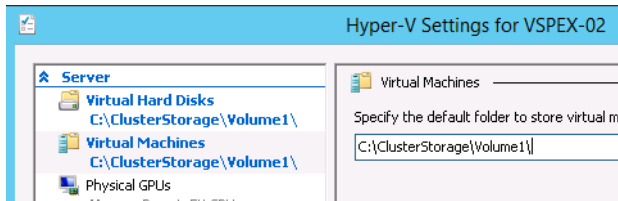
Sample PowerShell Scripts

These sample PowerShell scripts and input files are provided as examples to assist in in the rapid deployment of this VSPEX environment. They should be reviewed for compliance with customer policies and naming conventions. They were tested within the lab environment where this system was configured. Security in your environment may not allow these scripts to run in your environment. No warranty or support is implied by their inclusion within this document. They were included to provide you with a starting point if you want to automate some steps.

Cisco Scripts

UcsConfig.ps1

```
<#

 UcsConfig Version=0.1
 4-September-2013
 Created by Tim Cerling
 tcerling@cisco.com

 Execution string:  .\UcsConfig.ps1 -path <path> -validateOnly -toConsole
   <path> - location of input file UcsConfig.xml
   -validateOnly - only validate contents of UcsConfig.xml.  Does not update UCS
   -toConsole - output log file to console instead of log file

/#>

Param
(
    [Parameter(Mandatory=$false,Position=0)]
    [String]$path = (Get-Location),

    [Parameter(Mandatory=$false)]
    [Switch]$validateOnly = $false,

    [Parameter(Mandatory=$false)]
    [Switch]$toConsole = $false
)

#
# Function Definitions
```

```
#
# --------------------------------------------------------------------------

# Function to write log information to either log file or console
Function Write-Log ($content, $type)
{
    $n = (get-pscallstack).Length - 2
    $lineNum = ((get-pscallstack)[$n].Location -split " line ")[1]

    switch ($type)
    {
        Normal
        {
            If ($ToConsole)
            {
                Write-Host -ForegroundColor Green "$lineNum $(Get-Date -Format
"HH:mm:ss") - $content"
            }
            Else
            {
                Add-Content -Path $logFilePath -Value "$lineNum - $(Get-Date -Format
g): $content"
            }
        }
        Error
        {
            If ($ToConsole)
            {
                Write-Host -ForegroundColor Red -BackgroundColor Black "ERROR at
line $lineNum `n$content"
            }
            Else
            {
                Add-Content -Path $logFilePath -Value "ERROR at line $lineNum -
$(Get-Date -Format g): $content"
            }
        }
    }
}

#
# Regular expressions used for validation
#
# --------------------------------------------------------------------------

$validIPAddress = @"
^([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9]|[1-9][0-9]|1[0-9][0-9]|
2[0-4][0-9]|25[0-5])){3}$
"@

$validHex = @"
^([A-Fa-f0-9]{2})
"@
```

```
$validMAC = @"
^([0-9A-Fa-f]){2}(\:([0-9A-Fa-f]){2}){5}$
"@

$validWWN = @"
^([0-9A-Fa-f]){2}(\:([0-9A-Fa-f]){2}){7}$
"@

$validUUID = @"
^([A-Fa-f0-9]){4}(\-([A-Fa-f0-9]){12})
"@

$validSuffix = @"
^([A-Fa-f0-9:]{5})
"@

$validNumList = @"
^([0-9,-])
"@

$validDigit = @"
^([0-9])
"@

#
# Definition of constants
#
# ------------------------------------------------------------------------------

$startTime = Get-Date
$originalPath = Get-Location
$errTag = $False
$logFile = "UcsConfig.log"

$chassisDiscoveryOptions = @()          #Table of chassis discovery options
$chassisDiscoveryOptions +=, ("1-link")
$chassisDiscoveryOptions +=, ("2-link")
$chassisDiscoveryOptions +=, ("4-link")
$chassisDiscoveryOptions +=, ("8-link")
$chassisDiscoveryOptions +=, ("platform-max")

$diskPolicyOptions = @()                #Table of local disk policy options
$diskPolicyOptions +=, ('any-configuration')
$diskPolicyOptions +=, ('no-local-storage')
$diskPolicyOptions +=, ('no-raid')
$diskPolicyOptions +=, ('raid-striped')
$diskPolicyOptions +=, ('raid-mirrored')
$diskPolicyOptions +=, ('raid-mirrored-striped')
$diskPolicyOptions +=, ('raid-striped-parity')
$diskPolicyOptions +=, ('raid-striped-dual-parity')
```

```
$fiPortRoles = @()                          #Table of server roles for FI port
configuration
$fiPortRoles +=, ('server')
$fiPortRoles +=, ('uplink')
$fiPortRoles +=, ('appliance')
$fiPortRoles +=, ('fcoe')
$fiPortRoles +=, ('')

$poolType = @()                             #Table of types of pools
$poolType +=, ('MAC')
$poolType +=, ('UUID')
$poolType +=, ('WWNN')
$poolType +=, ('WWPN')

$objectTable = @()                          #Table of found objects in ucsConfig.xml
file
$objectTable +=, ('TimeZone', 0)
$objectTable +=, ('NTP', 0)
$objectTable +=, ('MgmtIP', 0)
$objectTable +=, ('CallHome', 0)
$objectTable +=, ('ChassisDiscovery', 0)
$objectTable +=, ('SubOrg', 0)
$objectTable +=, ('SANWWPN.SPAprimary', 0)
$objectTable +=, ('SANWWPN.SPAsecondary', 0)
$objectTable +=, ('SANWWPN.SPBprimary', 0)
$objectTable +=, ('SANWWPN.SPBsecondary', 0)
$objectTable +=, ('QoS', 0)
$objectTable +=, ('PowerPolicy', 0)
$objectTable +=, ('ScrubPolicy', 0)
$objectTable +=, ('MaintenancePolicy', 0)
$objectTable +=, ('DiskPolicy', 0)
$objectTable +=, ('BIOSPolicy', 0)
$objectTable +=, ('PlacementPolicy', 0)
$objectTable +=, ('FI', 0)
$objectTable +=, ('FCslot1', 0)
$objectTable +=, ('FCslot2', 0)
$objectTable +=, ('PC', 0)
$objectTable +=, ('Pools', 0)
$objectTable +=, ('VLANs', 0)
$objectTable +=, ('VNICtemplate', 0)
$objectTable +=, ('VHBAtemplate', 0)
$objectTable +=, ('BootPolicy', 0)
$objectTable +=, ('SPTemplate', 0)
$objectTable +=, ('ServiceProfile', 0)


##############################################################################
##############################################################################
#
# Start of Code
#
# ---------------------------------------------------------------------------
```

```
# Change to path entered on command line
If (Test-Path $path -PathType Container)
{
    Set-Location $path
}
    Else
    {
        $errTag = $True
        Write-Host "Invalid path" -ForegroundColor Red
    }

# Read input file
If (Test-Path "$path\ucsconfig.xml")
{
    try {$ucsConfig = [XML] (Get-Content "$path\UcsConfig.xml") }
    catch {$errTag = $True; Write-Host "Invalid UcsConfig.xml" -ForegroundColor Red}
}
    Else
    {
        $errTag = $True
        Write-Host "Missing UcsConfig.xml" -ForegroundColor Red
    }

# Create a log file in the same directory from which the script is running
If (!$errTag)
{
    If (!$toConsole)
    {
        $localPath = Split-Path (Resolve-Path $MyInvocation.MyCommand.Path)
        $logFilePath = Join-Path $localPath $logFile
        If (Test-Path($logFilePath))
        {
            Write-Host "Deleting existing log file"
            Remove-Item $logFilePath
        }
        Write-Host "Creating new log file $logfilePath"
        $trash = New-Item -Path $localPath -Name $logFile -ItemType "file"
    }
}

# Import required modules
if ((Get-Module |where {$_.Name -ilike "CiscoUcsPS"}).Name -ine "CiscoUcsPS")
    {
     Write-Host "Loading Module: Cisco UCS PowerTool Module"
     Import-Module CiscoUcsPs
    }

$trash = set-ucspowertoolconfiguration -supportmultipledefaultucs $false

# Test whether to continue processing
If ($errTag)
{
    Set-Location $originalPath
```

```
    $endTime = Get-Date
    $elapsedTime = New-TimeSpan $startTime $endTime
    Write-Host -ForegroundColor Yellow -BackgroundColor Black
"`n`n-----------------------------------------------------`n"
    Write-Log "Elapsed time:
$($elapsedTime.Hours):$($elapsedTime.Minutes):$($elapsedTime.Seconds)"
    Write-Log "End of processing.`n"
    Exit
}


##############################################################################
##############################################################################
# ---------------------------------------------------------------------------
#
# Start validating the UcsConfig.xml file
#
# ---------------------------------------------------------------------------
##############################################################################

$error.Clear()
Write-Log "Validating contents of $path\UcsConfig.xml"

# Validate UCSM IP address
$tmp1 = $ucsConfig.VSPEX.UCSMIP.trim()
$tmp = Test-Connection $tmp1 -Count 1 -Quiet
If ($tmp)
{
    Write-Log "UCSM IP address $tmp1 is reachable." "Normal"
}
    Else
    {
        Write-Log "UCSM IP address $tmp1 is unreachable." "Error"
        $errTag = $True
    }

# Validate Management IP settings
$ucsConfig.VSPEX.MgmtIP | ForEach-Object {$_.Pool} | ForEach-Object {
    $tmp1 = $_
    If ($tmp1 -ne $null)
    {
        For ($i=0; $i -lt $objectTable.length; $i++)
        {
            $obj = $objectTable[$i]
            If ($obj[0] -eq 'MgmtIP') {$obj[1] = 1; Break}
        }
        $tmp2 = $_.Name.trim()
        $tmp1 | ForEach-Object {
            $tmp = $False
            $tmp3 = $_.Order.trim()
            $tmp4 = $_.Start.trim()
            $tmp5 = $_.End.trim()
            $tmp6 = $_.Gateway.trim()
            $tmp7 = $_.PrimaryDNS.trim()
```

```
          $tmp8 = $_.SecondaryDNS.trim()
          If (!($tmp3 -eq 'default' -or $tmp3 -eq 'sequential'))
          {
              Write-Log "Invalid management pool assignment order - Name=$tmp2
Start=$tmp3" "Error"
              $tmp = $True
          }
          If (!($tmp4 -match $validIPaddress))
          {
               Write-Log "Invalid management pool start IP address - Name=$tmp2
Start=$tmp4" "Error"
              $tmp = $True
          }
          If (!($tmp5 -match $validIPaddress))
          {
               Write-Log "Invalid management pool end IP address - Name=$tmp2
Start=$tmp5" "Error"
                $tmp = $True
          }
          If ($tmp4 -ge $tmp5)
          {
               Write-Log "Management pool end IP address must be greater than start
- Name=$tmp2 Start=$tmp4 End=$tmp5" "Error"
                $tmp = $True
          }
          If (!($tmp6 -match $validIPaddress))
          {
               Write-Log "Invalid management pool gateway IP address - Name=$tmp2
Start=$tmp6" "Error"
              $tmp = $True
          }
          If ($tmp7 -ne '0.0.0.0')
          {
               If (!($tmp7 -match $validIPaddress))
               {
                   Write-Log "Invalid management pool primary DNS IP address -
Name=$tmp2 Primary=$tmp7" "Error"
                   $tmp = $True
               }
          }
          If ($tmp8 -ne '0.0.0.0')
          {
               If (!($tmp8 -match $validIPaddress))
               {
                   Write-Log "Invalid management pool secondary DNS IP address -
Name=$tmp2 Secondary=$tmp8" "Error"
                   $tmp = $True
               }
          }
          If (!$tmp)
          {
               Write-Log "Management IP pool - Name=$tmp2 Order=$tmp3 Start=$tmp4
End=$tmp5 G/W=$tmp6 Primary=$tmp7 Secondary=$tmp8" "Normal"
```

```
            }
            Else
            {
                Write-Log "Invalid management IP pool - Name=$tmp2 Order=$tmp3
Start=$tmp4 End=$tmp5 G/W=$tmp6 Primary=$tmp7 Secondary=$tmp8" "Error"
                $errTag = $True
            }
        }
    }
}

# Validate Chassis discovery setting
$tmp1 = $ucsConfig.VSPEX.ChassisDiscovery
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'ChassisDiscovery') {$obj[1] = 1; Break}
    }
    $tmp1 = $tmp1.trim()
    For ($i=0; $i -lt $chassisDiscoveryOptions.length; $i++)
    {
        If ($tmp1 -eq $chassisDiscoveryOptions[$i])
        {
            Write-Log "Valid chassis discovery - $tmp1" "Normal"
            Break
        }
    }
    If ($i -eq $chassisDiscoveryOptions.length)
    {
        Write-Log "Invalid chassis discovery option - $tmp1" "Error"
        $errTag = $True
    }
}

# Validate SAN WWPN values
$tmp1 = $ucsConfig.VSPEX.SANWWPN.SPAprimary
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'SANWWPN.SPAprimary') {$obj[1] = 1; Break}
    }
    $tmp1 = $tmp1.trim()
    If ($tmp1 -match $validWWN)
    {
        Write-Log "Valid Port-A Primary WWPN  - $tmp1" "Normal"
        $sanSPAprimary = $tmp1
    }
    Else
    {
```

```
        Write-Log "Invalid Port-A Primary WWPN   - $tmp1" "Error"
        $errTag = $True
    }
}

$tmp1 = $ucsConfig.VSPEX.SANWWPN.SPAsecondary
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'SANWWPN.SPAsecondary') {$obj[1] = 1; Break}
    }
    $tmp1 = $tmp1.trim()
    If ($tmp1 -match $validWWN)
    {
        Write-Log "Valid Port-A Secondary WWPN - $tmp1" "Normal"
        $sanSPAsecondary = $tmp1
    }
    Else
    {
        Write-Log "Invalid Port-A Secondary WWPN - $tmp1" "Error"
        $errTag = $True
    }
}

$tmp1 = $ucsConfig.VSPEX.SANWWPN.SPBprimary
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'SANWWPN.SPBprimary') {$obj[1] = 1; Break}
    }
    $tmp1 = $tmp1.trim()
    If ($tmp1 -match $validWWN)
    {
        Write-Log "Valid Port-B Primary WWPN   - $tmp1" "Normal"
        $sanSPBprimary = $tmp1
    }
    Else
    {
        Write-Log "Invalid Port-B Primary WWPN   - $tmp1" "Error"
        $errTag = $True
    }
}

$tmp1 = $ucsConfig.VSPEX.SANWWPN.SPBsecondary
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
```

```powershell
        If ($obj[0] -eq 'SANWWPN.SPBsecondary') {$obj[1] = 1; Break}
    }
    $tmp1 = $tmp1.trim()
    If ($tmp1 -match $validWWN)
    {
        Write-Log "Valid Port-B Secondary WWPN - $tmp1" "Normal"
        $sanSPBsecondary = $tmp1
    }
    Else
    {
        Write-Log "Invalid Port-B Secondary WWPN - $tmp1" "Error"
        $errTag = $True
    }
}

# Validate Power Control Policies
$tmp1 = $ucsConfig.VSPEX.PowerPolicy
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'PowerPolicy') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.PowerPolicy | ForEach-Object {$_.Var} | ForEach-Object {
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Priority.trim()
        $tmp3 = $_.Priority.trim()
        If ($tmp3 -eq 'no-cap')
        {
            Write-Log "Valid Power Control Policy on entry - Name=$tmp1
Priority=$tmp3" "Normal"
        }
        Else
        {
            If ([int]$tmp2 -ge 1 -and [int]$tmp2 -le 10)
            {
                Write-Log "Valid Power Control Policy on entry - Name=$tmp1
Priority=$tmp2" "Normal"
            }
            Else
            {
                Write-Log "Invalid Power Control Policy on entry - Name=$tmp1
Priority=$tmp3" "Error"
                $errTag = $True
            }
        }
    }
}

# Validate Scrub Policies
$tmp1 = $ucsConfig.VSPEX.ScrubPolicy
If ($tmp1 -ne $null)
```

```
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'ScrubPolicy') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.ScrubPolicy | ForEach-Object {$_.Var} | ForEach-Object {
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Descr
        $tmp3 = $_.DiskScrub.trim()
        $tmp3 = $tmp3.tolower()
        $tmp4 = $_.BiosScrub.trim()
        $tmp4 = $tmp4.tolower()

        If (($tmp3 -eq 'yes' -or $tmp3 -eq 'no') -and ($tmp4 -eq 'yes' -or $tmp4 -eq
'no'))
        {
            Write-Log "Valid Scrub Policy on entry - Name=$tmp1 Descr=$tmp2
Disk=$tmp3 Bios=$tmp4" "Normal"
        }
        Else
        {
            Write-Log "Invalid Scrub Policy on entry - Name=$tmp1 Descr=$tmp2
Disk=$tmp3 Bios=$tmp4" "Error"
            $errTag = $True
        }
    }
}

# Validate Maintenance Policies
$tmp1 = $ucsConfig.VSPEX.MaintenancePolicy
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'MaintenancePolicy') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.MaintenancePolicy | ForEach-Object {$_.Var} | ForEach-Object {
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Descr.trim()
        $tmp3 = $_.Policy.trim()
        $tmp3 = $tmp3.tolower()
        If ($tmp3 -eq 'immediate' -or $tmp3 -eq 'timer-automatic' -or $tmp3 -eq
'user-ack')
        {
            Write-Log "Valid Maintenance Policy on entry - Name=$tmp1 Descr=$tmp2
Policy=$tmp3" "Normal"
        }
        Else
        {
            Write-Log "Invalid Maintenance Policy on entry - Name=$tmp1 Descr=$tmp2
Policy=$tmp3" "Error"
```

```
            $errTag = $True
        }
    }
}


# Validate Local Disk Policies
$tmp1 = $ucsConfig.VSPEX.DiskPolicy
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'DiskPolicy') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.DiskPolicy | ForEach-Object {$_.Var} | ForEach-Object {
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Mode.trim()
        $tmp3 = $_.Protect.trim()
        For ($i=0; $i -lt $diskPolicyOptions.length; $i++)
        {
            If (($tmp2 -eq $diskPolicyOptions[$i]) -and ($tmp3 -eq 'yes' -or $tmp3
-eq 'no'))
            {
                Write-Log "Valid local disk policy - Name=$tmp1 Mode=$tmp2
Protect=$tmp3" "Normal"
                Break
            }
        }
        If ($i -eq $diskPolicyOptions.Length)
        {
            Write-Log "Invalid local disk policy - Name=$tmp1 Mode=$tmp2
Protect=$tmp3" "Error"
            $errTag = $True
        }
    }
}


# Validate FI port definitions  - NOTE: this section does not handle FC ports.
$tmp1 = $ucsConfig.VSPEX.FI.Var
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'FI') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.FI | ForEach-Object {$_.Var} | ForEach-Object {
        $tmp1 = $_.SlotID.trim()
        $tmp2 = $_.PortID.trim()
        $tmp3 = $_.Role.trim()
        If ($tmp1 -eq '1')
        {
            For ($i=0; $i -lt $fiPortRoles.length; $i++)
```

```
              {
                    If ((([int]$tmp2 -ge 1 -and [int]$tmp2 -le 32) -and ($tmp3 -eq
$fiPortRoles[$i]))
                    {
                        If ($tmp3 -ne '')
                        {
                             Write-Log "Valid FI port role definition - SlotID=$tmp1
PortID=$tmp2 Role=$tmp3" "Normal"
                        }
                        Break
                    }
                }
            If ($i -eq $fiPortRoles.length)
            {
                    Write-Log "Invalid FI port role definition - SlotID=$tmp1
PortID=$tmp2 Role=$tmp3" "Error"
                    $errTag = $True
            }
        }
        If ($tmp1 -eq '2')
        {
            For ($i=0; $i -lt $fiPortRoles.length; $i++)
            {
                    If ((([int]$tmp2 -ge 1 -and [int]$tmp2 -le 16) -and ($tmp3 -eq
$fiPortRoles[$i]))
                    {
                        If ($tmp3 -ne '')
                        {
                             Write-Log "Valid FI port role definition - SlotID=$tmp1
PortID=$tmp2 Role=$tmp3" "Normal"
                        }
                        Break
                    }
                }
            If ($i -eq $fiPortRoles.length)
            {
                    Write-Log "Invalid FI port role definition - SlotID=$tmp1
PortID=$tmp2 Role=$tmp3" "Error"
                    $errTag = $True
            }
        }
        If ($tmp1 -ne '1' -and $tmp1 -ne '2')
        {
            Write-Log "Invalid Slot number - $tmp1" "Error"
            $errTag = $true
        }
    }
}

# Validate FC port definitions  - NOTE: this section handles FC ports.
$tmp1 = $ucsConfig.VSPEX.FCslot1.PortID
If ($tmp1 -ne $null)
{
```

```
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'FCslot1') {$obj[1] = 1; Break}
    }
    $tmp1 = $tmp1.trim()
    If ($tmp1 -ne '')
    {
        If (($tmp1 % 2 -eq 0) -or ($tmp1 -ge 32))
        {
            Write-Log "Invalid Fixed Module FC Port - must not be an even integer or
> 31 - $tmp1" "Error"
            $errTag = $true
        }
        Else
        {
            Write-Log "Valid Fixed Module FC port starting at - Port=$tmp1" "Normal"
        }
    }
}

$tmp1 = $ucsConfig.VSPEX.FCslot2.PortID
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'FCslot2') {$obj[1] = 1; Break}
    }
    $tmp1 = $tmp1.trim()
    If ($tmp1 -ne '')
    {
        If (($tmp1 % 2 -eq 0) -or ($tmp1 -ge 16))
        {
            Write-Log "Invalid Expansion Module FC Port - must not be an even
integer or > 15 - $tmp1" "Error"
            $errTag = $true
        }
        Else
        {
            Write-Log "Valid Expansion Module FC port starting at - Port=$tmp1"
"Normal"
        }
    }
}

# Validate Port Channel configuration
$tmp = $False
$tmp1 = $ucsConfig.VSPEX.PC.AName
$tmp2 = $ucsConfig.VSPEX.PC.BName
$tmp3 = $ucsConfig.VSPEX.PC.APortID
$tmp4 = $ucsConfig.VSPEX.PC.BPortID
$tmp5 = $ucsConfig.VSPEX.PC.Slot
```

```
$tmp6 = $ucsConfig.VSPEX.PC.Port1
$tmp7 = $ucsConfig.VSPEX.PC.Port2

If ($tmp1 -ne $null -and $tmp2 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'PC') {$obj[1] = 1; Break}
    }
    $tmp1 = $tmp1.trim()
    $tmp2 = $tmp2.trim()
    $tmp3 = $tmp3.trim()
    $tmp4 = $tmp4.trim()
    $tmp5 = $tmp5.trim()
    $tmp6 = $tmp6.trim()
    $tmp7 = $tmp7.trim()
    If ($tmp1 -eq $tmp2)
    {
        Write-Log "Port Channel names must be different on each fabric -
A-Name=$tmp1 B-Name=$tmp2" "Error"
        $tmp = $True
    }

    If ($tmp3 -eq $tmp4)
    {
        Write-Log "Port Channel PortID must be different on each fabric -
A-PortID=$tmp3 B-PortID$tmp4" "Error"
        $tmp = $True
    }

    If ($tmp5 -ne '1' -and $tmp5 -ne '2')
    {
        Write-Log "Port Channel Slot must be '1' or '2' - SlitID=$tmp5" "Error"
        $tmp = $True
    }
    Else
    {
        Switch ($tmp5)
        {
            1
            {
                If (([int]$tmp6 -lt 1 -or [int]$tmp6 -gt 32) -or ([int]$tmp7 -lt 1
-or [int]$tmp7 -gt 32) -or ($tmp6 -eq $tmp7))
                {
                    Write-Log "Invalid port number for Slot 1 - Port1=$tmp6
Port2=$tmp7" "Error"
                    $tmp = $True
                }
            }
            2
            {
```

```
                If (([int]$tmp6 -lt 1 -or [int]$tmp6 -gt 16) -or ([int]$tmp7 -lt 1
-or [int]$tmp7 -gt 16) -or ($tmp6 -eq $tmp7))
                {
                    Write-Log "Invalid port number for Slot 2 - Port1=$tmp6
Port2=$tmp7" "Error"
                    $tmp = $True
                }
            }
        }
    }
}

If (!$tmp)
{
    Write-Log "Fabric A VPC - A-Name=$tmp1 A-PortID=$tmp3 Slot=$tmp5 Port1=$tmp6
Port2=$tmp7" "Normal"
    Write-Log "Fabric B VPC - B-Name=$tmp2 B-PortID=$tmp4 Slot=$tmp5 Port1=$tmp6
Port2=$tmp7" "Normal"
}
Else
{
    Write-Log "Invalid fabric A VPC - A-Name=$tmp1 A-PortID=$tmp3 Slot=$tmp5
Port1=$tmp6 Port2=$tmp7" "Normal"
    Write-Log "Invalid fabric B VPC - B-Name=$tmp2 B-PortID=$tmp4 Slot=$tmp5
Port1=$tmp6 Port2=$tmp7" "Normal"
    $errTag = $True
}

# Validate the various types of pools
$tmp1 = $ucsConfig.VSPEX.Pools
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'Pools') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.Pools | ForEach-Object {$_.Var} | ForEach-Object {
        $tmp1 = $_.Type.trim()
        $tmp2 = $_.Name.trim()
        $tmp3 = $_.From.trim()
        $tmp4 = $_.To.trim()
        If ($tmp2 -eq 'default') {$tmp2 = $tmp2.tolower()}
        For ($i=0; $i -lt $poolType.length; $i++)
        {
            If ($tmp1 -eq $poolType[$i])
            {
                Switch ($tmp1)
                {
                    MAC
                    {
                        If (($tmp3 -match $validMAC -and $tmp4 -match $validMAC)
-and ($tmp3 -lt $tmp4))
```

```
                            {
                                  Write-Log "Valid MAC pool - Name=$tmp2 From=$tmp3
To=$tmp4" "Normal"
                            }
                            Else
                            {
                                  Write-Log "Invalid MAC pool - Name=$tmp2 From=$tmp3
To=$tmp4" "Error"

                                  $errTag = $True
                            }
                      }
                      UUID
                      {
                            If (($tmp3 -match $validUUID -and $tmp4 -match $validUUID)
-and ($tmp3 -lt $tmp4))
                            {
                                  Write-Log "Valid UUID pool - Name=$tmp2 From=$tmp3
To=$tmp4" "Normal"
                            }
                            Else
                            {
                                  Write-Log "Invalid UUID pool - Name=$tmp2 From=$tmp3
To=$tmp4" "Error"

                                  $errTag = $True
                            }
                      }
                      WWNN
                      {
                            If (($tmp3 -match $validWWN -and $tmp4 -match $validWWN)
-and ($tmp3 -lt $tmp4))
                            {
                                  Write-Log "Valid WWNN pool - Name=$tmp2 From=$tmp3
To=$tmp4" "Normal"
                            }
                            Else
                            {
                                  Write-Log "Invalid WWNN pool - Name=$tmp2 From=$tmp3
To=$tmp4" "Error"

                                  $errTag = $True
                            }
                      }
                      WWPN
                      {
                            If (($tmp3 -match $validWWPN -and $tmp4 -match $validWWPN)
-and ($tmp3 -lt $tmp4))
                            {
                                  Write-Log "Valid WWPN pool - Name=$tmp2 From=$tmp3
To=$tmp4" "Normal"
                            }
                            Else
                            {
                                  Write-Log "Invalid WWPN pool - Name=$tmp2 From=$tmp3
To=$tmp4" "Error"
```

```
                                $errTag = $True
                        }
                }
            }
            Break
            }
        }
        If ($i -eq $poolType.length)
        {
            Write-Log "Invalid pool type - Type=$tmp1 Name=$tmp2 From=$tmp3
To=$tmp4" "Error"
            $errTag = $True
        }
    }
}

#Validate VLAN definitions
$tmp1 = $ucsConfig.VSPEX.VLANs
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'VLANs') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.VLANs | ForEach-Object {$_.Var} | ForEach-Object {
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Fabric.trim()
        $tmp3 = $_.ATag.trim()
        $tmp4 = $_.BTag.trim()
        $tmp5 = $_.DefaultNet.trim()
        If ($tmp2 -eq 'common' -or $tmp2 -eq 'diff' -or $tmp2 -eq 'faba' -or $tmp2
-eq 'fabb')
        {
            $tmp = $False
            If (!($tmp5 -eq 'yes' -or $tmp5 -eq 'no'))
            {
                Write-Log "Invalid default net value - Name=$tmp1 DefaultNet=$tmp5"
"Error"
                $tmp = $True
            }
            Switch ($tmp2)
            {
            common
            {
                If (!(($tmp3 -eq '') -or ([int]$tmp3 -ge 1 -and [int]$tmp3 -le
4095)))
                {
                    Write-Log "Invalid VLAN tag value - Name=$tmp1 Fabric=$tmp2
ATag=$tmp3" "Error"
                    $tmp = $true
                }
            }
```

```
diff
{
    If (!(($tmp3 -eq '') -or ([int]$tmp3 -ge 1 -and [int]$tmp3 -le
4095)))
    {
        Write-Log "Invalid VLAN tag value - Name=$tmp1 Fabric=$tmp2
ATag=$tmp3" "Error"
        $tmp = $True
    }
    If (!(($tmp4 -eq '') -or ([int]$tmp4 -ge 1 -and [int]$tmp4 -le
4095)))
    {
        Write-Log "Invalid VLAN tag value - Name=$tmp1 Fabric=$tmp2
BTag=$tmp4" "Error"
        $tmp = $true
    }
}
faba
{
    If (!(($tmp3 -eq '') -or ([int]$tmp3 -ge 1 -and [int]$tmp3 -le
4095)))
    {
        Write-Log "Invalid VLAN tag value - Name=$tmp1 Fabric=$tmp2
ATag=$tmp3" "Error"
        $tmp = $True
    }
}
fabb
{
    If (!(($tmp4 -eq '') -or ([int]$tmp4 -ge 1 -and [int]$tmp4 -le
4095)))
    {
        Write-Log "Invalid VLAN tag value - Name=$tmp1 Fabric=$tmp2
BTag=$tmp4" "Error"
        $tmp = $true
    }
}
}
If (!$tmp)
{
    Write-Log "Valid VLAN definition - Name=$tmp1 Fabric=$tmp2
ATag=$tmp3 BTag=$tmp4 DefaultNet=$tmp5" "Normal"
}
Else
{
    Write-Log "Invalid VLAN definition - Name=$tmp1 Fabric=$tmp2
ATag=$tmp3 BTag=$tmp4 DefaultNet=$tmp5" "Error"
    $errTag = $True
}
}
Else
{
```

```
                Write-Log "Invalid fabric configuration - Name=$tmp1 Fabric=$tmp2"
"Error"
                $errTag = $True
            }
        }
}


# Validate VNIC template information
$tmp1 = $ucsConfig.VSPEX.VNICTemplate
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'VNICTemplate') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.VNICTemplate | ForEach-Object {$_.Var} | ForEach-Object {
        $tmp = $False
        $tmp1 = $_.Name
        $tmp2 = $_.MTU.trim()
        If (!($tmp2 -eq '1500' -or $tmp2 -eq '9000'))
        {
            Write-Log "Invalid MTU - should be 1500 or 9000 - Name=$tmp1 MTU=$tmp2"
"Error"
            $tmp = $true
        }
        $tmp3 = $_.Fabric.trim()
        $tmp3 = $tmp3.toupper()
        If (!($tmp3 -eq 'A-B' -or $tmp3 -eq 'B-A'))
        {
            Write-Log "Invalid fabric - should be A-B or B-A - Name=$tmp1
Fabric=$tmp3" "Error"
            $tmp = $true
        }
        $tmp4 = $_.Type.trim()
        $tmp4 = $tmp4.tolower()
        If (!($tmp4 -eq 'updating-template' -or $tmp4 -eq 'initial-template'))
        {
            Write-Log "Invalid template type - Name=$tmp1 Type=$tmp4" "Error"
            $tmp = $true
        }
        $tmp5 = $_.Native.trim()
        $tmp5 = $tmp5.tolower()
        If (!($tmp5 -eq 'no' -or $tmp5 -eq 'yes'))
        {
            Write-Log "Invalid native mode - must be yes or no - Native=$tmp5"
"Error"
            $tmp = $true
        }
        If (!$tmp)
        {
            Write-Log "Valid VNIC template - Name=$tmp1 MTU=$tmp2 Fabric=$tmp3
Type=$tmp4 Native=$tmp5" "Normal"
```

```
        }
        Else
        {
                Write-Log "Invalid VNIC template - Name=$tmp1 MTU=$tmp2 Fabric=$tmp3
Type=$tmp4 Native=$tmp5" "Error"
                $errTag = $True
        }
    }
}

# Validate virtual HBA template
#     <Var Name='FabChn-A' Descr='Fabric A vHBA' Fabric='A' VSAN='default'
Type='updating-template' WWNpool='Pool-AF' Qos='' />
$tmp1 = $ucsConfig.VSPEX.VHBATemplate
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'VHBATemplate') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.VHBATemplate | ForEach-Object {$_.Var} | ForEach-Object {
        $tmp = $False
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Fabric.trim()
        $tmp3 = $_.Type.trim()
        If ($tmp1 -eq '')
        {
                Write-Log "Missing name for vHBA template - Fabric=$tmp2 Type=$tmp3"
"Error"
                $tmp = $True
        }
        If (!($tmp2 -eq 'A' -or $tmp2 -eq 'B'))
        {
                Write-Log "Invalid Fabric for vHBA template - Name=$tmp1 Fabric=$tmp2"
"Error"
                $tmp = $True
        }
        If (!($tmp3 -eq 'initial-template' -or $tmp3 -eq 'updating-template'))
        {
                Write-Log "Invalid template type for vHBA template - Name=$tmp1
Type=$tmp3" "Error"
                $tmp = $True
        }
        If (!$tmp)
        {
                Write-Log "Valid vHBA template - Name=$tmp1 Fabric=$tmp2 Type=$tmp3"
"Normal"
        }
        Else
        {
                Write-Log "Invalid vHBA template - Name=$tmp1 Fabric=$tmp2 Type=$tmp3"
"Normal"
```

```
            $errTag = $True
        }
    }
}


# Validate boot policies
$tmp1 = $ucsConfig.VSPEX.BootPolicy
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'BootPolicy') {$obj[1] = 1; Break}
    }
    $ucsConfig.VSPEX.BootPolicy | ForEach-Object {$_.PolicyName} | ForEach-Object {
        $tmp1 = $_
        $tmp2 = $_.Name.trim()
        $tmp1 | ForEach-Object {$_.Var} | ForEach-Object {
            $tmp = $False
            $tmp3 = $_.Type.trim()
            $tmp4 = $_.Device1.trim()
            $tmp5 = $_.Device2.trim()
            $tmp6 = $_.PrimaryFabric
            If (!($tmp3 -eq 'Local' -or $tmp3 -eq 'VNIC' -or $tmp3 -eq 'VHBA'))
            {
                Write-Log "Device type must be Local, VNIC, or VHBA - Name=$tmp2
Type=$tmp3" "Error"
                $tmp = $True
            }
            If ($tmp3 -eq 'Local')
            {
                If (!($tmp4 -eq 'localdisk' -or $tmp4 -eq 'cdrom' -or $tmp4 -eq
'floppy'))
                {
                    Write-Log "Local device must be localdisk, cdrom, or floppy -
Name=$tmp2 Device1=$tmp4" "Error"
                    $tmp = $True
                }
            }
            If ($tmp5 -ne '')
            {
                If (!($tmp6 -eq 'A' -or $tmp6 -eq 'B'))
                {
                    Write-Log "Primary Fabric must be A or B - Name=$tmp2
Device1=$tmp4" "Error"
                    $tmp = $True
                }
            }
            If (!$tmp)
            {
                Write-Log "Valid boot policy - Name=$tmp2 Type=$tmp3 Device1=$tmp4
Device2=$tmp5" "Normal"
            }
```

```
            Else
            {
                 Write-Log "Invalid boot policy - Name=$tmp2 Type=$tmp3 Device1=$tmp4
Device2=$tmp5" "Error"
                 $errTag = $True
            }
        }
    }
}

# TimeZone - just check for presence - no validation - accept what is there
$tmp1 = $ucsConfig.VSPEX.TimeZone
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'TimeZone') {$obj[1] = 1; Break}
    }
}

# NTP - just check for presence - no validation - accept what is there
$tmp1 = $ucsConfig.VSPEX.NTP.Var
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'NTP') {$obj[1] = 1; Break}
    }
}

# CallHome - just check for presence - no validation - accept what is there
$tmp1 = $ucsConfig.VSPEX.CallHome.InUse
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'CallHome') {$obj[1] = 1; Break}
    }
}

# SubOrg - just check for presence - no validation - accept what is there
$tmp1 = $ucsConfig.VSPEX.SubOrg.Var
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'SubOrg') {$obj[1] = 1; Break}
    }
}
```

```
# QoS - just check for presence - no validation - accept what is there
$tmp1 = $ucsConfig.VSPEX.QoS
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'QoS') {$obj[1] = 1; Break}
    }
}

# BIOSPolicy - just check for presence - no validation - accept what is there
$tmp1 = $ucsConfig.VSPEX.BIOSPolicy.Var
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'BIOSPolicy') {$obj[1] = 1; Break}
    }
}

# PlacementPolicy - just check for presence - no validation - accept what is there
$tmp1 = $ucsConfig.VSPEX.PlacementPolicy.Var
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'PlacementPolicy') {$obj[1] = 1; Break}
    }
}

# SPTemplate - just check for presence - no validation - accept what is there
$tmp1 = $ucsConfig.VSPEX.SPTemplate.Template
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'SPTemplate') {$obj[1] = 1; Break}
    }
}

# ServiceProfile - just check for presence - no validation - accept what is there
$tmp1 = $ucsConfig.VSPEX.ServiceProfile.Var
If ($tmp1 -ne $null)
{
    For ($i=0; $i -lt $objectTable.length; $i++)
    {
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'ServiceProfile') {$obj[1] = 1; Break}
```

```
    }
}

# Display to operator which objects do not have any defined values
$tmp = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[1] -eq '0')
    {
        $tmp1 = $obj[0]
        Write-Host "No defined values for object $tmp1"
        $tmp = $true
    }
}

# Test for Validation errors or Validate only run.  If found, wrap up and shut down.
If ($errTag -or $validateOnly)
{
    If ($errTag)
    {
        Write-Host -ForegroundColor Red -BackgroundColor Black "`n`nProcessing
stopped due to detected errors"
    }
    Set-Location $originalPath
    $endTime = Get-Date
    $elapsedTime = New-TimeSpan $startTime $endTime
    Write-Host -ForegroundColor Yellow -BackgroundColor Black
"`n`n--------------------------------------------------------`n"
    Write-Log "Elapsed time:
$($elapsedTime.Hours):$($elapsedTime.Minutes):$($elapsedTime.Seconds)" "Normal"
    Write-Log "End of processing." "Normal"
    Exit
}

# Missing values is not necessarily an error.  Ask if operator wishes to continue.
If ($tmp)
{
    Write-Host "`nYou have some missing values.  Do you wish to continue without
them?"
    $tmp1 = Read-Host "You must enter 'YES' (no quotes) to continue"
    If ($tmp1 -ne 'YES') {Exit}
}

##############################################################################
##############################################################################
# ----------------------------------------------------------------------------
#
# Configure UCS with the contents of UcsConfig.XML
#
# ----------------------------------------------------------------------------
##############################################################################
```

```
# Login to UCSM IP address
$ucsmIP = $UcsConfig.VSPEX.UCSMIP.trim()
$Error.Clear()
Write-Host -BackgroundColor Black -ForegroundColor White "`n`n    Enter proper
credentials to access UCSM`n"
$ucsCreds = Get-Credential
$ucsHandle = Connect-Ucs $ucsmIP $ucsCreds
$ucsDomain = $ucsHandle.Ucs
If ($error.length -lt 1)
{
    Write-Log "Successful login to UCS domain $ucsDomain" "Normal"
}
Else
{
    Write-Log "Invalid login to $ucsmIP" "Error"
    exit
}
$orgRoot = Get-UcsOrg -Level root

# Set timezone
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'TimeZone' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $error.Clear()
    $tmp1 = $ucsConfig.VSPEX.TimeZone.trimend(" ")
    $trash = Get-UcsTimezone | Set-UcsTimezone -AdminState "enabled" -Timezone $tmp1
-Force
    If ($error.length -lt 1)
    {
        Write-Log "Set timezone to $tmp1" "Normal"
    }
    Else
    {
        Write-Log "ERROR setting timezone to $tmp1" "Error"
        $errTag = $True
    }
}

# Set NTP servers
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'NTP' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.NTP | ForEach-Object {$_.Var} | ForEach-Object {
```

```
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $trash = Add-UcsNtpServer -Name $tmp1 -ModifyPresent
        If ($error.length -lt 1)
        {
            Write-Log "Set NTP server to $tmp1" "Normal"
        }
        Else
        {
            Write-Log "ERROR setting NTP server to $tmp1" "Error"
            $errTag = $True
        }
    }
}

# Set Management IP Pool values
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'MgmtIP' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.MgmtIP | ForEach-Object {$_.Pool} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_
        $tmp2 = $_.Name.trim()
        $tmp3 = $_.Descr
        $tmp1 | ForEach-Object {
            $tmp = $False
            $tmp4 = $_.Order.trim()
            $tmp5 = $_.Start.trim()
            $tmp6 = $_.End.trim()
            $tmp7 = $_.Gateway.trim()
            $tmp8 = $_.PrimaryDNS.trim()
            $tmp9 = $_.SecondaryDNS.trim()

            Start-UcsTransaction
              $mo = $orgRoot | Add-UcsIpPool -Name $tmp2 -Descr $tmp3
-AssignmentOrder $tmp4 -ModifyPresent
              $trash =$mo | Add-UcsIpPoolBlock -From $tmp5 -To $tmp6 -DefGw $tmp7
-PrimDns $tmp8 -SecDns $tmp9 -ModifyPresent
            Complete-UcsTransaction | Out-Null

            If ($error.length -lt 1)
            {
                Write-Log "Management IP Pool - Name=$tmp2 Descr=$tmp3 Order=$tmp4"
"Normal"
                Write-Log "                    - From=$tmp5 To=$tmp6 G/W=$tmp7
Primary=$tmp8 Secondary=$tmp9" "Normal"
            }
            Else
```

```
            {
                  Write-Log "ERROR Management IP Pool - Name=$tmp2 Descr=$tmp9
Order=$tmp3" "Error"
                  Write-Log "                              - From=$tmp4 To=$tmp5 G/W=$tmp6
Primary=$tmp7 Secondary=$tmp8" "Error"
                  $errTag = $True
            }


      }
    }
}

# Set Call Home values
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'CallHome' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $error.Clear()
    $tmp1 = $ucsConfig.VSPEX.CallHome.InUse.trim()
    If ($tmp1 -eq '1')
    {
        $tmp1 = $ucsConfig.VSPEX.CallHome.SmtpSrv.trimend(" ")
        $tmp2 = $ucsConfig.VSPEX.CallHome.Address.trimend(" ")
        $tmp3 = $ucsConfig.VSPEX.CallHome.ContactName.trimend(" ")
        $tmp4 = $ucsConfig.VSPEX.CallHome.ContactPhone.trimend(" ")
        $tmp5 = $ucsConfig.VSPEX.CallHome.ContactEmail.trimend(" ")
        $tmp6 = $ucsConfig.VSPEX.CallHome.CustomerID.trimend(" ")
        $tmp7 = $ucsConfig.VSPEX.CallHome.ContractID.trimend(" ")
        $tmp8 = $ucsConfig.VSPEX.CallHome.SiteID.trimend(" ")
        $tmp9 = $ucsConfig.VSPEX.CallHome.SmtpFrom.trimend(" ")
        $tmp10 = $ucsConfig.VSPEX.CallHome.SmtpRecipient.trimend(" ")
        Start-UcsTransaction
          $trash = Get-UcsCallhome | Set-UcsCallhome -AdminState on
-AlertThrottlingAdminState on -Force
          $trash = Get-UcsCallhomeSmtp | Set-UcsCallhomeSmtp -Host $tmp1 -Port 25
-Force
          $trash = Get-UcsCallhomeSource | Set-UcsCallhomeSource -Addr $tmp2
-Contact $tmp3 -Email $tmp5 -Contract $tmp7 -Customer $tmp6 `
             -From $tmp9 -Phone $tmp4 -ReplyTo $tmp9 -Site $tmp8 -Urgency debug
-Force
          $trash = Get-UcsCallhomeProfile -Name full_txt | Add-UcsCallhomeRecipient
-Email $tmp10 -ModifyPresent
        Complete-UcsTransaction | Out-Null
        If ($error.length -lt 1)
        {
            Write-Log "Set call home for $tmp3" "Normal"
        }
        Else
        {
```

```
                        Write-Log "ERROR setting call home for $tmp3" "Error"
                        $errTag = $True
                }
        }
}

# Set Chassis discovery setting
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
        $obj = $objectTable[$i]
        If ($obj[0] -eq 'ChassisDiscovery' -and $obj[1] -eq '1') {$present = $true;
Break}
}
If ($Present)
{
        $error.Clear()
        $tmp1 = $ucsConfig.VSPEX.ChassisDiscovery.trim()
        $trash = $orgRoot | Get-UcsChassisDiscoveryPolicy |
Set-UcsChassisDiscoveryPolicy -Action $tmp1 `
                -LinkAggregationPref "port-channel" -Rebalance "user-acknowledged" -Force
        If ($error.length -lt 1)
        {
                Write-Log "Set chassis discovery policy to $tmp1" "Normal"
        }
        Else
        {
                Write-Log "ERROR setting chassis discovery policy to $tmp1" "Error"
                $errTag = $True
        }

# Set Number of Chassis
        $tmp1 = Get-UcsChassis
        ForEach ($chassis in $tmp1)
        {
                $error.Clear()
                $tmp2 = $chassis.ID
                $trash = Get-UcsChassis -Id $tmp2 | Set-UcsChassis -AdminState
"re-acknowledge" -Force
                If ($error.length -lt 1)
                {
                        Write-Log "Chassis $tmp2 acknowledged" "Normal"
                }
                Else
                {
                        Write-Log "ERROR acknowledging chassis $tmp2" "Error"
                        $errTag = $True
                }
        }
}

# Set Organizations
$present = $false
```

```
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'SubOrg' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.SubOrg | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name
        $tmp2 = $_.Descr
        $trash = $orgRoot  | Add-UcsOrg -Name $tmp1 -Descr $tmp2 -ModifyPresent
        If ($error.length -lt 1)
        {
            Write-Log "Set organization Name=$tmp1 Description=$tmp2" "Normal"
        }
        Else
        {
            Write-Log "ERROR setting organization Name=$tmp1 Description=$tmp2"
"Error"
            $errTag = $True
        }
    }
}

# Set QoS Policies
$error.Clear()
$trash = Get-UcsBestEffortQosClass | Set-UcsBestEffortQosClass -Mtu "9000" -Force |
Out-Null
If ($error.length -lt 1)
{
    Write-Log "Set Best Effort QoS Class to MTU=9000" "Normal"
}
Else
{
    Write-Log "ERROR setting Best Effort QoS Class to MTU=9000" "Error"
    $errTag = $True
}

$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'QoS' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{

     # Platinum
    $error.Clear()
    $tmp1 = $ucsConfig.VSPEX.QoS.Platinum.trim()
    If ($tmp1 -ne "")
    {
```

```
#           Start-UcsTransaction
            $trash = Get-UcsQosClass -Priority "platinum" | Set-UcsQosClass -Mtu
"9000" -Force
            $mo = Add-UcsQosPolicy -Name $tmp1 -ModifyPresent
            $trash = $mo | Get-UcsVnicEgressPolicy | Set-UcsVnicEgressPolicy -Prio
"platinum" -Force
#         Complete-UcsTransaction | Out-Null
          If ($error.length -lt 1)
          {
              Write-Log "Set platinum QoS Class to MTU=9000" "Normal"
          }
          Else
          {
              Write-Log "ERROR setting platinum QoS Class to MTU=9000" "Error"
              $errTag = $True
          }
     }

      # Gold
     $error.Clear()
     $tmp1 = $ucsConfig.VSPEX.QoS.Gold.trim()
     If ($tmp1 -ne "")
     {
#           Start-UcsTransaction
            $trash = Get-UcsQosClass -Priority "gold" | Set-UcsQosClass -Mtu "9000"
-Force
            $mo = Add-UcsQosPolicy -Name $tmp1 -ModifyPresent
            $trash = $mo | Get-UcsVnicEgressPolicy | Set-UcsVnicEgressPolicy -Prio
"gold" -Force
#         Complete-UcsTransaction | Out-Null
          If ($error.length -lt 1)
          {
              Write-Log "Set gold QoS Class to MTU=9000" "Normal"
          }
          Else
          {
              Write-Log "ERROR setting gold QoS Class to MTU=9000" "Error"
              $errTag = $True
          }
     }

      # Silver
     $error.Clear()
     $tmp1 = $ucsConfig.VSPEX.QoS.Silver.trim()
     If ($tmp1 -ne "")
     {
#           Start-UcsTransaction
            $trash = Get-UcsQosClass -Priority "silver" | Set-UcsQosClass -Mtu "9000"
-Force
            $mo = Add-UcsQosPolicy -Name $tmp1 -ModifyPresent
            $trash = $mo | Get-UcsVnicEgressPolicy | Set-UcsVnicEgressPolicy -Prio
silver -Force
#         Complete-UcsTransaction | Out-Null
```

```
        If ($error.length -lt 1)
        {
            Write-Log "Set silver QoS Class to MTU=9000" "Normal"
        }
        Else
        {
            Write-Log "ERROR setting silver QoS Class to MTU=9000" "Error"
            $errTag = $True
        }
    }

     # Bronze
    $error.Clear()
    $tmp1 = $ucsConfig.VSPEX.QoS.Bronze.trim()
    If ($tmp1 -ne "")
    {
#        Start-UcsTransaction
         $trash = Get-UcsQosClass -Priority "bronze" | Set-UcsQosClass -Mtu "9000"
-Force
         $mo = Add-UcsQosPolicy -Name $tmp1 -ModifyPresent
         $trash = $mo | Get-UcsVnicEgressPolicy | Set-UcsVnicEgressPolicy -Prio
bronze -Force
#        Complete-UcsTransaction | Out-Null
    If ($error.length -lt 1)
        {
            Write-Log "Set bronze QoS Class to MTU=9000" "Normal"
        }
        Else
        {
            Write-Log "ERROR setting bronze QoS Class to MTU=9000" "Error"
            $errTag = $True
        }
    }

}

# Set Power Control Policies
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'PowerPolicy' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.PowerPolicy | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Priority.trim()
        If ($tmp1 -eq 'Default') {$tmp1 = $tmp1.tolower()}
        $trash = $orgRoot | Add-UcsPowerPolicy -Name $tmp1 -Prio $tmp2
-ModifyPresent
        If ($error.length -lt 1)
```

```
        {
                Write-Log "Set power control policy Name=$tmp1 to Priority=$tmp2"
"Normal"
        }
        Else
        {
                Write-Log "ERROR setting power control policy Name=$tmp1 to
Priority=$tmp2" "Error"
                $errTag = $True
        }
    }
}

# Set Scrub Policies
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'ScrubPolicy' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.ScrubPolicy | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Descr
        $tmp3 = $_.DiskScrub.trim()
        $tmp3 = $tmp3.tolower()
        $tmp4 = $_.BiosScrub.trim()
        $tmp4 = $tmp4.tolower()
        If ($tmp1 -eq 'Default') {$tmp1 = $tmp1.tolower()}
        $trash = $orgRoot  | Add-UcsScrubPolicy -Name $tmp1 -Descr $tmp2 -DiskScrub
$tmp3 -BiosSettingsScrub $tmp4 -PolicyOwner "local" -ModifyPresent
        If ($error.length -lt 1)
        {
                Write-Log "Set scrub policy Name=$tmp1 Desc=$tmp2 Disc=$tmp3 Bios=$tmp4"
"Normal"
        }
        Else
        {
                Write-Log "ERROR setting scrub policy Name=$tmp1 Desc=$tmp2 Disc=$tmp3
Bios=$tmp4" "Error"
                $errTag = $True
        }
    }
}

# Set Maintenance Policies
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
```

```
    If ($obj[0] -eq 'MaintenancePolicy' -and $obj[1] -eq '1') {$present = $true;
Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.MaintenancePolicy | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Descr.trim()
        $tmp3 = $_.Policy.trim()
        $tmp3 = $tmp3.tolower()
        $trash = $orgRoot  | Add-UcsMaintenancePolicy -Name $tmp1 -Descr $tmp2
-UptimeDisr $tmp3 -ModifyPresent
        If ($error.length -lt 1)
        {
            Write-Log "Set maintenance policy Name=$tmp1 Descr=$tmp2 Policy=$tmp3"
"Normal"
        }
        Else
        {
            Write-Log "ERROR setting maintenance policy Name=$tmp1 Descr=$tmp2
Policy=$tmp3" "Error"
            $errTag = $True
        }
    }
}


# Set Local Disk Policies
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'DiskPolicy' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
#Set-UcsQosClass -QosClass (Get-UcsQosClass -Priority gold) -AdminState enabled -Mtu
9000 -Force
    $ucsConfig.VSPEX.DiskPolicy | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Mode.trim()
        $tmp2 = $tmp2.tolower()
        $tmp3 = $_.Descr.trim()
        $tmp4 = $_.Protect.trim()
        $tmp4 = $tmp4.tolower()
        $trash = $orgRoot | Add-UcsLocalDiskConfigPolicy -Name $tmp1 -Mode $tmp2
-Descr $tmp3 -ProtectConfig $tmp4 -ModifyPresent
        If ($error.length -lt 1)
        {
            Write-Log "Set Local Disk Config Policy Name=$tmp1 Mode=$tmp2
Descr=$tmp3 Protect=$tmp4" "Normal"
        }
```

```
        Else
        {
            Write-Log "ERROR setting Local Disk Config Policy Name=$tmp1 Mode=$tmp2
Descr=$tmp3 Protect=$tmp4" "Error"
            $errTag = $True
        }
    }
}

# Set BIOS Policy
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'BIOSPolicy' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.BIOSPolicy | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Descr
        $tmp3 = $_.VpQuietBoot.trim()
        Start-UcsTransaction
          $mo = $orgRoot  | Add-UcsBiosPolicy -Name $tmp1 -Descr $tmp2
-RebootOnUpdate "no" -ModifyPresent
          $mo | Set-UcsBiosVfQuietBoot -VpQuietBoot $tmp3 -Force | Out-Null
        Complete-UcsTransaction | Out-Null
        If ($error.length -lt 1)
        {
            Write-Log "Create BIOS Policy - Name=$tmp1 Descr=$tmp2 QuietBoot=$tmp3"
"Normal"
        }
        Else
        {
            Write-Log "ERROR create BIOS Policy - Name=$tmp1 Descr=$tmp2
QuietBoot=$tmp3" "Error"
            $errTag = $True
        }
    }
}

# Set vNIC/vHBA Placement Policy
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'PlacementPolicy' -and $obj[1] -eq '1') {$present = $true;
Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.PlacementPolicy | ForEach-Object {$_.Var} | ForEach-Object {
```

```
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.SlotMapping.trim()
        $tmp3 = $_.Selection.trim()
        Start-UcsTransaction
          $mo = $orgRoot  | Add-UcsPlacementPolicy -Name $tmp1 -MezzMapping $tmp2
-ModifyPresent
          $trash = $mo | Add-UcsFabricVCon -Fabric "NONE" -Id "1" -Select $tmp3
-Share "shared" -Transport "ethernet","fc" -ModifyPresent
          $trash = $mo | Add-UcsFabricVCon -Fabric "NONE" -Id "2" -InstType "auto"
-Placement "physical" -Select "all" -Share "shared" -Transport "ethernet","fc"
-ModifyPresent
          $trash = $mo | Add-UcsFabricVCon -Fabric "NONE" -Id "3" -InstType "auto"
-Placement "physical" -Select "all" -Share "shared" -Transport "ethernet","fc"
-ModifyPresent
          $trash = $mo | Add-UcsFabricVCon -Fabric "NONE" -Id "4" -InstType "auto"
-Placement "physical" -Select "all" -Share "shared" -Transport "ethernet","fc"
-ModifyPresent
        Complete-UcsTransaction | Out-Null
        If ($error.length -lt 1)
        {
            Write-Log "Create vNIC/vHBA Placement Policy - Name=$tmp1 Mapping=$tmp2
Selection=$tmp3" "Normal"
        }
        Else
        {
            Write-Log "ERROR creating vNIC/vHBA Placement Policy - Name=$tmp1
Mapping=$tmp2 Selection=$tmp3" "Error"
            $errTag = $True
        }
    }
}

# FI port definitions - assumption that both fabrics are configured identically
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'FI' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.FI | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.SlotID.trim()
        $tmp2 = $_.PortID.trim()
        $tmp3 = $_.Role.trim()
        $tmp4 = $_.UsrLbl
        $tmp5 = $_.VLAN.trim()
        $tmp6 = $_.Native.trim()
        $tmp6 = $tmp6.tolower()
        $tmp7 = $_.Mode.trim()
        $tmp7 = $tmp7.tolower()
```

```
        $tmp8 = $_.QoS.trim()
        $tmp8 = $tmp8.tolower()

        If ($tmp3 -ne '')
        {
            Switch ($tmp3)
            {
                Appliance
                {
                    Start-UcsTransaction
                        $trash = Get-UcsApplianceCloud | Get-UcsVlan -Name $tmp5
-LimitScope | Add-UcsVlanMemberPort -SwitchId "A" -SlotId $tmp1 `
                            -PortId $tmp2 -AdminState "enabled" -IsNative $tmp6 -Name
"" -ModifyPresent
                        $trash = Get-UcsFabricApplianceCloud -Id "A" |
Add-UcsAppliancePort -Slot $tmp1 -PortId $tmp2 -PortMode $tmp7 -Prio $tmp8 `
                            -UsrLbl $tmp4 -AdminSpeed "10gbps" -AdminState "enabled"
-FlowCtrlPolicy "default" -Name "" -NwCtrlPolicyName "default" `
                            -PinGroupName "" -ModifyPresent
                        $trash = Get-UcsApplianceCloud | Get-UcsVlan -Name $tmp5
-LimitScope | Add-UcsVlanMemberPort -SwitchId "B" -SlotId $tmp1 `
                            -PortId $tmp2 -AdminState "enabled" -IsNative $tmp6 -Name
"" -ModifyPresent
                        $trash = Get-UcsFabricApplianceCloud -Id "B" |
Add-UcsAppliancePort -SlotId $tmp1 -PortId $tmp2 -PortMode $tmp7 -Prio $tmp8 `
                            -UsrLbl $tmp4 -AdminSpeed "10gbps" -AdminState "enabled"
-FlowCtrlPolicy "default" -Name "" -NwCtrlPolicyName "default" `
                            -PinGroupName "" -ModifyPresent
                    Complete-UcsTransaction | Out-Null
                    If ($error.length -lt 1)
                    {
                        Write-Log "Set fabric port - Slot=$tmp1 Port=$tmp2
Role=$tmp3 UsrLbl=$tmp4" "Normal"
                        Write-Log "                - VLAN=$tmp5 Native=$tmp6
Mode=$tmp7 QoS=$tmp8" "Normal"
                    }
                    Else
                    {
                        Write-Log "ERROR setting fabric port - Slot=$tmp1 Port=$tmp2
Role=$tmp3 UsrLbl=$tmp4" "Error"
                        Write-Log "                           - VLAN=$tmp5
Native=$tmp6 Mode=$tmp7 QoS=$tmp8" "Error"
                        $errTag = $True
                    }
                }
                FCoE
                {
                    $trash = Get-UcsFiSanCloud -Id "A" | Add-UcsFabricFcoeSanEp
-SlotId $tmp1 -PortId $tmp2 -UsrLbl $tmp4 -Name "" `
                        -AdminState "enabled" -ModifyPresent
                    $trash = Get-UcsFiSanCloud -Id "B" | Add-UcsFabricFcoeSanEp
-SlotId $tmp1 -PortID $tmp2 -UsrLbl $tmp4 -Name "" `
                        -AdminState "enabled" -ModifyPresent
```

```
                    If ($error.length -lt 1)
                    {
                            Write-Log "Set fabric port - Slot=$tmp1 Port=$tmp2
Role=$tmp3 UsrLbl=$tmp4" "Normal"
                    }
                    Else
                    {
                            Write-Log "ERROR setting fabric port - Slot=$tmp1 Port=$tmp2
Role=$tmp3 UsrLbl=$tmp4" "Error"
                            $errTag = $True
                    }
                }
                Server
                {
                        $trash = Get-UcsFabricServerCloud -Id "A" | Add-UcsServerPort
-SlotId $tmp1 -PortId $tmp2 -UsrLbl $tmp4 -Name "" `
                            -AdminState "enabled" -ModifyPresent
                        $trash = Get-UcsFabricServerCloud -Id "B" | Add-UcsServerPort
-SlotId $tmp1 -PortId $tmp2 -UsrLbl $tmp4 -Name "" `
                            -AdminState "enabled" -ModifyPresent
                    If ($error.length -lt 1)
                    {
                            Write-Log "Set fabric port - Slot=$tmp1 Port=$tmp2
Role=$tmp3 UsrLbl=$tmp4" "Normal"
                    }
                    Else
                    {
                            Write-Log "ERROR setting fabric port - Slot=$tmp1 Port=$tmp2
Role=$tmp3 UsrLbl=$tmp4" "Error"
                            $errTag = $True
                    }
                }
                Uplink
                {
                        $trash = Get-UcsFiLanCloud -Id "A" | Add-UcsUplinkPort -SlotId
$tmp1 -PortId $tmp2 -UsrLbl $tmp4 -Name "" `
                            -AdminSpeed "10gbps" -AdminState "enabled" -FlowCtrlPolicy
"default" -ModifyPresent
                        $trash = Get-UcsFiLanCloud -Id "B" | Add-UcsUplinkPort -SlotId
$tmp1 -PortId $tmp2 -UsrLbl $tmp4 -Name "" `
                            -AdminSpeed "10gbps" -AdminState "enabled" -FlowCtrlPolicy
"default" -ModifyPresent
                    If ($error.length -lt 1)
                    {
                            Write-Log "Set fabric port - Slot=$tmp1 Port=$tmp2
Role=$tmp3 UsrLbl=$tmp4" "Normal"
                    }
                    Else
                    {
                            Write-Log "ERROR setting fabric port - Slot=$tmp1 Port=$tmp2
Role=$tmp3 UsrLbl=$tmp4" "Error"
                            $errTag = $True
                    }
```

```
                }
            }
        }
    }
}

# Fibre Channel Port definition
$fiA = Get-UcsFiSanCloud -Id "A"
$fiB = Get-UcsFiSanCloud -Id "B"

$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'FCslot1' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    [int]$itmp1 = $ucsConfig.VSPEX.FCslot1.PortID
    If ($itmp1 -ne 0)
    {
        $error.Clear()
        $tmp2 = $ucsConfig.VSPEX.FCslot1.UsrLbl
        Start-UcsTransaction
          For ($i=32; $i -ge $itmp1; $i--)
          {
              $trash = $fiA | Add-UcsFcUplinkPort -SlotId 1 -PortId $i -UsrLbl $tmp2
-AdminState "enabled" -Name "" -ModifyPresent
          }
        Complete-UcsTransaction | Out-Null
        Start-UcsTransaction
          For ($i=32; $i -ge $itmp1; $i--)
          {
              $trash = $fiB | Add-UcsFcUplinkPort -SlotId 1 -PortId $i -UsrLbl $tmp2
-AdminState "enabled" -Name "" -ModifyPresent
          }
        Complete-UcsTransaction | Out-Null

        If ($error.length -lt 1)
        {
            Write-Log "Set Fixed Module FC Port - Port=$itmp1 UsrLbl=$tmp2" "Normal"
        }
        Else
        {
            Write-Log "ERROR setting Fixed Module FC Port - Port=$itmp1
UsrLbl=$tmp2" "Error"
            $errTag = $True
        }
    }
}

$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
```

```
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'FCslot2' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    [int]$itmp1 = $ucsConfig.VSPEX.FCslot2.PortID
    If ($itmp1 -ne 0)
    {
        $error.Clear()
        $tmp2 = $ucsConfig.VSPEX.FCslot2.UsrLbl
        Start-UcsTransaction
          For ($i=16; $i -ge $itmp1; $i--)
          {
              $trash = $fiA | Add-UcsFcUplinkPort -SlotId 2 -PortId $i -UsrLbl $tmp2
-AdminState "enabled" -Name "" -ModifyPresent
          }
        Complete-UcsTransaction | Out-Null
        Start-UcsTransaction
          For ($i=16; $i -ge $itmp1; $i--)
          {
              $trash = $fiB | Add-UcsFcUplinkPort -SlotId 2 -PortId $i -UsrLbl $tmp2
-AdminState "enabled" -Name "" -ModifyPresent
          }
        Complete-UcsTransaction | Out-Null

        If ($error.length -lt 1)
        {
            Write-Log "Set Expansion Module FC Port - Port=$itmp1 UsrLbl=$tmp2"
"Normal"
        }
        Else
        {
            Write-Log "ERROR setting Expansion Module FC Port - Port=$itmp1
UsrLbl=$tmp2" "Error"
            $errTag = $True
        }
    }
}

# Port Channel
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'PC' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $tmp1 = $ucsConfig.VSPEX.PC.AName.trim()
    $tmp2 = $ucsConfig.VSPEX.PC.BName.trim()
    $tmp3 = $ucsConfig.VSPEX.PC.APortID.trim()
    $tmp4 = $ucsConfig.VSPEX.PC.BPortID.trim()
```

```
    $tmp5 = $ucsConfig.VSPEX.PC.Slot.trim()
    $tmp6 = $ucsConfig.VSPEX.PC.Port1.trim()
    $tmp7 = $ucsConfig.VSPEX.PC.Port2.trim()

    $error.Clear()
    Start-UcsTransaction
      $mo = Get-UcsFiLanCloud -Id A | Add-UcsUplinkPortChannel -Name $tmp1 -PortId
$tmp3 `
          -AdminState "enabled" -AdminSpeed "10gbps" -FlowCtrlPolicy "default"
-ModifyPresent
      $trash = $mo | Add-UcsUplinkPortChannelMember -SlotId $tmp5 -PortId $tmp6
-AdminState "enabled" -ModifyPresent
      $trash = $mo | Add-UcsUplinkPortChannelMember -SlotId $tmp5 -PortId $tmp7
-AdminState "enabled" -ModifyPresent
    Complete-UcsTransaction | Out-Null
    If ($error.length -lt 1)
    {
        Write-Log "Set Port Channel - Name=$tmp1 PortChannelID=$tmp3
Slot/Port/Port=$tmp5/$tmp6/$tmp7" "Normal"
    }
    Else
    {
        Write-Log "ERROR Set Port Channel - Name=$tmp1 PortChannelID=$tmp3
Slot/Port/Port=$tmp5/$tmp6/$tmp7" "Error"
        $errTag = $True
    }

    $error.Clear()
    Start-UcsTransaction
      $mo = Get-UcsFiLanCloud -Id B | Add-UcsUplinkPortChannel -Name $tmp2 -PortId
$tmp4 `
          -AdminState "enabled" -AdminSpeed "10gbps" -FlowCtrlPolicy "default"
-ModifyPresent
      $trash = $mo | Add-UcsUplinkPortChannelMember -SlotId $tmp5 -PortId $tmp6
-AdminState "enabled" -ModifyPresent
      $trash = $mo | Add-UcsUplinkPortChannelMember -SlotId $tmp5 -PortId $tmp7
-AdminState "enabled" -ModifyPresent
    Complete-UcsTransaction | Out-Null
    If ($error.length -lt 1)
    {
        Write-Log "Set Port Channel - Name=$tmp2 PortChannelID=$tmp4
Slot/Port/Port=$tmp5/$tmp6/$tmp7" "Normal"
    }
    Else
    {
        Write-Log "ERROR Set Port Channel - Name=$tmp2 PortChannelID=$tmp4
Slot/Port/Port=$tmp5/$tmp6/$tmp7" "Error"
        $errTag = $True
    }
}

# Various Pools
$present = $false
```

```
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'Pools' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.Pools | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Type.trim()
        $tmp2 = $_.Name.trim()
        $tmp3 = $_.From.trim()
        $tmp4 = $_.To.trim()
        $tmp5 = $_.Order.trim()
        $tmp5 = $tmp5.tolower()
        $tmp6 = $_.Org.trim()
        $tmp7 = $_.Descr
        If ($tmp2 -eq 'default') {$tmp2 = $tmp2.tolower()}

        If ($tmp6 -eq "root") {$mo = $orgRoot}
        Else {$mo = $orgRoot | Get-UcsOrg -Name $tmp6 -LimitScope}
        Switch ($tmp1)
        {
            MAC
            {
                Start-UcsTransaction
                $mo_1 = $mo | Add-UcsMacPool -Name $tmp2 -AssignmentOrder $tmp5
-Descr $tmp7 -ModifyPresent
                $trash = $mo_1 | Add-UcsMacMemberBlock -From $tmp3 -To $tmp4
-ModifyPresent
                Complete-UcsTransaction | Out-Null
                If ($error.length -lt 1)
                {
                    Write-Log "Add MAC Pool - Name=$tmp2 Org=$tmp6 Descr=$tmp7
From/To=$tmp3-$tmp4" "Normal"
                }
                Else
                {
                    Write-Log "ERROR Add MAC Pool - Name=$tmp2 Org=$tmp6 Descr=$tmp7
From/To=$tmp3-$tmp4" "Error"
                    $errTag = $True
                }
            }
            UUID
            {
                Start-UcsTransaction
                $mo_1 = $mo | Add-UcsUuidSuffixPool -Name $tmp2 -AssignmentOrder
$tmp5 -Descr $tmp7  -ModifyPresent
                $trash = $mo_1 | Add-UcsUuidSuffixBlock -From $tmp3 -To $tmp4
-ModifyPresent
                Complete-UcsTransaction | Out-Null
                If ($error.length -lt 1)
                {
```

```
                    Write-Log "Add UUID Pool - Name=$tmp2 Org=$tmp6 Descr=$tmp7
From/To=$tmp3-$tmp4" "Normal"
                }
                Else
                {
                    Write-Log "ERROR Add UUID Pool - Name=$tmp2 Org=$tmp6
Descr=$tmp7 From/To=$tmp3-$tmp4" "Error"
                    $errTag = $True
                }
        }
        WWNN
        {
                Start-UcsTransaction
                  $mo_1 = $mo | Add-UcsWwnPool -Name $tmp2 -AssignmentOrder $tmp5
-Purpose "node-wwn-assignment" -Descr $tmp7 -ModifyPresent
                  $trash = $mo_1 | Add-UcsWwnMemberBlock -From $tmp3 -To $tmp4
-ModifyPresent
                Complete-UcsTransaction | Out-Null
                If ($error.length -lt 1)
                {
                    Write-Log "Add WWNN Pool - Name=$tmp2 Org=$tmp6 Descr=$tmp7
From/To=$tmp3-$tmp4" "Normal"
                }
                Else
                {
                    Write-Log "ERROR Add WWNN Pool - Name=$tmp2 Org=$tmp6
Descr=$tmp7 From/To=$tmp3-$tmp4" "Error"
                    $errTag = $True
                }
        }
        WWPN
        {
                Start-UcsTransaction
                  $mo_1 = $mo | Add-UcsWwnPool -Name $tmp2 -AssignmentOrder $tmp5
-Purpose "port-wwn-assignment" -Descr $tmp7 -ModifyPresent
                  $trash = $mo_1 | Add-UcsWwnMemberBlock -From $tmp3 -To $tmp4
-ModifyPresent
                Complete-UcsTransaction | Out-Null
                If ($error.length -lt 1)
                {
                    Write-Log "Add WWPN Pool - Name=$tmp2 Org=$tmp6 Descr=$tmp7
From/To=$tmp3-$tmp4" "Normal"
                }
                Else
                {
                    Write-Log "ERROR Add WWPN Pool - Name=$tmp2 Org=$tmp6
Descr=$tmp7 From/To=$tmp3-$tmp4" "Error"
                    $errTag = $True
                }
        }
    }
  }
}
```

```
# VLAN definitions
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'VLANs' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.VLANs | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Fabric.trim()
        $tmp3 = $_.ATag.trim()
        $tmp4 = $_.BTag.trim()
        $tmp5 = $_.DefaultNet.trim()
        Switch ($tmp2)
        {
            Common
            {
                $trash = Get-UcsLanCloud | Add-UcsVlan -Name $tmp1 -Id $tmp3
-DefaultNet $tmp5 -CompressionType "included" -Sharing "none" -ModifyPresent
            }
            Diff
            {
                $trash = Get-UcsFiLanCloud -Id "A" | Add-UcsVlan -Name $tmp1 -Id
$tmp3 -DefaultNet $tmp5 -CompressionType "included" -Sharing "none" -ModifyPresent
                $trash = Get-UcsFiLanCloud -Id "B" | Add-UcsVlan -Name $tmp1 -Id
$tmp4 -DefaultNet $tmp5 -CompressionType "included" -Sharing "none" -ModifyPresent
            }
            FabA
            {
                $trash = Get-UcsFiLanCloud -Id "A" | Add-UcsVlan -Name $tmp1 -Id
$tmp3 -DefaultNet $tmp5 -CompressionType "included" -Sharing "none" -ModifyPresent
            }
            FabB
            {
                $trash = Get-UcsFiLanCloud -Id "B" | Add-UcsVlan -Name $tmp1 -Id
$tmp4 -DefaultNet $tmp5 -CompressionType "included" -Sharing "none" -ModifyPresent
            }
        }
        If ($error.length -lt 1)
        {
            Write-Log "Add VLAN - Name=$tmp1 Fabric=$tmp2 ATag=$tmp3 BTag=$tmp4
DefaultNet=$tmp5" "Normal"
        }
        Else
        {
            Write-Log "ERROR Add VLAN - Name=$tmp1 Fabric=$tmp2 ATag=$tmp3
BTag=$tmp4 DefaultNet=$tmp5" "Error"
            $errTag = $True
        }
```

```
    }
}

# VNIC Templates
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'VNICTemplate' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.VNICTemplate | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.MTU.trim()
        $tmp3 = $_.Fabric.trim()
        $tmp3 = $tmp3.toupper()
        $tmp4 = $_.MACpool.trim()
        $tmp5 = $_.QoS.trim()
        $tmp6 = $_.VLAN.trim()
            $tmp7 = $_.Order.trim()
        $tmp8 = $_.Type.trim()
        $tmp8 = $tmp8.tolower()
        $tmp9 = $_.Native.trim()
        $tmp9 = $tmp9.tolower()
        $tmp10 = $_.Org.trim()

        If ($tmp10 -eq "root") {$org = $orgRoot}
        Else {$org = $orgRoot | Get-UcsOrg -Name $tmp10 -LimitScope}

        Start-UcsTransaction
          $mo = $org | Add-UcsVnicTemplate -Name $tmp1 -Mtu $tmp2 -SwitchId $tmp3
-IdentPoolName $tmp4 `
            -QosPolicyName $tmp5 -TemplType $tmp8 -ModifyPresent
          $trash = $mo | Add-UcsVnicInterface -Name $tmp6 -DefaultNet $tmp9
-ModifyPresent
        Complete-UcsTransaction | Out-Null
        If ($error.length -lt 1)
        {
            Write-Log "Add VNIC template - Name=$tmp1 MTU=$tmp2 Fabric=$tmp3
MACPool=$tmp4 Type=$tmp9" "Normal"
        }
        Else
        {
            Write-Log "ERROR Add VNIC template - Name=$tmp1 MTU=$tmp2 Fabric=$tmp3
MACPool=$tmp4 Type=$tmp9" "Error"
            $errTag = $True
        }
    }
}


# vHBA Templates
```

```
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'VHBATemplate' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.VHBATemplate | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Descr
        $tmp3 = $_.Fabric.trim()
        $tmp3 = $tmp3.toupper()
        $tmp4 = $_.VSAN.trim()
        $tmp5 = $_.Type.trim()
        $tmp5 = $tmp5.tolower()
        $tmp6 = $_.WWNpool.trim()
        $tmp7 = $_.QoS.trim()
        $tmp8 = $_.Org.trim()

        If ($tmp8 -eq "root") {$org = $orgRoot}
        Else {$org = $orgRoot | Get-UcsOrg -Name $tmp8 -LimitScope}

        Start-UcsTransaction
          $mo = $org | Add-UcsVhbaTemplate -Name $tmp1 -Descr $tmp2 -SwitchId $tmp3
-TemplType $tmp5 -IdentPoolName $tmp6 `
              -MaxDataFieldSize 2048 -PinToGroupName "" -PolicyOwner "local"
-QosPolicyName $tmp7 -StatsPolicyName "default" -ModifyPresent
          $trash = $mo | Add-UcsVhbaInterface -Name $tmp4 -ModifyPresent
        Complete-UcsTransaction | Out-Null

        If ($error.length -lt 1)
        {
            Write-Log "Add VHBA template - Name=$tmp1 Descr=$tmp2 Fabric=$tmp3
VSAN=$tmp4 Type=$tmp5 Pool=$tmp6 QoS=$tmp7" "Normal"
        }
        Else
        {
            Write-Log "ERROR Add VHBA template - Name=$tmp1 Descr=$tmp2 Fabric=$tmp3
VSAN=$tmp4 Type=$tmp5 Pool=$tmp6 QoS=$tmp7" "Error"
            $errTag = $True
        }
    }
}

# Boot Policies
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'BootPolicy' -and $obj[1] -eq '1') {$present = $true; Break}
}
```

```
If ($Present)
{
    $ucsConfig.VSPEX.BootPolicy | ForEach-Object {$_.PolicyName} | ForEach-Object {
        $error.Clear()
        $i = 0
        $tmp1 = $_
        $tmp2 = $_.Name.trim()
        $tmp3 = $_.Descr
        $tmp4 = $_.Org.trim()

        If ($tmp4 -eq "root") {$org = $orgRoot}
        Else {$org = $orgRoot | Get-UcsOrg -Name $tmp4 -LimitScope}

        $mo = $org | Add-UcsBootPolicy -Name $tmp2 -Descr $tmp3 -EnforceVnicName
"yes" -PolicyOwner "local" `
            -RebootOnUpdate "no" -ModifyPresent

        $tmp1 | ForEach-Object {$_.Var} | ForEach-Object {
            $i++
            $tmp4 = $_.Type.trim()
            $tmp5 = $_.Device1.trim()
            $tmp6 = $_.Device2.trim()
            $tmp7 = $_.PrimaryFabric

            Switch ($tmp4)
            {
                Local
                {
                    Switch ($tmp5)
                    {
                        cdrom
                        {
                            $trash = $mo | Add-UcsLsbootVirtualMedia -Access
"read-only" -Order $i -ModifyPresent
                        }
                        floppy
                        {
                            $trash = $mo | Add-UcsLsbootVirtualMedia -Access
"read-write" -Order $i -ModifyPresent
                        }
                        localdisk
                        {
                            $mo_1 = $mo | Add-UcsLsbootStorage -Order $i
-ModifyPresent

                            $trash = $mo_1 | Add-UcsLsbootLocalStorage
                        }
                    }
                }
                VHBA
                {
                    If ($tmp7 -eq 'A')
                    {
                        Start-UcsTransaction
```

```
                        $mo_2 = $mo | Add-UcsLsbootStorage -Order $i
-ModifyPresent
                        $mo_2_1 = $mo_2 | Add-UcsLsbootSanImage -Type "primary"
-VnicName $tmp5 -ModifyPresent
                        $trash = $mo_2_1 | Add-UcsLsbootSanImagePath -Lun 0 -Type
"primary" -Wwn $sanSPAprimary -ModifyPresent
                        $trash = $mo_2_1 | Add-UcsLsbootSanImagePath -Lun 0 -Type
"secondary" -Wwn $sanSPBprimary -ModifyPresent
                        $mo_2_2 = $mo_2 | Add-UcsLsbootSanImage -Type "secondary"
-VnicName $tmp6 -ModifyPresent
                        $trash = $mo_2_2 | Add-UcsLsbootSanImagePath -Lun 0 -Type
"primary" -Wwn $sanSPAsecondary -ModifyPresent
                        $trash = $mo_2_2 | Add-UcsLsbootSanImagePath -Lun 0 -Type
"secondary" -Wwn $sanSPBsecondary -ModifyPresent
                    Complete-UcsTransaction | Out-Null
                }
                Else
                {
                    Start-UcsTransaction
                    $mo_2 = $mo | Add-UcsLsbootStorage -Order $i
-ModifyPresent
                        $mo_2_1 = $mo_2 | Add-UcsLsbootSanImage -Type "primary"
-VnicName $tmp5 -ModifyPresent
                        $trash = $mo_2_1 | Add-UcsLsbootSanImagePath -Lun 0 -Type
"primary" -Wwn $sanSPBprimary -ModifyPresent
                        $trash = $mo_2_1 | Add-UcsLsbootSanImagePath -Lun 0 -Type
"secondary" -Wwn $sanSPAprimary -ModifyPresent
                        $mo_2_2 = $mo_2 | Add-UcsLsbootSanImage -Type "secondary"
-VnicName $tmp6 -ModifyPresent
                        $trash = $mo_2_2 | Add-UcsLsbootSanImagePath -Lun 0 -Type
"primary" -Wwn $sanSPBsecondary -ModifyPresent
                        $trash = $mo_2_2 | Add-UcsLsbootSanImagePath -Lun 0 -Type
"secondary" -Wwn $sanSPAsecondary -ModifyPresent
                    Complete-UcsTransaction | Out-Null
                }
            }
            VNIC
            {
                $mo_1 = $mo | Add-UcsLsbootLan -Order $i -Prot "pxe"
-ModifyPresent
                $trash = $mo_1 | Add-UcsLsbootLanImagePath -VnicName $tmp5
-BootIpPolicyName "" -ISCSIVnicName "" -ImgPolicyName "" -ImgSecPolicyName ""
-ProvSrvPolicyName "" -Type "primary"
            }
        }
        If ($error.length -lt 1)
        {
            Write-Log "Add boot policy - Name=$tmp2 Descr=$tmp3" "Normal"
        }
        Else
        {
            Write-Log "ERROR Add boot policy - Name=$tmp1 Descr=$tmp2" "Error"
            $errTag = $True
```

```
            }
        }
    }
}

# Service Profile Template
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'SPTemplate' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.SPTemplate | ForEach-Object {$_.Template} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_
        $tmp2 = $_.Name.trim()
        $tmp3 = $_.Descr

        Start-UcsTransaction
          $tmp4 = $_.BIOSProfileName.trim()
          $tmp5 = $_.BootPolicyName.trim()
          $tmp6 = $_.LocalDiskPolicy.trim()
          $tmp7 = $_.MgmtIPPool.trim()
          $tmp8 = "pooled"
          If ($tmp7 -eq "") {$tmp8 = "none"}
          $tmp9 = $_.PowerPolicyName.trim()
          $tmp10 = $_.ScrubPolicyName.trim()
          $tmp11 = $_.UUIDpool.trim()
          $tmp12 = $_.MaintPolicyName.trim()
          $tmp13 = $_.HostFwPolicyName.trim()
          $tmp14 = $_.MgmtAccessPolicyName.trim()
          $tmp15 = $_.MgmtFwPolicyName.trim()
          $tmp16 = $_.StatsPolicyName.trim()
          $tmp17 = $_.Org.trim()
          $tmp18 = $_.WwnnPoolName.trim()
          If ($tmp17 -eq "root") {$org = $orgRoot}
          Else {$org = $orgRoot | Get-UcsOrg -Name $tmp17 -LimitScope}

          $mo = $org | Add-UcsServiceProfile -Name $tmp2 -Type "updating-template"
-ModifyPresent `
              -BiosProfileName $tmp4 -BootPolicyName $tmp5 -LocalDiskPolicyName
$tmp6 -ExtIPPoolName $tmp7 -ExtIPState $tmp8 `
              -PowerPolicyName $tmp6 -ScrubPolicyName $tmp10 -IdentPoolName $tmp11
-MaintPolicyName $tmp12 `
              -HostFwPolicyName $tmp13 -MgmtAccessPolicyName $tmp14
-MgmtFwPolicyName $tmp15 -StatsPolicyName $tmp16
          $trash = $mo | Add-UcsVnicFcNode -Addr "pool-derived" -IdentPoolName
$tmp18 -ModifyPresent
          $trash = $mo | Add-UcsServerPoolAssignment -ModifyPresent -Name
"All-chassis" -Qualifier "all-chassis" -RestrictMigration "no"
          $trash = $mo | Set-UcsServerPower -State "admin-up" -Force
```

```
#       Process all the VNICs
        $tmp100 = $_.VNICs
        $i = 0
        $tmp100 | ForEach-Object {$_.Var} | ForEach-Object {
            $tmp4 = $_.Name.trim()
            $tmp5 = $_.Templ.trim()
            $i++
            $trash = $mo | Add-UcsVnic -Name $tmp4 -NwTemplName $tmp5
-AdaptorProfileName "Windows" -Order $i -ModifyPresent
        }

#       Process all the VHBAs
        $tmp100 = $_.VHBAs
        $tmp100 | ForEach-Object {$_.Var} | ForEach-Object {
            $tmp4 = $_.Name.trim()
            $tmp5 = $_.Templ.trim()
            $i++
            $mo_1 = $mo | Add-UcsVhba -Name $tmp4  -NwTemplName $tmp5
-AdaptorProfileName "Windows" -Order $i -ModifyPresent
            $trash = $mo_1 | Add-UcsVhbaInterface -Name "" -ModifyPresent
        }
        Complete-UcsTransaction | Out-Null
        If ($error.length -lt 1)
        {
            Write-Log "Add Service Profile Template - Name=$tmp2 Descr=$tmp3"
"Normal"
        }
        Else
        {
            Write-Log "ERROR Service Profile Template - Name=$tmp1 Descr=$tmp2"
"Error"
            $errTag = $True
        }
    }
}

# Create Service Profiles from Templates
$present = $false
For ($i=0; $i -lt $objectTable.length; $i++)
{
    $obj = $objectTable[$i]
    If ($obj[0] -eq 'ServiceProfile' -and $obj[1] -eq '1') {$present = $true; Break}
}
If ($Present)
{
    $ucsConfig.VSPEX.ServiceProfile | ForEach-Object {$_.Var} | ForEach-Object {
        $error.Clear()
        $tmp1 = $_.Name.trim()
        $tmp2 = $_.Templ.trim()
        $tmp3 = $_.Org.trim()
        If ($tmp3 -eq "root") {$org = $orgRoot}
        Else {$org = $orgRoot | Get-UcsOrg -Name $tmp3 -LimitScope}
```

```
        $mo = Get-UcsServiceProfile -Name $tmp2 -Org $org
        If ($mo -eq $null)
        {
              Write-Log "ERROR Service Profile - invalid template name - Name=$tmp1
Templ=$tmp2 Org=$tmp3" "Error"
              $errTag = $True
        }
        Else
        {
              $trash = $mo | Add-UcsServiceProfileFromTemplate -NewName @($tmp1)
-DestinationOrg $org
        }

        If ($error.length -lt 1)
        {
              Write-Log "Add Service Profile - Name=$tmp1 Templ=$tmp2 Org=$tmp3"
"Normal"
        }
        Else
        {
              Write-Log "ERROR Service Profile - Name=$tmp1 Templ=$tmp2 Org=$tmp3"
"Error"
              $errTag = $True
        }
    }
}


# -------------------------------------------------------------------------------
#
# Wrap up processing and close down
#
# -------------------------------------------------------------------------------

Disconnect-Ucs
If ($errTag)
{
    Write-Host -ForegroundColor Red -BackgroundColor Black "`n`nErrors detected."
}
Set-Location $originalPath
$endTime = Get-Date
$elapsedTime = New-TimeSpan $startTime $endTime
If ($toconsole)
{
    Write-Log "Elapsed time:
$($elapsedTime.Hours):$($elapsedTime.Minutes):$($elapsedTime.Seconds)" "Normal"
    Write-Log "End of processing." "Normal"
}
Else
{
    Write-Host -ForegroundColor Yellow -BackgroundColor Black
"`n`n------------------------------------------------------------`n"
```

```
    Write-Host "Elapsed time:
$($elapsedTime.Hours):$($elapsedTime.Minutes):$($elapsedTime.Seconds)"
    Write-Host "End of processing."
```

UcsConfig.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
 UcsConfig Version=0.1
 4-September-2013
 Created by Tim Cerling
 tcerling@cisco.com

 Take care in editing this file.  It is better to comment out the detail of an
 object  rather than deleting it if you do not want to enter values.  This will
 retain the structure of the XML should you want to reuse it with more or different
 values in the future.  For example, if you have already defined Management IP
 addresses and do not want to make any additional changes or additions, you could
 do something like this:

 <MgmtIP>
#    <Pool Name='MgmtIP' Descr='Service Profile management IPs' >
#      <Order>sequential </Order>
#      <Start>10.5.177.200 </Start>
#      <End>10.5.177.249 </End>
#      <Gateway>10.5.177.1 </Gateway>
#      <PrimaryDNS>0.0.0.0 </PrimaryDNS>
#      <SecondaryDNS>0.0.0.0 </SecondaryDNS>
#    </Pool>
  </MgmtIP>



-->

<VSPEX>

<!-- UCSM IP address.  Can be either FQDN or numeric. -->
  <UCSMIP>10.29.130.100 </UCSMIP>

<!-- Timezone for this UCS domain -->
  <TimeZone>America/Los_Angeles (Pacific Time) </TimeZone>

<!-- List of FQDN or IP addresses of NTP servers -->
  <NTP>
    <Var Name='1.ntp.esl.cisco.com' />
    <Var Name='2.ntp.esl.cisco.com' />
  </NTP>

  <MgmtIP>
    <Pool Name='MgmtIP' Descr='Service Profile management IPs' >
      <Order>sequential </Order>
      <Start>10.5.177.200 </Start>
```

```
        <End>10.5.177.249 </End>
        <Gateway>10.5.177.1 </Gateway>
        <PrimaryDNS>0.0.0.0 </PrimaryDNS>
        <SecondaryDNS>0.0.0.0 </SecondaryDNS>
      </Pool>
    </MgmtIP>

<!-- Optional CallHome definitions.  To make use of this, define InUse to equal 1
-->
    <CallHome>
      <InUse>0 </InUse>  <!-- To define, InUse=1.  To not define, InUse=0. -->
      <SmtpSrv>smtprelay.customer.com </SmtpSrv>
      <Address>123 Main Street, Anytown, CA 54321 </Address>
      <ContactName>First Last </ContactName>
      <ContactPhone>+15551234567 </ContactPhone>
      <ContactEmail>contact@customer.com </ContactEmail>
      <CustomerID>12345 </CustomerID>
      <ContractID>12345 </ContractID>
      <SiteID>12345 </SiteID>
      <SmtpFrom>UCSstringCallHome@customer.com </SmtpFrom>
      <SmtpRecipient>contact@customer.com </SmtpRecipient>
    </CallHome>

<!-- Chassis Discovery - 1-link, 2-link, 4-link, 8-link, platform-max -->
    <ChassisDiscovery>2-link </ChassisDiscovery>

<!-- Sub-Organizations to create. -->
<!-- NOTE:  Only supports sub-organizations to root.  -->
    <SubOrg>
      <Var Name='VSPEX' Descr='For all VSPEX work' />
    </SubOrg>

<!--SAN WWPN.  Provide EMC SPA/SPB primary and secondary ports WWPNs.  Used for
creating boot policy. -->
    <SANWWPN>
      <SPAprimary>50:06:01:65:08:60:06:A1 </SPAprimary>
      <SPAsecondary>50:06:01:64:08:60:06:A1 </SPAsecondary>
      <SPBprimary>50:06:01:6D:08:60:06:A1 </SPBprimary>
      <SPBsecondary>50:06:01:6C:08:60:06:A1 </SPBsecondary>
    </SANWWPN>

<!-- Associate QoS with custom names -->
<!-- NOTE: assigning a value forces an MTU of 9000 -->
    <QoS>
      <Platinum>LiveMigration </Platinum>
      <Gold>iSCSI </Gold>
      <Silver> </Silver>
      <Bronze> </Bronze>
    </QoS>

<!-- Power Control Policy -->
    <PowerPolicy>
      <Var Name='default' Priority='no-cap' />
```

```
    <Var Name='Cap_1' Priority='1' />
    <Var Name='Cap_2' Priority='2' />
    <Var Name='Cap_3' Priority='3' />
    <Var Name='Cap_4' Priority='4' />
    <Var Name='Cap_5' Priority='5' />
    <Var Name='Cap_6' Priority='6' />
    <Var Name='Cap_7' Priority='7' />
    <Var Name='Cap_8' Priority='8' />
    <Var Name='Cap_9' Priority='9' />
    <Var Name='Cap_10' Priority='10' />
    <Var Name='NoCap' Priority='no-cap' />
  </PowerPolicy>

<!-- Scrub Policies -->
  <ScrubPolicy>
    <Var Name='NoScrub' Descr='Do not scrub' DiskScrub='no' BiosScrub='no' />
    <Var Name='DiskScrub' Descr='Scrub disk' DiskScrub='yes' BiosScrub='no' />
    <Var Name='BiosScrub' Descr='Scrub Bios' DiskScrub='no' BiosScrub='yes' />
    <Var Name='AllScrub' Descr='Scrub disk and Bios' DiskScrub='yes' BiosScrub='yes'
/>
  </ScrubPolicy>

<!-- Maintenance policies -->
  <MaintenancePolicy>
    <Var Name='Immediate' Descr='Immediately reboot on profile change'
Policy='immediate' />
    <Var Name='UserAck' Descr='User acknowledge reboot on profile change'
Policy='user-ack' />
    <Var Name='Timer-auto' Descr='Timer reboot on default schedule'
Policy='timer-automatic' />
  </MaintenancePolicy>

<!-- Local Disk Policy -->
  <DiskPolicy>
    <Var Name='AnyConfiguration' Mode='any-configuration' Descr='Any Disk
Configuration' Protect='yes' />
    <Var Name='NoLocal' Mode='no-local-storage' Descr='Ignore local storage'
Protect='yes' />
    <Var Name='NoRAID' Mode='no-raid' Descr='No RAID storage' Protect='yes' />
    <Var Name='RAID0' Mode='raid-striped' Descr='RAID 0 Striped' Protect='yes' />
    <Var Name='RAID1' Mode='raid-mirrored' Descr='RAID 1 Mirrored' Protect='yes' />
    <Var Name='RAID10' Mode='raid-mirrored-striped' Descr='RAID 10 Mirrored and
Striped' Protect='yes' />
    <Var Name='RAID5' Mode='raid-striped-parity' Descr='RAID 5 Striped Parity'
Protect='yes' />
    <Var Name='RAID6' Mode='raid-striped-dual-parity' Descr='RAID 6 Striped Dual
Parity' Protect='yes' />
  </DiskPolicy>

<!-- BIOS Policy.  Currently only No Quiet Boot is programmed. -->
<!-- NOTE:  Do not change this entry.  Use UCSM to create different BIOS policies
-->
  <BIOSPolicy>
```

```
    <Var Name='NoQuietBoot' Descr= 'No quiet boot' VpQuietBoot='disabled' />
  </BIOSPolicy>

<!-- vNIC/vHBA Placement Policy.  -->
<!-- NOTE:  Do not change this entry.  Use UCSM to create different placement
policies -->
<!-- If these entries get changed, the logic will also need to be changed to reflect
-->
  <PlacementPolicy>
    <Var Name='AssignedOnly' SlotMapping='round-robin' Selection='assigned-only' />
    <Var Name='ExcludeDynamic' SlotMapping='round-robin' Selection='exclude-dynamic'
/>
    <Var Name='ExcludeUnassign' SlotMapping='round-robin'
Selection='exclude-unassigned' />
  </PlacementPolicy>

<!-- FI port definitions.  Assumed FI-A and FI-B are configured identically.
Recommended to include UsrLbl.
    Defining role as '' causes no change to the configuration
    Last four variables only applicable to Appliance role for iSCSI and SMB.

    NOTE: Currently only supports 6248UP

    NOTE - NOTE - NOTE  FC ports are special.  They are defined in the FCslot1 and
FCslot2 objects, not here.
    See description for FCslot1 and FCslot2 objects.
-->

  <FI>
    <Var SlotID='1' PortID='1' Role='Server' UsrLbl='Blade Server' VLAN='n/a'
Native='n/a' Mode='n/a' QoS='n/a' />
    <Var SlotID='1' PortID='2' Role='Server' UsrLbl='Blade Server' VLAN='n/a'
Native='n/a' Mode='n/a' QoS='n/a' />
    <Var SlotID='1' PortID='3' Role='Server' UsrLbl='Blade Server' VLAN='n/a'
Native='n/a' Mode='n/a' QoS='n/a' />
    <Var SlotID='1' PortID='4' Role='Server' UsrLbl='Blade Server' VLAN='n/a'
Native='n/a' Mode='n/a' QoS='n/a' />
    <Var SlotID='1' PortID='5' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='6' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='7' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='8' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='9' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='10' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='11' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='12' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
```

```
    <Var SlotID='1' PortID='13' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='14' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='15' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='16' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='17' Role='Uplink' UsrLbl='Uplink Port' VLAN='n/a'
Native='n/a' Mode='n/a' QoS='n/a' />
    <Var SlotID='1' PortID='18' Role='Uplink' UsrLbl='Uplink Port' VLAN='n/a'
Native='n/a' Mode='n/a' QoS='n/a' />
    <Var SlotID='1' PortID='19' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='20' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='21' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='22' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='23' Role='Appliance' UsrLbl='10 GE SMB' VLAN='SMB'
Native='no' Mode='Access' QoS='gold' />
    <Var SlotID='1' PortID='24' Role='Appliance' UsrLbl='10 GE SMB' VLAN='SMB'
Native='no' Mode='Access' QoS='gold' />
    <Var SlotID='1' PortID='25' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='26' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='27' Role='' UsrLbl='' VLAN='' Native='no' Mode='Access'
QoS='n/a' />
    <Var SlotID='1' PortID='28' Role='' UsrLbl='' VLAN='' Native='no' Mode='Access'
QoS='n/a' />
    <Var SlotID='1' PortID='29' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='30' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='31' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='1' PortID='32' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='2' PortID='1' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='2' PortID='2' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='2' PortID='3' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='2' PortID='4' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='2' PortID='5' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
    <Var SlotID='2' PortID='6' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
```

```
     <Var SlotID='2' PortID='7' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
     <Var SlotID='2' PortID='8' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
     <Var SlotID='2' PortID='9' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
     <Var SlotID='2' PortID='10' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
     <Var SlotID='2' PortID='11' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
     <Var SlotID='2' PortID='12' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
     <Var SlotID='2' PortID='13' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
     <Var SlotID='2' PortID='14' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
     <Var SlotID='2' PortID='15' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
     <Var SlotID='2' PortID='16' Role='' UsrLbl='' VLAN='n/a' Native='n/a' Mode='n/a'
QoS='n/a' />
   </FI>

<!-- Fibre Channel port defintions.  Assumes FC ports configured on same ports on
each FI.

     The PortID value is the port number for the first port to be defined as FC.
All following ports
     will be included as FC ports.  FCslot1 is for Module 1.  FCslot2 is for
expansion module 2 and it
     assumes that it is a unified ports expansion module.
-->
   <FCslot1 PortID='' UsrLbl='' />
   <FCslot2 PortID='' UsrLbl='' />

<!-- Port Channel definition.  Assumes PC configured on same ports on each FI -->
   <PC>
     <AName>VPC201 </AName>
     <APortID>201 </APortID>
     <BName>VPC202 </BName>
     <BPortID>202 </BPortID>
     <Slot>1 </Slot>
     <Port1>17 </Port1>
     <Port2>18 </Port2>
   </PC>

<!-- Various pools.  Multiple pools of same type can be defined. -->
   <Pools>
     <Var Type='MAC' Name='VSPEX-99-MAC' From='00:25:B5:99:00:00'
To='00:25:B5:99:00:FF' Order='sequential' Org='VSPEX' Descr='' />
     <Var Type='UUID' Name='VSPEX-99-UUID' From='0099-000000000001'
To='0099-000000000040' Order='sequential' Org='VSPEX' Descr='' />
     <Var Type='WWNN' Name='VSPEX-99-WWNN' From='20:00:00:25:B5:99:00:00'
To='20:00:00:25:B5:99:00:3F' Order='sequential' Org='VSPEX' Descr='' />
```

```
      <Var Type='WWPN' Name='VSPEX-99-WWPN' From='20:00:00:25:B5:99:00:40'
To='20:00:00:25:B5:99:00:FF' Order='sequential' Org='VSPEX' Descr='' />
  </Pools>

<!-- VLAN definitions -->
<!-- Use ATag for common configuration VLAN tag -->
<!-- Fabric can be defined as Common, Diff, FabA, or FabB -->
  <VLANs>
      <Var Name='Mgmt' Fabric='Common' ATag='1' BTag='' DefaultNet="yes" />
      <Var Name='VMaccess' Fabric='Common' ATag='10' BTag='' DefaultNet="no" />
      <Var Name='CSV' Fabric='Common' ATag='12' BTag='' DefaultNet="no" />
      <Var Name='LiveMigration' Fabric='Common' ATag='11' BTag='' DefaultNet="no" />
      <Var Name='ClusComm' Fabric='Common' ATag='13' BTag='' DefaultNet="no" />
      <Var Name='iSCSI' Fabric='Diff' ATag='18' BTag='19' DefaultNet="no" />
      <Var Name='SMB' Fabric='Diff' ATag='16' BTag='17' DefaultNet="no" />
      <Var Name='VEM' Fabric='Common' ATag='100' BTag='' DefaultNet="no" />
  </VLANs>

<!-- VNIC template definitions -->
  <VNICtemplate>
      <Var Name='Mgmt' MTU='1500' Fabric='A-B' MACpool='VSPEX-99-MAC' Qos=''
VLAN='Mgmt' Order='1' Type='updating-template' Native='yes' Org='root' />
      <Var Name='CSV' MTU='9000' Fabric='A-B' MACpool='VSPEX-99-MAC' Qos='' VLAN='CSV'
Order='2' Type='updating-template' Native='yes' Org='root' />
      <Var Name='LiveMigration' MTU='9000' Fabric='A-B' MACpool='VSPEX-99-MAC'
Qos='LiveMigration' VLAN='LiveMigration' Order='3' Type='updating-template'
Native='yes' Org='root' />
      <Var Name='VMaccess' MTU='1500' Fabric='B-A' MACpool='VSPEX-99-MAC' Qos=''
VLAN='VMaccess' Order='4' Type='updating-template' Native='no' Org='root' />
      <Var Name='ClusComm' MTU='1500' Fabric='B-A' MACpool='VSPEX-99-MAC' Qos=''
VLAN='ClusComm' Order='5' Type='updating-template' Native='no' Org='root' />
      <Var Name='iSCSI-A' MTU='9000' Fabric='A-B' MACpool='VSPEX-99-MAC' Qos='iSCSI'
VLAN='iSCSI' Order='6' Type='updating-template' Native='no' Org='root' />
      <Var Name='iSCSI-B' MTU='9000' Fabric='B-A' MACpool='VSPEX-99-MAC' Qos='iSCSI'
VLAN='iSCSI' Order='7' Type='updating-template' Native='no' Org='root' />
      <Var Name='SMB-A' MTU='9000' Fabric='A-B' MACpool='VSPEX-99-MAC' Qos='iSCSI'
VLAN='SMB' Order='8' Type='updating-template' Native='no' Org='root' />
      <Var Name='SMB-B' MTU='9000' Fabric='B-A' MACpool='VSPEX-99-MAC' Qos='iSCSI'
VLAN='SMB' Order='9' Type='updating-template' Native='no' Org='root' />
      <Var Name='VEM' MTU='1500' Fabric='B-A' MACpool='VSPEX-99-MAC' Qos='' VLAN='VEM'
Order='10' Type='updating-template' Native='no' Org='root' />
  </VNICtemplate>

<!-- Virtual HBA templates -->
  <VHBAtemplate>
      <Var Name='VSPEX-99-FabA' Descr='Fabric A vHBA' Fabric='A' VSAN='default'
Type='updating-template' WWNpool='VSPEX-99-WWPN' Qos='' Org='VSPEX' />
      <Var Name='VSPEX-99-FabB' Descr='Fabric B vHBA' Fabric='B' VSAN='default'
Type='updating-template' WWNpool='VSPEX-99-WWPN' Qos='' Org='VSPEX' />
  </VHBAtemplate>

<!-- Boot Policy -->
<!-- Order of Vars specifies order of devices in boot policy -->
```

```xml
<!-- Type Local can have device1 equal to cdrom, localdisk, or floppy  -->
  <BootPolicy>
    <PolicyName Name='VSPEX-SAN-A-Boot' Descr='Fibre Channel Boot Fabric A'
Org='VSPEX' >
      <Var Type='Local' Device1='cdrom' Device2='' PrimaryFabric='' />
      <Var Type='VHBA' Device1='FabChn-A' Device2='FabChn-B' PrimaryFabric='A' />
    </PolicyName>
    <PolicyName Name='VSPEX-SAN-B-Boot' Descr='Fibre Channel Boot Fabric B'
Org='VSPEX' >
      <Var Type='Local' Device1='cdrom' Device2='' PrimaryFabric='' />
      <Var Type='VHBA' Device1='FabChn-B' Device2='FabChn-A' PrimaryFabric='B' />
    </PolicyName>
  </BootPolicy>

<!-- Service Profile Template -->
  <SPTemplate>
    <Template>
      <Name>VSPEX-99-BootA </Name>
      <Descr>VSPEX-99 Boot from SAN Fabric A </Descr>
      <BIOSProfileName> </BIOSProfileName>
      <BootPolicyName>VSPEX-SAN-A-Boot </BootPolicyName>
      <LocalDiskPolicy>NoLocal </LocalDiskPolicy>
      <MgmtIPpool>MgmtIP </MgmtIPpool>
      <PowerPolicyName>NoCap </PowerPolicyName>
      <ScrubPolicyName>NoScrub </ScrubPolicyName>
      <UUIDpool>VSPEX-99-UUID </UUIDpool>
      <MaintPolicyName>UserAck </MaintPolicyName>
      <HostFwPolicyName> </HostFwPolicyName>
      <MgmtAccessPolicyName> </MgmtAccessPolicyName>
      <MgmtFwPolicyName> </MgmtFwPolicyName>
      <StatsPolicyName>default </StatsPolicyName>
      <Org>VSPEX </Org>
      <WwnnPoolName>VSPEX-99-WWNN </WwnnPoolName>
      <VNICs>
        <Var Name='Mgmt' Templ='Mgmt' />
        <Var Name='LiveMigration' Templ='LiveMigration' />
        <Var Name='CSV' Templ='CSV' />
        <Var Name='VMaccess' Templ='VMaccess' />
        <Var Name='ClusComm' Templ='ClusComm' />
        <Var Name='VEM' Templ='VEM' />
      </VNICs>
      <VHBAs>
        <Var Name='FabChn-A' Templ='VSPEX-99-FabA'/>  <!-- Name must match value of
Devicex in boot policy -->
        <Var Name='FabChn-B' Templ='VSPEX-99-FabB'/>
      </VHBAs>
    </Template>
    <Template>
      <Name>VSPEX-99-BootB </Name>
      <Descr>Boot from SAN Fabric B </Descr>
      <BIOSProfileName> </BIOSProfileName>
      <BootPolicyName>VSPEX-SAN-B-Boot </BootPolicyName>
      <LocalDiskPolicy>NoLocal </LocalDiskPolicy>
```

```
    <MgmtIPpool>MgmtIP </MgmtIPpool>
    <PowerPolicyName>NoCap </PowerPolicyName>
    <ScrubPolicyName>NoScrub </ScrubPolicyName>
    <UUIDpool>VSPEX-99-UUID </UUIDpool>
    <MaintPolicyName>UserAck </MaintPolicyName>
    <HostFwPolicyName> </HostFwPolicyName>
    <MgmtAccessPolicyName> </MgmtAccessPolicyName>
    <MgmtFwPolicyName> </MgmtFwPolicyName>
    <StatsPolicyName>SPT-Test </StatsPolicyName>
    <Org>VSPEX </Org>
    <WwnnPoolName>VSPEX-99-WWNN </WwnnPoolName>
    <VNICs>
      <Var Name='Mgmt' Templ='Mgmt' />
      <Var Name='LiveMigration' Templ='LiveMigration' />
      <Var Name='CSV' Templ='CSV' />
      <Var Name='VMaccess' Templ='VMaccess' />
      <Var Name='ClusComm' Templ='ClusComm' />
      <Var Name='VEM' Templ='VEM' />
    </VNICs>
    <VHBAs>
      <Var Name='FabChn-B' Templ='VSPEX-99-FabB' />X
      <Var Name='FabChn-A' Templ='VSPEX-99-FabA' />
    </VHBAs>
  </Template>
</SPTemplate>

<ServiceProfile>
  <Var Name='VSPEX-01' Templ='VSPEX-99-BootA' Org='VSPEX' />
  <Var Name='VSPEX-02' Templ='VSPEX-99-BootB' Org='VSPEX' />
  <Var Name='VSPEX-03' Templ='VSPEX-99-BootA' Org='VSPEX' />
  <Var Name='VSPEX-04' Templ='VSPEX-99-BootB' Org='VSPEX' />
  <Var Name='VSPEX-05' Templ='VSPEX-99-BootA' Org='VSPEX' />
  <Var Name='VSPEX-06' Templ='VSPEX-99-BootB' Org='VSPEX' />
</ServiceProfile>

</VSPEX>
```

Add-UcsHyperVFeatures.ps1

```
#
Write-Host ""
Write-Host "Install the MPIO and Failover Clustering features and the Hyper-V role"
Write-Host ""
Write-Host -ForegroundColor Yellow "Installing Hyper-V will cause the system to
reboot"
Write-Host ""

$srvr = Read-Host "Enter computer name of server on which to install MPIO"

Write-Host ""
Write-Host "Installing the MPIO feature"
Install-WindowsFeature -Name Multipath-IO -ComputerName $srvr
-IncludeManagementTools
```

```
Invoke-Command -ComputerName $srvr -ScriptBlock `
{
    Write-Host "Add new vendor and product IDs for MPIO"
#    Values for EMC VNX
    $trash = New-MSDSMSupportedHw -VendorId "DGC" -ProductId "LUNZ"
    $trash = New-MSDSMSupportedHw -VendorId "DGC" -ProductId "VDISK"
    $trash = New-MSDSMSupportedHw -VendorId "DGC" -ProductId "RAID 0"
    $trash = New-MSDSMSupportedHw -VendorId "DGC" -ProductId "RAID 1"
    $trash = New-MSDSMSupportedHw -VendorId "DGC" -ProductId "RAID 10"
    $trash = New-MSDSMSupportedHw -VendorId "DGC" -ProductId "RAID 5"
    $trash = New-MSDSMSupportedHw -VendorId "DGC" -ProductId "VRAID"
    Write-Host "List of configured vendor and product IDs"
    Get-MSDSMSupportedHW | Select VendorId, ProductId | ft
}

Write-Host ""
Write-Host "Installing the Failover Clustering feature"
Install-WindowsFeature -Name Failover-Clustering -ComputerName $Srvr
-IncludeManagementTools

Write-Host ""
Write-Host "Installing the Hyper-V role"
Install-WindowsFeature -Name Hyper-V -ComputerName $Srvr -IncludeManagementTools
-Restart
```

Set-UcsHyperVAdapters.ps1

```
# Note that the $ucsIP variable needs to be changed to reflect customer environment

$ucsIP     = "192.168.14.100"

if ((Get-Module | Where {$_.Name -ilike "CiscoUcsPS"}).Name -ine "CiscoUcsPS")
    {
    Write-Host "Loading Module: Cisco UCS PowerTool Module"
    Import-Module CiscoUcsPs
    }

$trash = set-ucspowertoolconfiguration -supportmultipledefaultucs $false

# Connect to UCSM

$ucsCreds = Get-Credential
$UCSMHandle = Connect-Ucs $UcsIP $ucsCreds

Write-Host ""
Write-Host -ForegroundColor Yellow "Entered name of host must match case of service
profile name"
$srvr = Read-Host "Enter the name of the Hyper-V host to target"
Write-Host ""
```

```
[int]$hostNum = Read-Host "Enter a numeric value between 1-254 to use as the host
number"

Write-Host ""
Write-Host "Not all NICs should have their IP address altered, e.g. Mgmt and iSCSI
boot NICs"
$in = Read-Host "Enter a comma separated list of NICs to ignore"
$in2 = $in -replace " ",""
$ignoreNic = $in2 -split ","

Write-Host ""
$org = Read-Host "Enter Sub-Organization name of Service Profile, or 'root'"
If ($org.Length -eq 0) {$org = "root"}
$orgLevel = Get-UcsOrg -Name $org
$svcProfile = $orgLevel.DN + "/" + $srvr
Write-Host ""

# Retrieve table of NICs from the UCS Profile

$ducsVnics = Get-UcsVnic -ServiceProfile $svcProfile
If ($ducsVnics.length -eq 0)
{
    Write-Host -ForegroundColor Red "Invalid Service Profile name - $svcProfile"
    Disconnect-Ucs
    Exit
}

# This is a special check to remove the dyanmic virtual function vNICs created by
having VM-FEX defined
$ucsVnics = @()
Foreach ($d in $ducsVnics)
{
    If ($d.addr -ne "derived")
    {
        $ucsVnics +=, $d
    }
}

Write-Host "$srvr has the following vNICs"
$ucsVnics.Name

$vlans = Get-UcsLanCloud | Get-UcsVlan | Select Name, Id
$assignedIP = @()

# Rename the NICs on the server to match the NIC name of the service profile
# If NIC is not one entered to be ignored, change the IP address

ForEach ($u in $UcsVnics)
{
    $adapterConfig = (Get-WMIobject Win32_NetworkAdapterConfiguration -namespace
"root\CIMV2" -computername $srvr | `
        Where-Object {$_.MACaddress -eq $u.Addr})
```

```
    $hostNic = (Get-WMIobject Win32_NetworkAdapter -computername $srvr |
Where-Object {$_.Index -eq $adapterConfig.Index})
    If ($hostNic.NetconnectionID -ne $u.name)
    {
        $tmp = $hostNic.NetconnectionID ; $tmp_1 = $u.Name
        Write-Host "Changing NIC $tmp to be named $tmp_1"
        $hostNic.NetconnectionID=$u.Name
        $trash = $hostNic.Put()
    }
    $check = $FALSE

    Foreach ($ig in $ignoreNic)
    {
        If ($ig -eq $u.Name) {$check = $TRUE}
    }

    If (!$check)
    {
        Foreach ($v in $vlans)
        {
            If ($v.Name -eq $u.name)
            {
                $adapterConfig.DHCPenabled = $False
              $adapterConfig.SetDynamicDNSRegistration($false) | out-null
                $newIP = "192.168." + $v.Id + "." + $hostNum
                Foreach ($aIP in $assignedIP)
                {
                    If ($newIP -eq $aIP)
                    {
                        $octets = ($newIP.split("."))
                        [int]$lastOctet = $octets[3]
                        $lastOctet++
                        $newip = $octets[0] + "." + $octets[1] + "." + $octets[2] +
"." + $lastOctet
                    }
                }
                $adapterConfig.enablestatic($newIP,"255.255.255.0") | out-null
                $assignedIP +=, $newIP
                Write-Host $u.Name "new IP > $newIP"
            }
        }
    }
}
```

Set-UcsHyperVRemoteMgmt.ps1

```
#
#  Set-UcsHyperVRemoteMgmt.ps1
```

```
#
# This script works on a variety of settings that are easiest done from the
# local machine to make it remotely manageable by a management workstation.
#
# To find rule names
#    Get a list of possible groups
#    Get-NetFirewallRule | Select DisplayGroup -Unique | Sort DisplayGroup
#
# To list the applicable rules that may be set.
#    Get-NetFirewallRule | Where { $_.DisplayGroup -Eq "Remote Volume Management"} |
Format-Table Name

# Ensure Server Manager remoting is enabled
Configure-SMRemoting.exe -Enable

# Set some firewall rules

#  Enable ping requests in and out
Set-NetFirewallRule -Name "FPS-ICMP4-ERQ-In" -Enabled True -Profile Any
Set-NetFirewallRule -Name "FPS-ICMP6-ERQ-In" -Enabled True -Profile Any
Set-NetFirewallRule -Name "FPS-ICMP4-ERQ-Out" -Enabled True -Profile Any
Set-NetFirewallRule -Name "FPS-ICMP6-ERQ-Out" -Enabled True -Profile Any

#  Enable remote volume management - firewall rules need to be set on both
#  source and destination computers
#  ***NOTE*** Policy must also be set on system to "Allow remote access
#  to the Plug and Play interface"
#  This is done with gpedit.msc locally or gpedit for domain policy
Set-NetFirewallRule -Name "RVM-VDS-In-TCP" -Enabled True -Profile Any
Set-NetFirewallRule -Name "RVM-VDSLDR-In-TCP" -Enabled True -Profile Any
Set-NetFirewallRule -Name "RVM-RPCSS-In-TCP" -Enabled True -Profile Any

#  Enable DCOM management requests in
Set-NetFirewallRule -Name "ComPlusNetworkAccess-DCOM-In" -Enabled True -Profile Any

#  Enable remote service management
Set-NetFirewallRule -Name "RemoteSvcAdmin-In-TCP" -Enabled True -Profile Any
Set-NetFirewallRule -Name "RemoteSvcAdmin-NP-In-TCP" -Enabled True -Profile Any
Set-NetFirewallRule -Name "RemoteSvcAdmin-RPCSS-In-TCP" -Enabled True -Profile Any

#  Enable Remote Event Log Management
Set-NetFirewallRule -Name "RemoteEventLogSvc-In-TCP" -Enabled True -Profile Any
Set-NetFirewallRule -Name "RemoteEventLogSvc-NP-In-TCP" -Enabled True -Profile Any
Set-NetFirewallRule -Name "RemoteEventLogSvc-RPCSS-In-TCP" -Enabled True -Profile
Any

#  Enable Remote Scheduled Tasks Management
Set-NetFirewallRule -Name "RemoteTask-In-TCP" -Enabled True -Profile Any
Set-NetFirewallRule -Name "RemoteTask-RPCSS-In-TCP" -Enabled True -Profile Any

#  Enable Windows Firewall Remote Management
Set-NetFirewallRule -Name "RemoteFwAdmin-In-TCP" -Enabled True -Profile Any
Set-NetFirewallRule -Name "RemoteFwAdmin-RPCSS-In-TCP" -Enabled True -Profile Any
```

```
#  Enable WMI management requests in
Set-NetFirewallRule -Name "WMI-WINMGMT-In-TCP" -Enabled True -Profile Any

#  Enable Remote Shutdown
Set-NetFirewallRule -Name "Wininit-Shutdown-In-Rule-TCP-RPC" -Enabled True -Profile
Any

#  Set some services to automatically start and start them.
Set-Service -Name PlugPlay -StartupType Automatic
Start-Service PlugPlay
Set-Service -Name RemoteRegistry -StartupType Automatic
Start-Service RemoteRegistry
Set-Service -Name vds -StartupType Automatic
Start-Service vds

#  Enable Remote Desktop
(Get-WmiObject Win32_TerminalServiceSetting -Namespace
root\cimv2\TerminalServices).SetAllowTsConnections(1,1) | Out-Null
(Get-WmiObject -Class "Win32_TSGeneralSetting" -Namespace
root\cimv2\TerminalServices -Filter
"TerminalName='RDP-tcp'").SetUserAuthenticationRequired(0) | Out-Null

#  Enable Remote Desktop rules for all profiles
Set-NetfirewallRule -Name "RemoteDesktop-UserMode-In-TCP" -Enabled True -Profile Any
Set-NetfirewallRule -Name "RemoteDesktop-UserMode-In-UDP" -Enabled True -Profile Any
```

EMC Sample Scripts

Master_Boot_Lun.xml

```
<StorageParams>
  <Array>VSPEX-VNX5400</Array>
  <UCSAddress>10.5.177.10</UCSAddress>
  <Servers>
    <Server>
      <ServerName>VSPEX-01</ServerName>
      <IPAddress>10.29.130.21</IPAddress>
        <luns>
          <label> Master_Boot_2012R2</label>
          <pool>VSPEX_Pool_01_R5</pool>
          <size>50GB</size>
        </luns>
    </Server>
  </Servers>
</StorageParams>
```

Storage_Luns.xml

```xml
<!--  This configuration file is used by the following three scripts:
  PrepMasterBoot-AddViaWWPN.ps1
  ProcessStorageRequests.ps1
  PostClone_AddViaWWPN.ps1

This is just a sample.  It must be altered to reflect the customer configuration.

Create <Server> objects for each physical blade, plus one extra for the Master
Boot image LUN.  It is recommended to alternate the pool used for creating the
LUNs so half the servers boot from one pool and the other half boot from the
Other.
-->

<StorageParams>
  <Array> VNX5400</Array>
  <UCSAddress>10.5.177.10</UCSAddress>
  <Servers>
    <Server>
      <ServerName>VSPEX-01</ServerName>
      <IPAddress>10.29.130.31</IPAddress>
        <luns>
          <label>VSPEX-01</label>
        <pool>VSPEX_Pool_1</pool>
        <size>50GB</size>
          </luns>
    </Server>
    <Server>
      <ServerName>VSPEX-02</ServerName>
      <IPAddress>10.29.130.32</IPAddress>
        <luns>
          <label>VSPEX-02</label>
        <pool> VSPEX_Pool_2</pool>
        <size>50GB</size>
          </luns>
    </Server>
    <Server>
      <ServerName>VSPEX-03</ServerName>
      <IPAddress>10.29.130.33</IPAddress>
        <luns>
          <label>VSPEX-03</label>
        <pool> VSPEX_Pool_1</pool>
        <size>50GB</size>
          </luns>
    </Server>
    <Server>
      <ServerName>VSPEX-04</ServerName>
      <IPAddress>10.29.130.34</IPAddress>
        <luns>
          <label>VSPEX-04</label>
        <pool> VSPEX_Pool_2</pool>
        <size>50GB</size>
```

```
          </luns>
      </Server>
      <Server>
        <ServerName>VSPEX-05</ServerName>
        <IPAddress>10.29.130.35</IPAddress>
          <luns>
            <label>VSPEX-05</label>
         <pool> VSPEX_Pool_1</pool>
         <size>50GB</size>
            </luns>
      </Server>
      <Server>
        <ServerName>VSPEX-06</ServerName>
        <IPAddress>10.29.130.36</IPAddress>
          <luns>
            <label>VSPEX-06</label>
         <pool> VSPEX_Pool_2</pool>
         <size>50GB</size>
            </luns>
      </Server>
    </Servers>
</StorageParams>
```

PrepMasterBoot-AddViaWWPN.ps1

```
#-------------------------------------------------------------------------------
# Filename:       PrepMasterBoot_AddViaWWPN.ps1
# Description:    Set up Cisco UCS ServiceProfile to do Boot from SAN from VNX5400
# Input file:     Master_Boot_Lun.xml
#-------------------------------------------------------------------------------
#
# Uses an XML file with the following schema.  This same schema is used by
#  - PrepMastBoot-AddViaWWPN.ps1
#  - Process Storage Requests.ps1
#  - PostClone_AddViaWWPN.ps1
#
# <StorageParams>
#   <Array>VNX5400</Array>
#   <UCSAddress>10.5.177.10</UCSAddress>
#   <Servers>
#     <Server>
#       <ServerName>VSPEX-01</ServerName>
#       <IPAddress>192.168.10.31</IPAddress>
#         <luns>
#           <label>Master_Boot_2012R2</label>
#           <pool>VSPEX_Pool_1</pool>
#           <size>50GB</size>
#         </luns>
#     </Server>
#   </Servers>
# </StorageParams>
#
#-------------------------------------------------------------------------------
```

```
$global:rootPath = Split-Path -Parent $MyInvocation.MyCommand.Path
$myxmlfile = $global:rootPath + "\Master_Boot_Lun.xml"

Function ReadStorageConfig ([String]$filename) {
    $xmlConfigFile = [xml](Get-Content $filename )
    $global:StorageConfig = $xmlConfigFile.SelectSingleNode( '/StorageParams' )
}

ReadStorageConfig $myxmlfile

Import-Module CiscoUcsPS
Import-Module ESIPSToolkit

Function LUNExists {
   param ($TGTLUN)
   $Val = Get-EmcLUN $TGTLUN -Silent
   if ($Val -eq $null) {return $false} else {return $true}
}

Function reghostexists {
    param ($tgthost)
    $val = Get-EmcStorageRegisteredHost $tgthost
    If ($Val -eq $null) {Return $false}
    Else {Return $true}
}

$StorageArray = Get-EMCStorageSystem -ID $global:StorageConfig.Array -Silent

If ($StorageArray -eq $null)
{
   Write-Host "ERROR: Array" $Array "is not known or registered under that name."
   Exit 1
}


Update-EmcSystem $StorageArray

# Prompt user for connection to UCS environment
If ($UCS -eq $null) {$UCS = Connect-Ucs $global:StorageConfig.UCSAddress}


ForEach ($entry in $global:StorageConfig.Servers.Server)
{
    ForEach ($lun in $entry.luns)
    {
        Write-Host $entry.Servername, $lun.label
    }
}
# Check for pre-existing LUN
If (LUNExists $global:StorageConfig.Servers.Server.luns.label)
{ # We present the LUN
```

```
    $MyServiceProfile = Get-UcsServiceProfile | where {$_.Name -eq
$global:StorageConfig.Servers.Server.ServerName}
    If ($MyServiceProfile -eq $null)
    {
        Write-Host "ERROR: Cannot find ServiceProfile"
$global:StorageConfig.Servers.Server.ServerName
        exit 1
    }
    Else
    {
#
# Extract out the WWPN initiator information for the Service Profile
#
        $MyvHBAs = Get-UcsVhba -ServiceProfile $MyServiceProfile
#
# Get the Gold Master that we plan to use
#
        $MasterLUN = get-EMCLun -ID $global:StorageConfig.Servers.Server.luns.label
-BlockStorageSystem $StorageArray
#
# Add all the initiators from the Service Profile to the Storage Group on the VNX
#
        ForEach ($vHBA in $MyvHBAs)
    {
            $HostRegistration = $vHBA.NodeAddr + ":" + $vHBA.Addr
              If (reghostexists $global:StorageConfig.Servers.Server.ServerName)
              {
                    $rg=get-emcstorageregisteredhost
$global:StorageConfig.Servers.Server.ServerName
                    Write-Host "New Init" $HostRegistration
                    New-EmcStorageRegisteredInitiator -registeredhost $rg -InitiatorIds
$HostRegistration
              }
              Else
              {
                    Write-Host "New Host" $HostRegistration
          New-EMCStorageRegisteredHost -StorageSystem $StorageArray -HostName
$global:StorageConfig.Servers.Server.ServerName -IpAddress
$global:StorageConfig.Servers.Server.IPAddress -HostBusAdapterIds $HostRegistration
              }
        }

    }
    If (LUNExists $MasterLUN)
     {
          Write-Host "unmask lun" $masterlun
          Set-EmcLunAccess -Lun $MasterLUN -InitiatorId $Hostregistration -HostName
$global:StorageConfig.Servers.Server.ServerName -HostIPAddress
$global:StorageConfig.Servers.Server.IPAddress -Available
     }
    Else
    { # We Fail, because the LUN cannot be found
       Write-host "ERROR: Cannot find the LUN:" $MasterLUN
```

```
        Exit 1
    }
}
```

ProcessStorageRequests.ps1

```
#-------------------------------------------------------------------------------
# Filename:      ProcessStorageRequests.ps1
# Description:   Create LUNs based on xml file
# Input file:    Storage_Luns.xml
#
#-------------------------------------------------------------------------------
#
# Uses an XML file with the following schema.  This same schema is used by
#  - PrepMastBoot-AddViaWWPN.ps1
#  - Process Storage Requests.ps1
#  - PostClone_AddViaWWPN.ps1
#
# <StorageParams>
#    <Array>VNX5400</Array>
#    <UCSAddress>10.5.177.10</UCSAddress>
#    <Servers>
#      <Server>
#        <ServerName>VSPEX-01</ServerName>
#        <IPAddress>192.168.10.31</IPAddress>
#          <luns>
#            <label>VSPEX-01</label>
#            <pool>VSPEX_Pool_1</pool>
#            <size>50GB</size>
#          </luns>
#      </Server>
#    </Servers>
# </StorageParams>
#
#-------------------------------------------------------------------------------

$global:rootPath = Split-Path -Parent $MyInvocation.MyCommand.Path
$myxmlfile = $global:rootPath + "\Storage_Luns.xml"

function ReadStorageConfig ([String]$filename) {
$xmlConfigFile = [xml](Get-Content $filename )
$global:StorageConfig = $xmlConfigFile.SelectSingleNode( '/StorageParams' )
}

ReadStorageConfig $myxmlfile

Import-Module ESIPSToolkit

function LUNExists {
    param ($TGTLUN)
```

```
    $Val = Get-EmcLUN $TGTLUN -Silent
    if ($Val -eq $null) {return $false} else {return $true}
}

$StorageArray = get-EMCStorageSystem -ID $global:StorageConfig.Array -silent

if ($StorageArray -eq $null)
{
    Write-Host "ERROR: Array" $Array "is not known or registered under that name."
    exit 1
}

Update-EmcSystem $StorageArray

function createluns {
    foreach ($entry in $global:StorageConfig.Servers.Server) {
        foreach ($lun in $entry.luns) {
            IF (LUNExists $lun.label)
            { Write-Host "LUN" $lun.label "already exists."}
            else
        {
# We need to create the LUN
            write-host "Creating LUN" $lun.label
            $pool = get-emcstoragepool $lun.pool
            $Size = invoke-expression $lun.size
            $NewLUN = New-EmcLun -Pool $pool -Name $lun.label -Capacity $Size
-Description $lun.label
            }
        }
    }
}

createluns
```

ProcessClones.ps1

```
#--------------------------------------------------------------------------------
# Filename:     ProcessClones.ps1
# Description:  Create Clones from Source LUN
# Input file:   ProcessClones.xml
#
#--------------------------------------------------------------------------------
#
# Uses an XML file with the following schema
# <StorageParams>
#   <CloneGroupName>BooT_Luns</CloneGroupName>
#   <VNXBlockSPAAddress>10.5.223.128</VNXBlockSPAAddress>
#   <SourceLUN>Master_Boot_2012R2</SourceLUN>
#     <TargetLUNs>
#       <lun>VSPEX-01</lun>
```

```
#        <lun>VSPEX-02</lun>
#        <lun>VSPEX-03</lun>
#        <lun>VSPEX-04</lun>
#        <lun>VSPEX-05</lun>
#        <lun>VSPEX-06</lun>
#     </TargetLUNs>
# </StorageParams>
#
#-------------------------------------------------------------------------------

$global:rootPath = Split-Path -Parent $MyInvocation.MyCommand.Path
$myxmlfile = $global:rootPath + "\ProcessClones.xml"
function ReadStorageConfig ([String]$filename) {
    $xmlConfigFile = [xml](cat $filename )
    $global:StorageConfig = $xmlConfigFile.SelectSingleNode( '/StorageParams' )
    cls
    if ($global:StorageConfig.TargetLUNs.lun.count -gt "8")
    {
        write-host "There are" $StorageConfig.TargetLUNs.lun.count "clone targets,
only 8 are supported concurrently"
        start-sleep 10
        exit
    }
}


ReadStorageConfig $myxmlfile

write-host "Warning You are about to overwrite the following LUNs:" -foregroundcolor
Black -backgroundcolor Yellow
foreach ($entry in $global:StorageConfig.TargetLUNs.LUN) {
    write-host $entry
}

write-host "With the contents of LUN" $global:StorageConfig.SourceLUN
$prompt =  Read-Host "Please type 'overwrite' to continue"
if ($prompt -ne "overwrite")
{
    write-host $prompt "is not valid, exiting"
    exit 1
}

function CloneStart {
    $clonesource = get-emclun $global:StorageConfig.SourceLUN
    write-host "Creating Clone Group" $global:StorageConfig.CloneGroupName
    write-host "Adding Source LUN" $clonesource
    naviseccli -address $global:StorageConfig.VNXBlockSPAAddress clone
-createclonegroup -name $global:StorageConfig.CloneGroupName -Luns
$clonesource.ArrayLunID -o

    foreach ($entry in $global:StorageConfig.TargetLUNs.LUN)
    {
        $clonetarget=get-emclun $entry
        write-host "Adding Clone Target LUN" $clonetarget
```

```
            naviseccli -address $global:StorageConfig.VNXBlockSPAAddress clone -addclone
-Name $global:StorageConfig.CloneGroupName -Luns $clonetarget.ArrayLunId -syncrate
high -o
    }
}

function CloneSyncCheck {
    $synchronized=naviseccli -address $global:StorageConfig.VNXBlockSPAAddress clone
-listclone -name $global:StorageConfig.CloneGroupName -percentsynced | select-string
PercentSynced
    foreach ($entry in $synchronized)
    {
        $test=$entry -split ":"
        foreach ($sentry in $test[1])
        {
            $sentry=$sentry.trim()
            write-host $sentry "% Copied per Clone target"
            if ($sentry -ne "100")
            {
                start-sleep 20
                CloneSyncCheck
            }
        }
    }
}

function CloneFracture {
    $fracture=naviseccli -address $global:StorageConfig.VNXBlockSPAAddress clone
-listclone -name $global:StorageConfig.CloneGroupName | select-string CloneID
    foreach ($entry in $fracture)
    {
        $test=$entry -split ":"
        foreach ($sentry in $test[1])
        {
            $sentry=$sentry.trim()
            write-host "Fracturing Clone Target" $sentry
            naviseccli -address $global:StorageConfig.VNXBlockSPAAddress clone
-fractureclone -Name $global:StorageConfig.CloneGroupName -CloneId $sentry -o
        }
    }
}

function CloneDelete {
    $cdelete=naviseccli -address $global:StorageConfig.VNXBlockSPAAddress clone
-listclone -name $global:StorageConfig.CloneGroupName | select-string CloneID
    foreach ($entry in $cdelete)
    {
        $test=$entry -split ":"
        foreach ($sentry in $test[1])
        {
            $sentry=$sentry.trim()
            write-host "Deleting Clone Target" $sentry
```

```
            navisecli -address $global:StorageConfig.VNXBlockSPAAddress clone
-removeclone -Name $global:StorageConfig.CloneGroupName -CloneId $sentry -o
        }
    }
}

function CloneGroupDelete {
    write-host "Deleting Clone Group" $global:StorageConfig.CloneGroupName
    $cdelete=navisecli -address $global:StorageConfig.VNXBlockSPAAddress clone
-destroyclonegroup -name $global:StorageConfig.CloneGroupName -o
}

CloneStart
CloneSyncCheck
CloneFracture
CloneDelete
CloneGroupDelete
```

ProcessClones.xml

```
<StorageParams>
  <SourceLUN>Master_Boot_2012R2</SourceLUN>
  <CloneGroupName>Boot_Luns</CloneGroupName>
  <VNXBlockSPAAddress>10.5.223.128</VNXBlockSPAAddress>
    <TargetLUNs>
     <lun>VSEPX-01</lun>
     <lun>VSPEX-02</lun>
     <lun>VSPEX-03</lun>
     <lun>VSPEX-04</lun>
     <lun>VSPEX-05</lun>
     <lun>VSPEX-06</lun>
  </TargetLUNs>
</StorageParams>
```

PostClone_AddViaWWPN.ps1

```
#-------------------------------------------------------------------------------
# Filename:     PostClone_AddViaWWPN.ps1
# Description:  Set up Cisco UCS ServiceProfile to do Boot From SAN from VNX5400
# Input File:   Storage_Luns.xml
#-------------------------------------------------------------------------------
#
# Uses an XML file with the following schema.  This same schema is used by
#  - PrepMastBoot-AddViaWWPN.ps1
#  - Process Storage Requests.ps1
#  - PostClone_AddViaWWPN.ps1
#
# <StorageParams>
#   <Array>VNX5400</Array>
#   <UCSAddress>10.5.177.10</UCSAddress>
#   <Servers>
```

```
#       <Server>
#         <ServerName>VSPEX-01</ServerName>
#         <IPAddress>192.168.10.31</IPAddress>
#           <luns>
#              <label>VSPEX-01</label>
#              <pool>VSPEX_Pool_1</pool>
#              <size>50GB</size>
#           </luns>
#       </Server>
#     </Servers>
# </StorageParams>
#
#-------------------------------------------------------------------------------

$global:rootPath = Split-Path -Parent $MyInvocation.MyCommand.Path
$myxmlfile = $global:rootPath + "\Storage_Luns.xml"

function ReadStorageConfig ([String]$filename) {
    $xmlConfigFile = [xml](Get-Content $filename )
    $global:StorageConfig = $xmlConfigFile.SelectSingleNode( '/StorageParams' )
}

ReadStorageConfig $myxmlfile

Import-Module CiscoUcsPS
Import-Module ESIPSToolkit

function LUNExists {
    param ($TGTLUN)
    $Val = Get-EmcLUN $TGTLUN -Silent
    if ($Val -eq $null) {return $false} else {return $true}
}

function reghostexists {
    param ($tgthost)
    $val = get-emcstorageregisteredhost $tgthost
    if ($Val -eq $null) {return $false} else {return $true}
}

$StorageArray = get-EMCStorageSystem -ID $global:StorageConfig.Array -silent

if ($StorageArray -eq $null)
{
    Write-Host "ERROR: Array" $Array "is not known or registered under that name."
    exit 1
}


Update-EmcSystem $StorageArray

# Prompt user for connection to UCS environment
if ($UCS -eq $null) {$UCS = connect-ucs $global:StorageConfig.UCSAddress}
```

```
foreach ($entry in $global:StorageConfig.Servers.Server) {
    foreach ($lun in $entry.luns) {
        write-host $entry.Servername, $lun.label

# Check for pre-existing LUN
        IF (LUNExists $lun.label)
        {
# We present the LUN
            $MyServiceProfile = Get-UcsServiceProfile | where {$_.Name -eq
$entry.ServerName}
            if ($MyServiceProfile -eq $null)
            {
                Write-Host "ERROR: Cannot find ServiceProfile"
$global:StorageConfig.Servers.Server.ServerName
                exit 1
            }
            else
            {
#
# Extract out the WWPN initiator information for the Service Profile
#

                $MyvHBAs = Get-UcsVhba -ServiceProfile $MyServiceProfile

#
# Get the Boot LUN that we plan to use
#
                $BootLUN = get-EMCLun -ID $lun.label -BlockStorageSystem
$StorageArray

#
# Add all the initiators from the Service Profile to the Storage Group on the VNX
#

                foreach ($vHBA in $MyvHBAs)
                {
                    $HostRegistration = $vHBA.NodeAddr + ":" + $vHBA.Addr
                    if (reghostexists $entry.ServerName)
                    {
                        $rg=get-emcstorageregisteredhost $entry.ServerName
                        write-host "New Init" $HostRegistration
                        New-EmcStorageRegisteredInitiator -registeredhost $rg
-InitiatorIds $HostRegistration
                    }
                    else
                    {
                        write-host "New Host" $HostRegistration
                        New-EMCStorageRegisteredHost -StorageSystem $StorageArray
-HostName $entry.ServerName -IpAddress $entry.IPAddress -HostBusAdapterIds
$HostRegistration
                    }
                }
```

```
        }
        if (LUNExists $BootLUN)
        {
            write-host "unmask lun" $BootLun
            Set-EmcLunAccess -Lun $BootLUN -InitiatorId $Hostregistration
-HostName $entry.ServerName -HostIPAddress $entry.IPAddress -unAvailable
        }
        else
        {
# We Fail, because the LUN cannot be found

            Write-host "ERROR: Cannot find the LUN:" $BootLUN
            exit 1
        }
    }
  }
}
```

Cisco Nexus 1000V Installation

Cisco Nexus 1000V Series Switches provide a comprehensive and extensible architectural platform for virtual machine and cloud networking. The switches are designed to accelerate server virtualization and multi-tenant cloud deployments in a secure and operationally transparent manner for environments like Microsoft's Private Cloud. Download the distribution software from the location specified in the Software Revision table at the beginning of this document and expand it into a temporary directory.

Create Two Virtual Supervisor Module Virtual Machines

The Nexus 1000V runs as a pair of virtual machines for high availability purposes. The Nexus 1000V distribution contains an ISO file (nexus-1000v.5.2.1.SM1.5.1.iso) that is used in the creation of the virtual machines that will run the Nexus 1000V software. Copy it to the Virtual Machine Manager library. (The VMM library is a standard Windows share, so normal procedures for putting simple files into the share work. Refresh the library location after the copy is completed.

1. From an elevated PowerShell window on a Virtual Machine Manager machine, navigate to the directory containing the extracted contents of the Nexus 1000V distribution. Find the Register-Nexus1000VVSMTemplate.ps1 script and execute it.

   Note You must execute this script from the directory in which it is found. It assumes a specific directory hierarchy.

   Note You might have to set the execution policy to bypass to get the script to run.

```
PS C:\temp\vsmtemplate> Set-ExecutionPolicy bypass -force
PS C:\temp\vsmtemplate> .\Register-Nexus1000VVSMTemplate.ps1
Script: System.Management.Automation.InvocationInfo.MyCommand.Path
Loading System Center Virtual Machine Manager Powershell Module...
Powershell module loaded.
Gathering System Center Library Server Info...
Valid SC Library info found.
Validate Nexus 1000V VSM Template Components...
Template copy is in progress...
Refreshing SCVMM Library...
Registering Cisco VSM Template in SCVMM Library...
Cisco Nexus 1000V VSM Template registered successfully.
PS C:\temp\vsmtemplate>
```

2. From one of the Virtual Machine Manager consoles, select VMs and Services.

3. Double-click on Create Virtual Machine. This will launch a wizard to assist in creating a virtual machine to be used for installing the Nexus 1000V software image.
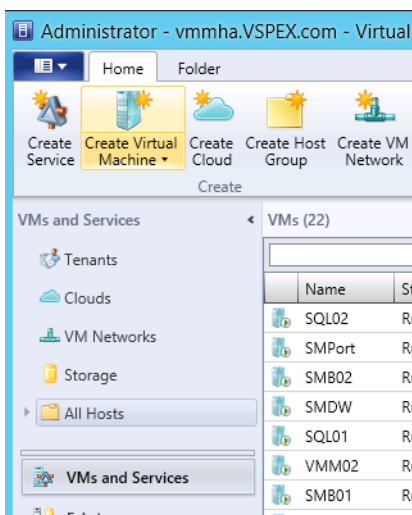
4. In the Select Source dialog window, select the radio button by Use an existing virtual machine, VM template, or virtual hard disk. Click Browse.

5. This launches a window that presents the contents of the VMM library. Select the Nexus1000V-VSM-Template and click OK.

6. Click Next to continue.
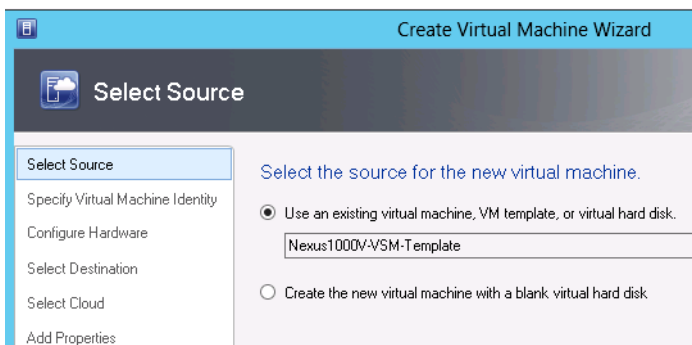
7. In the Specify Virtual Machine Identity dialog window, enter the name of the virtual machine.

8. Click Next to continue.
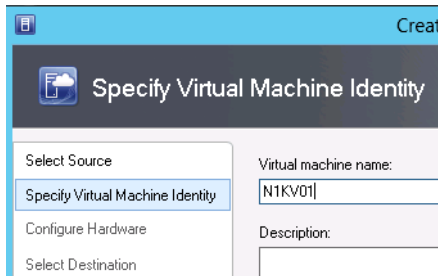
9. In the Configure Hardware dialog window, navigate to the Advanced settings and select Availability. Click the check box by Make this virtual machine highly available.

10. Click Next to continue.
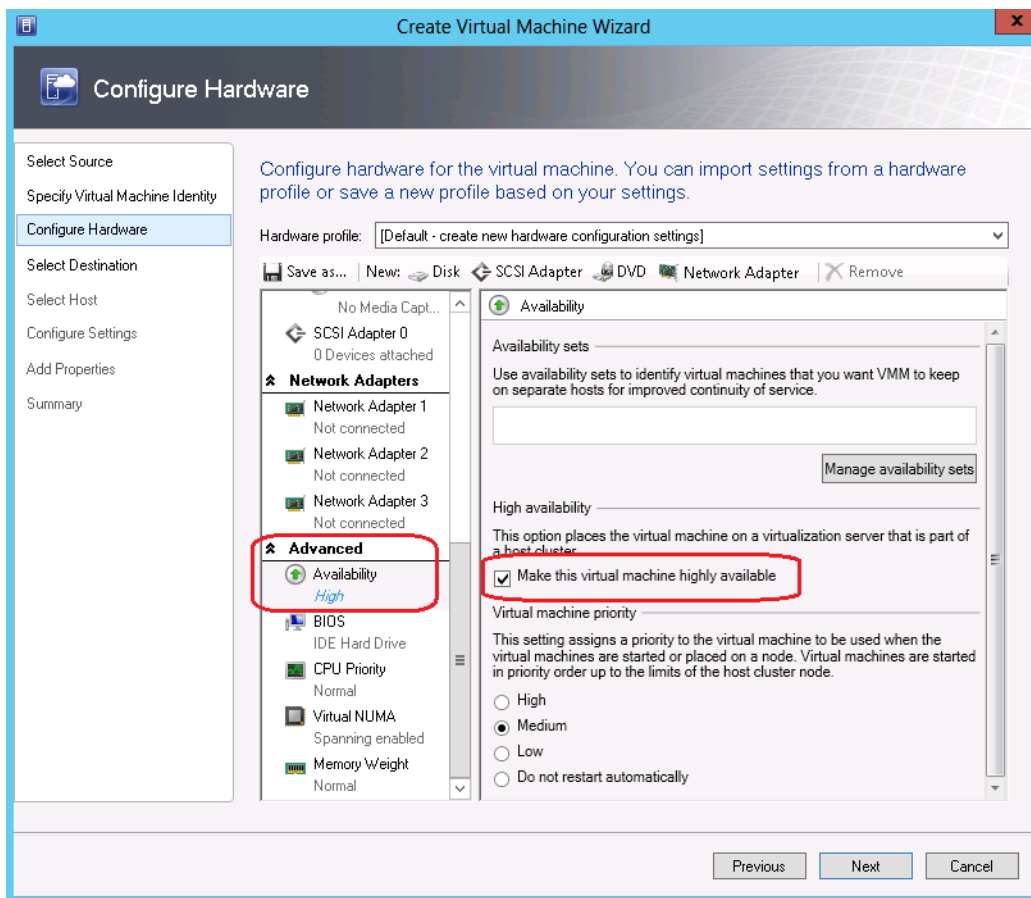
11. In the Select Destination dialog window, select the management group you have defined for your fabric management cluster.

12. Click Next to continue.
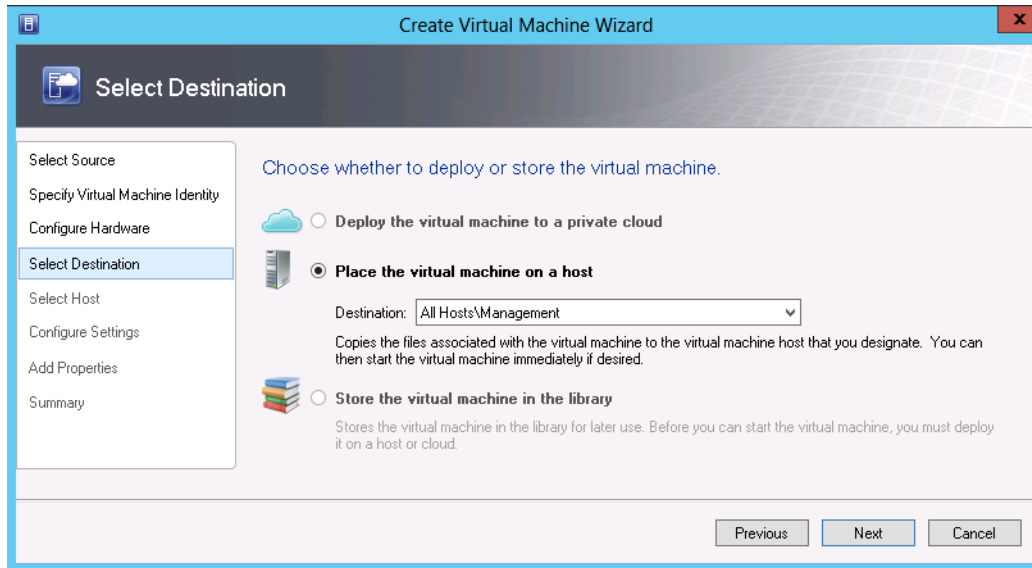
13. In the Select Host dialog window, select one of the fabric management hosts for deployment.

14. Click Next to continue.
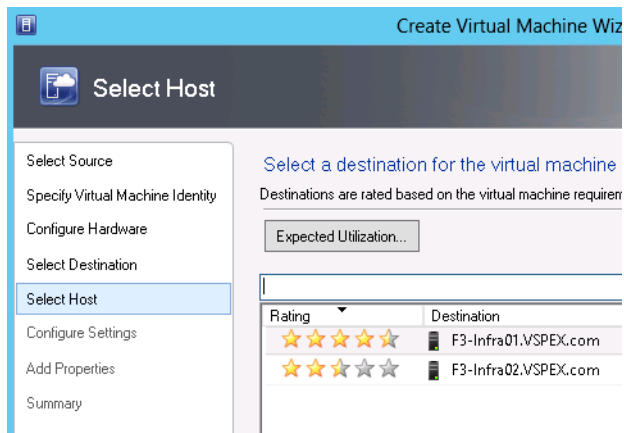
15. In the Configure Settings dialog window, ensure the Virtual machine path is pointing to the location you want to store the VM. This should be on one of the Cluster Shared Volumes.

16. Click Next to continue.
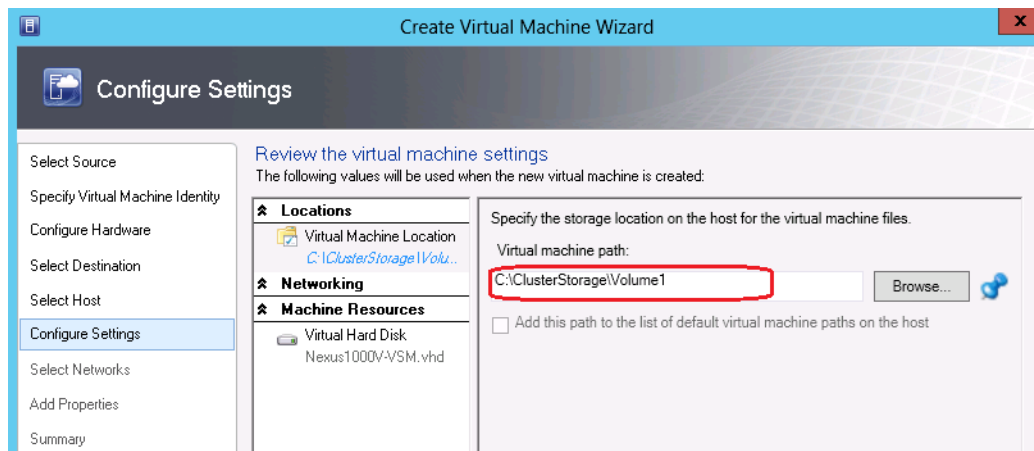
17. In the Select Networks dialog window ensure all network connections are to the VMaccess network.

18. Click Next to continue.

19. Click Next on the Add Properties dialog window.

20. Click Create on the Summary dialog window. Do not select the option to start the virtual machine.

21. Repeat the process to create a second VSM virtual machine.
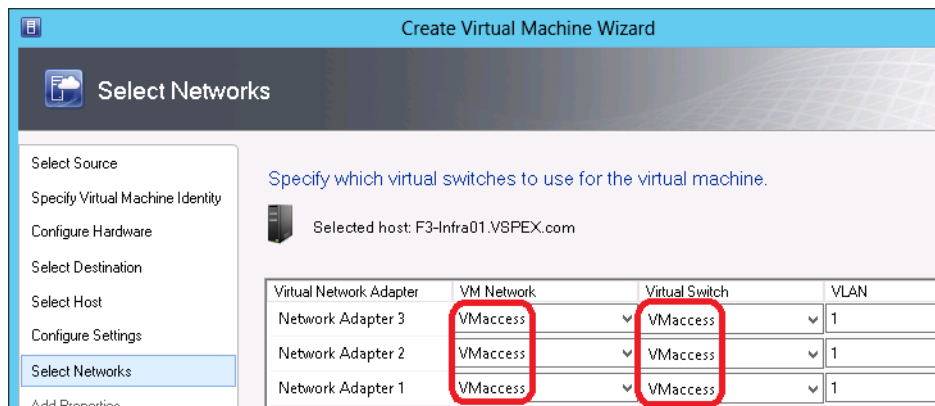
Configure the VSM

One of the components of the Nexus 1000V distribution media is an ISO file used for installing the software. As one of the first steps of this installation, you should have copied this ISO file to the VMM library.

> **Note** Make sure the name to be used as the switch name has a DNS and (optionally) an associated pointer (PTR) record in your DNS server. The switch name is a value entered during configuration, so it is not the name of the VMs. VMM uses DNS to find this VSM.

1. Within the VMM console, right-click the N1KV virtual machine, and select Properties.

2. On the Properties window, select Hardware Configuration. Select the Virtual DVD component.

3. Click the radio button by Existing ISO image file. VMM will display the ISO files within its library. Select the nexus-1000v.5.2.1.SM1.5.1.iso file.

4. If you have configured Constrained Delegation, click the check box by Share image file instead of copying it.

✎

Note Configuration of Constrained Delegation is covered in the section on setting up the VMM virtual machines.
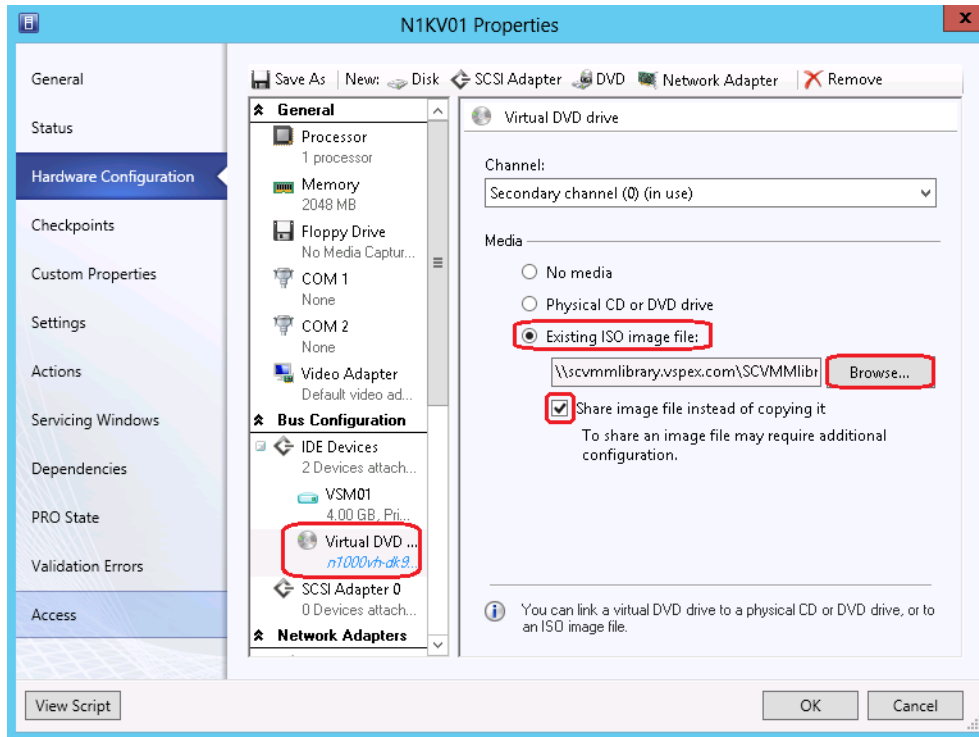
5. In VMM select the first N1KV virtual machine. Start it and connect to it through the console.

6. By default, Install Nexus 1000V and bring up the new image is highlighted. Either enter a return or let the timeout expire and the installation will begin.

7. There are two subsequent questions that will be answered with y automatically if you are not watching for them.
   – Do you want to format it (y/n)
   – Perform r/w tests (takes very long time) on target disks (y/n)
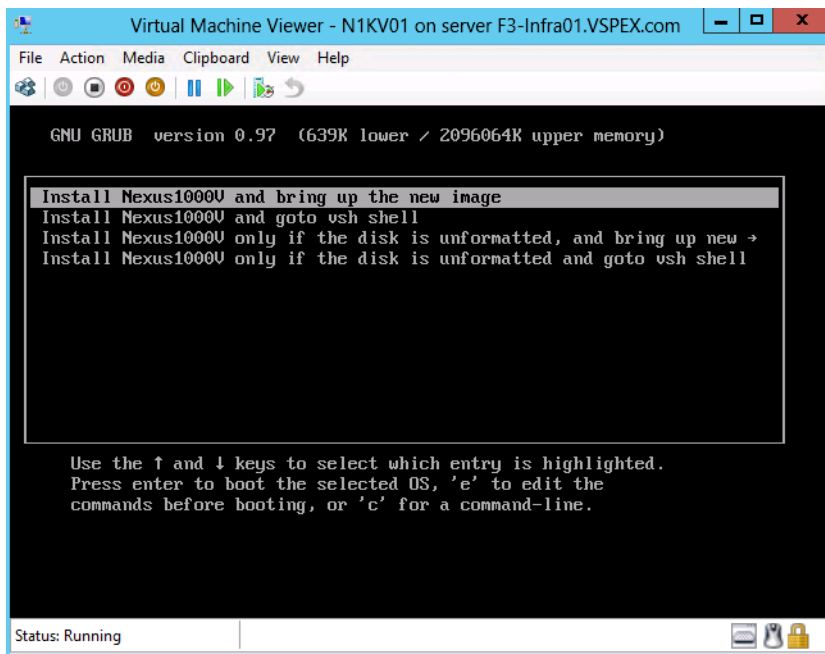
8. After some configuration, the system prompts for a new password for the admin account. Enter it and confirm it.

9. On the first VSM virtual machine, select primary for the HA role.

10. On the second VSM virtual machine, select secondary for the HA role.
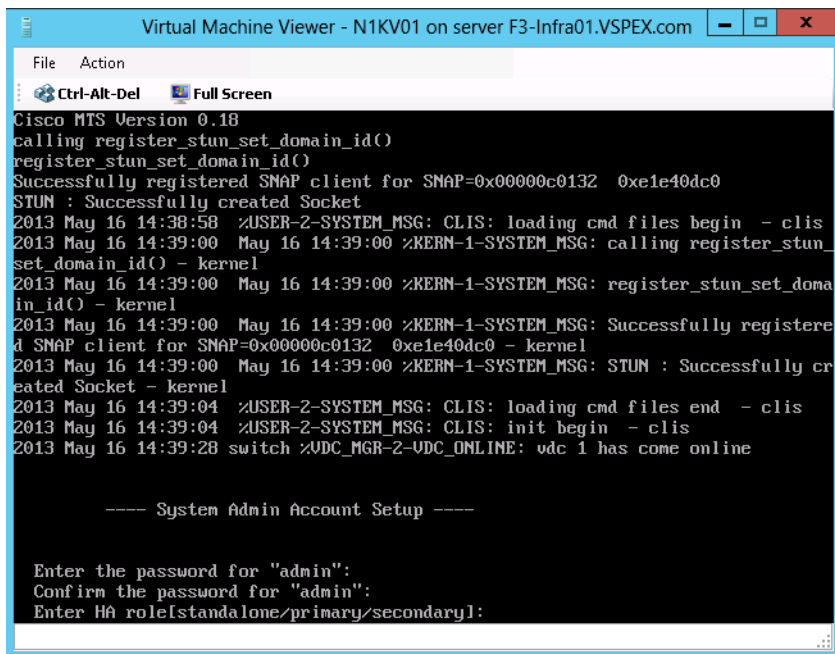
11. After selecting the HA role, you are asked for a domain id in the range 1-4095. This number is used when configuring HA, so it will be entered for both installs. <200>

12. The next question is asking to run the basic configuration dialog. Answer yes.

13. For the question to create another account, accept the default n.
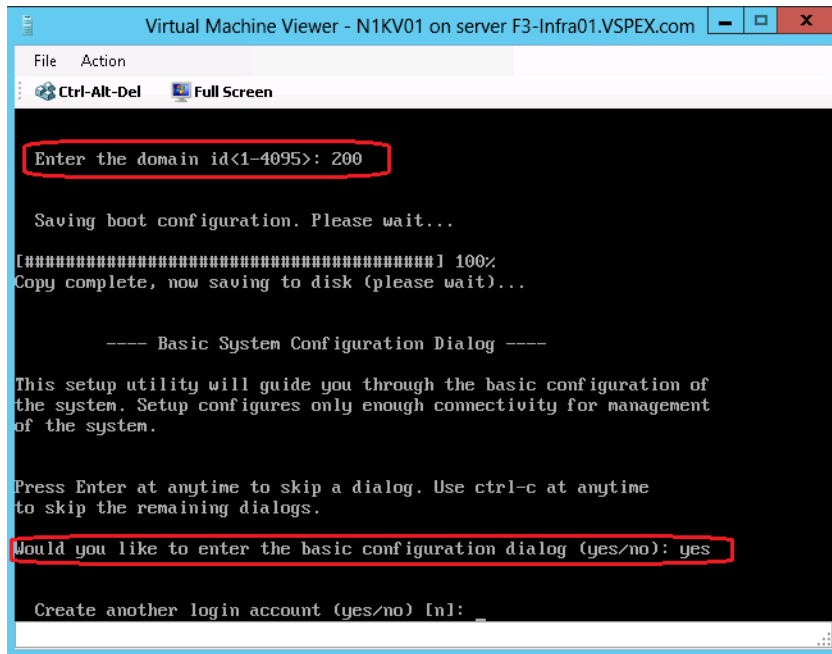
The next series of commands are answers similar to configuring a typical Nexus switch for out-of-band management.

14. Provide a character string to name the switch. (This is the name that needs the DNS entry). <Nexus1KV-Switch>

15. Configure out-of-bound management - y

16. Enter the IPv4 address that will be used for managing the switch. <10.29.130.95>

17. Enter the netmask for the address. <255.255.255.0>

18. Configure the default gateway - y

19. Enter the address for the default gateway. <10.29.130.1>

20. Configure advanced options - n

21. Reconfigure feature level - n

22. The next display is a summary of what has been entered and gives you the option to edit it if you need to change something.

23. When the configuration is saved, you are presented with a login prompt.

Note The name and IPv4 address is the name and address to be entered into your DNS.
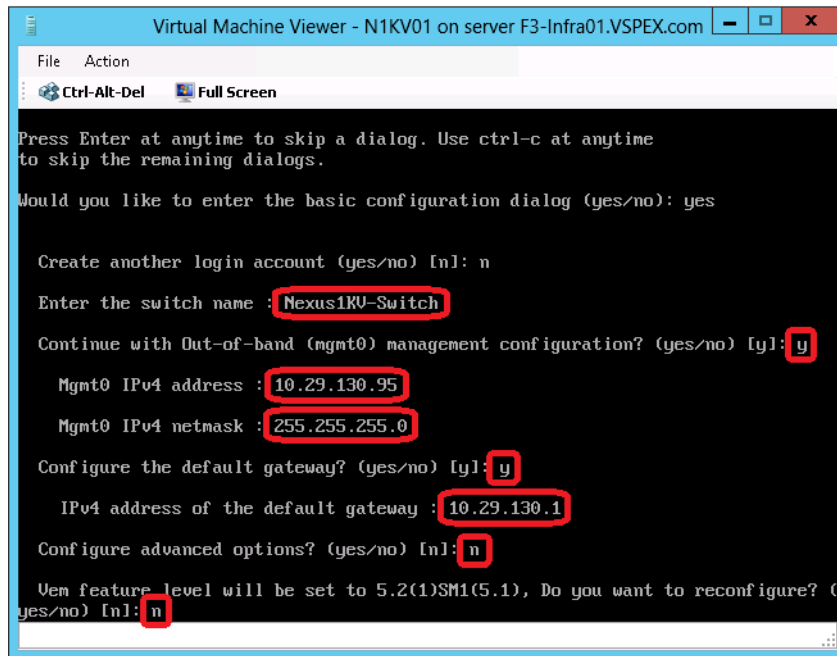
24. Repeat the process for the second VSM virtual machine, using the same admin password.

25. Answer the HA role with secondary.

26. It will ask to reboot and then ask for the domain number. Enter the same domain number as was entered when setting up the first VM.

27. The system will reboot and come up in standby mode.

28. From both nodes you should be able to issue the command show system redundancy status and receive a display as shown in the following screenshot.

29. This is from the standby N1KV.

```
Nexus1KV-Switch(standby)# show system redundancy status
Redundancy role
---------------
       administrative:    secondary
          operational:    secondary

Redundancy mode
---------------
       administrative:    HA
          operational:    HA

This supervisor (sup-2)
-----------------------
    Redundancy state:     Standby
    Supervisor state:     HA standby
      Internal state:     HA standby

Other supervisor (sup-1)
------------------------
    Redundancy state:     Active
    Supervisor state:     Active
      Internal state:     Active with HA standby
Nexus1KV-Switch(standby)# _
```

30. Within the VMM console, remove the ISO file from both virtual machines.
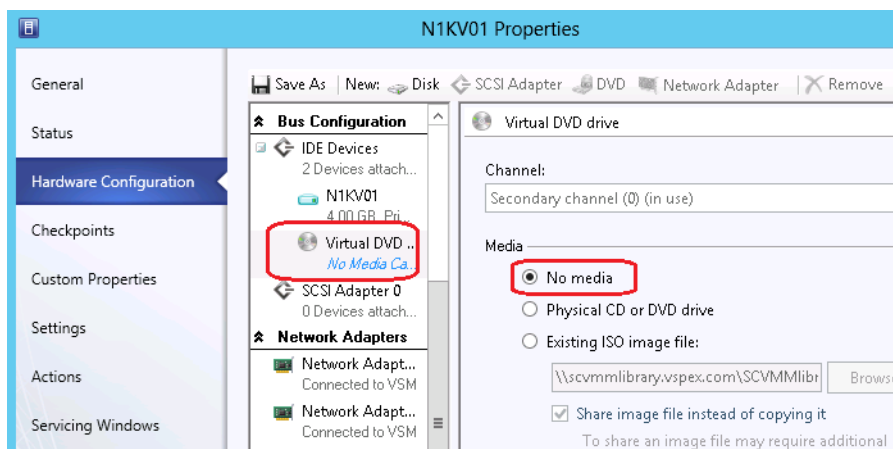
| Work from the primary VSM to continue with the configuration.<br><br>Enter the commands shown at right.<br><br>*&lt;FastTrack3&gt;* - user-defined name that will be used when defining a logical switch in VMM<br><br>*&lt;Fabric-Mgmt&gt;* - user-defined name of the management fabric. Member of just defined logical network.<br><br>*&lt;N1KV-pool-15&gt;* - user-defined name for a fabric management IP pool. Multiple pools can be created when managing multiple networks with N1KV.<br><br>*&lt;192.168.15.100 192.168.15.199&gt;* - pool of IP addresses to be managed<br><br>*&lt;192.168.15.0 255.255.255.0&gt;* - pool IP subnet and netmask<br><br>*&lt;192.168.15.1&gt;* - pool default gateway<br><br>*&lt;N1KV-MF-Public&gt;* - user-defined network segment name. Different network segments can be defined using different IP pools.<br><br>*&lt;15&gt;* - VLAN tag for management network<br><br>*&lt;AllAccess&gt;* - port profile created for later use in the definition of logical switch in VMM when configuring the virtual port.<br><br>*&lt;N1KV-MF-Uplink&gt;* - uplink port profile created for later use in the definition of the logical switch in VMM.<br><br>*&lt;N1KV_Uplink_Policy_FastTrack&gt;* uplink port profile for physical NIC. | ```
conf t

feature telnet

nsm logical network  <FastTrack3>
 exit

nsm network segment pool <Fabric-Mgmt>
 member-of logical network  <FastTrack3>
 exit

nsm ip pool template <N1KV-pool-15>
   ip address <192.168.15.100 192.168.15.199>
   network <192.168.15.0 255.255.255.0>
   default-router <192.168.15.1>
   exit

nsm network segment <N1KV-MF-Public>
   member-of network segment pool <Fabric-Mgmt>
   switchport access vlan <1>
   ip pool import template <N1KV-pool-15>
   publish network segment
   exit

port-profile type vethernet <AllAccess>
 no shutdown
 state enabled
 publish port-profile
 exit

port-profile type ethernet
<N1KV_Uplink_Policy_FastTrack>
   channel-group auto mode on mac-pinning
   no shutdown
   state enabled
   exit

nsm network uplink <N1KV-MF-Uplink>
   import port-profile
<N1KV_Uplink_Policy_FastTrack>
   allow network segment pool <Fabric-Mgmt>
   system network uplink
   publish network uplink
   exit

copy running-config startup-config
``` |

Configure Virtual Switch Extension Manager in VMM

1. On the Virtual Machine Manager virtual machine that is currently running the highly available Virtual Machine Manager service, install the Cisco Nexus 1000V switch extensions by running Nexus1000V-VSEMProvider-5.2.1.SM1.5.1.0.msi from an elevated PowerShell or command window.

2. Select the check box by the I accept the terms in the Licensing Agreement statement.

3. Click Install to continue.

4. A status screen will show the progress of the installation.

5. Click Finish to complete the installation.

> **Note** This may cause a loss of connection to the VMM console and you will have to reconnect.

6. In the VMM console, select Fabric. Under Networking, select Switch Extension Manager.

7. Click Add Resources and from the drop-down menu select Virtual Switch Extension Manager.

8. In the Enter connection setting for the extension manager to add dialog window, enter http://<Nexus1KV-Switch> for accessing the Nexus 1000V VSM you created earlier. This is the name you provided for the switch and for which you created the DNS entry, not the name of the virtual machine.

9. Click Browse to enter credentials for connecting to the VSM.

10. In the Select a Run As Account dialog window, click on the Create Run As Account.

11. In the Provide the details for this Run As account dialog window, enter a Name to be used for this account. Optionally, enter a Description.

12. In the User name box, enter the user ID for the administrative account (admin) created on the Nexus 1000V virtual machine.

13. Enter and confirm the Password for the administrator account on the Nexus 1000V virtual machine.

14. Make sure the check box by Validate domain credentials is cleared, as this account is not in AD.

15. Click OK to continue.

16. In the Select a Run As account dialog window, select the newly create VSM administrator account.

17. Click OK to continue.

18. Click Next when you return to the General screen to move to the Host Groups screen.

19. Select the box next to the All Hosts group.

20. Click Next to continue.

21. The next screen is a summary screen. Validate that entries were properly made. Click Finish when you have the correct values.

You will now see a listing for the Virtual Switch Extension Manager within VMM.

22. Using the Cluster Failover Manager on the VMM cluster, move the highly available Virtual Machine Manager instance to the second Virtual Machine Manager node.

23. Connect to the second Virtual Machine Manager node. Install the Cisco Nexus 1000V switch extensions by running Nexus1000V-VSEMProvider-5.2.1.SM1.5.1.0.msi.

24. Select the checkbox by I accept the terms in the License Agreement and click Install. Click Finish when the installation completes.

Copy Virtual Ethernet Module Installation Packager to the VMM Virtual Machines

Perform the following procedure on each Virtual Machine Manager node.

1. Copy Nexus1000V-VEM-5.2.1.SM1.5.1.0.msi to the following directory on each Virtual Machine Manager server:

C:\ProgramData\Switch Extensions Drivers

Configure a Logical Switch in VMM

1. In the VMM console, select Fabric. Under Networking select Logical Switches.

2. Click on Create Logical Switch in the ribbon.

3. Click Next on the Getting Started page.

4. Enter a Name and optional description for the logical switch being created.

5. Click Next to continue.

6. On the Extensions dialog window, clear the check box by Microsoft Windows Filtering Platform.

7. Select the check box by <Nexus1KV-Switch>.

8. Click Next to continue.

9. In the Uplink dialog window, select Team from the Uplink mode drop-down menu.

10. Click Add... to select the uplink profile previously created.

11. Select the Port profile that was created earlier when configuring VSM.

12. Click OK to continue.

13. Click Next on the Uplink dialog window.

14. In the Virtual Port dialog window, click Add.

15. In the Configure the virtual port dialog window, click Browse.

16. In the Select a Port Profile Classification dialog window, select Host Management.

17. Click Create Port Classification.

18. Enter a Name and Description to be used for the port classification.

19. Click OK to continue.

20. Select the just created port profile classification, and click OK to continue.

21. Back in the Configure the virtual port dialog window, select the check box by your virtual switch, and select the port profile created earlier from the drop-down box.

22. Click OK to continue.

23. That takes you back to the Virtual Port dialog window. Click Next in that window to bring up a Summary window.

24. In the Summary window, review your inputs.

25. Click Finish to continue.

26. Back in the VMM console, you will see the newly created Logical Switch.

Create the Logical Switch on the Hyper-V Hosts

Perform the following on each clustered Hyper-V node used for Fabric Management.

1. On the physical host, type ncpa.cpl from a PowerShell or command window to launch Network Connections.

2. Find the network adapter that will be used for the new logical switch and note the interface number (#9 in the shown screen shot). This is quite likely to be different on each node.

3. From the VMM console, select Fabric. Expand Servers > All Hosts > <host-group> Select the host from the previous step.

4. Select Properties.

5. In the Properties window, select Virtual Switches from the left-hand column.

6. Click New Virtual Switch and select New Logical Switch from the drop-down menu.

7. Select the new logical switch in the center panel. Select the Cisco VIC Ethernet Interface with the number obtained in the first step of this process.

8. Click OK to continue.

9. Click OK in the warning message to continue.

10. Repeat this process for each host in the cluster before proceeding to the next step.

11. Click Jobs to monitor the jobs progress. When completed it will show a status of Completed w/ Info until the logical switch is installed on all hosts in the cluster.

12. Open the Properties of the cluster and select Virtual Switches to see that the newly created logical switch is available to the cluster.

13. Click Cancel to exit the cluster properties window.

Create a Virtual Machine Network

1. In the Virtual Machine Manager console, select VMs and Services. Right-click VM Networks and select Create VM Network.

2. Enter a Name for the network. Ensure that the Logical network you are deploying is selected.
3. Click Next to continue.

4. In the Isolation dialog window, click the radio button by Specify an externally supplied VM Network. From the drop-down list for External VM network, select the network segment defined when configuring the VSM.
5. Click Next to continue.
6. On the Summary window, click Finish.

You can see the definition of the VM network in the VMM console.

Configure the Virtual Machine Manager Virtual Machine Properties

This example shows how to add the network managed by the Nexus 1000V to the VMM virtual machines. The same procedure would be used to add network adapters on this managed network to other virtual machines.

1. Login to the first Virtual Machine Manager virtual machine. Using Failover Cluster Manager console, identify the owner of the highly available Virtual Machine Manager instance. Move the Virtual Machine Manager instance to the second node, if it is owned by the first node, by right-clicking on the role, selecting Move > Best Possible Node.

2. When you see the role has successfully moved to the other node, shutdown the first VMM virtual machine by running following PowerShell command:

Stop-Computer

3. Log into the second VMM virtual machine and start the Virtual Machine Manager console. Select VMs and Services. Click All hosts.

4. Click the first Virtual Machine Manager virtual machine that is in a stopped state and select Properties.

5. Add a net network adapter to the VMM virtual machine by selecting Hardware Configuration from the left column. Click on Network Adapter and select Network Adapter from the drop-down list.

6. A new adapter will be created and added to the end of the list of existing adapters in the center pane. Select the newly created adapter.

7. Click the radio button by Connected to a VM Network.

8. Click Browse.

9. In the Select a VM Network dialog window, select the VM network created in the previous steps.

10. Click OK to continue.

11. Select Management Fabric from the Classification drop-down list under Virtual Switch.

12. Click OK to continue.

13. Select Jobs and monitor the job completion progress.

14. Start the VMM virtual machine by right-clicking on the VM and selecting Power On.

15. Log into the first VMM virtual machine. Using the Failover Cluster Manager console, move the highly available VMM role to the first VMM virtual machine.

16. Stop the second VMM virtual machine by issuing the following PowerShell command:

Stop-Computer -ComputerName <VMM02> -Force

```
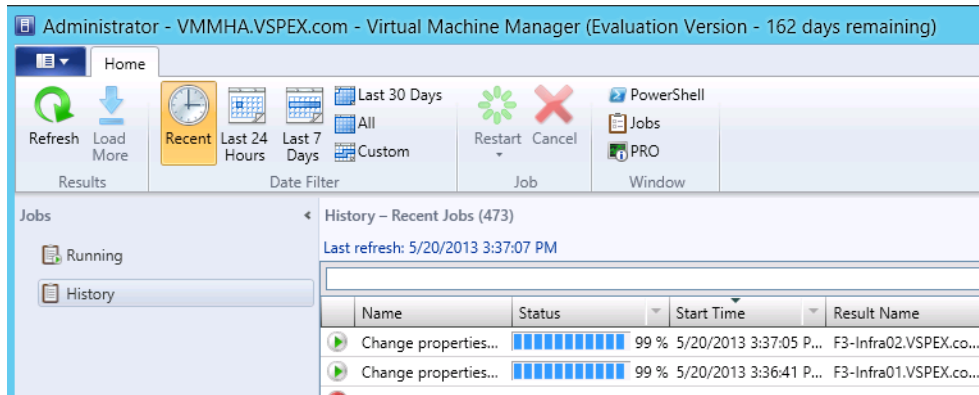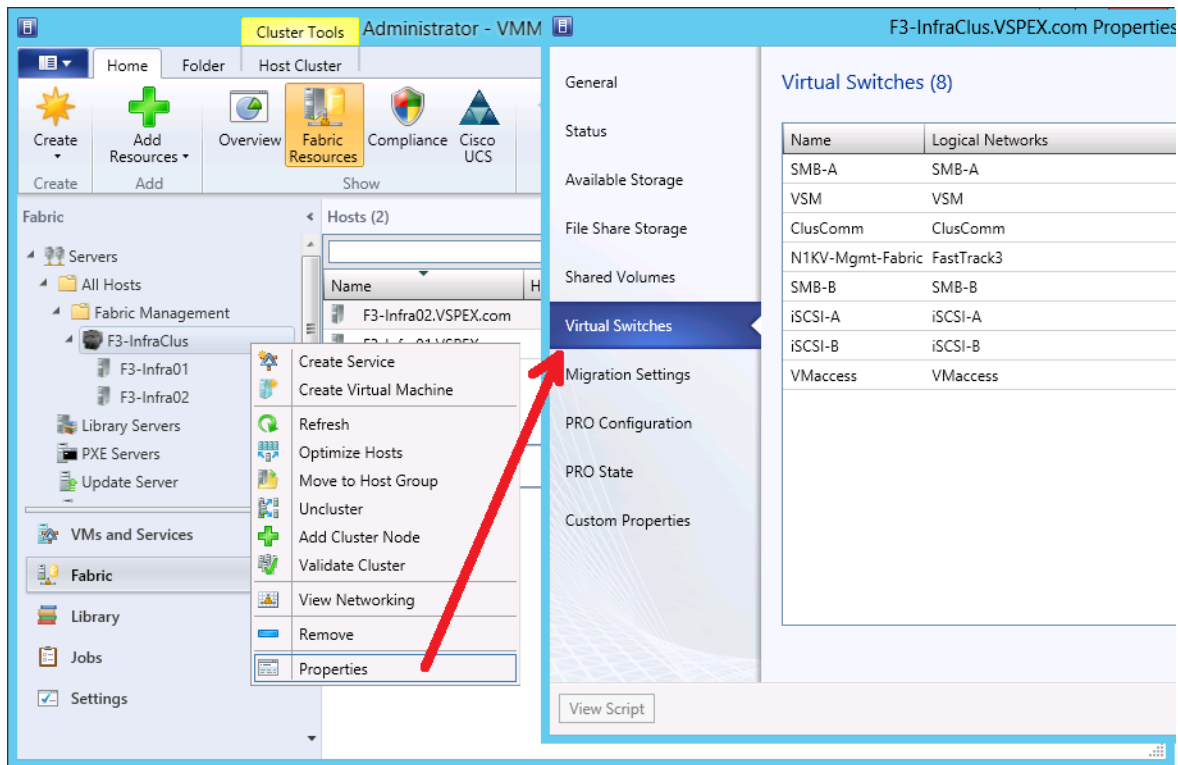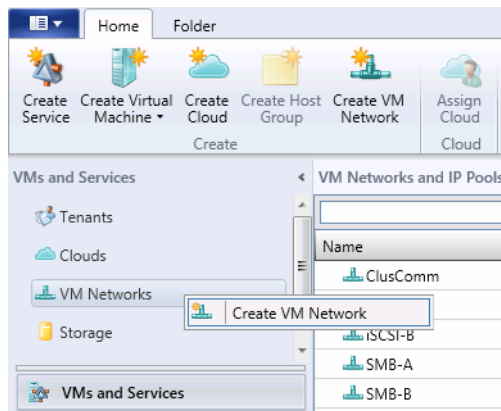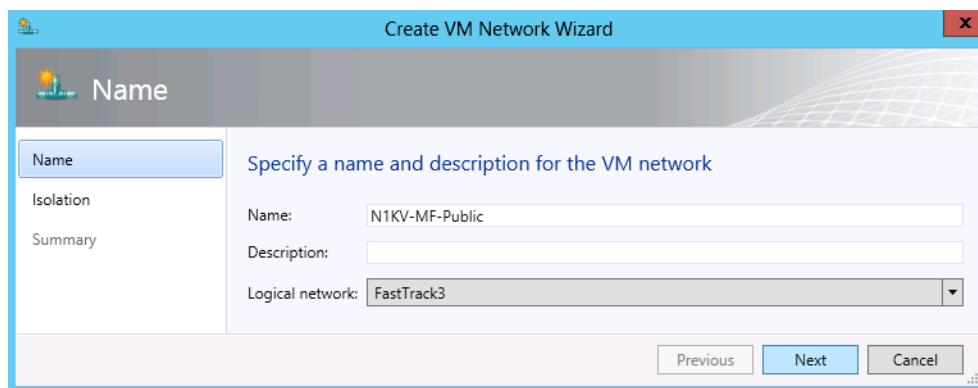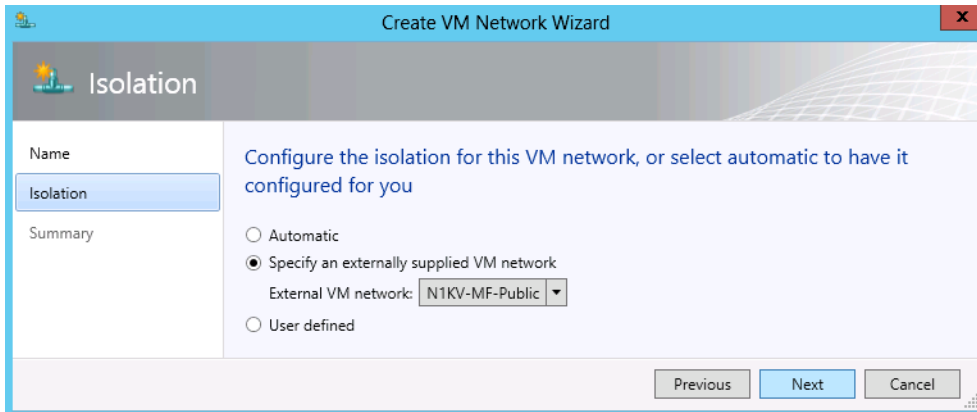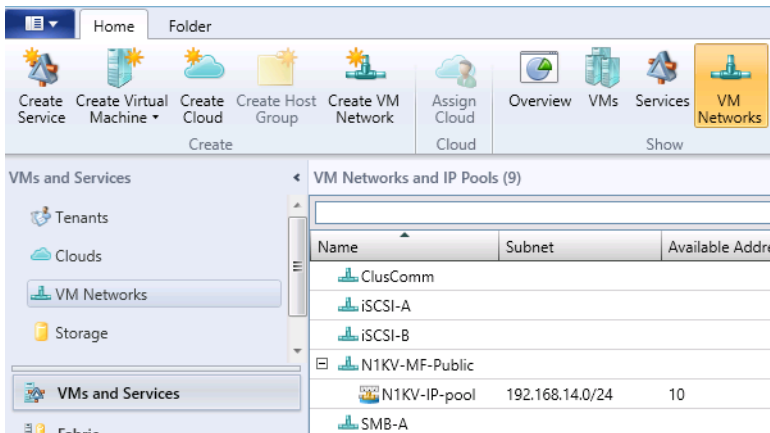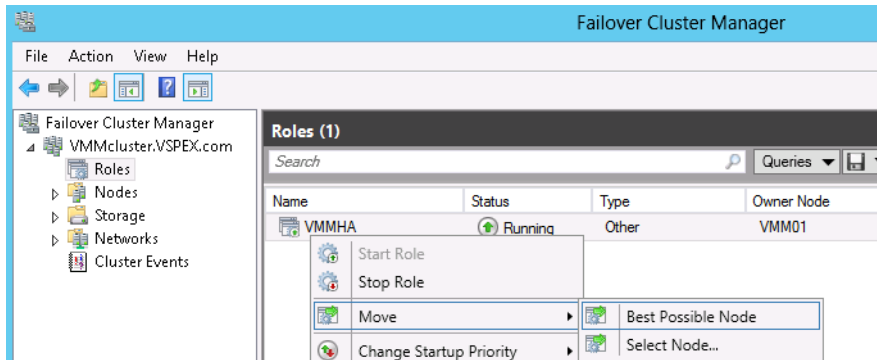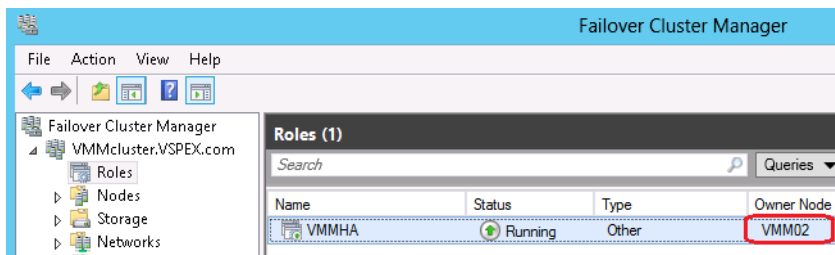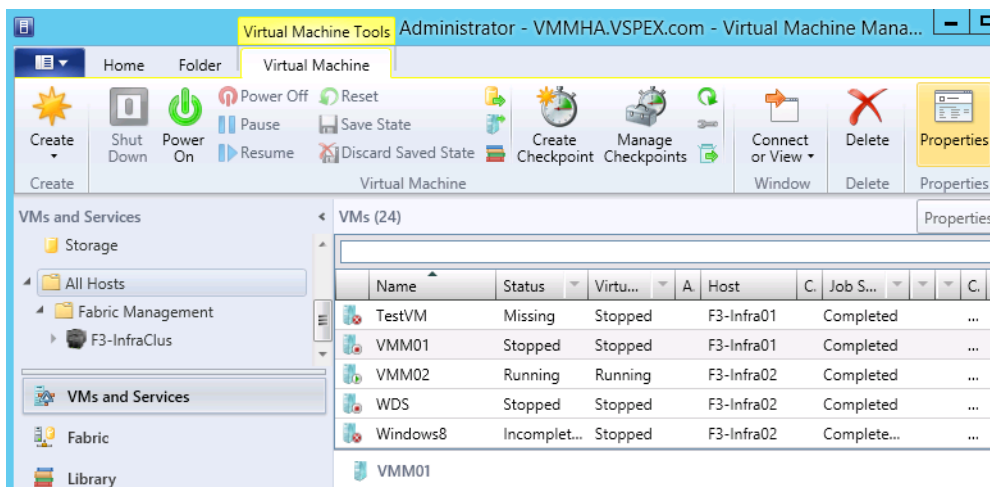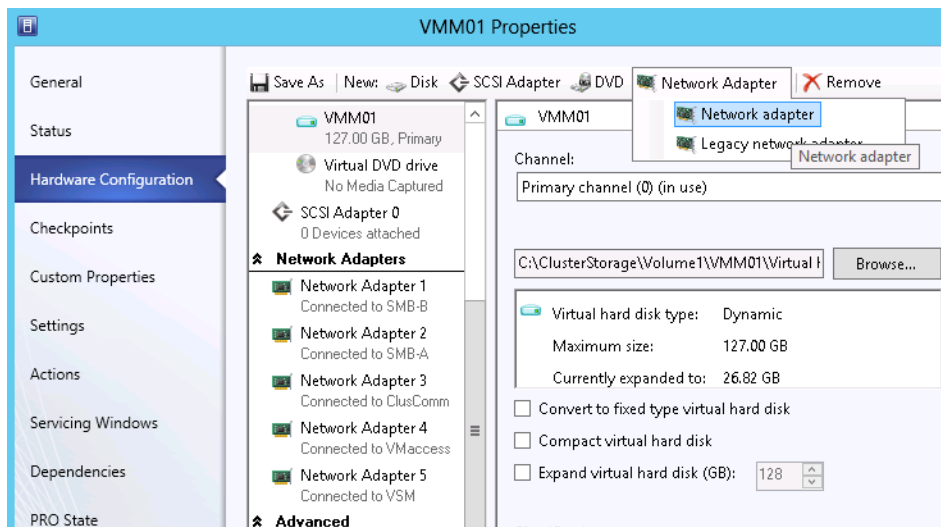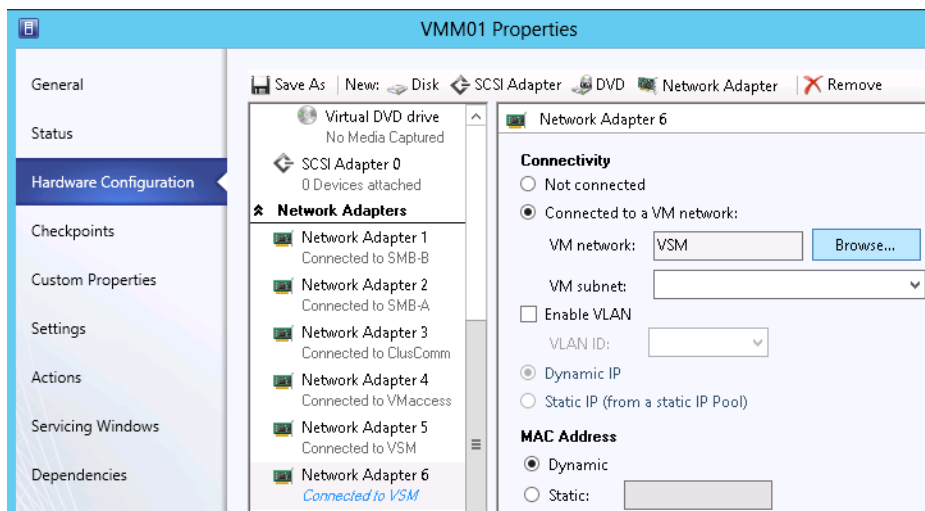Stop-Computer -ComputerName VMM02 -Force
```

17. Start the VMM console and select VMs and Services. Expand All Hosts. Right-click the stopped VMM virtual machine and select Properties.

18. Select Hardware Configuration. Scroll to Network Adapters in the center pane. Click Network Adapter and select Network adapter from the drop-down list to add a new network adapter to the virtual machine.

19. A new adapter will be created and added to the end of the list of existing adapters in the center pane. Select the newly created adapter.

20. Click the radio button by Connected to a VM Network.

21. Click Browse.

22. In the Select a VM Network dialog window, select the VM network created in the previous steps.

23. Click OK to continue.

24. Select Management Fabric from the drop-down list under Virtual Switch.

25. Click OK to continue.

26. Select Jobs and monitor the job completion progress.

27. When the job completes, start the VMM virtual machine.

Customer Configuration Worksheets

These worksheets provide a way to capture all the information needed before installation is started. It can also be left with the customer for future reference.

Common Server

Server Name	Purpose	Primary IP address
	Active Directory Domain Controller	
	DNS Primary	
	DNS Secondary	
	DHCP	
	NTP	
	SMTP	
	SNMP	
	System Center Virtual Machine Manager	

Hyper-V Servers

Server Name	Purpose	Mgmt IP	LiveMigration IP	CSV IP	
	Hyper-V				
	Hyper-V				
	Hyper-V				
	Hyper-V				
	Hyper-V				
	Hyper-V				

WWNN/WWPN

Device	Port	WWPN	WWNN
VSPEX-01	Fabric A		
	Fabric B		
VSPEX-02	Fabric A		
	Fabric B		
VSPEX-03	Fabric A		
	Fabric B		
VSPEX-04	Fabric A		
	Fabric B		

VSPEX-05	Fabric A		
	Fabric B		
VSPEX-06	Fabric A		
	Fabric B		

EMC VNX5400 Array

Array name	
Management IP	
Administrator account	

WWNN/WWPN

Port	**WWNN**	**WWPN**
SPA-A2		
SPA-A3		
SPB-B2		
SPB-B3		

Cisco UCS Network Infrastructure

Name	Purpose	IP	Subnet Mask	Default Gateway
	Cisco UCSM			
	Cisco UCS FI B			
	Cisco UCS FI B			
	Cisco Nexus 5548 A			
	Cisco Nexus 5548 B			
	vPC domain ID	(value)	N/A	N/A

VLAN Information

Name	Purpose	VLAN ID	Allowed Subnets
Mgmt	Infrastructure management		
LiveMigration	Live Migration		
CSV	Cluster Shared Volume		

VMaccess	Virtual Machine access		
VEM (optional)	Nexus 1000V		

Service Information

Account Name	Purpose	Password (optional, secure appropriately)
	Windows Server administrator	
admin	UCSM administrator	
admin	Nexus 5548 administrator	
nasadmin	EMC VNX array administrator	

Cabling

Cisco Nexus 5548 A

Local Port	Connection	Remote Device	Remote Port
Eth 1/1	10 GE	Cisco Nexus 5548 B	Eth 1/1
Eth 1/2	10 GE	Cisco Nexus 5548 B	Eth 1/2
Eth 1/17	10 GE	Cisco 6248 A	Eth 1/17
Eth 1/18	10 GE	Cisco 6248 B	Eth 1/17
Eth 1/29	10 GE	EMC SPA	A2
Eth 1/30	10 GE	EMC SPB	B2
FC 1/31	FC	Cisco 6248 A	FC 1/31
FC 1/32	FC	Cisco 6248 A	FC 1/32

Cisco Nexus 5548 B

Local Port	Connection	Remote Device	Remote Port
Eth 1/1	10 GE	Cisco Nexus 5548 A	Eth 1/1
Eth 1/2	10 GE	Cisco Nexus 5548 A	Eth 1/2
Eth 1/17	10 GE	Cisco 6248 B	Eth 1/18
Eth 1/18	10 GE	Cisco 6248 A	Eth 1/18
Eth 1/29	10 GE	EMC SPA	A3
Eth 1/30	10 GE	EMC SPB	B3
FC 1/31	FC	Cisco 6248 B	FC 1/31
FC 1/32	FC	Cisco 6248 B	FC 1/32

Cisco 6248 Fabric Interconnect A

Local Port	Connection	Remote Device	Remote Port
Eth 1/1	10 GE	Chassis 1 FEX A	Port 1
Eth 1/2	10 GE	Chassis 1 FEX B	Port 1
Eth 1/17	10 GE	Cisco 5548 A	Eth 1/17
Eth 1/18	10 GE	Cisco 5548 B	Eth 1/17
FC 1/31	FC	Cisco 5548 A	FC 1/31
FC 1/32	FC	Cisco 5548 A	FC 1/32

Cisco 6248 Fabric Interconnect B

Local Port	Connection	Remote Device	Remote Port
Eth 1/1	10 GE	Chassis 1 FEX A	Port 2
Eth 1/2	10 GE	Chassis 1 FEX B	Port 2
Eth 1/17	10 GE	Cisco 5548 B	Eth 1/18
Eth 1/18	10 GE	Cisco 5548 A	Eth 1/18
FC 1/31	FC	Cisco 5548 B	FC 1/31
FC 1/32	FC	Cisco 5548 B	FC 1/32

Appendix

FCoE Configuration

The following instructions assume a working environment that consists of adding FCoE direct capabilities.

VLANs

VLAN Name	VLAN Purpose	ID used in this document
FCoE-A	Fabric A traffic	101
FCoE-B	Fabric B traffic	102

Cabling

Local Device	Local Port	Connection	Remote Device	Remote Port
Nexus A	Eth 1/19	FCoE	Fab Interconnect A	Eth 1/19
	Eth 1/20	FCoE	Fab Interconnect A	Eth 1/20
	Eth 1/27	FCoE	EMC SPA A3	1 (A-13)
	Eth 1/28	FCoE	EMC SPB B3	1 (B-13)
Nexus B	Eth 1/19	FCoE	Fab Interconnect B	Eth 1/19
	Eth 1/20	FCoE	Fab Interconnect B	Eth 1/20
	Eth 1/27	FCoE	EMC SPA A3	0 (A-12)
	Eth 1/28	FCoE	EMC SPB B3	0 (B-12)
Fab Interconnect A	Eth 1/19	FCoE	Nexus A	Eth 1/19
	Eth 1/20	FCoE	Nexus A	Eth 1/20
Fab Interconnect B	Eth 1/19	FCoE	Nexus B	Eth 1/19
	Eth 1/20	FCoE	Nexus B	Eth 1/20
EMC SPA A3	0 (A-12)	FCoE	Nexus B	Eth 1/27
	1 (A-13)	FCoE	Nexus A	Eth 1/27
EMC SPB B3	0 (B-12)	FCoE	Nexus B	Eth 1/28
	1 (B-13)	FCoE	Nexus A	Eth 1/28

Nexus Configuration

The following configuration steps assume that the settings are being added into a running configuration.

Configure QoS Policy

Nexus A and Nexus B

```
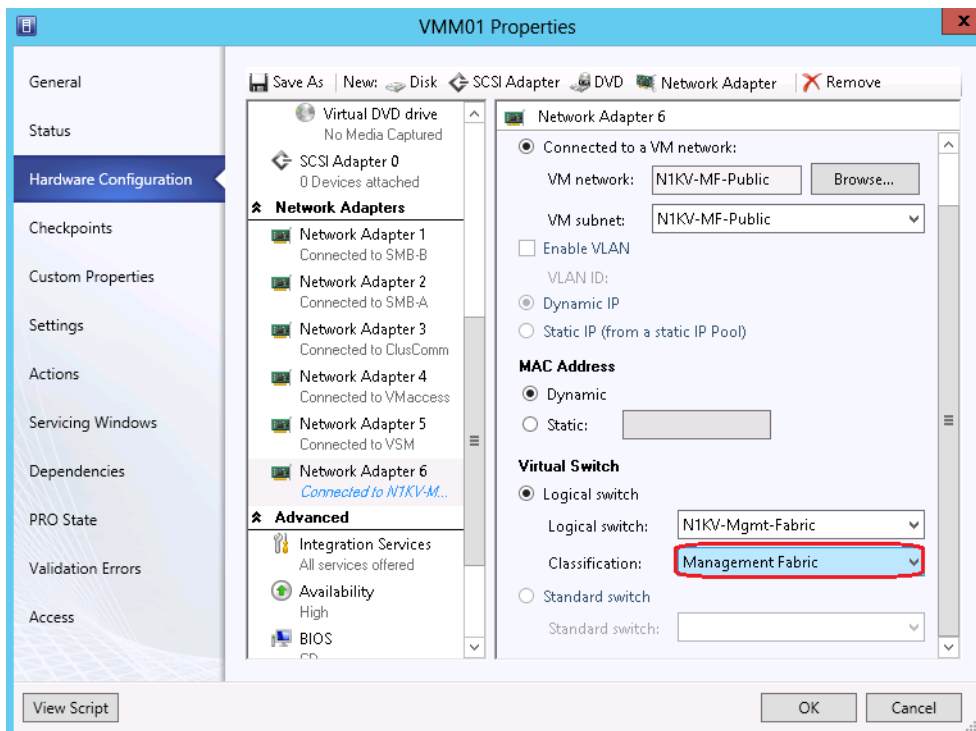policy-map type qos system-qos-policy
  class class-fcoe
    set qos-group 1
    exit
  exit
policy-map type queuing system-queue-in-policy
  class type queuing class-fcoe
    bandwidth percent 75
    class type queuing class-default
    bandwidth percent 25
    exit
  exit
policy-map type queuing system-queue-out-policy
  class type queuing class-fcoe
    bandwidth percent 75
    class type queuing class-default
    bandwidth percent 25
    exit
  exit
policy-map type network-qos system-nwqos-policy
```

```
      class type network-qos class-fcoe
        pause no-drop
        mtu 2158
      class type network-qos class-default
        mtu 9000
        exit
      exit
    system qos
      service-policy type qos input system-qos-policy
      service-policy type queuing input system-queue-in-policy
      service-policy type queuing output system-queue-out-policy
      service-policy type network-qos system-nwqos-policy
      exit
    copy running-config start-config
    show running-config ipqos
    class-map type qos class-fcoe
    class-map type queuing class-fcoe
      match qos-group 1
    class-map type queuing class-all-flood
      match qos-group 2
    class-map type queuing class-ip-multicast
      match qos-group 2
    policy-map type qos system-qos-policy
      class class-fcoe
        set qos-group 1
      class class-default
    policy-map type queuing system-queue-in-policy
      class type queuing class-fcoe
        bandwidth percent 75
      class type queuing class-default
        bandwidth percent 25
    policy-map type queuing system-queue-out-policy
      class type queuing class-fcoe
        bandwidth percent 75
      class type queuing class-default
        bandwidth percent 25
    class-map type network-qos class-fcoe
      match qos-group 1
    class-map type network-qos class-all-flood
      match qos-group 2
    class-map type network-qos class-ip-multicast
      match qos-group 2
    policy-map type network-qos jumbo
      class type network-qos class-default
        mtu 9000
        multicast-optimize
    policy-map type network-qos system-nwqos-policy
      class type network-qos class-fcoe
        pause no-drop
        mtu 2158
      class type network-qos class-default
        mtu 9000
        multicast-optimize
    system qos
      service-policy type qos input system-qos-policy
      service-policy type queuing input system-queue-in-policy
      service-policy type queuing output system-queue-out-policy
      service-policy type network-qos system-nwqos-policy
```

VLANs

Nexus A

```
vlan 101
  name FCoE-A
  exit
show vlan
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Po10, Po13, Po14,
                                                 Eth1/1, Eth1/2, Eth1/3, Eth1/4
                                                 Eth1/5, Eth1/6, Eth1/7, Eth1/8
                                                 Eth1/9, Eth1/10, Eth1/11
                                                 Eth1/12, Eth1/13, Eth1/14
                                                 Eth1/15, Eth1/16, Eth1/17
                                                 Eth1/18, Eth1/21, Eth1/22
                                                 Eth1/23, Eth1/24, Eth1/25
                                                 Eth1/26, Eth1/27, Eth1/28
10   VMaccess                         active    Po10, Po13, Po14, Eth1/1
                                                 Eth1/2, Eth1/17, Eth1/18
11   LiveMigration                    active    Po10, Po13, Po14, Eth1/1
                                                 Eth1/2, Eth1/17, Eth1/18
12   CSV                              active    Po10, Po13, Po14, Eth1/1
                                                 Eth1/2, Eth1/17, Eth1/18
13   ClusComm                         active    Po10, Po13, Po14, Eth1/1
                                                 Eth1/2, Eth1/17, Eth1/18
16   SMB-A                            active    Po10, Po13, Po14, Eth1/1
                                                 Eth1/2, Eth1/17, Eth1/18
17   SMB-B                            active    Po10, Po13, Po14, Eth1/1
                                                 Eth1/2, Eth1/17, Eth1/18
18   iSCSI-A                          active    Po10, Po13, Po14, Eth1/1
                                                 Eth1/2, Eth1/17, Eth1/18
19   iSCSI-B                          active    Po10, Po13, Po14, Eth1/1
                                                 Eth1/2, Eth1/17, Eth1/18
101  FCoE-A                           active
200  VEM                              active    Po10, Po13, Po14, Eth1/1
                                                 Eth1/2, Eth1/17, Eth1/18

VLAN Type  Vlan-mode
---- ----- ----------
1    enet  CE
10   enet  CE
11   enet  CE
12   enet  CE
13   enet  CE
16   enet  CE
17   enet  CE
18   enet  CE
19   enet  CE
101  enet  CE
200  enet  CE
Primary  Secondary  Type            Ports
-------  ---------  --------------  -------------------------------------------
```

Nexus B

```
vlan 102
  name FCoE-B
```

```
   exit
```

Port Descriptions

Nexus A

```
interface Ethernet1/19
  description UCSM-A-Eth1/19-FCoE
  exit
interface Ethernet1/20
  description UCSM-A-Eth1/20-FCoE
  exit
interface Ethernet1/27
  description SPA-A3:1-FCoE
  switchport mode trunk
  switchport trunk allowed vlan 101
  spanning-tree port type edge trunk
  exit
interface Ethernet1/28
  description SPB-B3:1-FCoE
  switchport mode trunk
  switchport trunk allowed vlan 101
  spanning-tree port type edge trunk
  exit
show interface description
--------------------------------------------------------------------------------
Interface              Description
--------------------------------------------------------------------------------
fc1/29              --
fc1/30              --
fc1/31              --
fc1/32              --


--------------------------------------------------------------------------------
Port          Type   Speed   Description
--------------------------------------------------------------------------------
Eth1/1        eth    10G     Nexus-B:Eth1/1-Peerlink
Eth1/2        eth    10G     Nexus-B:Eth1/2-Peerlink
Eth1/3        eth    10G     --
Eth1/4        eth    10G     --
Eth1/5        eth    10G     --
Eth1/6        eth    10G     --
Eth1/7        eth    10G     --
Eth1/8        eth    10G     --
Eth1/9        eth    10G     --
Eth1/10       eth    10G     --
Eth1/11       eth    10G     --
Eth1/12       eth    10G     --
Eth1/13       eth    10G     --
Eth1/14       eth    10G     --
Eth1/15       eth    10G     --
Eth1/16       eth    10G     --
Eth1/17       eth    10G     UCSM-A eth1/17
Eth1/18       eth    10G     UCSM-B eth1/17
Eth1/19       eth    10G     UCSM-A-Eth1/19-FCoE
Eth1/20       eth    10G     UCSM-A-Eth1/20-FCoE
Eth1/21       eth    10G     --
Eth1/22       eth    10G     --
Eth1/23       eth    10G     --
Eth1/24       eth    10G     --
Eth1/25       eth    10G     --
Eth1/26       eth    10G     --
Eth1/27       eth    10G     SPA-A3:1-FCoE
```

```
Eth1/28        eth    10G     SPB-B3:1-FCoE


--------------------------------------------------------------------------------
Interface               Description
--------------------------------------------------------------------------------
Po10                    vPC Peer-link
Po13                    UCS-6248-A
Po14                    UCS-6248-B
```

Nexus B

```
interface Ethernet1/19
  description UCSM-B-Ethernet1/19-FCoE
  exit
interface Ethernet1/20
  description UCSM-B-Ethernet1/20-FCoE
  exit
interface Ethernet1/27
  description SPA-A3:0-FCoE
  switchport mode trunk
  switchport trunk allowed vlan 102
  spanning-tree port type edge trunk
  exit
interface Ethernet1/28
  description SPB-B3:0-FCoE
  switchport mode trunk
  switchport trunk allowed vlan 102
  spanning-tree port type edge trunk
  exit
```

Create Port Channel

Nexus A

```
interface Ethernet1/19
  switchport description UCSM-A-Eth1/19-FCoE
  exit
interface Ethernet1/20
  switchport description UCSM-A-Eth1/20-FCoE
  exit
interface Po105
  switchport description UCSM-A-FCoE
  exit
interface Ethernet1/19-20
  channel-group 105 mode active
  no shutdown
  exit
copy running-config start-config
show port-channel summary
Flags:  D - Down       P - Up in port-channel (members)
        I - Individual  H - Hot-standby (LACP only)
        s - Suspended   r - Module-removed
        S - Switched    R - Routed
        U - Up (port-channel)
        M - Not in use. Min-links not met
--------------------------------------------------------------------------------
Group Port-      Type    Protocol  Member Ports
      Channel
--------------------------------------------------------------------------------
10    Po10(SU)   Eth     LACP      Eth1/1(P)    Eth1/2(P)
```

```
13     Po13(SU)    Eth      LACP       Eth1/17(P)
14     Po14(SU)    Eth      LACP       Eth1/18(P)
105    Po105(SU)   Eth      LACP       Eth1/19(P)   Eth1/20(P)
```

Nexus B

```
interface Ethernet1/19
  switchport description UCSM-B-Eth1/19-FCoE
  exit
interface Ethernet1/20
  switchport description UCSM-B-Eth1/20-FCoE
  exit
interface Po106
  switchport description UCSM-B-FCoE
  interface Ethernet1/19-20
  channel-group 106 mode active
  no shutdown
  exit
copy running-config start-config
```

Add Port Channel Configuration

Nexus A

```
interface Po105
  switchport mode trunk
  switchport trunk allowed vlan 101
  no shutdown
  exit
copy running-config start-config
show running-config interface port-channel 105
interface port-channel105
  description UCSM-A-FCoE
  switchport mode trunk
  switchport trunk allowed vlan 101
```

Nexus B

```
interface Po106
  switchport mode trunk
  switchport trunk allowed vlan 102
  no shutdown
  exit
copy running-config start-config
```

Configure FCoE Fabric

Nexus A

```
interface vfc101
  bind interface ethernet1/27
  switchport trunk allowed vsan 101
  no shutdown
  exit
interface vfc102
  bind interface ethernet1/28
  switchport trunk allowed vsan 101
  no shutdown
  exit
```

```
vsan database
  vsan 101
  vsan 101 name Fabric_A
  vsan 101 interface vfc101
  vsan 101 interface vfc102
  exit
vlan 101
  fcoe vsan 101
  exit
copy running-config startup-config
show vsan 101
vsan 101 information
        name:Fabric_A  state:active
        interoperability mode:default
        loadbalancing:src-id/dst-id/oxid
        operational state:up
show vsan 101 membership
vsan 101 interfaces:
    vfc101            vfc102
```

Nexus B

```
interface vfc101
  bind interface ethernet1/27
  switchport trunk allowed vsan 102
  no shutdown
  exit
interface vfc102
  bind interface ethernet1/28
  switchport trunk allowed vsan 102
  no shutdown
  exit
vsan database
  vsan 102
  vsan 102 name Fabric_B
  vsan 102 interface vfc101
  vsan 102 interface vfc102
  exit
vlan 102
  fcoe vsan 102
  exit
copy running-config startup-config
```

Cisco Unified Computing System Configuration

Create FCoE Uplink Ports

On the **Equipment** tab, select **Fabric Interconnects** > **Fabric Interconnect A** > **Fixed Module** > **Ethernet Ports**. Select ports 19 and 20. Right-click and select **Configure as FCoE Uplink Port**. Click **Yes** on the confirmation window. Repeat on fabric B.	

Create VSANs

Select the **SAN** tab at the top left of the window. Expand the SAN Cloud tree. Right-click **VSANs** and select **Create VSAN**. Enter *<VSAN_A>* as the VSAN name for fabric A. Keep the **Disabled** option selected for the Default Zoning Select the **Fabric A** radio button. Enter the **VSAN ID** for fabric A. Enter the **FCoE VLAN ID** for fabric A. Click **OK** and then **OK** to create the VSAN.	
Right-click **VSANs**. Select **Create VSAN**. Enter *<VSAN_B>* as the VSAN name for fabric B. Keep the **Disabled** option selected for the Default Zoning Select the **Fabric B** radio button. Enter the **VSAN ID** for fabric B. Enter the **FCoE VLAN ID** for fabric B. Click **OK** and then **OK** to create the VSAN.	

Create a Fibre Channel Policy

Select to the **SAN** tab at the top of the left window. Go to **SAN > Policies > root**.

Right-click **Fibre Channel Adapter Policies** and click **Create Fibre Channel Adapter Policy**.

Use **Windows-EMC** as the name of the Fibre Channel Adapter Policy.

The default values are appropriate for most configurable items. Expand the Options dropdown. Set the **Link Down Timeout (MS)** option to **5000**.

Click **OK** to complete creating the FC adapter policy.

Click **OK**.

Create VHBA Templates

Select the **SAN** tab on the left of the window. Go to **Policies > root**.

Right-click **vHBA Templates**. Select **Create vNIC Template**.

Enter <*FCoE-A*> as the vHBA template name.

Select the **Fabric A** radio button.

Under **Select VSAN**, select **VSAN_A**. Under **WWN Pool**, select the previously created WWN pool.

Click **OK** to complete creating the vHBA template.

Right-click **vHBA Templates**. Select **Create vNIC Template**. Enter <***FCoE-B***> as the vHBA template name. Select the **Fabric B** radio button. Under **Select VSAN**, select **VSAN-B**. Under **WWN Pool**, select the previously created WWN pool. Click **OK** to complete creating the vHBA template.	

Determine VNX FCoE Ports and WWNs

In order to create boot policies, you need to determine the WWNs assigned to the FCoE ports.

In Unisphere, navigate to **System** > **Hardware** > **Storage Hardware**. Select the appropriate I/O modules, right-click and select **Properties**. Validate that you have the proper I/O module. Click **OK** to close the Properties window.	

Expand the slot to see the individual ports on that I/O module. Right-click the first port and select **Properties**. Note the relative port name (not the actual port on the I/O module) in the windows title bar for later use. Select the **Virtual N_Port** and click on **Properties**. Record the WWNN (left half) and WWPN (right half) displayed in the **Port Identification**. Click **Cancel** and **Cancel** to close the windows. Repeat for the other port on this I/O module and for the ports on the other SP.	

The following table lists the values used in this example.

SP	Port	Relative Port Name	WWNN	WWPN
A	0	A-12	50:06:01:60:88:60:06:A1	50:06:01:64:08:64:06:A1
	1	A-13		50:06:01:65:08:64:06:A1
B	0	B-12		50:06:01:6C:08:64:06:A1
	1	B-13		50:06:01:6D:08:64:06:A1 |