vmbooks

A Practical Guide to Business Continuity & Disaster Recovery

with VMware Infrastructure

Featuring Hardware & Software Solutions from:

AMD
Cisco
Dell
Emulex
Intel
NetApp
Sun Microsystems

A Practical Guide to Business Continuity & Disaster Recovery with VMware Infrastructure 3

Revision: 20080912
Item: VMB-BCDR-ENG-Q308-001

VMbook Feedback - VMware welcomes your suggestions for improving our VMbooks.
If you have comments, send your feedback to: vmbookfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, California 94304
www.vmware.com

vmbooks

vmware®

contents

vmbooks

About this VMware VMbook

This VMware® VMbook focuses on business continuity and disaster recovery (BCDR) and is intended to guide the reader through the step-by-step process of setting up a multisite virtual datacenter with BCDR services for designated virtual machines at time of test or during an actual event that necessitated the declaration of a disaster, resulting in the activation of services in a designated BCDR site.

Furthermore, this VMbook demonstrates how the VMware Infrastructure virtualization platform is a true enabler when it comes to architecting and implementing a multisite virtual datacenter to support BCDR services at time of test or disaster.

Intended Audience

This VMbook is targeted at IT professionals who are part of the virtualization team responsible for architecting, implementing and supporting VMware Infrastructure, and who want to leverage their virtual infrastructure to support and enhance their BCDR services. A typical virtualization team will contain members with skills in the following disciplines:

- Networking
- Storage
- Server virtualization
- Operating system administration (Windows, UNIX and Linux)
- Security administration

This virtualization team will also be called upon to work closely with business continuity program (BCP) team members whose responsibility is to work closely with business owners to determine the criticality of the business applications and their respective service level agreements (SLAs) as they relate to recovery point objectives (RPOs) and recovery time objectives (RTOs). The BCP team will also determine how those business applications map to business users who use the business applications services during their daily operations. The list of business application services then gets mapped to both physical and virtual systems, along with their appropriate dependencies. This list of systems

forms the basis of the BCDR plan that will be implemented in part by the virtualization team, as well as other IT teams that are responsible for the non-virtualized business applications services.

It is worth noting that this VMbook is also intended for those members of the BCP team who in addition to having a business background also have a background in information technology; they can leverage this VMbook as a reference when working with the members of the information technology team who are responsible for the deployment of the multisite virtual datacenters to support application services during a disaster event or during a scheduled BCDR test.

The members of the virtualization team play an important role as they are responsible for providing a reliable, scalable and secure virtual infrastructure to support the virtualized business applications services at time of disaster or during a scheduled BCDR test.

The success of any BCDR strategy is ultimately driven by the collaborative efforts of the business owners who interface with the BCP team who in turn interface with the information technology team who provide the infrastructure and means to facilitate the failover of the business application services at time of disaster or scheduled BCDR test.

Document Structure and Organization

This BCDR VMbook is divided into four sections as follows:

- **Part 1: Introduction and Planning.** This section introduces key concepts and outlines the planning process for virtualization-based BCDR.
- **Part 2: Design and Implementation.** This section provides guidance around the design and implementation of a virtualization-based BCDR solution.
- **Part 3: BCDR Operations.** This section outlines the steps involved in scheduled and unscheduled failover, failback and other key BCDR operations.
- **Part 4: Infrastructure Component Details.** This section provides detail about the specific hardware and software used to build out the BCDR solution described in this VMbook. The content of this section will vary from book to book as VMware develops BCDR solutions with various technology partners.

About the Authors

This VMbook was compiled by a team of VMware Certified Professionals with in-depth experience in enterprise information technology. The team was based in United States and in the United Kingdom. The VMware Infrastructure BCDR solution detailed in this book was setup in the VMware UK Office Datacenter, located in Frimley.

David Burgess is a senior technologist for VMware with 20 years of experience varying from UNIX kernel and compiler development, product marketing and pre-sales roles. David currently works in the UK with VMware customers in the financial services sector.

Prior to VMware, David worked for HP, Novadigm, Volantis, IBM and Sequent.

Lee Dilworth joined VMware in October 2005, working as a senior consultant in the VMware Professional Services organization. Since July 2007, Lee has taken on the challenge of the new specialist systems engineer role for platform and architecture, covering Northern Europe. In his current role, Lee's main responsibility is working with the Northern European systems engineers sharing his extensive VMware implementation experience in the form of in-depth architecture and platform workshops, presentations, proof-of-concept demonstrations, trade shows and executive briefings. Alongside Lee's day-to-day role, he is also responsible in Northern Europe for the BCDR pre-sales technical function.

Prior to joining VMware, Lee was a senior consultant for Siebel Systems, where he worked on Siebel implementations for their UNIX customer base. Prior to Siebel, Lee worked for four years as an AIX / DB2 specialist for IBM UK. During this time, Lee also co-authored an IBM Redbook on DB2 Performance Tuning.

Luke Reed is a server and desktop virtualization specialist systems engineer at NetApp, where he assists customers across the UK in designing and architecting storage solutions for VMware Infrastructure deployments.

Luke has more than eight years experience in the IT industry in a variety of technical, consulting and pre-sales roles.

Mornay Van Der Walt has more than 15 years experience in enterprise information technology, joining VMware as a senior enterprise and technical marketing solutions architect. Mornay is currently focusing on projects that leverage VMware Infrastructure as an enabler for business continuity and disaster recovery service solutions.

Prior to VMware, Mornay was a vice president and system architect at a financial services firm in New York City, where he was responsible for architecting and the management of the firm's core infrastructure services, including the implementation of VMware Infrastructure in a multisite environment to support both production and BCDR services. Mornay played an active role in the firm's BCDR program and served in the role of project manger for several major IT projects.

Prior to immigrating to the US in 1998 from South Africa, Mornay completed his studies in Electrical Engineering and spent five years working in the manufacturing and financial services industries.

Acknowledgements

This VMbook is the result of a collaborative effort that included many other members of the VMware team. Their contributions throughout the project ensured the ultimate success of this project:

- Harvey Alcabes, Sr. Product Marketing Manager, USA
- Marc Benatar, Systems Engineer, UK
- Steve Chambers, Solutions Architect, UK
- Chris Dye, Inside Systems Engineer, UK
- Andrea Eubanks, Sr. Director, Enterprise and Technical Marketing, USA
- Warren Olivier, Partner Field Systems Engineer, UK
- Henry Robinson, Director, Product Management, USA
- Rod Stokes, Manager, Alliance System Engineers, UK
- Dale Swan, Systems Engineer, UK
- Richard Thomchick, Interactive Editor, USA
- Simon Townsend, Manager, Systems Engineering, UK

VMware Partner Participation

The success of this project was in large part also due to the VMware partners listed below. These organizations provided the various pieces of the infrastructure components as detailed in Part 4 of this VMbook and provided access to engineering resources when appropriate.

- AMD (www.amd.com)
- CISCO (www.cisco.com)
- Dell (www.dell.com)
- Emulex (www.emulex.com)
- Intel (www.intel.com)
- NetApp (www.netapp.com)
- Sun Microsystems (www.sun.com)

PART I.
Introduction & Planning

Chapter 1. Introduction

For many years now, customers have been using VMware Infrastructure to enhance their existing business continuity and disaster recovery (BCDR) strategies, and to provide simplified BCDR for existing x86 platforms running virtual machines on VMware ESX™. The VMware ESX hypervisor provides a robust, reliable and secure virtualization platform that isolates applications and operating systems from their underlying hardware, dramatically reducing the complexity of implementing and testing BCDR strategies.

In simple terms, this involves the implementation of both non-replicated and replicated storage for the virtual machines in a given deployment of VMware Infrastructure. The replicated storage, in most cases has built-in replication capabilities, which are easily enabled. Replicating the storage presented to the VMware Infrastructure, even without array-based replication techniques, provides the basis for a BCDR solution. As long as there is sufficient capacity at the designated BCDR site, the virtual machines be protected independent of the underlying server, network and storage infrastructure; even the quantity of servers can be different from site to site. This is in contrast to a traditional x86 BCDR solution, which typically involves maintaining a direct 1:1 relationship between the production and BCDR sites in terms of server, network and storage hardware.

Replicating the storage and live virtual machines is simple, yet powerful, concept. However, there are a number of considerations to be made to implement this type of solution in an effective manner. To build a generic BCDR solution is extremely complex and most implementations both physical and virtual, while often automated, are heavily customized.

A number of VMware customers have built successful implementations based upon these basic principles. This VMbook documents these principles and also provides a practical guide to implementing a working BCDR solution with specific hardware and software components. By building and documenting a specific solution, it is possible to illustrate in real-world terms how VMware Infrastructure can be utilized to as an adaptable solution for multisite deployment.

Why Read this VMbook?

Unlike white papers, which merely provide analysis and prescriptive advice, this VMbook provides a step-by-step process for implementing VMware Infrastructure as a cost-effective BCDR solution to support the most common scenarios. The BCDR solution also provides instruction on how to fail back services to the designated primary datacenter after a scheduled test or business service interruption.

By following the guidelines in this VMbook, readers will be able to achieve the following objectives:

- **Create a scalable, fault-tolerant and highly available BCDR solution.** This VMbook demonstrates how to utilize VMware Infrastructure for both server- and desktop-based virtual machines that support both scheduled BCDR testing, as well as unplanned disaster events.
- **Demonstrate the viability of virtualization-based BCDR.** VMware provides customer-proven solutions that are designed to meet the availability needs of the most demanding datacenters. This VMbook will help readers demonstrate the viability of using VMware solutions for BCDR in both testing and production environments while continuing to leverage existing tools, processes and policies.
- **Reduce resistance to change and mitigate "fear of the unknown."** Virtualization is becoming ubiquitous, and this VMbook will help readers demonstrate the straightforward and undisruptive nature of managing availability with VMware Infrastructure overcoming resistance to change and dispelling common myths and misconceptions about virtualization.

What's in this VMbook

This VMbook explains the overall process and provide a detailed explanation around key issues such as storage replication and the management infrastructure necessary for operating the virtual machines in an appropriate way in the designated BCDR site. This document also discusses how to complete a failback of services after a disaster event.

To provide a framework for this VMbook, the authors architected and built a multisite virtual infrastructure datacenter that includes all the necessary infrastructure components: networking; storage with a data replication component; physical servers, Active Directory, with integrated DNS; and VMware virtualization to demonstrate how to execute a BCDR failover from the production site to the designated BCDR site in a semi-automated fashion by leveraging the VMware infrastructure as well as the VMware VI Perl Kit[1].

[1] http://www.vmware.com/support/developer/viperltoolkit/

What's Not in this VMbook

This VMbook will not guide the reader through the development of a detailed business continuity plan, as the development of such a plan is a function of the business and falls outside of the scope of this VMbook. It is worth stressing that the development of a detailed business continuity plan, the ongoing updates to the plan, along with the exercising of the plan on a regular basis will ensure the ultimate success of the business at time of disaster when faced with the activation of their services in their designated BCDR site.

This VMbook will not discuss VMware Site Recovery Manager in detail as it falls outside the scope of this VMbook. Site Recovery Manager is a new product from VMware that delivers pioneering disaster recovery automation and workflow management for a VMware virtualized datacenter. Site Recovery Manager integrates with VMware Infrastructure and VMware VirtualCenter to simplify the setup of recovery procedures, enabling non-disruptive testing of recovery plans and automating failover in a reliable and repeatable manner when site outages occur. For more information, visit the Site Recovery Manager Web page[2] or read the Site Recovery Manager Evaluator's Guide[3].

That said, this VMbook will provide very valuable insight into the considerations and design principles for a multisite virtual datacenter that includes array-based replication to facilitate the replication of VMFS datastores—a key prerequisite for implementing Site Recovery Manager. Therefore, this VMbook can be leveraged as a reference when planning to implement a Site Recovery Manager as a BCDR solution, providing principled guidance for the design and deployment of a robust, reliable multisite virtual datacenter.

---

[2] http://www.vmware.com/products/srm/

[3] http://www.vmware.com/pdf/srm_10_eval_guide.pdf

Chapter 2. Understanding and Planning for BCDR

This chapter provides introductory guidelines to reference when designing a BCDR strategy.

Technology alone is no guarantee of a rock-solid BCDR strategy. There is a significant amount of work that needs to be carried out that involves working directly with the various business units to document all the business processes, which then need to be mapped to the underlying business applications that support these business processes.

The service level agreements (SLAs) as they relate to recovery point objectives (RPOs) and recovery time objectives (RTOs) for each business process needs to be determined, documented and then related to each of the underlying business applications. The next task is determine how those business processes map to business users who use the business applications services during their daily operations, and lastly how all of this maps to underlying physical and virtual systems. Working out all of these relationships can be a complex process Depending on the size of the organization, these activities could take anywhere from a couple of weeks to as long as 12 months or more. Figure 2.1 illustrates a typical high-level BCDR workflow process.

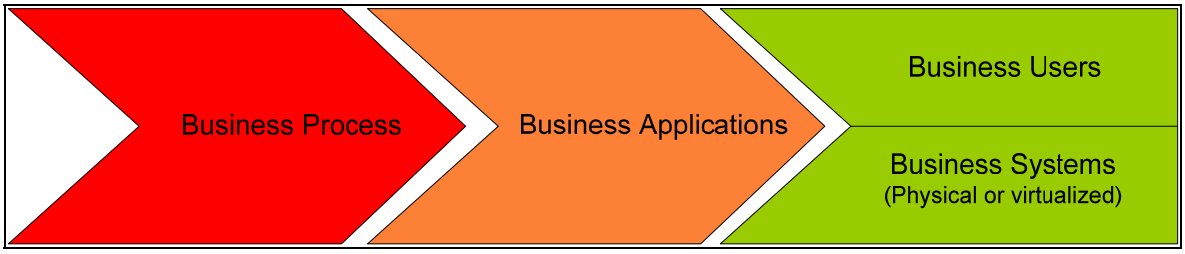


Figure 2.1 – Typical BCDR planning workflow process

In most instances, the work with the business units is typically completed by the members of the business continuity program (BCP) team who traditionally are not members of the information technology team. The members of the BCP team are more focused on the business processes and how these business processes rank in priority with respect to a restart of the business after a disaster event. In addition to the business process priority, the upstream and downstream dependencies of these processes also need to be understood and documented.

The list of business applications will also need to be mapped to systems both physical and virtual along with their appropriate dependencies. To generate this system mapping, the BCP team must work closely with the IT team that will assist the BCP team in generating the system list by working off the business application list. The resulting system list forms the basis of the BCDR plan, which is implemented in part by the virtualization team and other members of the information technology teams that are responsible for the non-virtualized business applications services and infrastructure that are required during a disaster event or during a scheduled BCDR test.

This VMbook assumes the BCP team has already completed the above process, often referred to as a business impact analysis (BIA) study, and has provided the IT team with the final systems list needed to build out the BCDR strategy. Detailed discussions on what it takes to complete a comprehensive BIA study are beyond the scope of this VMbook.

Design Considerations when Planning for BCDR

Network Address Space

There are really two scenarios to be considered from a network perspective:

- **Scenario 1.** Disparate networks in the designated production site and BCDR site.
- **Scenario 2.** Stretched VLANs across the designated production site and BCDR site.

Depending on the scenario, there will be implications when failing over services. With Scenario 1, there is a need to assign IP addresses for the failed over services, update the IP information on the failed over services and ensure DNS entries are updated correctly. With Scenario 2, there is no need to Re-IP and complete DNS updates for the failed over services to be restarted on the same network segment that is extended from the production site to the BCDR site.

Datacenter Connectivity

If the intent is to provide BCDR services based on array-based data replication (as this the intent in this VMbook), then a dedicated point-to-point connection is required between the two sites. The SLAs for WRT to RPO and RTO will ultimately drive the amount of bandwidth that is required to sustain the agreed upon SLAs of the business.

Storage Infrastructure

To build a BCDR solution that leverages capabilities such as live virtual machine migration, failover and load balancing, the SAN infrastructure must be configured to replicate between the production environments.

- Choices here could be iSCSI or Fibre Channel.
- Datastore type choices are VMFS, RDM or NFS.

Server Type

There are two basic choices when selecting physical servers to host VMware ESX:

- Traditional rack servers
- Blade servers

The choice of server type does have implications for infrastructure cabling. Blade servers greatly reduce cabling requirements (power, network, fiber) through the use of shared network and SAN switches that are integrated into the blade chassis, resulting in fewer network and fiber interconnects into the core network and SAN fabric switches when compared to deploying the same number of rack servers. For example, 14-blade servers in a blade chassis will require substantially less cabling when compared to deploying 14-rack servers of the same CPU socket and memory footprint.

DNS Services

DNS Infrastructure design and topology selection is beyond the scope of this VMbook. However, from a DNS Infrastructure / topology point of view, organizations must decide whether to:

- Use a dedicated DNS infrastructure to facilitate BCDR testing, as well as service failover at time of disaster that is isolated from the production DNS infrastructure.
- Use the same production DNS infrastructure that is configured to span geographically dispersed datacenters during your BCDR testing or service failover at time of disaster.

Active Directory Services

Active Directory design and topology selection is beyond the scope of this VMbook. However, as with DNS, organizations must choose whether to:

- Use a dedicated Active Directory to facilitate BCDR testing, as well as service failover at time of disaster that is isolated from the production DNS infrastructure.
- Use the same production Active Directory that is configured to span geographically dispersed datacenters during BCDR testing or service failover at time of disaster.

VirtualCenter Infrastructure

Automating the re-inventory of virtual machines in the BCDR datacenter (achieved in this VMbook via scripting) requires the deployment of a VMware VirtualCenter instance and supporting backend database in both datacenters.

NOTE: VMware Site Recovery Manager also requires a VirtualCenter instance in each datacenter to allow for the inventory of protected virtual machines and the creation of the Site Recovery Manager recovery plan on the VirtualCenter instance that is associated with the backup datacenter.

VMware ESX Host Infrastructure

The number of VMware ESX hosts required in each datacenter will ultimately be determined by the number of virtual machines needed to service in each datacenter. If the BCDR datacenter is also used to run development and testing (a common practice for some VMware customers), this will need to be taken into consideration when calculating the number of VMware ESX hosts required in the BCDR datacenter at time of disaster. It also affects whether or not development systems will be powered off to make resources available for the services that are being failed over from the production datacenter during the time of disaster.

Data Protection

This VMbook assumes that a backup infrastructure already exists and that data backups within the virtual machines are completed via the traditional backup methodologies used in the physical world. A backup agent is installed within each virtual machine, and the data backup-and-restore process is controlled by a master backup server.

Design Assumptions for this VMbook

- The network address space in each datacenter is disparate. Each datacenter will make use of static and DHCP IP addresses for the virtual machines.
- Connectivity between the two datacenters is via a dedicated circuit and not via VPN connectivity over the Internet.
- There is a single Active Directory that spans both datacenters and provides the following services:
    - User and Service authentication
    - DNS namespace services.
    - DHCP services for virtual desktops (VDI) and certain virtualized server workloads that can accommodate an automatic DHCP IP address change when floating between datacenters.
- Each datacenter is serviced by its own instance of VirtualCenter. There will be no replication of the VirtualCenter databases between datacenters.
- Data backups within the virtual machines are completed via the traditional backup methodologies that are used in the physical world. A backup agent is installed within each virtual machine and the data backup and restore process is controlled by a master backup server.
- For the purposes of the solution detailed in this VMbook, there will be a total of four VMware ESX hosts in **Site 1** (Production), to service virtual machines local to the datacenter on non-replicated storage, as well as the virtual machines that will float between datacenters via "data replication" on designated replicated storage.
- The four VMware ESX hosts in Site 1 will be logically grouped into two Recovery Groups to facilitate a partial failover of either **Recovery Group 1** or **Recovery Group 2** or a complete datacenter service failover of both Recovery Groups.

NOTE: Virtual machines on local non-replicated storage will not be failed over as these services are typically bound to the local datacenter. Services of this type are typically:

- Active Directory Domain Controllers
- Virus Engine and DAT update servers
- Security services (HIPS and NIPS)
- Print services
- And so on…

- **Site 2** contains a total of two VMware ESX hosts designated for BCDR, and two hosts designated for development. The two BCDR hosts will be able to service failed over virtual machines from one of two recovery groups: **Recovery Group 1** or **Recovery Group 2**. Should a total Site 1 failover be orchestrated, the two designated development hosts can be leveraged to provide the additional resources required to sustain the services failed over from Site 1, this will be accomplished by either shutting down the development environment or leveraging nested resource pools to throttle back resources assigned to the development environment.
- The BCDR solution calls for a SAN infrastructure with connectivity from the VMware ESX hosts in both datacenter over Fibre Channel to fabric switches for connectivity into the SAN.
- The VMFS data replication between the two datacenters will be array-based and determined by the type of SAN implemented in the BCDR solution.
- The re-inventory of the replicated virtual machines will be automated through the use of scripts that leverage the VMware SDK.

NOTE: VMware Site Recovery Manager completes the re-inventory of replicated virtual machines via the Site Recovery Manager configuration workflows which removes the need to create custom scripts to complete the virtual machine re-inventory tasks in site 2.

- Where required the re-IP of virtual machines that were failed over from Site 1 to Site 2 will be automated via scripts that leverage the VMware VI Perl Kit. The same will be true for virtual machines that are failed back from Site 2 to Site 1.
- VirtualCenter version 2.02 was used in each datacenter.
- VMware ESX Server (aka VMware ESX) version 3.02 was used in each datacenter.

NOTE: At the time this environment was built out, VirtualCenter 2.5 and VMware ESX 3.5 were not generally available. That said, the solution presented in this VMbook will work on VirtualCenter 2.5 and VMware ESX 3.5 as the concepts and design principles do not change with these later releases.

- VMware HA and VMware DRS will also be used in each datacenter to demonstrate fault tolerance and dynamic load balancing in addition to the data replication of the VMFS to support the BCDR solution.
- The VMware VI Perl Kit will be leveraged to build in the necessary automation to inventory and to re-IP virtual machines that are floating between datacenters via the data replication technology configured in the BCDR solution.

Chapter 3. Virtualization and BCDR

This chapter describes several key virtualization concepts as they relate to BCDR, as well as the properties and capabilities of VMware virtualization software that make it possible to build a robust, reliable and cost-effective BCDR solution.

Virtual Machines as a Foundation for BCDR

Virtual machines have inherent properties that facilitate the planning and implementation of a BCDR strategy.

- **Compatibility**. Virtual machines are compatible with all standard x86 computers.
- **Isolation**. Virtual machines are isolated from other each other as if physically separated.
- **Encapsulation**. Virtual machines encapsulate a complete computing environment.
- **Hardware independence**. Virtual machines run independently of underlying hardware.

The sections below describe these properties in greater detail.

Compatibility

Just like a physical computer, a virtual machine hosts its own guest operating system and applications, and has all the components found in a physical computer (motherboard, VGA card, network card controller, etc). As a result, virtual machines are completely compatible with all standard x86 operating systems, applications and device drivers, so you can use a virtual machine to run all the same software that you would run on a physical x86 computer.

Isolation

While virtual machines can share the physical resources of a single computer, they remain completely isolated from each other as if they were separate physical machines. If, for example, there are four virtual machines on a single physical server and one of the virtual machines crashes, the other three virtual machines remain available. Isolation is an important reason why the availability and security of applications running in a virtual environment is superior to applications running in a traditional, non-virtualized system.

Encapsulation

A virtual machine is essentially a software container that bundles or "encapsulates" a complete set of virtual hardware resources, as well as an operating system and all its applications, inside a software package. Encapsulation makes virtual machines incredibly portable and easy to manage, and VMware has built an array of technologies that take advantage of this portability and manageability to facilitate BCDR services.

Hardware Independence

Virtual machines are completely independent from their underlying physical hardware. For example, you can configure a virtual machine with virtual components (eg, CPU, network card, SCSI controller) that are completely different to the physical components that are present on the underlying hardware. Virtual machines on the same physical server can even run different kinds of operating systems (Windows, Linux, etc).

When coupled with the properties of encapsulation and compatibility, hardware independence gives you the freedom to move a virtual machine from one type of x86 computer to another without making any changes to the device drivers, operating system, or applications. Hardware independence also means that you can run a heterogeneous mixture of operating systems and applications on a single physical computer.

Virtual Infrastructure: A True Enabler for Sitewide BCDR

While the hypervisor provides a virtualization platform for a single computer, VMware technology provides the means to create an entire *virtual infrastructure* that aggregates the IT infrastructure, from the datacenter to the desktop, into flexible *resource pools* that map physical resources to business needs.

The VMware Infrastructure software suite creates a virtual infrastructure "layer" that decouples computing, networking and storage resources from their underlying physical hardware. Structurally, the virtual infrastructure layer consists of the following components:

- Single-node hypervisors ("virtualization platforms") to enable full virtualization of each x86 computer.
- A set of distributed infrastructure capabilities to optimize available resources among virtual machines across multiple virtualization platforms.

- Application and infrastructure management capabilities for controlling, monitoring and automating key processes such as provisioning, IT service delivery and BCDR.

The sections below describe these components in greater detail.

Virtualization Platforms

Hypervisors, also known as *virtualization platforms*, managing and monitor virtual machine access to hardware resources on a single physical computer. In general, virtualization platforms manage access to four core hardware resources:

- **Computing.** VMware virtualization platforms allow virtual machines to share access to 32- and 64-bit single-core and multicore CPUs, with support for up to four-way virtual symmetric multiprocessing (SMP).
- **Memory.** The VMware ESX hypervisor provides dynamic access to memory with management mechanisms such as *RAM overcommitment* and *transparent page sharing* that automatically expand or contract the amount of physical memory allocated each virtual machine as application loads increase and decrease.
- **Networking.** VMware virtualization platforms provide access to physical network adapters and also offer the ability to implement virtual LANs with virtual switches for network connectivity between virtual machines on the same host or across separate hosts.
- **Data storage.** VMware ESX allows virtual machines to access data stored on internal storage disks, or on shared storage devices such as Fibre Channel and iSCSI SANs, as well as NAS devices.

Not all hypervisors are the same. Some, such as VMware Workstation and VMware Fusion™, utilize "hosted" virtualization platforms that run as applications on a host operating system such as Windows, Mac OS® X or Linux. For BCDR, it is best to use a "bare-metal" hypervisor such as VMware ESX that runs directly on the computer hardware without the need for a host operating system. The bare-metal approach offers greater levels of performance, reliability and security, and is better equipped to leverage the powerful x86 server hardware found in most modern datacenters.

Distributed Infrastructure Capabilities

In addition to the hypervisor, VMware Infrastructure includes a set of distributed infrastructure capabilities that allow IT organizations to optimize service levels with failover, load balancing and

sitewide disaster recovery services for virtual machines. These services revolve around two key virtual infrastructure concepts: clusters, and resource pools.

VMware Cluster: A shared computing resource

A VMware Cluster is a group of individual VMware ESX hosts and associated components that provide a shared computing resource where the CPU and memory of that group can be considered as an aggregate pool. Initial implementations of virtual clusters used a shared storage mechanism to allow co-operation between the discrete server components; this is now known as the VMware Virtual Machine File System (VMFS).

VMFS: A Cluster File System for Virtual Machines

VMware VMFS is a cluster file system, optimized for virtual machines, that allows multiple VMware ESX hosts to share a common storage resource. This technology was released over four years ago and underpins the virtual infrastructure concept as well as most of the following technology components. Recent enhancements to VMware Infrastructure allow the use of other file system technologies, as well. In the first instance, the use of the network file system (NFS) as a storage resource through the VMware ESX datastore primitive. The datastore, be that VMFS- or NFS-based, provides the encapsulation technology that allows the virtual machines to be replicated as complete entities. When multiple VMware ESX hosts are joined via a shared storage resource and are managed by VirtualCenter, this is referred to as a *virtual cluster*, or simply a cluster.

- **High Availability (HA) clusters.** High availability services can be enabled at the cluster level. Checking a single checkbox enables failover protection for any workload, independent of operating system or application.
- **Distributed Resource Scheduler (DRS) clusters.** As with VMware HA, this feature can be enabled at the cluster level to automatically load balance any virtual machine placed in that cluster or enclosed resource pool. This allows for dynamic service level management of discrete groups of virtual machines, and is particularly useful when dealing with workload spikes in a policy-centric fashion.

Each root resource pool is aggregated in the cluster as a single entity. If there are four servers in the cluster, each with four CPUs, the clustered resource pool will have 16 CPUs, effectively extending the resource pool across multiple physical servers. These resources then can be subdivided by a central IT administrator, or by individual departmental units or application/service owners, without regard to the structure of the underlying hardware.

VMotion: Non-Disruptive Migration for Virtual Machines

VMotion is a VMware technology that provides the ability for virtual machines to move from physical host to physical host within a cluster without experiencing any downtime. This capability powers VMware HA and VMware DRS and, along with VMFS, provides the underlying foundation for hardware-independent disaster recovery.

Application and Infrastructure Management

VMware VirtualCenter provides centralized management for virtual machines and their VMware ESX hosts, allowing all of the functions and the configuration of the VMware ESX hosts, virtual machines, and virtual networking and storage layers to be managed from a single point of control. From a BCDR perspective, this is useful in that a central interface can be used to perform group wide functions (for example, to power on two hundred virtual machines).

NOTE: VMware Site Recovery Manager enhances and extends the capabilities of VMware VirtualCenter, leveraging array-based replication between protected sites and recovery sites to automate and optimize business continuity and disaster recovery protection for virtual datacenters. If a disaster occurs, Site Recovery Manager helps to quickly restore critical IT services, dramatically shortening the duration of a business outage. Site Recovery Manager is based on existing IT setup using virtual machines that VMware VirtualCenter manages. The Site Recovery Manager architecture ties workflow automation to third-party storage replication.

Leveraging Virtual Infrastructure for BCDR

Virtual Infrastructure provides the technology to combine groups of servers and manage them as an aggregated resource pool. Resource pools are an ideal way to abstract the underlying physical servers and present logical capacity, not the physical computers underneath.

From a service management perspective, resource pools provide a mechanism to solve some of the potential issues discussed in the partitioning section above. Additionally, they give the ability to effectively provide a fractional service. "In BCDR the service level will be 66 percent of production," but the cost of providing that BCDR service would be commensurate with that.

VMware Infrastructure provides mechanisms to test BCDR plans in complete isolation. The next step is to test the logical application functionality. In a physical environment, this can be very challenging as bringing up the BCDR environment essentially means taking the production system down. However in

a virtualized environment, organizations can power up complete services in isolation and test them accordingly without having to suspend live services.

VMware DRS, resource pools and clusters provide another feature which is difficult to envisage in the physical world. BCDR planning may make some sort of assumption about the length of time it would take to recover the production site (two weeks, three months, etc.). This is very often the case when entering into outsourced or shared BCDR facilities. If this period of time becomes extended, then the SLA agreed with the business as a short term acceptable compromise may become unsustainable. With VMware Infrastructure, it is possible to expand (or contract) the service with the addition of extra server capacity seamlessly and without down time to the guest workloads.

Further Reading

1. VMware Infrastructure 3 Architecture Overview:
   http://www.vmware.com/resources/techresources/410
2. VMware VirtualCenter Technical Best Practices:
   http://www.vmware.com/pdf/vc_technical_best.pdf
3. Configuring Virtual Machine Storage Layers in VMware Infrastructure:
   http://www.vmware.com/pdf/storage-layers-wp.pdf
4. VMware Virtual Machine File System – Technical Overview and Best Practices:
   http://www.vmware.com/resources/techresources/996
5. Resource Management with VMware DRS:
   http://www.vmware.com/pdf/vmware_drs_wp.pdf
6. VMware HA – Concepts and Best Practices:
   http://www.vmware.com/resources/techresources/402
7. VMware Virtual Networking Concepts:
   http://www.vmware.com/files/pdf/virtual_networking_concepts.pdf

PART II.
Design & Implementation

Chapter 4. High-Level Design Considerations

This section outlines some of the design considerations that may affect the approach taken when implementing a design such as the one undertaken here. Building the entire design from a green field perspective is a luxury most organizations don't have, as is the simplified environment used to demonstrate the guidance set forth in this VMbook. In an actual implementation, it is likely to encournter a larger number of LUNS, virtual machines and hosts than the ones shown in Figures 4.1 and 4.2. It is also likely that the Site 1 infrastructure will already be in place and running. This inevitably leads to some additional complications, but realistically, these must be addressed as well as the issue of maintenance over time in order to successfully build an effective BCDR solution.

Hardware Components

Servers

Virtualization allows this decision to be fairly flexible, but sufficient capacity must exist in the remote site to operate the productions load – even if that is agreed to be some fraction of the normal load. Additionally, if the remote site also has a production workload, some mechanism must be considered to ratio the resources appropriately.

Storage

Here are some questions and considerations to take into account when designing the storage component of a BCDR solution:

- Is array-based replication going to be used? Will it be synchronous or asynchronous?
- Will virtual machines be replicated by a tape mechanism?
- Is there a need to protect against corruption by maintaining a point in time copy behind the primary copy?
- What is the granularity of failover going to be?

Cost and distance between sites are normally the main drivers behind these decisions.

Networking

Some IP mobility will be required for the solution to work. There are a number of ways to achieve this, which are discussed later. Hardware components such as load balancers may be part of the solution and will need to be geographically dispersed.

System Management

The management infrastructure will also have to survive the failure so duplication or replication will also have to be considered for this aspect of the service.

Time

Time factors a number of the design decisions.

- The ability to failback especially is governed by the time spend in BCDR and the rate of change of data.
- SLA required. For a short period of time the business might deem a lower level of service acceptable while 'normal' service is resumed. The duration that the business will accept this lower level of surface is often undefined although the business will have some expectations. Active – Active designs have the ability to absorb existing workloads; this may be to the detriment of other services often development and test. How long can the business sustain operations without being able to test patches or in the longer view test and deploy new applications?
- How quickly do I need to recover (RTO)? Many solutions to network and storage failovers can take significant periods of time. Large DNS "pushes" can take several hours.

Typical Configuration

Figure 4.2 shows the target architecture with both Site 1 and Site 2 and a regular relationship between the LUNS and replication groups. In the first instance, this may not be the case, so to make the replication strategy simple some alignment work will need to be done. This will be required, even if the intention is only to deal with the case of a complete site failover, as it leads to an understanding of the dependencies of the applications and services needed in a BCDR scenario.

Moving a large number of IT services to a different location is a complex task, and understanding the detailed relationships between them is the first step in making this possible. There are some tools

available to assist in this mapping process; SMARTS from EMC would be an example of such a product. Virtualization does make this somewhat simpler to achieve, but the interrelationship problem is common in both the physical and virtual worlds. A number of considerations should be made prior to making any design choices. These roughly fall into the following categories:

- Granularity of failover
- Replication
- Resource management
- Namespace mapping
- VI Networking

Granularity of Failover

The high-level business plan will drive a macro level view of what pieces of the overall estate are critical; it is unlikely that this plan will have any detailed view of the sub-services that are required to run the services and any sequence inherently involved. The high-level plan may exclude explicitly a number of services which do not need to be accounted for, but for the remainder there needs to be an understanding of the relationships between the storage, other applications, and so on. It may also be desirable to be able to selectively fail over parts of the business rather than just plan for a complete failover. The former approach is the one adopted in this VMbook. In order to be able to selectively achieve this functionality a number of considerations must be made with respect to particular storage dependencies, but also including application dependencies.

Storage Alignment

If using array-based replication in which LUNS are "randomly" distributed amongst the physical or logical storage groupings, it is going to be very hard to isolate specific groups of workloads. For example, three virtual machines with two logical LUNS would have dependencies on six physical LUNS or VMFS volumes – top half of Figure 4.1. Relationships could exist between two of the virtual machines as they could potentially share a volume (VMFS) by each having a LUN on a shared volume – lower part of figure 4.2. That relationship essentially binds them together from a BCDR perspective. In Figure 4.1, the lower alignment would mean that all three machines are either protected or not protected, and without complex procedures it will be impossible to bring any one virtual machine up for test purposes without potentially compromising the other two.

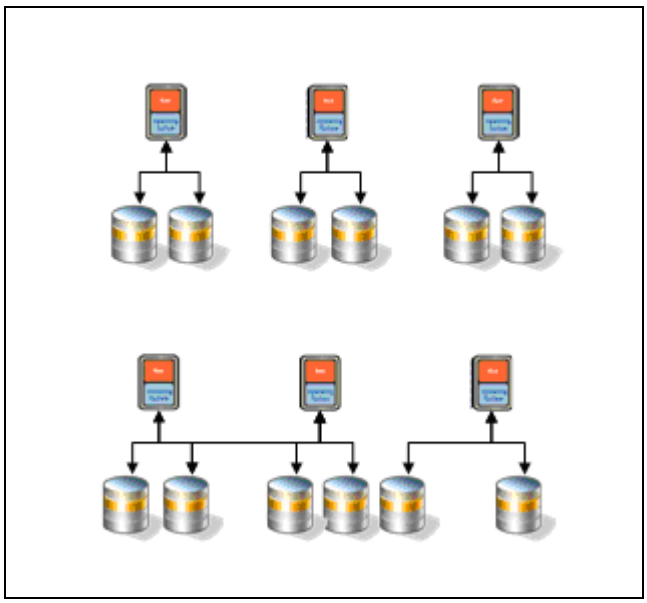


Figure 4.1: LUN-to-virtual machine relationship

In fact, with modern storage techniques, this problem can be circumvented in a test case, but an operational failover model this would certainly be ruled out if an organization wants just one of the machines.

The authors of this VMbook were able to enforce a good alignment of applications; however, it is likely that some operational best practice would have to be implemented to maintain this situation and in a non-greenfield site, some disk re-organization may have to be performed – potentially using technologies such as VMware Storage VMotion.

Applications

As with storage, it may also be useful to run a similar exercise to produce an application model and dependency mapping. Relationships between applications are also important to understand. If partially fail over occurs, the organization must understand the implications of having related workloads move to different sites, and what the additional network latency might mean operationally. A number of organizations have adopted a cell, pod, or grouping function of some sort. Here a regular unit of compute, network and storage is deployed, this may be a relatively large amount of physical infrastructure, and related services are deployed into these cells as needed. An IT process tries to

"affinity" the services to known groups to minimize the number of external dependencies required for that "pod" to run effectively at any location. In this case, the authors aligned the applications by business unit and then allocated the storage in the same way.

This process may not work if IT consumption is modest (10 to 30 workloads/guests), but it is a model that has merit on even a medium scale. The authors of this VMbook deployed approximately 70 virtual machines in total, and it can be seen in the later chapters that a small number of virtual machines benefits from some alignment.

Infrastructure Services

The high-level business design may not specify business applications explicitly so services that these applications depend on will certainly not be specified. Examples of these types of services could be DNS, Active Directory. The DR plan must accommodate these but frequently we find they are left out. For many of these types of applications it is ok to leave them out as it is much easier to duplicate them, this is the approach taken in this book. You may consider this as part of the application mapping above but in our experience these are dealt with separately by most organizations and typically are where plans can fall down, typically because they get overlooked. A good example is an application that makes a hardcoded assumption about the IP address. Many of these services are relatively stable over time; this is probably why they get overlooked. However, they generally are easy to distribute/duplicate or recover from scratch. In our approach we duplicated network services and replicated domain services to both sites. This leads to a simple approach and seems to be effective in most cases we have seen.

Data Replication

The granularity study will be invaluable in understanding the replication strategy that you want implement. The main considerations here are: rate of change and groupings. Groupings are driven by the granularity study which we referred to above. Then a deeper level of analysis is required to establish the rate of change which determines WAN bandwidth requirements etc. This in conjunction with the Recovery Point Objectives can drive decisions around synchronous and asynchronous replication technologies. With these in mind it is then possible to have detailed discussions with the storage teams to line up the capabilities of the storage layer and its granularity to ensure a good mapping between the two.

Of course there are different ways of achieving the replication, we used the array technology to achieve this step but as long as the RPOs are met it may be possible to achieve a level of replication via

a continuous backup recovery cycle using VMware VCB for example instead of using an array based solution.

With a replication strategy comes the question "how long do you want to be in DR"? This seems to be often overlooked. It can have a large impact on the design considerations. Firstly the scope of the DR changes substantially: if you put some long times in your scenarios – i.e. I can never go back, or Site 1 is totally destroyed and takes 18 months to rebuild. This can be particularly significant clearly if you have to co-locate with rental charges etc. However in most scenarios the replication function at some stage will need to be reversed or maybe Site 2 becomes your permanent primary site. Depending on the duration and rate of change at some stage it will be more optimal to start the replication process from scratch rather than catch up from snapshots/change logs etc. In our design we enabled enough storage to operate at DR capacity on Site 2 and have enough storage to give some level of protection and duplication capability in an extended DR scenario.

Resource Management

The target architecture chosen here was for an active-active design, which means that consideration must be given to the workload in the recovery site. What will happen to the active load when the additional load from the protected site is started? As discussed in Chapter 3, VMware Infrastructure provides a number of mechanisms to manage this. If you intend to run a reduced SLA with the business during a DR event, it may be worth considering different duration scenarios. What happens if we are in 'DR' for 6 weeks versus 6 months etc? This is where the application mapping and granularity work can help. For example, in a short term DR or the first part of a long term DR, you may choose not to re-instate your patch management infrastructure.

With the output of the granularity study and the resource management scenarios it will be possible to build a simple service catalogue concept. This will guide the use of resource pools and other mapping functions that will be required to make the DR design manageable over time. It is not likely that once implemented the design will be static for say a given number of virtual machines, you will want to add virtual machines and additional services over time so it is worth making these design decisions that can accommodate these additions over time.

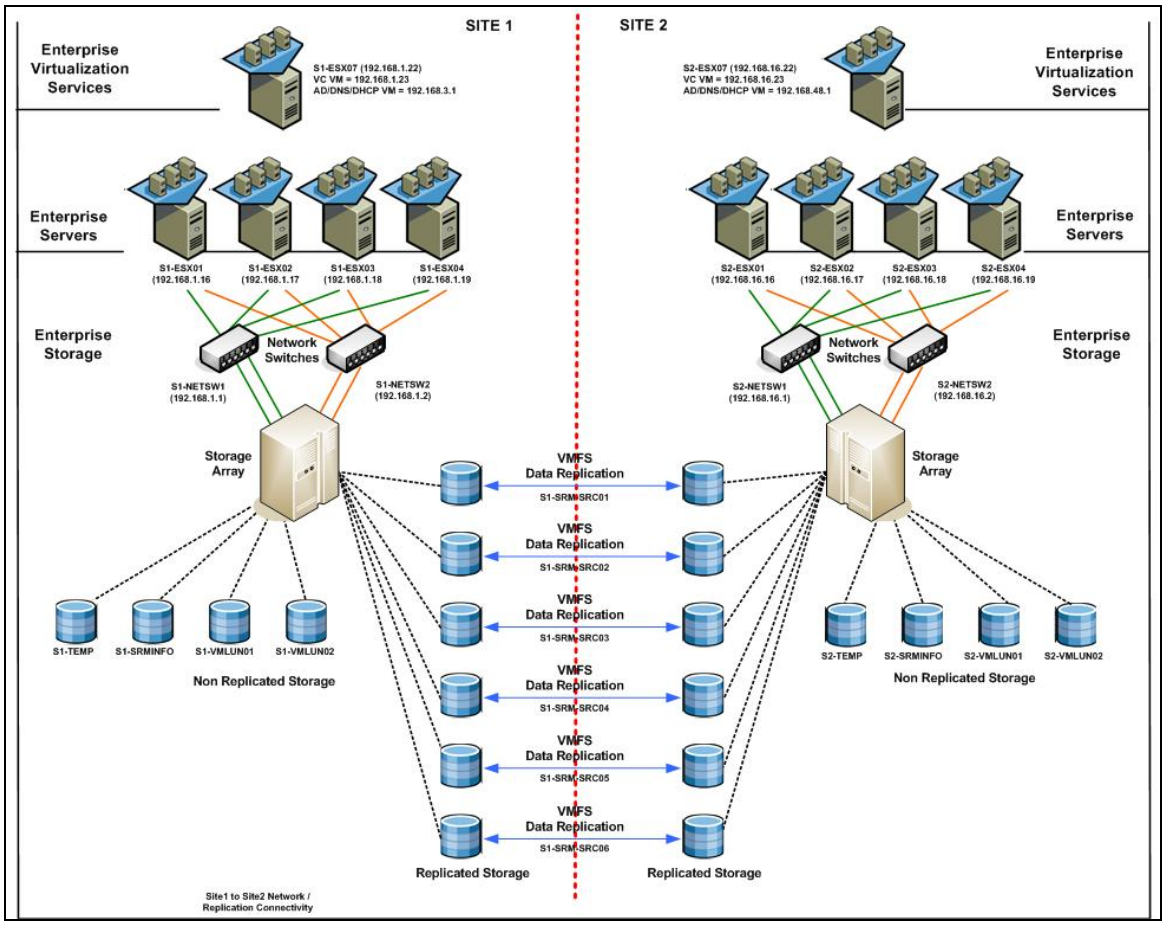


Figure 4.2: Virtualized datacenter with multisite BCDR

VirtualCenter Name Space Management

In any implementation naming conventions are key in allowing rapid understanding of what is happening or how best to implement change. This is even more important when the naming convention has to hold in more than one location. A sample list for consideration:

- Datastores
- Virtual machine naming
- Network portgroups
- Folder names
- Resource pools

For a more complete list, refer to http://vpp-dev-1.vmware.com/home/docs/DOC-1022.

Datastores

Data Stores will be replicated in some form or another. VMware Infrastructure provides a number of protection mechanisms to avoid unwanted duplicates and manage them effectively when they are desirable. Good naming conventions that are consistent from the LUN upward can greatly simplify the configuration and the understanding therein by new or inexperienced staff. The approach taken in this VMbook was to tag the datastore with its source location, business function (also aligned with our granularity scheme), and finally an index.

The snapshot capability in VMware Infrastructure was also turned off so that datastore names would persist across a fail over. Care should be taken with this approach as the safety features enabled by this feature are also turned off. VMware Infrastructure provides a feature that uniquely identifies the LUN via a signature record. If the LUN is duplicated via the array and presented as active, VMware Infrastructure will recognize by default that this is a duplicate and will not present the duplicate VMFS storage. By overriding this property, VMware Infrastructure in the remote site is allowed to immediately present the VMFS (post a rescan of the SAN luns) With that in mind, be especially vigilant about creating duplicates for non-BCDR purposes.

Virtual Machine Naming.

Consider a naming convention that highlights protected and non-protected virtual machines. The examples in figures 4.3 and 4.4 includes a tag that designates its home origin. However, this may not be suitable in an operational failover model, where the virtual machine may not have a notional "home."

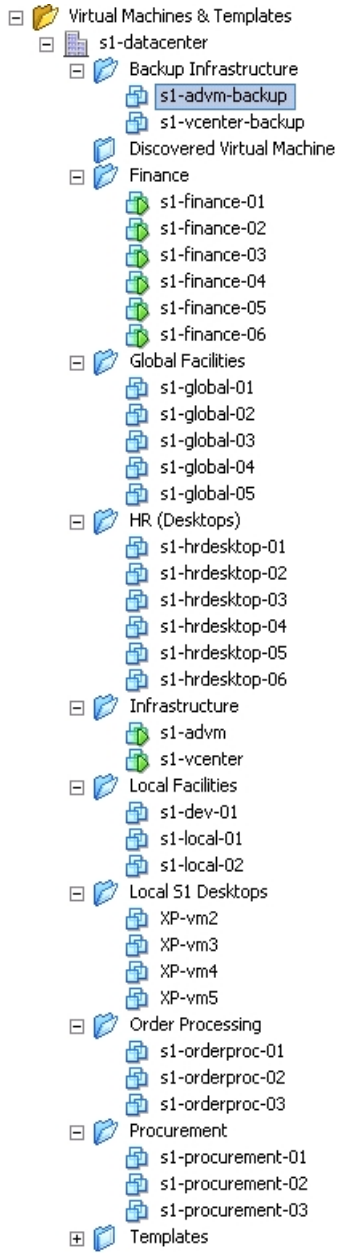


Figure 4.3: Site 1 Tree Folder

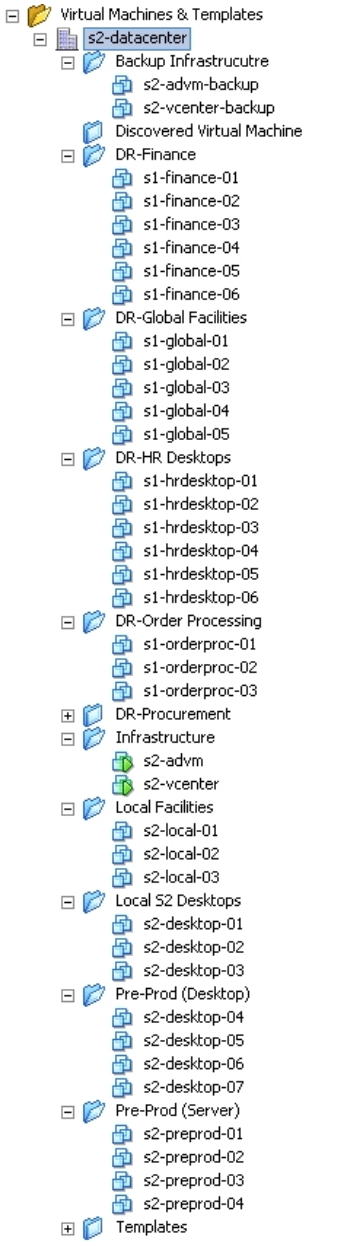


Figure 4.4: Site 2 Tree Folder

Folder Names

The authors of this VMbook used folders in Virtual Center to group logical functions together. While not strictly required, it is a useful feature when trying to locate particular virtual machines especially if the environment has a large number of virtual machines.

Networking

VMware Infrastructure provides a mechanism to allow the virtual NICS to be associated with a logical portgroup. The portgroup provides a number of controls which override equivalent controls applied at the virtual switch level. The virtual switch is then associated with a number of physical NICS. The name of the portgroup is used to associate the virtual NIC with the virtual machines that use it, so it is useful to consider the connection scenarios in both sites. The association is made in the VMX file. This means that the replication technology will bring this association with it. There are two potential considerations here then, first is if the portgroup name pre-exists and the second is when it doesn't.

Resource Pools

It is not considered good practice to use resource pools as logical or organizational containers for virtual machines. The implementation outlined in this VMbook had a close mapping between the resource pools and the folder structure for purposes of documentation. In a real-world implementation it is likely that the service catalogue idea above would be used to drive the placement of any one given virtual machine to a specific resource pool based on its performance profile.

Chapter 5. Implementing a VMware BCDR Solution

Design Considerations

For most organizations, the design of a BCDR solution is a fairly custom process. While the design principles and considerations are mainly common designers typically have to make a number of compromises. This section discusses the design principles used to establish the baseline design implemented in this paper.

In a real-world scenario, there would be an interaction with the business owners to establish SLAs and these would drive design considerations. The implementation outlined in this VMbook was designed to apply generically to as many cases as possible and was based in part on interviews with senior architects within the VMware customer base to determine a "level set" in terms of needs, requirements, and so on. Typical questions asked of these architects include the following:

1. What type of SLAs do you have with the business?
    a. Recovery Point Objectives
    b. Recovery Time Objectives
2. Do you use a third-party datacenter (rented space) or a dedicated facility that you own and operate?
3. How do you replicate data and systems configurations?
4. What level of granularity are you looking to achieve in a disaster situation? Just site failover or partial site failover?
5. How do you handle network failover, IP address space and other related issues?

Findings and Observations

This section summarizes the findings that guided the approach used in this VMbook.

1. Division of Responsibility

   An interesting finding was that most people have a division of responsibility in their BCDR process. One group typically had the responsibility to bring the operating environment up. There as then a second group responsible for getting the application platforms up after the operating environment is stable.

2. Secondary Site Availability

   The second finding, possibly more predictable, was that most had their own disaster recover sites or production sites that could double as DR sites.

3. Granular failover

   A common thread was also the move toward a more active-active or even three way solutions was desirable. Here spare capacity in the second production datacenter should be used to enable DR; obvious cost savings etc. Less expected was the desire to see granular fail over capability. In the situation where there are two discrete business functions within a company we may wish to DR one of the businesses functions but leaves the other running in situ. We believe this is routed in the desire to move toward a more operational failover capability and that complete site failure is not the only consideration that needs to be made.

4. Testing

   A large number of respondents were unable to test a failback scenario. Several technology issues arise here, such as the ability to easily reverse data flow on replicated LUNS,and situations in which site loss becomes a different recovery case from an operational failback.

5. Capacity Management

   Capacity management was a concern; for example, what happens if an operational load is moved to a second site with its own operational load?

6. Network Reconfiguration

   Network identity was a big problem. There was a mixture of opinion on the pros and cons of stretched VLANs and many clients have to re-ip servers in DR.

7. Storage

   Understanding the storage management complexity was also sited. Possibly because storage management is often a discrete function in many medium and large organizations.

8. SLAs

   RPO and RTO objectives were not consistent across the board so no real trend was noted here. In our design we used both synchronous and asynchronous replication schemes to simulate zero data loss scenarios. We also set an RTO of 30 minutes.

With these findings in mind, the following assumptions were made for the initial version of this VMbook:

- Assume two active datacenters.
    - Both fully operational with different workloads.
- Partial failover should be catered for.
- There should be a live instance of the VirtualCenter Server in both sites
    - As it turns out this simplifies a number of technical considerations.
- Resource pools will be used to abstract the servers and provide capacity management domains.
- Assume different IP name space in BCDR site and that all the workloads will have to undergo an IP address change.
- Assume array-based replication.
    - Replication is typically well understood but interaction with VMware Infrastructure less well so.
- Granularity. The authors of this VMbook decided to create a logical boundary that includes storage and CPU capacity. In a failover scenario one or many of these recovery groups should be capable of being failed over, and back, individually. It is assumed that application dependency between recovery groups is understood by the application teams.
- Assume sufficient memory and network capacity. In a real–world scenario, this would be an additional consideration. Network connectivity is less of an issue as the VMware Infrastructure

abstraction can isolate organizations from port count considerations. Memory, however is significant. Without enough minimum memory, the machines will perform badly (or in the worst case not restart) if Site 2 is constrained versus the incoming workload.

VirtualCenter Design

To meet a number of the desired design criteria, the authors of this VMbook used a two-site approach with an active VirtualCenter instance at each location. These are referred to as **Site 1** and **Site 2** throughout the documentation. Both sites are setup to have an active workload required to run the day-to-day activities present at each site. Additionally, at Site 1 there is a group of services that must be capable of being failed over to Site 2 and run there successfully. These are known as the protected services. The design will also accommodate work loads from Site 2 hosted on Site 1 although this was not implemented at this stage.

This approach has a number of benefits:

- There is no initial bootstrap problem of getting a replica of VirtualCenter operational; this is assumed to be in place at the alternative site.
- Sites can vary in their hardware content as far as capacity is concerned. The protected services are assigned to resource pools and not specific server entities. This would allow for different hardware, storage and network topologies to be deployed in both sites.
- We can actively use the resources in Site 2 for day-to-day workloads.

Potential downsides are:

- Requirement for additional VirtualCenter licenses
- During failover VirtualCenter inventory must be migrated from Site 1 to Site 2, or the new workloads must be mapped to a pre-existing standby inventory structure at Site 2.

The first is generally not an issue as the value obtained from the second estate is easily justified if not already in place. The second point is addressable by leveraging the VirtualCenter API, and can be fully automated. This step, however, is possibly the most crucial after data replication. While it is true that the isolation and encapsulation features of a virtual machine make BCDR substantially easier – just replicating the storage is only the first step and this next step is one of the issues observed most frequently in real-life implementations.

An alternative approach might be to replicate the VirtualCenter instance. This has the initial disadvantage in having to re-register any ESX hosts in the secondary site. This can be straightforward if the ESX hosts are essentially identical at both sites and of course can be automated through the VirtualCenter API. A variation of this approach is to replicate the ESX boot LUNS within the replication framework and have the Site 2 ESX hosts servers boot directly from these replicated LUNS. In this case the identity of the ESX servers is retained and the re-registration process is not required either for the hosts or the virtual machines. While this approach has merits from a simplicity perspective, it is really only suitable where there is a 1:1 mapping of production and disaster recovery hardware, which is not always available, and requires investment over time to maintain in that state.

In either case, the VirtualCenter instance in Site 2 is presented with the replicated LUNS and the virtual machines associated with them are then registered with the service. Once these two steps are complete we are then ready to manipulate the virtual machines as far as their network configuration etc. is concerned. This will be required with whichever VirtualCenter approach is used – this is covered in the networks section.

As shown in figure 5.1, the logical view of the configuration we have used in this paper. Site 1 holds virtual machines from an imaginary HR and Finance department and Site 2 hosts a number of Developer and other production services. Virtual machines can be executed in either site with protection for the machines in Site 1.

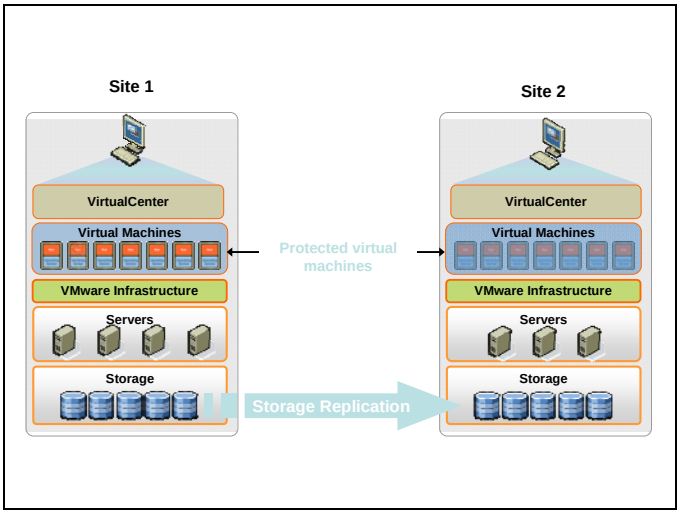


Figure 5.1 – Virtual Machine Storage Replication

Protection Groups and VirtualCenter

A protection group is a collection of virtual machines and associated storage that can be failed over to an alternative site. The LUNS/VMFS' associated with a protection group are assumed to be replicated at the array level and in the event of a fail over these will be presented on the second site.

Protection groups also make an assumption that in a failover an event the application functionality is either contained within or can tolerate the additional latency that might be incurred to other protection groups. If you want the ability to failover just one of these groups there is an assumption that the applications that were once close together can now work just as well in separate sites. If you visualize a set of virtual machines within one protection group they interact most likely via network connections to other protection groups or static non protected virtual machines. In a failover of that single protection group all of those external network connections will have to 'rubber band' to another site.

There are certain special cases that can be seen here a) fail over all protection groups in the event of a complete site failure and b) if BCDR requirements are modest, all of the required virtual machines can be placed into a single protection group. This VMbook will demonstrate the ability to fail some or all of the groups.

Dealing with the VirtualCenter Inventory Structure

Having made the decision to failover due to test or some real-life issue or disaster, presenting the protected storage at the alternative site is the first step. This is covered in more detail in the Chapter 10. Having presented the storage to the VMware ESX hosts in Site 2 and made them visible to Site 2, the Site 2 VirtualCenter instance must now be told about these new virtual machines. This process is known as "registration." VirtualCenter provides an organizational structure for this registration, called "folders." Registration can be performed prior to the failover or at invocation but it has to be performed before VirtualCenter can start/manage any of the services associated. The process can be manually achieved by using the datastore browser and using the right-click options to "register" a virtual machine. This is quite straightforward, but for many hundreds of virtual machines would be time consuming.

Folders and Resource Pools

VirtualCenter has a number of mechanisms to control organization (folder structure) of virtual machines as well as capacity management (resource pools). It is worth thinking about the business

organization at some depth to make sure the following design will work in an operational sense for your particular organization. Additionally, it is worth understanding that VirtualCenter views (hosts and clusters/virtual machines and templates) are distinctly held in the VirtualCenter object namespace so are actually not related at all. This can be a little confusing to the first-time VirtualCenter user as Hosts and Clusters view has the ability to incorporate a folder structure in addition to the folder structure present in the virtual machine view, so it is worth spending some time becoming familiar with these concepts. In this VMbook, the authors have adopted the use of both structures to give maximum flexibility while giving reasonable levels of organization for the virtual machines.

Having developed the organization model in this approach, it was then necessary to map this structure to the paired VirtualCenter and have that be a subset of that Servers own organization with the intention to use the second site in an active mode.

Additionally, it will be a requirement to maintain this structure over time and either a) produce an operational procedure that keeps both structures in sync, or b) create an automatic process that achieves the same thing, so that as new virtual machines are added to the production side, its shadow is created in the BCDR structure.

To logically organize virtual machines into protection groups Virtual Machine & Templates view was used to develop a three level structure. At the second level, nodes in the tree with the naming convention "ProtectionGroup N" contain all of the virtual machines in a specific protection group. Each protection group has specific LUNS associated with them. Operationally, it will be important to maintain the relationship between the protection group and its associated LUNS. It is possible to use the Map function in VirtualCenter to observe this relationship and also to ensure that there are no cross-protection-group relationships.

For the BCDR environment outlined in this VMbook, the authored began by creating three logical containers in the Virtual Machines and Templates View, as illustrated in Figure 5.2.

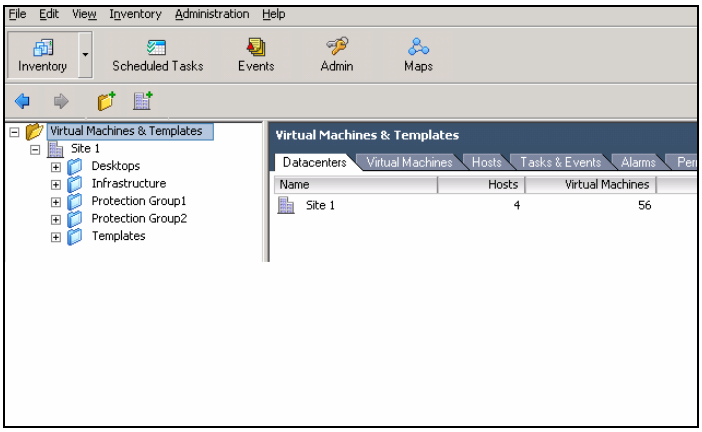


Figure 5.2 - Protection Groups in Site 1

In this case there is a folder for desktops and for local Infrastructure. Additionally, are two protection group folders: Protection Group 1 and Protection Group 2. In the event of a site or partial failover, only the virtual machines contained within these folders will be restarted; machines within Infrastructure or Desktops folders would potentially stay offline. In the configuration used for this VMbook, there are a number of local file and print servers and two active directory controllers. These are not replicated by the storage arrays so are not associated with protection groups; instead, their functionality is replicated by duplicating the services in Site 2. The services they provide must be duplicated or replicated in some other fashion to Site 2 as they will most likely provide services that the virtual machines in the protection groups rely on.

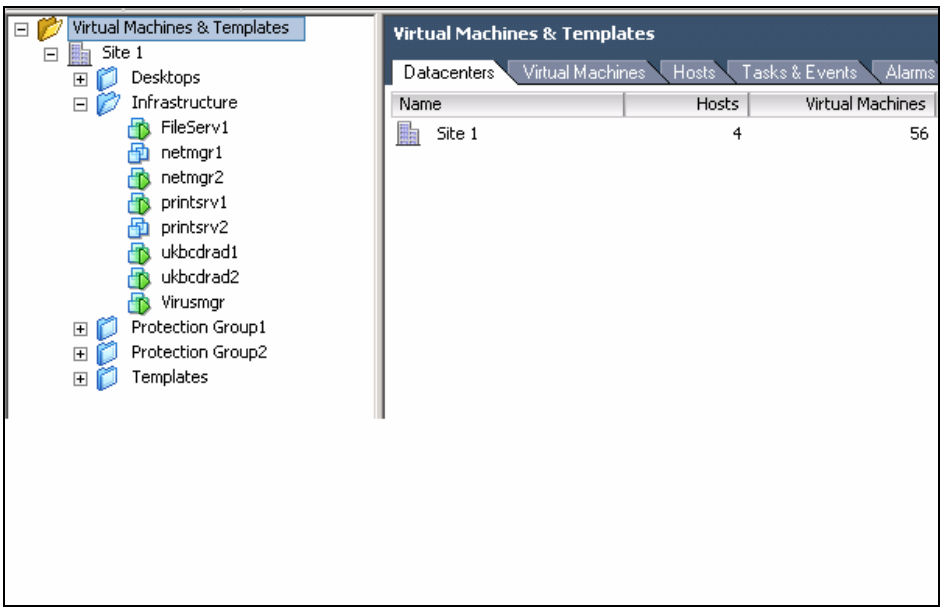


Figure 5.3 - Protection group screenshot 2

Within each protection groupthere is a sub-structure which allows for a variable business function to be represented. In this example, there is a simple delineation of Database and Application.

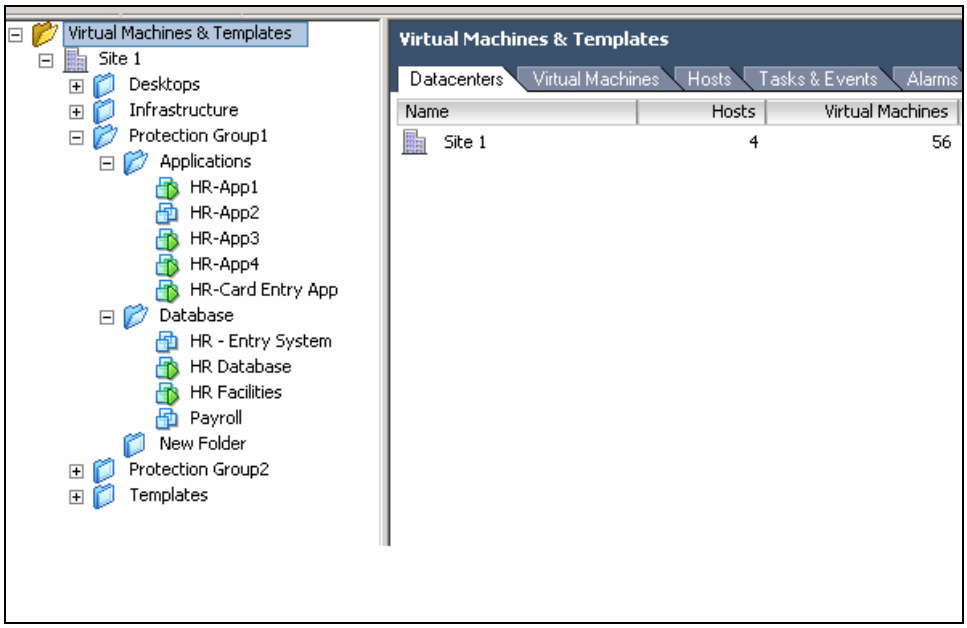


Figure 5.4 – Site 1 Protection Group 1

As shown in Figure 5.4, there are a number of applications associated with a Human Resources Function, and these are subdivided into Database and Application Categories. Figure 5.5 shows the applications from the Finance Department in Protection Group 2.

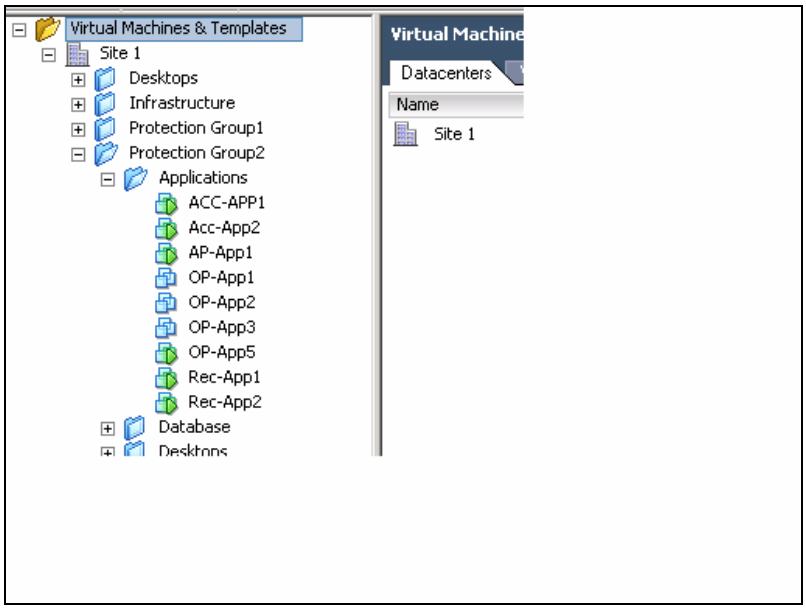


Figure 5.5 – Site 1 Protection Group 2

Hosts & Clusters

In concert with the folder structure, a secondary Host and Cluster view is implemented to control the resources allocated. The authors of this VMbook created a two-tier resource pool model. The upper tier is named Desktop, Infrastructure, Production, Test. In the second tier, under Production, Database and Application resource pools contain all of the machines to be protected (see Figure 5.6).

The Virtual Machine view is replicated in the Hosts and Clusters view; however, in the Hosts and Clusters view, this is a simple two-level structure. All servers that are located in a folder Applications associated with a protection groupare located under a single Applications Resource Pool in the Cluster view. In the Figure 5.6, the HR applications can be seen in the same group as the other financial applications.

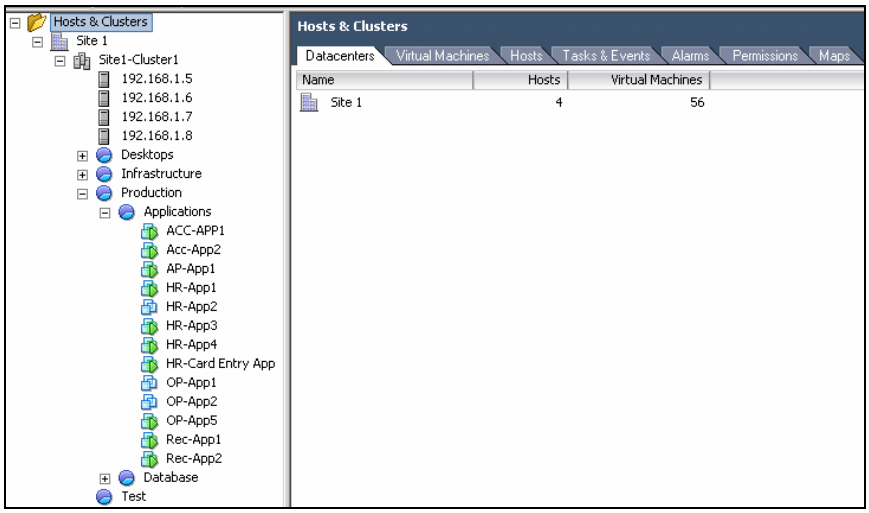


Figure 5.6 - Hosts and Clusters in Site 1

Resource pools to abstract the resources available in the sites. This has a number of benefits not least of which is provided by the VMware DRS product which can manage in a policy which will control the possible resource contention that may occur when a second site takes on some or the entire load from another. From a replication perspective, this will be relatively static but will need to be maintained over time.

Failover Structure

In the target VirtualCenter server (Site 2), provisions must be made for the incoming workloads to be accommodated within a pre-existing organizational structure. The resource pool construct is used to manage the resources to be applied to the appropriate functions, say mix between: development/application and database. So this is a fairly automatic and straight forward mapping.

Organizationally, folder structure is replicated for each protection group from Site 1 to a subfolder in Site 2. This means that in a failure all of the migrated services can be located in a single place and maintained according to a similar structure that is essentially a duplication of the source system.

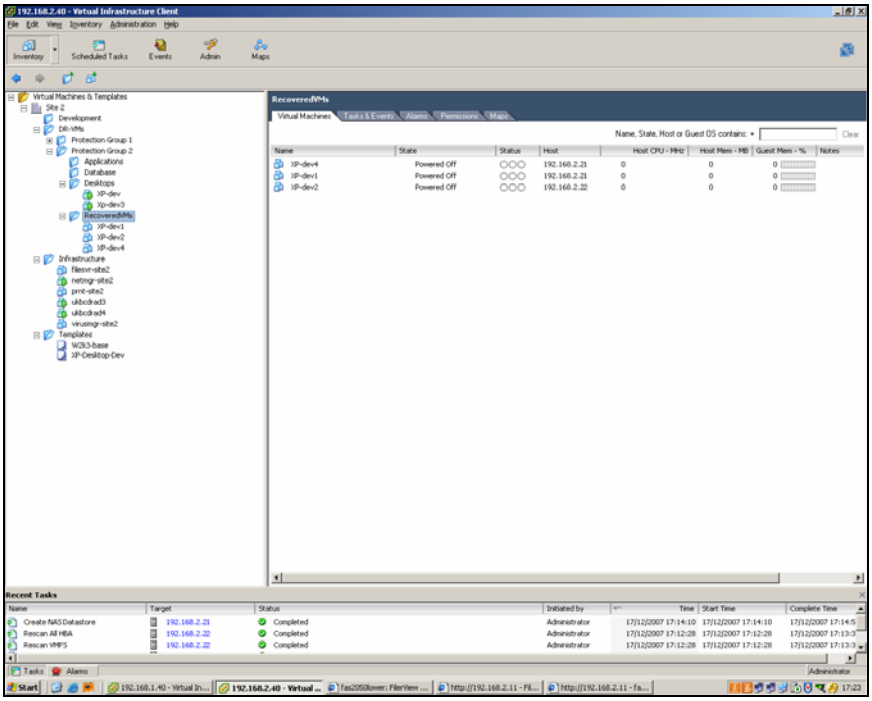


Figure 5.7 - Site 2 Inventory

As shown in Figure 5.7, there is a folder for the regular users of this site, the development team. There are additional folders at the high level for each of the protection groups. Under each protection group there are a number of other folders for Application, Database and so on.

In the process developed to register the virtual machines automatically, a "holding tank" is created for virtual machines as they are added. This is a folder called Recovered virtual machines. This allows the visual monitoring of the process in a single location. The administrator then can manually move the virtual machines to a logical structure at a later stage while the virtual machines are up and running; so with no impact to the RTO. The registration process can also be configured to target the virtual machines to specific folders for an additional level of automation.

The automated approach allows for a dynamic environment of virtual machines being placed into protection groups. During the registration phase for the virtual machines, this requires the administrator to check if they pre-exist in the structure, say from a previous test, un-register the previous entry and then re-register.

The Infrastructure virtual machines that will provide local services to any failed over services can also be seen here. In addition, it may be desirable to have services associated with a protection groupthat would not be present in a normal case. Here in Site 2 there are auxiliary desktop virtual machines

associated with the Protection Group 2. These are local services to Site 2 and are not replicated but are associated with the protection group.

Having established the target VirtualCenter organization, the incoming virtual machines must be registered into this structure. The following section discusses the issue of registration of the protected virtual machines into the target VirtualCenter infrastructure. This is done by showing initially the manual process and then moving on to show how it can be automated.

Before virtual machines that are present on replicated VMFS volumes can be accessed, VirtualCenter must register the entities into its schema. In the "source" VirtualCenter instance, this is taken care of automatically when a virtual machine is created in the first instance through the VirtualCenter interface, or when a virtual machine is imported through the VMware Converter process. On the target VirtualCenter, the virtual machines must be added to the schema and carried out via the Add to Inventory dialogue in VirtualCenter. It is a current requirement in the VirtualCenter architecture that to register a virtual machine the storage must be present. To achieve this, present a replica of the production LUN to ESX hosts in the target virtual infrastructure. This step is also the first thing to do once the replicated LUNS are visible in the target site if the intention is to restart the machines in the target site.

The registration process of virtual machines on previously unseen LUNs is relatively straightforward and once complete will be persistent in the target VirtualCenter database, even if the LUN is only presented transiently. If the LUN is subsequently taken away the entries for the virtual machines will change state to a "grayed out" version of their normal solid state. In the case where the design replicates the VirtualCenter database, this step is not needed.

Registration is a one-time process in most situations. If there are a small number of virtual machines, it is perfectly possible that an administrator could go through this process each time. The process itself can be accomplished only a few clicks and typically takes no more than 30 seconds maximum to achieve. If you have hundreds of virtual machines, you may want to consider automating this process.

To see how you might go about this automation lets first look at the manual process of presenting a copy of a replicated LUN (a third copy if you like) and adding the virtual machines it contains to the inventory. This is the equivalent of a DR test, in a real failover you do exactly the same for the replicated LUN and you would just miss out the step of creating a third copy.

If we consider the small protection group: Protection Group 1. It consists of two replicated VMFS storage areas: RP1 and RP2. You can use the MAP tab in VirtualCenter to graphically show the

relationships between the storage and the hosted virtual machines. In Figure 5.8 illustrates a number of virtual machines that, along with RP1 and RP2, form Protection Group 1.

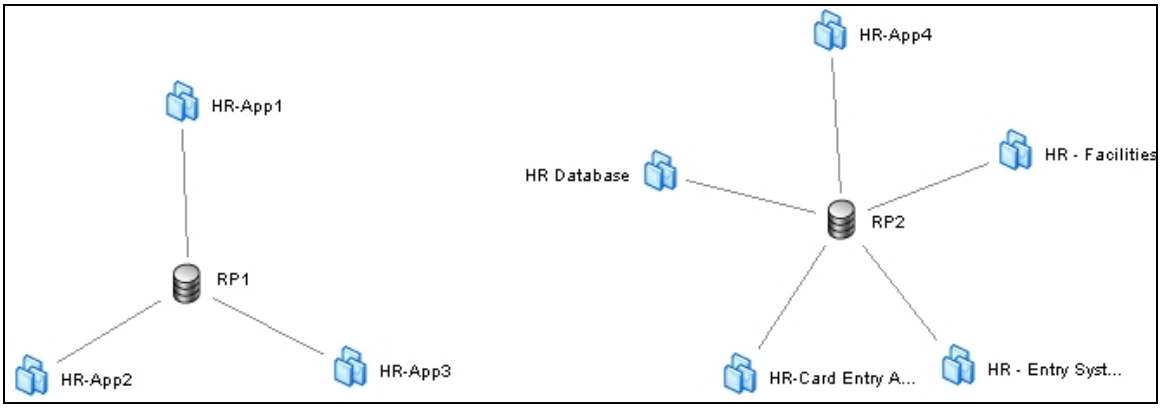


Figure 5.8 – Map View of Protection Group 1

As you can see, a number of virtual machines can be associated with the HR function within the simulated business. To see this view, select a VMware ESX host within the Inventory pane of the selected datacenter object, then select MAP on the righthand pane of the VirtualCenter interface. Next, select Storage as the view you want and use the filter options to remove the unwanted components.

The Protection Group

In the sample protection group, there are two database servers: HR Database, and HR – Entry System. There are also a number of application servers HR-App1 through 4 and HR – Entry System. While the names, and even the functions, of the virtual machines in this example are not that important, it will be fairly clear that for larger protection groups, a clear naming convention is desirable. This view is also useful to check that "stray" virtual machines are not on the replicated LUNS that shouldn't be. In this case, these services all sit on two VMFS volumes: RP1 and RP2 in Site 1. Because these are protected, the storage array has been configured to duplicate them to the storage array in Site 2; these are referred to as the "replicas."

The Third Copy

At Site 2, we have copies of these two LUNS. First, a clone copy of the replica LUN was made at the target site, Site 2. The process may vary from array vendor to vendor, but from an ESX perspective, the same effect is achieved. Now there are three copies of the LUN : one at Site 1 (the real one), a replica at Site 2 (read only) and a clone of the replica (writable point in time copy of the replica).

Most modern array types allow you to create a writeable "version" of the replica LUN in the target site. Different array technologies use different techniques and you can refer to the Storage section for more details on this, but they are often extremely efficient and initially often are close to zero bytes of additional storage to implement. We use this type of feature to initially "show" a copy of the replicated LUN as it would be in a real failover scenario. This is useful not only for the registration process, but also for test scenarios as we can test the failover scenario without removing the protection of the primary LUNS in the replication group. Some implementations also call for this type of function to keep a point in time copy of an environment. This can protect against wide-scale corruption, for example, which would be replicated by the underlying array.

Having made a clone of the replicated LUN, we will refer to this as the clone, typically it will be writable but offline. So initially the LUN presentation on the target infrastructure will still look something like the Figure 5.9. Again, it is easy to be expecting the LUN or associated VMFS to become visible, it is normal that after making the clones the LUNs will not be initially visible to the ESX Server.

Just considering RP1 initially in more detail, the following few screenshots illustrate what you might expect to see. At Site 1, RP1 is represented by a 150GB LUN, which happens to be on a SAN path vmhba1:0:0. At Site 2, we have a replica of this LUN which we have arranged to have the same path; it may not have this convenient match of course in a more complex environment. This LUN is continuously replicated from Site 1. Because of this replication the first copy at Site 2 is not writable by any server in Site 2 (while the replication is in place). So from an ESX perspective it is ignored. If you consider Figure 5.12, there is no VMFS associated with vmhba1:0:0, although the LUN can be clearly seen in Figure 5.9.

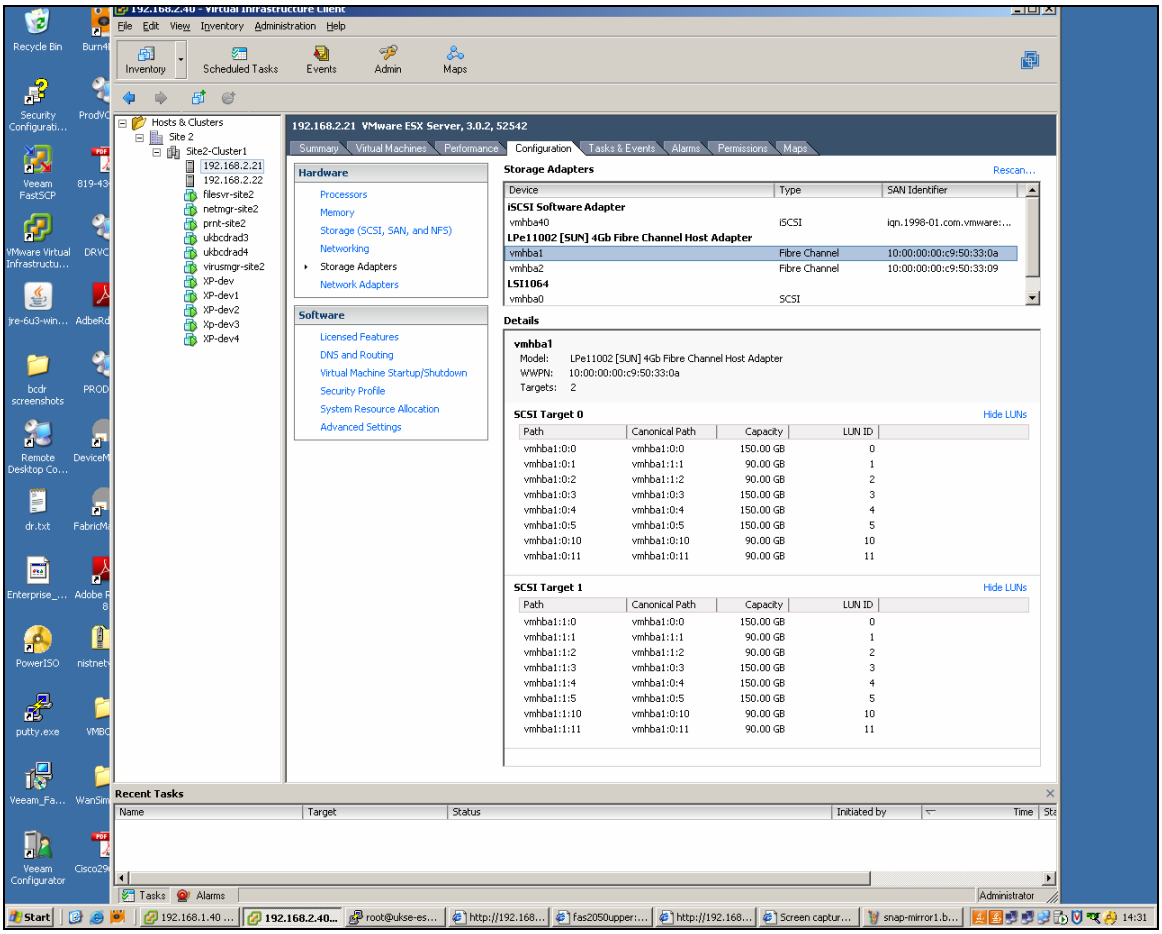


Figure 5.9 - Site 2 Configuration of LUNS Before Clone

Having created the clone of the replica, if we now use the array management to make the third copy visible and any appropriate san zoning, we can make the VMFS volumes present on these clones visible in Site 2.

Depending on the array type, various activities on the storage array may have to be completed. Having enabled the volumes, if we issue a rescan command from the VirtualCenter interface (in Site 2), we see the results in the Figure 5.10.

An additional LUN has been presented: vmhba1:0:6. This contains a writable point in time copy of vmhba1:0:0 (on Site 2). It will have the VMFS identity of vmhba1:0:0 (from Site 1), so the behavior of the ESX servers toward this LUN will depend upon how we have configured the snapshot mode controls, as clearly we already have a LUN identified by that path vmhba1:0:0. In this case, vmhba1:0:0 is in fact the running replica in read only mode on Site 2, but it could equally be a different volume/VMFS.

In our configuration, VMware ESX will just present the clone LUN as a valid VMFS volume, the details of how this are achieved are discussed in the storage chapters in part III of this book. The end result is shown in Figure 5.12.

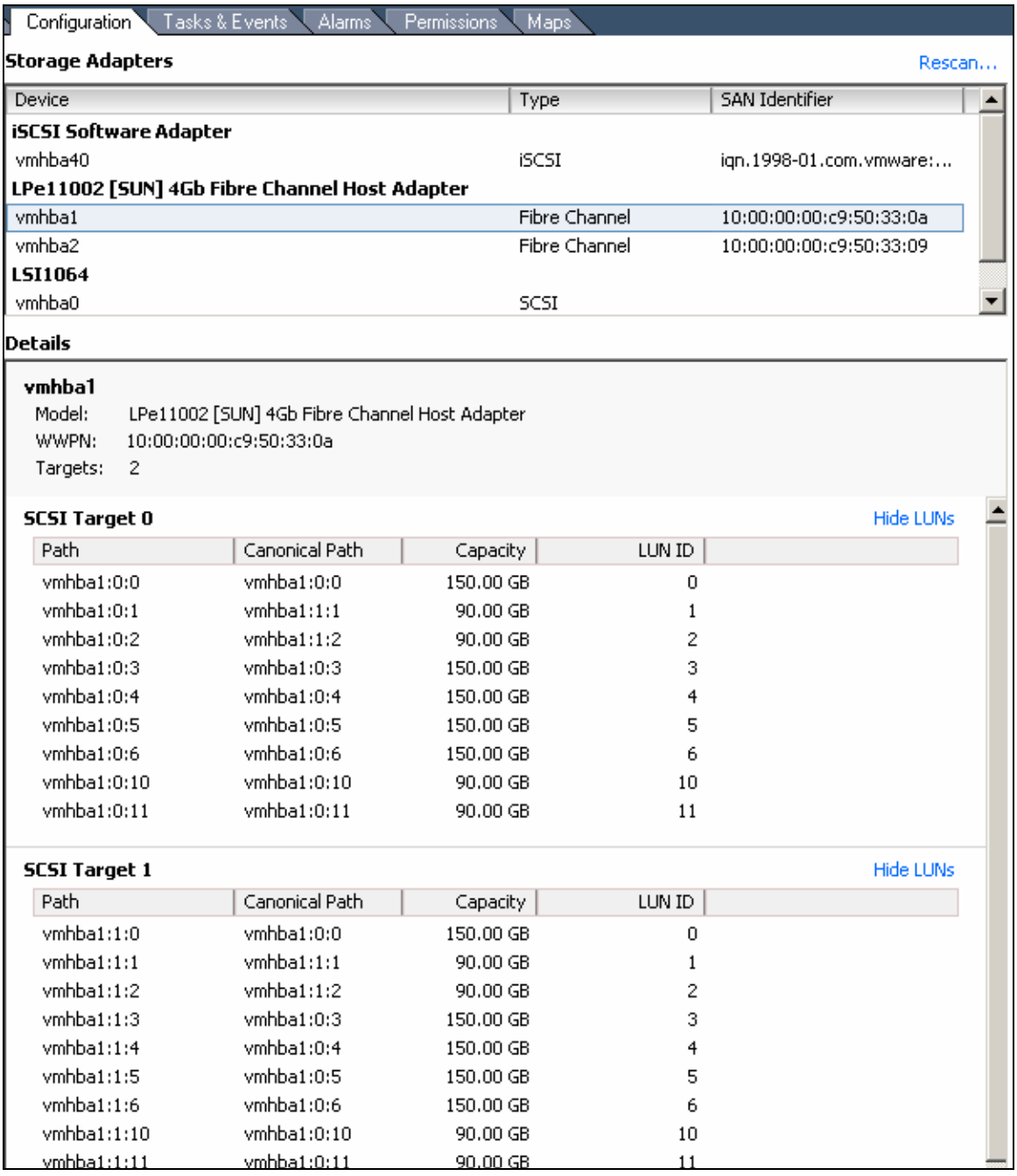


Figure 5.10 - Site 2 After Clone creation of RP1

If we now repeat this process for RP2 and rescan for new VMFS volumes, we will now observe two "new" VMFS volumes, and of course a second new LUN vmhba1:0:7 if we look in the storage controller view.

At Site 2, before we make the clones , the view is as shown in Figure 5.11. After the LUNs have been cloned and presented, we see Figure 5.12. So the end result from a VMFS perspective is straightforward, but it can be seen that a number of storage management functions have to be carried out.

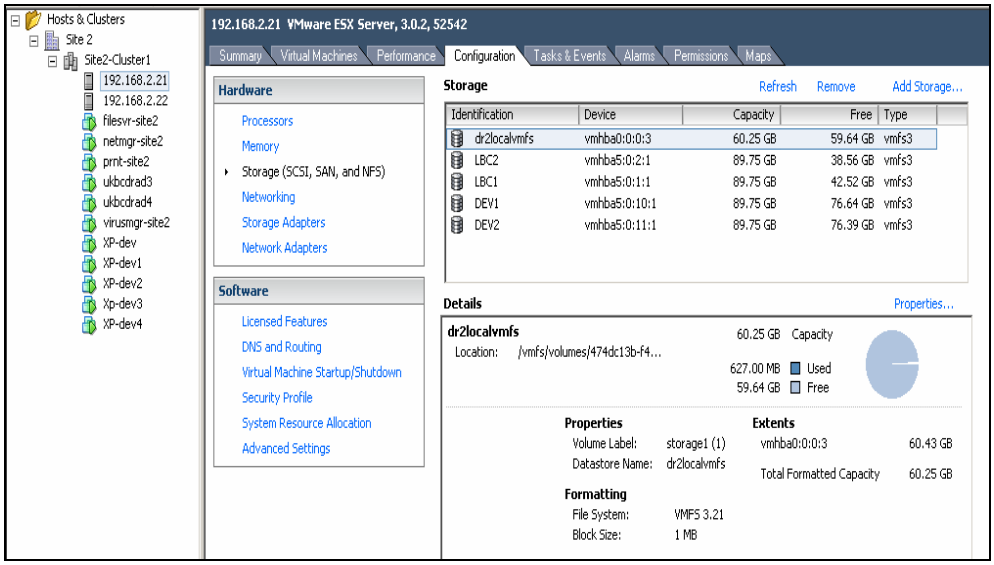


Figure 5.11 - Before VMFS Refresh

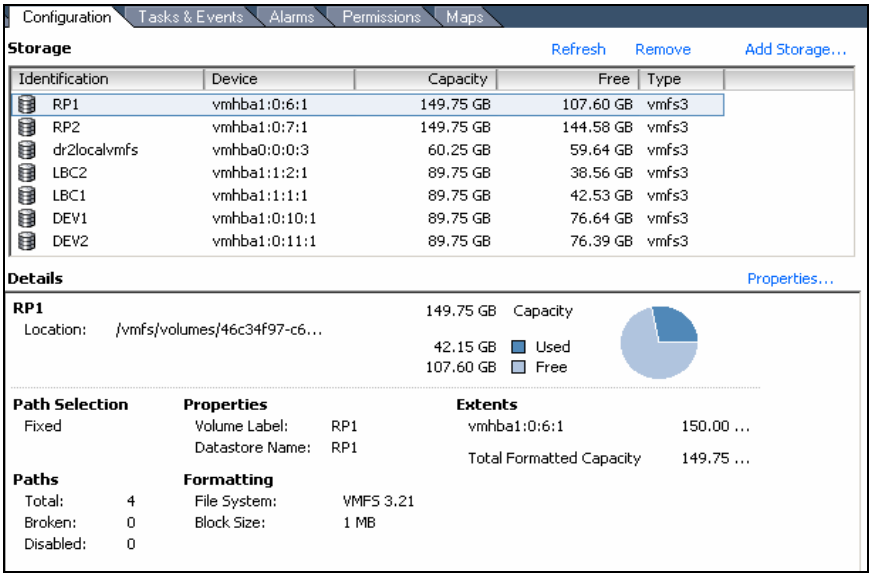


Figure 5.12 - After VMFS Refresh

Now we have presented copies of the VMFS volumes, we can now use the datastore browser function at Site 2 to inspect the volume. Double-clicking the VMFS volume will launch the browser and you see a panel as shown in Figure 5.13. For each VMFS volume, you will be able to see a directory structure for each virtual machine located on that LUN.

NOTE: if the virtual machine was created initially and later renamed in VirtualCenter the name of the folder will be the same as the initial VirtualCenter virtual machine name. If you are unsure it is possible to read the VMX file inside the directory and check for the registered VirtualCenter name.

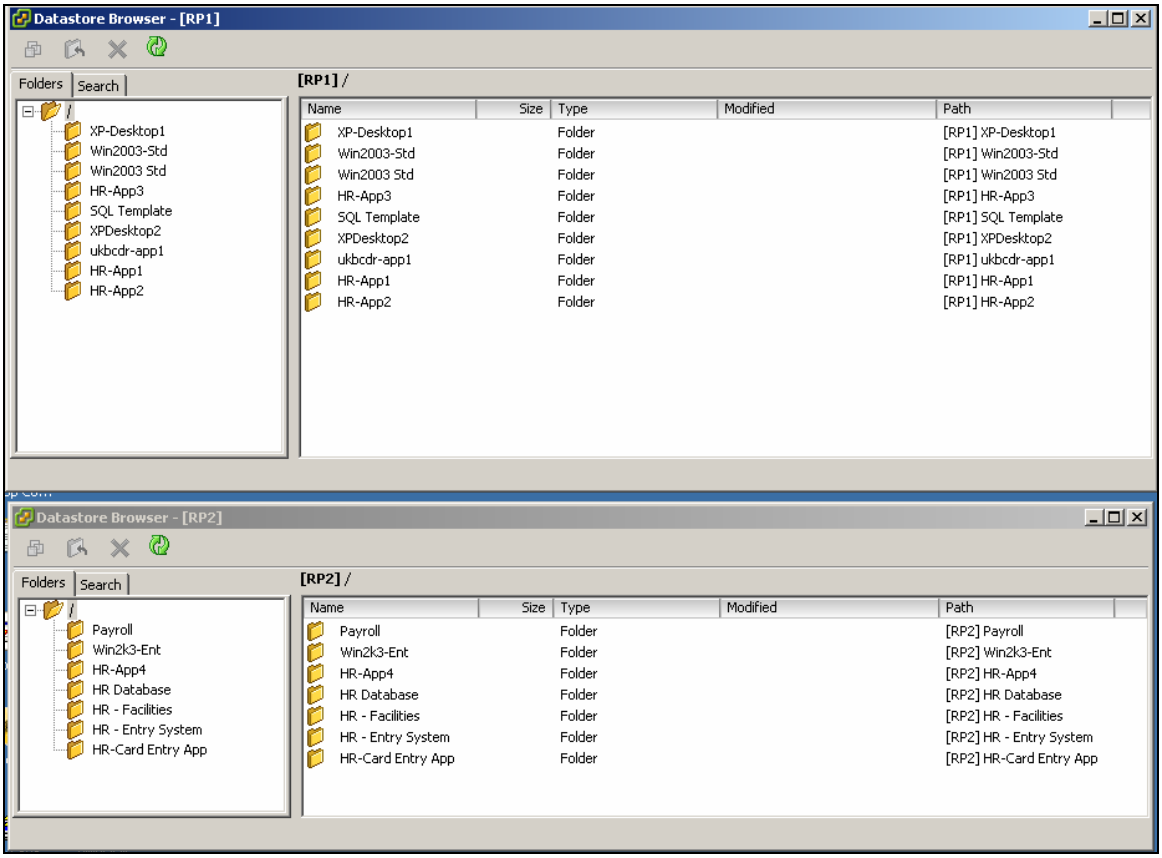


Figure 5.13 - Verifying VMFS Contents via the Datastore Browser

If we navigate to one of the folders again by just double-clicking the folder, we can go through the manual registration process. Figure 5.14 shows the registration of the HR Database virtual machine taking place. We double click the HR Database folder; at the top of the screenshot you can partially see the contents of the folder. Selecting the file HR Database.vmx, we can right click and select add to inventory following the dialogue that ensues.

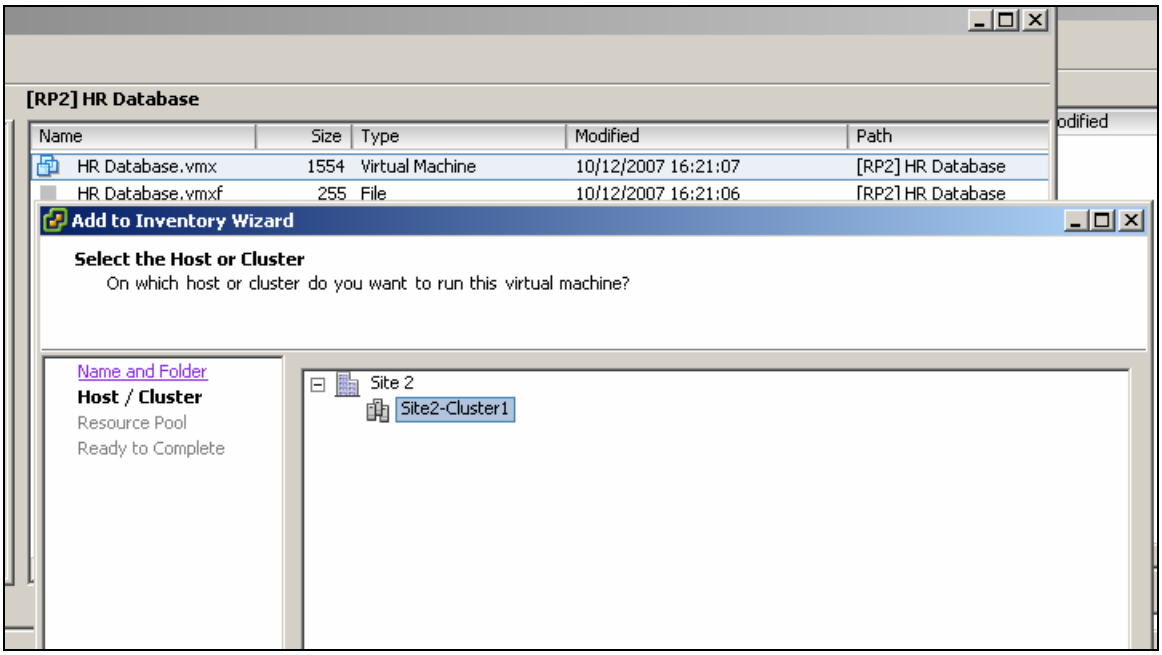


Figure 5.14: Adding Virtual Machines to the Site 2 Inventory.

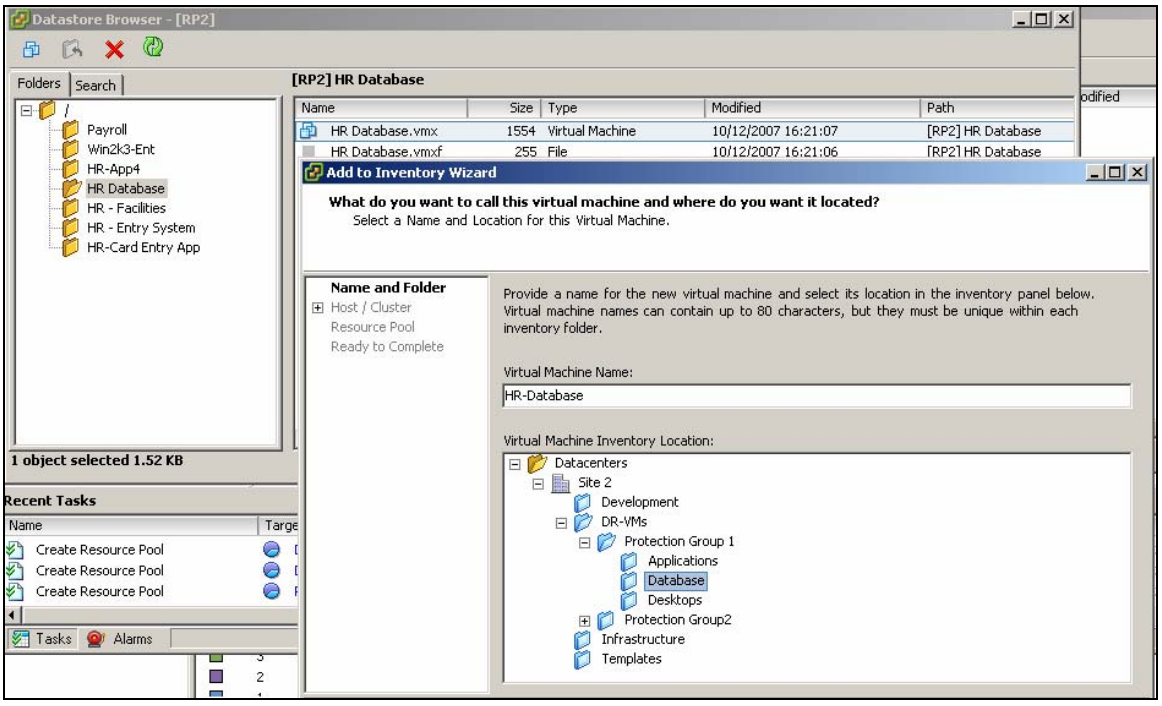


Figure 5.15 - Selecting the Virtual Machine Folder Destination

We can place the "new" virtual machine in the existing folder structure. In this case, we have built a container folder to hold any virtual machine that is executed in failover mode. We can create substructures beneath this if we need to.

Now allocate the virtual machine to a resource pool, as shown in Figure 5.16. This allows the capacity management of the new virtual machines entering the environment versus the normal workload to be managed effectively.

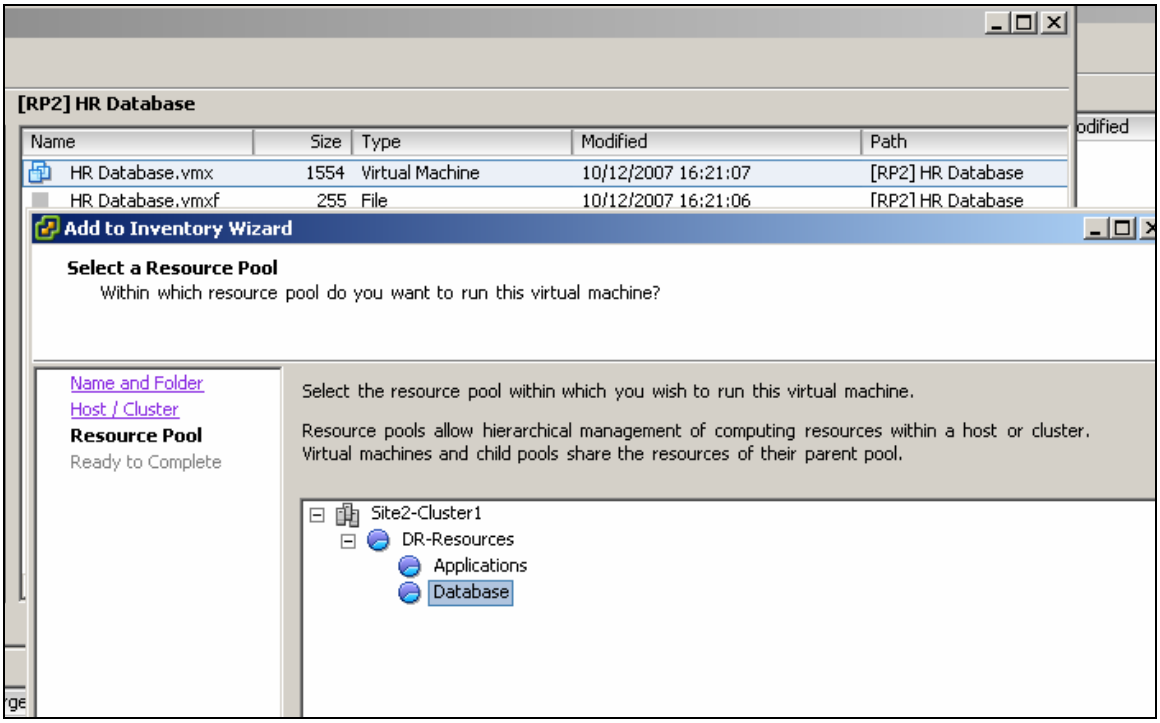


Figure 5.16 - Selecting a Resource Pool

Figure 5.17 shows the virtual machine registered at it new location in the target VirtualCenter database in a powered-off state.

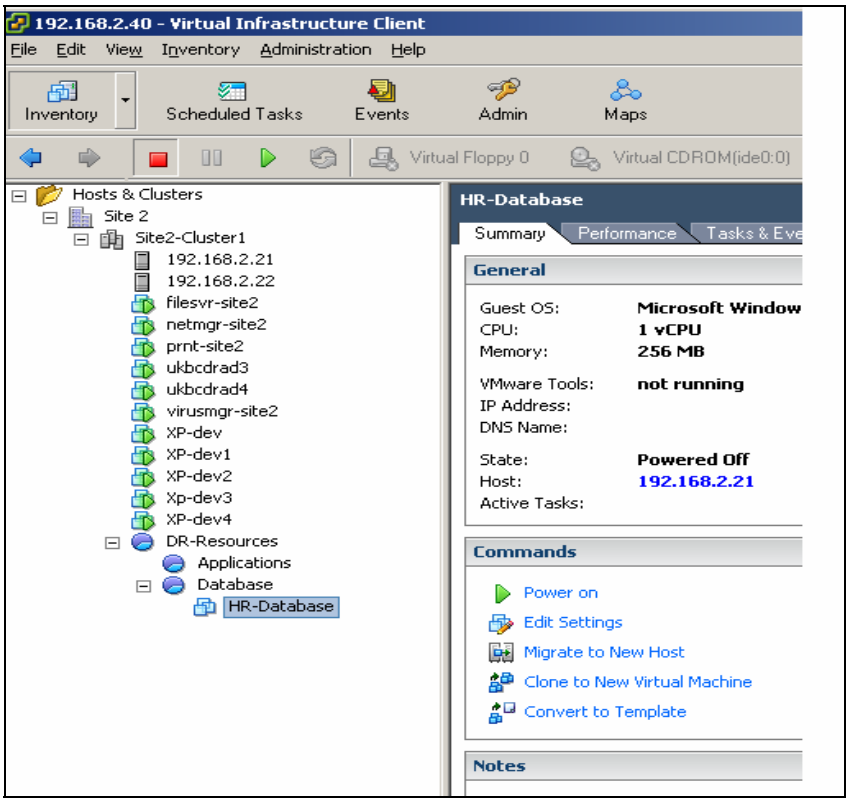


Figure 5.17 - VirtualCenter Inventory View

If we now attempt to power on the virtual machine, we will see the dialogue shown in Figure 5.18. This is because we have moved the relative location of the virtual machine. We are going to keep the UUID in this case.

NOTE: This would be certainly the case in a real DR and for test purposes is the most likely scenario. If you have used this technique to make a copy of a virtual machine that you want to have a distinct identity then you should choose to create a new UUID for the virtual machine.

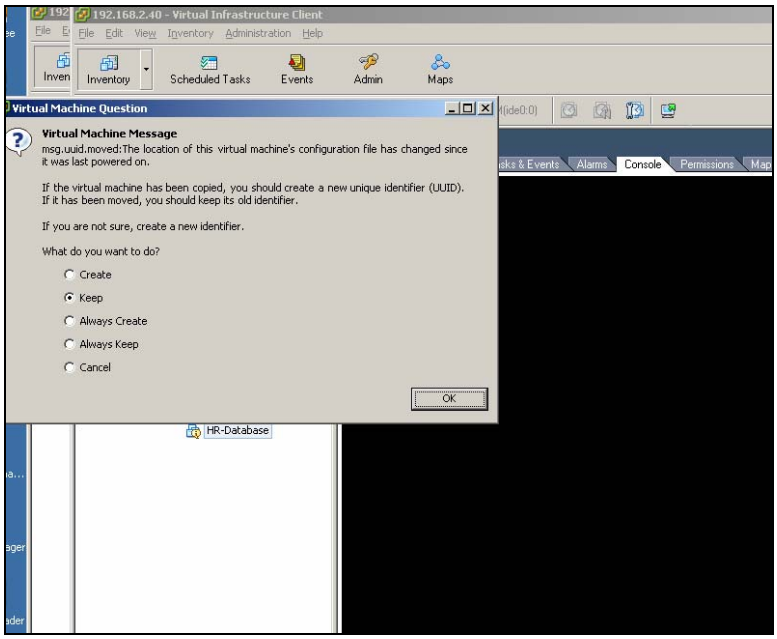


Figure 5.18 - Powering On a Virtual Machine

While the machine powers up successfully, it will not be able to see the network unless there is a replicated VLAN structure in place. If the machine has a hard coded IP address this will not have changed, as in this case.

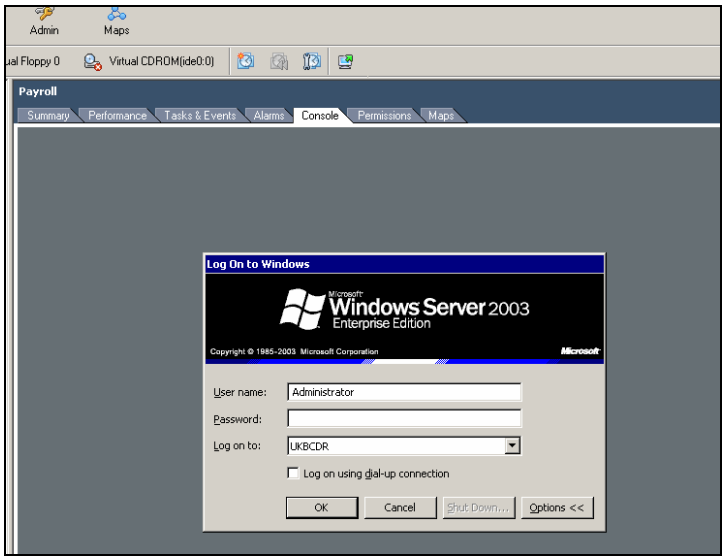


Figure 5.19 - Virtual Machine Console View

If the switch name that the virtual machine is connected to does not exist in the secondary site, then the virtual machine will not be able to attach the Virtual NIC to a switch, so the link state of the virtual

machine will be down. While this is an issue that has to be overcome to get operational in DR, it does provide a mechanism to safely bring up a virtual machine in a secondary site and test it potentially without it interfering with any production systems in the second site. So while it is possible to normalize the switch naming convention we have chosen not to do this to provide this level of isolation. This can be seen in Figure 5.20. The virtual machines configuration dialogue shows the virtual machine is not connected to the network, although connect at power on is checked. Using the dropdown menu, we can see there is a different switch available, but by default the virtual machine will try and use the configured switch from Site 1, i.e. Site 1-LAN, and hence the virtual machine is disconnected and will show a link status down inside the guest OS.

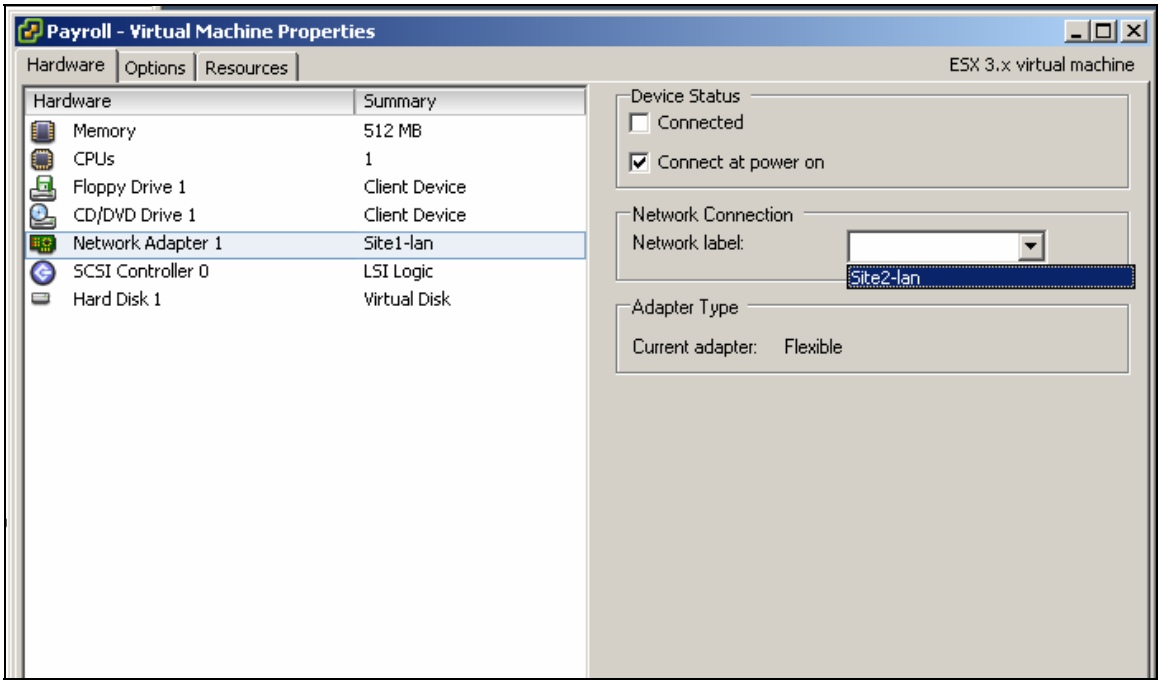


Figure 5.20 -Virtual Machine Network Connection

Maintenance considerations

This process is very straightforward and the persistence will ensure that even if we take away the temporary LUN the virtual machine entry will remain. Re-presenting the LUN will be sufficient to enable the virtual machine to be powered on. With a large number of virtual machines, it may be desirable to automate this process, but even if this is not required it will be required to maintain this structure over time to enable rapid failover. The provisioning process is often the best place to define this activity and in many cases fits in quite well. If a virtual machine requires a DR function typically it

will not get signed off through the provisioning process until DR is proven so it is straightforward to amend the process such that this step is included.

Automation

To automate this process for larger number of virtual machines we created a perl script that invokes the VirtualCenter SDK to implement all of the functions above on a group of virtual machines based on some input parameters.

The following diagrams show the execution of that script against the second protection group which has a larger number of virtual machines associated with it.
Using the MAP tab in VirtualCenter we can see the Protection Group 2 mappings of virtual machines to LUNS. The red circles highlight Protection Group 2 LUNS. RP5, which is based on an NFS file store, was introduced to show that the principles apply equally to the different storage classes supported in VMware Infrastructure.

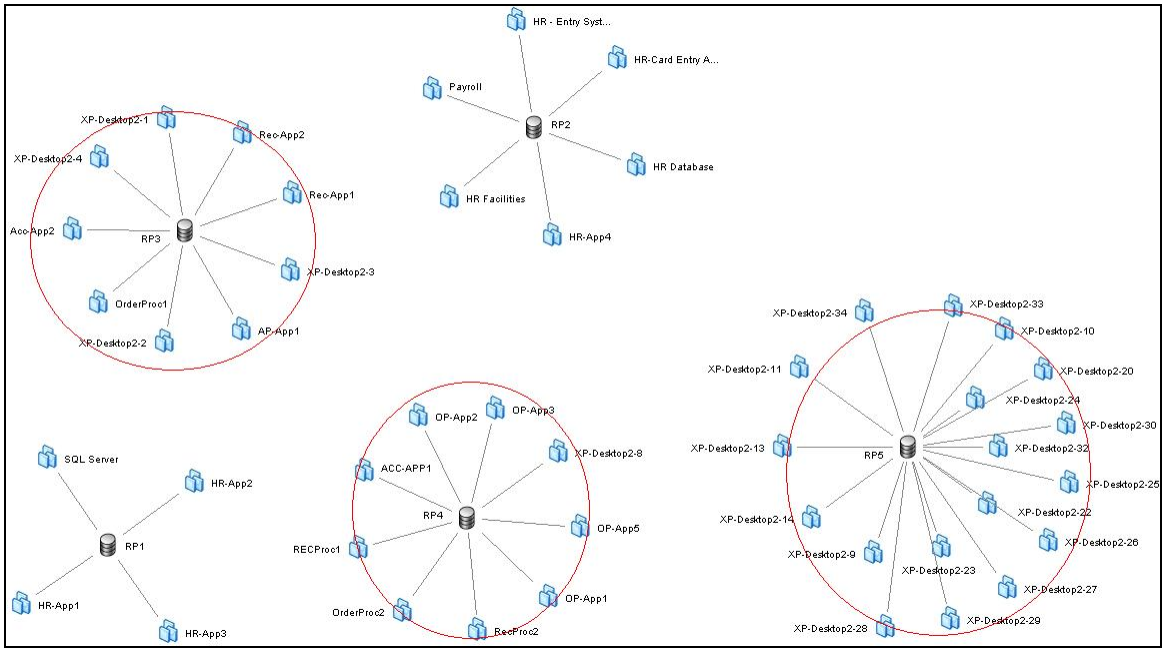


Figure 5.21: Protection Group Map View with Virtual Machines and Their Associated Datastores

In Site 1, VirtualCenter appears as shown in Figure 5.22:

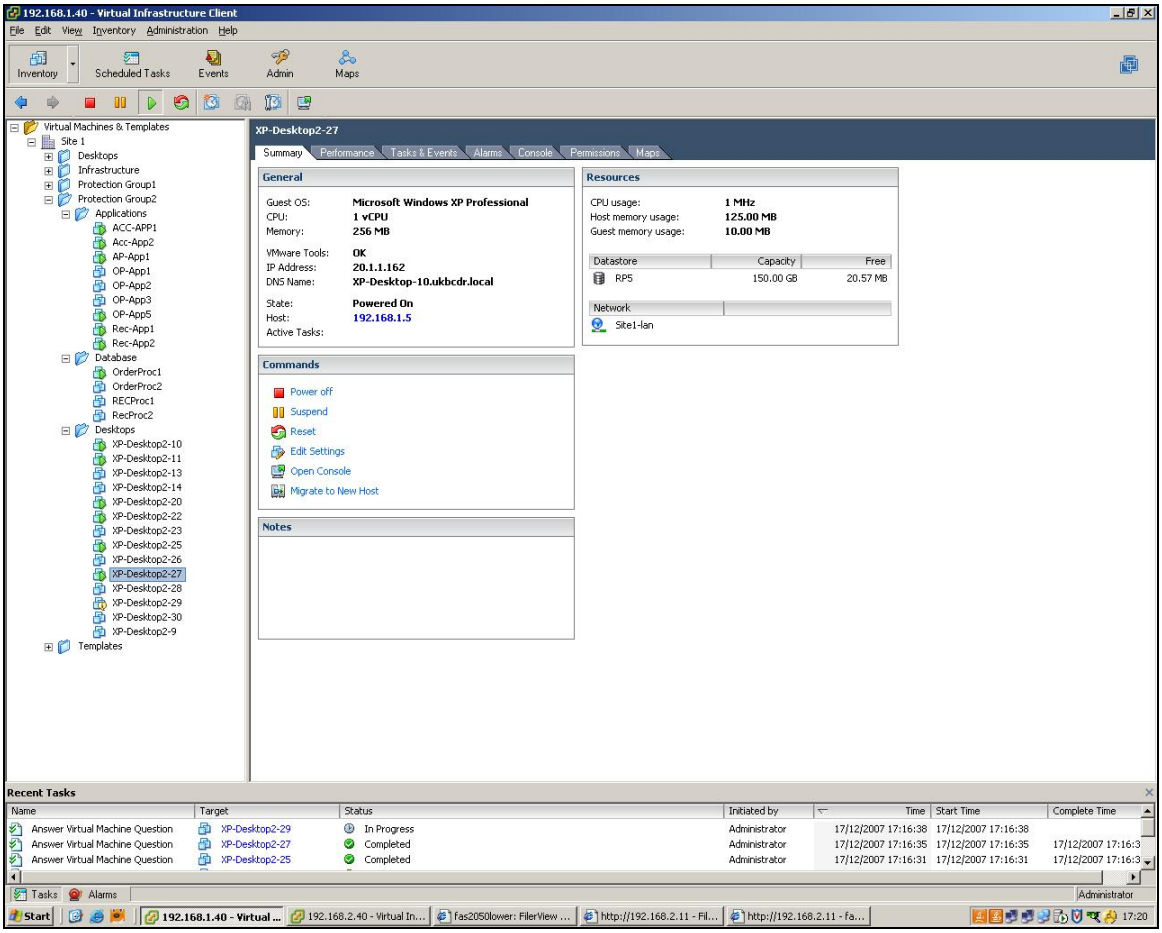


Figure 5.22: Site 1 inventory view

VirtualCenter in Site 2 appears as shown in Figure 5.23:

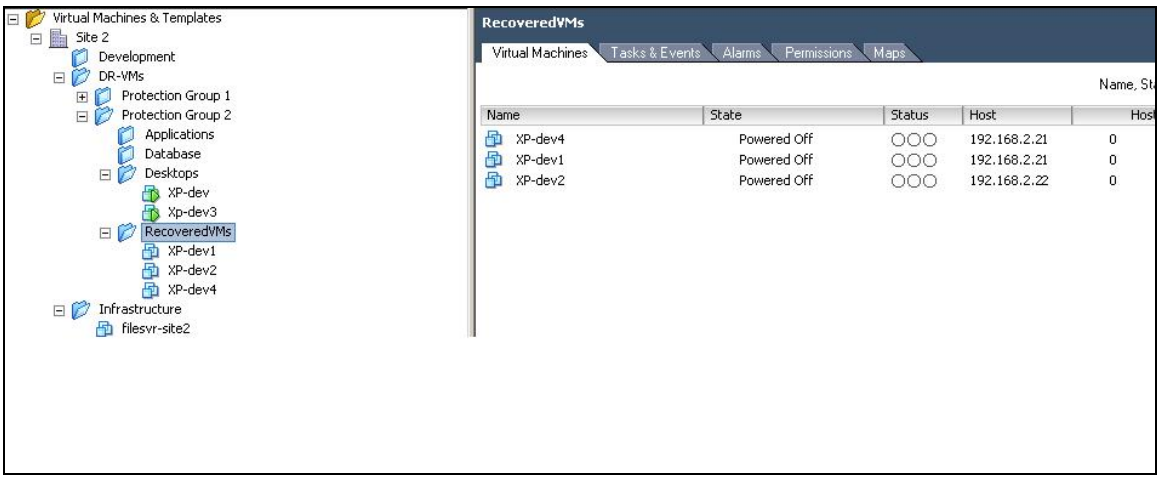


Figure 5.23 - Site 2 Inventory View

Figure 5.24 shows what the datastore browser from the Site 2 instance of VirtualCenter will look like if we present duplicates of the LUNS RP4, RP5 and RP6 to Site 2 in the data browser.

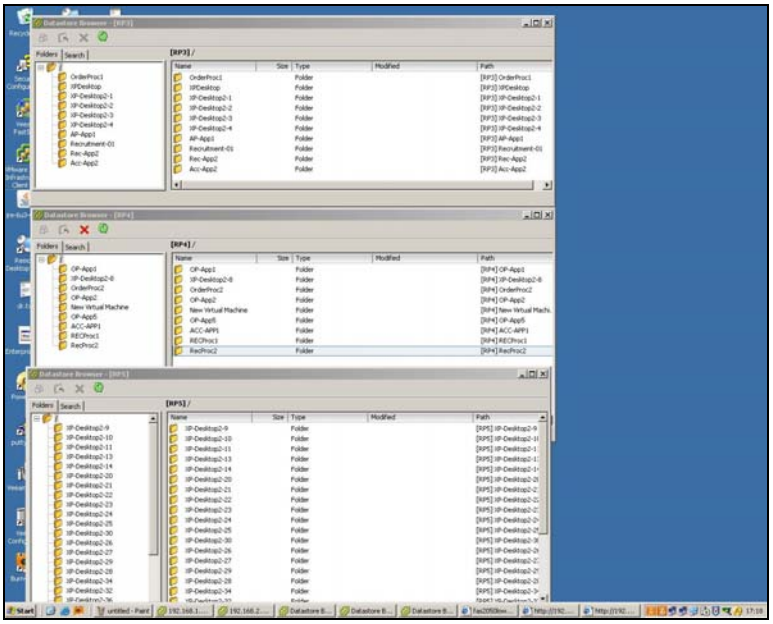


Figure 5.24 - Browsing Replicated Datastores

This is essentially the same as the manual steps shown above, except that we have a greater number of virtual machines in each of the datastores. Invoking the vm-recovery script (see Appendix A: BCDR Failover Script), we see the entries registered in the Site 2 VirtualCenter structure.

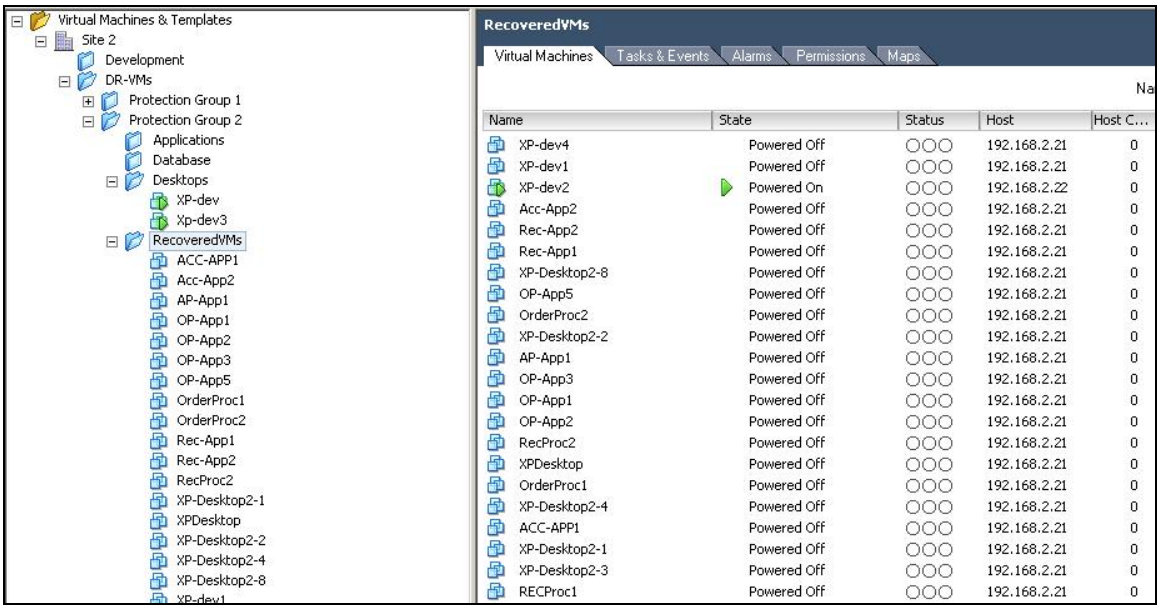


Figure 5.25 - Virtual Machine Registration Using Automated Recovery Script

After the registration process the script is capable of performing reconfiguration tasks and altering the power state of the virtual machine. So, for example, we could connect the virtual machine to a test switch or a production switch. Figure 5.26 shows the machines all being powered on.

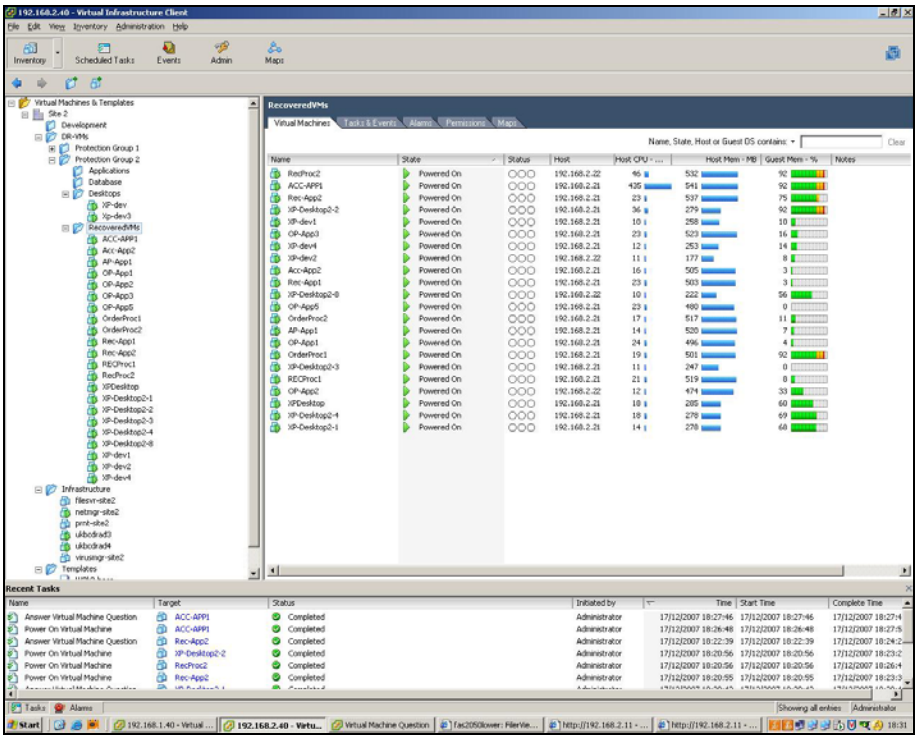


Figure 5.26 - Powering On Virtual Machines

In some cases, the reconfiguration steps can be performed by the virtual machine itself. Figure 5.27 shows the result of a virtual machine that executes a network reconfiguration when it detects that it is in the DR "mode." In this particular case, VMtools is used to execute a set of commands after a power on event.

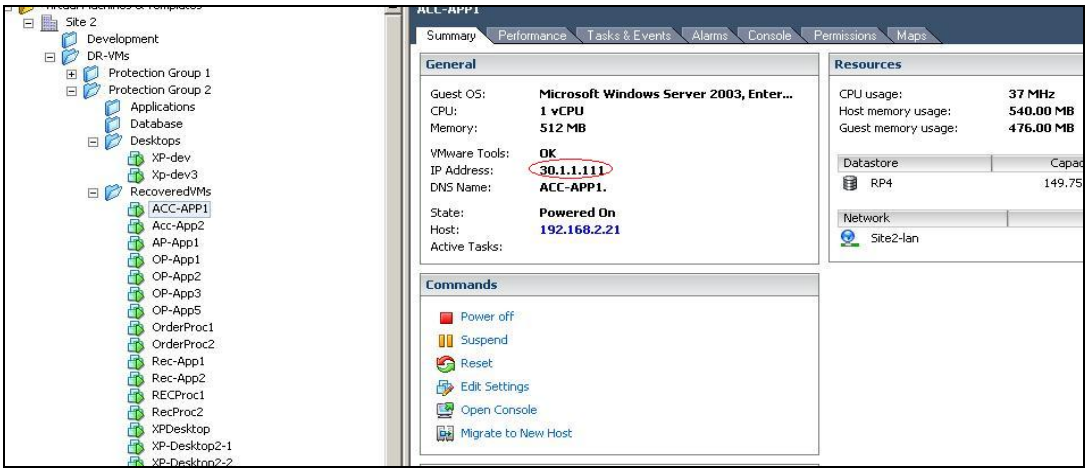


Figure 5.27 - Virtual Machine Network Reconfiguration

Chapter 6. Advanced & Alternative Solutions

Many VMware customers have used designs such as the one explored in this VMbook to implement efficient, predictable and rapid failover protection of their virtual infrastructure, in many cases providing protection for many hundreds of workloads within minutes of potential disaster situations. This section provides a brief look at a few of these advanced and alternative solutions.

BCDR for All

In some respects, the "active/active" model, while attractive and highly functional, may not be appropriate or cost-effective for smaller organizations. This is not to suggest that comprehensive BCDR plan is not critical for smaller business; in many cases, it can be argued that smaller companies face greater risks as they have less business resilience purely because of scale. However, some of the techniques described in this chapter can be adapted to suit smaller single-site situations and budgets.

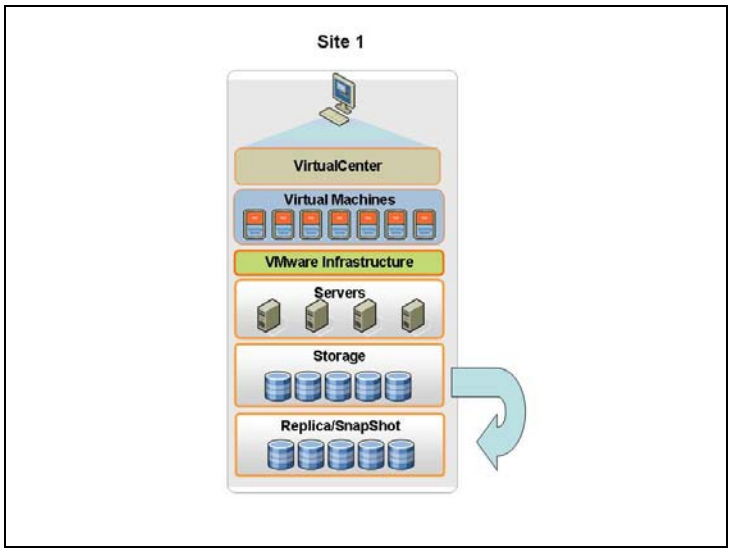


Figure 6.1 - Single-Site Replication

The organization of the BCDR plan can be directly used to test what might occur and what could be recovered in a timely fashion. Going through this process alone may illuminate weaknesses or unforeseen issues with existing plans. VMware Infrastructure provides the perfect environment in which a solid plan can be tested and thoroughly debugged using either local array-based replication or traditional backup-and-restore techniques. This can be achieved potentially with no net hardware increase into the estate and can raise confidence in BCDR planning.

In combination with modern array-based snapshots, (or perhaps in combination with tape or alternative offsite storage), additional levels of resilience can be quickly achieved without significant capital expenditures. VMware virtualization technology makes it much simpler to recover these images in a timely and reliable way.

Work Area Recovery

This VMbook demonstrates BCDR for both server and desktop workloads in the VMware Infrastructure Inventory, making very little distinction between the two workload types. One of the advantages of virtualization, especially, when you host virtual desktops in a central datacenter, is that there is very little difference from a BCDR perspective. In fact, desktops can even be DHCP-enabled, reducing networking complexities.

The authors of this VMbook have encountered an increasing trend to use virtual machines to provide desktop services in the event of building loss or access problems due to threats to infrastructure or, even more recently, pandemic planning. Here the desktops may be traditionally hosted on PCs in offices. Providing desktop services for situations where these are not accessible is generally becoming known as "Work Area Recovery."

Most implementations of this service are dynamic in nature and utilize virtual machine technology to replace the old scheme of a rented or shared facility populated with PCs that could be built up or had to be maintained in synch with the "corporate build." However, you will still require the shared desktop facility to provide desk space or you could consider home working.

Virtualization assists with the rapid deployment of up-to-date images. For example, you can maintain an image bank of 20-30 images that contain applications appropriate for a number of work scenarios. VMware Infrastructure can be used to automate the deployment of many images for each user coming into the work area recovery solution. These only need to be deployed at invocation, and only the base image needs to be maintained at the appropriate level.

Upon invocation, a set of desktop images is created on demand for a targeted set of users. The main challenge of this approach is to provide sufficient images in an appropriate time for the solution to be viable. 50 or 60 desktops can be provisioned very quickly, but traditionally this has involved a file copy of the entire virtual machine, so certainly could be time-consuming for many hundreds and may be un acceptable in the many thousands arena.

A number of new technologies can be used here to enhance this type of solution. One rapidly evolving technique is to use storage snapshots to duplicate LUNS holding approximately 10 virtual

machines. The initial deployment may be for 10 virtual machine of type A. Then use the storage array to make copy on write versions of this base LUN. The latest disk technologies from most storage vendors can make these duplicates extremely rapidly as they are generally pointer copy type activities. The virtual machines then need to be registered, possible as our solution described here, and then powered on and probably with Microsoft based operating systems a sysprep process run. It is conceivable in this manner that many thousands of virtual machines could be created in a few hours.

A second area of advance is another virtualization technology: application virtualization, At a high level, application virtualization can be thought of as providing "containers" for the individual applications loaded onto a single operating system instance. For most desktops, there will be many applications. These all have the potential to interact with each other in unforeseen ways. Application virtualization simplifies this interaction greatly and also allows the applications to be loaded on demand by the user. Application virtualization is beyond the scope of this VMbook, but a number of different solutions exist for this approach and indeed VMware has technologies in this space. The potential enhancement to the above is that you no longer need a large bank of images, potentially reducing to one but realistically two or three for various generations of operating system at least. However, the user gets a very generic operating system and then applications are installed to that instance as demanded.

Lastly, a number of emerging technologies would allow for the operating system itself to be streamed to the virtual machine, and possibly not require even a disk image to be created at all. Applications could demand functions of the OS that they need and those alone. With memory densities approaching 250GB, this approach looks increasingly feasible and possibly has implications outside hosted desktops.

Clearly, whatever approach is adopted does require spare, reserved or scavenged resources from existing VMware Infrastructure.

Physical-to-Virtual BCDR

In this approach, physical servers are protected by failover to redundant virtual machines. Typically in this scenario, a physical server that is to be protected is taken out of service, a conversion process is run to build a virtual image of the server, possibly with VMware Converter. Once this is successfully established in a second site, the original server is brought back into service. On a defined schedule the virtual machine is maintained by updates from the live server. There are a number of established third-party software vendors that provide this type of functionality, many of which are VMware technology partners. To find specific vendors, visit the VMware TAP catalog[4] and search for "ISV system software".

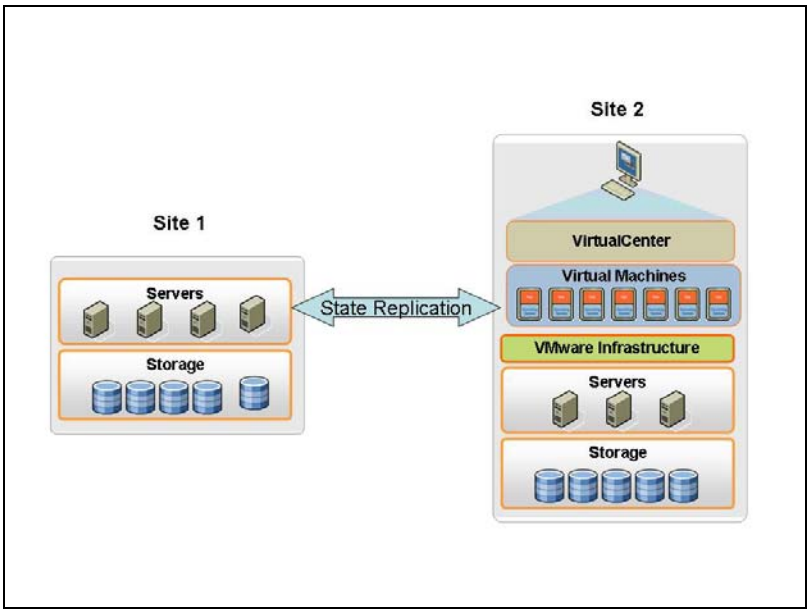


Figure 6.2 - State Replication with a Single VirtualCenter Instance

There are a number of ways to achieve this step. Simple backup with a restore to the virtual machine; has the benefit that you are continually testing your restore process. Continuous replication – a number of commercial products are available that allow an agent process to duplicate changes from one machine to another on a continuous basis.

[4] http://www.vmware.com/tapcatalog/public/listing.php

Active/Active: A Single-Site View

The solution described here does provide for an active/active situation which is beneficial from a resource and utilization perspective. We have assumed a significant distance between both sites and a significant workload at both sites. This has driven a twin Virtual Center design. It may be considered desirable to have a single instance of Virtual Center.

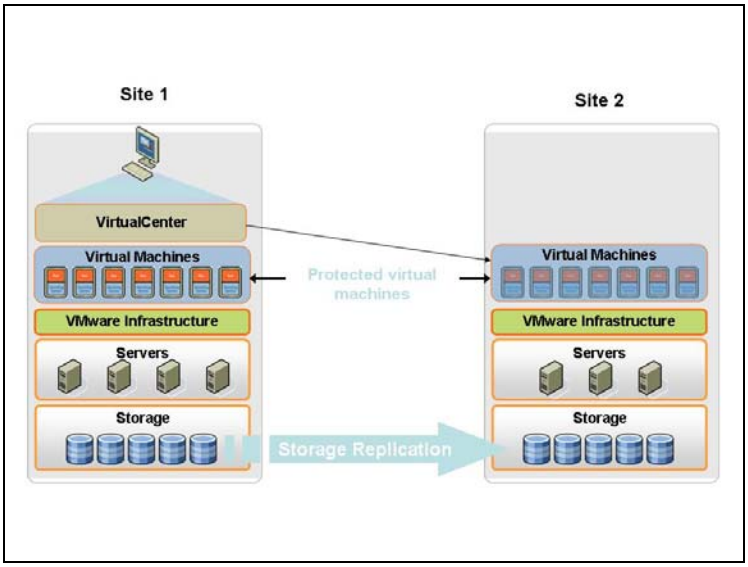


Figure 6.3 - Example of Multisite Replication

The design above can be easily modified to accommodate this approach but in our view would require both sites to be 'relatively' close in millisecond terms for VirtualCenter to remain effective. The only major modification that would be required would be some additional resilience design work to be done for the VirtualCenter instance itself. VirtualCenter can be implemented in a cluster, put into a separate VMware HA cluster or alternatively some sort of rapid recovery system would have to be put in place.

Active – Active – Active

For global organizations, there is some attraction for running a three-way solution. The logic is: one center in each major territory running up to 60 percent capacity. In the event of any one center being unavailable the other two-thirds can be bought to bear. Traditionally, most array replication technologies have been peer-to-peer, so this becomes quite complex in this environment. It is possible that for limited scope a three-way mirror could be established, but to make it general-purpose solution is very challenging.

Moving the replication function to the SAN layer might be a solution in the near term. In this case, the SAN itself has intelligence about traffic flows and write splitting and LUN duplication can be done at the fabric level rather instead of at the array level, enabling a general-purpose hub-and-spoke model for storage. As the server emits the I/O, the fabric detects it as an update and sends two copies of the data to Site 2 and Site 3. With these types of technologies, a synchronous and asynchronous write might also be considered. The first hop to site 2 could be within the metropolitan area and could easily be synchronous and the second could be asynchronous over a much further distance. These technologies are beginning to mature and a good example of this is the EMC RecoverPoint™ technology.

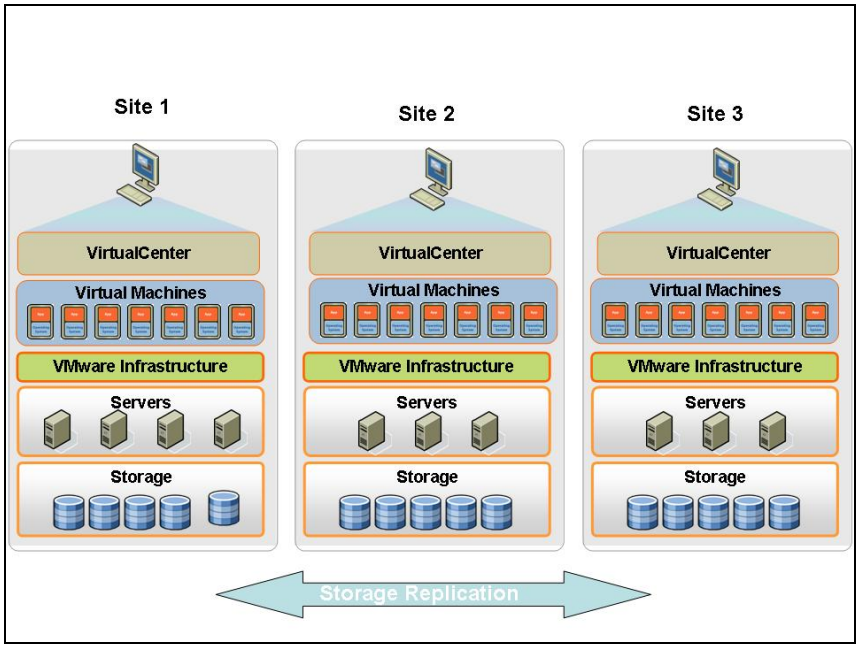


Figure 6.4 – Storage Replication Across Multiple Sites

Operational Failover

One trend we have seen consistently is to move BCDR generally to be a "business-as-usual" function. This is sometimes referred to as "operational failover" and simply means that, on a regular basis, a group of services will be failed over to a second site and operated as normal at that site. There seem to be two main drivers for this approach. Firstly, by building the failover scenarios into the day-to-day operations of the IT infrastructure, it is much more likely to be a success when a real disaster strikes. As organizations become more aware and regulated in this area, this is becoming an increasing priority.

The second opportunity that is potentially exploitable is in the more strategic and longer term acquisition and management of the datacenter infrastructure itself. The building real estate and fit out. These are large capital investments usually and recent trends have to be to consolidate into fewer larger centers. The life of these centers is limited to some extent and some provision has to be made for upgrades. One example might be to only deploy to 4/5^{ths} of the available space allowing some swing space to allow refits. The other aspect of this consolidation is that fewer centers carry a higher risk if a total loss should occur. This means that a more money has to be spent on these larger centers to offset these risks, multiple power feeds etc. A more agile datacenter would allow different approaches to be considered for these challenging problems. Virtual Infrastructure provides the first step, unlinking the direct relationship between the physical assets. With the addition of an operational failover capability the notion of building a new center or shuffling space within an existing center becomes much less risky. The ability to rapidly take advantage of new facilities could have large capital expenditure implications. WAN-based VMotion, as one could think of it, could be a radical shift in the way we think about procuring, operating and decommissioning datacenter facilities.

PART III.
BCDR Operations

Chapter 7. Service Failover / Failback Planning

This chapter provides guidance on the steps needed to complete a service failover and failback and follows the basic principles set forth in Chapter 5.

Planning for Service Failover

At the simplest level, virtual machines run guest operating systems which in turn run applications, which in turn provide an IT service to the business. It is this service that must be protected from a BCDR point of view.

Most customers moving their datacenters to a virtual architecture for servers, network, storage and recovery usually start with one side of the picture. In BCDR terms, this is usually the referred to as production, source side, protected site, primary datacenter or in our case Site 1.

The example architecture constructed in this document follows the most common approach most customers take. We have an existing environment, Site 1, that represents our production IT service and we now wish to protect this through replication to an alternate site, Site 2.

It must be noted that this left to right protection (Site 1 > Site 2) can of course be combined with right to left protection which is a true active / active datacenter model. In that model services can be moved from Site 1 to Site 2 or Site 2 to Site 1. In either case the concepts are the same.

The Figure 7.1 illustrates how the inbuilt properties of virtual machines combined with a replicated architecture allow you to map resources from one virtual environment to another and at the same time enable failover of an IT service.

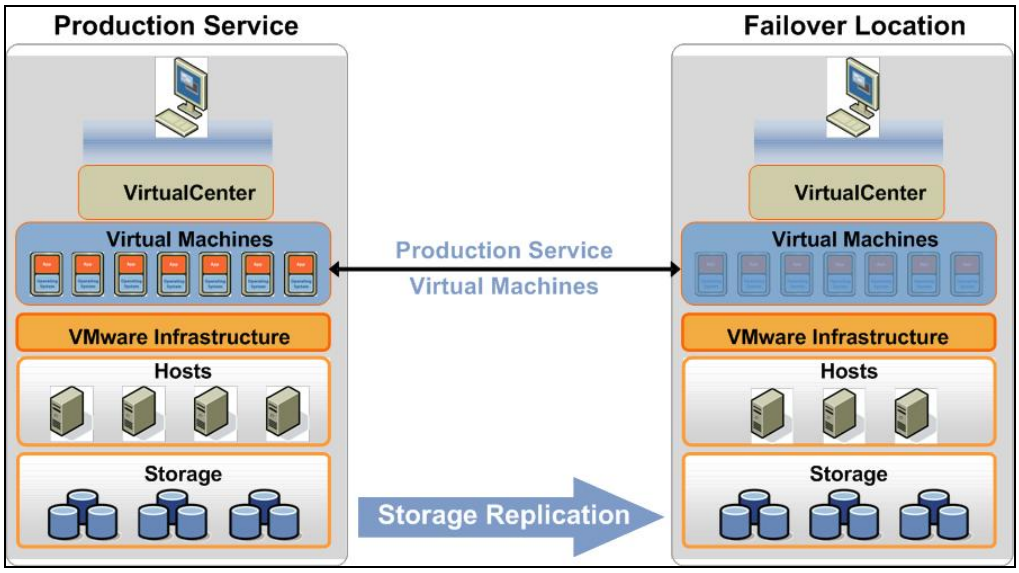


Figure 7.1 – BCDR Virtual Infrastructure Overview

The VMware Infrastructure layer presents a common architecture on which the virtual machines can run. Virtual machines are encapsulated on the storage, are isolated from one another and not dependent on the same hardware being available at the failover location.

Failover Architecture Components

Before you can progress to failover testing or actual site failover there are a number of components that need to be built to ensure your virtual infrastructure is ready to support this kind of workflow.

As shown previously in Figure 7.1, storage replication forms the foundation of the recovery process. Storage is replicated at the failover site, then presented to the virtual infrastructure architecture at the failover location, at which point the virtual machines can then be activated. The key question at this point is: how do you perform this activation reliably? The kinds of questions that start to appear in typical planning stages are:

- How do you get the virtual machines into the configuration at the failover site?
- How do you make the virtual machine icons appear in VirtualCenter?
- Does all this happen on its own? Can you script it?
- Do you have to do this per virtual machine?
- What else do you need to change? Storage configuration? Networking configuration?

- Will these virtual machines interfere with their source/live counterparts if you are running a failover test?

Before failover, the following tasks will need to be reviewed and completed:

- Virtual Infrastructure Configuration (Primary and Secondary sites)
- Storage Configuration (LUN presentation / virtual machine layout)
- Storage Replication (replication configuration)
- Current Network Failover Strategy (IP addressing / VLANs)
- Network Configuration (available subnets at failover site / VLANs)

Each task may involve different steps depending on your environment.

Virtual Infrastructure Dual-Site Configuration

One of the key aspects to review when looking to move your virtual infrastructure to a BCDR model is continuity. To improve continuity, it is critical to ensure your configurations are standardized across both Sites where at all possible. Having standard configurations at both the ESX level and VirtualCenter level will improve the usability and understanding of the solution implemented.

Configuration changes can be as simple as ensuring you use meaningful and logical naming conventions for resources such as datastore names, portgroup names and virtual machine names.

The authors of this VMbook used two datacenters, Site 1 and Site 2. All datastores were prefixed s1- or s2- depending on their point of origin, portgroup names also contained reference to their local site. If your networks (vlans) spanned sites then a common name could be used, say VLAN160 could exist as a portgroup name at both sites if that corresponded to the same physical network resource at both sites.

Other considerations at this stage include things like cluster objects within VirtualCenter. For example, at Site 1 you can have a resource pool called "FinanceServers," and in that pool are all the Finance teams production Windows 2003 virtual machines. These virtual machines are also in a folder called "FinanceServers".

At the failover site, it would make sense to create similar containers for these virtual machines to run in during failover; however you may wish to prefix these with something to indicate they contain a failed over service. At Site 2, the structure may be "Site1-FinanceServers" for the resource pool name and

"Site1-Finance Servers" for the folder name. The resulting inventory map from Site 1 to Site 2 is easy to understand when viewed through VirtualCenter.

Figure 7.2 illustrates VirtualCenter output for two separate sites where the administrative teams have mapped business functions to folders and then replicated this structure at the opposite site ready for failover.

The naming convention used here was chosen to help show the logical function of the virtual machines and other components in screenshots. There is an argument that the machine name should just be an index, say m1001. This index could be used to reference a database such as a CMDB for further detail about it function and characteristics. As organizations are adopting more automated process management principles this could be a better strategy. The main point here though is to ensure consistency. VMware is creating a number of initiatives in this area with configuration management vendors and experts to create best practices for ongoing operations[5].

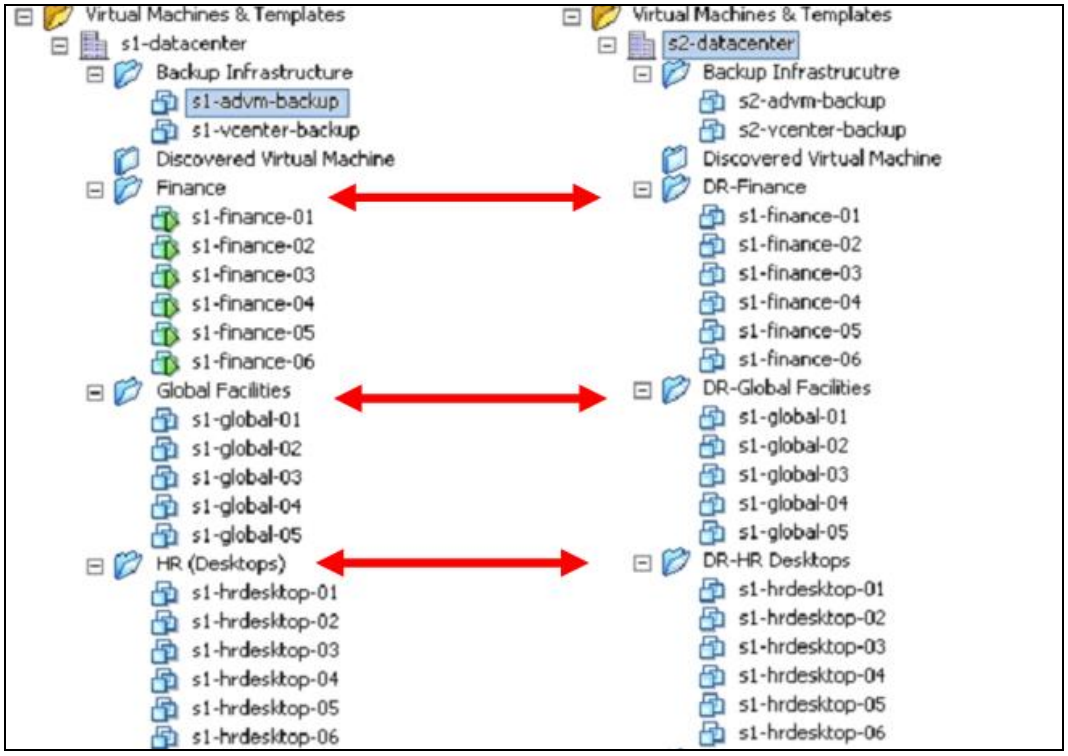


Figure 7.2 – Two-Site VirtualCenter Comparison

[5] http://vpp-dev-1.vmware.com/home/index.jspa

Storage Configuration

When reviewing the storage configuration, you need to understand the logical and physical layout and how it is configured at both your primary site and failover site. This should extend to fabric configurations and LUN placements. Again, simple designs tend to work most effectively for this purpose and simple naming conventions make relationships easy to understand when dealing with replicated storage environments.

Using Figure 7.2 as an example, the "Finance" virtual machines reside on the shared storage present at Site 1 and are replicated to Site 2. There is a need to ensure that the replica copies of this storage are presented to and available to the correct VMware ESX hosts at the failover datacenter.

For example, if your failover site contains multiple clusters of VMware ESX hosts, then it will usually not be the case that you need to make the replicated storage presentable to all clusters. Typically you will decide which services will failover to which cluster at the failover site in advance and create the associated placeholders (resource pools / folders) within that target cluster.

For further details on the storage configuration used within this document, please refer to chapter 8 and 9.

Storage Replication

Virtual machines typically reside on VMFS datastores which are created on LUNs that reside on the storage arrays. These LUNs are replicated between the two sites using the storage array vendors' replication technology. Each array vendor will produce a replication solution for their supported storage arrays which replicate data at the block level within the LUN at the primary site to a copy LUN at the recovery site. During business-as-usual (BAU) periods the LUN at the recovery site is presented as a read only copy to any VMware ESX hosts connected to it at that site. The LUN is read only at this stage as it is being kept in a synchronized state with its partner LUN (or source LUN). Synchronization levels states are commonly referred to as synchronous or asynchronous.

It is not a prerequisite of the VMware administrator to also be an expert in storage replication technology although when moving your architecture to a DR capable solution a basic understanding of the technology used at your location will ensure that the storage team create the underlying replication solution as expected and the virtual infrastructure team know which LUNs (and datastores) should be used in order that their virtual machines can be protected.

One simple concept is to understand at what level your storage replication solution is replicating. Within any storage array their will be multiple logical concepts that represent units of storage, terms such as ldevs, hypers, raid groups, array groups, DR groups, consistency groups, LUNs, volumes, flexible volumes, flexclones all have different meanings and some are used by some storage vendors whereas others are not.

From the VMware ESX side of things, what is important in your storage design is the unit which is used as the basis for replication. For example, some storage vendors have the concept of a volume, within that volume are LUNs and it is the LUNs that are presented to ESX and turned into VMFS datastores. However the replication technology in this vendors case works at the volume level and it is possible that a volume could contain more than one LUN, so what?

If you did not wish to replicate all LUNs within that volume, then this design would not work because you would be replicating LUNs that were not required as part of your BCDR design. In this simple example, a better design to provide more granularity and flexibility for replication would be to have a single volume per LUNs.

NOTE: Involve the storage team in the design and ensure your storage is laid out efficiently to allow you maximum flexibility when it comes to choosing which VMFS datastores to replicate and which can remain local to their site.

Once the storage design is finalized the storage team will then enable storage replication. The final decision might be on the type of replication to choose and by that we mean synchronous or asynchronous as mentioned earlier. Factors affecting the type of synchronization to deploy are beyond the scope of this chapter/book but will usually be decided based upon criticality of the applications within the virtual machine and recovery point objectives.

Note that using a mix of synchronous and asynchronous replicated LUNs in a VMware environment is completely possible, although you should also consider multitier applications and interdependencies between applications.

Failover Planning

Now that the storage is replicating from one datacenter to the other a determination must be made about how to gain access to the virtual machines within this storage and how to make this process reliable and repeatable. The following section discusses the issue of registration of the protected virtual machines (virtual machines) into the target VirtualCenter infrastructure.

Before virtual machines contained within the replicated LUNs can be accessed from within VMware VirtualCenter we must be able to "see" them. By "see" we mean have their icons appear inside VirtualCenter at the failover location.

Adding existing virtual machines into VirtualCenter is a process commonly known as registration, basically adding the virtual machine to the existing inventory.

Within the primary site VirtualCenter inventory this registration process was taken care of automatically when we created each of the virtual machines. At the failover site we must add the virtual machines to the VirtualCenter inventory.

It is a current requirement within the VirtualCenter architecture that in order to register a virtual machine the storage that virtual machine resides on must be present and accessible to the VMware ESX hosts within that site. To make the replicated storage available at the failover site and achieve this we are going to present a replica of the production LUN to the target VirtualCenter server. This step is also the first thing to do once we have the replicated LUNS visible in the target site if we want to restart the machines in the target site.

The registration process of virtual machines on previously unseen LUNs is relatively straight forward and once complete will be persistent in the target VirtualCenter database, even if the LUN is only presented transiently. If the LUN is subsequently taken away the entries for the virtual machines (icons in the VirtualCenter inventory display) will change state to become a "greyed" out version of their normal active/valid state.

Registration is a one off process in most situations. If you have a small number of virtual machines it is perfectly possible that you could go through this process each time you invoke DR. The process itself as you will see is only a few mouse clicks and takes around 30 seconds to achieve. If you have hundreds of virtual machines, you may want to consider automating this process as even if you have several helpers it could take some time just to complete this step. To see how you might go about this automation lets first look at the manual process.

If we consider the small protection group: Protection Group 1. It consists of two replicated VMFS storage areas RP1 and RP2. You can use the MAP tab in Virtual Center to graphically show the relationships between the Storage and the hosted virtual machines. In the figure 7.3 we can see a number of virtual machines that along with RP1 and RP2 form Protection Group 1.

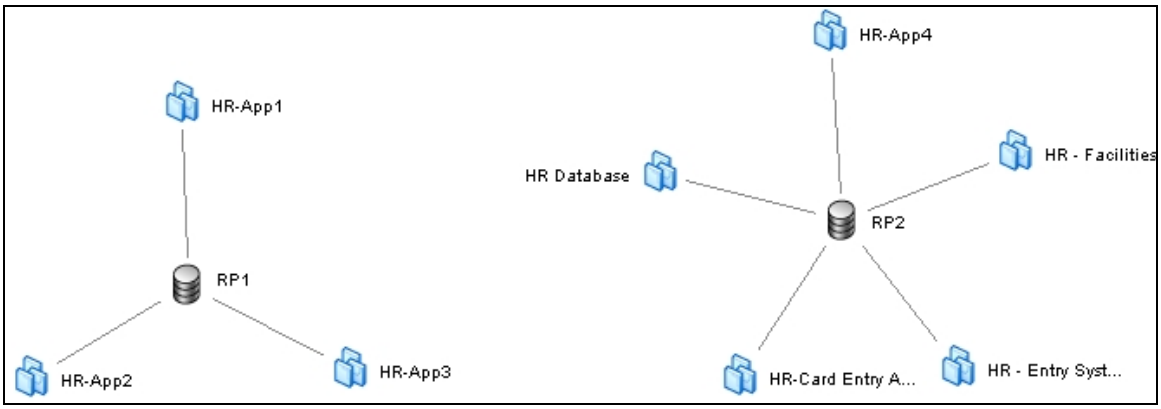


Figure 7.3 - Protection Group 1

Here we can see a number of virtual machines associated with the HR function within our simulated inventory. We have two database servers: HR Database and HR – Entry System, we also have a number of application servers HR-App1 through 4 and HR – Entry System. While the names, and even the functions, of the virtual machines in this example are not that important it will be fairly clear that for larger protection groups a clear naming convention is desirable. This view is also useful to check that 'stray' virtual machines are not on the replicated LUNS that shouldn't be.

In the storage chapter 8 we show how to effectively replicate the LUNS, in this case RP1 and RP2. Most modern array types allow you to create a writeable 'version' of the replica LUN in the target site. Different array technologies use different techniques and you can refer to the Storage section for more details on this, but they are often extremely efficient and initially often are close to zero bytes of additional storage to implement.

We use this type of feature to initially show a copy of the replicated LUN as it would be in a real failover scenario. This is useful not only for the registration process but also for test scenarios as we can test the failover scenario without removing the protection of the primary LUNS in the replication group. Some implementations also call for this type of function to keep a 'snap shot' of environment at a known time. This can protect against wide scale corruption for example which would be blindly replicated by the underlying array.

NOTE: For further detail on the presentation of LUNs (and snapshot LUNs) please refers to the storage connectivity and storage platform chapters 8 and 9.

Service Failback Planning

This section covers some high-level considerations and concepts that should be reviewed when planning failback. This section is intended to simply provide some basic starting points to help with the planning process.

The most advanced application of service failback is a scenario which involves deliberately failing over an IT service, running at the new location for an agreed period and then failing back to the previous location at a later date. This is typically categorized as "Operational Failover". The majority of implementations would not follow this model and would only perform failover as a result of an unplanned datacenter outage.

When performing a service failback, a number of decisions need to be made that are both technical and non-technical before the process can begin. For this reason, the most common approach is to failback on a per-application or per-service level rather than failing back everything at once.

Here are some non-technical factors that could be fed into a failback decision and process:

- Time. Failover invoked how long has service been running at recovery site? Business Decision: If length of time substantial how will another outage for failback impact business again if time is great and effective state is business as usual at failover site. Would failover site be promoted to primary?
- Primary site destroyed by some kind of disaster. Business decision: would primary site be rebuilt in same location?
- Primary site damaged but repairable. Business decision: how long will repairs take?
- Are services running at reduced SLA / QoS levels whilst running at failover site?

Secondly technology based capabilities / limitations will also influence failback strategy:

- Time. As with non-technical time is major factor. Length of time running at failover site will effect ability to recover / restore data incrementally to primary site. Long period may equal full copy restores.

- Storage solution capability. Different storage solutions have varying levels of restore and reversal capability when it comes to moving data back to primary site. This needs to be reviewed.
- Is any hardware replacement required? Has only a subset of the hardware been destroyed or damaged beyond repair? Any replacements will need to be reconfigured.

To try and illustrate one basic decision tree for failback the figure below depicts an outline workflow that should be thought through when contemplating a failback strategy.

What should become clear is that failback decisions will, to a certain degree, depend on what caused the original failover to take place. This is where a site disaster recovery audit is applicable in defining the possible scenarios and likely outcomes.

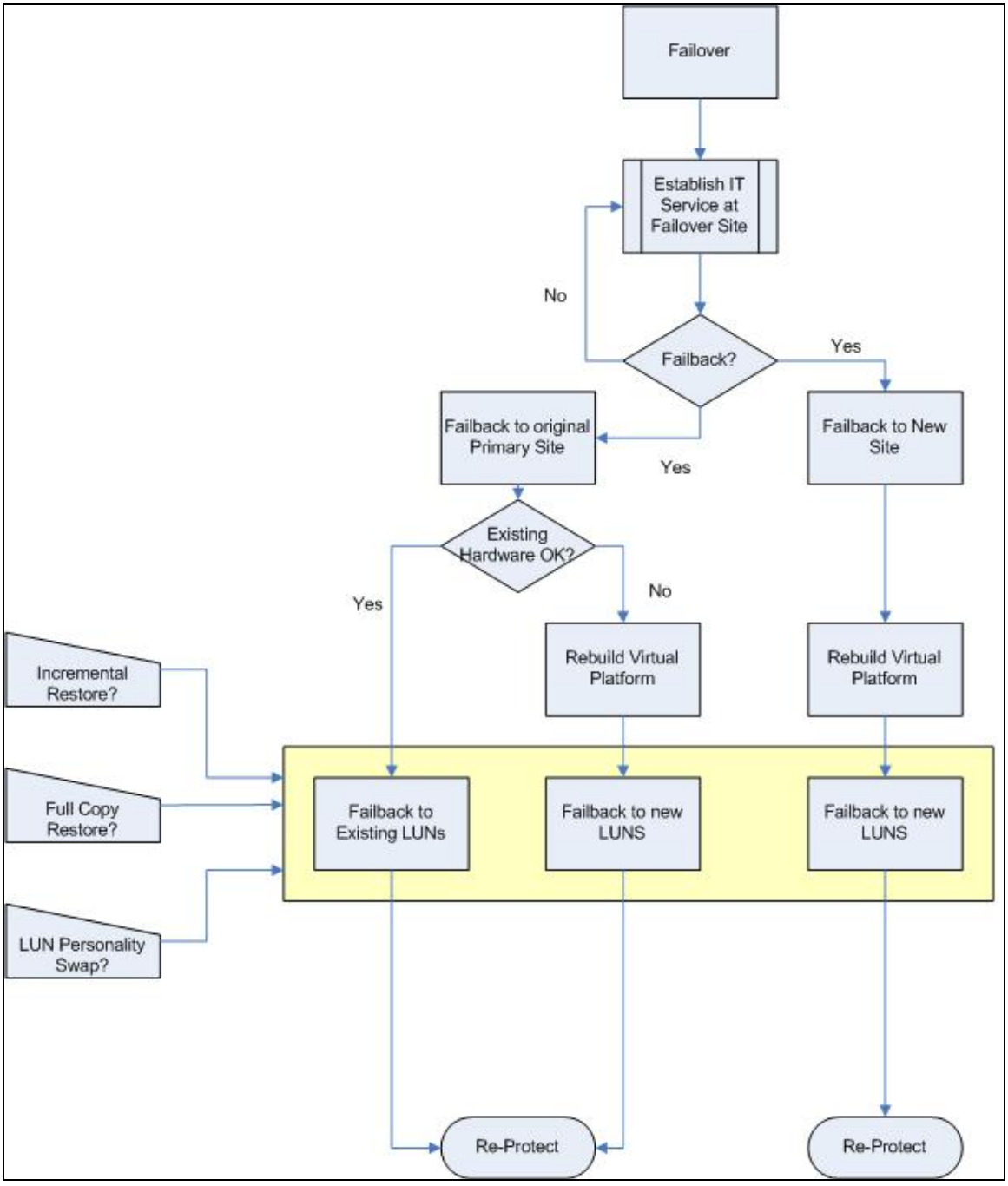


Figure 7.4 – Example Outline Failback Process

From a VMware Infrastructure point of view, there are a number of steps that must be followed when performing failback:

- Storage team must configure and start reverse replication before failback can occur.
- Virtual machines must be powered off at the recovery site.

- Virtual machines must be re-inventoried at the source site.
- Virtual machines must be powered on following the same workflow steps as failover.
- Machines will be re-addressed and appropriate infrastructure components will be updated (such as DNS) where necessary.

Virtual Infrastructure Storage Failback

As mentioned previously in this chapter, the process of failing back the storage containing your virtual machine disk files. During failback, this is typically achieved in stages. Once failback has occurred, the first step is to ensure that all data changes made at Site 2 (failover site) since the initial failover occurred are replicated back to Site 1 (primary site).

Depending on your storage solution, there are a number of ways to re-synchronize storage at Site 1 with the storage at Site 2. Considerations here are technology based and, as previously described, the method chosen will be affected by time spent in DR and rate of data change.

To illustrate this point, Figure 7.5 shows the logical view of storage at Site 1 being updated with changes made during DR to storage contained at Site 2. Storage array technology deployed will determine the technique used.

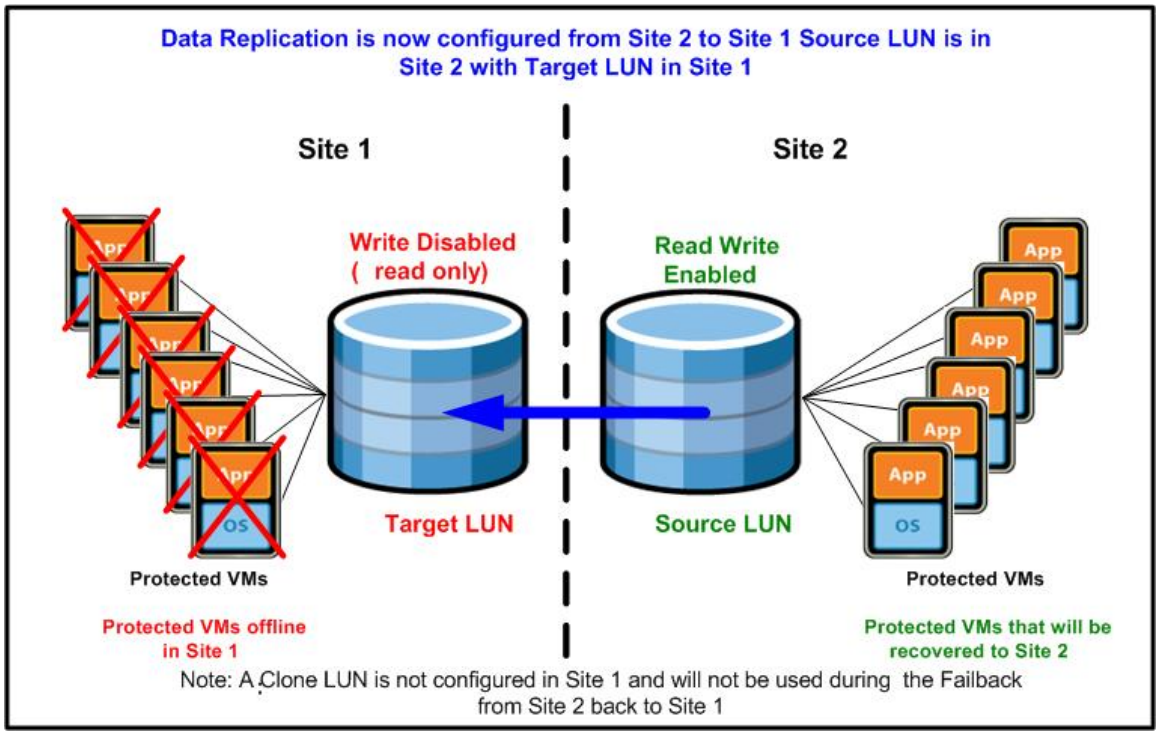


Figure 7.5 - Reverse Replication Site 2 to Site 1

Once data changes have been re-synchronized from Site 2 back to Site 1 the next step is to re-establish the original state, which is Site 1 is live and replicates its data to Site 2. Again, the process to achieve this will be determined by the storage array solution being used. To illustrate this final part of the process Figure 7.6 shows that Site 1 storage is now live and is once again is replicating its data to the Site 2 storage. Figure 7.6 also shows that the virtual machines in Site 1 are now once again protected by Site 2.

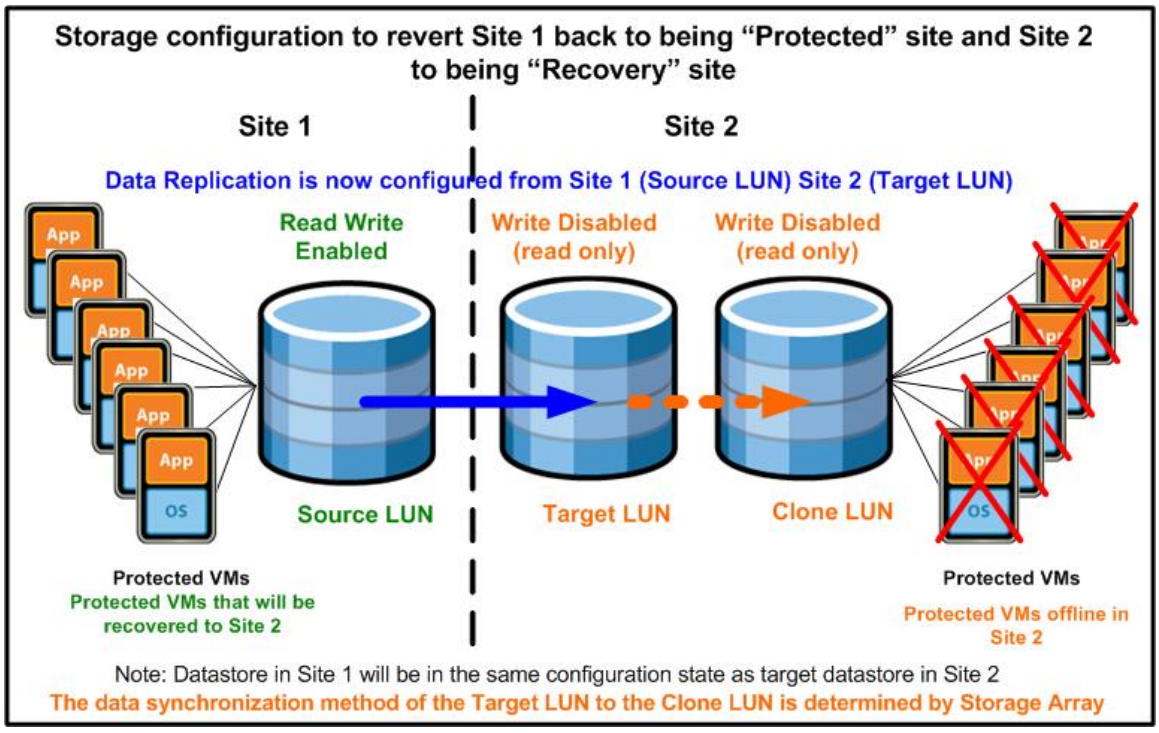


Figure 7.6 - Re-Protect Site 1

In summary, the failback process can be a complex planning problem and should not be underestimated. From a technology perspective, make sure you understand fully the implications of networking and, more particularly, storage replication reversal and re-instantiation of your business as usual protection schemes. Understand the one-to-many relationship between the different failback scenarios; going to DR can have a number of causes but you always get one outcome (hopefully). Failing back will always have multiple potential outcomes as complete loss is normally a genuine consideration. Consider time as a variable in your planning work-flow and make regular and documented tests if possible to ensure success.

Chapter 8. Service Failover Testing

Failover Considerations

Having considered the high-level principles and the logical issues related to moving the virtual infrastructure Inventory to another site, as well as the alternative solutions discussed in Chapter 6, this section looks in more detail at the mechanics of performing the failover in an automated fashion.

BCDR plans have traditionally been documented as runbooks – i.e., what to do if disaster strikes. Increasingly, this runbook is being automated to make the process more predictable and less prone to error. The ability to test this plan is also a key consideration.

In real life, this will be much more complex, but the process shown in Figure 8.1 details the basic mapping of the inventory in site 2.

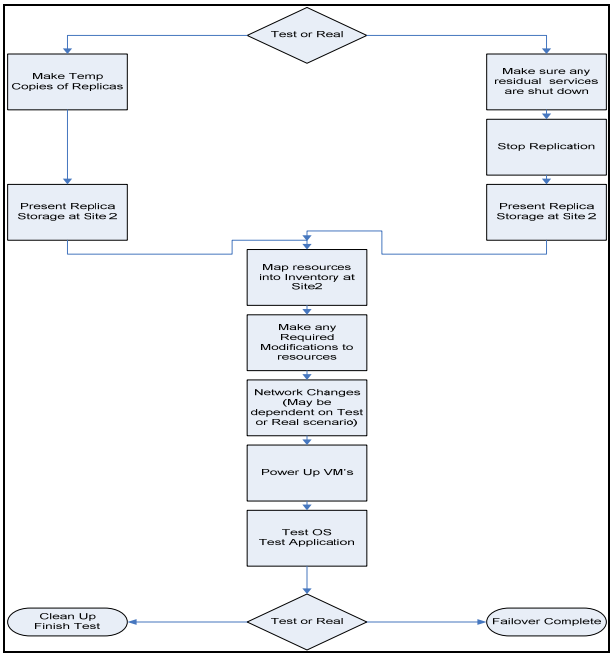


Figure 8.1 Basic Flow of Virtual Infrastructure Recovery/Test

Additional Considerations

Networking – During a failover, it is important to determine what needs to be tested in what order, for example an initial success criteria could be to get all your virtual machines registered in the failover VirtualCenter inventory and successfully powered on. This might not initially include establishing

network connectivity as this may be part of a later test (changing the GuestOS IP Address for example). It may therefore be useful to bring all the virtual machines online with their NIC cards disconnected (this can be automated via the VMware SDK) especially if during failover you have a stretched network across sites and the virtual machines will come up with same ip addresses they had at their source site, Site 1. Other networking considerations are further discussed in chapter 9.

Capacity – Very simply, have you got enough (predominantly we are talking CPU / Memory resource)? We cannot simply assume that every solution will consist of a "like for like" active / active datacenter model with same number VMware ESX hosts running at both sites which are in turn running same number of virtual machines at the same utilization levels and these hosts are of the same hardware type and configuration. It could be that the failover site runs a reduced number of VMware ESX hosts and usually these hosts are only running your Dev / Test / UAT environments. In the event of DR or failover testing then it could be the case that to perform the test or failover successfully the virtual machines running at the failover site would need to be powered down or suspended in order to free up capacity (CPU and Memory) for the virtual machines being recovered as part of the failover or test.

VirtualCenter "Look and Feel" – Usually those responsible for designing the VMware virtualization architecture are not the same people then tasked with running the architecture on a day to day basis. Part of your design work should definitely consider how you will represent the failover capability through the VMware VirtualCenter GUI. What kind of naming conventions will you apply for the object used to hold the recovered virtual machines? When will these be created and more importantly populated with virtual machine icons. How will you familiarize the operations team with the new structure? How do they currently identify virtual machine failure and taking that a stage further would they actually be the ones responsible for running the recovery process?

Actual Service Failover

During an actual service failover event we would typically not use clone (or replica or snapshot) copies of LUNs at the failover site. In the event of an actual outage we would typically write enable the replica (or target) LUN at the failover site and present this LUN (s) to the VMware virtual infrastructure layer. From that point forward recovery of the virtual machines would follow the same process already described for failover testing in Chapter 5.

Figure 8.2 illustrates the basic difference in storage presentation for actual service failover. Note this is a high-level illustration and individual storage vendors have different solutions / recommendations for achieving the same state. Always work with your storage vendor and obtain their best practice for this stage of recovery/failover.

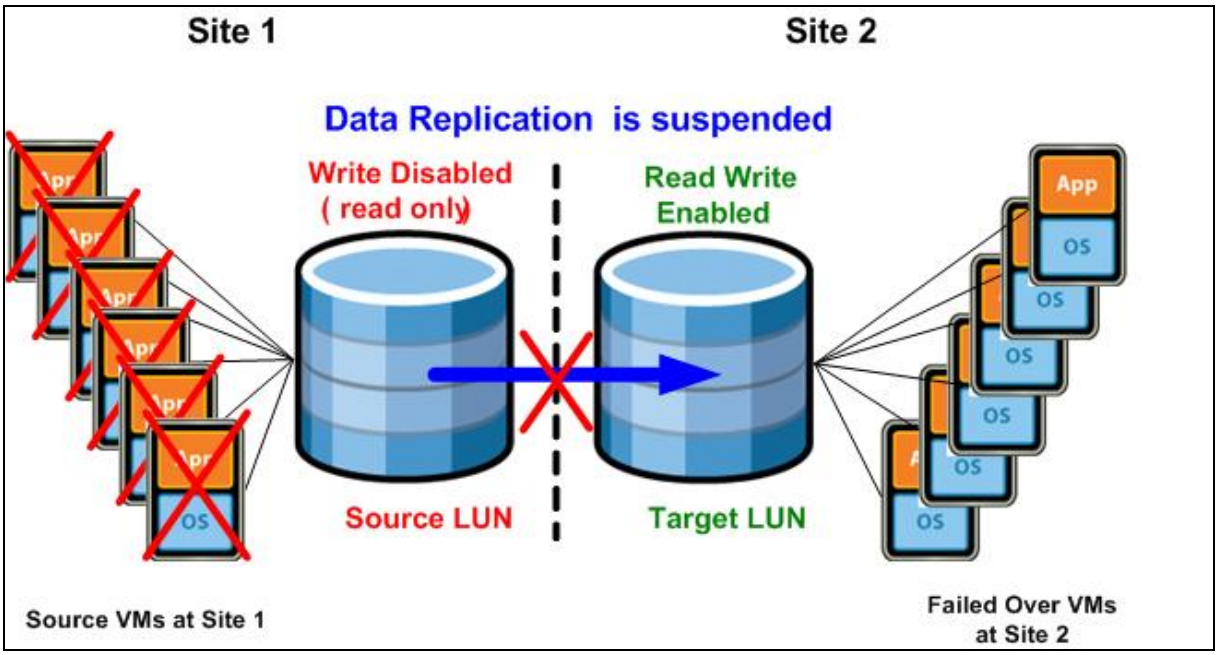


Figure 8.2 – Actual Failover (Storage Level)

Apart from the storage layer differences utilized during actual failover other elements of your recovery plan will also be called into action during actual failover. If we assume that a basic IT service could be provided by (and usually always will be) more than any one single virtual machine then these dependencies need to be understood and built in to your recovery process.

Any recovery plan should therefore also include at least the following considerations

- External resource or application dependencies. For example any networking and storage resources, dependencies on other applications must be determined to ensure the virtual machine can be brought up at the recovery site.
- Virtual machine startup order.
- Target Virtual Infrastructure Inventory
- Checkpoints that occur between startup of virtual machines. One simple example would be a multitier application such as a CRM environment. CRM systems will usually consist of a web tier, messaging tier and a RDBMS backend. All of theirs tiers or levels could be provided entirely as virtual machines. You can start to see here that to test or recover that IT service you will need to test or recover all of the tiers as a single unit to ensure all virtual machines within those tiers are recovered to the same point in time. These kind of considerations need to be planned alongside your infrastructure design work.

- Define a User Acceptance Test (UAT) for each application that will be part of the recovery plan.

Service Failover Automation

One of the most common requirements in any VMware BCDR design is automation. We have discussed in this chapter the basic workflow for performing virtual machine failover and it can be seen that for large numbers of VMware ESX hosts and virtual machines there are potentially a number of manual steps that could in a real instance of failure be a source for human error.

To automate failover and ensure a consistent, reliable, error free approach most customers will look to script all or the vast majority of the failover processes.

One key thought to always remember with the VMware architecture is that management and automation is one of the key drivers for moving to a virtual platform. The VMware VirtualCenter API allows customers to programmatically run all of the same tasks that can be performed manually through the VirtualCenter UI.

To date the most popular way of achieving automation has been through the use of the VMware Perl ToolKit available from vmware.com. This toolkit allows customers to create reliable scripts that can perform all required steps to control their VMware environments. For those new to the perl language the toolkit also ships with a large catalogue of sample scripts that illustrate how to perform many of the key operations needed to achieve a scripted failover solution.

VMware have also released another toolkit based on the Microsoft powershell framework which provides the same capabilities as the perl toolkit. Both are available to download at vmware.com.

To automate the scripting side of our solution in this book the VMware Perl Toolkit was utilized to produce a script that would carry out the following tasks:

- Rescan VMware ESX HBAs.
- Search selected (failed over) Datastores for virtual machine configuration files.
- Compare discovered virtual machines in datastores to VirtualCenter Inventory.
- Register virtual machines not contained in inventory.
- Un-Register / Re-Register any discovered virtual machines with stale or existing entry in VirtualCenter Inventory.
- Remap virtual machines portgroup to valid recovery site portgroup.

- Connect / Disconnect virtual machines network card at power on.
- Power on discovered virtual machines in sequential batches.

The resulting script produced is wholly contained within the appendices of this book. In this chapter we will take a look at sub sections of the script to illustrate some key tasks that the script is carrying out on our behalf.

During use the BCDR Perl script would be invoked using a desktop shortcut that executes a batch file but during development you have the chance to make the first significant decision, what command line options should our script allow?

As an example with our script to invoke via a bat file / shortcut you would need to add the command line syntax into the bat file / shortcut in its entirety. In our case this would be:

```
perl -w <script>.pl --server 192.168.2.40 --url https://192.168.2.40/sdk
--username administrator --password vmware --datacenter "Site 2" --path
"DR-virtual machines/Protection Group 1/RecoveredVMs" --datastore DEV1
DEV2 --network false --pool ReActivatedVMs --portgroup "Site2-lan"
```

The command line options used above are detailed in the following table:

Option	Option Description
--server	IP Address or FQDN of VirtualCenter server script will run against
--url	SDK URL relating to –server IP Address or FQDN, format is https://<ipaddress>/sdk
--username	VirtualCenter Username
--password	VirtualCenter Password
--datacenter	Name of the VirtualCenter datacenter object within which the failed over virtual machines will be registered and powered on. (required)
--folder	Name of the folder within which you wish your virtual machines to be registered, this option is only for folders that have no child folders or are

	children of another folder. (optional)
--path	Absolute folder path. If you wish virtual machines to be registered within a folder that IS a sub-folder use the path option. For example if the target folder is "RecoveredVMs" and this folder is a sub-folder nested two levels down underneath parent folders "DR-virtual machines > Protection Group 1 >" then your path is "DR-virtual machines/Protection Group 1/RecoveredVMs". (optional)
--datastore	Datastore search specification. This option allows you to specify the datastores the script will search for your virtual machines (search is based upon finding vmx files). You can specify a space separated list of datastore names such as: VMFS1 VMFS2 VMFS3 VMFS4 or you if your datastore naming convention follows a stand such as Site1VMFS2 Site1VMFS2 Site1VMFS3 Site1VMFS4 you can wildcard this to reduce the command line, this could be reduced to Site1* (required)
--network	Determines if virtual machine NIC is connected / disconnected. NIC cards within the virtual machines will start "connected" or "disconnected" at PowerOn. If this option is set to false then NICs will be disconnected. true means connected. (optional)
--pool	Resource Pool name. The pool option dictates the name of the resource pool the recovered virtual machines will be assigned to. In our environment a generic pool was created to house all failed over virtual machines. This facilitated easy identification and management of those virtual machines that are part of the failover process. (required)
--portgroup	Portgroup Name. The portgroup option specifies the name of the portgroup that the virtual machines being failed over will connect to at the failover datacenter. It cannot be assumed that the same portgroups will exist at both sites so this option allows you to change the setting for all virtual machines failed over. (optional)

To capture these options in the script we use a basic structure that allows us to define the command line option name, type, if it is required (denoted by required => 1) or not and a help message, this is shown in the following script section:

```
my %opts = (
                datacenter => {
                    type => "=s",
                    help => "Datacenter name",
                    required => 1,
                    },
                datastore => {
                    type => '=s@{1,}',
                    help => "DataStore wildcards",
                    required => 1,
                    },
                folder => {
                    type => "=s",
                    help => "folder",
                    required => 0,
                    },
                 path => {
                    type => "=s",
                    help => "Sub Folder to add the VM(s) to",
                    required => 0,
                    },
                 portgroup => {
                    type => "=s",
                    help => "Port Group for recovered virtual machines",
                    required => 0,
```

```
                    },
                network => {
                    type => "=s",
                    help => "Recovered virtual machines Network connection
status is required true / false",
                    required => 0,
                    },
                 pool => {
                    type => "=s",
                    help => "Recovered virtual machines Network resource
pool",
                    required => 0
                    },
              );
```

One interesting challenge was virtual machine folder selection within VirtualCenter. During recovery the virtual machines being recovered are registered (or re-registered is a stale inventory item exists) within VirtualCenter.

Using the RegisterVM method we need to pass into the script the folder object under which we want our recovered virtual machines to be attached. Therefore we need to allow the end user the ability to pass this "target" folder name into the script as a variable (in our case the variable "folder").

One subtle issue however was how to handle non-unique folder names. For example let's assume we have two parent folders, one called "Finance" and one called "Procurement". Within each parent folder is a child folder called "Tier1-VMs". We now have two folders called "Tier1-VMs" each with a different parent folder so we need a way for the script to identify which child folder we are referring to for recovery.

To accomplish this we allow the user to pass in simply the folder name (if unique) or a patch to the folder if non-unique. For example if the target folder is "Tier1-VMs" within the "Finance" parent folder then your path is "Finance/Tier1-VMs". Following code snippet shows this taking place:

```
####################
# Folder Selection #
```

```
####################


## alternative way of getting folders .... replaces your folder
selection

## basically, if path is set then it selects the folder from the path

## otherwise, simply find the folder that was specified


my $folder_view;

# if the user provided a path then find the folder this way


if (Opts::option_is_set ('path')) {

    my $searchPath = '/'.$datacenter.'/vm/'.$path;

    my $searchIndex = Vim::get_view(mo_ref =>
Vim::get_service_content()->searchIndex);

    my $folder = $searchIndex->FindByInventoryPath (inventoryPath =>
$searchPath);

    die ("Folder not found.\n") unless (defined $folder);

    $folder_view = Vim::get_view (mo_ref => $folder);


    }

else {

    die ("Must supply either --path or --folder") unless
(Opts::option_is_set ('folder'));

    my $folder_views = Vim::find_entity_views (view_type => 'Folder',

                                               filter => {'name' =>
$folderName},

                                               begin_entity =>
$datacenter_view);

    die ("Folder $folderName not found.\n") unless (defined
$folder_views);
```

```
    die ("Folder $folderName not unique.\n") unless ($#{$folder_views}
== 0);      # only one entry
    $folder_view = shift @$folder_views;
    }

############################
# End Folder Selection    #
############################
```

Once we have our virtual machine folder destination and list of discovered virtual machines for recovery the key step is to register these virtual machines within VirtualCenter. The primary reason for re-registering a virtual machine whose icon already existed is to ensure the object reference is correctly defined so that when the operator works against the virtual machine performing a power on operation for example they do not receive any strange errors relating to object instance not set or device inaccessible. If a virtual machine had been previously registered using a different storage location that could be a risk so by re-registering the discovered virtual machine we ensure a clean configuration for the end user.

During virtual machine registration we do not pass in the name of the virtual machine to be used within virtual center. Using RegisterVM method if the display name is omitted then the method will use the display name contained within the virtual machines configuration file which keeps things clean and simple. Finally at virtual machine registration we also now pass in our value (given via the command line) for target resource pool.

```
print "\n*******************************************************\n";
print "* Protected Virtual Machines Unregister / Re-Register *\n";
print "*******************************************************\n";

my $new_vms;
foreach my $path (keys %vmsByPath) {
    print "\n" . $path . "\n";
    my $vmname = $vmsByPath{$path}->name;
```

```
        if ($vmname ne 'not registered'){
            my $vm_view = $vmsByPath{$path};
            my $powerstate = $vm_view->runtime->powerState->val;
            print "VM Power on State is: " . $vm_view->runtime->powerState-
    >val . "\n";
            if ($powerstate eq 'poweredOn'){
                print "Powering off " . $vm_view->name . "\n";
                $vm_view->PowerOffVM();
                print "Poweroff successfully completed\n";
                }
            print "Unregistering VM.......\n";
            $vm_view->UnregisterVM();
            }
        print "Re-Registering VM with VMX Path: " . $path . "\n";
        my $new_vm = $folder_view->RegisterVM(path => $path, asTemplate =>
    'false', pool => $pool_view);
        push (@$new_vms, $new_vm);
        }


    print "\n";
```

As with any script designed for production use we need to ensure that we, as far as possible, cater for all possibilities and this bring us on nicely to virtual machine UUID questions. Depending on what configuration changes have been made to a virtual machine the status symbol next to your virtual machine may sometimes change from the normal green “powered on” symbol to a variety of other symbols one of which is a question mark. When the question mark icon is display this is simply VirtualCenter telling you there is a question to answer via the UI. If a virtual machine's location has been altered in any way or a stale inventory item has been deleted and re-added then VirtualCenter will prompt the user a question asking if the virtual machines UUID should be changed. At this stage in the UI a radio button select list is usually presented.

In terms of automation having your recovery process hang whilst these questions potentially appear would not be ideal, therefore it is useful to have a subroutine in the script that can answer the question for you. In our case we elect to always "KEEP" the uuid, the routine used to put this in place is shown next:

```
##########################################
# Sub Routine for UUID Question Answer   #
##########################################

sub checkAndKeepUUIDs {
    my $vm_views = Vim::find_entity_views(view_type =>
'VirtualMachine', filter => { 'runtime.question.id' => '\d+'});
    foreach my $vm_view (@{$vm_views}) {
        if (defined($vm_view->runtime->question->id)) {
            print "Keeping VM UUID for " . $vm_view->config->name
. "\n";
            my $vm_question_info = $vm_view->runtime->question;
            my $vm_question_id = $vm_question_info->id;
            my $vm_question_answer_choice = 2 ;#Keep
            $vm_view->AnswerVM(questionId => $vm_question_id,
answerChoice => $vm_question_answer_choice);
            }
        }
}
```

One of the more involved items for successful virtual machine recovery that we wished to automate was virtual machine network connect / disconnect and portgroup remapping. Via the command line we wanted to be able to let the user have the virtual machines recovered but with all of their network cards disconnected at power on, this would be useful for testing purposes where you simply wanted to test the storage recovery but by default did not wish to connect the virtual machines to any

network. We also wanted to let the user redefine the portgroup used by the virtual machine to allow it to connect to a valid portgroup.

The reason the portgroup defined for the virtual machine may need to be remapped is simply because the virtual machines original portgroup may not exist (or be defined) at the target/failover datacenter.

For example you had chosen to prefix all your portgroups in a particular datacenter with the name of the geographical location then they would be unique to that site. In our solution if we have a portgroup named "Site1-ProductionLAN" and at the failover location, Site 2, a portgroup named "Site2-ProductionLAN" then any virtual machines being recovered from Site 1 to Site 2 that were originally connected to the "Site1-ProductionLAN" portgroup would not automatically know to now connect to "Site2-ProductionLAN" portgroup. The script needs to make this mapping change if we wish to automate that process. The next section of our code was used to achieve this, note the name of the failover site portgroup to be used as the target is supplied to the script via a command-line option.

```
##########################################
# Sub Routine for VM Network disconnect    #
##########################################

sub disconnectNetwork {
    my ($vm_reconfig_view, $vm_reconfig_network_devices) = @_;
    my @vm_reconfig_devices;
    print "Setting Networks...\n";
    foreach my $network_device (@{$vm_reconfig_network_devices})
{
        my $network_address_type = $network_device->addressType;
        my $network_device_controller_key = $network_device-
>controllerKey;
        my $network_device_key = $network_device->key;
        my $network_device_mac_address = $network_device-
>macAddress;
```

```
        my $network_unit_number = $network_device->unitNumber;

        my $network_wake_on_lan_enabled = $network_device-
>wakeOnLanEnabled;

        #print "Setting Network: " . $network_device->deviceInfo-
>summary . "  Connected=" . $nicopt ."\n";

        #my $network_backing_info =
VirtualEthernetCardNetworkBackingInfo->new(deviceName =>
$network_device->backing->deviceName, network => $network_device-
>backing->network);

        my $network_backing_info =
VirtualEthernetCardNetworkBackingInfo->new(deviceName =>
$portgroup, network => $network_device->backing->network);

        print "Setting Network: " . $portgroup . "  Connected=" .
$nicopt ."\n";

        #my $network_connect_info = VirtualDeviceConnectInfo-
>new(allowGuestControl => 'true', connected => 'false',
startConnected => 'false');

        my $network_connect_info = VirtualDeviceConnectInfo-
>new(allowGuestControl => 'true', connected => $nicopt,
startConnected => $nicopt);

        my $network_device_info = Description->new(label =>
$network_device->deviceInfo->label, summary => $network_device-
>deviceInfo->summary);

        my $network_config_spec_operation =
VirtualDeviceConfigSpecOperation->new('edit');

        my $vm_reconfig_device = VirtualPCNet32-
>new(addressType=> $network_address_type, backing =>
$network_backing_info, connectable => $network_connect_info,
controllerKey => $network_device_controller_key, deviceInfo =>
$network_device_info, key => $network_device_key, macAddress =>
$network_device_mac_address, unitNumber => $network_unit_number,
wakeOnLanEnabled => $network_wake_on_lan_enabled);
```

```
        my $network_config_spec = VirtualDeviceConfigSpec-
>new(device => $vm_reconfig_device, operation =>
$network_config_spec_operation);

        push(@vm_reconfig_devices, $network_config_spec);

        }

    my $vm_reconfig_spec = VirtualMachineConfigSpec-
>new(deviceChange => \@vm_reconfig_devices);

    $vm_reconfig_view->ReconfigVM_Task(spec =>
$vm_reconfig_spec);

}
```

To view the failover script in its entirety, see "**Appendix A: BCDR Failover Script**."

Having presented the storage volumes as necessary we then just execute this script to establish the VM's in the Site 2 Inventory, make the appropriate changes to any properties and then power the virtual machines on. At this point you are now ready to test/operate.

PART IV. Solution Architecture Details

Chapter 9. Network Infrastructure Details

The purpose of this chapter is to describe the network architecture deployed across both Site 1 and Site 2 datacenters. We will also discuss general considerations and issues that should be investigated before deciding upon your BCDR architecture using VMware Infrastructure.

General Networking Considerations

Modern applications are almost always dependent on networking to some degree or another and so physically moving a service from one site to another represents a networking challenge whether the services are hosted in virtual machines or physical machines alike. Network addressing is not generally location independent mainly due to poor application use of network addressing technologies that have been available for many years. However we have to deal with applications and configurations which depend on hard coded IP addresses for example. There are two main ways that this problem is dealt with in DR solutions.

Firstly maintain the IP addresses. So despite the services moving and the hosting servers being in different physical locations we take with us the IP address configuration to the new location. The second approach involves completely changing the IP address during the transition into the recovered site. Each approach has several implementation variations which are summarized below. It may also be possible that you may have to contemplate both solutions.

Solution 1: Fixed IP Addresses

From a BCDR process perspective this is by far the easiest to implement but has a number of potential problems which seem in practice to make it the least popular approach. In this category there are two main sub-types.

- Stretched VLAN
    - Here the layer 2 network technology is made available simultaneously in both locations. In simple terms this means you can move a server and its IP (layer 3) configuration to the second site and the network will route the traffic to the new location transparently. This is obviously trivial to deal with from a server perspective but has a number of issues. Firstly it will require networking equipment that can managed this so called stretched VLANS, this is less of a problem as this is now widely available. The second and more difficult problem

is that by stretching the VLAN the potential fault domain is extended to both sites essentially becoming a single point of failure. While this is unlikely occurrence it is and has happened that a broadcast storm is started but cannot be isolated. We have found mixed opinion about this last issue and have seen many successful implementations as well as 'we will never implement this technology here'.

- Mobile VLAN
    - To obtain the benefits of the solution above without stretching the VLAN across multiple sites it is possible to implement a mobile VLAN. Here any given VLAN would be present at Site 1 or Site 2 but never both. In order to maintain the IP address space in the event of a failover it is possible to programmatically arrange for the router infrastructure to move the VLANs from one site to another. So in a failover scenario the VLANS would move with the associated protection groups. There are two potential draw backs to this approach we have come across. 1) In the event of a failure you have to move the whole VLAN, which maybe ok but may affect the granularity considerations that we discussed in the protection group discussion. If there is a 1:1 mapping of VLAN to protection group this would not be a problem. 2) It is more difficult to test the failover mechanism as in order to perform the test the live VLANs must be taken out of production use.
- Complete Layer 3 redundancy
    - Here you remove any VLAN technology and adopt a complete layer 3 approach to all of your estate. This will require more routing capacity but gives significant flexibility advances. This may still require IP change in the test mode however, but in real failovers the router infrastructure can take care of the new location automatically.
- Dark Site
    - Here the complete VC and ESX site is replicated to a dark site. Here the IP addressing is typically not an issue and all that is required is to swing the routing from the clients to the new location. This can be more challenging if the dark site is shared – in this case changing the IP address as below may be an appropriate strategy.

Solution 2: Changing IP Addresses

This approach seems to be the most prevalent we have seen. It takes the form of changing the IP address of every server that is involved in the failover. This again has a number of different implementation options.

The major draw back of this approach is that it can be very time consuming and requires the incoming network to 'learn' that the application that was at x is now at y. Even if the x and y are logical names DNS entries typically have to be changed/flushed throughout the network and cached entries in network tables have to be updated or flushed, so it is not uncommon to be faced with multiple hours of downtime while these state changes take place.

Generally speaking these problems are similar for both physical and virtual implementations. Although virtual machines do add some additional flexibility.

Implementation options:

- Script the server to change its IP state by detecting the LAN segment and making an appropriate choice at boot time.
- Use DHCP to associate a fixed IP address based on the MAC address of the server. The MAC address mapping in DR can be different so the server would resolve a different IP address in the DR mode but the IP address would be fixed in both cases. DHCP is often frowned upon in datacenter environments so this is often not a considered approach.
- Use Virtual NICs. In each guest you can associate multiple NICS. These are free in the virtual world. A static IP address can be associated with each NIC. For non DR use attach the NIC to the 'live' virtual switch and the second NIC can be either not attached or attached to a virtual switch with no uplink. In either case the guest will discover an inactive network segment and remove it from routing tables etc. In a failover scenario it is possible to manipulate the virtual machine to connect the alternative NIC only. In this case you achieve a known static IP address with a change in the DR scenario without requirement to interact with the guest directly or indirectly with a script.

Virtual Infrastructure Considerations

VirtualCenter

Virtual Center can be considered a standard Windows workload as far as site failover is concerned. If the VC database is in a 'new' location after a failover name resolution etc should be in place such that the DSN will be correct, any application that uses the SDK will have to informed if the VC name changes etc.

VMware ESX hosts

VI generally gives additional flexibilities over and above what would be possible for physical servers. An additional consideration is the network infrastructure supporting the ESX servers themselves. At minimum there will be a management LAN segment to consider. It is best practice that the management segment be separate from the virtual machine LAN traffic. VMware publish a number of whitepapers on networking and configuration options two which are good first reads are:

http://www.vmware.com/files/pdf/virtual_networking_concepts.pdf

http://www.vmware.com/pdf/esx3_vlan_wp.pdf

In most Enterprise environments there may be additional management segments, VMotion, backup and network storage segments need to be managed/re-established. In an active-active setup these aspects are all taken care of as part of the site build and are fully independent. In an active passive setup or where the Virtual Center server IP address changes then at a minimum the ESX servers will have to be re-registered on the new subnets if they are different and any domain registration changes or flushes performed before the ESX servers are made active. This can be avoided if the whole infrastructure can be moved over and you have control of the IP network in the remote site.

IP Storage

If you have IP storage solution involved in your infrastructure you most also consider the change in IP address implications for active passive setups. For iSCSI the ESX server management LAN must also be able to route to the vmkernel IP subnet, so routing must be appropriate for this. NFS storage will be dependent on DNS names in most configurations so again this will have to be in place before any virtual machines can be restarted. NFS is also dependent on passwords so these will need to be synchronized at both sites.

DNS

VirtualCenter and related applications are highly dependent on DNS. It is crucial that you have forward and reverse lookups operational as well as short and long name reservations.

Solution Approach

The solution we adopted for this document was effectively to choose the more complex option of changing the IP addresses of the servers. The alternative was to assume VLAN was stretched or moved during failover which would be simple to document but we felt more value would be derived from recoding the IP address of each server.

From a high level perspective we selected a discrete IP address range in each site for both the VMware Infrastructure management infrastructure and the guest operating systems. I.e. in a failover we change all of the protected virtual machines IP address.

From a VMware Infrastructure perspective all of the backing infrastructure is setup as part of the build out and remains static during the failover events so this aspect is very straightforward.

At each site we separated the virtual machine and Management Infrastructure onto two separate subnets/VLANS and provided a third for the router to use for inter site traffic.

In fact a fourth network existed to provide an IP replication transport. Logically this could be viewed as in Figure 9.1:

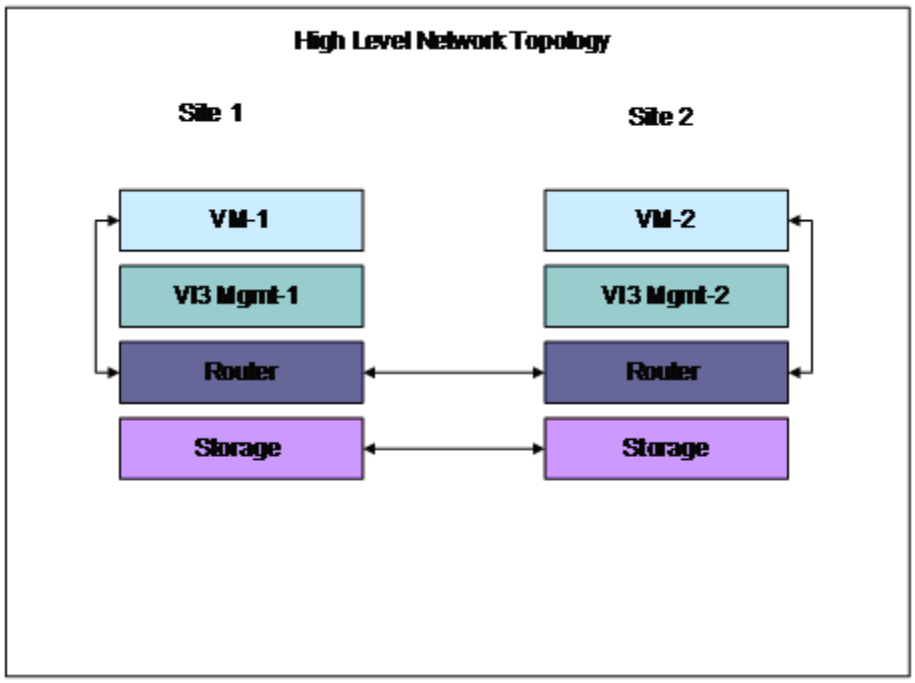


Figure 9.1 – Network Concept

There is a routed network between the two sites as far as the virtual machines are concerned. From a services infrastructure perspective we set up a single Active directory domain which is represented in both sites by two Windows 2003 Server guest and the domain is replicated amongst the four servers. DNS and DHCP is provided locally at each site and the server IP addresses for every server guest are fixed. Desktop virtual machines are allocated local addresses via DHCP.

With the networking hardware available to us we were not able to implement multiple VLANS for the virtual machines. This would be a simple addition and would give additional granularity and control at fail-over time. You may want to consider associating a VLAN per protection group and use Layer 3 capable switches to provide inter protection group routing.

Hardware Topology

From a network hardware and connectivity perspective both sites actually look very similar, examples for Site 1 and Site 2 are shown below in Figure 9-2 and Figure 9-3.

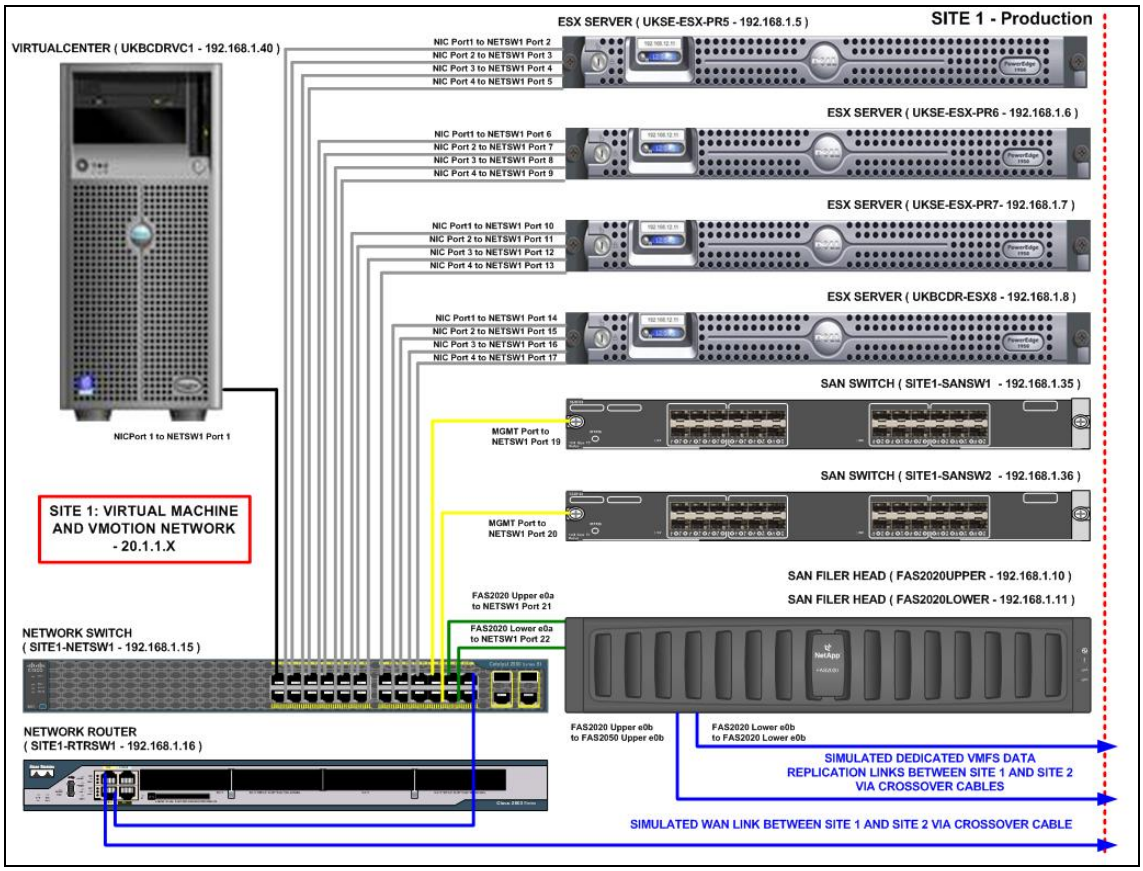


Figure 9.2 – Site 1 Network Connectivity

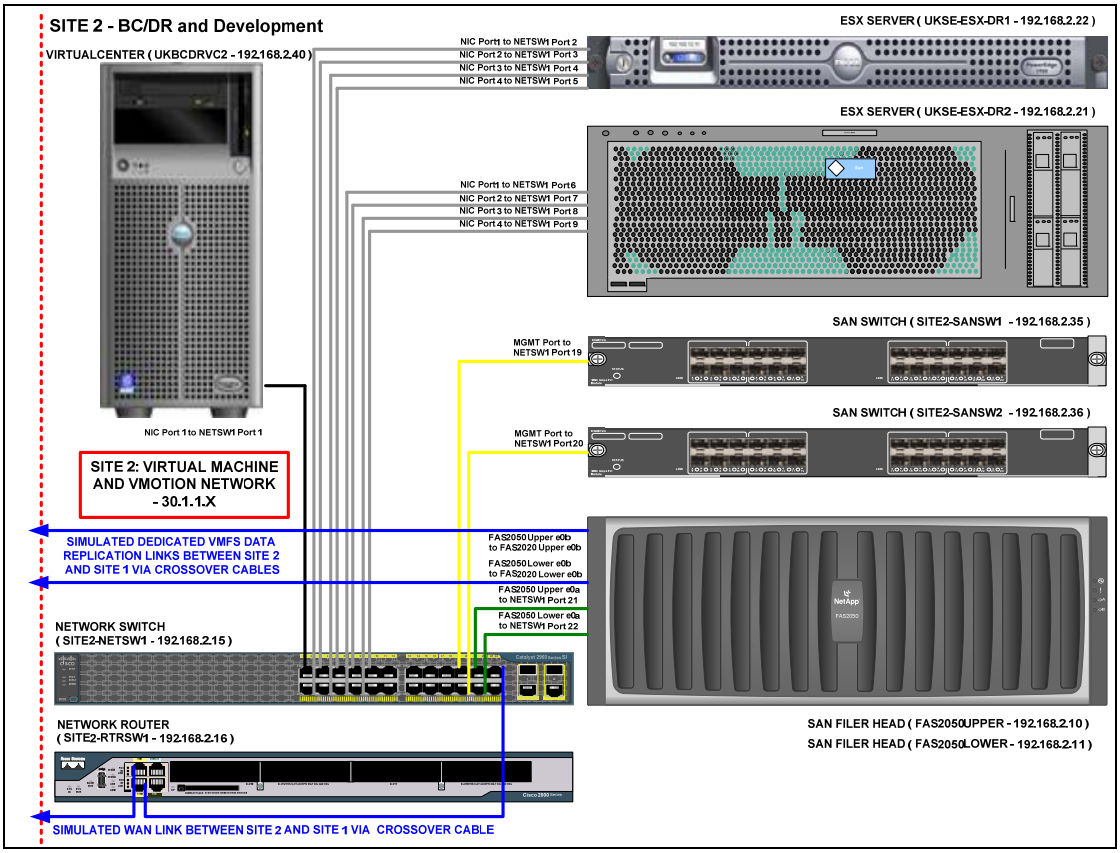


Figure 9.3 – Site 2 Network Connectivity

VMware ESX Configuration

As with other areas of VMware ESX configuration we tried to keep the basic design as simple as possible using default settings. Four portgroups were created on a single vSwitch and we used four of the six physical NICS present in the servers. Figure-9.4 depicts the configuration used, note that only the four physical used ports are show attached to the ESX host.

The four ports used consisted of the two onboard ports and two ports taken from a four port NIC card. Each portgroup was configured with two adapters. All portgroups were configured with an active / standby configuration except for the virtual machine portgroup which was configured active / active.

Also note that the adapters assigned to each portgroup are taken from either card (onboard and PCI) for redundancy.

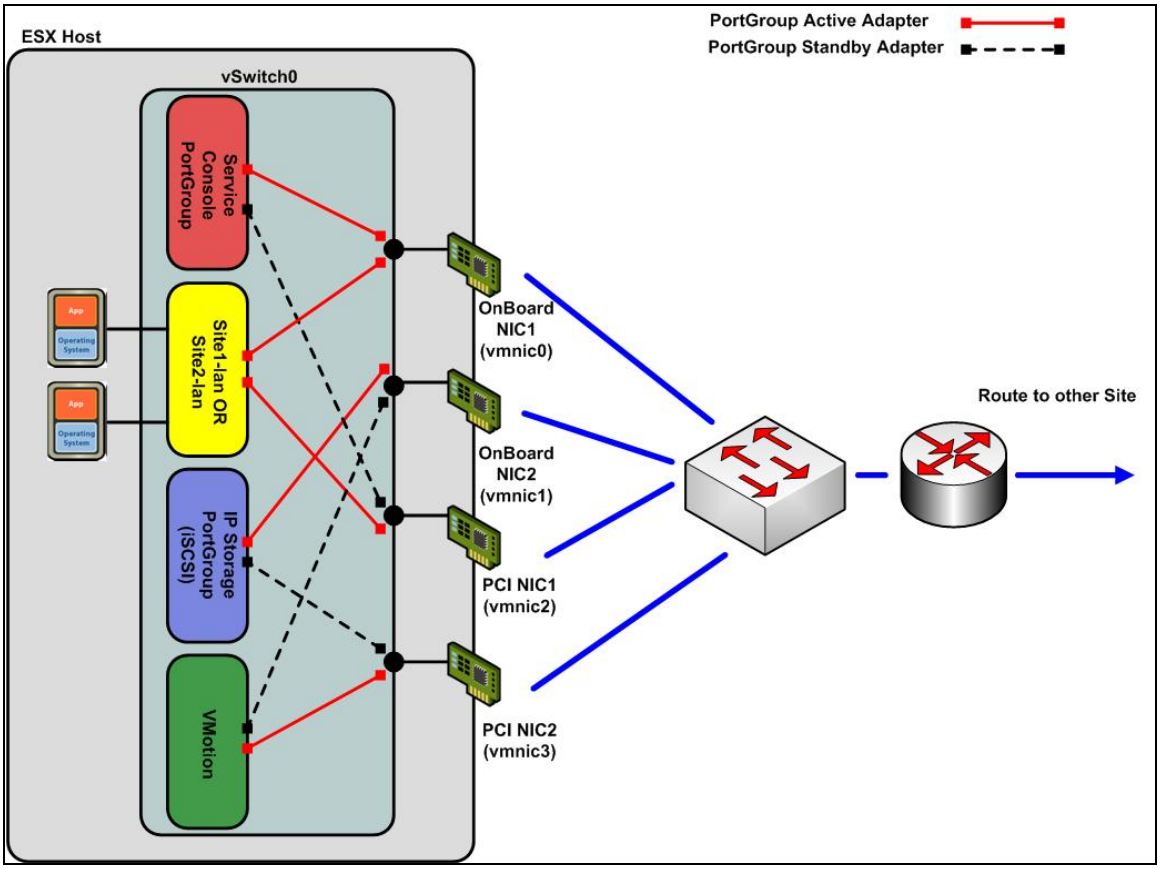


Figure 9.4 – Virtual Networking Design

Note that due to available hardware at time of writing a single physical switch was used. A more resilient design would have included at least two switches as per Figure-9.5. It would then be possible to spread the uplink connections across the two switches and physical cards.

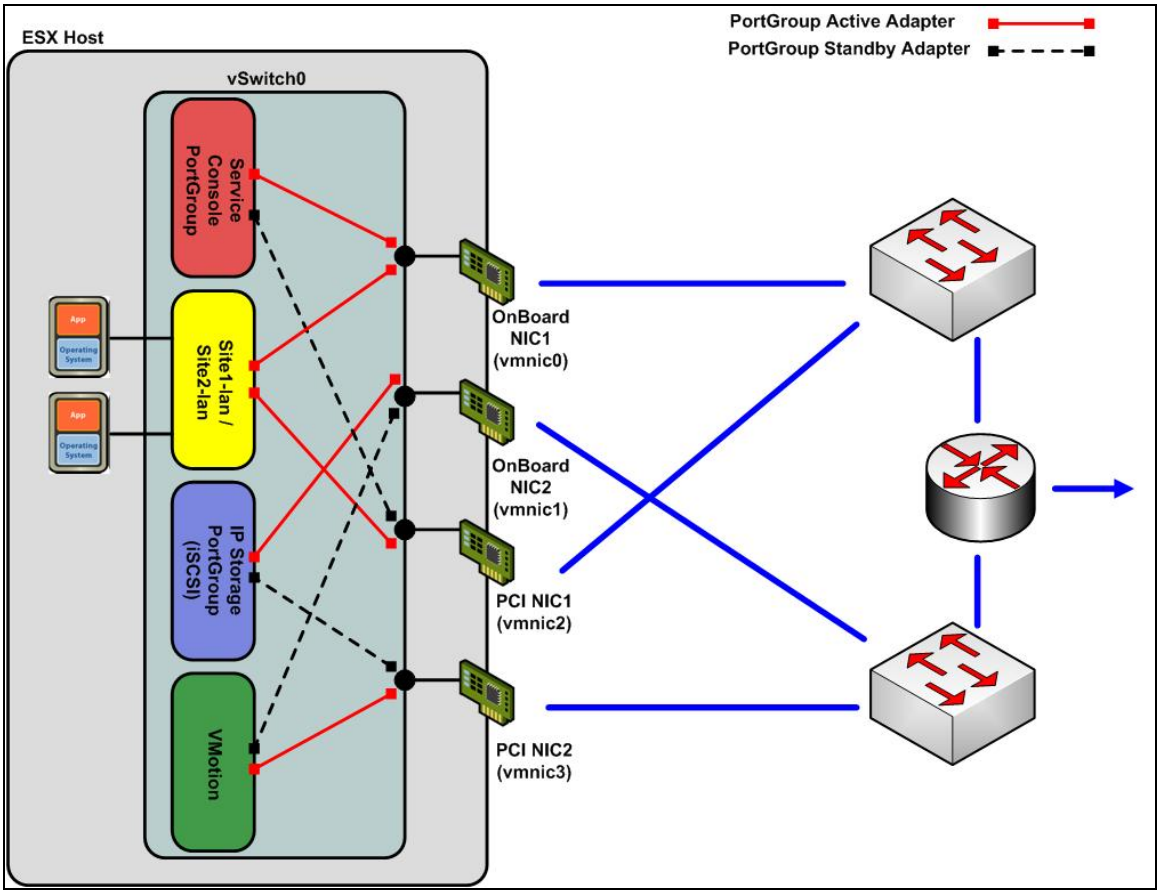


Figure 9.5 – Expanded Core Network Design Example

For each of the previous virtual network designs a single virtual switch was used for simplicity. Settings across the virtual switch were left as default. The properties for vSwitch0 are shown in Figure-9.6, note load balancing is set to "Port ID" as no etherchannel is configured in this environment:

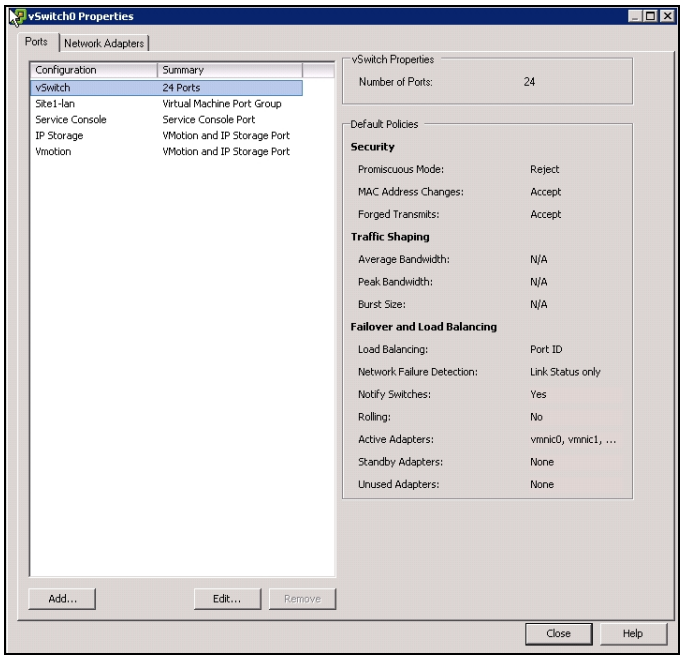


Figure 9.6 – vSwitch0 Properties

In Figure-9.7 you can see the IP address setups for the Storage network (NFS and iSCSI), VMotion and the Service Console interfaces.

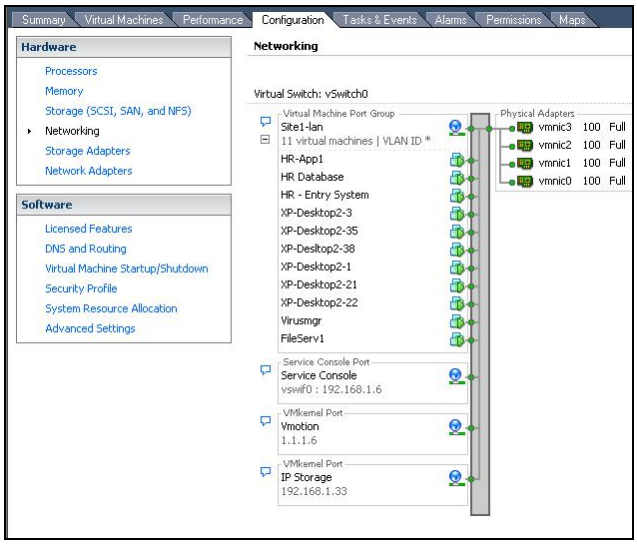


Figure 9.7 – Network Configuration

Drilling down into a couple of the PortGroup settings as examples. In Figure-9.8 we show the use of multiple adapters but in standby mode. Vmnic0 is the primary port for the Console but in the event of a failure it can also use Vmnic1. This does require a shared segment with any traffic on Vmnic1 – which

in this case is VMotion. In Figure-9.9 we show the main virtual machine traffic network configured onto a single subnet with dual paths provided by Vmnic3 and Vmnic4, we are not using the standby feature here.

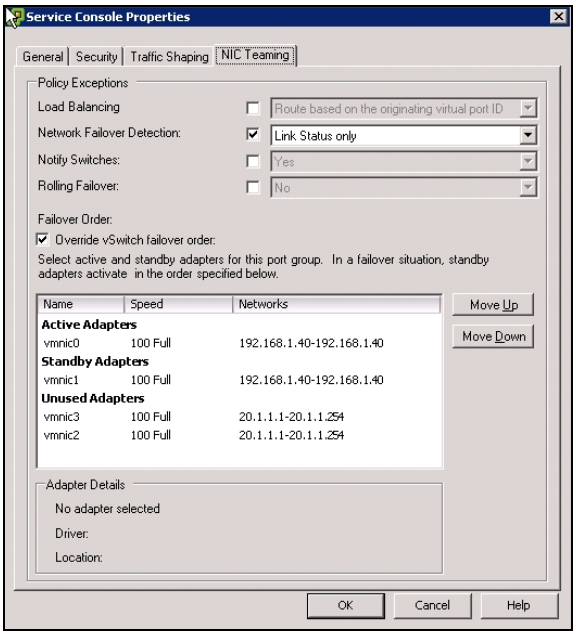


Figure 9.8 – Service Console NIC Teaming

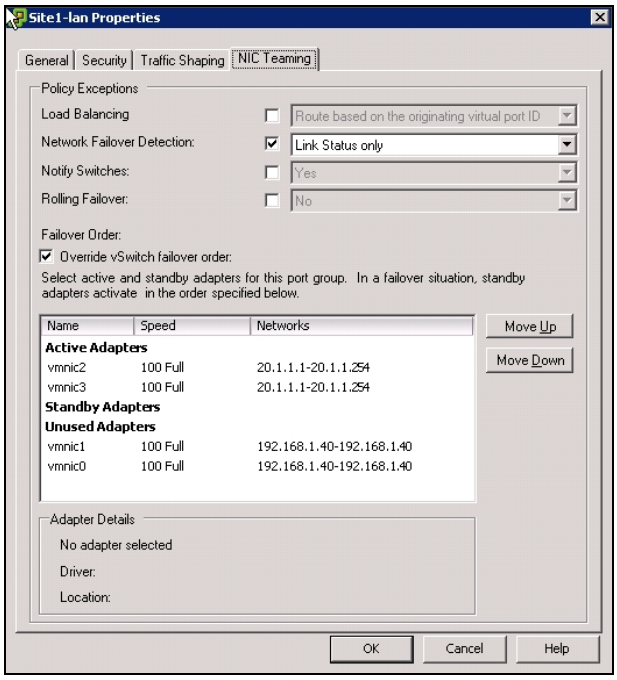


Figure 9.9 – Virtual Machine Network NIC Teaming

In summary we had four subnets at each site:

Site 1	
Production	20.1.1.X
Management	192.168.1.N
VMotion	1.1.1.N
IP Storage	192.168.1.N (Shared with Console)
Site 2	
Production	30.1.1.X
Management	192.168.2.N
VMotion	1.1.1.N

IP Storage	192.168.2.N (Shared with Console)

In Site 1 all virtual machines are connected to a Portgroup Site1-Lan. When a fail-over occurs the configuration of the virtual machine will be carried to the new site. In this design the LAN for production virtual machines is represented by a Portgroup Site 2-Lan. Therefore, the default behavior of a virtual machine will that it will not be able to see any networking connectivity initially at poweron. This is a design decision you will have to make and is easy to implement either way round; we felt the safety aspect of having the virtual machine isolated initially out weighed the convenience.

Many applications are sensitive to having duplicates un-expectedly turn up and additionally in our design we do have a router connecting both sites, so you may want to consider what happens if Site 1 is partially up and is in the process of being taken down whilst a failover is initiated. One such application in this category is Microsoft Active Directory and you should always be very careful about duplicates of this service. AD is protected using a replication technology of its own across both sites. A schematic of the AD setup used is shown in Figure-9.10.

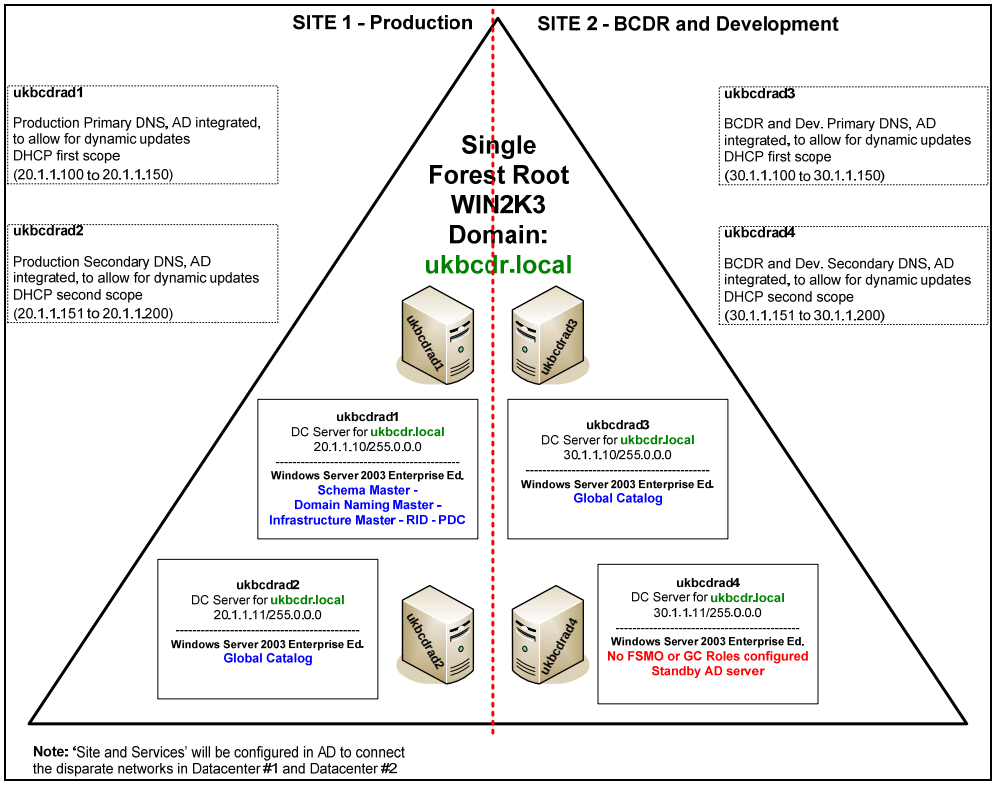


Figure 9.10 - Active Directory Implementation

Network Failover

Earlier in the chapter we discussed "Solution 1: Changing IP Addresses" and how there are many different ways to achieve this. When discussing this topic at your own site one important fact applies, whatever you typically do now will work for a virtual machine. When working on a virtual infrastructure your virtual machine's guest operating system is the same you previously used on physical servers so processes you previously had in place for handling IP address changes inside Windows for example will work in your virtual machines too.

For the virtual machines that are failed over, we need to change their IP addresses so that they will work on the 30.1.1.N network. To achieve this we used a very simple script executed by the guest operating system at startup. Here is the script:

```
strComputer = "."
strTargetAddress = "20.1.1.50"
arrIPAddressDR = Array("30.1.1.50")
arrSubnetMask = Array("255.255.255.0")
strGateway = Array("30.1.1.254")
strGatewayMetric = Array(1)

Set objWMIService = GetObject("winmgmts:\\" & strComputer &
"\root\cimv2")
Set colItems = objWMIService.ExecQuery _
    ("Select * From Win32_NetworkAdapterConfiguration Where IPEnabled =
True")

For Each objItem in colItems
    arrIPAddresses = objItem.IPAddress
    For Each strAddress in arrIPAddresses
        If strAddress = strTargetAddress Then
        Wscript.Echo "Found target IP Address of: " & strTargetAddress
        errEnableStatic = objItem.EnableStatic(arrIPAddressDR,
arrSubnetMask)
      errGateways =
objNetAdapter.SetGateways(strGateway,strGatewaymetric)
      Wscript.Echo "IP Address changed to DR Address of: 30.1.1.50"
      Else
      Wscript.Echo "No match for target address or already on DR
address"
      Wscript.Echo "Current Address is: " & strAddress
      Wscript.Echo "Non-DR IP Address is: " & strTargetAddress
      End If
    Next
Next
```

The script basically checks for the existence of a specific IP address and, if found, switches this address to the "failover" or DR IP address. Note than in many customer environments, IP addresses will not be switched and VLANs will be switched over/activated at the failover site.

If you are going to use a script such as the one previously shown you obviously do not want this script to fire when the virtual machine boots in its original "primary" site location, Site 1 in our environment. So how do we programmatically control when the scripts fire within the virtual machines?

One simple way to trigger a script inside a virtual machine on startup is to utilize VMware Tools PowerOn scripts. When VMware Tools is installed within your virtual machine you will notice a small VMware tools icon appears in the windows task bar. Double-clicking this icon will popup the VMware Tools options box as shown in Figure 9.11:

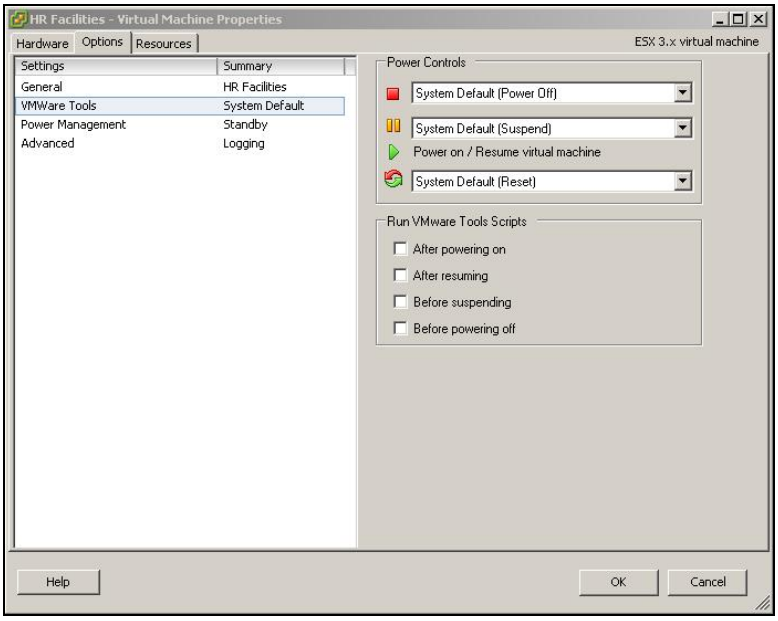


Figure 9.11 – VMware Tools Options

Highlighting the "Options" tab as shown in Figure 9.11 will display the "Run VMware Tools Scripts" checkboxes. By default these are unchecked. Note the first checkbox is to run a script "After Powering On". This is the stage we use to trigger our IP address change script.

The next question is "Ok so I can see how I can trigger a script but todo this manually for EVERY virtual machine is going to take sometime to tick that little checkbox?". This is where we can again make use of the VIPerlToolKit.

We can create a script that will toggle this checkbox on / off for ALL virtual machines. The process for a failover would be:

1. Failover Storage
2. Activate virtual machine with their NICs disconnected

3. Run perl script to "tick" the VMware tools "After Powering On.." checkbox
4. Reboot virtual machines with NICs connected
5. Script will trigger and re-ip all virtual machines
6. Run perl script again to un-tick the "After Powering On…" checkbox so that any subsequent reboots at the failover site do not recheck the IP address settings.

Refer to **Appendix B: VMware Tools Script** for view the actual code used in this script.

A sample command link invocation of the tools script is as follows:

```
perl -w ToolsConfig.pl --server 192.168.1.40 --url
https://192.168.1.40/sdk --username administrator --password vmware –
folder "FailOverVMs" --option afterPowerOn --value yes
```

The IP address (192.168.1.40) in the above corresponds to the VirtualCenter IP address for the VirtualCenter server located at the failover site (Site 2) where we are failing over to.

Other Considerations

There are a number of different ways to achieve network address failover within your virtual machines. There is no one correct method, each environment, network and datacenter will be different. The most important factor is to decide upon a method that is reliable, simple and works for you.

Another example? Add a second NIC card to the virtual machine and pre-configure it for the DR site. One issue with this approach is you only have 4 vNICS currently to configure so if your guest OS requires more than two subnets this approach cannot be taken.

On failback you will have to do the reverse process or have the script autodetect the prevailing network conditions and adjust itself accordingly.

You should allow for enough ports on the virtual switches to be available. Each switch can have up to 1016 ports but by default they are set to 56. While adding more ports is easy, it will require a reboot of the VMware ESX host and will delay the power-on process.

Chapter 10. Storage Connectivity

The purpose of this chapter is to describe the storage connectivity architecture deployed and described in this document that has been configured across both Site 1 and Site 2. This chapter will explain the layout of the storage fabrics and networks.

The overall goal when configuring the storage connectivity was to keep the design simple whilst also providing the ability to illustrate some key points to bear in mind when building your BCDR solution on top of VMware virtual infrastructure.

The design employed contains a mix of SAN fabric and IP Storage (NFS).

This chapter will not discuss VMware virtual infrastructure SAN deployment options, or other similar administration and managements concepts in any detail as this is beyond the scope of this book. For further detail on these concepts please refer to the VMware virtual infrastructure documentation and technical papers such as the VMware SAN System Design and Deployment Guide[6].

Storage Connectivity Topology

At this stage it is important to have a clear understanding of the overall storage connectivity without adding in the VirtualCenter datastore concepts at this point. We will cover the logical design more in Chapter 11.

Figure-10.1 below illustrates the storage connectivity topology utilized across our Site 1 and Site 2 datacenters. Within this figure we represent the local and replicated storage as generic entities only.

[6] http://www.vmware.com/pdf/vi3_san_design_deploy.pdf

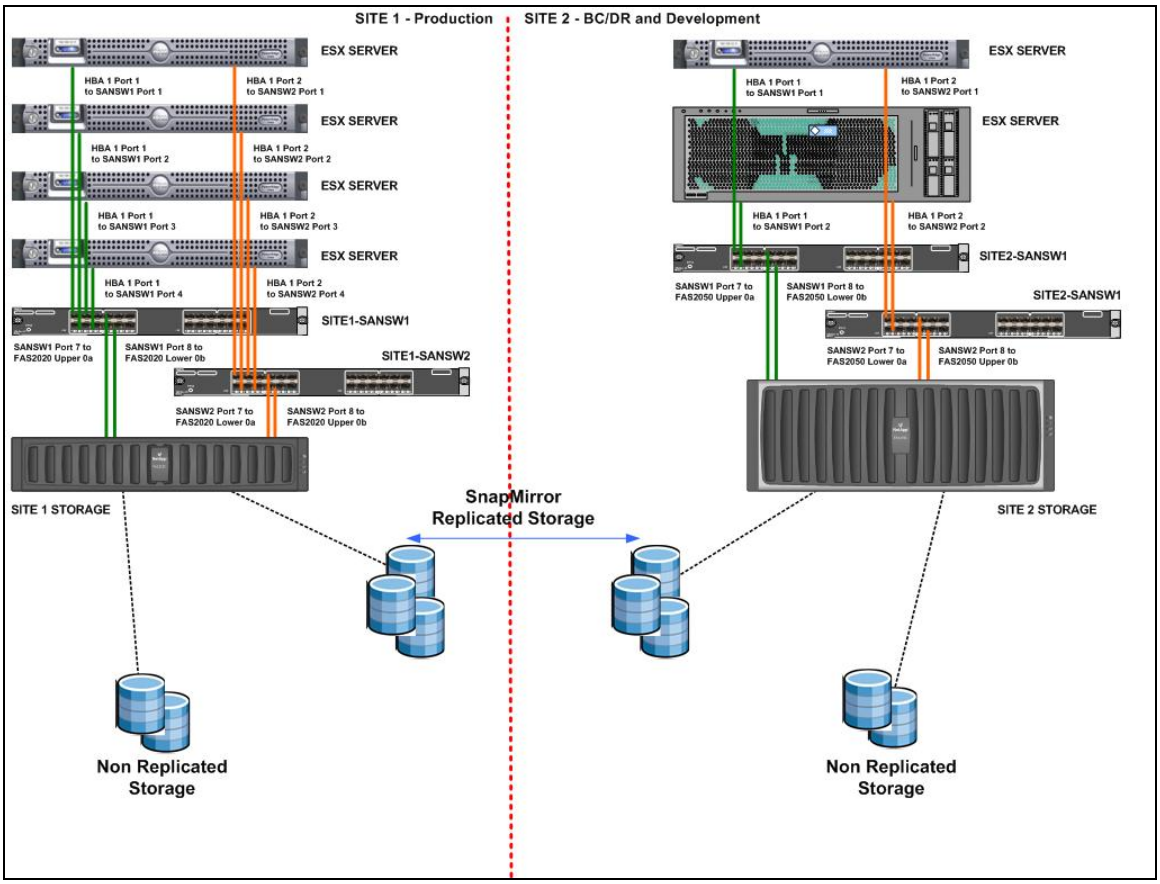


Figure 10.1 – Site 1 and Site 2 Physical Storage Layout

Starting at a high level two Cisco DS-C9124 Fabric switches were deployed at both Site 1 and Site 2 as shown in Figure-10.1 above. Two switches were used resiliency. Each switch operated in its own independent fabric with its own zoning configuration (i.e there was no interconnect cable between switches co-located at either site).

Host Components

Very simply the host components in a VMware Infrastructure design refer to the Host Bus Adapters (HBAs) installed in the VMware ESX hosts and also the HBA drivers that enable ESX to communicate to the SAN fabric via the HBAs.

Each VMware ESX host was installed with a single dual port HBA. The HBAs installed were Emulex LPe11002-M4 Dual Channel 4Gb/s Fibre Channel PCI Express HBA. All HBA Bios settings were left at the default settings for the purposes of this configuration.

Figure 10.2 – Emulex LPe11002-M4

Each port of the Emulex HBA was connected to a separate SAN switch (fabric) for redundancy.

For each site we can see this in more detail in the figures contained in the next section "Fabric Components

Fabric Components

Site 1 and Site 2 each contained two Cisco DS-C9124 SAN Fabric switches to provide resiliency across the SAN infrastructure. For the design to be realistic it must be fault tolerant during fabric disruptions, FC switch failures, or other conditions such as RSCN storms.

Virtual machines are protected from SAN errors by SCSI emulation. VMware ESX is well suited for error recovery, and guards against I/O subsystem malfunctions that may impact the underlying applications. VMware ESX has a built-in multipathing algorithm that automatically detects an error condition and chooses an alternative path to continue servicing data or application requests.

Connectivity between the SAN switches, VMware ESX hosts and Storage components is shown in the following tables and figures. For clarity we will separate out Site 1 and Site 2 and also each switch within each site.

Each Site contained two SAN switches and two SAN fabrics. At each site these are defined as:

- Site 1:
    - Site1-sansw1 (192.168.1.35)
    - Site2-sansw2 (192.168.1.36)
- Site 2:
    - Site2-sansw1 (192.168.2.35)
    - Site2-sansw2 (192.168.2.36)

IP Addresses stated for each SAN switch represent the management connection for each SAN switch. Management subnets were 192.168.1.x for Site 1 and 192.168.2.x for Site 2.

Management addresses for VMware ESX hosts, NetApp Storage and Cisco Network hardware were also created on these subnets. IP address ranges for these additional components are detailed elsewhere in this guide.

Site 1 Configuration

The SAN/fabric configuration utilized in Site 1 is represented by Figure-10.3 below:

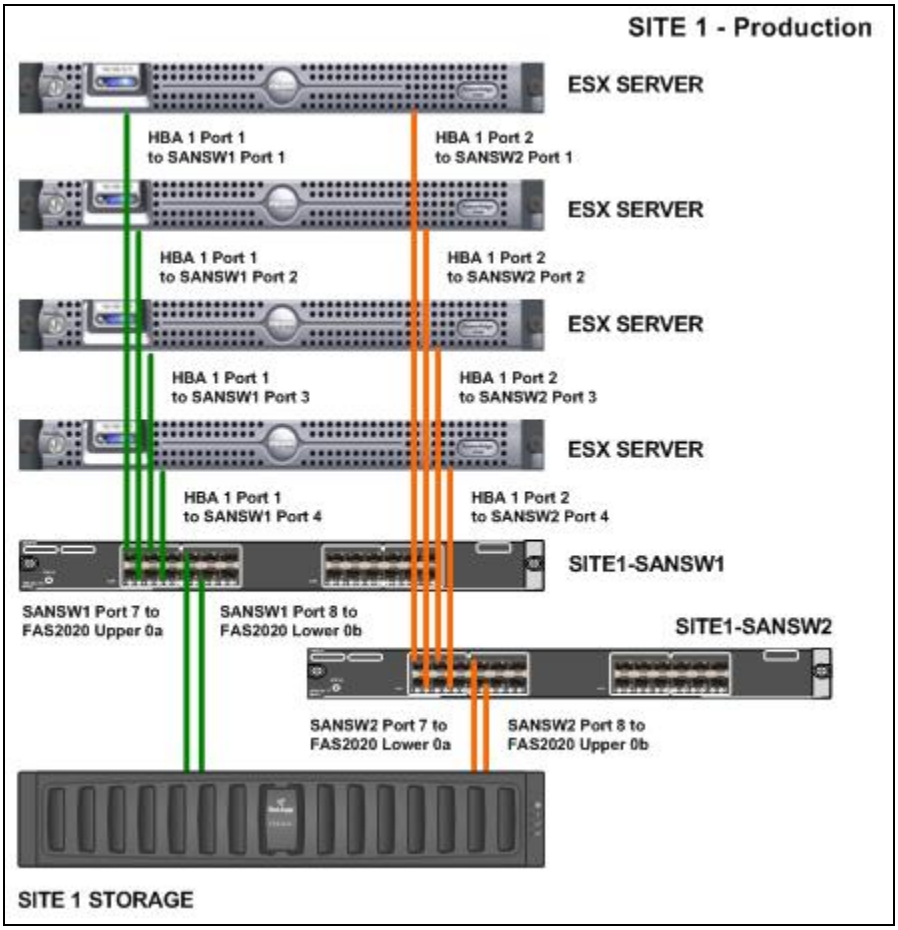


Figure 10.3 - Site 1 SAN Connections

The figures above show how each port in the four VMware ESX HBAs are connected to a distinct SAN switch and also how each port in the Storage HBAs is connected to their respective SAN switches. Recall each ESX host in our solution contains a single dual port 4Gb/s Emulex HBA.

The two Cisco SAN switches located at Site 1 are utilized as per Tables 1 and 2 and Figure-10.4 and Figure-10.5.

Site 1 – "site1-sansw1" (192.168.1.35)

Zone	Type	Switch Interface	Name	WWN
Zone1	WWN	site1-sansw1 fc1/4	Emulex	10:00:00:00:c9:6a:fb:66
Zone1	WWN	site1-sansw1 fc1/2	Emulex	10:00:00:00:c9:6c:50:86
Zone1	WWN	site1-sansw1 fc1/3	Emulex	10:00:00:00:c9:6c:50:ba
Zone1	WWN	site1-sansw1 fc1/1	Emulex	10:00:00:00:c9:6c:5a:f4
Zone1	WWN	site1-sansw1 fc1/7	NetApp	50:0a:09:81:98:ec:39:a1
Zone1	WWN	site1-sansw1 fc1/8	NetApp	50:0a:09:82:88:ec:39:a1

Table 1 – "site1-sansw1" connections

Figure 10.4 - Site 1 SAN Switch 1

Site 1 – "site1-sansw2" (192.168.1.36)

Zone	Type	Switch Interface	Name	WWN
Zone1	WWN	site1-sansw2 fc1/4	Emulex	10:00:00:00:c9:6a:fb:67
Zone1	WWN	site1-sansw2 fc1/2	Emulex	10:00:00:00:c9:6c:50:87
Zone1	WWN	site1-sansw2 fc1/3	Emulex	10:00:00:00:c9:6c:50:bb
Zone1	WWN	site1-sansw2 fc1/1	Emulex	10:00:00:00:c9:6c:5a:f5
Zone1	WWN	site1-sansw2 fc1/8	NetApp	50:0a:09:81:88:ec:39:a1
Zone1	WWN	site1-sansw2 fc1/7	NetApp	50:0a:09:82:98:ec:39:a1

Table 2 – "site1-sansw2" connections

Figure 10.5 - Site 1 SAN Switch 2

Site 2 Configuration

The SAN/fabric configuration utilized in Site 2 is represented by the Figure 10.6.

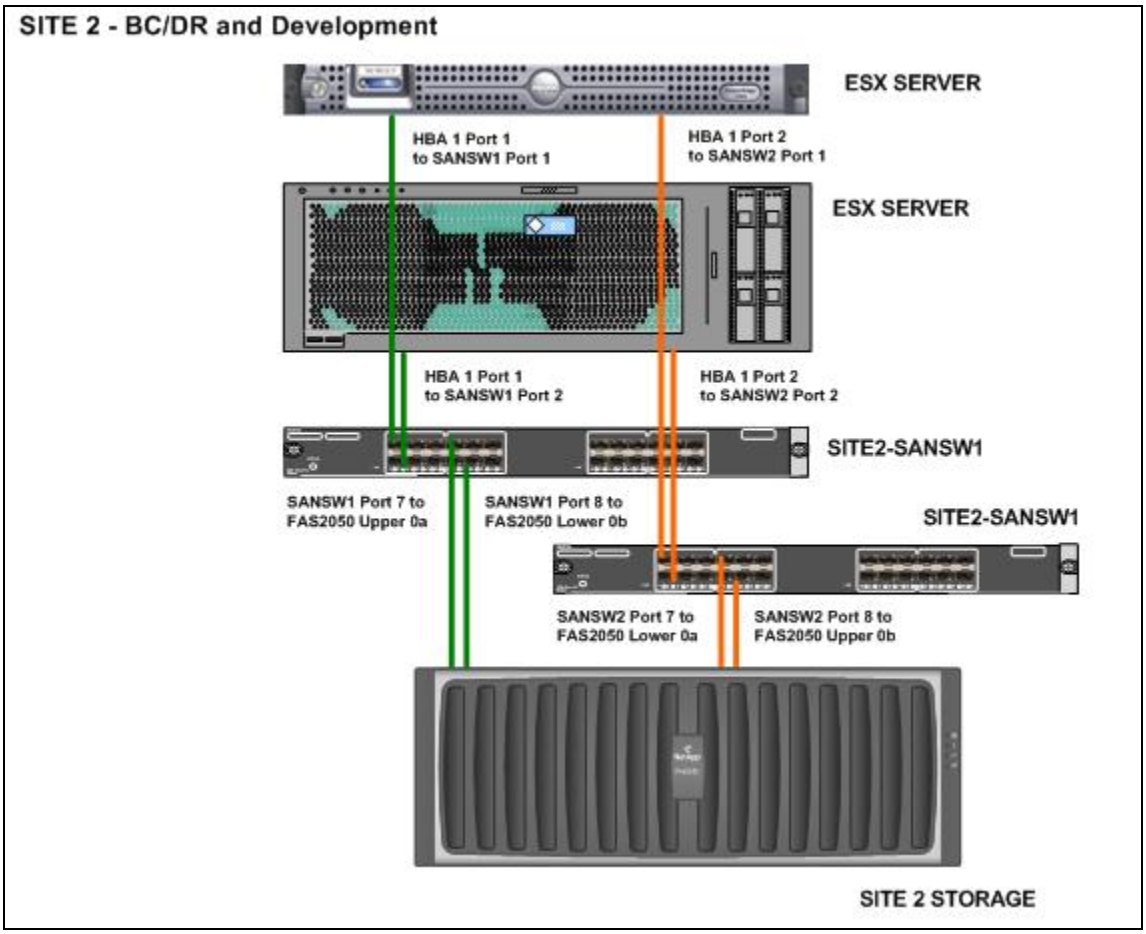


Figure 10.6 - Site 2 SAN Connections

The Figure-10.6 shows how each port in the two VMware ESX HBAs are connected to a distinct SAN switch and also how each port in the Storage HBAs is connected to their respective SAN switches. Recall each ESX host in our solution contains a single dual port 4Gb/s Emulex HBA.

The two Cisco SAN switches located at Site 2 are utilized as per the Table 3 and Table 4 and Figure-10.7 and Figure-10.8.

Site 2 – "site2-sansw1" (192.168.2.35)

Zone	Type	Switch Interface	Name	WWN
Zone1	WWN	site2-sansw1	Emulex	10:00:00:00:c9:50:33:0a

		fc1/1		
Zone1	WWN	site2-sansw1 fc1/2	Emulex	10:00:00:00:c9:6c:50:7e
Zone1	WWN	site2-sansw1 fc1/7	NetApp	50:0a:09:81:88:6c:37:dd
Zone1	WWN	site2-sansw1 fc1/8	NetApp	50:0a:09:82:98:6c:37:dd

Table 3 – "site2-sansw1" connections

Figure 10.7 - Site 2 SAN Switch 1

Site 2 – "site2-sansw2" (192.168.2.36)

Zone	Type	Switch Interface	Name	WWN
Zone1	WWN	site2-sansw2 fc1/1	Emulex	10:00:00:00:c9:50:33:09
Zone1	WWN	site2-sansw2 fc1/2	Emulex	10:00:00:00:c9:6c:50:7f
Zone1	WWN	site2-sansw2 fc1/7	NetApp	50:0a:09:81:98:6c:37:dd
Zone1	WWN	site2-sansw2	NetApp	50:0a:09:82:88:6c:37:dd

		fc1/8		

Table 4 – "site2-sansw2" connections

Figure 10.8 - Site 2 SAN Switch 2

Storage Components

For the purposes of this document the storage components utilized were supplied by NetApp. Two different models were used. For Site 1 the storage component was a FAS2020, for Site 2 the storage component was a FAS2050.

The choice of moderately specified storage components was deliberate and is designed to illustrate the flexibility of creating a flexible business continuity ready environment with VMware Infrastructure.

Fabric-attached storage (FAS) systems offer simultaneous support for Fibre Channel SAN, IP SAN, and network-attached storage (NAS) from the same unified, scalable storage architecture.

FAS2020: The FAS2020 implements 3.6TB of raw capacity into a 2U enclosure with optional external expansion for an additional 21TB of raw capacity.

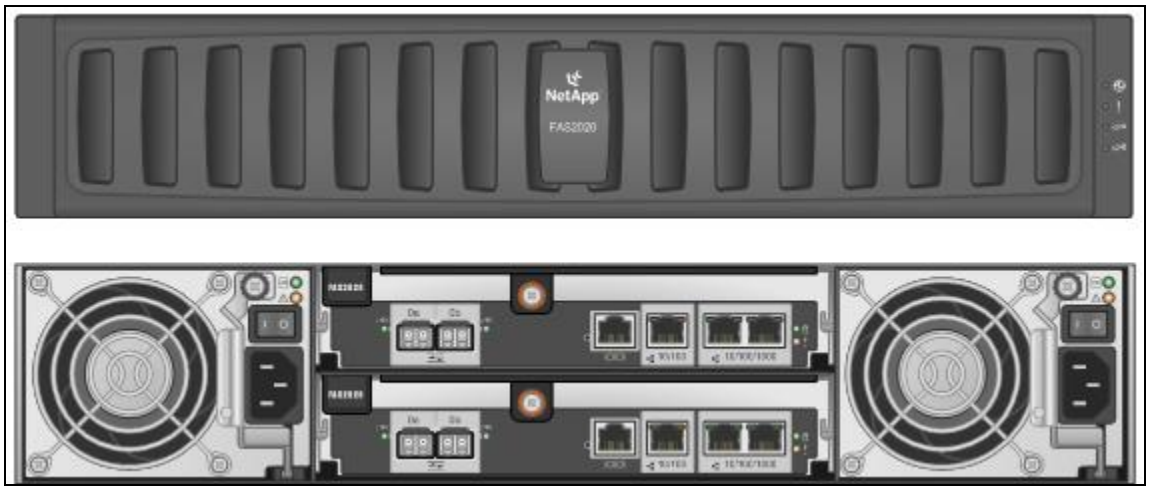


Figure 10.9 - FAS2020 (front / rear)

The FAS2020 controllers each provide dual gigabit ethernet ports, and dual 4Gb/sec Fibre Channel ports. Systems are typically deployed in dual controller configurations.

FAS2050: The FAS2050 provides 6TB raw capacity in a 4U enclosure, scaling to 69TB raw capacity with optional external expansion drives.

Figure 10.10 - FAS2050 (front/rear)

The FAS2050 controllers each provide dual gigabit Ethernet ports, dual 4Gb/s Fibre Channel ports, and an expansion slot which can be used for a variety of expansion cards, including Fibre Channel adapters for SAN, disk or tape connectivity, iSCSI adapters, and Ultra320 SCSI adapters. Systems are typically deployed in dual controller configurations.

For more info see http://www.netapp.com/us/products/storage-systems/

Storage Multipathing and Path Failover

All externally configured LUNs hosted by the NetApp storage arrays are accessible by the ESX host via two fibre channel paths through two separate switch fabrics. Each VMware ESX has a dual port 4Gb/s Emulex fibre channel HBA installed.

Native VMware ESX Multipathing is used to provide the multipathing function.

VMware ESX has two possible multipathing policies: Most Recently Used (MRU) and Fixed. The recommended policy for use with VMware ESX is determined automatically by the type of Storage Array being attached. By default, the multipathing policy is set to Fixed.

The NetApp arrays support both MRU and Fixed policies. When using a MRU policy a failed path is **not** automatically recovered to fully fault tolerant status, remaining on the path it failed over to. After any array failure operational procedures must be in place to resolve this post-failure situation. For this reason, NetApp recommends using the Fixed multipathing policy wherever possible. For further information on how to configure ESX multipathing with NetApp storage, please see: http://media.netapp.com/documents/tr_3428.pdf

When dual controller systems are deployed in Fibre Channel configurations, Data ONTAP manages how the Fibre Channel ports behave during failover.

By default, in a dual controller or active/active configuration, each of the controllers share a single global WWNN. This ensures that the controllers appear as one single system on the fabric.

For example, in our configuration we configured LUNs on each controller on the FAS2020.

ESX servers in see those LUNs presented via 4 paths (as each controller has 2 x 4Gb connections to the SAN), see the "Paths – total" value in the Figure-10.11 (screenshot taken by selecting an ESX host in VirtualCenter and then selecting the configuration tab):

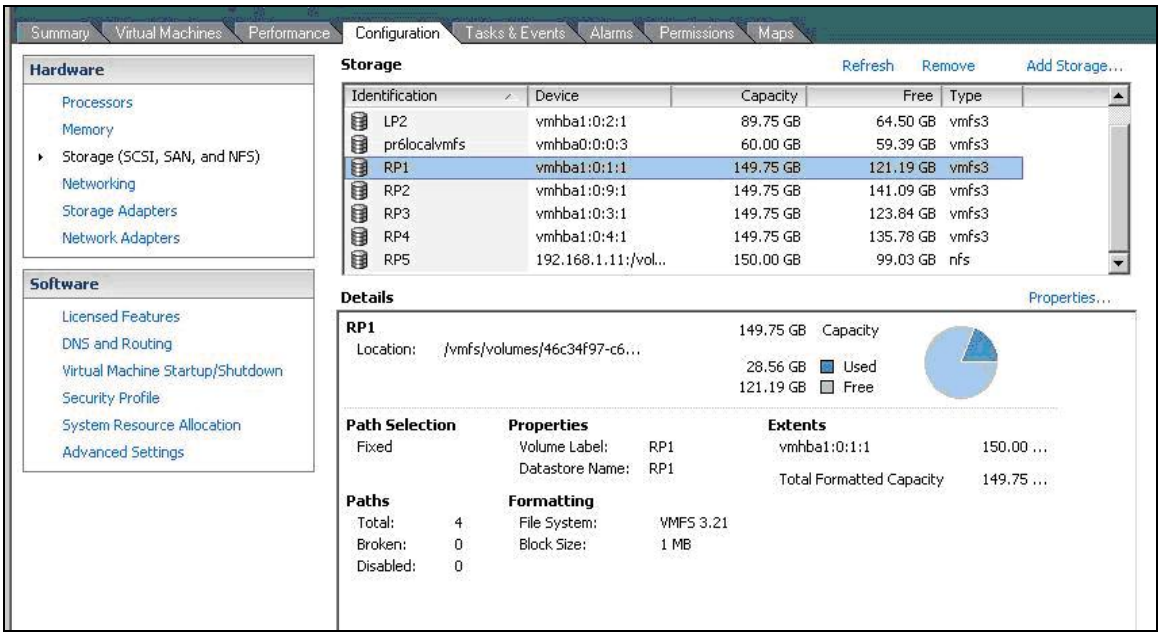


Figure 10.11 - LUN Path Example

Should one controller fail, we would see 2 paths appear as "broken". Access to the LUN would continue via the remaining paths on the surviving controller.

For a more detailed review of the inbuilt resiliency and multipathing capabilities of VMware ESX please refer to the VMware VMware Infrastructure documentation and technical papers available on vmware.com.

SAN Zoning

Zoning provides access control in the SAN topology; it defines which HBAs can connect to which Storage Processors (NetApp's). Specific information on considerations for SAN Zoning and best practices with VMware Infrastructure can be found in the SAN System Design and Deployment Guide: http://www.vmware.com/pdf/vi3_san_design_deploy.pdf

In our environment as has been detailed each site contained two SAN switches and each individual switch represented one SAN fabric. To keep the design simple, each fabric contained s single zone. Zone members for each switch were listed in Tables1-4 within the "Fabric Components" section of this chapter.

Within each fabric a single Zone/Zoneset "Zone1" was activated.

It must be stated that for the purposes of this document the SAN fabric design has been kept deliberately straightforward as the purpose of the document is not to educate on SAN best practice.

Within a more complex SAN environment, zoning defines and configures the necessary security and access rights for the entire SAN. Zones are typically, created for each group of servers that access a shared group of storage devices and volumes.

The most common uses of zoning are:

- Zoning for security and isolation — Zones defined independently within the SAN to remove interference with activity going on in other zones.
- Zoning for shared services — For example backups.
- Multiple storage arrays — Using separate zones, each storage array is managed separately from the others. Removes concern for access conflicts between servers.

NOTE: For other zoning best practices, check with the specific vendor of the storage

array you plan to use.

Network File System (NFS) Connectivity

In our environment we have created a single NFS datastore named RP5 that is replicated between Site 1 and Site 2 using Snapmirror. When utilizing IP based storage in a VMware environment best practice is to separate the IP storage traffic, NFS in our case, from your other IP based network traffic by implementing a separate network or VLAN. You will typical already have networks defined for your virtual machines and possibly for VMotion so for your IP based storage you should create an additional network.

In virtual networking terms within your VMware environment this will typically be held with a specific portgroup for your IP based storage. Creating NFS datastores, using/creating VMKernel Portgroups, and NFS settings are extensively discussed within the standard VMware documentation so we will not repeat that information here. For further information on NFS datastore configuration please refer to the VMware ESX Configuration Guide relevant to your ESX version (3.x or 3.5.x).

Very simply to enable NFS connectivity we need to define a VMkernel portgroup for our IP based storage connections. This VMkernel portgroup type requires an IP address. Figure-10.12 is an example from one of our Site 1 ESX hosts:

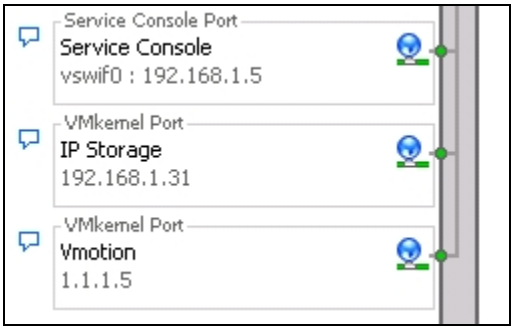


Figure 10.12 - Portgroups

The volume on the storage exported as an NFS mount in our environment is labeled at the storage layer as volume RG2FV2. This volume is then exported to each of the vmkernel ip addresses, used for IP storage connectivity, within our ESX hosts located at that site. Do not export your NFS volumes to your "Service Console" portgroup ip address, as the NFS client utilized for virtual machine storage is integrated into the VMKernel and does not run within the Service Console.

If we view the "Manage NFS Exports" view within NetApp FilerView we can review the configuration of the export, see Figure-10.13:

☐	/vol/RG2FV3	Read-Write Access (All Hosts) Root Access (192.168.1.33,192.168.1.32,192.168.1.31,192.168.1.30) Security (sys)

Figure 10.13 - NFS Export Options

On your ESX hosts you also need to open the firewall ports for the "NFS Client" group as shown in Figure-10.14 "Configuration > Security Profile" output:

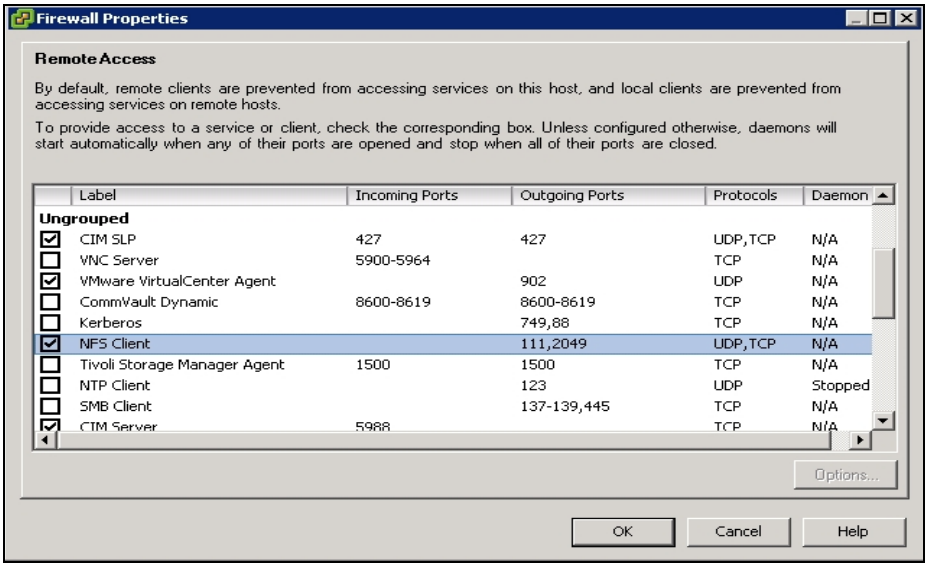


Figure 10.14 - ESX Firewall Properties

Once our NFS datastore RP5 was export we can see it in our datastore inventory and browse it as we do any other datastore, Figure-10.15:

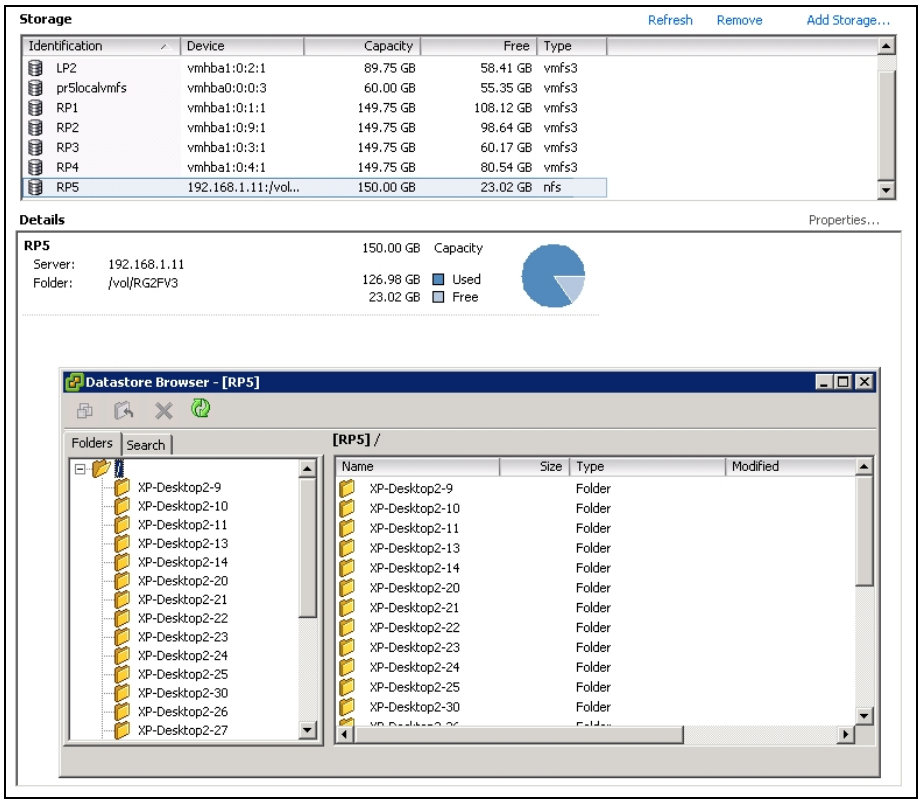


Figure 10.15 - Datastore Inventory / Browse

Storage Replication Connectivity

For the purposes of this book we have utilized a NetApp FAS2020 at our Site 1 datacenter and a NetApp FAS2050 at out Site 2 datacenter.

The overview figure below depicts the storage layout of the two sites. In this figure we can see a high level representation of the connectivity between the two storage systems. Depending on your choice of storage the connection will typically be across Fibre Channel or Ethernet based networks. In our case we have used Ethernet connectivity, as the FAS2000 series at time of writing does not support the use of Fibre Channel as the transport for replication traffic. Replication over Fibre Channel is supported on many other FAS models.

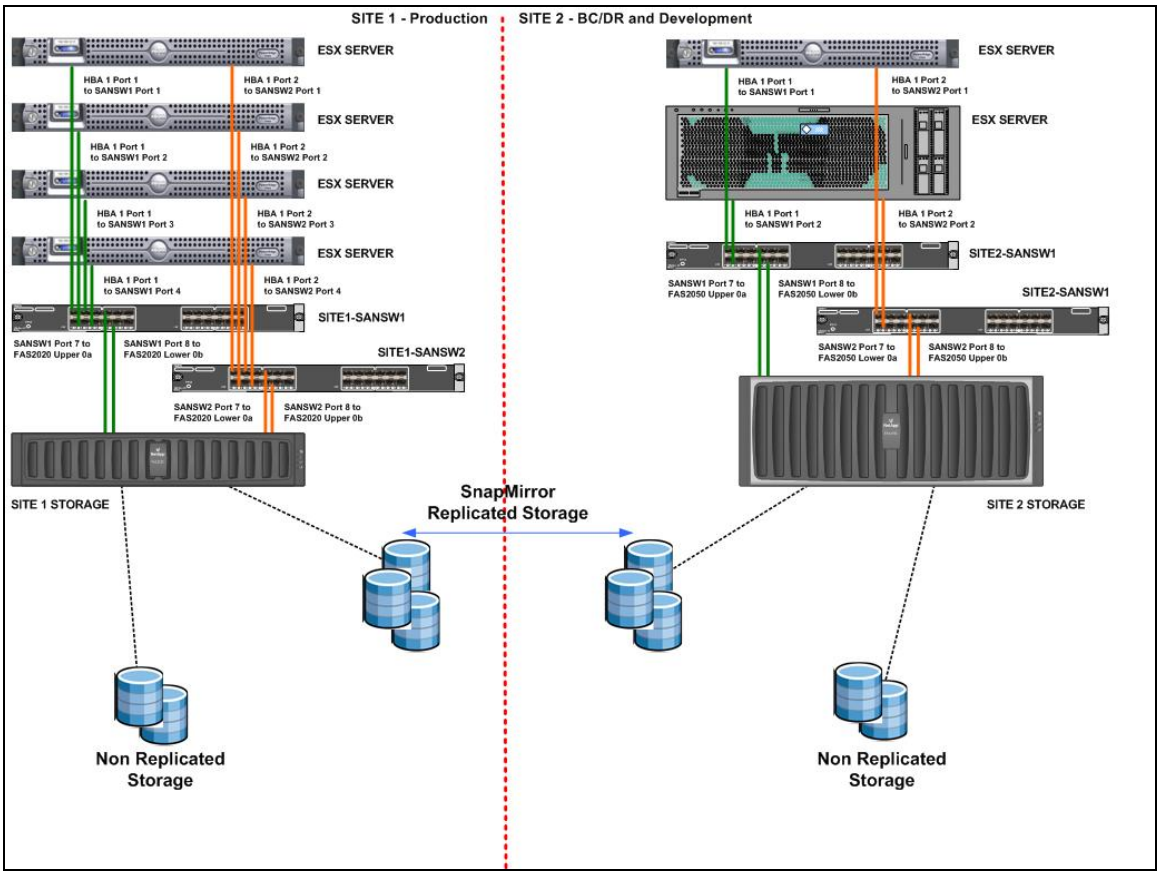


Figure 10.16 - Snapmirror Connectivity

Each FAS system is divided up into two controllers, commonly known as "heads" and thereafter upper and lower heads. Our naming convention simply takes advantage of the fact that we have different models of array and therefore we use this to simply distinguish which controller we are referring to.

This gives us at Site 1:

- fas2020Upper (192.168.1.10 management ipaddress)
- fas2020Lower (192.168.1.11 management ipaddress)

And at Site 2:

- fas2050Upper (192.168.2.10 management ipaddress)
- fas2050Lower (192.168.2.11 management ipaddress)

To provide connectivity between the two storage systems, for SnapMirror Replication, we implemented a dedicated private network using a single, small network switch and a single subnet 11.11.11.x.

To establish connectivity between each controller using our basic network, we configured one of the GigE interfaces in each controller as a dedicated replication interface on the 11.11.11.x subnet.

Although SnapMirror can share interfaces with other traffic, in certain configurations may be desirable to minimize the possibility of network contention. For this reason, it is recommended to use a private network for SnapMirror wherever possible.

For further information on SnapMirror best practices, read the NetApp SnapMirror Best Practices Guide[7], available from the NetApp Web site. please see http://media.netapp.com/documents/tr_3446.pdf

Whilst the single switch configuration used here was fit for purpose, in terms of allowing us to demonstrate BCDR across two storage systems whilst providing a level of assured bandwidth for replication at the storage layer, it does not represent a recommended networking configuration, which should typically not include such a single point of failure.

To illustrate the configuration we will take a look at each sites array in turn.

Site 1

Located in Site 1 we have the FAS2020 array which is split as previously described into two controllers, those being fas2020Upper and fas2020Lower. For each controller we allocated a replication IP address to allow the counterpart controller it would be replicating to at Site 2 to identify it correctly on the network. Our replication IP address for Site 1 were:

- fas2020Upper (11.11.11.1 replication IP address)
- fas2020Lower (11.11.11.3 replication IP address)

Using FilerView management interface we can see the interface selection for each controller or head, for the "upper" controller fas2020upper Figure-10.17 shows:

[7] http://media.netapp.com/documents/tr_3446.pdf

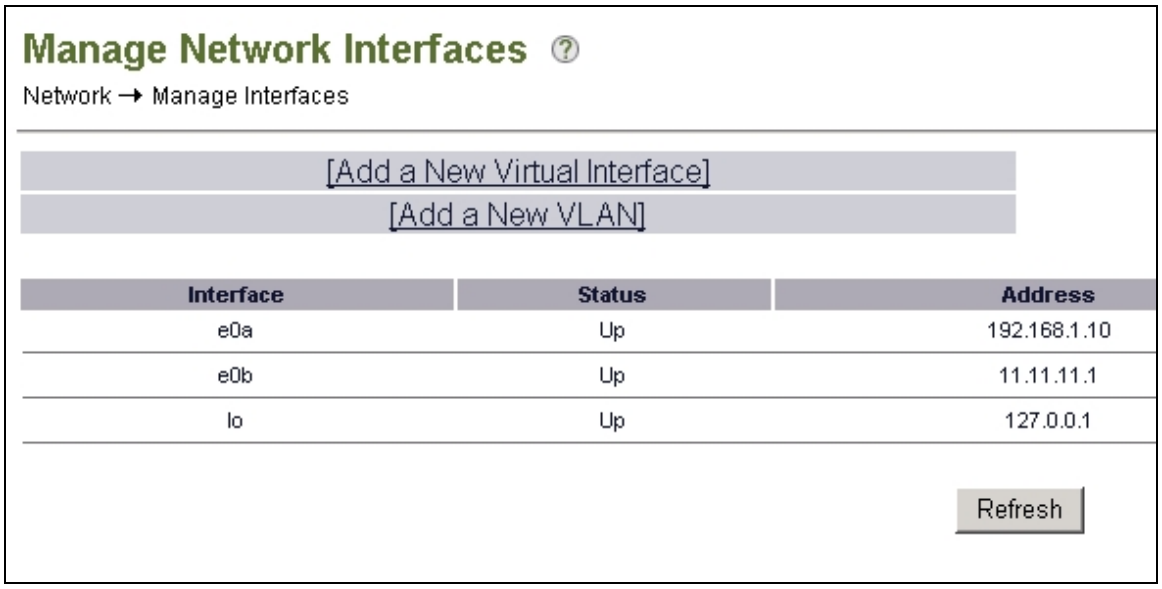


Figure 10.17 - fas2020upper Network Interfaces

Figure-10.18 for fas2020lower shows:

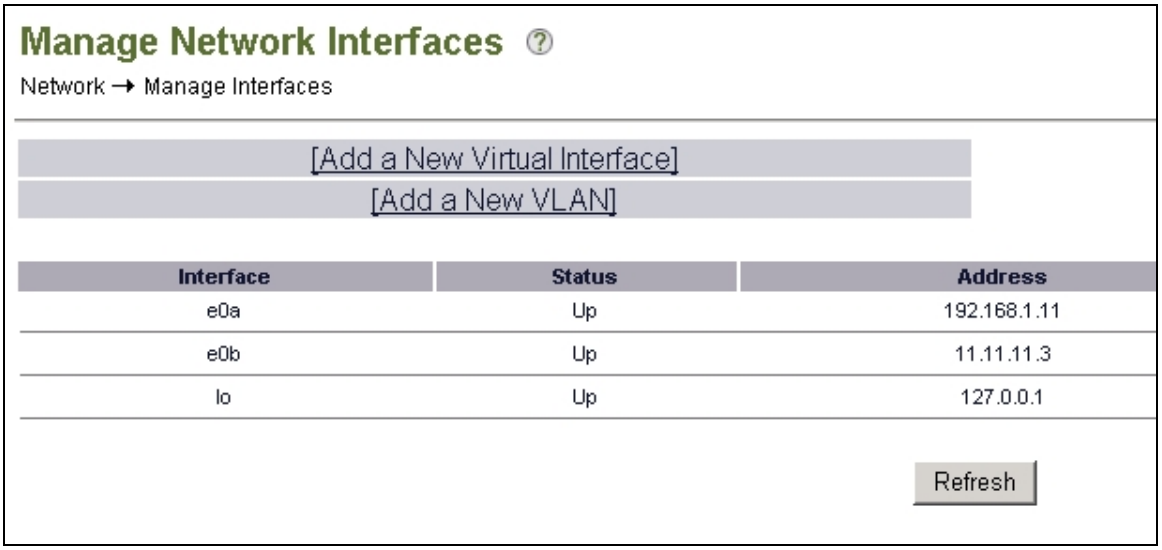


Figure 10.18 - fas2020lower Network Interfaces

Taking a look at the rear view of the FAS2020 itself, Figure-10.19, we can map this interfaces to show their location and intended destinations:

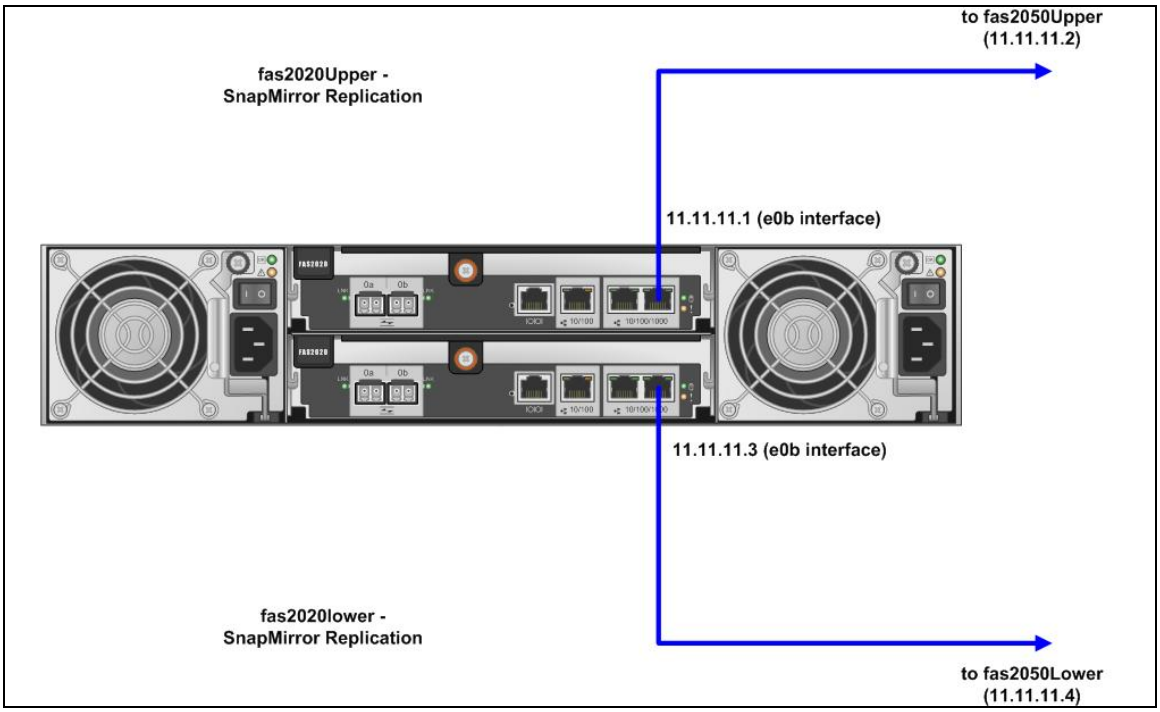


Figure 10.19 - Site 1 FAS Replication Connectivity

Site 2

Located in Site 2 we have the FAS2050 array which is split as previously described into two "heads" those being fas2050Upper and fas2050Lower. Our replication interface IP addresses for Site 2 were:

- fas2050upper (11.11.11.2 replication IP address)
- fas2050lower (11.11.11.4 replication IP address)

Using FilerView again we can see the interface selection for the "upper" controller fas2050upper, Figure-10.20:

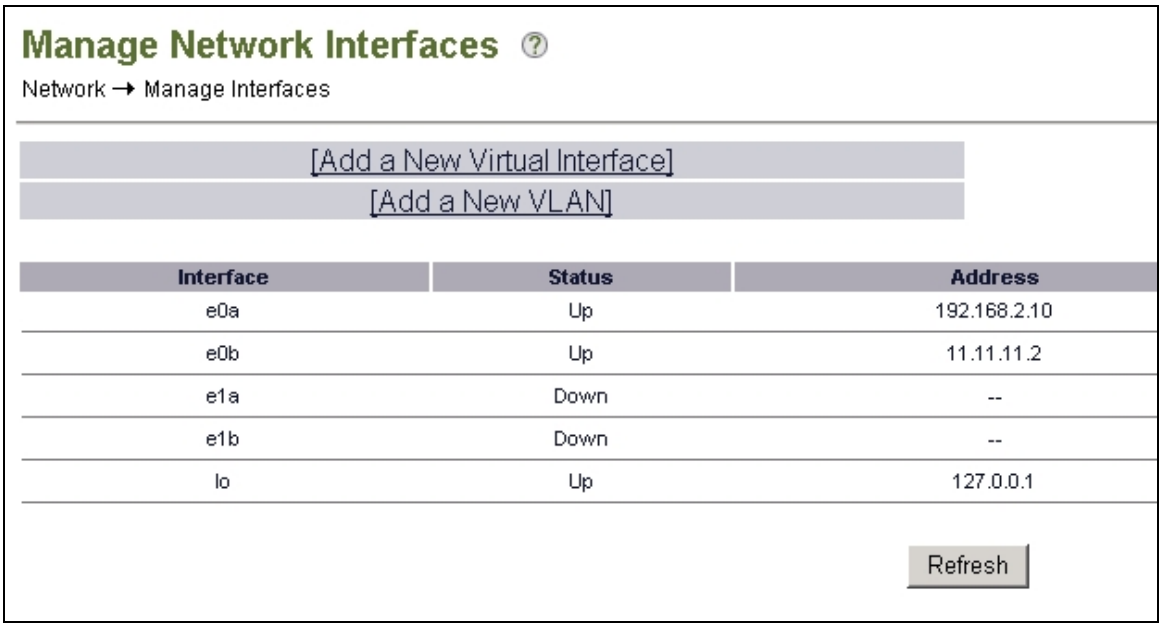


Figure 10.20 - fas2050upper Network Interfaces

Figure-10.21 for fas2050lower we have:

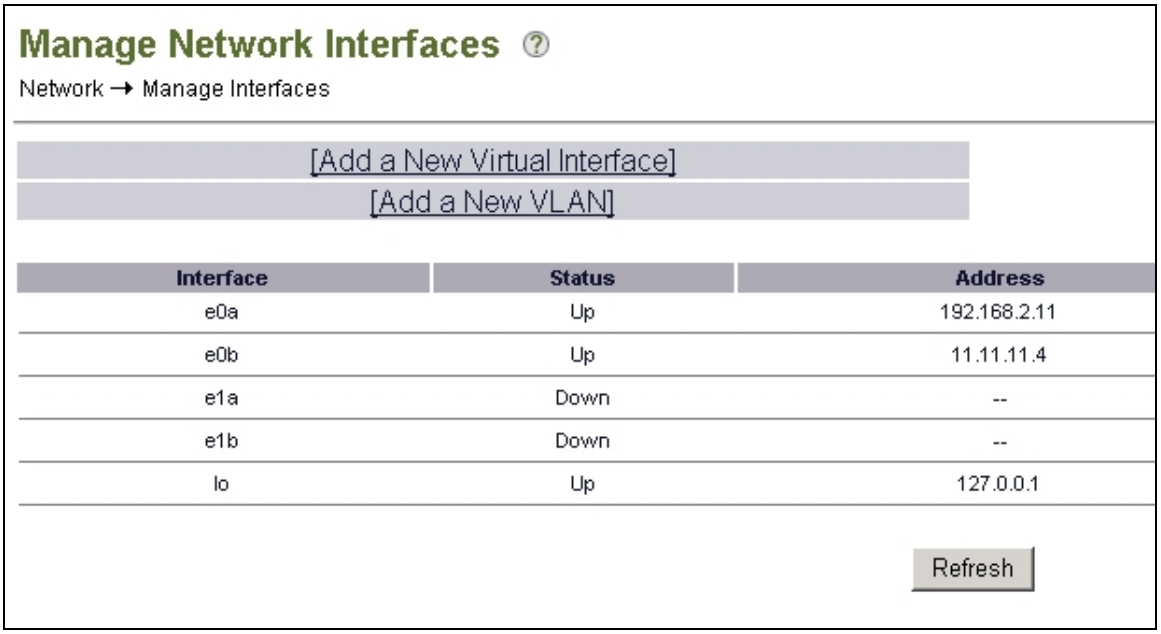


Figure 10.21 - fas2050lower Network Interfaces

Taking a look at the rear view of the FAS2050 itself in Figure-10.22 we can map this interfaces to show their location and intended Site 1 destinations:

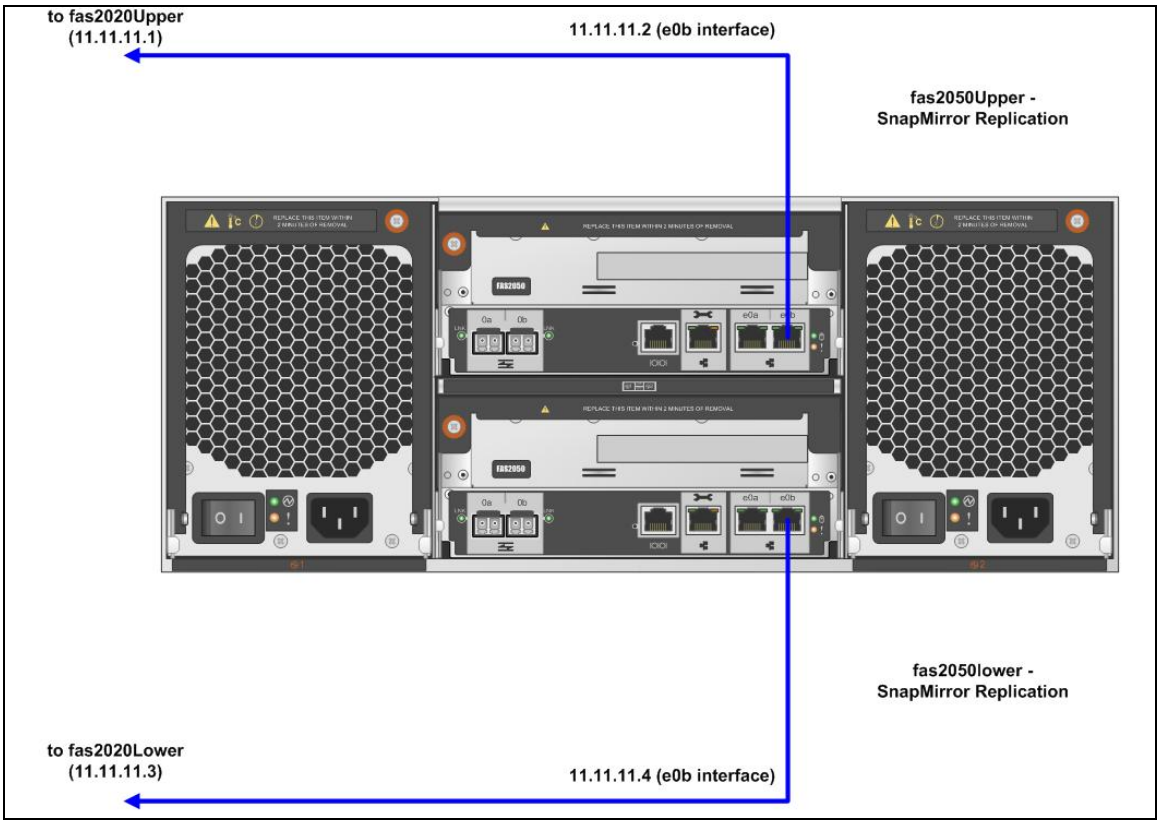


Figure 10.22 - Site 2 FAS Replication Connectivity

We can now combine the previous figures and insert some high-level representations of our volumes (and VMFS datastores within them) to provide an illustration of how the configuration for SnapMirror replication fits together in our environment. Figure-10.23 shows this:

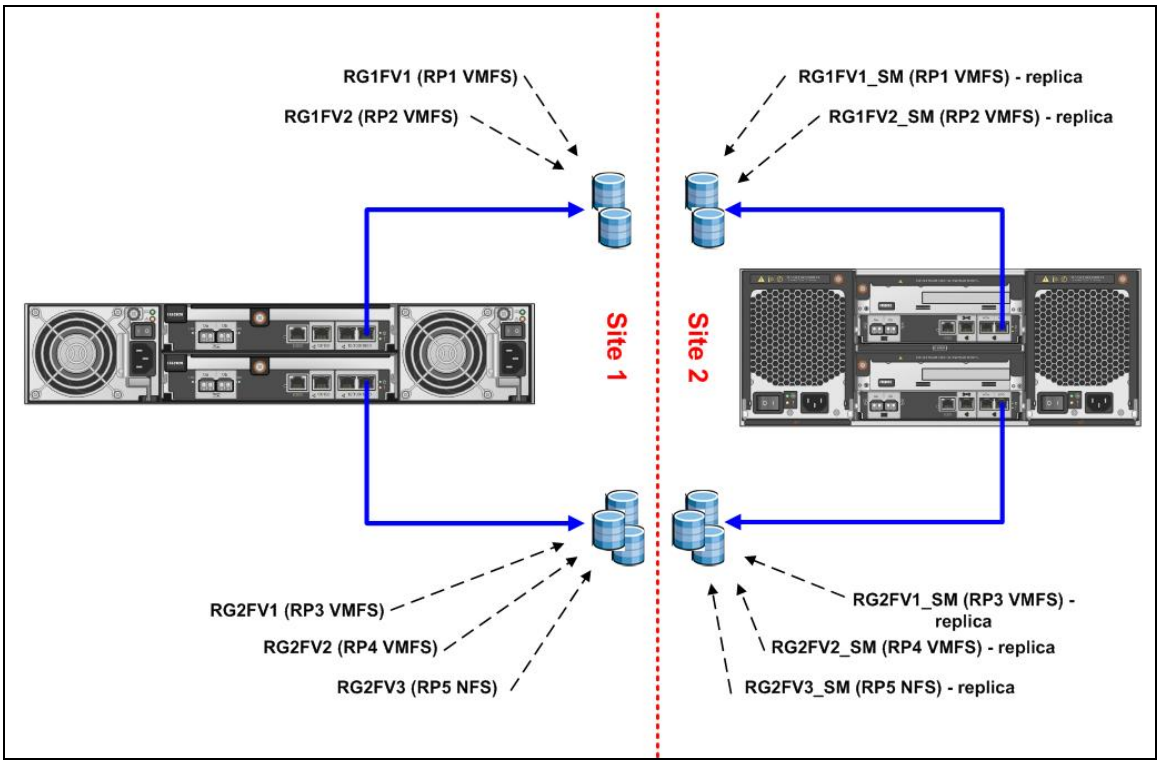


Figure 10.23 - Site 1 and Site 2 Replication Overview

NetApp dual controller FAS systems provide high availability by ensuring that the failure of any single component within the storage system does not compromise the availability or integrity of data.

NetApp dual controller FAS systems also ensure that data replication continues, despite the failure of any single component, including a controller.

For more information, see the Active/Active Controller Configuration Overview and Best Practice Guidelines - http://media.netapp.com/documents/wp_3450.pdf

Chapter 11. Storage Platform Details

The purpose of this chapter is to describe the storage platform and architecture utilized as the basis for this guide whilst also detailing the technologies used and reasoning behind the sample design.

The overall goal when laying out the storage was to keep the design simple whilst also providing the ability to illustrate some key points to bear in mind when building your BCDR solution on top of VMware virtual infrastructure.

The design employed contains a mix of replicated VMFS LUNs and also an NFS datastore. The LUNs have been grouped into atomic units of failover which we have labeled as protection groups, this concept will be explained in more detail later in this chapter. Local non-replicated LUNs containing VMFS volumes are also in evidence at each datacenter location.

This chapter will not discuss VMware virtual infrastructure storage options, VMFS creation or other similar administration and managements concepts in any detail as this is beyond the scope of this book. For further detail on these concepts please refer to the VMware virtual infrastructure documentation and technical papers

VMware Infrastructure Documentation (3.0.x / 3.5.x)

Virtual Infrastructure 3.0.x	
Basic System Administration	http://www.vmware.com/pdf/vi3_301_201_admin_guide.pdf
Server Configuration Guide	http://www.vmware.com/pdf/vi3_301_201_server_config.pdf
SAN Configuration Guide	http://www.vmware.com/pdf/vi3_301_201_san_cfg.pdf
Online Library	http://pubs.vmware.com/vi301/wwhelp/wwhimpl/common/html/switch.htm

VMware Infrastructure 3.5.x	
Basic System Administration	http://www.vmware.com/pdf/vi3_35/esx_3/r35/vi3_35_25_admin_guide.pdf
VMware ESX 3 Configuration Guide	http://www.vmware.com/pdf/vi3_35/esx_3/r35/vi3_35_25_3_server_config.pdf

Fibre Channel SAN Configuration Guide	http://www.vmware.com/pdf/vi3_35/esx_3/r35/vi3_35_25_san_cfg.pdf
Online Library	http://pubs.vmware.com/vi35/wwhelp/wwhimpl/common/html/switch.htm
SAN System Design and Deployment Guide	http://www.vmware.com/pdf/vi3_san_design_deploy.pdf

Site 1 / Site 2 Storage Topology

Figure-11.1 below illustrates the storage topology utilized across our Site 1 and Site 2 datacenters.

Starting with Site 1 we have two local LUNs each 90GB of useable space each formatted with the VMware VMFS File system. These local LUNs are presented to all ESX hosts and are non-replicated storage. These VMFS datastores are used to hold local core infrastructure service(s) virtual machines such as:

- Active directory
- DNS
- DHCP

Typically these types of virtual machines do not need to be located on replicated storage as counterpart virtual machines already exist at Site 2 in our design.

Site 1 VMware ESX hosts are also presented with five replicated LUNs (four VMFS datastores and one NFS datastore (RP5) as shown in Figure-1. The five datastores are named RP1 through RP5 and are then divided up between two logical groups which have been called “Protection Group 1” and “Protection Group 2”. For a reminder of the protection group concept please refer back to the design considerations section of chapter 3 in this book.

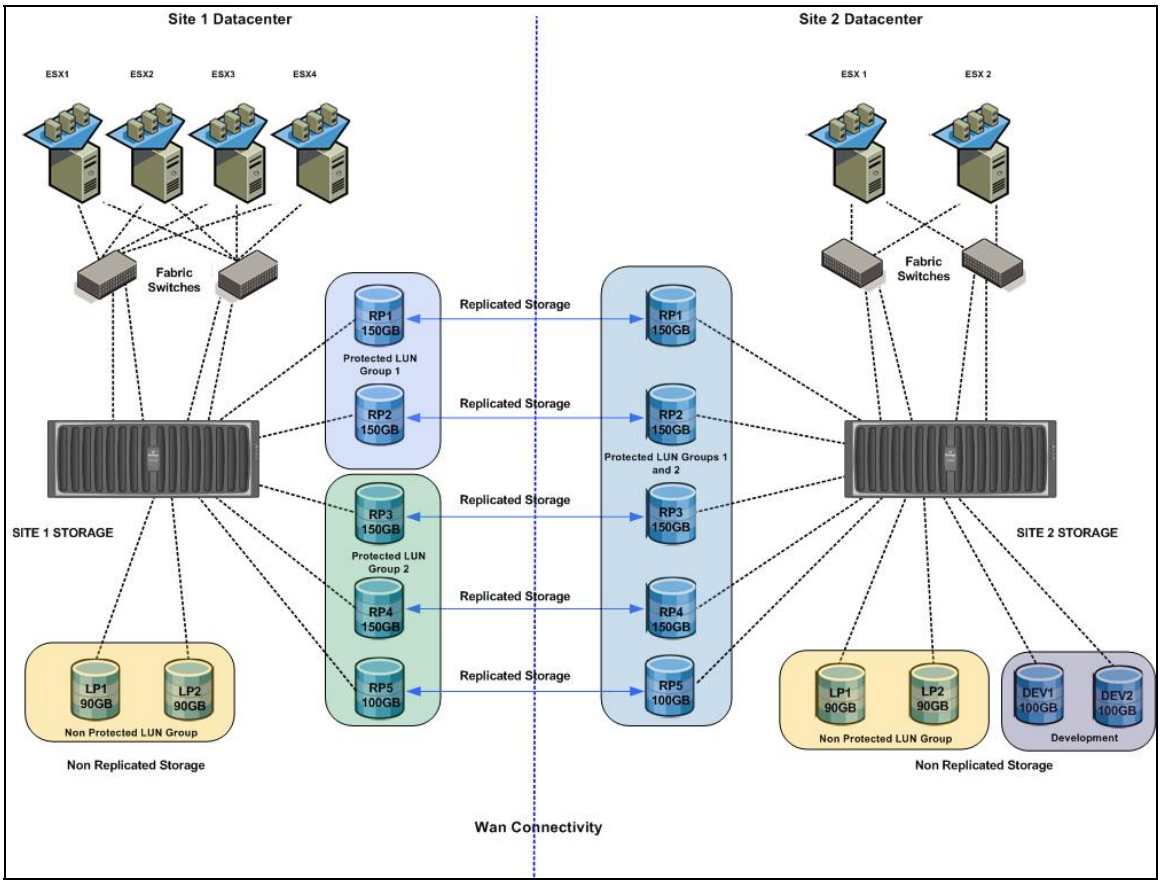


Figure 11.1 - Storage Topology

Site 2 VMware ESX systems are presented with four local LUNs all of which are non-replicated VMFS storage. These are labeled as LP1,LP2 (90GB each) and DEV1 and DEV2 (100 GB each). These local LUNs are used as datastores for virtual machines local to Site 2 only. LUNs LP1 and LP2 contain Site 2 infrastructure virtual machines, DEV1 and DEV2 are used to contain hypothetical test and development virtual machines.

VMware Infrastructure Setup

Where possible all ESX settings have been kept default. As detailed later in this chapter certain LVM settings were altered to enable replicated volumes to be accessed and these are described in the appropriate section.

All other ESX storage configuration settings were left as default and are described as per the standard VMware Infrastructure documentation. It is beyond the scope of this guide to describe storage configuration and presentation to ESX.

NetApp FAS System Setup

There are many different ways to store and access data when using NetApp storage systems with VMware Infrastructure 3. Each method offers unique advantages, and involves unique considerations.

ESX servers can connect to storage in the form of LUNs presented over Fibre Channel or iSCSI. These LUNs can be formatted with VMFS, or presented directly to a virtual machine as a RAW disk. ESX servers also support NFS volumes, exported from a NAS appliance.

Virtual machines, with the appropriate networking configuration, can access LUNs via a software iSCSI initiator within the virtual machine, as well as CIFS (for windows virtual machines) and NFS (for Unix / Linux virtual machines).

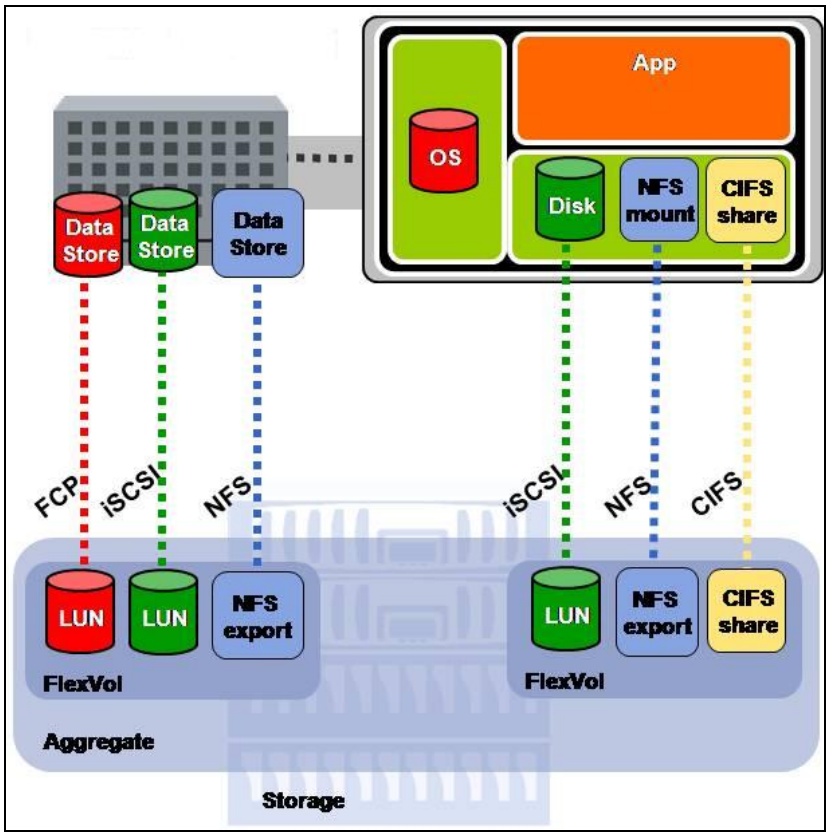


Figure 11.2 - Multi Protocol Support

For more information, read the NetApp technical report, "NetApp and VMware Infrastructure Storage Best Practices.[8]"

For the same reasons that it is beneficial to abstract servers from CPUs and other physical resources, it is also beneficial to abstract data volumes from disk drives.

NetApp FAS storage subsystems perform this abstraction with the use of Flexible Volumes. A NetApp FlexVol® resides in an aggregate, which is a pool of physical disk drives, shared by other Flexible Volumes.

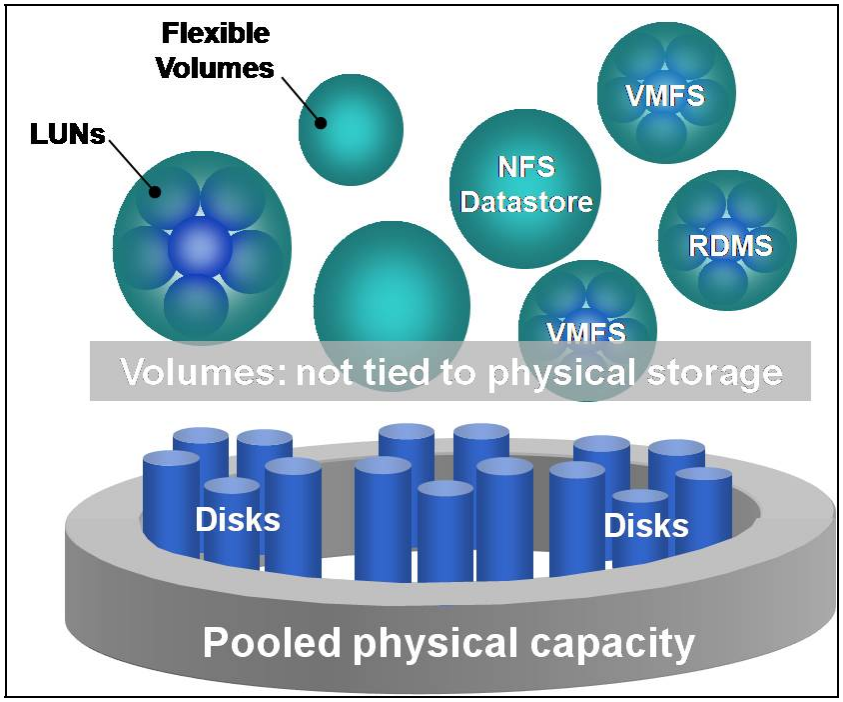


Figure 11.3 - NetApp Flexible Volumes

NetApp FlexVols can be made available to Unix users via the Network File System (NFS) and Windows users via the Common Internet File System (CIFS).

NetApp FlexVols also function as a container for LUNs, which provide block based storage via the Fibre Channel Protocol (FCP) or the Internet SCSI protocol (iSCSI).

[8] http://www.netapp.com/us/library/technical-reports/tr_3428.html

Any NetApp FAS system can store and serve data over all these protocols at the same time, providing the ultimate in configuration flexibility.

For the purposes of our storage design we assign a single LUN per FlexVol. Figure-11.4 depicts the FlexVol configuration at Site 1:

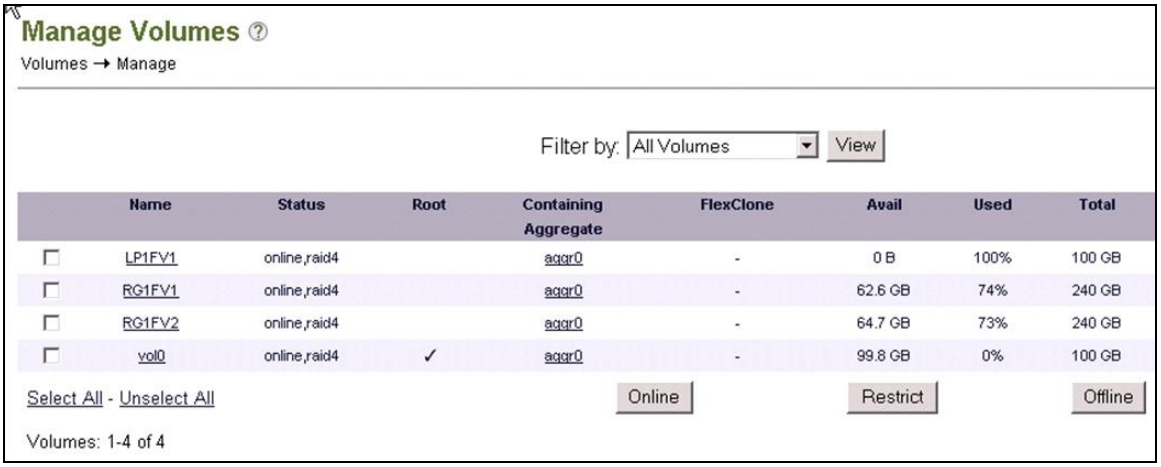


Figure 11.4 - Site 1 FlexVols Upper Controller

Figure-11.4 shows that we have created three FlexVols for the purposes of this book. They are:

- LP1FV1 (100 GB)
- RG1FV1 (240 GB)
- RG1FV2 (240 GB)

Any FlexVol name beginning with "L" will hold LUNs that will contain local VMFS datastores and those beginning with "R" will hold LUNs that will be replicated to Site 2 and contain VMFS (or NFS in the case of RP5) datastores containing the virtual machines we are protecting.

In the storage provisioning section that follows we will show the LUNs contained within these FlexVols to further illustrate this example.

Storage Provisioning

NetApp FlexVols can contain multiple LUNs as was previously stated for our configuration a single LUN is assigned to each FlexVol. When planning any replication architecture it is important to understand the level at which your vendors replication technology works.

A simple example would be, assume we had chosen to create a FlexVol of 1TB and then within that FlexVol we had created three LUNs each of 250 GB. These LUNs would be presented to our ESX servers ready for use. If we then populate the LUNs with virtual machines and subsequently decide we wish to replicate the LUNs we may have a problem. Suppose we only wish to replicate two of the LUNs but our FlexVol contains all three. As the replication tool works at the FlexVol level we will now have to replicate a LUN that does not require protection via replication. We can of course change the layout but this is work that could have been avoided had we consulted with the storage team during our storage layout planning stage.

For further information on configuring Flexible and Volumes, and recommended Virtual Machine Data Layouts for use with VMware Infrastructure 3, see http://www.netapp.com/us/library/technical-reports/tr_3428.html

To carry on our illustration from the previous section the Figure-11.5 displays the LUNs available on one of the Site 1 controllers (2020Upper):

LUN	Description	Size	Status	Maps Group : LUN ID
/vol/LP1FV1/LP1.lun	LP1 lun	90 GB	online	RG1 : 0
/vol/RG1FV1/RP1.lun	RP1 Lun	150 GB	online	RG1 : 1
/vol/RG1FV2/RP2.lun	An optional description of the LUN.	150 GB	online	RG1 : 9

Figure 11.5 - Site 1 LUNs Upper Controller

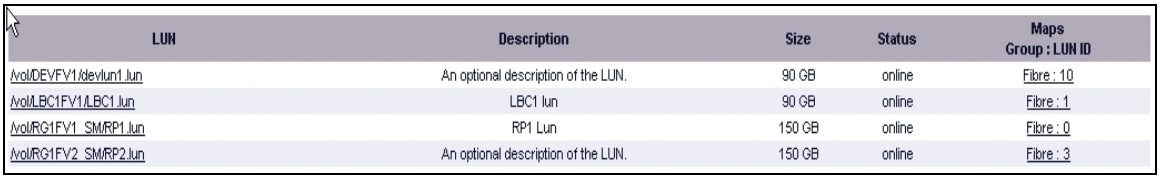
Each LUN is displayed with a full path which also includes the name of its FlexVol so we have:

- /vol/LP1FV1/LP1.lun (lunid 0)
- /vol/RG1FV1/RP1.lun (lunid 1, replicated via SnapMirror)
- /vol/RG1FV1/RP2.lun (lunid 9, replicated via SnapMirror)

For the second controller at Site 1 (2020lower) we have (not pictured):

- /vol/LP2FV1/LP2.lun (lunid 2)
- /vol/RG2FV1/RP3.lun (lunid 3, replicated via SnapMirror)
- /vol/RG2FV1/RP4.lun (lunid 4, replicated via SnapMirror)

For the array at Site 2 the two controllers each had a similar setup, the Upper controller (2050Upper) can be shown in Figure-11.6 as:



LUN	Description	Size	Status	Maps Group : LUN ID
/vol/DEVFV1/devlun1.lun	An optional description of the LUN.	90 GB	online	Fibre : 10
/vol/LBC1FV1/LBC1.lun	LBC1 lun	90 GB	online	Fibre : 1
/vol/RG1FV1_SM/RP1.lun	RP1 Lun	150 GB	online	Fibre : 0
/vol/RG1FV2_SM/RP2.lun	An optional description of the LUN.	150 GB	online	Fibre : 3

Figure 11.6 -Site 2 LUNs Upper Controller

Again each LUN is displayed with a full path which also includes the name of its FlexVol so for the Upper controller we have:

- /vol/DEVFV1/devlun1.lun (lunid=10)
- /vol/LBCFV1/LBC1.lun (lunid=1)
- /vol/RG1FV1_SM/RP1.lun (lunid=0, SnapMirror replica copy)
- /vol/RG1FV2_SM/RP2.lun (lunid=3, SnapMirror replica copy)

For the second controller at Site 2 (2050lower) we have (not pictured):

- /vol/DEVFV2/devlun2.lun (lunid=11)
- /vol/LBC2FV1/LBC2.lun (lunid=2)
- /vol/RG2FV1_SM/RP3.lun (lunid=4, SnapMirror replica copy)
- /vol/RG2FV2_SM/RP4.lun (lunid=5, SnapMirror replica copy)

All LUNs are exposed to the ESX servers over Fibre Channel, with appropriate LUN masking and Fabric zoning to ensure secure connectivity.

You will notice the SnapMirror destination copies of our LUNs have been assigned different LUN ids to the SnapMirror source LUNs. In our configuration, this is not an issue as we have set the LVM.EnableResignature and LVM.DisallowSnapshotLUN options to 0.

As storage arrays create exact replicas of the source volumes (LUN's, datastores however you want to visualize them), being replicated from Site 1 in our case, all information including the unique signature (and VMFS datastore label, if applicable) is replicated to the target array at Site 2. If a copy of a VMFS volume is presented to any VMware ESX host by default (using the default settings) the ESX host automatically masks the replicated volume. See Figure 11.7 for an overview of these advanced parameters.

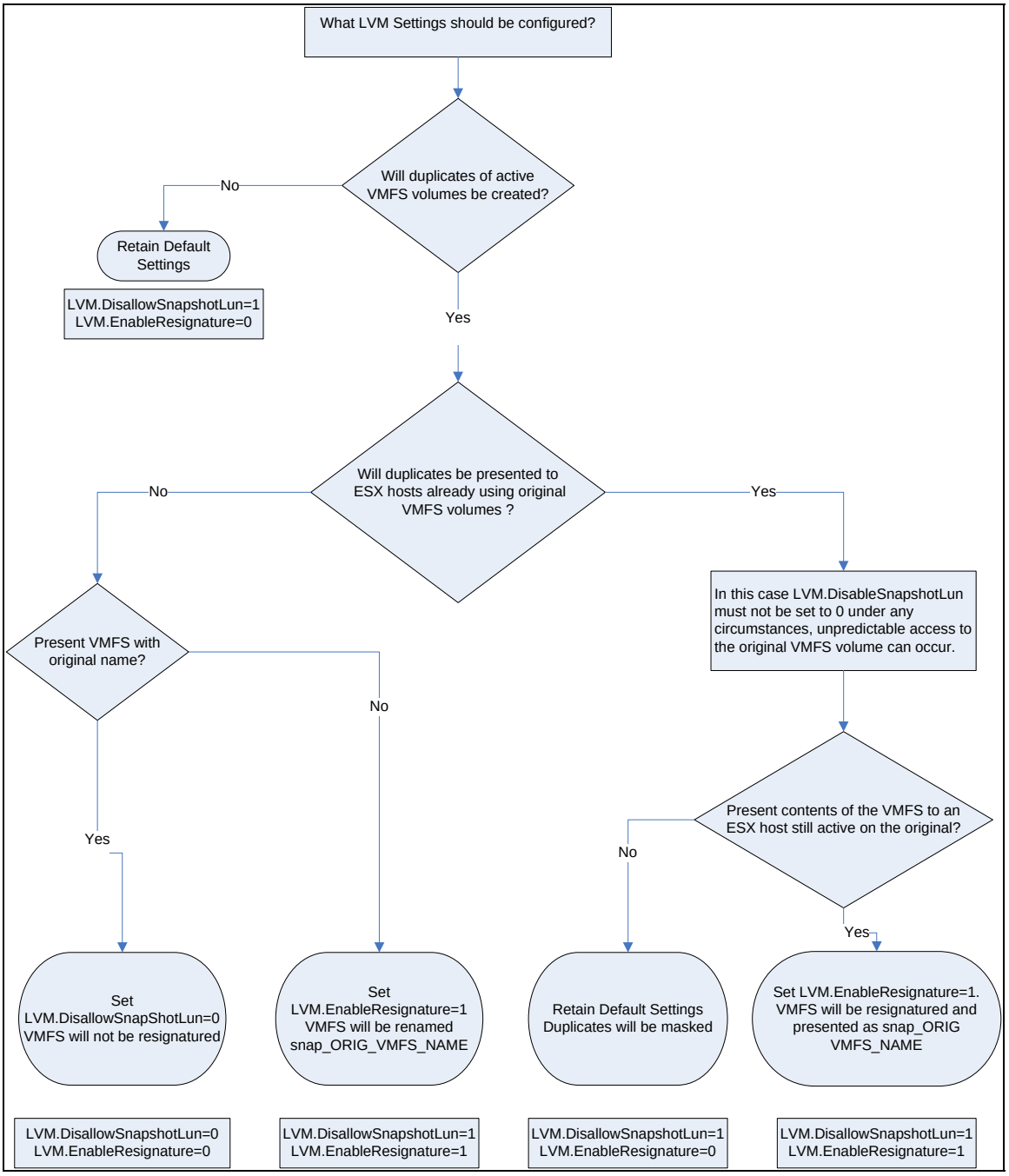


Figure 11.7 - LVM Options

The device which holds the copy volume is determined by comparing the signature stored on the device with the computed signature. Replica devices (volumes), for example, have different unique IDs from their source devices (volumes) with which they are associated. What this means is that the signature for the replica device (volume) differs from the one stored on it. This enables ESX server to always identify the copy correctly.

As discussed previously ESX Server provides two different mechanisms to access copies of VMFS volumes. We already know these to be the advanced configuration parameters LVM.DisallowSnapshotLun and LVM.EnableResignature. These two configuration parameters control the behavior of the ESX VMkernel when presented with copies of a VMware file system (VMFS volume).

If LVM.DisallowSnapshotLun is set to 0 (as in our solution), the copy of the data is presented with the same label name and signature as the source device (volume). However, caution should be excercised in environments where an ESX Servers has access to both source and target devices, in this case the parameter should be left at its default value of 1.

If LVM.EnableResignature is set to 1, the VMFS volume holding the copy of the VMware file system is automatically resignatured with the computed signature (using the UID and LUN number of the target device). In addition users will notice in VirtualCenter that the VMFS label is appended to include "snap-x", where x is a hexadecimal number that can range from 0x2 to 0xffffffff. The default value for this parameter is 0. If this parameter is changed to 1, the advanced parameter LVM.DisallowSnapShotLun is ignored.

With our solution we have two datacenters, Site 1 and Site 2 and have zoned our fabric in such a way that ESX hosts at Site 1 cannot access devices (volumes, LUNs) located in Site 2 and vice versa. The result here is that at Site 2 we can safely set LVM.DisallowSnapShotLun =0.

In our solution the LVM.EnableResignature parameter will not be changed for any of the ESX Servers at the remote site, Site 2. If we did set LVM.EnableResignature =1 then all volumes that are considered to be copies of the original data would be resignatured. If you resignature the copies once they are at the remote site, Site 2, then this will make failing back those copies to Site 1 more complex.

In the interests of simplicity, it is recommended to use the assign a replicated LUN the same LUN id in each site, wherever possible.

Any LUNs not described as being part of the SnapMirror replication stack are local VMFS datastores used to contain virtual machines that are not required to be protected via replication across sites. Such virtual machines would typically be DHCP, DNS, Active Directory virtual machines as an example.

Not shown above but described later, we have also provisioned a FlexVol, exported via NFS. The export settings permit access only to the ESX servers in each site. The ESX servers mount the NFS volume as a datastore.

Network File System (NFS) Provisioning

NetApp Flexible Volumes can be accessed via the Network File System once configured as an export. Once mounted as a datastore to ESX servers, virtual machines Disk (VMDK) and other configuration files can be stored directly on the volume.

In this configuration, we have configured a single FlexVol for access via NFS, named RG2FV3. This volume is mounted as an NFS datastore called RP5, accessible to all ESX servers in Site 1, and is used to store virtual machines. This datastore is also replicated via SnapMirror.

Figure-11.8 depicts a screenshot that shows the NFS datastore (RP5) configured within VirtualCenter.

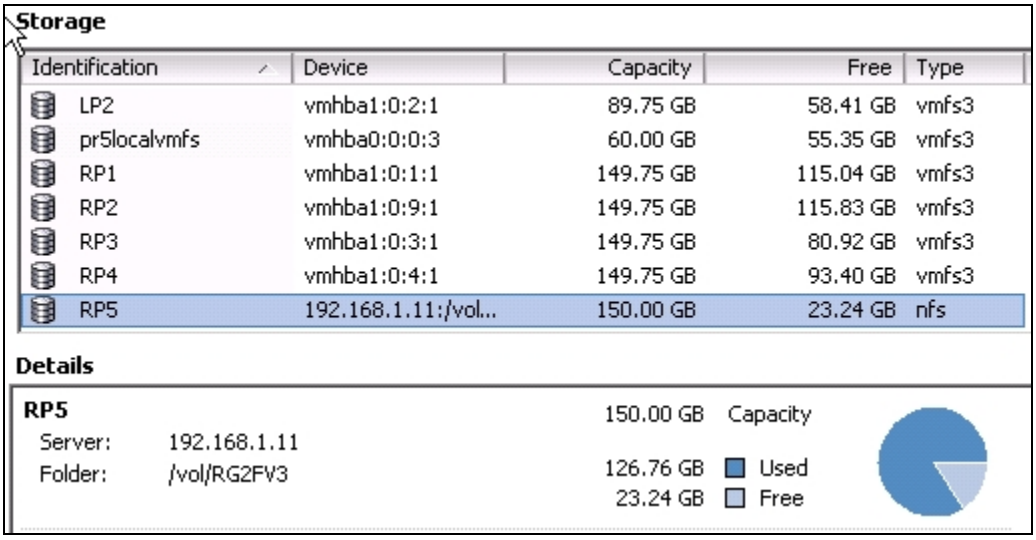


Figure 11.8 - NFS Datastore

Network Connectivity

The NetApp FAS2000 series controllers each have dual onboard Gigabit Ethernet ports (**e0a** & **e0b**).

The FAS2050 controller includes a PCI expansion slot, which can be used to add additional Ethernet or Fibre Channel ports.

In this case, we are using the onboard ports only.

- **e0a** on each controller connects to the main network in each site, used for management as well as NFS connections.

- **e0b** on each controller connects to a separate network, which is routed between sites. This connection is used for SnapMirror replication.

Figure-11.9 demonstrates how this configuration appears in FilerView:

Interface	Status	Address
e0a	Up	192.168.1.10
e0b	Up	11.11.11.1
lo	Up	127.0.0.1

Figure 11.9 - FAS Network Interfaces

Fibre Channel Connectivity

The NetApp FAS2000 series storage arrays include dual onboard 4Gb Fibre Channel ports, providing connectivity for Storage Area Networks (SAN) and / or attaching additional disk capacity.

In this case, sufficient storage capacity was provided by the onboard drives, so both Fibre Channel ports on both controllers on each storage system were configured as Fibre Channel target ports, allowing them to connect to a SAN.

Each Controller in a NetApp system will use the same Fibre Channel World Wide Node Name (WWN), appearing as one physical system to the fabric. This not only improves fabric manageability, but also ensures local controller failovers are totally transparent and non-disruptive to hosts.

Figure-11.10 shows a high-level view of the two Sites in terms of SAN fabric. This configuration is described in detail in Chapter 10. For our purposes here we simply need to illustrate that each FAS storage system is connected to two fabric switches at each site:

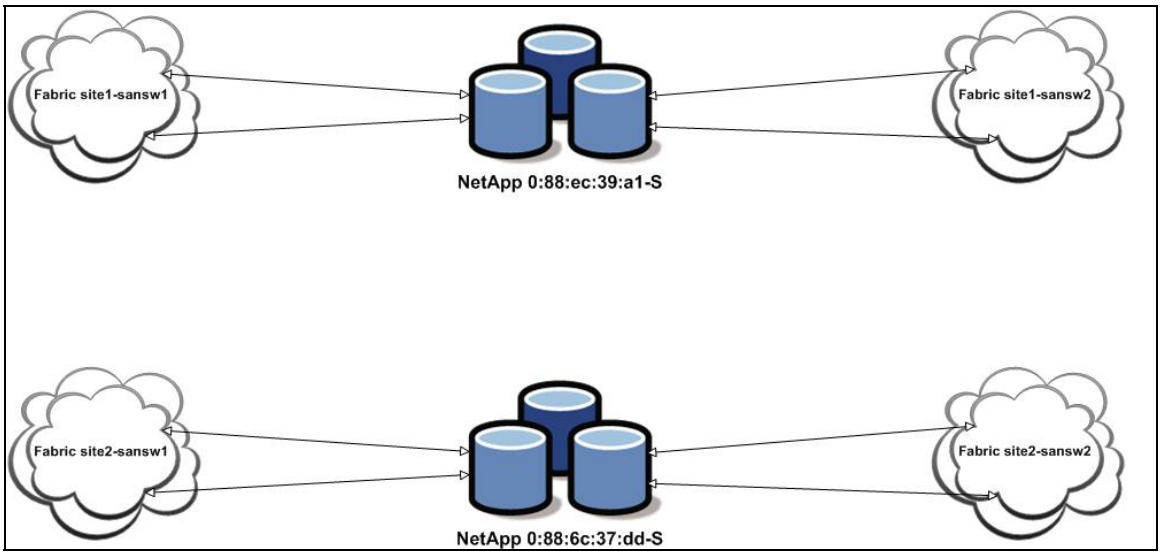


Figure11.10 - Site 1 & Site 2 Fabric Overview

Each ESX host in our environment contains a single dual port Emulex HBA. Each port of the HBA is connected to one of the switches located within that ESX Hosts site. For example an ESX Host in Site 1 will have a connection to both the switch "site1-sansw1" and "site1-sansw2". Storage connectivity is described in more detail in the previous chapter.

FAS Storage Replication

The foundation of protecting virtual machines within this book is the use of storage replication.

Most storage solutions available today have the capability to replicate volumes / LUNs across datacenters. With the FAS storage solution used in our infrastructure we make use of NetApp's SnapMirror technology to replicate Volumes from one FAS storage system to another.

In this section we will walk through the creation of a replica volume and show how the LUN contained within it is represented in Virtual Center at the various stages of its creation. Before we proceed with the example it will be of use to look at SnapMirror in a bit more detail.

SnapMirror Overview

Server consolidation initiatives, combined with Storage consolidation, help to simplify infrastructures, making it much easier to implement a comprehensive strategy to protect against hardware, software, or even site failures.

Technologies such as VMotion play an important role in protecting against failures at the server level, however any business continuity and disaster recovery strategy should take into account the network and storage levels.

There are many technology solutions that can play a role in such a strategy, such as:

Backups

Backups provide a way to recover lost data from an archival medium (tape or disk).

Eliminating Single Points of Failure

Redundant hardware technologies also help mitigate the damage caused by hardware issues or failures.

Data Replication (In Region)

By synchronously replicating data to a nearby site, virtual machines and applications can be rapidly recovered from the point of failure, should the primary site become unavailable for any reason.

Traditionally, synchronous replication sites are located within the same geographical region (less than 100 km apart), and Fibre Channel connectivity is provided between sites. In addition, the process of synchronously replicating data incurs a performance overhead.

Data Replication (Out of Region)

In order to protect against a disaster that affects an entire region, data can be replicated over longer distances.

For more information, read the technical report, "High Availability and Disaster Recovery for VMware Using NetApp SnapMirror and MetroCluster[9]," available from the NetApp Web site.

Storage Replication with SnapMirror

NetApp SnapMirror provides a fast and flexible enterprise solution for mirroring or replicating data over local area, wide area, and Fibre Channel (FC) networks, see Figure-11.11.

---

[9] http://partners.netapp.com/go/techontap/matl/VMworld_TR_DR.pdf

SnapMirror can be a key component in implementing enterprise data protection strategies. If a disaster occurs at a source site, businesses can access mission-critical data from a mirror on a remote NetApp system, ensuring uninterrupted operation.

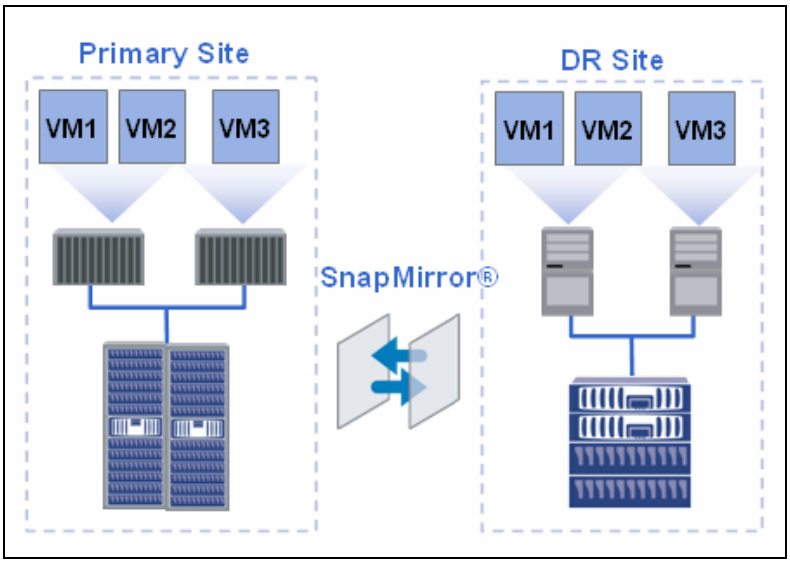


Figure 11.11 - NetApp SnapMirror

SnapMirror is typically configured on a per volume basis, and works with NetApp Snapshots™ to send only changed data blocks to the disaster recovery storage, reducing network utilization.

SnapMirror with FlexClones, discussed later on in this chapter, enables space-efficient copies to be created on the disaster recovery storage for other uses such as testing, development, and quality assurance, without affecting the production system. SnapMirror also enables centralized backup of data to tape from multiple datacenters, reducing investment in tape infrastructure as well as offloading the production system from tape backups.

SnapMirror can replicate data to one or more NetApp storage systems, ensuring protection against both site and regional disasters. You configure a replication schedule that fits your business needs: synchronously, semi-synchronously or asynchronously, at intervals from minutes to hours to days.

Recovery Point Consistency

There are many levels at which the consistency of data contained within a backup can be a factor, and there are many different terms used to describe various states of consistency.

To clarify the terminology used here, we will describe the various levels at which data consistency can be achieved,

First, let's consider the general path that IO will follow in a VMware Infrastructure, shown in Figure-11.12:

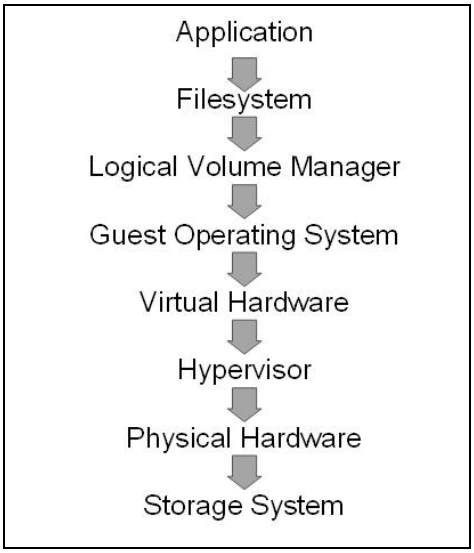


Figure 11.12 - I/O path in a virtual infrastructure

In this case, the application is the source of IO. As an example this could a Microsoft Office file that a user is working on, or a Microsoft SQL Server database (although applications such as SQL server include more advanced features such as transaction logs, which introduce additional considerations to data consistency).

Although some applications can be configured to write to raw devices, we will continue to use Microsoft Office as an example. Office writes data to files stored within a File system (typically NTFS), however when a user opens and modifies a file, those changes are not immediately saved.

When Office automatically saves, or when the user manually saves the file, the File system captures those writes into memory using what is often referred to as a "File system buffer" or "Log File". Those writes will remain in memory until the File system flushes the data to disk at a convenient time.

At this point the Logical Volume Manager, using an IO request / acknowledgement sequence that extends to the underlying storage system or device, ensures that the updated version of the users Office file has been written to the underlying storage system or device, before the File system flush operation is completed.

In our BCDR case, the underlying storage array is replicating data to our Disaster Recovery site. In the event that we have to failover to our DR site, the storage array in our DR site will contain a point in time copy of our data.

The currency of this copy of data, or rather, what point in time it is at relative to our original copy in our primary / protected site, is often referred to as "lag". The storage replication technology in use can affect the lag.

For example synchronous replication, typically used over short distances, will maintain a copy of data in the DR site that is current up to the point of failure in the primary / protected site. This can be thought of as "zero lag".

Asynchronous replication technologies, typically used to replicate over longer distances, will maintain a copy of data in the DR site which is within a defined period of lag.

When dealing with storage based replication, data consistency is a factor regardless of the lag. The storage system can only replicate the data that has been written to it.

It is possible that changes have been written by the application and are captured in the applications transaction log, but have not been committed to the application's database. Alternatively, continuing our Microsoft Office example, a file has been written to the file system, but those changes not yet been flushed to the storage system.

Although ensuring consistency at an application level is difficult to achieve for Microsoft Office, there are common methods for achieving application consistent point in time copies for some applications. For example, storage vendors provide application consistent backup capabilities for applications such as SQL Server, Exchange, Sharepoint, Oracle and SAP.

From the VMware perspective to ensure consistency at the file system level, the vmware tools package includes a sync driver, which can be used to quiesce virtual machines prior to taking a VMware snapshot.

So there are three levels at which data consistency should be considered.

Storage Consistency

The Office file stored on our storage system or device must include all the data that the volume manager has written to the storage system or device, in order for the file to be consistent at a storage level.

Enterprise class storage systems generally include features such as NVRAM (Non-Volatile RAM) to ensure data integrity and consistency.

If, for example, the users PC experienced a hardware failure, another PC could be used to read the users' storage system or device and recover data from a point of storage consistency. This means the user can, at best, recover the file from the last time the File system flushed itself.

However, depending on the File system being used, it is possible that the user may not be able to recover the file at all, because the File system itself may be corrupt. To address this, operating systems have, for many years, included resiliency to recover File systems on from a storage consistent state.

In the case of NTFS and EXT3, journaling is used to recover the file system. For more information, see "What is NTFS?[10]" or look up the EXT3 Wikipedia article[11].

It is worth noting that, in the case of NTFS, and depending on the version of Windows, CHKDSK may need to be run upon recovery to repair the file system.

The term storage consistent backup, as used here, refers to a copy of the data which contains all the data written to the storage system or device up to the point where the backup started.

File System Consistency

The office file stored in the file system must include all changes that were written to the file system, including any changes that were stored in the log file / file system buffer in memory, in order for the file to be consistent at a file system level.

In the event that the users PC experienced a power failure, the user will be able to recover the latest version of the file that was saved. However, any changes made since the last save will have been lost.

The term file system consistent backup, as used here, refers to a copy of the data which contains all the data written to the file system, as well as all the data written to the storage system or device up to the point where the backup started.

A common method of achieving file system consistency for a backup is to initiate a flush of the file system, prior to taking a backup of the data. During the backup process, new writes are typically prevented from being written to disk until the backup has completed.

---

[10] http://technet2.microsoft.com/windowsserver/en/library/59a9462a-cbdd-45e7-828b-12c6cd9ae4781033.mspx?mfr=true

[11] http://en.wikipedia.org/wiki/Ext3

The VMware tools package includes a sync driver, which when used in conjunction with VMware snapshots, takes file system consistent point in time copies of virtual machine disks. When combined with NetApp Snapshot and SnapMirror, this can provide multiple file system consistent backups, both locally and remotely.

Figure-11.13 demonstrates how the combination and co-ordination of NetApp & VMware snapshot technology ensures storage and file system consistency.

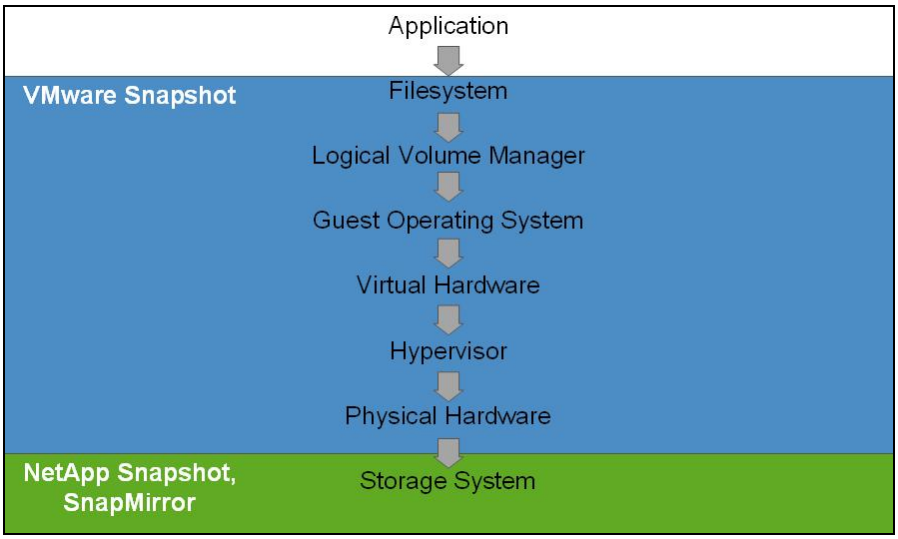


Figure 11.13 - SnapMirror coordination with VMware snapshots

Application Consistency

The office file stored in the file system must include all changes made up to the point of power failure.

Whilst this level of consistency is not available for Microsoft Office files, more sophisticated applications such as SQL server include features such as transaction logs to ensure changes to a database can be recovered and re-applied if necessary. In the case of SQL server, it is desirable for the database itself must be intact, with any outstanding transactions completed in order for it to be recovered the transaction logs to be replayed to recover up to the point of failure.

The term Application Consistent backup, as used here, refers to a copy of all of the application data, which contains all the data written, by one or more users, to the application up to the time when the backup started, as well as the application logs containing all transactions up to the time the backup completed. This includes any and all data written to the file system and underlying storage system or device.

A common method of achieving an Application Consistent backup is to use backup tools which are integrated with specific applications.

For example, NetApp provides Application Consistent backups for SQL Server, Exchange, Sharepoint, Oracle and SAP via the SnapManager suite of products.

Figure-11.14 demonstrates how the SnapManager suite can integrate into a virtual machine.

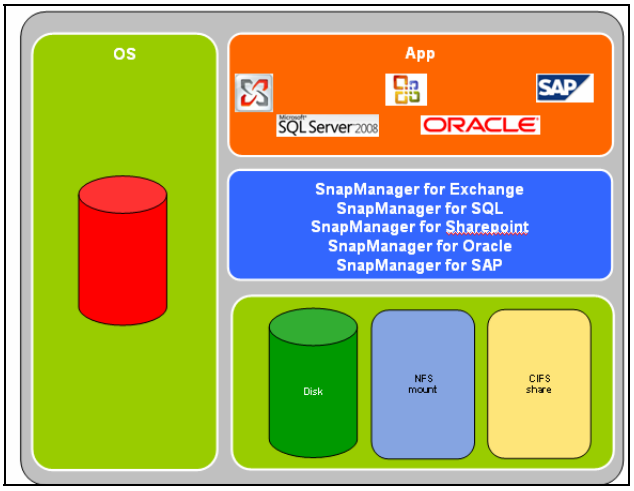


Figure 11.14 - Snap Manager virtual machine integration

For more information on NetApp SnapManager products, please see http://www.netapp.com/us/products/management-software/

Site 1 FAS Configuration for Replication

SnapMirror, from a manageability perspective, pulls data from the source to the destination. As such, SnapMirror relationships are typically managed from the destination storage array.

However, an essential initial step in configuring a SnapMirror relationship is to permit the destination storage array access to the source. In our configuration our destination array in this example is our Site 2 Upper controller also known on our network as fas2050upper.

On the Site 1 FAS Upper controller we therefore need to grant access to fas2050upper via the "Manage SnapMirror Remote Access" screen in filerview. This is shown in Figure-11.15:

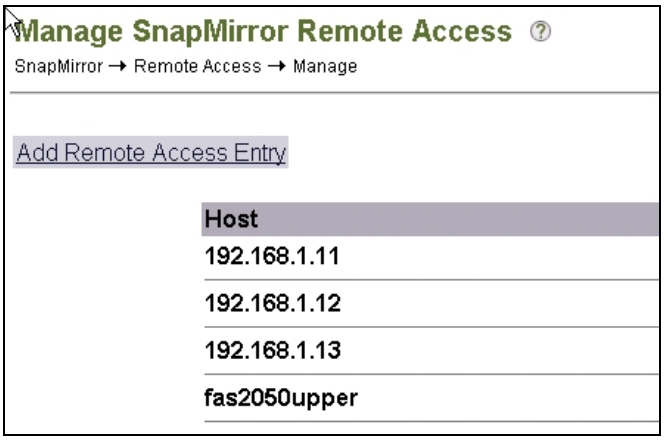


Figure 11.15 - Snapmirror Remote Access

Access can be granted on a per storage controller, or per volume basis to any other NetApp storage controller, specified by hostname or IP address.

Site 2 FAS Configuration for Replication

Before a SnapMirror relationship can be configured, a destination volume (the volume that you wish to replicate data to) must first be created. The destination volume must be configured to be the same size (or larger) than the source volume.

Once the destination volume has been created on the destination storage system, it must be restricted, to ensure that only SnapMirror can write data to it.

Then the SnapMirror relationship can be configured. This can be done via FilerView, using the "Add Snapmirror Entry" button.

You will then be prompted to supply the information required to create a new relationship, see Figure-11.15. We will assume here that we wish to create a new SnapMirror volume that uses our Site 1 volume "RG1FV1" as its source. Within the Site 2 Upper controller we will name the replica volume "RG1FV2_SM" where _SM is used in our naming convention to denote "SnapMirror":

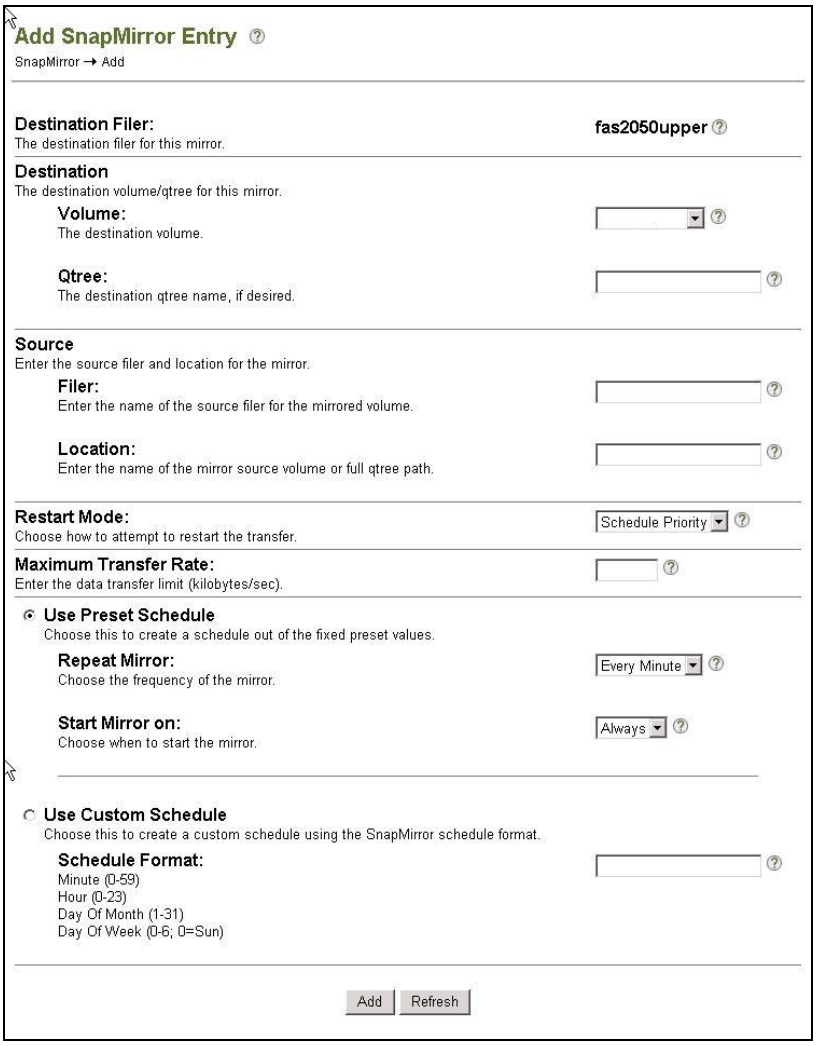


Figure 11.16 - Add Snapmirror

In Figure-11.15 which shows our "Add SnapMirror Entry" screen we have three key piece of information to add:

- Destination > Volume = RG1FV1_SM
- Source > Filer = fas2020upper (or by IP address 11.11.11.1 which is its replica IP address)
- Source > Location = RG1FV1

Once the SnapMirror relationship has been configured, it must be initialized. Initialization is performed via the "Manage SnapMirror > Advanced" screen in filerview for the volume you wish to initialize. This is not illustrated.

The initialization process performs a complete baseline transfer, replicating all data in the source volume to the destination volume. Once the initialization has completed, the SnapMirror relationship will be updated according to the configured schedule, or continuously in the case of Synchronous SnapMirror relationships.

For more information on configuring SnapMirror in synchronous, a-synchronous, or semi-synchronous modes, please see the NetApp SnapMirror Best Practices Guide
http://media.netapp.com/documents/tr_3446.pdf

After initialization if we now review our "Manage SnapMirror" screen in filerview we can see the entry (for purposes of illustration the second entry has also been prebuilt) in Figure-11.17:

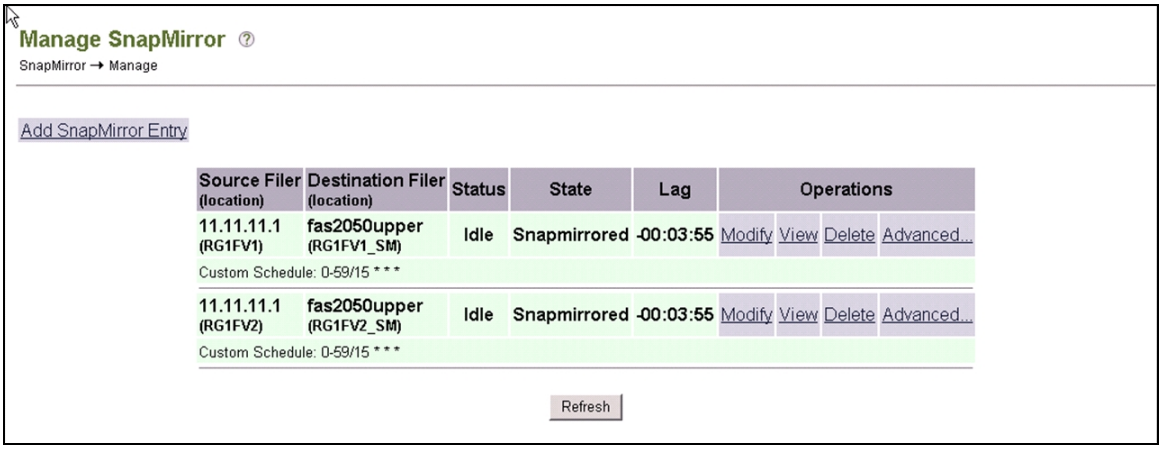


Figure 11.17 - Snapmirror Volumes

In the case of A-Synchronous SnapMirror, once the SnapMirror relationship is established between the source and destination volumes, the SnapMirror relationship will display a status of "Idle" until the next transfer, known as a SnapMirror update, takes place. When the next update will take place will depend on the replication schedule defined. You can gauge the difference (in time) between the source and destination volumes by monitoring the "Lag" for each SnapMirror relationship.

If we review the "Manage Volumes" screen in filerview, shown in Figure-11.18, we can see our two replica volumes. In our configuration, we have used an naming convention for our SnapMirror Destination volumes, which in our case is the SnapMirror source volume name with an _SM suffix.:

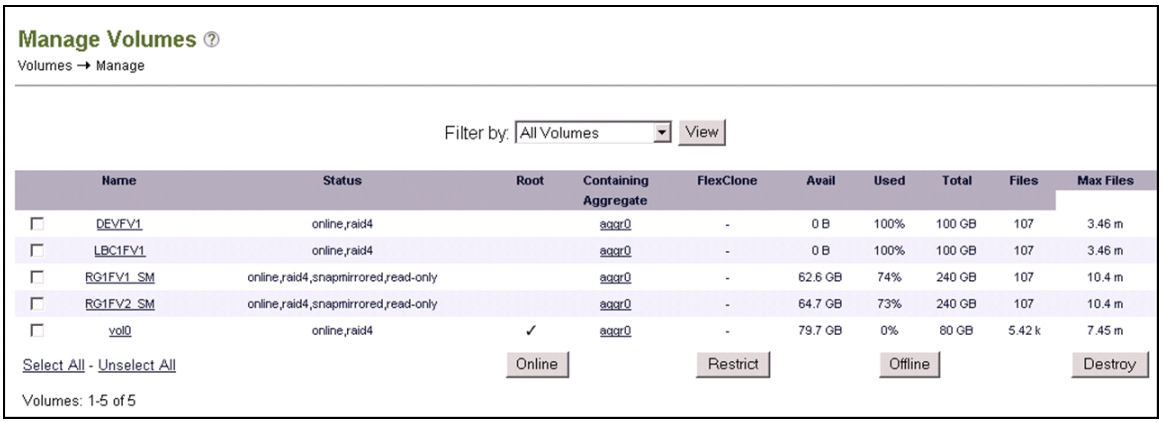


Figure 11.18 - Manage Volumes (Site 2 Upper Controller)

Volumes are defined as:

- RG1FV1_SM
- RG1FV2_SM

At this point it can also be seen that each volume has a status of "read-only". Whilst replication is active the volumes will always display "read-only". If you wish to run BCDR tests then you will make use of FlexClone volumes described later in this chapter. The only time that your SnapMirror volumes will be split or broken-off from their source volumes is usually during a real failover event when you have lost access to the source storage. In our case this would mean losing access to Site 1.

At this point we can review the Storage Adapters screen for one of our ESX hosts located in our Site 2 datacenter. Recall that the lunids for our SnapMirrored LUNs are:

- Upper Controller (fas2050upper)
    - /vol/RG1FV1_SM/RP1.lun (lunid=0)
    - /vol/RG1FV2_SM/RP2.lun (lunid=3)
- Lower Controller (fas2050lower)
    - /vol/RG2FV1_SM/RP3.lun (lunid=4)
    - /vol/RG2FV2_SM/RP4.lun (lunid=5)

If we now issue a "rescan" for new devices and for VMFS volumes via VirtualCentre we will see that our lunids appear, Figure-11.19:

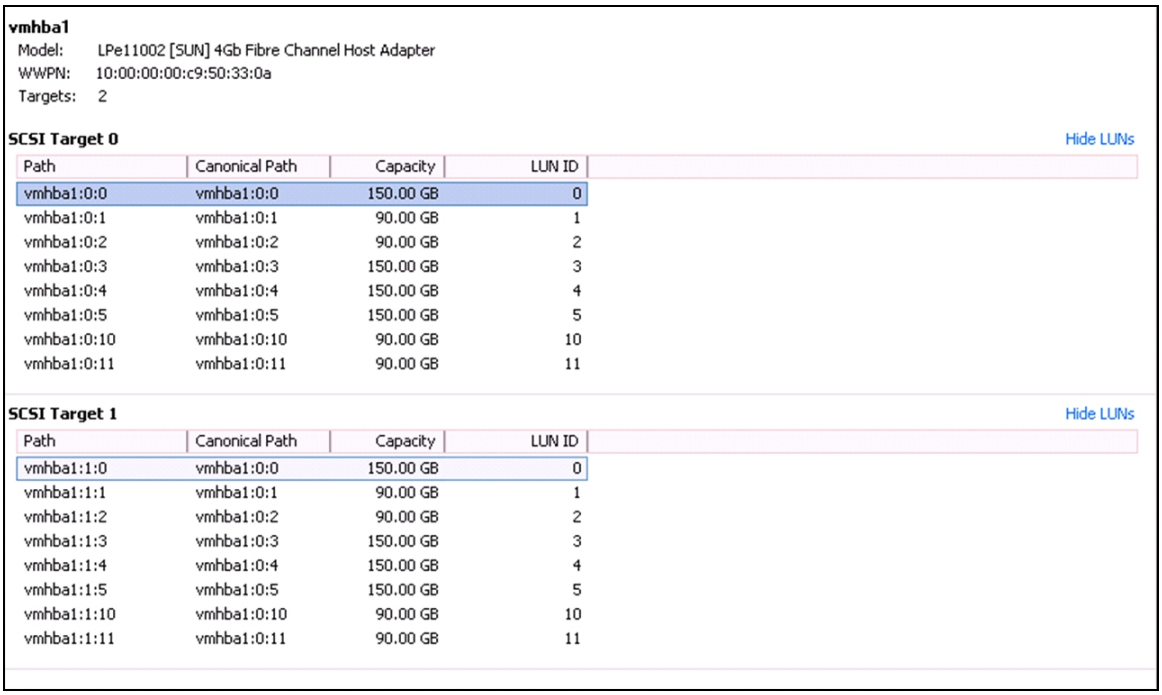


Figure 11.19 - ESX Rescan HBAs

It is also important at this stage to take a look at the VMFS datastores we can now see via the Storage screen in VirtualCenter, Figure-11.20:

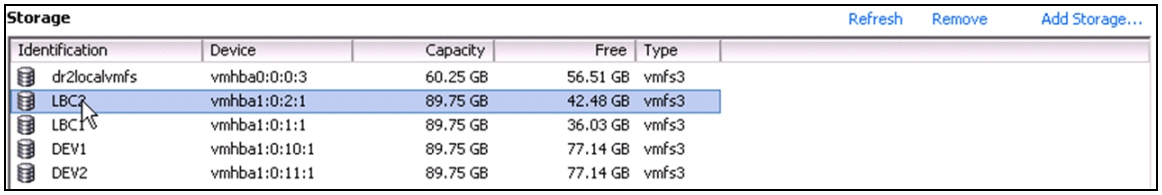


Figure 11.20 - VMFS Datastore Inventory

Please note in the "VirtualCenter Storage Datastores" Figure-11.20 we cannot see any new VMFS datastores even though we ran the rescan for both devices and VMFS volumes as per Figure-11.21:

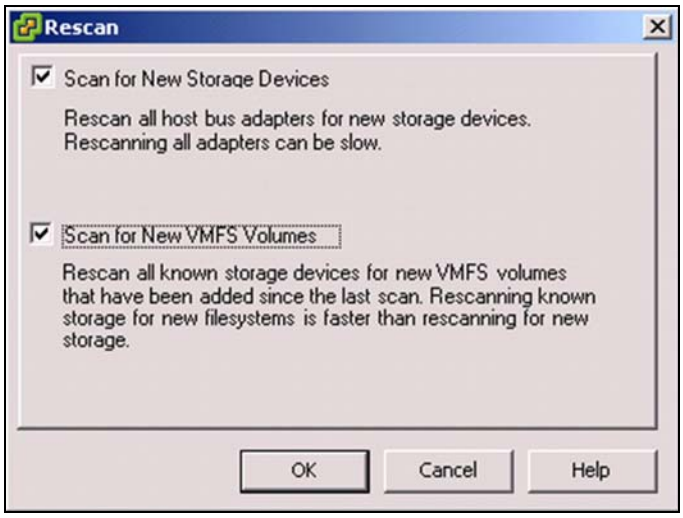


Figure 11.21 - Rescan HBAs

The reason we do not see new/additional datastores in our inventory at this point is because at this stage we have not broken off the relationship between the source and destination volumes at the storage level or in other words the replica copies we have access to at Site 2 are still presented to our ESX hosts as read-only LUNs which means we cannot access the VMFS datastores at this time.

Recall that this example is to show what happens when we break the relationship between a single source/destination pair and that the pair in question are the volumes RG1FV1 at Site 1 and RG1FV1_SM at Site 2. The replicated LUN contained within the volume RG1FV1_SM is called RP1.lun and contains the RP1 VMFS datastore.

We will now proceed to break off the replication link between the source and destination volumes. To achieve this we need to login to filerview on the Site 2 Upper controller and navigate to the "Manage SnapMirror" view, from there we drill down on the "Advanced" URL for the RG1FV1_SM volume shown previously in Figure-11.17.

The view that then appears is shown in Figure-11.22:

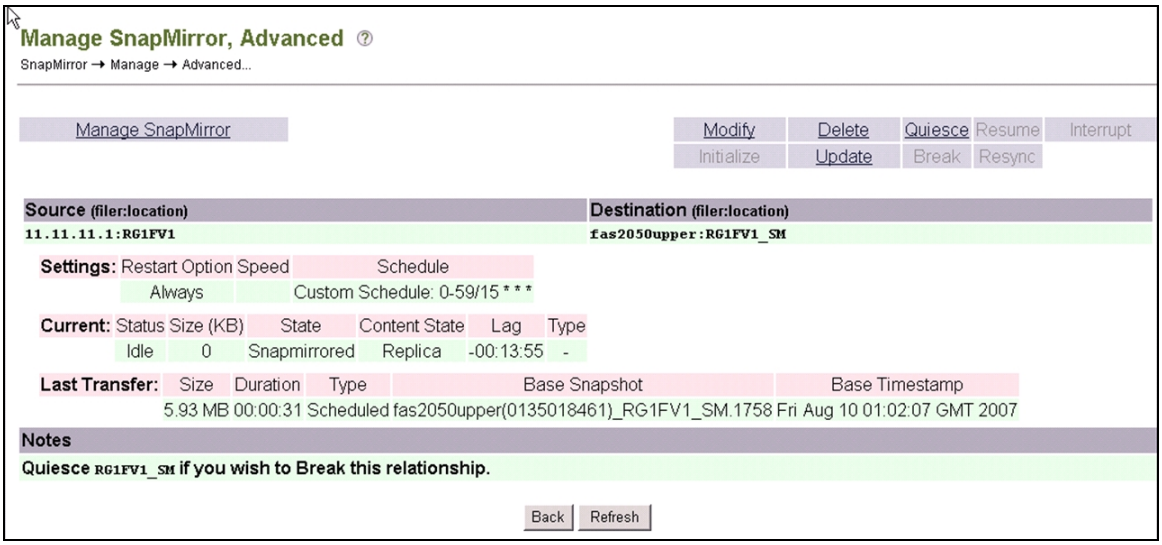


Figure 11.22 - Snapmirror Volume Advanced

Before we can break the relationship between the source and destination volumes we must tell SnapMirror to quiesce the link. The quiesce command instructs SnapMirror to complete any transfers which are currently in progress, but new transfers will not take place. This is especially useful when SnapMirror is operating in Synchronous or Semi-Synchronous mode. To quiesce a SnapMirror relationship, simply click the "Quiesce" drill down shown above. We will now be asked to confirm this request as follows in Figure-11.23:

Figure 11.23 - Quiesce Snapmirror Volume

Now that we have successfully "Quiesced" the RG1FV1_SM volume the "Manage SnapMirror" view shown below reflects this in the "State" field. Also note at this stage that the "Break" drill down is not activated as per Figure-11.24:

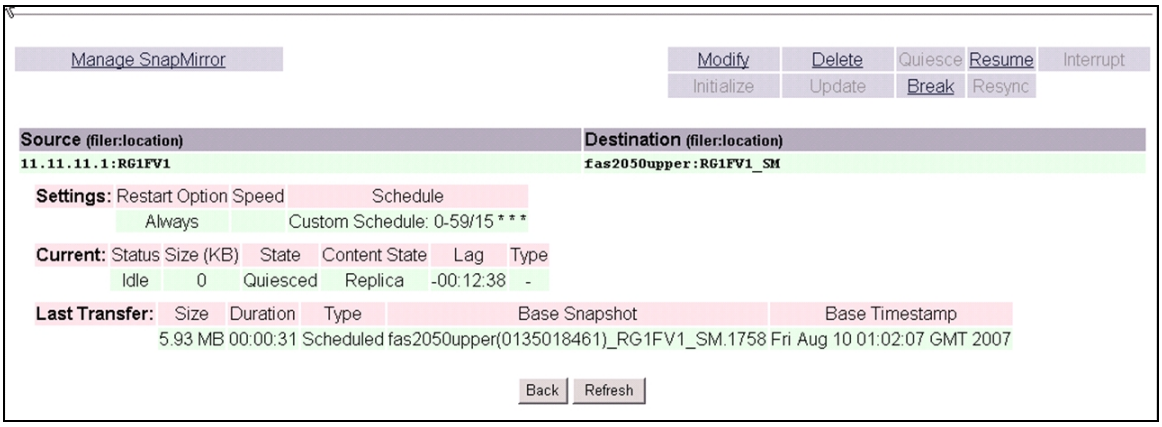


Figure 11.24 - Break Snapmirror Volume

Next step is to click the "Break" drill down and confirm that at this point we do wish to break the volume from it source, Figure-11.25:

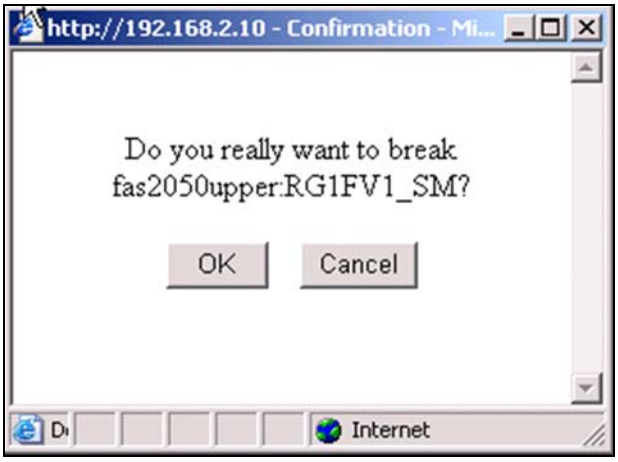


Figure 11.25 - Confirm Snapmirror Break

We have now successfully broken the RG1FV1_SM volume but in this instance broken does not mean something bad to us! If we review the "Manage SnapMirror" view shown below we can see that the "Quiesced" state has now changed to show "Broken-Off".

One important message also displayed in the "Manage SnapMirror" output is:

"snapmirror break: Destination RG1FV1_SM is now writable"

This message is fairly self explanatory. It is telling us that this volume, and hence any LUNs within it, are now read-write to any host that have access. In our case our ESX hosts now have read-write access to this volume and its LUNs.

It is worth noting in Figure-11.26 that the "Resync" drill down is now active, we will refer back to this later.

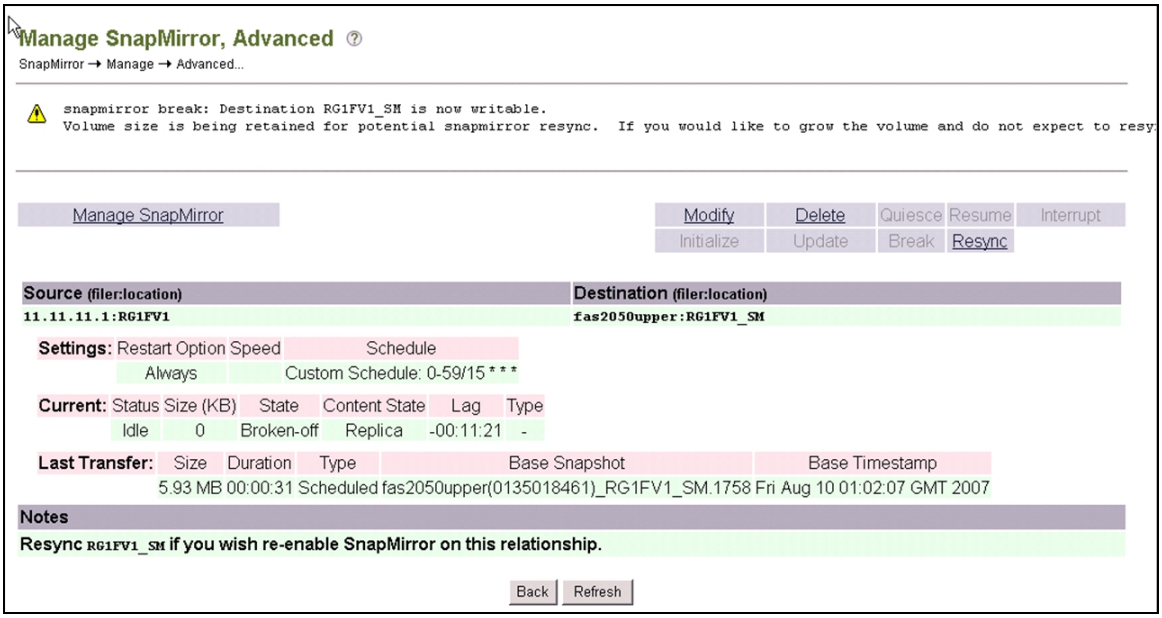


Figure 11.26 - Snapmirror Volume Broken-off

We know from our previous checks that all of the replicated LUNs were visible to our HBAs even when their volumes were read-only (check in Figure-11.27 below for LUN ids 0,3,4,5), so what has changed?

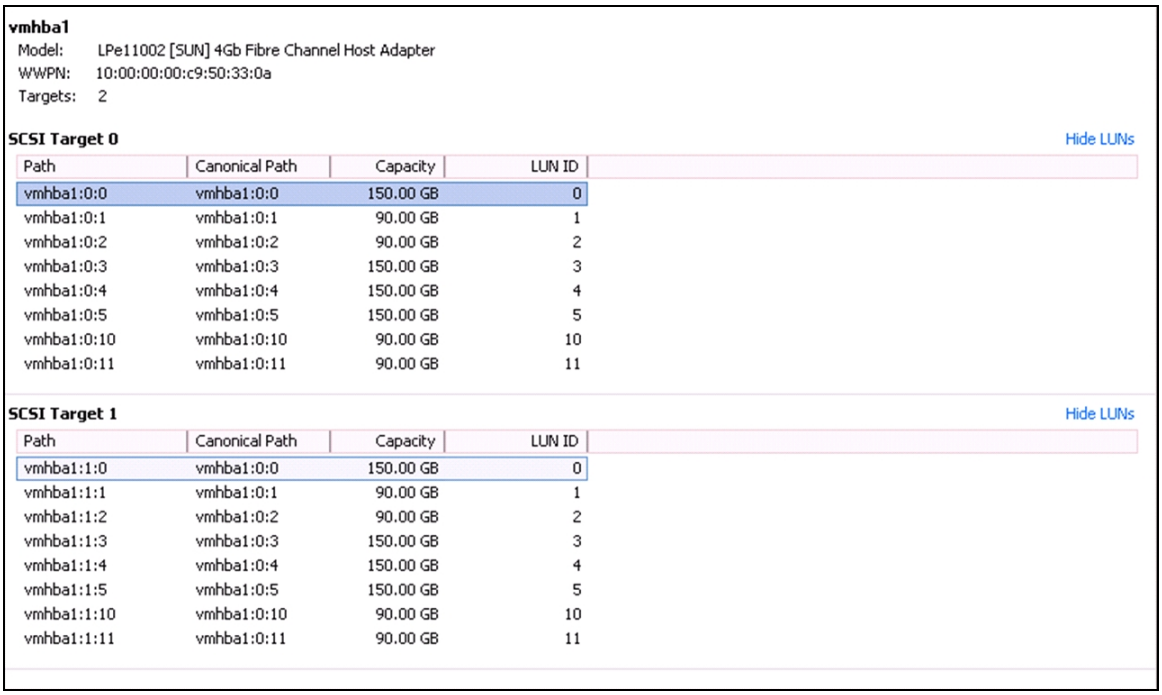


Figure 11.27 - Site 2 ESX Storage Adapter Example

If we now look again at our "Storage" view for one of the ESX hosts located in our Site 2 datacenter and click the "Refresh" drill down we will see that our datastore RP1 now appears in our inventory, shown in Figure-11.28:

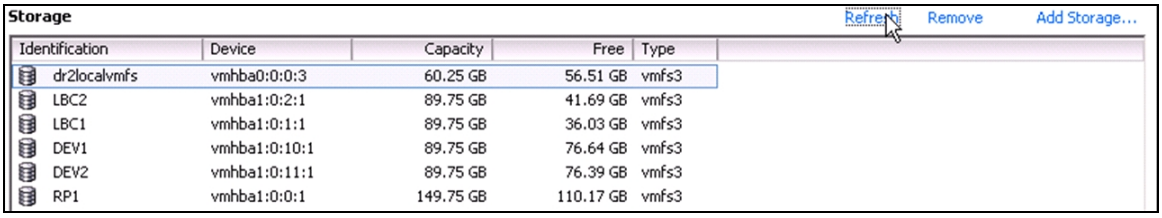


Figure 11.28 - VMFS Datastores Storage Inventory

Our datastore is now available in read-write mode and therefore we can now browse this datastore. At this point we should recall our VMware Infrastructure configuration settings mentioned earlier in this chapter. As is shown in the Figure-11.28 our datastore has appeared in the inventory with the same VMFS name as we gave it in Site 1, so why is this? Recall that we changed our advanced ESX LVM setting DisallowSnapshotLun from its default. To recap:

- Default: LVM.DisallowSnapshotLun=1
- Our Value: LVM.DisallowSnapshotLun=0

These settings are configured in VirtualCenter for each ESX host under "Configuration > Advanced Settings", Figure-11.29:

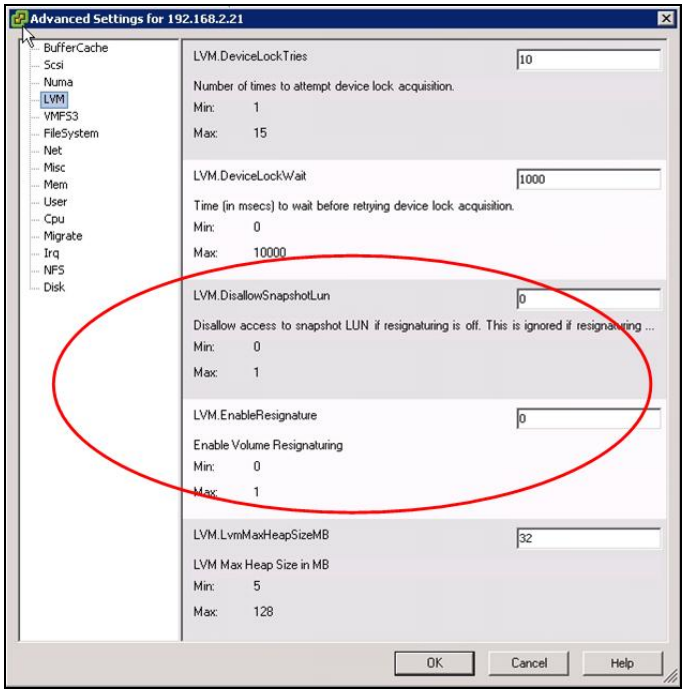


Figure 11.29 - VMware ESX Advanced LVM Settings

We know in our environment that the ESX hosts located in Site 2 will never be zoned into or connected to the storage systems located in Site 1 therefore in a failover situation we wish the VMFS datastores to come online at Site 2 using the same names that were given to them in Site 1.

Having the datastores appear with the same names is very useful when using scripts to automate the reconfiguration and startup of your virtual machines.

By setting LVM.DisallowSnapshotLun=0 we are telling ESX not to treat the LUN as a snapshot and to grant access to it using its original datastore name.

As a note of caution, it is worth noting, that by applying these settings, we cannot mount additional copies or clones of our LUNs in Site 2. Should you wish to mount multiple copies or clones of the same LUN, you should set LVM.EnableResignature to 1.

A common mistake or misunderstanding we have seen is that sometimes people assume you need to play with the LVM.EnableResignature setting to connect to replicated LUNs at your failover location. That is not the case and in architectures similar to the one used in this book you should also be using LVM.DisallowSnapshotLun=0 if your failover ESX hosts cannot access your source LUNs. Whilst we

have used the term "source LUN" here, this concept also applies to LUNs replicated with other replication methods, which may involve different terminology such as "R1 LUNs" or "Primary LUNs".

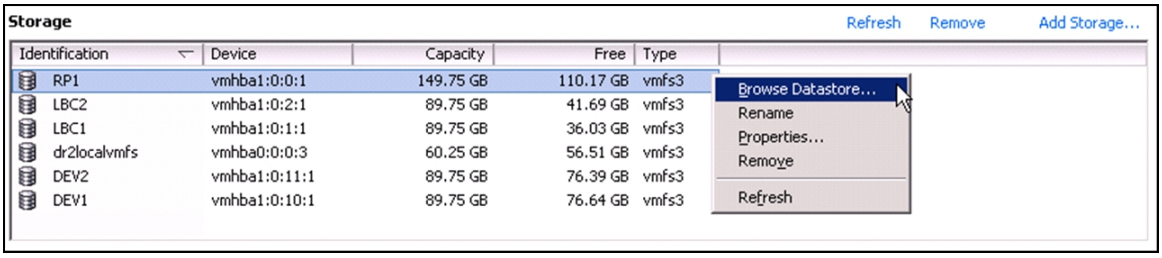


Figure 11.30 - Browse RP1 Datastore

Now that we can browse the datastore, Figure-11.30 we are able to drill down into a virtual machine folder and add it to our Site 2 inventory by right clicking on its vmx file as shown in Figure-11.31:

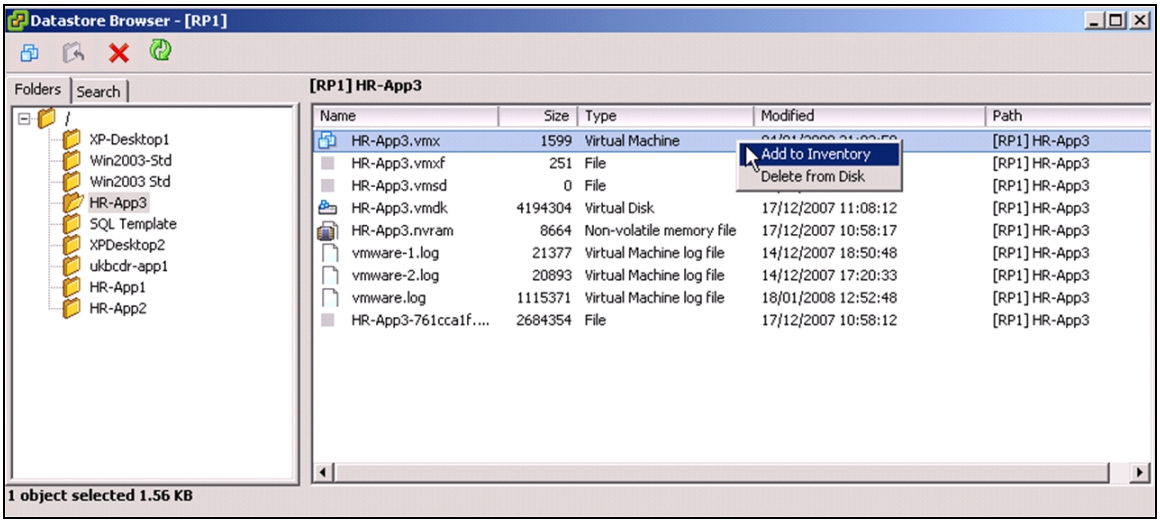


Figure 11.31 - Add Virtual Machine to Inventory

Once we have added the virtual machine to our Site 2 inventory we can try and power it on for the first time from our replicated LUN. Note at this point we are simply performing a power on test and we have not discussed accessing network resources or making changes to the virtual machines portgroup. These concepts are discussed elsewhere in this book. For now we simply illustrate that we can power on the virtual machine, Figure-11.32:

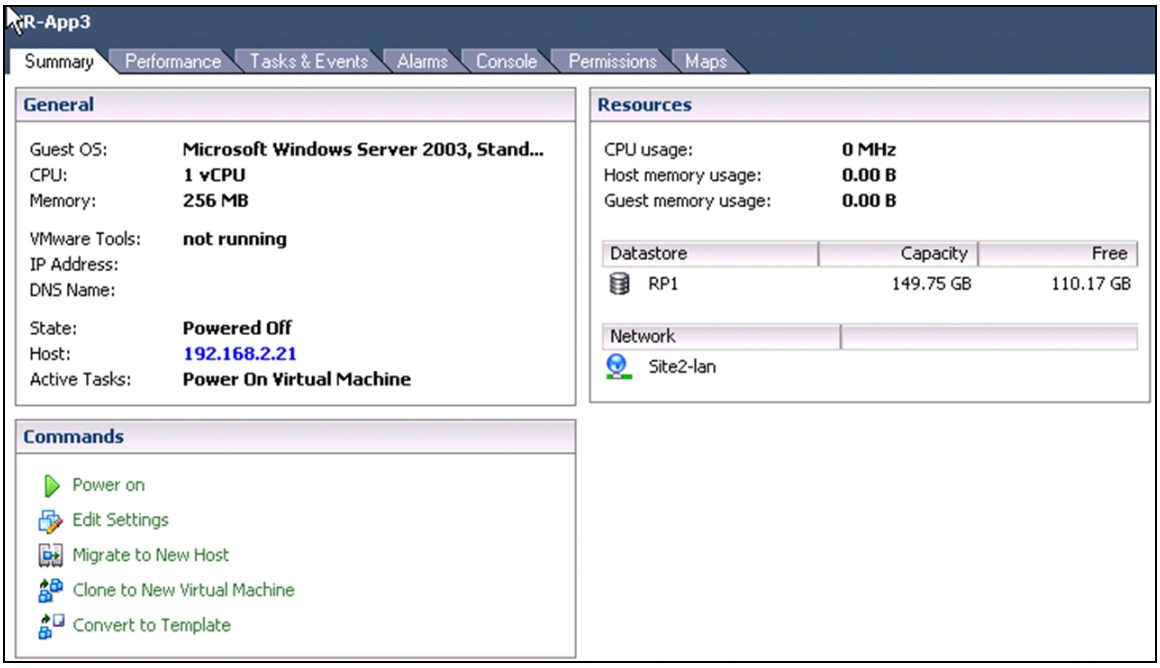


Figure 11.32 - Virtual Machine Summary

We can now see that our virtual machine powered on successfully, Figure-11.33:

Figure 11.33 - Virtual Machine Power On Completion

At this point we can login to the virtual machine console and verify that the virtual machine has started successfully. As has been shown in Figures-11.32 and 11.33 we are using "HR-App3" as our test virtual machine, we can now show the virtual machine console in Figure-11.34:

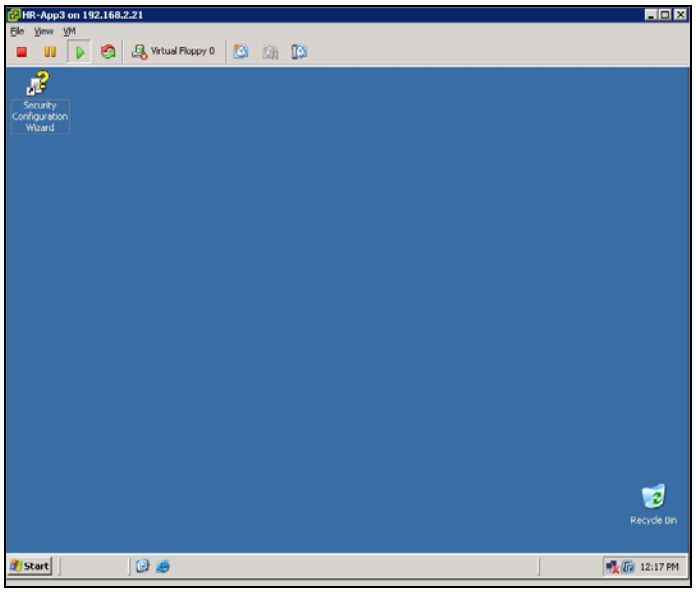


Figure 11.34 - Virtual Machine "HR-App3" Console

Now that we have successfully broken the relationship between the source and destination volumes and illustrated how to successfully bring a virtual machine online we can move on to the final concepts that need to be covered when splitting or breaking connections between volumes (LUNs). Those concepts are:

- Restoring connection between Site 1 and Site 2 volumes so that any changes to the Site 2 volume are discarded and the volume is re-synced with its Site 1 source once more.
- Failing back after a period of outage such that we now wish to KEEP the changes made to the Site 2 volume and have these changes pushed back to the Site 1 source so that the Site 1 volume will contain any changes we have made whilst we have been running our business in production at Site 2

Restore Replication from Site 1 to Site 2

First we will deal with the simple case of reconnecting our source and destination volumes back together again and letting the replication technology put the two volumes back in sync with each other. Any changes made in Site 2, since the SnapMirror relationship was broken-off, will be lost.

Let us first recap the entities we are using for this illustration:

- Site 2 Volume: RG1FV1_SM

- Site 2 Lun: /vol/RG1FV1_SM/RP1.lun
- Site 2 VMFS Datastore: RP1
- Site 2 Virtual Machine: HR-App3

- Site 1 Volume: RG1FV1
- Site 1 Lun: /vol/RG1FV1/RP1.lun
- Site 1 VMFS datastore: RP1
- Site 1 Virtual Machine: HR-App3

At this point our virtual machine is still powered on so we start by powering down that virtual machine and any other virtual machines we may have brought online from the RP1 datastore in Site 2.

Once the virtual machines are powered down for the RP1 datastore we can switch back to the storage system management utility, filerview and access the "Manage SnapMirror, Advanced" screen that we had used previously to break off the destination volume from its source, Figure-11.35:

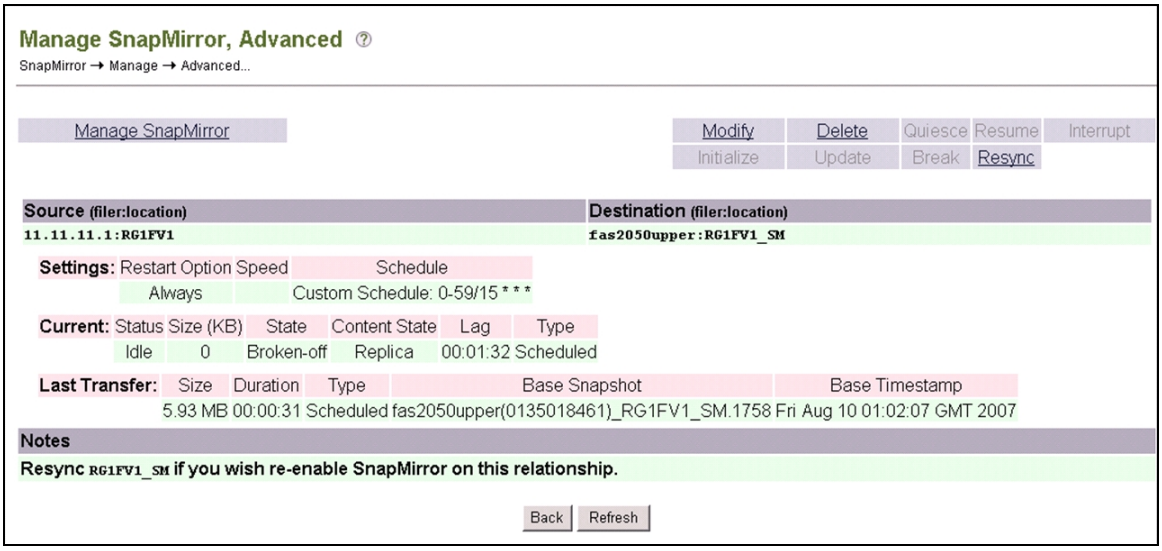


Figure 11.35 - Snapmirror Volume Resync

We mentioned earlier in this section to take note of the "Resync" drill down on this screen. This is now the time where we will utilize this feature. We need to allow our replication tool, SnapMirror, to resync this volume with its source volume (RG1FV1) located in Site 1.

To initiate this process we simply click the "Resync" drill down, shown in Figure-11.35, and will then be presented with the confirmation dialog box, Figure-11.36:

Figure 11.36 - Confirm Snapmirror Resync

At the prompt we click the "OK" button to let SnapMirror start the resync process with the source volume. Once we have selected the "OK" button we will be placed back at the "Manage SnapMirror, Advanced" screen once more, as shown in Figure-11.37:

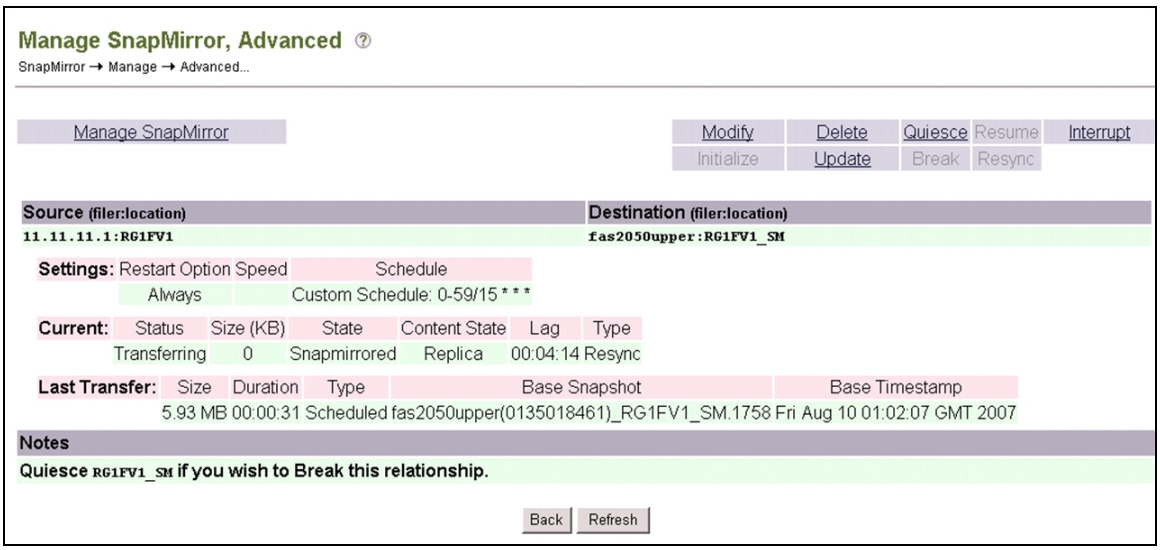


Figure 11.37 - Snapmirror Volume Resync Status

We can note in Figure-11.37 our "State" value has changed from "Broken-off" back to "Snapmirrored" and we can see our "Status" is shown as "Transferring" meaning we are now re-syncing this

destination volume at Site 2 (RG1FV1_SM) with any changes made to the source volume that have taken place since we broke the relationship between the two volumes.

Once our volumes are back in sync one of the changes that will happen implicitly is that the volume RG1FV1_SM in Site 2 will now change from being read-write to read-only as far as our ESX hosts are concerned.

If we switch back to VirtualCenter at Site 2 and pick one of our ESX hosts and select the storage screen and hit "Refresh" we can see that the RP1 datastore has once more vanished, see Figure-11.38, as the volume it resides in is now read-only so ESX will not grant access to it.

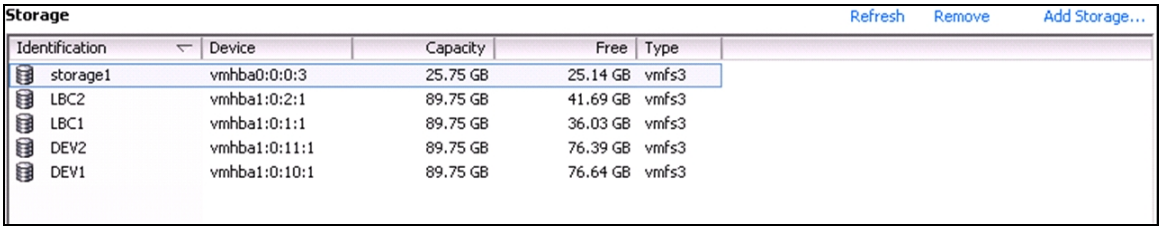


Figure 11.38 - VMFS Datastore Inventory

Failback Site 2 to Site 1

The case of failback is often viewed upon with dread when we are dealing with a scenario where we wish the source (Site 1) to accept the changes made to the broken-off volumes during our outage at Site 2. The purpose of this simple example in the context of this chapter is to simply illustrate at the storage level how this process can work. This process does not take into account the types of applications that could be running within your virtual machines or what kind of transactional recovery may or may not be needed post failback.

In this example we will make a simple change to our virtual machine, HR-App3, whilst it is powered on in Site 2 using the broken-off volume RG1FV1_SM and we will then resync the source and destination volumes as we did in the first example except this time we will reverse the flow so that the Site 1 volume is the volume that is changed when the resync occurs i.e we are failing back the change to the source.

As before the entities we are using for this illustration:

- Site 2 Volume: RG1FV1_SM
- Site 2 Lun: /vol/RG1FV1_SM/RP1.lun

- Site 2 VMFS datastore: RP1
- Site 2 Virtual Machine: HR-App3

- Site 1 Volume: RG1FV1
- Site 1 Lun: /vol/RG1FV1/RP1.lun
- Site 1 VMFS datastore: RP1
- Site 1 Virtual Machine: HR-App3

We will not go over the process of splitting and breaking off the source and destination volume as we have already covered this process in the last section. We will assume at this point our virtual machine is still powered on and is running off the split volume in Site 2.

We will make a very simple addition to the desktop of our virtual machine, we will start with the default desktop, Figure-11.39:

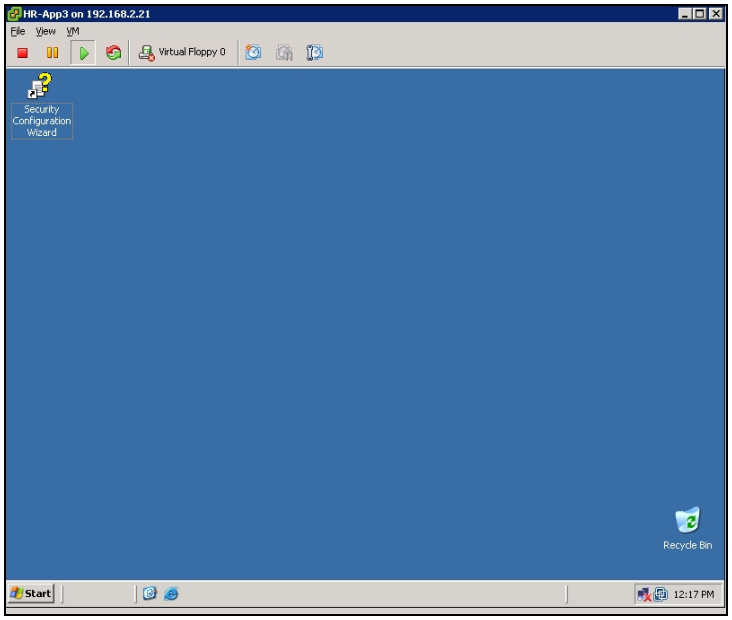


Figure 11.39 -Site 2 Virtual Machine Default Desktop

We will now add a new folder to this desktop called "Failback Test", Figure-11.40:

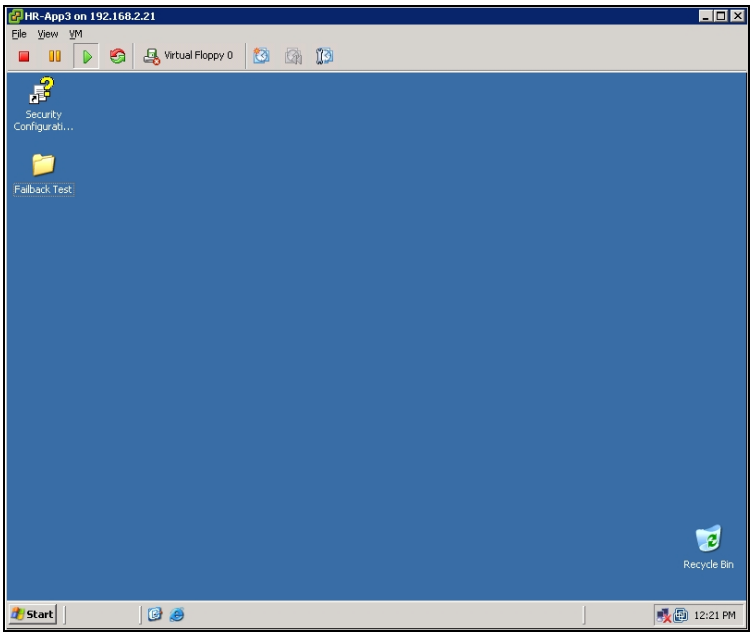


Figure 11.40 - Site 2 Virtual Machine Modified Desktop

Now we have our change, albeit a very basic one, we can begin the process of replicating back to the primary site.

Using SnapMirror, we can initiate a "Re-Sync" operation. This will transfer all changes applied to the destination volume in Site 2, since the last SnapMirror update, back to the original source volume in Site 1.

This does, however, assume that Site 1 was not physically destroyed and that the Storage System and data in Site 1 have been recovered successfully.

To begin the SnapMirror re-sync operation, we will issue the following command on the Storage Controller in Site 1:

```
snapmirror resync -S fas2050upper: RG1FV1_SM  fas2020upper:
RG1FV1
```

If the data stored in the Storage System in Site 1 was not recovered, or if we are now replicating to a new NetApp Storage System, we would create a new Flexible Volume on the new Storage Controller, and use an initialize operation on the new Storage Controller in Site 1:

```
Snapmirror initialize –S fas2050upper:RGFV1_SM
fas2020upper:RG1FV1
```

Once the re-sync or initialize operation has finished transferring changes back to Site 1, we can power down our virtual machine.

Now that the virtual machine is powered down, we will initiate a SnapMirror update to transfer any remaining changes (which occurred during the power down). This can be done by issuing the following command on the storage controller in Site 1:

```
snapmirror update -S fas2050upper: RG1FV1_SM  fas2020upper: RG1FV1
```

Now that the update has completed, we must make the volume in Site 1 available for writes, and ready to resume replication from site 1 to site 2.

On the storage controller in site 1, we must issue a break command:

```
snapmirror break RG1FV1
```

To begin replicating from site 1 to site 2, restoring our original SnapMirror configuration, we must issue the following command on the storage controller in site 2:

```
snapmirror resync RG1FV1_SM
```

Once the Site 1 volume, RG1FV1, has been updated with all of the changes made to the Site 2 volume, RG1FV1_SM we can switch back to the VirtualCenter console at Site 1 and power on our virtual machine "HR-App3" on the ESX host 192.168.1.8 and check to see if our change made over in Site 2 has found its way back through the storage replication layer and is now present in our failed back virtual machine, Figure 11.41:

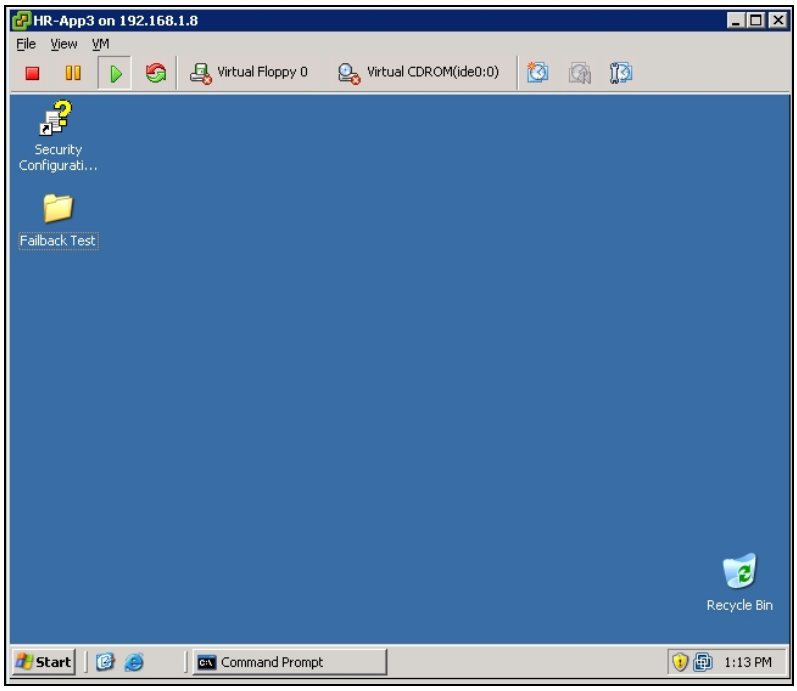


Figure 11.41 -Site 1 Virtual Machine Desktop

As we see in Figure-11.41 the folder "Failback Test" is present on the virtual machines desktop meaning that the volume located in Site 1 is now an up to date copy of the volume we split at Site 2. At this point we assume our failback test has been a success and we can resync the volumes using our original flow (Site 1 to Site 2) by simply logging back onto our Site 2 storage management interface, navigating to our "Manage Snapmirror, Advanced" screen for the volume RG1FV1_SM and hitting the "Resync" drill down as we have shown before in Figure-11.34.

Planned Failover Testing using FAS Storage

FAS Setup for Planned Failover Testing

One of the most important aspects of any successful BCDR design is the ability to conduct reliable and realistic planned failover tests without affecting the core replication architecture.

Most storage solutions available today have the capability to create writeable snapshot based copies of LUNs that may be part of a replicated pair. Depending on your storage solution the name given to this entity will differ. Some common terms are FlexClone, business continuity Volume (BCV) or Snap Clone.

With the FAS storage solution used in our infrastructure we make use of FlexClone volumes to create writeable copies of our replicated volumes (which contain our LUNs) which are then used for our testing simulations.

In this section we will walk through the creation of such a volume and show how the LUN contained within it is represented in VirtualCenter at the various stages of its creation.

To provide a more detailed overview we should say that a FlexClone volume is a writable point-in-time image of a FlexVol™ volume or another FlexClone volume.

FlexClone volumes add a new level of agility and efficiency to storage operations. They take only a few seconds to create and are created without interrupting access to the parent FlexVol volume.

FlexClone volumes use space very efficiently, leveraging the Data ONTAP architecture to store only data that changes between the parent and clone. This provides huge potential cost savings through reduced hardware acquisition costs, as well as space and energy. In addition to all these benefits, clone volumes have the same high performance as other kinds of volumes.

FlexClone volumes leverage Snapshot functionality, meaning that a FlexClone volume will read blocks owned by its parent volume, and write new or changed blocks to new space.

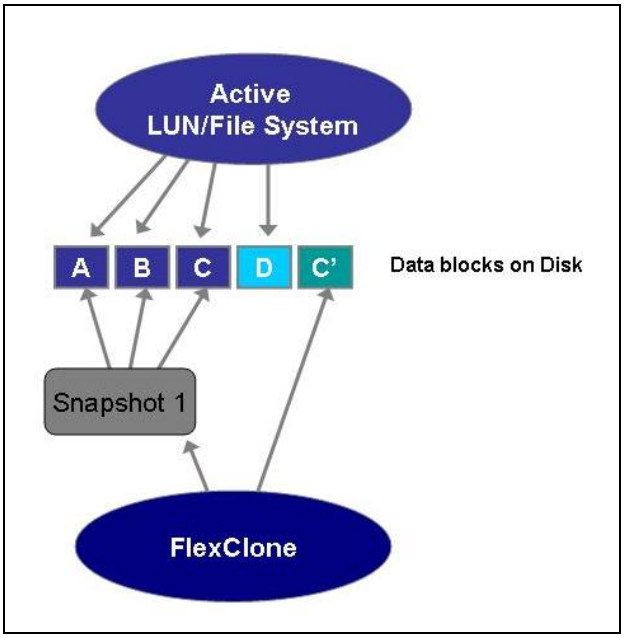


Figure 11.42 - FlexClone Initial Creation

In the Figure-11.42 example, blocks A, B & C are "owned" by the active LUN or File System, but are also used by snapshot 1, which is the basis for the FlexClone Volume. Block D is "owned" exclusively by the active LUN or File System, and block C' is owned exclusively by the FlexClone Volume.

A FlexClone volume can also be "split" from its parent, meaning that shared blocks will be copied in the background to new space, resulting in the parent and clone each owning a full copy of the blocks. Figure 11.43 shows this.

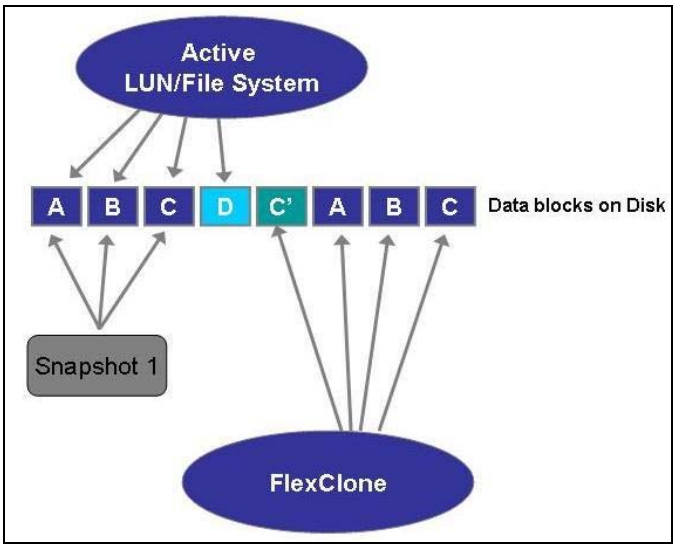


Figure 11.43 - FlexClone Data Blocks After Split Operation

FlexClone volumes have all the capabilities of FlexVol volumes, which means they can grow and shrink on the fly. You can Snapshot a FlexClone volume, you can replicate a FlexClone volume, and you can even Clone a FlexClone volume. This also means that FlexClone volumes get the same access to resources as any other volume, unless administrators wish to regulate that, using FlexShare to assign priorities on a per volume basis.

For more information on FlexShare, please see the "FlexShare™ Design and Implementation Guide" - http://media.netapp.com/documents/tr_3459.pdf

For more information on FlexClone Volumes, please see "A Thorough Introduction to FlexClone™ Volumes" - http://media.netapp.com/documents/tr_3347.pdf

SnapMirror typically replicates data from one Flexible Volume to another. As such, SnapMirror destination volumes are read only, until the SnapMirror relationship is broken.

This provides several challenges to disaster recovery testing. Typically customers are forced to choose between stopping replication (leaving the primary site exposed), or deploying additional capacity at the alternate site, to store additional copies of replicated data.

FlexClones offers a third choice – create space efficient clones with no interruption to data replication.

Our design is based on the premise that a separate instance of VirtualCenter has been deployed at the alternate site (Site 2 in our case), the FlexClone volumes can then be presented to ESX hosts within Site 2.

If that volume contains a LUN, formatted with VMFS, then the ESX servers will be able to mount the VMFS datastore. If the volume is used as an NFS datastore, then it can be exported, and mounted by the ESX servers. If the volume contains LUNs used as Raw Device Maps (RDMs), then these can be presented to the ESX servers, and later added to virtual machines.

In addition, FlexClone volumes can be used to register virtual machines into VirtualCenter in the alternate site – provided that all the virtual machine files are made visible to the ESX servers in the alternate site.

Using a generic example (Primary Site and DR Site), we have a VMFS datastore named "Datastore1", it resides on a NetApp LUN, inside a NetApp FlexVol. This FlexVol is replicated, using SnapMirror, the BCDR site.

We can take a FlexClone of the SnapMirror destination volume, and present the LUN to the VMware ESX hosts in the BCDR site. The VMware ESX hosts will see "Datastore1" contained on the new LUN, and you can register and power up the virtual machines in "Datastore1" , see Figure-11.44:

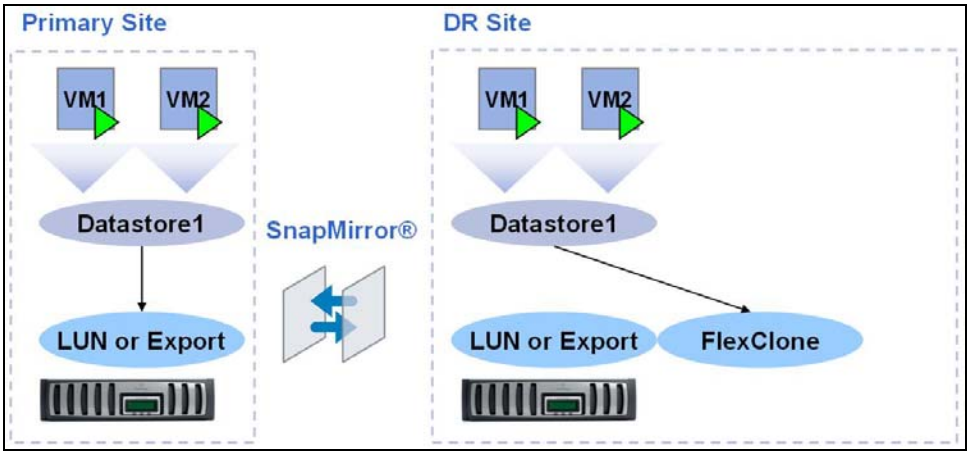


Figure 11.44 - Flexclone Usage

Now the virtual machines are registered, the FlexClone volume can be destroyed. At this point "Datastore1" will be offline. In the event of a disaster affecting the primary site, the SnapMirror destination volume can be made available to the VMware ESX hosts in the BCDR site, Figure 11.45. They will see "Datastore1" now online (albeit via a different canonical path).

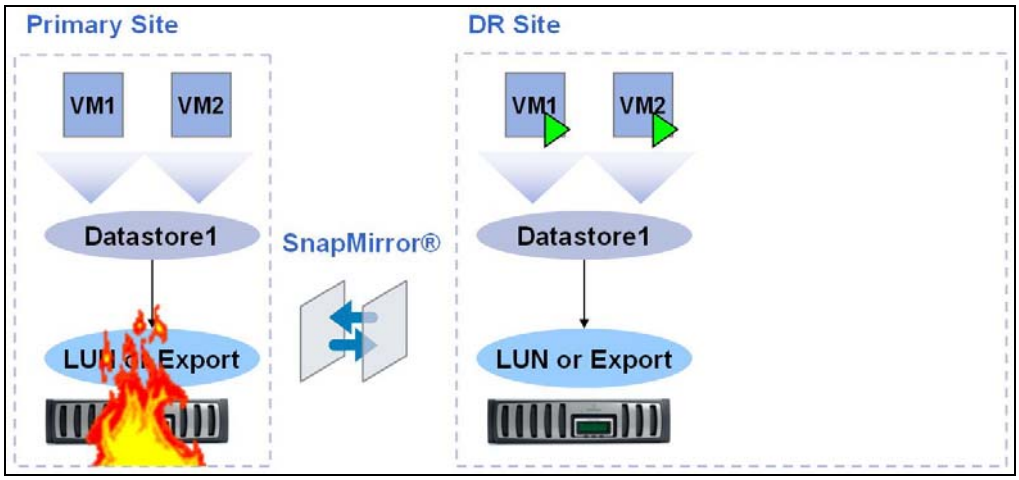


Figure 11.45 - Primary Site Outage

Once Datastore1 is online, the virtual machines, can be immediately powered up. They have already been registered, and previously tested.

FlexClone Presentation within VMware Infrastructure

To illustrate the use of FlexClones within the VMware environment described in this book for the purposes of BCDR testing we first need to choose work on the FAS storage system located at our Site 2 datacenter as this storage system contains the replicated copies of our Site 1 datastores.

The Figure 11.46 shows the volumes available for us to select at Site 2, in this example we will be working against the "Upper" controller:

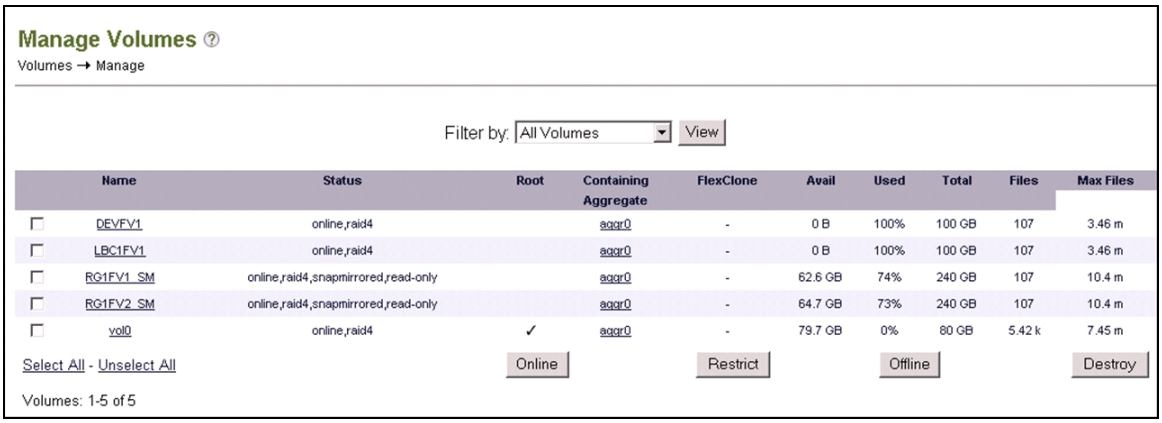


Figure 11.46 - Manage Volumes (Site 2 Upper Controller)

Recall from Site 1 our replicated volumes all begin with "R", hence in the above output these are shown as:

- RG1FV1_SM
- RG1FV2_SM

The _SM suffix is part of our naming convention and is short for "SnapMirror" to indicate that these volumes are the right side of a left > right replicated pair i.e the copies that will be used in the event of Site 1 suffering a major outage or disaster. The couterpart volumes at Site 1 have the same names minus the _SM suffix. It is important when deciding upon your storage naming convention to choose a scheme that makes sense to you as the administrator.

For the purpose of this example we will create a FlexClone volume for the **RG1FV1_SM** volume. Now that we have selected this volume as our candidate we can switch to the "Manage LUNs" view, Figure 11.47, to see what LUNs this volume may contain:

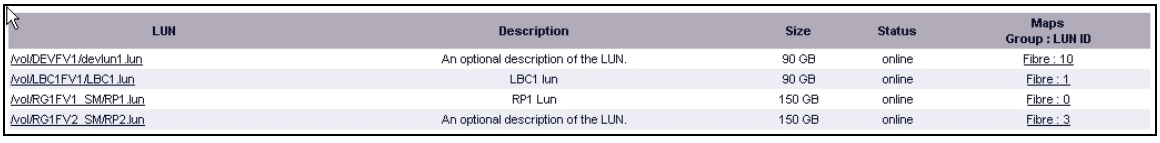
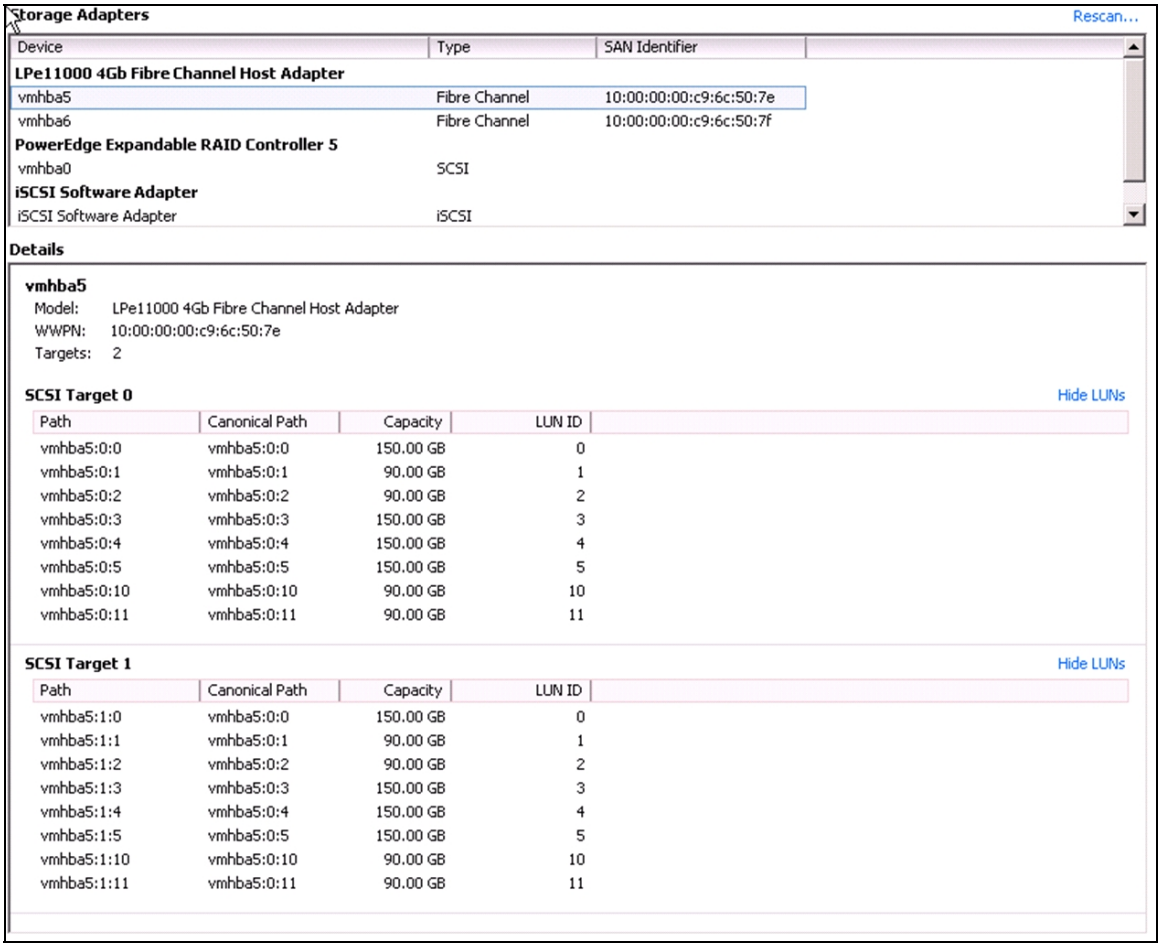


Figure 11.47- Site 2 FAS Manage LUNs (Upper Controller)

Using the path as the indication of which volume a LUN belongs to we can see that our volume, RG1FV1_SM contains a single LUN with a lunid of 0:

- /vol/RG1FV1_SM/RP1.lun (lunid=0)

Before we make any changes to our storage configuration we should at this stage take a look at the VirtualCenter "Storage Adapters" output, Figure-11.48, for one of our ESX hosts that reside in Site 2 to see if VMware ESX can actually see this LUN:



Storage Adapters Rescan...

Device	Type	SAN Identifier
LPe11000 4Gb Fibre Channel Host Adapter		
vmhba5	Fibre Channel	10:00:00:00:c9:6c:50:7e
vmhba6	Fibre Channel	10:00:00:00:c9:6c:50:7f
PowerEdge Expandable RAID Controller 5		
vmhba0	SCSI	
iSCSI Software Adapter		
iSCSI Software Adapter	iSCSI	

Details

vmhba5
Model: LPe11000 4Gb Fibre Channel Host Adapter
WWPN: 10:00:00:00:c9:6c:50:7e
Targets: 2

SCSI Target 0 Hide LUNs

Path	Canonical Path	Capacity	LUN ID
vmhba5:0:0	vmhba5:0:0	150.00 GB	0
vmhba5:0:1	vmhba5:0:1	90.00 GB	1
vmhba5:0:2	vmhba5:0:2	90.00 GB	2
vmhba5:0:3	vmhba5:0:3	150.00 GB	3
vmhba5:0:4	vmhba5:0:4	150.00 GB	4
vmhba5:0:5	vmhba5:0:5	150.00 GB	5
vmhba5:0:10	vmhba5:0:10	90.00 GB	10
vmhba5:0:11	vmhba5:0:11	90.00 GB	11

SCSI Target 1 Hide LUNs

Path	Canonical Path	Capacity	LUN ID
vmhba5:1:0	vmhba5:0:0	150.00 GB	0
vmhba5:1:1	vmhba5:0:1	90.00 GB	1
vmhba5:1:2	vmhba5:0:2	90.00 GB	2
vmhba5:1:3	vmhba5:0:3	150.00 GB	3
vmhba5:1:4	vmhba5:0:4	150.00 GB	4
vmhba5:1:5	vmhba5:0:5	150.00 GB	5
vmhba5:1:10	vmhba5:0:10	90.00 GB	10
vmhba5:1:11	vmhba5:0:11	90.00 GB	11

Figure 11.48 - Site 2 Storage Adapters Example

It is clear that we can see lunid 0 in the Figure-11.48 output so we can see the LUN. As the LUN is part of a replicated pair that is active and by active we mean replicated the LUN is read only at this point so any VMFS volumes contained within it are not visible. We know that this LUN should contain a VMFS datastore named **RP1** so we can check for its existence via the VirtualCenter "Storage" screen, Figure 11.49:

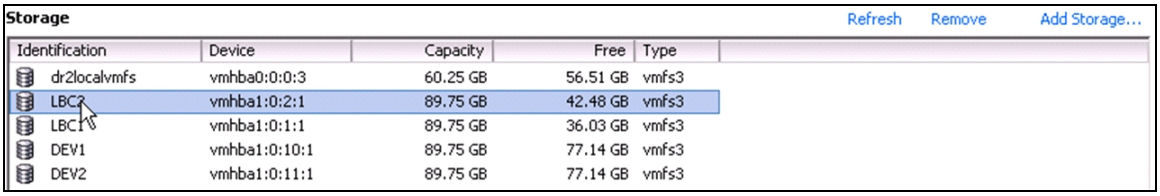


Figure 11.49 - Site 2 VMFS Datastores

As we can see, even if we "Refresh" this view no datastore named RP1 appears. This is working as designed as we should not be presented with the datastore whilst the LUN is read-only to VMware ESX.

We can now proceed to create our FlexClone volume on the FAS storage device for the existing volume RG1FV1_SM. To help us complete this task we use the FlexClone wizard.

The first step is to provide a name for the FlexClone volume, select the parent volume (the Flexible Volume you wish to clone RG1FV1_SM in our case), and select a space reservation guarantee for the FlexClone volume. Figure 11.50 depicts this.

In this case, we want our FlexClone to only consume space for the changes we will make to it, so we have selected a guarantee of none.

For more information on space guarantees, please see "Thin Provisioning in a NetApp SAN or IP SAN Enterprise Environment" - http://media.netapp.com/documents/tr_3483.pdf

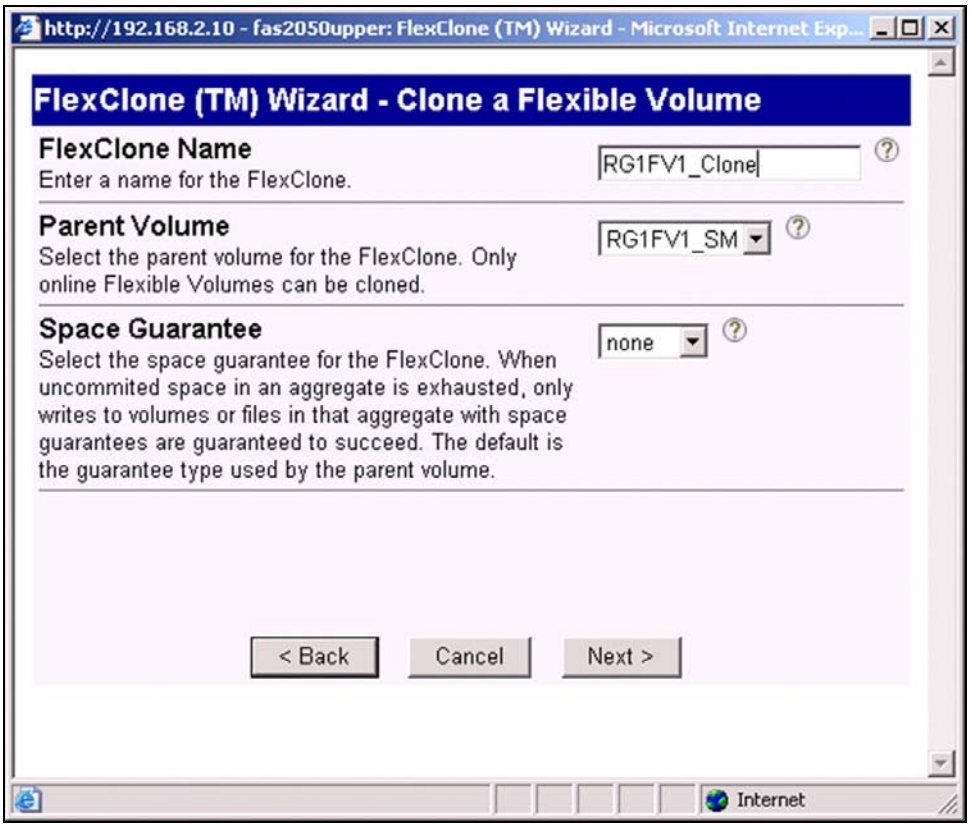


Figure 11.50 - Flexclone Creation

To expand our existing naming convention whenever we create a FlexClone volume we simply change the suffix from _SM to _Clone. A FlexClone volume based on RG1FV1_SM will therefore be RG1FV1_Clone as shown in Figure 11.50.

In the next stage of the wizard, Figure 11.51, you can select a snapshot to use as the basis for the FlexClone, or you can create a new Snapshot, to base the FlexClone on the current point in time.

Note, if the parent volume is a SnapMirror destination volume, it will be read-only until the SnapMirror relationship is broken-off. For this reason, you cannot select “create new” in the FlexClone wizard. Instead, you must either create a new snapshot on the SnapMirror source volume, on the primary storage system, or create the FlexClone based on an existing Snapshot, such as the last SnapMirror update snapshot.

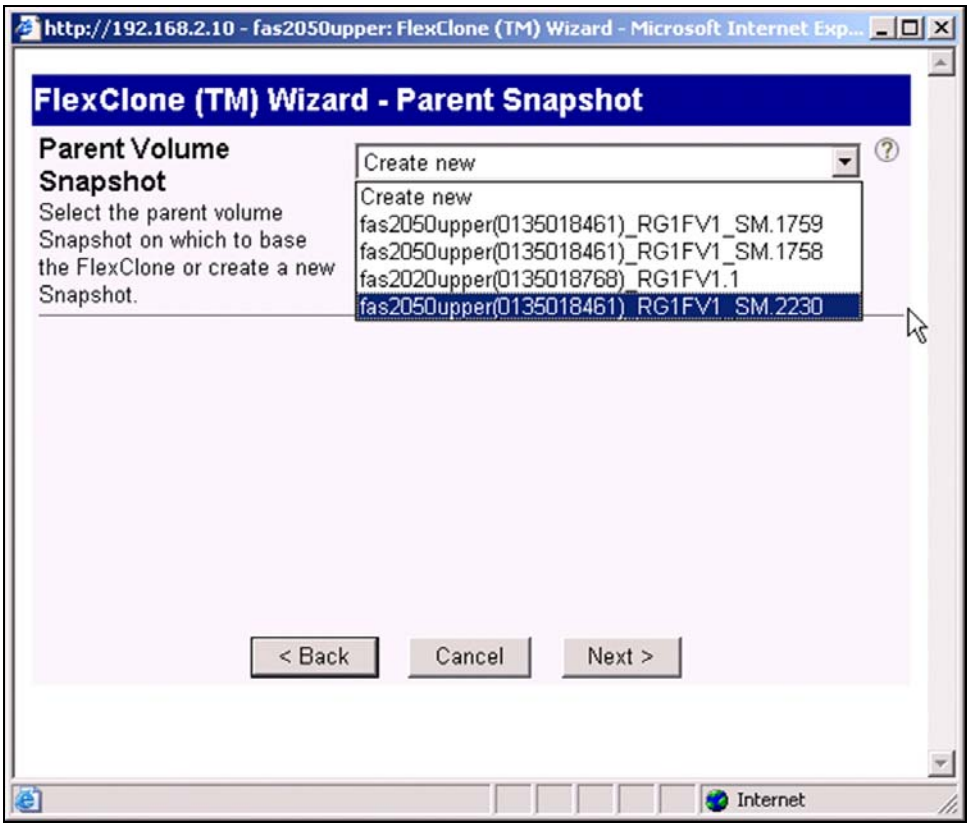


Figure 11.51 - Select Parent Snapshot Volume

The final stage of the wizard, Figure 11.52, is to confirm the parameters you have supplied, and click "commit" to create the FlexClone volume.

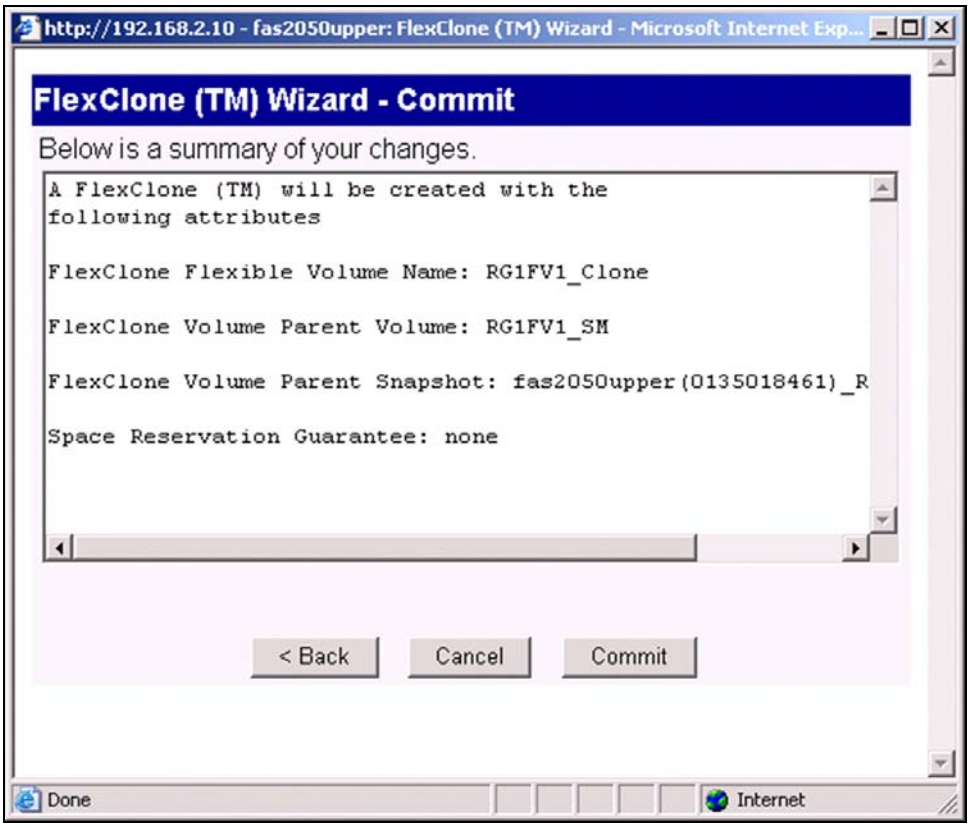


Figure 11.52 - Commit Flexclone Changes

We can now revisit the "Manage Volumes" screen to verify that our FlexClone volume has been created successfully, Figure 11.53.

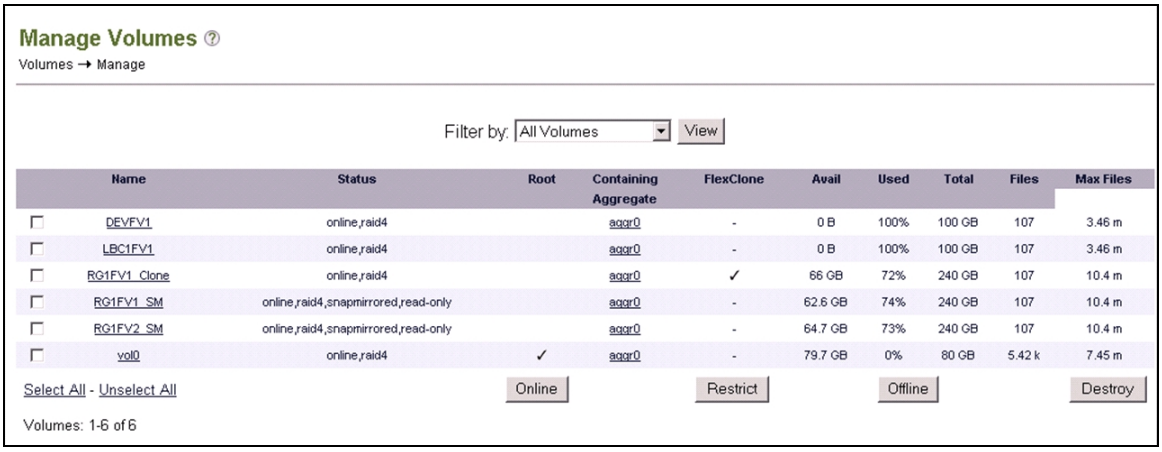


Figure 11.53 - Manage Volumes Post Flexclone Creation

We now have a new volume in our list called RG1FV1_Clone so our FlexClone has now been created successfully.

As our FlexClone volume contains a LUN, it must be brought online and mapped to an initiator group (igroup) before it will be visible to our VMware ESX hosts.

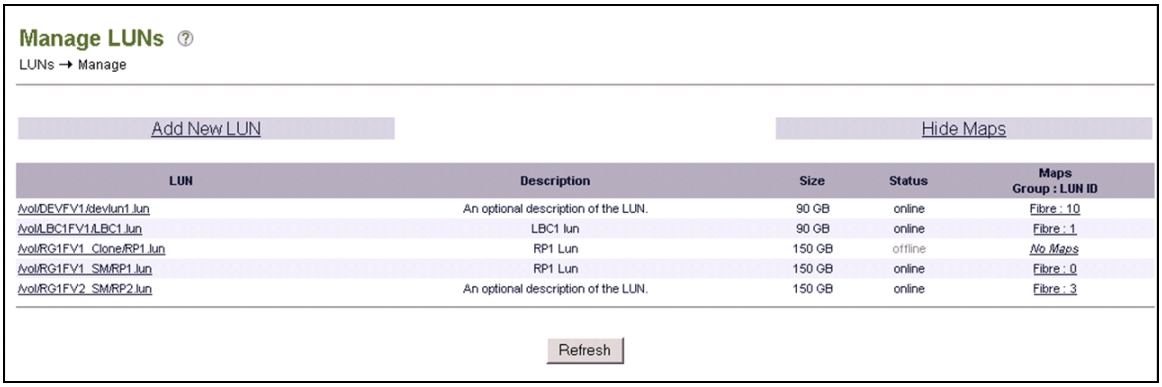


Figure 11.54 - Offline FlexClone Volume LUN

In Figure 11.54 we can see that the LUN contained within our FlexClone volume is "offline" and defined as:

- /vol/RG1FV1_Clone/RP1.lun

It is also worth noting this lun in Figure 11.54 is not showing an initiator group or lunid mapping value, instead we see the entry "No Maps".

We must now bring the LUN "online" and also map it to our "Fabric" initiator group as we are presenting these LUNs over fibre channel fabric. From Filerview we select LUNs – Manage, and click on the LUN contained in the FlexClone volume.

First we will map the LUN to an initiator group by clicking on the "No Maps" URL shown in Figure 11.54, this will present the screen output shown in Figure 11.55:

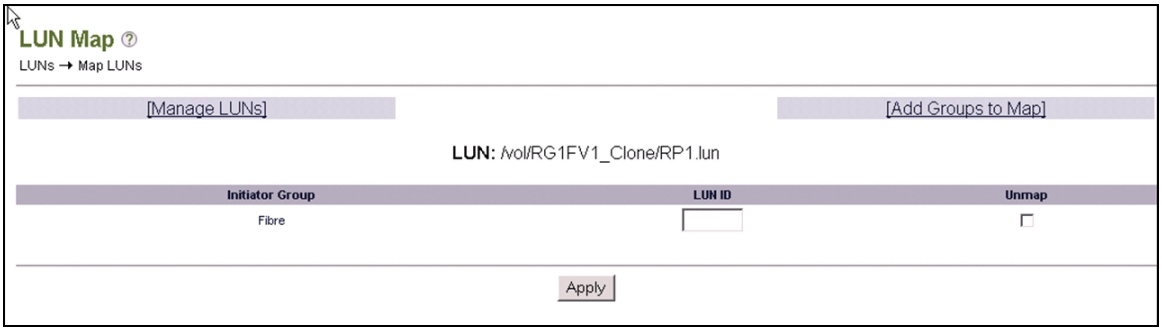


Figure 11.55 - LUN Mapping

To map our LUN we now select the "Add Groups to Map" link show in Figure-11.55 and we will then be presented with output as per Figure 11.56:

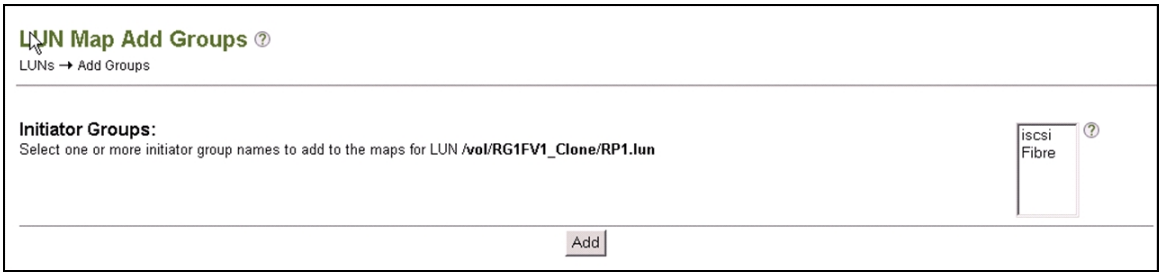


Figure 11.56 - Add Groups

We can see in Figure-11.56 that we only have two groups to select from, iSCSI or Fibre, we will select the Fibre group as, in this case, we wish to present our clone LUN to our ESX servers using Fibre Channel. The Fibre initiator group contains a list of the WWNs we need to present to, in this case the HBA WWNs for the ESX hosts contained within our Site 2 datacenter. Figure-11.57 shows the initiator groups screen.

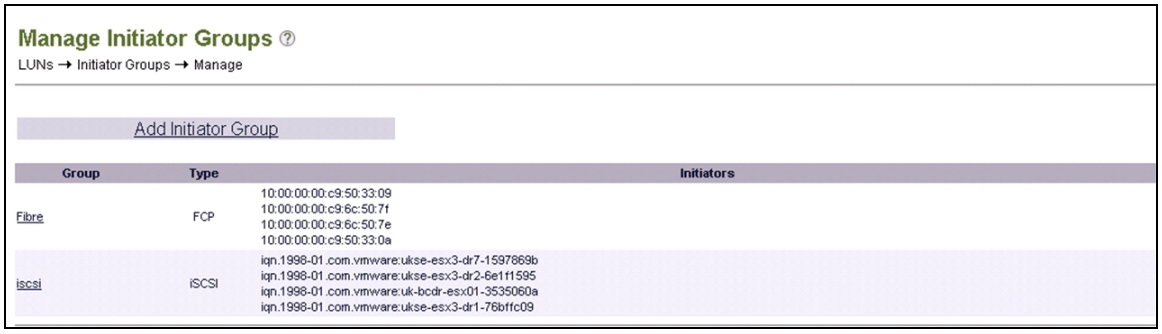


Figure 11.57 - Initiator Groups

The final step is to now bring the LUN online. We revert back to the "Manage LUNs" screen show previously in Figure-11.54 and this time drill down on the link that represents our LUN name "/vol/RG1FV1_Clone/RP1.lun". This will take us to the "Modify LUN" screen for this LUN shown in Figure 11.58:

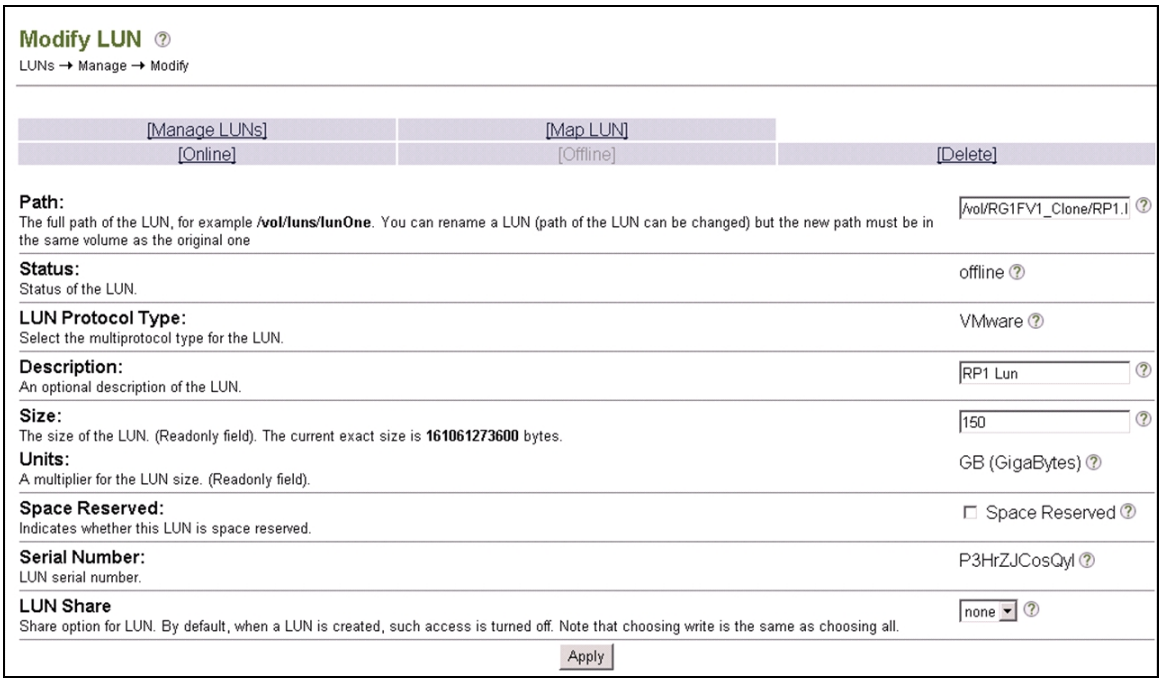


Figure 11.58 -Modify LUN

Clicking the "Online" button on this page shown in Figure 11.58 followed by the "Apply" button will now bring the LUN online as shown in Figure 11.59:

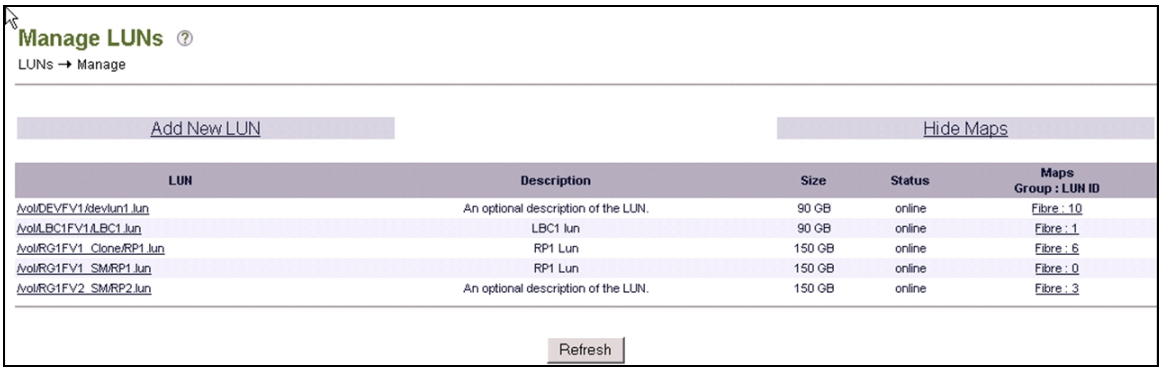


Figure 11.59 - LUN Online Status

Once the LUN is online, and mapped to our ESX servers' we need to issue a rescan of our storage adapters via VirtualCenter, using the "Rescan" button in the top right hand corner of the Configuration, Storage Adapters tab in VirtualCenter, for each VMware ESX host.

Note in Figure 11.59 that our LUN has a lunid value of 6. Reason for knowing this will become important shortly. The process of issuing rescans can also be achieved via the VMware APIs for VMware Infrastructure.

We now change to the VirtualCenter > Storage Adapters screen for one of our VMware ESX hosts, see Figure 11.60, there is no lunid = 6 shown at this time:

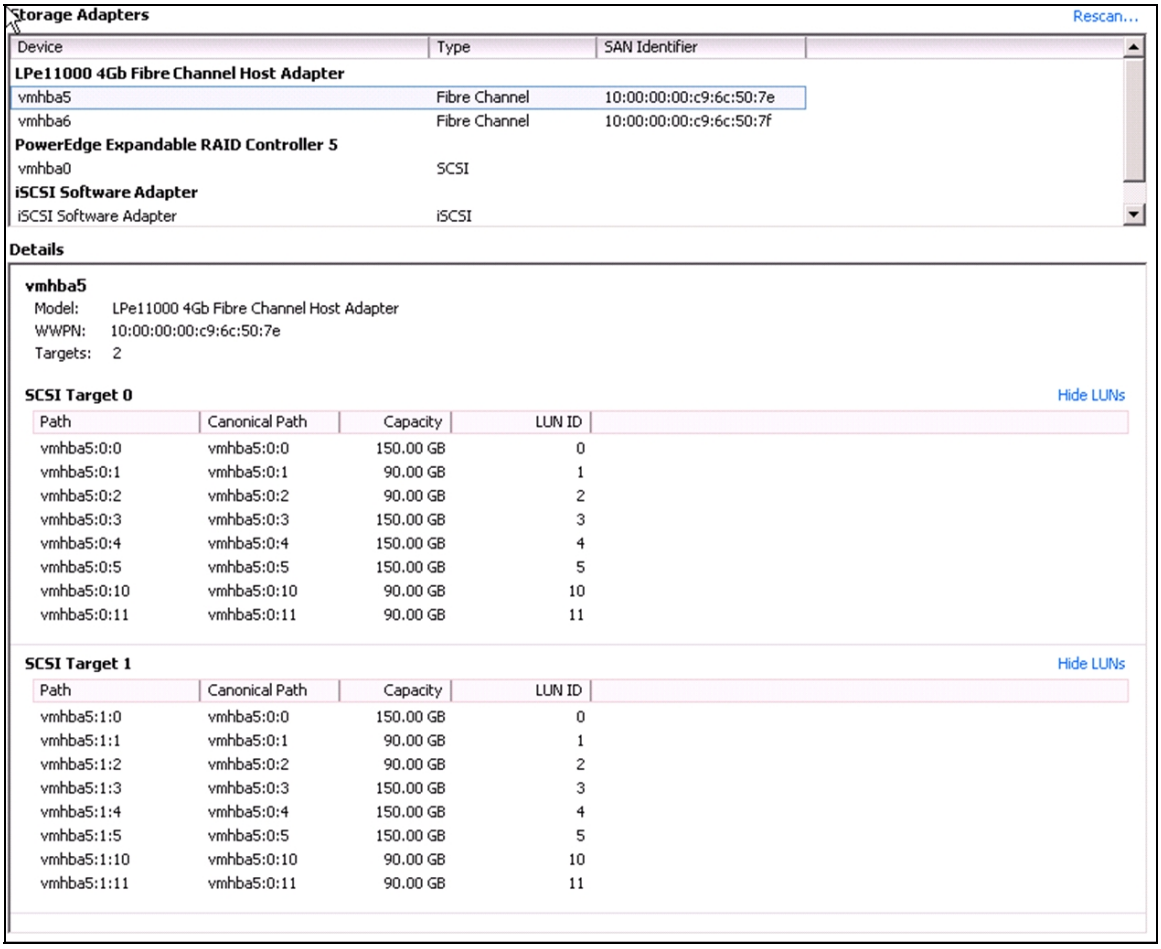


Figure 11.60 - Site 2 Storage Adapters Pre HBA Rescan

We now issue a rescan for storage devices, Figure11.61:

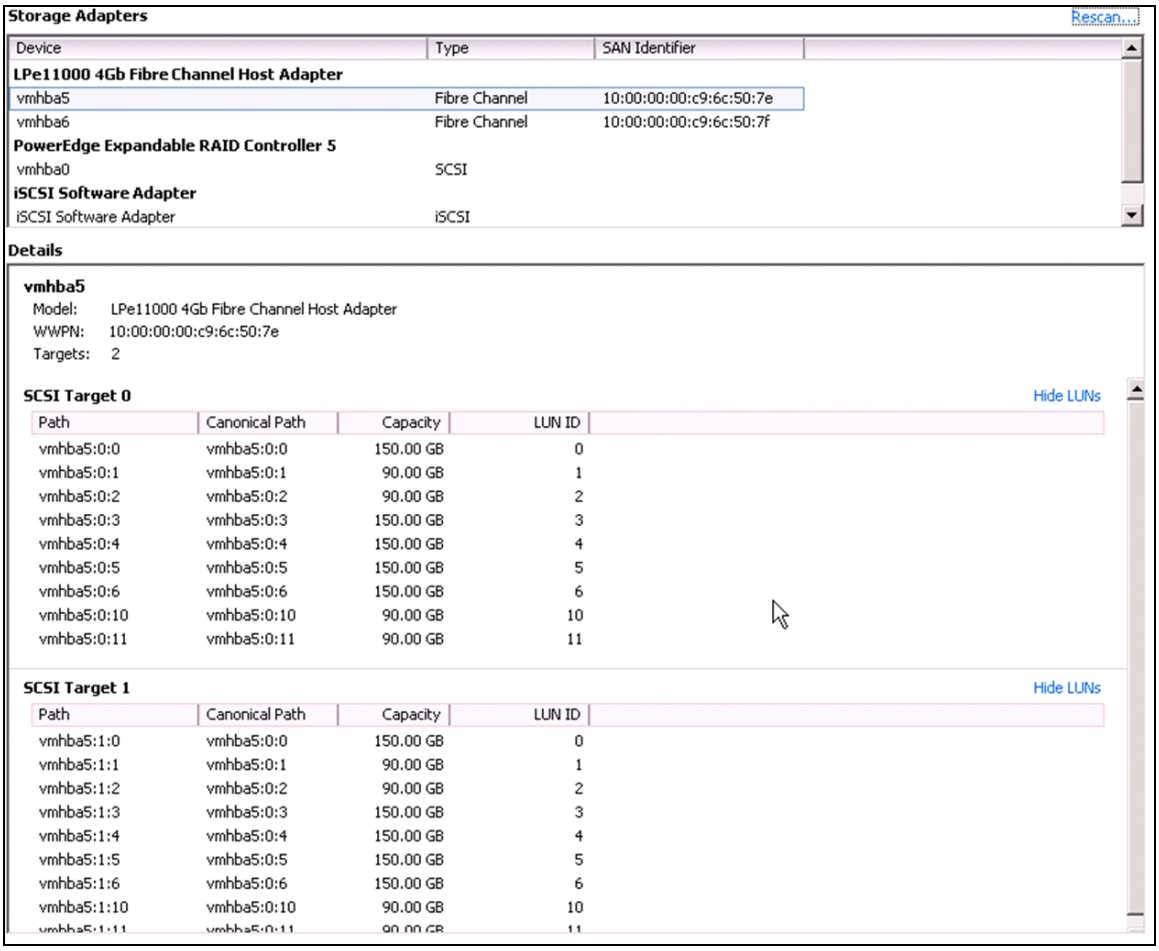


Figure 11.61 - Site 2 Storage Adapters Post HBA Rescan

We can now see in Figure-11.61 a LUN with lunid=6 has appeared in our list. If we now move to the "Storage" screen in VirtualCenter and click the "Refresh" link, Figure-11.62, we see that we now have access to the datastore named RP1. This datastore is based on our FlexClone volume and the key here is that the replicated LUNs are still in operation meaning we can now work against the clone for testing purposes:

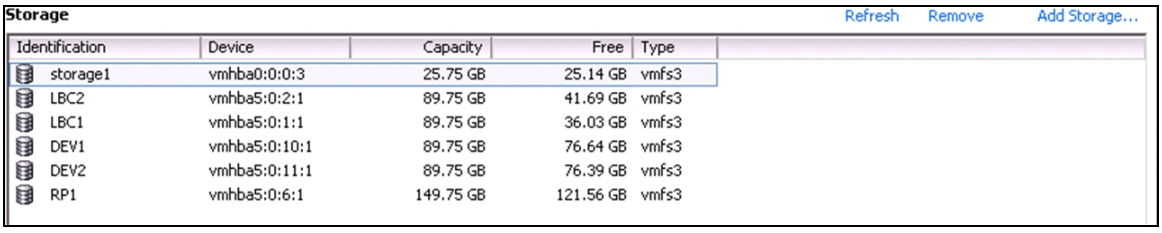


Figure 11.62 - VMFS Datastore Inventory Post Refresh

As with any other writeable datastore we can of course browse the RP1 datastore and have access to the VirtualMachines, Figure-11.63:

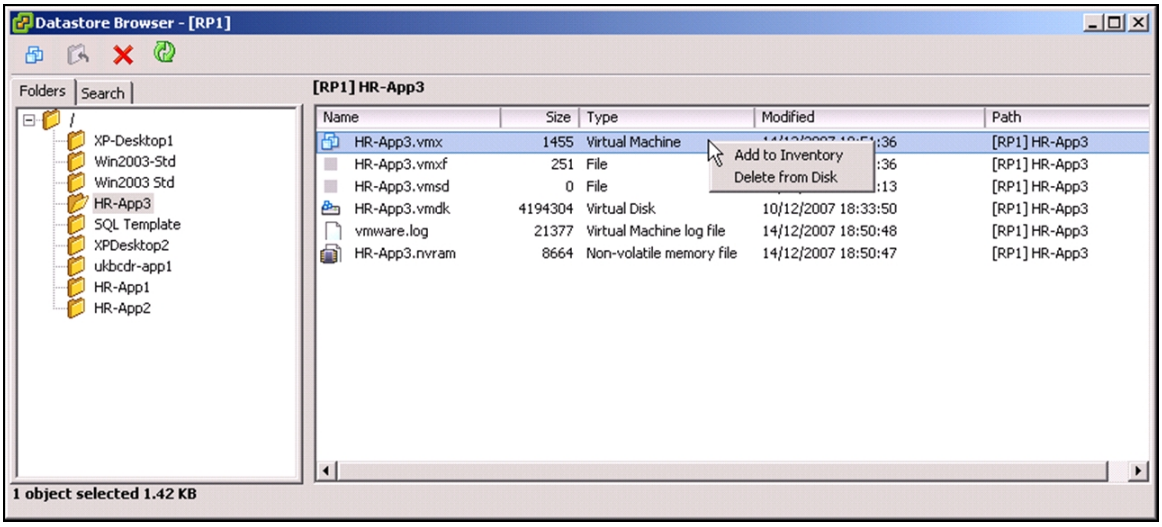


Figure 11.63 - Browse RP1 Datastore

FlexClone Volumes of NFS datastores

If the FlexClone volume will be used as a datastore, accessed via NFS, then once the FlexClone volume has been created, the FlexClone volume must be exported via NFS. This can be done via the Add Export button on the Manage NFS Exports page in FilerView, Figure-11.64.

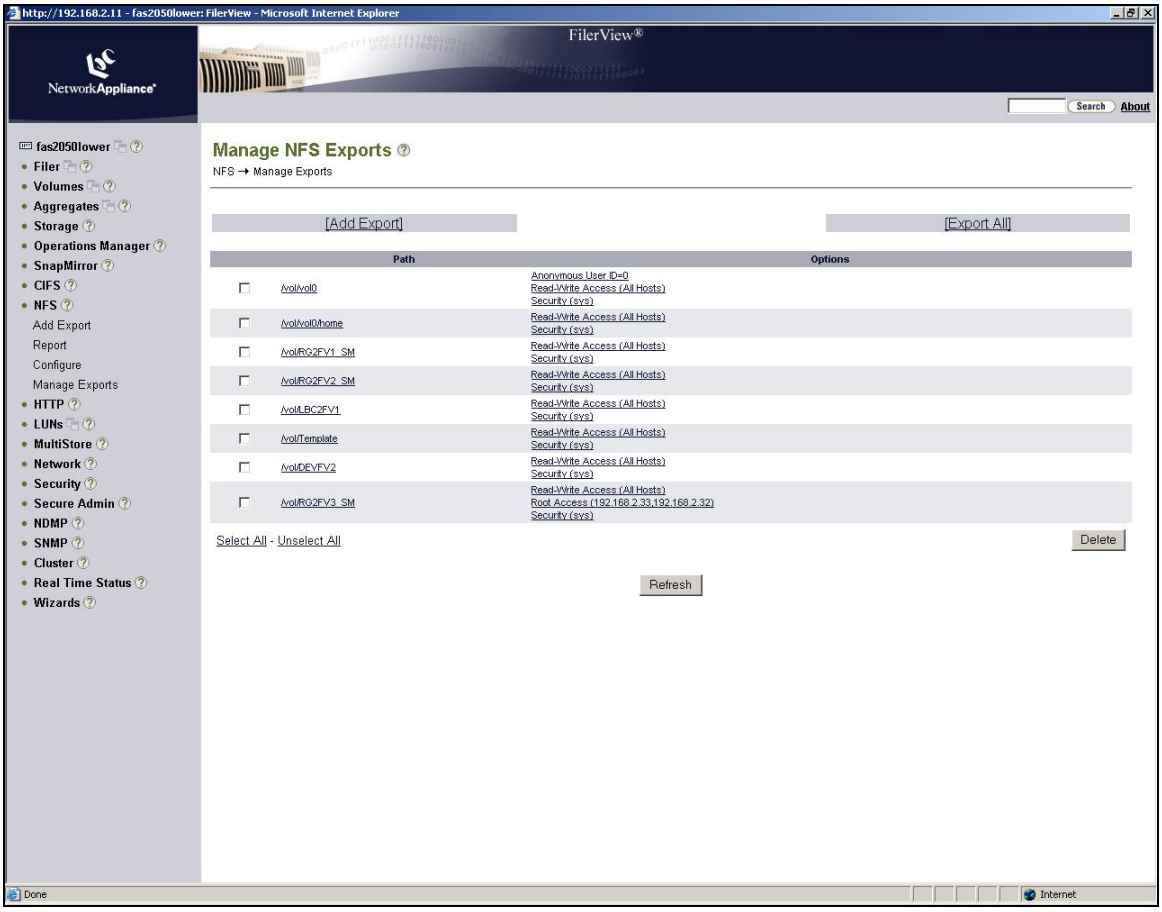


Figure 11.64 - Manage NFS Exports

The next step is to create a new NFS datastore in Virtual Center, for each ESX server, against the newly exported FlexClone Volume, Figure-11.65.

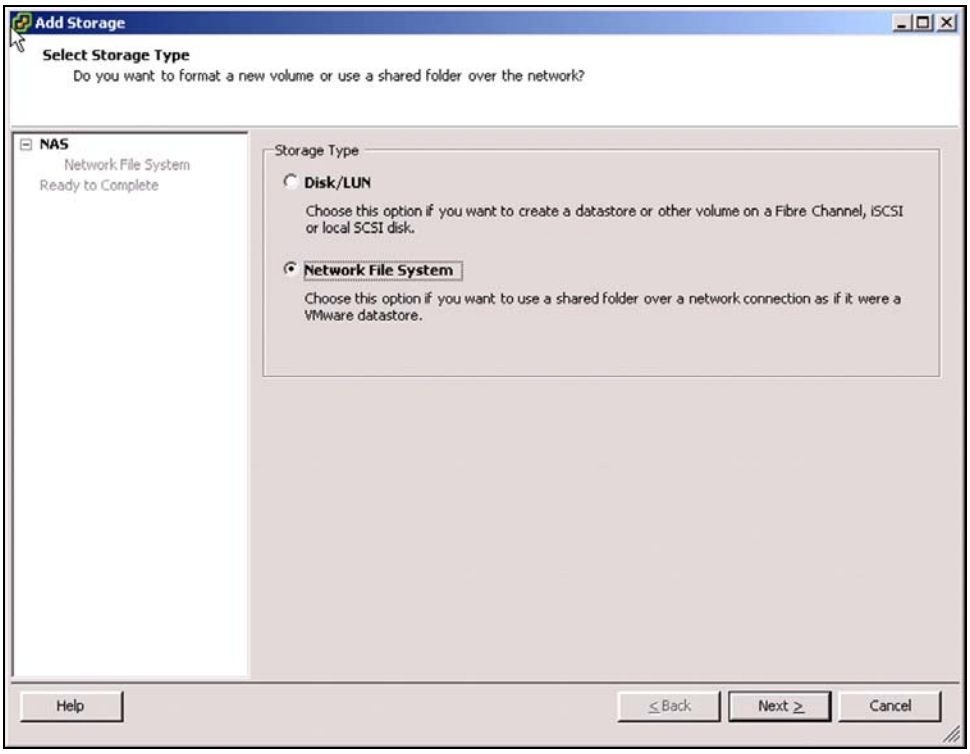


Figure 11.65 - Add NFS Storage

From the "Add Storage" wizard, select "Network File System", Figure-11.66:

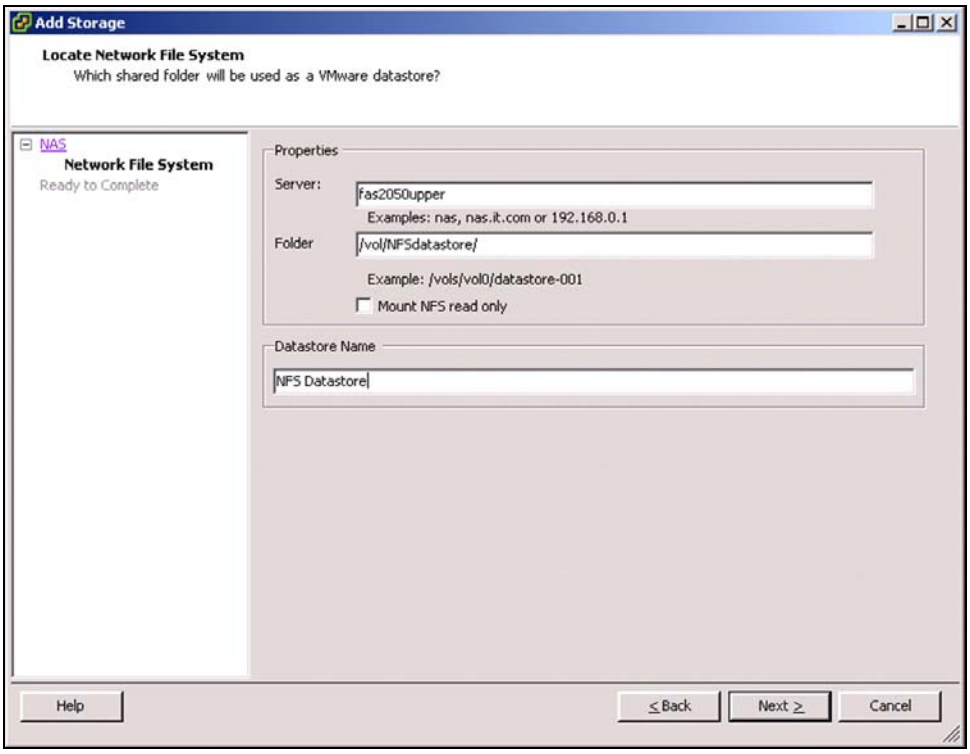


Figure 11.66 - Add Storage Datastore Name

Supply the IP address or DNS name of the Storage Controller, on which the FlexClone Volume was created, along with the path name to the FlexClone volume (in the form of /vol/FlexCloneVolumeName). Then provide a name for the datastore.

One the wizard completes, you will see your new NFS datastore on the FlexClone volume.

Chapter 12. Server Platform Details

In this chapter, we will review the server platforms that were selected for this BCDR solution. When we look at the type of physical servers you can select to run VMware ESX Server, there are two basic choices.

- Traditional rack servers
- Blade servers

The choice of server type does have implications on the infrastructure cabling side, for example you can greatly reduce your infrastructure cabling requirements (power, network, fiber) with a blade server deployment as blade servers will typically use shared network and SAN switches that are integrated into the blade chassis resulting in fewer network and fiber interconnects into the core network and SAN fabric switches when compared to deploying the same number of rack servers, for example 14-blade servers in a blade chassis will require substantially less infrastructure cables when compared to deploying 14-rack servers of the same CPU socket and memory footprint.

For the BCDR solution that we implemented in this VMbook we chose to use rack servers as we were only going to have a total of four physical servers in each datacenter and the infrastructure cabling required for the four servers in each datacenter is manageable; the use of a fully populated blade chassis in each datacenter would have been an overkill for the purposes of this VMbook.

We also chose to use rack servers from two different vendors—Dell, which were used in the Site 1, and Sun Microsystems, which were used in Site 2—to illustrate the benefits of hardware independence made possible through virtualization . When coupled with the properties of encapsulation and compatibility, hardware independence gives you the freedom to move a virtual machine from one type of x86 computer to another without making any changes to the device drivers, operating system, or applications. Hardware independence also means that you can run a heterogeneous mixture of operating systems and applications on a single physical computer.

- There will be a total of four VMware ESX Servers in Datacenter #1 (Production), to service virtual machines local to the datacenter on non-replicated storage, as well as the virtual machines that will float between datacenters via 'data replication' on designated replicated storage.

- The four VMware ESX in Site 1 were logically grouped into two Recovery Groups to facilitate a partial failover of either 'Recovery Group1' or Recovery Group 2' or a complete datacenter service failover of both Recovery Groups.

NOTE: Virtual machines on local non-replicated storage will not be failed over as these services are typically bound to the local datacenter. Services of this type are typically:

- Microsoft Active Directory Domain Controllers
- Virus Engine and DAT update servers
- Security services (HIPS and NIPS)
- Print services
- And so on…

- There will be a total of two designated BCDR VMware ESX servers and two designated Development VMware ESX servers in Datacenter #2 (BCDR and Development). The two designated BCDR VMware ESX hosts will be able to service failed over virtual machines from either 'Recovery Group1' or 'Recovery Group 2'. Should a total Datacenter #1 failover be orchestrated, the two designated Development VMware ESX hosts can be leveraged to provide the additional resources required to sustain the services failed over from Datacenter #1, this will be accomplished by either shutting down the development environment or leveraging nested Resource Pools to throttle back resources assigned to the development environment.

Site 1: Dell Server Platform

For Site 1 we selected the Dell PowerEdge 1950 servers configured with two dual-core 3.2 GHz CPUs and 4GB of memory. The Dell servers were installed with a single dual-port Emulex LPe11002-M4 Dual Channel 4Gb/s Fibre Channel PCI Express HBA. Refer to Figure 12.1 for a closer look at the Dell PowerEdge 1950 server.

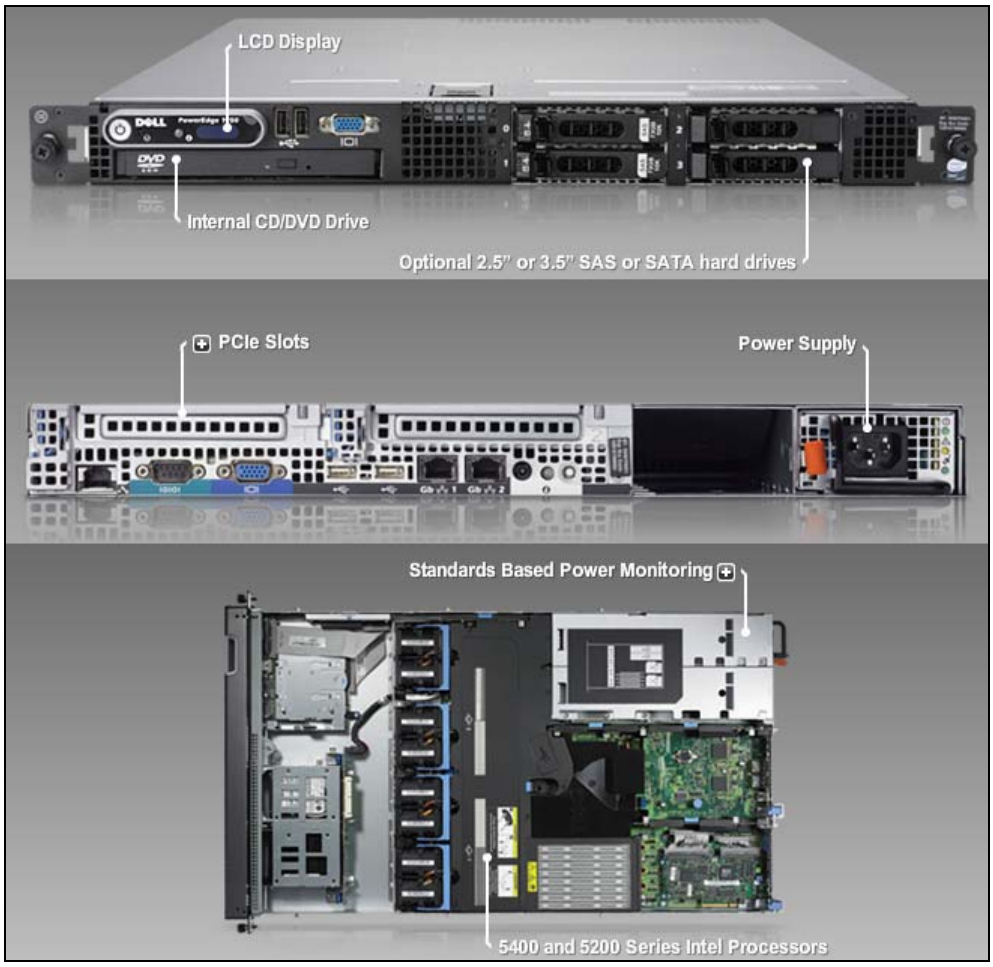


Figure 12.1 DellPoweredge 1950 Server

The Dell PowerEdge 1950 server platform offers the following benefits:

Performance in a Small Form Factor

The ultra-thin Intel® based DellTM PowerEdgeTM 1950 III 2-socket 1U rack server provides maximum performance while requiring minimal space and energy needs, helping to reduce energy and datacenter costs. The high concentration of computing power and redundancy makes the PowerEdge 1950 III the perfect choice for high performance computing clusters (HPCC), SAN front-end, web and infrastructure applications, especially where datacenter real estate is at a premium.

Streamlined Virtualization

Dell's PowerEdge 1950 III offers outstanding virtualization performance in a 2-socket 1U server combined with optional, factory integrated virtualization capabilities. Dell continues to simplify

virtualization by streamlining virtualization deployment and providing ease of use in virtual infrastructures. By factory integrating VMware ESXi, customers receive VMware capabilities and migration of virtual machines within a few clicks of a mouse.

These virtualization-ready server configurations provide an easy path to virtualization through factory-integrated hypervisors and recommended hardware configurations that support an ideal virtual infrastructure.

Use Energy Optimally

Power infrastructure and consumption already account for nearly half the cost of a typical datacenter and could soon represent half of the overall IT budget. To help enterprises address this challenge, the Dell PowerEdge 1950 III includes energy efficient components as well as optional onboard real-time power monitoring capabilities that can help reduce unnecessary power consumption. Dell is the only tier-one server vendor that is 100% committed to industry standards, which is why this solution adheres to the PMBus specification, enabling interoperability with other vendors' products. Dell is the only major server vendor to offer a lineup of mainstream servers in which key components have been optimized for low energy consumption; one of these servers is the PowerEdge Energy Smart 1950 III.

Simplify Management

Complex IT environments typically mean higher IT costs. The PowerEdge 1950 III includes features which help to simplify server deployment and consolidation tasks. Dell delivers fully assembled and highly tested machines that don't require expensive integration services. Remote management options including Wake-on-LAN and iSCSI boot help increase network manager productivity. The PowerEdge 1950 III also helps ease local system administration with its internal USB port and optional internal DVD.

Full details on the Dell PowerEdge1950 server can be found at:

http://www.dell.com/content/products/productdetails.aspx/pedge_1950_3?c=us&cs=RC956904&l=en&s=hied

Site 2: Sun Microsystems Server Platform

For Site 2 we selected two SunFire models. The SunFire X4200 and the SunFire X4600. The SunFire X4200 was configured with two dual core AMD Opteron 2.3 GHz CPUs and 8 GB of memory. The SunFire X4600 was configured with eight dual core AMD Opteron 2.6 GHz CPUs and 32 GB of memory.

Both Sun servers were installed with a single dual port Emulex LPe11002-M4 Dual Channel 4Gb/s Fibre Channel PCI Express HBA. Refer to Figure 12.2 and Figure 12.3 for a closer look at the SunFire X4200 and X4600 servers.

For pure growth potential and speed in a 2RU rack server, the Sun Fire X4200 M2 server is fast, reliable, and expandable, from one-way to eight-way processing. For flexibility, it also runs Solaris OS, Linux, Windows, and VMware

The SunFire X4200 server platform offers the following benefits:

- One or two quad-core/dual-core AMD Opteron 2000 Series processors
- Enterprise RAS features: redundant, hot-swappable components
- Includes Integrated Lights Out Manager (ILOM)
- Runs Solaris OS, Linux, Windows and VMware supported by Sun
- More network expandability: four Gigabit Ethernet ports in a 2RU server

The Sun Fire X4600 M2 server, now with quad-core AMD Opteron processors, packs the power of eight processors into a compact 4RU, energy-efficient system. Upgrading processors is simple and non-disruptive. Its virtualization capabilities can help your datacenter scale to several times its capacity in the same space.

The SunFire X4600 server platform offers the following benefits:

- Industry's first 4RU modular x64 server expandable from 2 to 8 processors
- Quad-Core AMD Opteron processing: up to 32 cores in a single system
- Supports up to 256GB of memory to accelerate data access and analysis
- Choice of Solaris, Linux, Windows, VMware (ESX 3.5 with 256GB support)

Full details on the SunFire X4200 server can be found at:

http://www.sun.com/servers/entry/x4200/

Full details on the SunFire X4600 server can be found at:

http://www.sun.com/servers/x64/x4600/

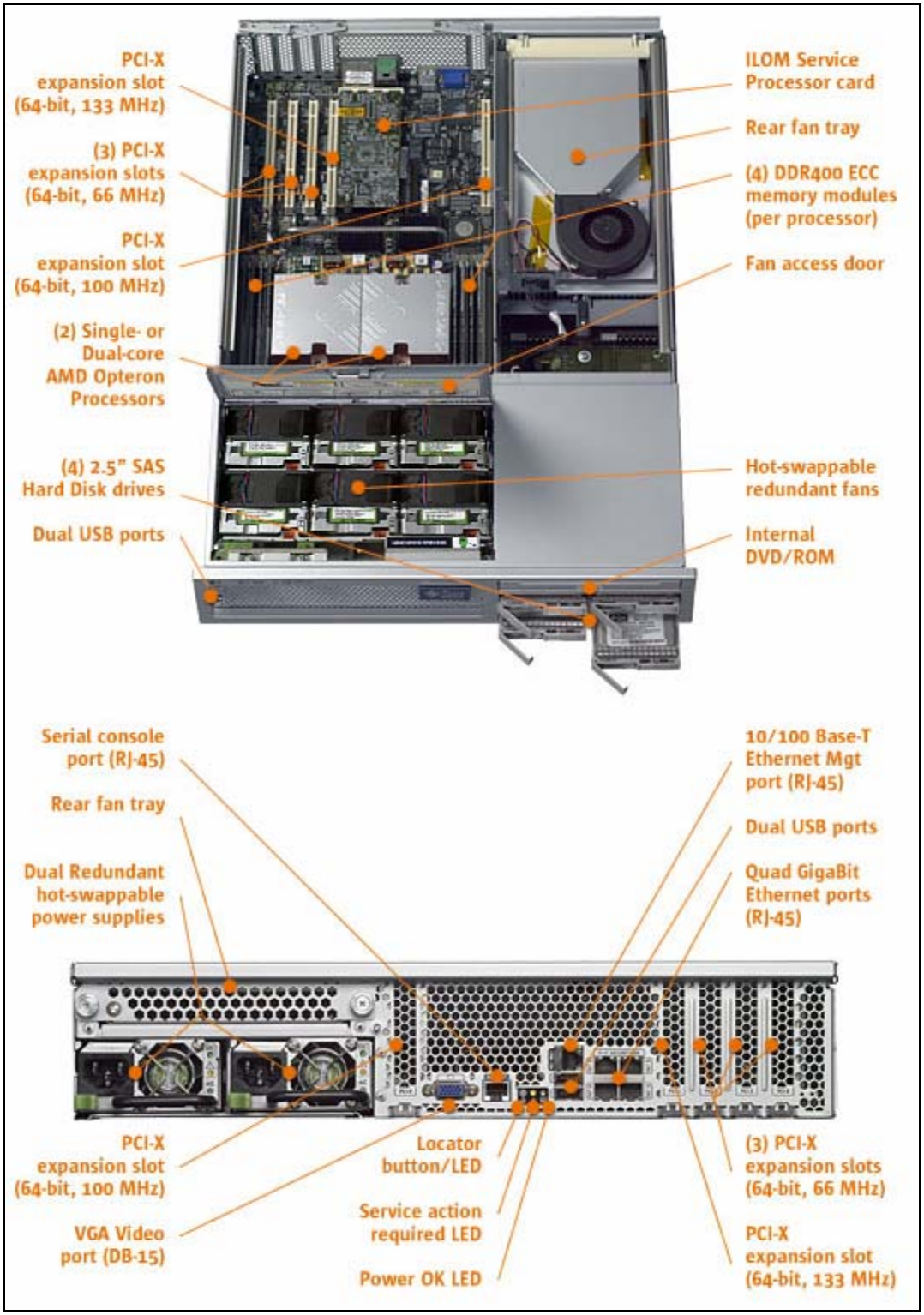


Figure 12.2 SunFire X4200

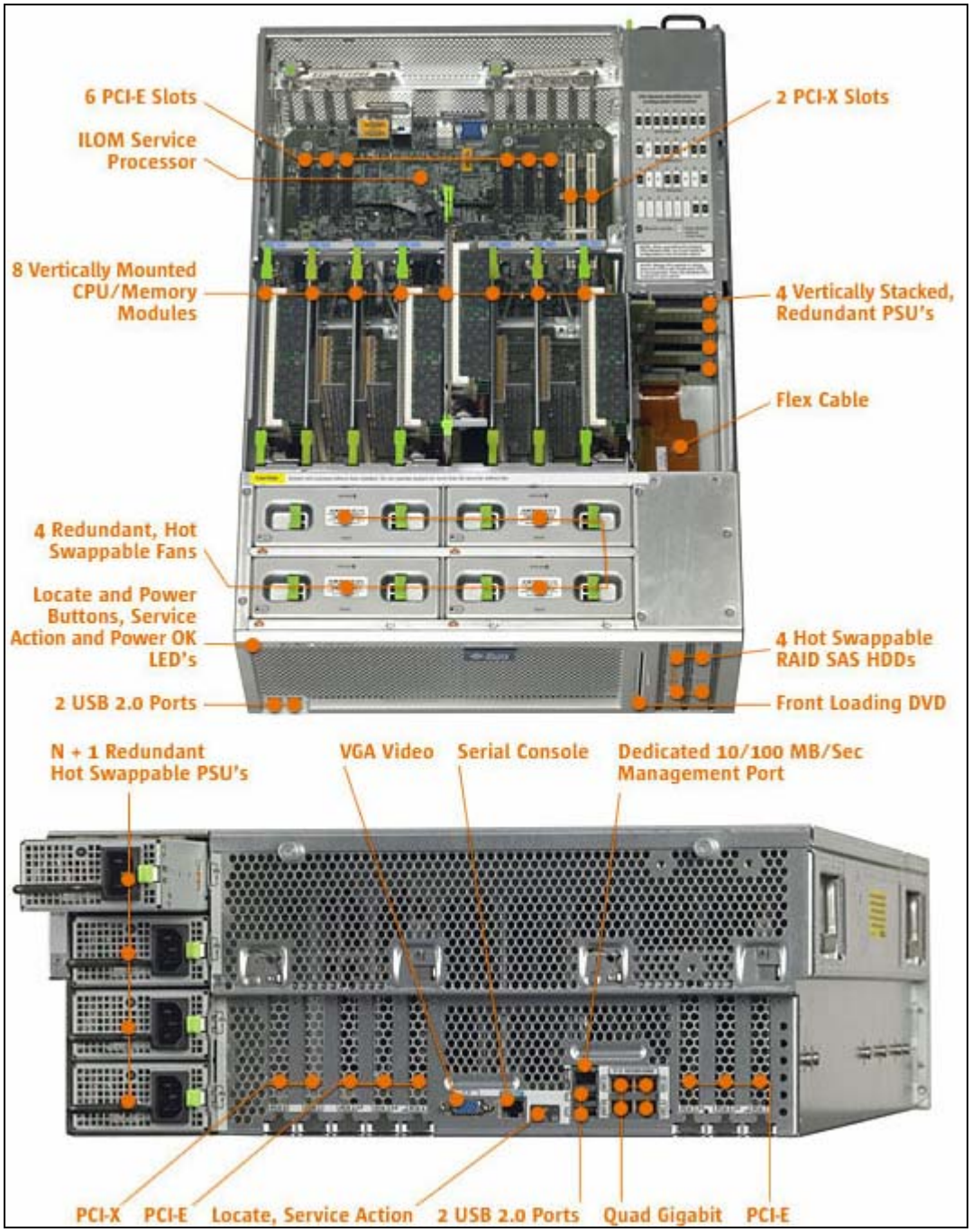


Figure 12.3 SunFire X4600

Appendix A: BCDR Failover Script

```
#!/usr/bin/perl -w
#

use strict;
use warnings;

use Getopt::Long;
use VMware::VILib;
use VMware::VIRuntime;

my @folder;
my @fparent;
my $begin;


$Util::script_version = "1.0";

my %opts = (
               datacenter => {
                   type => "=s",
                   help => "Datacenter name",
                   required => 1,
                   },
               datastore => {
                   type => '=s@{1,}',
                   help => "DataStore wildcards",
                   required => 1,
                   },
               folder => {
                   type => "=s",
                   help => "folder",
                   required => 0,
                   },
                path => {
                   type => "=s",
                   help => "Sub Folder to add the virtual machine(s) to",
```

```
                   required => 0,
                   },
                portgroup => {
                   type => "=s",
                   help => "Port Group for recovered virtual machines",
                   required => 0,
                   },
                network => {
                   type => "=s",
                   help => "Recovered virtual machine's Network connection
status is required true / false",
                   required => 0,
                   },
                 pool => {
                   type => "=s",
                   help => "Recovered virtual machine's Network resource pool",
                   required => 1
                   },
              );

# read/validate options and connect to the server
Opts::add_options(%opts);
Opts::parse();
Opts::validate();
Util::connect();

my $starttime = time_stamp();

# find datacenter
my $datacenter = Opts::get_option('datacenter');
my $datastore = Opts::get_option('datastore');
my $folderName = Opts::get_option('folder');
my $path = Opts::get_option('path');
my $nicopt = Opts::get_option('network');
my $portgroup = Opts::get_option('portgroup');
my $pool = Opts::get_option('pool');



my $datacenter_view = Vim::find_entity_view(view_type => 'Datacenter',
                                            filter => { name => $datacenter
});
Fail("Datacenter $datacenter not found\n") unless ($datacenter_view);
```

```
#my $pool_view = Vim::find_entity_view(view_type => 'ResourcePool',filter => {
'name' => "^$pool\$"});
my $pool_view = Vim::find_entity_view(view_type => 'ResourcePool');
Fail ("Resource Pool $pool not found\n") unless ($pool_view);

# Folder Selection
# alternative way of getting folders .... replaces your folder selection
# basically, if path is set then it selects the folder from the path
# otherwise, simply find the folder that was specified

my $folder_view;
# if the user provided a path then find the folder this way

if (Opts::option_is_set ('path')) {
    my $searchPath = '/'.$datacenter.'/vm/'.$path;
    my $searchIndex = Vim::get_view(mo_ref => Vim::get_service_content()-
>searchIndex);
    my $folder = $searchIndex->FindByInventoryPath (inventoryPath =>
$searchPath);
    die ("Folder not found.\n") unless (defined $folder);
    $folder_view = Vim::get_view (mo_ref => $folder);

    }
else {
    die ("Must supply either --path or --folder") unless (Opts::option_is_set
('folder'));
    my $folder_views = Vim::find_entity_views (view_type => 'Folder',
                                               filter => {'name' =>
$folderName},
                                               begin_entity =>
$datacenter_view);
    die ("Folder $folderName not found.\n") unless (defined $folder_views);
    die ("Folder $folderName not unique.\n") unless ($#{$folder_views} == 0);
      # only one entry
    $folder_view = shift @$folder_views;
    }
# End Folder Selection

#  get all hosts under this datacenter
my $host_views = Vim::find_entity_views(view_type => 'HostSystem',
                                                    begin_entity =>
$datacenter_view);
my $vm_views = Vim::find_entity_views(view_type => 'VirtualMachine',
                                                    begin_entity =>
$datacenter_view);

#  Prime the %datastore array with the browser objects for each datastore
```

```
#  Will speed up searches for each virtual machine

print "\n";
print "\n*******************************************\n";
print "* Site1 -> Site 2 Failover Activation Script *\n";
print "*******************************************\n";

my %datastore;
my %vmsByPath;                          # list indexed by path ... value has
vmView or "not registered"
my %vmsByName;                          # list index by name ... value has
vmView
PrepSearches ($host_views, $datastore);
foreach (keys %datastore) {
    my $files = ListVMs ($_);
    }
foreach my $vm_view (@$vm_views) {
    if (! defined $vm_view->config) {
       print "Skipping ", $vm_view->name, "\n";
       next;
       }
    my $path = $vm_view->config->files->vmPathName;
    if (exists $vmsByPath{$path}) {
        $vmsByPath{$path} = $vm_view;
        $vmsByName{$vm_view->name} = $vm_view;
        }
    else {
    print "Virtual Machine " . $vm_view->name . " not found check datastore
online!!!\n"; }
    }

print "\n";
print "\n*********************************\n";
print "* Protected Virtual Machines Found *\n";
print "*********************************\n";

foreach my $vm (keys %vmsByPath) {
my $name = ($vmsByPath{$vm} eq 'not registered') ? 'not registered' :
$vmsByPath{$vm}->name;
#    print $vm, ":", $vmsByPath{$vm}->name, "\n";
print $vm, ":", $name, "\n";
    }

print "\n";
print "\n*****************************************************\n";
print "* Protected Virtual Machines Unregister / Re-Register *\n";
```

```
print "*******************************************************\n";

my $new_vms;
foreach my $path (keys %vmsByPath) {
    print "\n" . $path . "\n";
    my $vmname = $vmsByPath{$path}->name;
    if ($vmname ne 'not registered'){
        my $vm_view = $vmsByPath{$path};
        my $powerstate = $vm_view->runtime->powerState->val;
        print "virtual machine Power on State is: " . $vm_view->runtime-
>powerState->val . "\n";
        if ($powerstate eq 'poweredOn'){
            print "Powering off " . $vm_view->name . "\n";
            $vm_view->PowerOffVM();
            print "Poweroff successfully completed\n";
            }
        print "Unregistering virtual machine.......\n";
        $vm_view->UnregisterVM();
        }
    print "Re-Registering virtual machine with VMX Path: " . $path . "\n";
    my $new_vm = $folder_view->RegisterVM(path => $path, asTemplate =>
'false', pool => $pool_view);
    push (@$new_vms, $new_vm);
    }


print "\n";
print
"\n*****************************************************************\n";
print "* Protected Virtual Machines Power On and Network Reconfiguration *\n";
print "*****************************************************************\n";


# Verify and PowerOn
my $new_vm_views = Vim::get_views (mo_ref_array => $new_vms);
sanitizeAndPowerOn($new_vm_views);


my $endtime = time_stamp();
print "Start time was: " . $starttime . "\n";
print "End time was: " . $endtime . "\n";
# disconnect from the server
Util::disconnect();

# Sub Routine for virtual machine PowerOn Verification     #
```

```
sub sanitizeAndPowerOn {
    my ($vm_views) = @_;
    foreach my $vm_view (@$vm_views) {
        print "\nSantizing virtual machine: " . $vm_view->config->name . "\n";
        my $vm_network_devices = getVMNetworkDevices($vm_view);
        my $vm_removable_media = getVMRemovableMedia($vm_view);
        if (!$vm_network_devices) {
            print "WARNING! No network devices found. \n";
        } else {
            disconnectNetwork($vm_view, $vm_network_devices);
        }
        if (!$vm_removable_media) {
            print "WARNING! No removable media devices found. \n";
        } else {
            disconnectMedia($vm_view, $vm_removable_media);
            }
        }

    staggerPowerOn($vm_views);
}

# Sub Routine for PowerOn Staggering
# staggerfactor hardcoded to 1 variable
# commented out of command opts
sub staggerPowerOn {
    my ($vm_views) = @_;
    my $vm_count = @$vm_views;
    print "\n" . $vm_count . " virtual machines to power on \n";
    while($vm_count != 0) {
#       my $stagger_factor = $opts{stagger_factor};
        my $stagger_factor = 3;

        my $vm_power_on_tasks = undef;
        my $vm_power_on_task = undef;
        until(($stagger_factor == 0) || ($vm_count == 0)){
            my $vm_view = shift(@{$vm_views});
            print "\nPowering On virtual machine: " . $vm_view->config->name .
"\n";
            my $vm_power_on_task = $vm_view->PowerOnVM_Task();
            push(@{$vm_power_on_tasks}, $vm_power_on_task);
            $vm_count--;
            $stagger_factor--;
            }
        while (!powerOnTasksCompleted($vm_power_on_tasks)) {
            sleep 3; #Go easy on VirtualCenter and take a nap for a bit
```

```
            checkAndKeepUUIDs(); #Check to see if we've been presented with
the UUID question
            }
        }
}

# Sub Routine for PowerOn completion checks
sub powerOnTasksCompleted {
        my($power_on_task_refs) = @_;
        my $power_on_task_views = Vim::get_views(mo_ref_array =>
$power_on_task_refs);
        my $completed = "True"; #Assume task run successful unless we discover
otherwise
        foreach (@{$power_on_task_views}) {
                my $power_on_task_state = $_->info->state;
                print $_->info->entityName . ": PowerOnVM Task Status - " .
$power_on_task_state->{'val'} . "\n";
                if (($power_on_task_state->{'val'} ne "error") &&
($power_on_task_state->{'val'} ne "success")) {
                        $completed = 0; #At least one task is either queued or
still running :(
                }
        }
        return($completed);
}



# Sub Routine for UUID Question Answer
sub checkAndKeepUUIDs {
    my $vm_views = Vim::find_entity_views(view_type => 'VirtualMachine',
filter => { 'runtime.question.id' => '\d+'});
    foreach my $vm_view (@{$vm_views}) {
        if (defined($vm_view->runtime->question->id)) {
            print "Keeping virtual machine UUID for " . $vm_view->config->name
. "\n";
            my $vm_question_info = $vm_view->runtime->question;
            my $vm_question_id = $vm_question_info->id;
            my $vm_question_answer_choice = 2 ;#Keep
            $vm_view->AnswerVM(questionId => $vm_question_id, answerChoice =>
$vm_question_answer_choice);
            }
        }
}

# Sub Routine for virtual machine Network Device Listing
sub getVMNetworkDevices {
      my ($vm_view) = @_;
```

```
	my $vm_network_devices;
	my $vm_devices = $vm_view->config->hardware->device;
	foreach my $device (@$vm_devices) {
		if ($device->deviceInfo->label =~ m/Network/) {
			push (@{$vm_network_devices}, $device);
		}
	}
	return $vm_network_devices;
}

# Sub Routine for virtual machine Removable Media Listing
sub getVMRemovableMedia {
	my ($vm_view) = @_;
	my $vm_removable_media;
	my $vm_devices = $vm_view->config->hardware->device;
	foreach my $device (@$vm_devices) {
		if ($device->deviceInfo->label =~ m/Drive/) {
			push (@{$vm_removable_media}, $device);
		}
	}
	return $vm_removable_media;
}

# Sub Routine for virtual machine Network disconnect
sub disconnectNetwork {
    my ($vm_reconfig_view, $vm_reconfig_network_devices) = @_;
    my @vm_reconfig_devices;
    print "Setting Networks...\n";
    foreach my $network_device (@{$vm_reconfig_network_devices}) {
        my $network_address_type = $network_device->addressType;
        my $network_device_controller_key = $network_device->controllerKey;
        my $network_device_key = $network_device->key;
        my $network_device_mac_address = $network_device->macAddress;
        my $network_unit_number = $network_device->unitNumber;
        my $network_wake_on_lan_enabled = $network_device->wakeOnLanEnabled;
        #print "Setting Network: " . $network_device->deviceInfo->summary . "
Connected=" . $nicopt ."\n";
        #my $network_backing_info =  VirtualEthernetCardNetworkBackingInfo-
>new(deviceName => $network_device->backing->deviceName, network =>
$network_device->backing->network);
        my $network_backing_info =  VirtualEthernetCardNetworkBackingInfo-
>new(deviceName => $portgroup, network => $network_device->backing->network);
        print "Setting Network: " . $portgroup . "  Connected=" . $nicopt
."\n";
        #my $network_connect_info = VirtualDeviceConnectInfo-
>new(allowGuestControl => 'true', connected => 'false', startConnected =>
'false');
```

```
        my $network_connect_info = VirtualDeviceConnectInfo-
>new(allowGuestControl => 'true', connected => $nicopt, startConnected =>
$nicopt);
        my $network_device_info = Description->new(label => $network_device-
>deviceInfo->label, summary => $network_device->deviceInfo->summary);
        my $network_config_spec_operation = VirtualDeviceConfigSpecOperation-
>new('edit');
        my $vm_reconfig_device = VirtualPCNet32->new(addressType=>
$network_address_type, backing => $network_backing_info, connectable =>
$network_connect_info, controllerKey => $network_device_controller_key,
deviceInfo => $network_device_info, key => $network_device_key, macAddress =>
$network_device_mac_address, unitNumber => $network_unit_number,
wakeOnLanEnabled => $network_wake_on_lan_enabled);
        my $network_config_spec = VirtualDeviceConfigSpec->new(device =>
$vm_reconfig_device, operation => $network_config_spec_operation);
        push(@vm_reconfig_devices, $network_config_spec);
        }
    my $vm_reconfig_spec = VirtualMachineConfigSpec->new(deviceChange =>
\@vm_reconfig_devices);
    $vm_reconfig_view->ReconfigVM_Task(spec => $vm_reconfig_spec);
}

# Sub Routine for virtual machine Media Disconnect
sub disconnectMedia {
    my ($vm_reconfig_view, $vm_reconfig_media) = @_;
    my @vm_reconfig_devices;
    print "Disconnecting Removable Media...\n";
    foreach my $media_device (@{$vm_reconfig_media}) {
        my $media_controller_key = $media_device->controllerKey;
        my $media_device_key = $media_device->key;
        my $media_unit_number = $media_device->unitNumber;
        print "Disconnecting: " . $media_device->deviceInfo->summary. "\n";
        my $media_connect_info = VirtualDeviceConnectInfo-
>new(allowGuestControl => 'true', connected => 'false', startConnected =>
'false');
        my $media_config_spec_operation = VirtualDeviceConfigSpecOperation-
>new('edit');
        my $vm_reconfig_device;
        if ($media_device->deviceInfo->label =~ m/CD/) { # Disconnect CDROM
           my $media_backing_info = VirtualCdromAtapiBackingInfo-
>new(deviceName => "/dev/cdrom");
           my $media_device_info = Description->new(label => $media_device-
>deviceInfo->label, summary => "/dev/cdrom");
           $vm_reconfig_device = VirtualCdrom->new(backing =>
$media_backing_info, connectable => $media_connect_info, controllerKey =>
$media_device->controllerKey, deviceInfo=> $media_device_info, key =>
$media_device->key, unitNumber => $media_unit_number);
        } else { # Disconnect Floppy
           my $media_backing_info = VirtualFloppyDeviceBackingInfo-
>new(deviceName => "/dev/fd0");
```

```
           my $media_device_info = Description->new(label => $media_device-
>deviceInfo->label, summary => "/dev/fd0");
           $vm_reconfig_device = VirtualFloppy->new(backing =>
$media_backing_info, connectable => $media_connect_info, controllerKey =>
$media_device->controllerKey, deviceInfo=> $media_device_info, key =>
$media_device->key, unitNumber => $media_unit_number);
            }
       my $media_config_spec = VirtualDeviceConfigSpec->new(device =>
$vm_reconfig_device, operation => $media_config_spec_operation);
       push(@vm_reconfig_devices, $media_config_spec);
       }
   my $vm_reconfig_spec = VirtualMachineConfigSpec->new(deviceChange =>
\@vm_reconfig_devices);
   $vm_reconfig_view->ReconfigVM(spec => $vm_reconfig_spec);
}



#  The PrepSearches routine simply caches the datastore array with the
datastore browser object #  for.  %datastore indexed by datastore name and
holds a pointer to the datastore browser
# object

sub PrepSearches {
   my ($host_views, $dsList) = @_;
   my $pattern = join ('|', @$dsList);
   my @datastore_array;
   foreach my $host (@$host_views) {
       foreach my $datastore (@{$host->datastore}) {
           push (@datastore_array, $datastore);
           }
        }
   my $datastores = Vim::get_views (mo_ref_array => \@datastore_array);
   foreach my $datastore (@$datastores) {
# skip unless the datastore matches the pattern
      next unless ($datastore->info->name =~ m/$pattern/);
      my $browser = Vim::get_view (mo_ref => $datastore->browser);
      my $name = $datastore->info->name;
      $datastore{$name} = $browser;
      }
   }

#  ListVMs simply caches the files in the respective virtual machine directory
#  Strip out datastore and directory header from spec
#  Populate the %list, index by path name, and value is the fileSize
#  Return pointer to %list structure

sub ListVMs {
```

```
    my ($dsName) = @_;
    my $browser = $datastore{$dsName};
    my $flags = new FileQueryFlags (fileSize => 'true', fileType => 'false',
modification => 'true');
    my $searchSpec = new HostDatastoreBrowserSearchSpec (details => $flags,
matchPattern => ['*.vmx']);
    my $results = $browser->SearchDatastoreSubFolders(datastorePath =>
"[$dsName]", searchSpec => $searchSpec);
    foreach my $result (@$results) {
        my $files = $result->file;
        foreach my $file (@$files) {
            my $path = $result->folderPath.$file->path;
            $vmsByPath{$path} = "not registered";
            }
        }
    }

#   FindSize
#   Takes a filename, strips out the leading path and datastore, and returns
the matching string
#   Should be generalized, just in case the file is stored somewhere else

sub FindSize {
    my ($list, $filename) = @_;
    if ($filename =~ /\[/) {
        $filename =~ /\[(.*\/)([^\/]*)$/;
        $filename = $2;
        }
    if (exists $list->{$filename}) {
        return $list->{$filename};
        }
    else { return "unknown"};
    }

my $sec;
my $min;
my $hour;
my $mday;
my $mon;
my $year;
my $wday;
my $yday;
my $isdst;

sub time_stamp {
  my ($d,$t);
```

```
  my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);

        $year += 1900;
        $mon++;
        $d = sprintf("%4d-%2.2d-%2.2d",$year,$mon,$mday);
        $t = sprintf("%2.2d:%2.2d:%2.2d",$hour,$min,$sec);
        return($d,$t);
}


sub Fail {
    my ($msg) = @_;
    Util::disconnect();
    die ($msg);
    exit ();
    }
```

Appendix B: VMware Tools Script

```
#!/usr/bin/perl -w

use strict;
use warnings;

use VMware::VIM2Runtime;
use VMware::VIRuntime;
use VMware::VILib;
use Getopt::Long;

my @options = ('afterPowerOn', 'afterResume', 'beforeGuestReboot',
'beforeGuestShutdown', 'beforeGuestStandby');


my %opts = (
   datacenter => {
      type => "=s",
      help => "VirtualCenter Datacenter Object name is required",
      required => 1,
   },
   folder => {
         type => "=s",
         help => "DR Virtual Machines Folder name is required",
         required => 1,
   },
   display => {
            type => "=s",
            help => "Display Current Values Only (yes/no)",
            required => 0,
   },
   option  => {
      type     => "=s",
      variable => "option",
      help     =>
"afterPowerOn|afterResume|beforeGuestReboot|beforeGuestShutdown|beforeGu
estStandby",
      required => 1},
   value  => {
      type     => "=s",
      variable => "newValue",
      help     => "New Tools Value (yes/no)",
      required => 0},
);


##################################################
# Get Command Line Options                       #
##################################################

Opts::add_options(%opts);
Opts::parse();
Opts::validate();
```

```
Util::connect();
my $option = Opts::get_option ('option');
my $value = Opts::get_option ('value');
my $display = Opts::get_option ('display');



#########################
# Additional validation #
#########################

if (!defined (Opts::get_option('datacenter') ||
Opts::get_option('folder') || Opts::get_option('service_url'))) {
   print "ERROR: --datacenter or --folder must be specified\n\n";
   help();
   exit (1);
}


####################################
# Verify Command Line Options Exist #
####################################

# DataCenter
my $datacenter = Opts::get_option('datacenter');
my $datacenter_view = Vim::find_entity_view(view_type => 'Datacenter',
                                            filter => { name =>
$datacenter });
if (!$datacenter_view) {
   die "\nABORT: Datacenter '" . $datacenter . "' not found\n";
}

# DR VM Folder
my $folder = Opts::get_option('folder');
my $folder_check = Vim::find_entity_view(view_type => 'Folder',filter =>
{ name => "^$folder\$"});
if (!$folder_check) {
   die "\nABORT: DR VM Folder '" . $folder . "' not found\n";
}


#########################################################
# Get all hosts under this datacenter                   #
#########################################################

my $host_views = Vim::find_entity_views(view_type => 'HostSystem',
                                        begin_entity =>
$datacenter_view);

###############
# Print hosts #
###############

my $counter = 1;
print "\nHosts found:\n";


foreach (@$host_views) {
   print "$counter: " . $_->name . "\n";
   $counter++;
}
```

```
##################
# Find all views #
##################

my $folder_view = Vim::find_entity_view(view_type => 'Folder',filter =>
{ name => "^$folder\$"});

if (!$folder_view) {
   die "DR VM Folder '" . $folder . "' not found\n";
}


########################
# Print folders found #
########################

print "\nDR Folder found:\n";


########################################
# in line below add @ if passing array #
########################################

foreach ($folder_view) {
      print $_->name . "\n";
}


###############
# Print vm's #
###############

$counter = 1;
print "\nVM's found in DR Folder:\n";

# get all VM's under the DR Folder in this datacenter

my $vm_views = Vim::find_entity_views(view_type =>
'VirtualMachine',begin_entity => $folder_view);

Fail ("No Registered VM's found in DR Folder ". $folder_view->name .
"\n") unless (@$vm_views);

foreach (@$vm_views) {
   print "$counter: " . $_->name . " (Current Power State of VM is: " .
$_->runtime->powerState->val. ")" . "\n";
   $counter++;
}


############################
# Display Current PowerOn #
############################

eval {
      displayCurrentOptionValue($datacenter_view);
};

if ($@) {
    Fail ($@) unless (ref($@));
    Fail ($@->fault_string);
}

############################
```

```
# Update Tools Setting    #
############################

if ($display eq 'no') {
  eval {
      updateToolsSetting($datacenter_view);
      }
}



############################
# Display current setting #
############################

sub displayCurrentOptionValue {
my $filter_vms = Vim::find_entity_views(view_type =>
'VirtualMachine',begin_entity => $folder_view);
      foreach my $filter_vm(@$filter_vms) {
      print "\nVM is: " . $filter_vm->name . "\n";
  print "VM tools " . $option . " value is currently: " . $filter_vm-
>config->tools->$option . "\n";
      }
}

########################
# Update Tools Setting #
########################

sub updateToolsSetting {
      my $filter_vms2 = Vim::find_entity_views(view_type =>
'VirtualMachine',begin_entity => $folder_view);
      foreach my $filter_vm2(@$filter_vms2) {
  my $toolState = $filter_vm2->config->tools;
if (Opts::option_is_set ('option')) {
    Fail ("Usage: please specify a --value argument.\n") unless
(Opts::option_is_set ('value'));
    my $option = Opts::get_option ('option');
    my $optionString = join ("|", @options);
    Fail ("Usage: Invalid option $option.  Valid options are " . join ("
,", @options))
        unless ($option =~ /$optionString/);
    my $newSetting = Opts::get_option ('value');
    Fail ("Usage: --value argument must be yes or no") unless
($newSetting =~ /yes|no/);
    my $newValue = ($newSetting eq 'yes') ? 1 : 0;
    my $toolsConfigInfo = new ToolsConfigInfo ($option => $newValue);
    my $virtualMachineConfigSpec = VirtualMachineConfigSpec->new (
                                tools => $toolsConfigInfo);
    print "\nUpdating " . $filter_vm2->name . "\n";
    eval {
        $filter_vm2->ReconfigVM( spec => $virtualMachineConfigSpec);
        };
    if ($@) {
        print "Reconfiguration failed.\n $@";
        }
    $filter_vm2->update_view_data();
    }
$toolState = $filter_vm2->config->tools;
print "\nCurrent VMware Tools Script Settings.\n";
#print "Tools Version: ", $toolState->toolsVersion, "\n";
foreach my $setting (@options) {
    print "$setting: ", ($toolState->$setting) ? "yes" : "no", "\n";
```

```
                }
            }
        }
        ##############################################
        # Fail / Help / Banner and  Usage Messages #
        ##############################################

        sub Fail {
           my ($msg) = @_;
        #   eval {Util::disconnect();};
        #   die ($msg);
            print $msg;
           }

        sub help {
           my $help_text = <<'END';

        END
           print $help_text;
        }


        ##############################
        # Disconnect from the server #
        ##############################

Util::disconnect();
```

vmbooks