A Smartphone Design Approach to User Communication Interface for Administering Storage System Network

Weider D. Yu

Computer Engineering Department, San Jose State University

San Jose (Silicon Valley), CA 95192-0180, United States

Tel: 1-408-924-7365 E-mail: weider.yu@sjsu.edu

Xiao Su

Computer Engineering Department, San Jose State University

San Jose (Silicon Valley), CA 95192-0180, United States

Tel: 1-408-924-7366 E-mail: xiao.su@sjsu.edu

Jason Hansen

Computer Engineering Department, San Jose State University

San Jose (Silicon Valley), CA 95192-0180, United States

E-mail: perlhack40@gmail.com

Received: August 5, 2012 Accepted: November 11, 2012 Published: December 16, 2012

DOI: 10.5296/npa.v4i4.2194 URL: http://dx.doi.org/10.5296/npa.v4i4.2194

Abstract

This paper investigates the feasibility and potential of using a mobile smartphone as a user administration control for storage system network. It includes the design and implementation of a mobile smartphone software system that can be used to perform system administration activities on the storage system without the need of using a laptop or desktop system. In the paper the design of some effective metaphors to present the storage system on the limited display area of a mobile smartphone device and the set of system features chosen for implementation are presented. Current storage system administration is limited to using storage controller console or Microsoft Windows Clients. The solution described in this paper provides an alternative smart interface to the storage system. This research effort is focused on investigating the potential of using mobile smartphone software application as a smart user interface protocol to communicate with the storage system network.

Keywords: Mobile computing, mobile smartphone, mobile user interface, portability, storage system network.

1. Introduction

This paper is focused on investigating the potential and feasibility of using mobile smartphone software as a smart user interface to communicate with a storage system network. The effort does not attempt to expose every possible feature of a storage system network, but endeavors to demonstrate the process of transforming the functions of a system management platform called DataONTAP. DataONTAP is an on-demand, efficient, flexible and scalable storage operating system platform to assist system administrators to manage data and applications, and to scale-out storage infrastructure growth on the NetApp storage system network to a more available and accessible smartphone environment, with the advantages of enhanced mobility and smart interface for system administrators.

1.1 Proposed Areas of Study and Academic Contribution

The primary interest in the research work is to develop a model to map a complex system, such as the NetApp storage system administration, to an interface that is as limited as the iPhone OS. The scope of the work is to allow the manipulation of the standard filer configuration and management through the small screen and limited capabilities of the iPhone [1].

Section 2 describes the current state of the art. Section 3 of this paper discusses the requirements and constraints of the application and the environment. Section 4 describes the system architecture and subsystems. Section 5 describes the technologies used in the development of the research work. Section 6 describes the design of the proposed system. Section 7 describes the implementation of the system and how elements from a Windows based User Interface (UI) are mapped to elements of an iOS application [2]. Section 8 provides a summary, and Section 9 includes the conclusion statements and elaborates the major points regarding the research work mentioned in the paper.

2. Current State of the Art

Management of NetApp storage systems is handled through a variety of interfaces. The storage system has a Command Line Interface (CLI) that is accessible through Secured Shell (SSH), Telnet, and a serial console connection. The storage systems also present a web based utility for configuration and interaction called FilerView. There are two client side applications for managing storage system that are provided by NetApp, Operations Manager and System Manager. System Manager is a client side tool that integrates with Microsoft Management Console to manage the storage system [3]. Operations Manager is a client side monitoring tool that gives a single place to get performance data from a set of storage systems [4]. NetApp also provides a Software Development Kit (SDK) for custom tools and automation of the storage systems. Many of the tools described above leverage the SDK for access to the storage systems. The supported version of the SDK includes tools for Windows and UNIX operating systems and C++, C#, Java, and Perl languages [5]. NetApp also has published an unsupported Objective-C SDK called CocoaONTAP [5] [6] that provides access for applications built for Apple Mac OSX systems and the iPhone [6] [7]. The CocoaONTAP SDK is the foundation of this research.

From the finding of a recent InformationWeek 2013 Mobile Device Management and Security Survey, it was found that both smartphones and tablets have been playing an important role in raising company business productivity. For smartphones, the percentage of the companies which agreed the statement is increased from 82% (in 2011) to 87% (in 2012). For tablets, the percentage is increased from 79% (in 2011) to 91% (in 2013). This is a trend. More and more companies have been actively looking into advanced features available on the smart mobile devices to apply them in designing new software tools and processes to enhance the efficiency and productivity of business operations. In recent years, the User Interface (UI) features and capabilities available on smartphones (and tablets) become a major new software design effort area for research and development. Most of smart mobile software vendors provide a set of software tools to allow software engineers to design mobile software applications [3] [8] [9] [10].

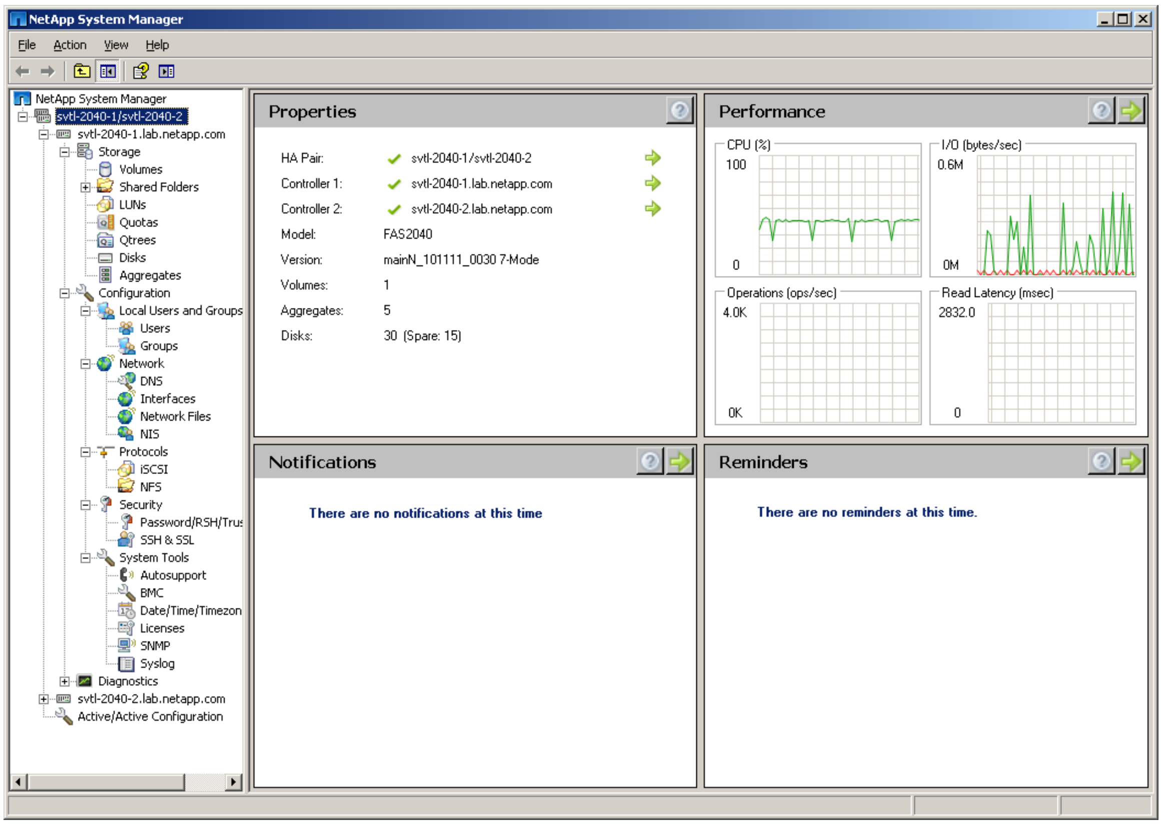
As mobile software code increases so will the number of software faults and vulnerabilities; hence the need for adopting a secured software development process model. The mobile software development process and its lifecycle are becoming major focus for software engineering research. The area like mobile software security and vulnerability is attracting researchers to gain deeper and effective solutions.

3. System Analysis Process

In order to be able to design and develop a mobile application that provides some of the features and functionality of System Manager it is necessary to analyze the existing application. The analysis that we were able to do was limited to the application UI and the interfaces that that application uses when accessing the storage system. Further analysis and possible code reuse could be achieved with access to the source code for the application to be converted.

3.1 Application UI Analysis

The analysis of the user interface of the application can be achieved by examining each element of the UI and the functions that those elements provide. The analysis breaks the application down into individual components to determine the functionality that is implemented. In the case of System Manager (shown in Fig. 1), the left side frame of the application consists of a list of areas that can be managed through the application.



HA Pair: ✓ svtl-2040-1/svtl-2040-2
Controller 1: ✓ svtl-2040-1.lab.netapp.com
Controller 2: ✓ svtl-2040-2.lab.netapp.com
Model: FAS2040
Version: mainN_101111_0030 7-Mode
Volumes: 1
Aggregates: 5
Disks: 30 (Spare: 15)

There are no notifications at this time

There are no reminders at this time.

Figure 1. System Manager

The application is broken down in to the following areas indexed by the left frame.

1. Storage

This category contains a collection of features for storage management. The right hand frame shows some frequent tasks when this item is selected.

1.1. Volumes

This category presents volume information in the right hand frame. This information includes, name, size, containing aggregate and other information about the volumes in the storage system. This window also allows for creation of new volumes and management functions for existing volumes.

1.2. Shared Folders

This category only contains the Exports subcategory which contains information about the exported directories and volumes on the storage system.

1.2.1. Exports

This section shows the detail about exported file systems on the storage system.

1.3. LUNs

This section shows the LUN information for SAN related storage on the storage system.

1.4. Quotas

This section shows the quota information in use on the storage system. Quotas are a mechanism that administrators can use to limit the amount of space available to any one group or individual using the storage system.

1.5. Qtrees

This section shows the qtree usage on the storage system. Qtrees allow administrators to control storage use in more detail than is allowed through volumes. Qtrees are similar to directories in a Microsoft Windows system but can be exported by the storage system as well as having quotas applied to them.

1.6. Disks

This section shows the details of the disks in the system including the current usage of the disks, and the node in an HA pair that owns them. This section also allows spare disks to be added to aggregates.

1.7. Aggregates

This section provides details of aggregates similar to what is shown for volumes by the Volumes section. This section also allows for the creation of new aggregates and other aggregate management functionality.

2. Configuration

2.1. Local Users and Groups

This section allows for administration of users and groups that are locally managed by the storage system.

2.2. Network

The networking section has four subsections for DNS, NIS, Interfaces and Network Files. These sections allow the administrator to manage the name services, network interface settings and the host files.

2.3. Protocols

This section allows the administrator to manage iSCSI and NFS storage information. If CIFS is configured on the storage system it should also appear in this section.

2.4. Security

This section handles the management of security settings related to the password of the storage system.

2.5. System Tools

The System Tools section provides a set of functions that allow the administrator to manage some miscellaneous features that are not easily included in other parts of the application. This includes things like AutoSupport, date/time settings, license management, and system logs.

3. Diagnostics

The systems I was using were unable to use the Diagnostics section so I am uncertain of the features that are contained in this section.

4. Active/Active Configuration

This section is outside of the sets of entries for each node of the storage system. This section allow the storage administrator to manage to configuration of the HA Pair for node failover. This section also allows to administrator to force a takeover or giveback of in the storage system.

This breakdown of functions and responsibilities summarizes the features that can be implemented by a management application. System Manager is a general management application and therefore must provide a broad a set of functions as possible.

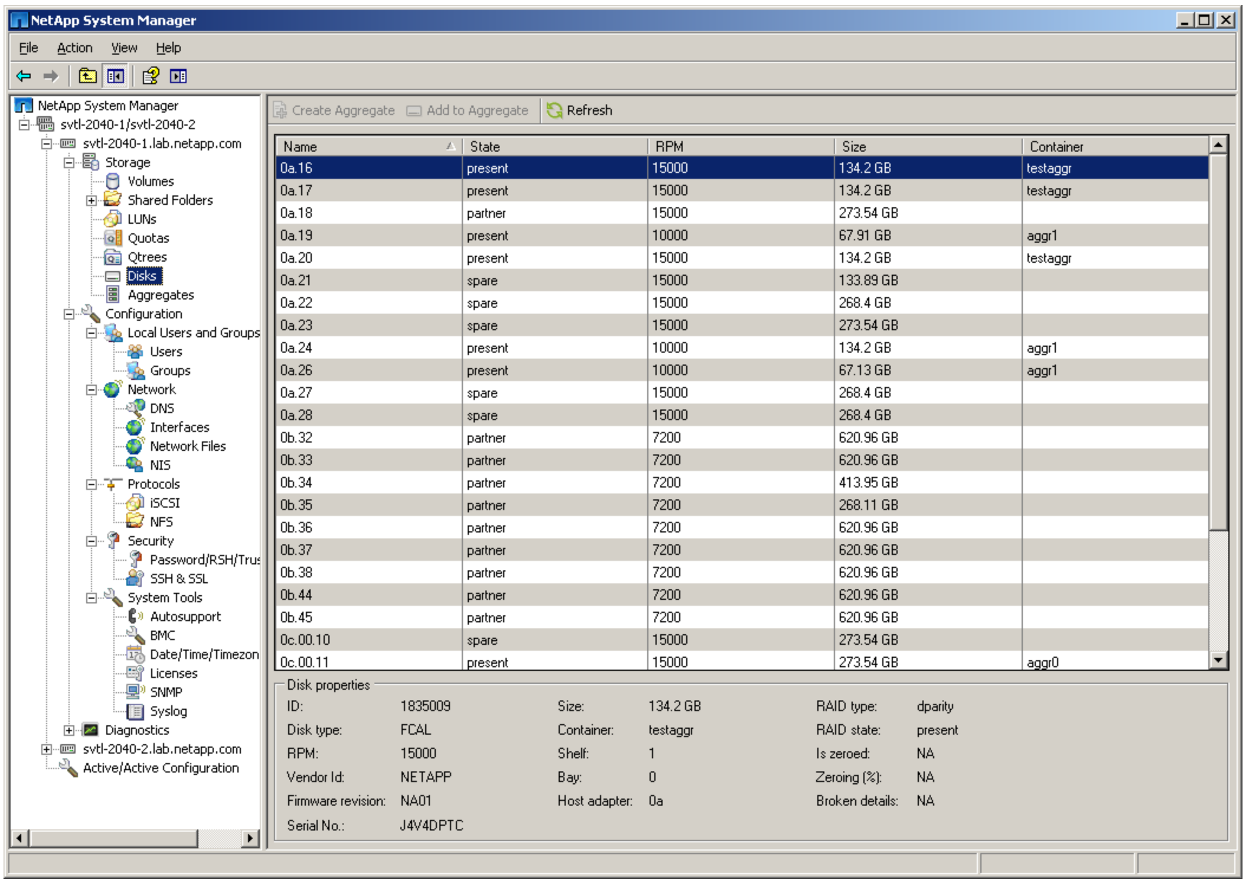
The target application for a mobile device has a different set of goals and needs. The mobile device usage pattern is going to be more limited. The initial feature selection is based on a set of use cases where an administrator receives a notification of a problem and wishes to resolve it quickly. The notifications about storage space and usage on a storage system and failover events were the triggers for the use cases that were selected. These use cases will require access to disk, aggregate and volume information about the storage system as well as the Active/Active control described above.

In order to provide the needed information to map the functionality between System Manager and the mobile application, further analysis of the planned components is needed.

3.1.1 Disk Component Analysis

The Disks function of System Manager provides a table of details about the disks in the system. The table consists of Name, State, RPM, Size and Container. Each disk has a disk name which is includes the HBA port and one or more numeric identifier that define the shelf and bay that the disk occupies. The State field indicates the current usage state of the disk. Present indicates that it is a disk in a volume or aggregate local to the node. A spare state indicates that the disk is owned by the local node but is not currently in a volume or aggregate. A partner state indicates that the disk is owned by the partner node in an HA pair, other usage information is not given for partner disks. Broken disks are reported as broken state. The RPM column indicates the RPM of the disk, Fiber Channel and SAS disks are generally 10,000 or 15,000 RPM while SATA disks are generally 7,200 RPM disks. The size field of the table indicates the usable size of the disk in gigabytes. The Container field indicates that aggregate or volume that the disk is a part of if it is currently in the present state.

Below the table of disks is detailed view of information about the disk that is selected. The detailed view shows the RPM, Size, Container, and RAID state that are presented in the summary table. Also in this view is the disk ID, disk type (FCAL, SAS, SATA, etc.), and several other fields that provide more detail about the disk. See Fig. 2.



Name	State	RPM	Size	Container
0a.16	present	15000	134.2 GB	testaggr
0a.17	present	15000	134.2 GB	testaggr
0a.18	partner	15000	273.54 GB	
0a.19	present	10000	67.91 GB	aggr1
0a.20	present	15000	134.2 GB	testaggr
0a.21	spare	15000	133.89 GB	
0a.22	spare	15000	268.4 GB	
0a.23	spare	15000	273.54 GB	
0a.24	present	10000	134.2 GB	aggr1
0a.26	present	10000	67.13 GB	aggr1
0a.27	spare	15000	268.4 GB	
0a.28	spare	15000	268.4 GB	
0b.32	partner	7200	620.96 GB	
0b.33	partner	7200	620.96 GB	
0b.34	partner	7200	413.95 GB	
0b.35	partner	7200	268.11 GB	
0b.36	partner	7200	620.96 GB	
0b.37	partner	7200	620.96 GB	
0b.38	partner	7200	620.96 GB	
0b.44	partner	7200	620.96 GB	
0b.45	partner	7200	620.96 GB	
0c.00.10	spare	15000	273.54 GB	
0c.00.11	present	15000	273.54 GB	aggr0

Disk properties

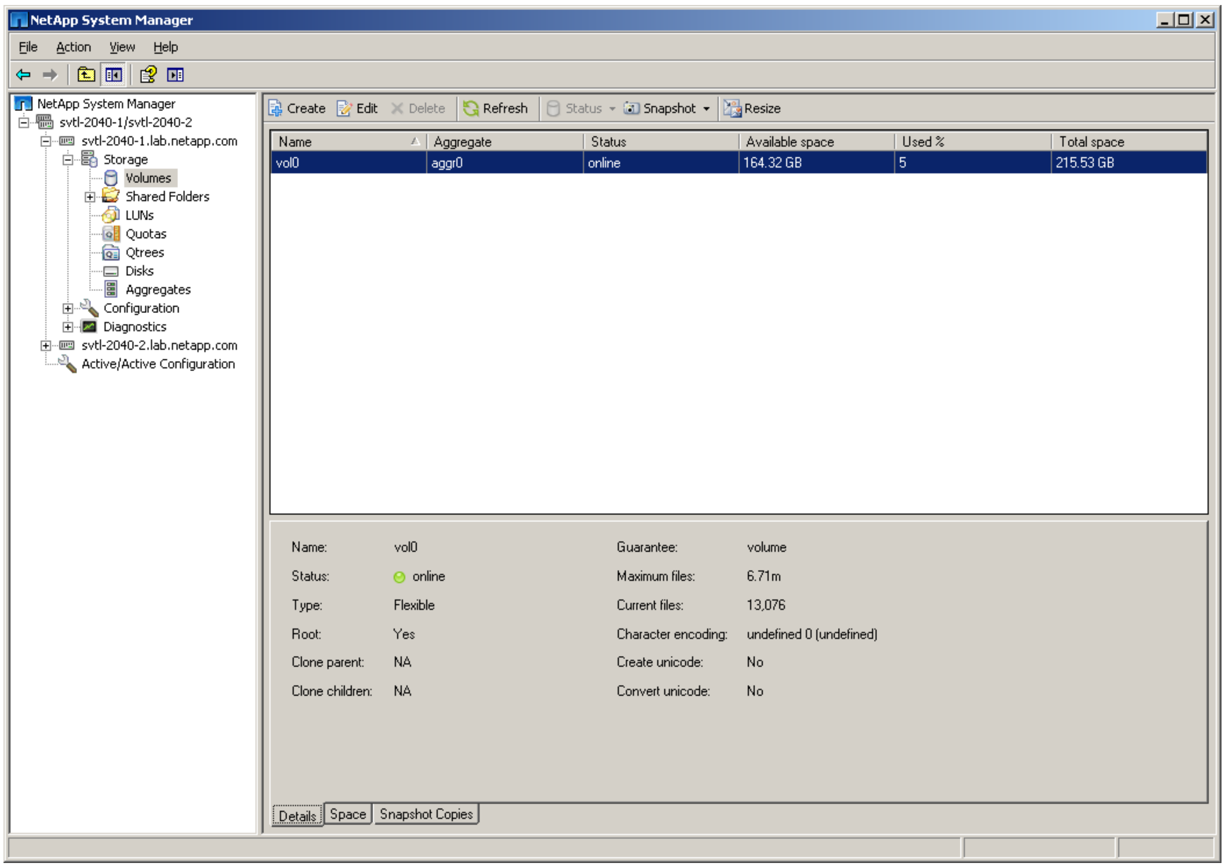
ID:	1835009	Size:	134.2 GB	RAID type:	dparity
Disk type:	FCAL	Container:	testaggr	RAID state:	present
RPM:	15000	Shelf:	1	Is zeroed:	NA
Vendor Id:	NETAPP	Bay:	0	Zeroing (%):	NA
Firmware revision:	NA01	Host adapter:	0a	Broken details:	NA
Serial No.:	J4V4DPTC				

Figure 2. Disk View

The Disk window also provides functions to add a disk to an existing aggregate as well as creating new aggregates from multiple selected disks. The only other function provided by this interface is a refresh button to poll the storage system for any updates that may have occurred since the application was started.

3.1.2 Volume Component Analysis

The right portion of the Volume window is very similar to the Aggregate window. The view presents a summary table of volumes that includes the name, hosting aggregate, status, available and total space and the percentage of used space. The lower half of the view contains a set of tabs that provide further information about the selected volume. See Fig. 3 for the details of the view.



NetApp System Manager
- svtl-2040-1/svtl-2040-2
  - svtl-2040-1.lab.netapp.com
    - Storage
      - Volumes
      - Shared Folders
      - LUNs
      - Quotas
      - Qtrees
      - Disks
      - Aggregates
    - Configuration
    - Diagnostics
  - svtl-2040-2.lab.netapp.com
  - Active/Active Configuration

Name	Aggregate	Status	Available space	Used %	Total space
vol0	aggr0	online	164.32 GB	5	215.53 GB

Name:	vol0	Guarantee:	volume
Status:	online	Maximum files:	6.71m
Type:	Flexible	Current files:	13,076
Root:	Yes	Character encoding:	undefined 0 (undefined)
Clone parent:	NA	Create unicode:	No
Clone children:	NA	Convert unicode:	No

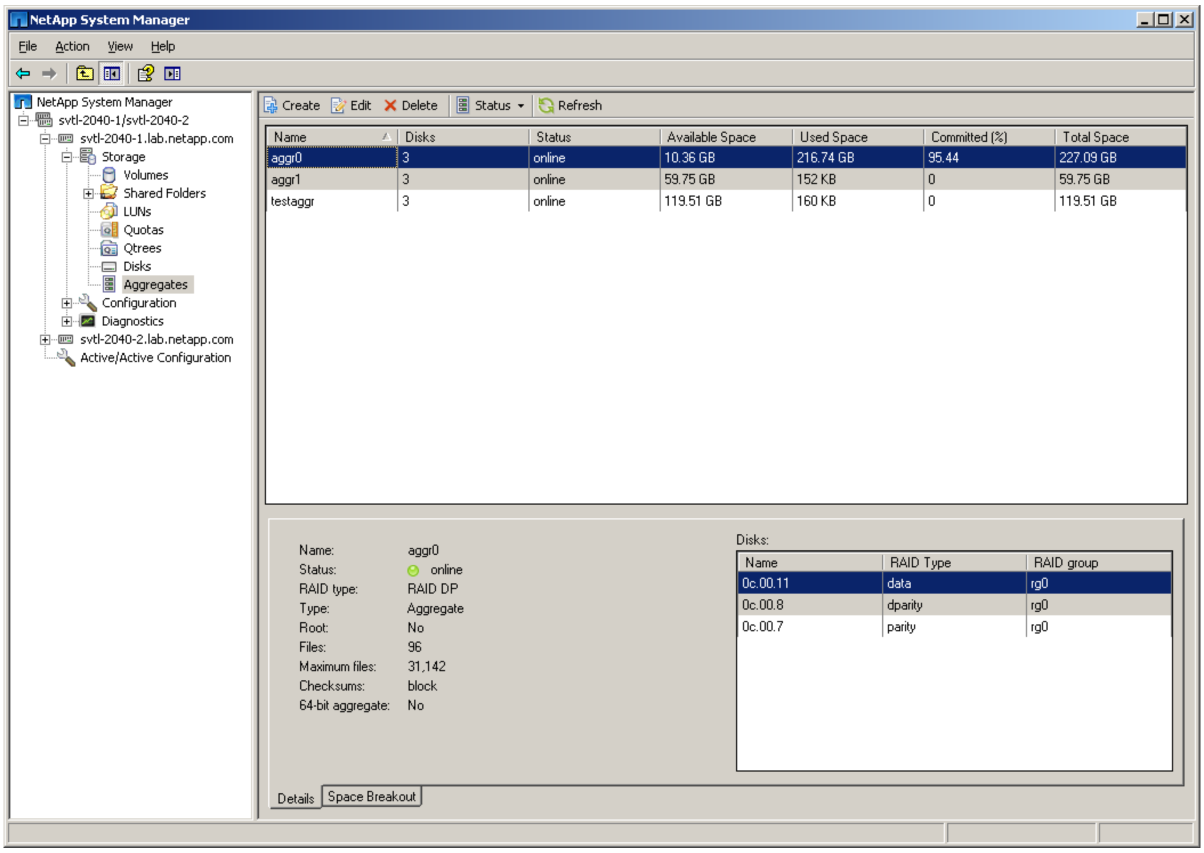
Figure 3. Volume View

The Volume view also provides functions to create and destroy volumes, modify volume settings and create snapshots of volumes.

3.1.3 Aggregate Component Analysis

The upper right portion of the Aggregate window (shown in Fig. 4) shows a table of the aggregates in the system. The table has columns for the name, number of disks, status, available space, used space, percent committed and total space. The lower half of the aggregate view is divided between two tabs, Details and Space Breakout both of which display information about the aggregate selected in the summary table. The details tab shows information such as RAID type, type of aggregate (aggregate or traditional volume), Root flag, files, maximum files, Checksum Type and whether or not it is a 64-bit aggregate. The right half of the Details tab contains a summary table of the disks in the aggregate with disk name, usage type (parity, dparity or data), and the RAID group that the disk belongs to. The Space breakout tab shows space usage in a table and a graph for the aggregate. The table also shows the method of space guarantee for the volumes.

The aggregate view provides functionality to create aggregates, offline and destroy of aggregates, and editing some of the settings of the aggregates.



Name	Disks	Status	Available Space	Used Space	Committed (%)	Total Space
aggr0	3	online	10.36 GB	216.74 GB	95.44	227.09 GB
aggr1	3	online	59.75 GB	152 KB	0	59.75 GB
testaggr	3	online	119.51 GB	160 KB	0	119.51 GB

Name:	aggr0
Status:	online
RAID type:	RAID DP
Type:	Aggregate
Root:	No
Files:	96
Maximum files:	31,142
Checksums:	block
64-bit aggregate:	No

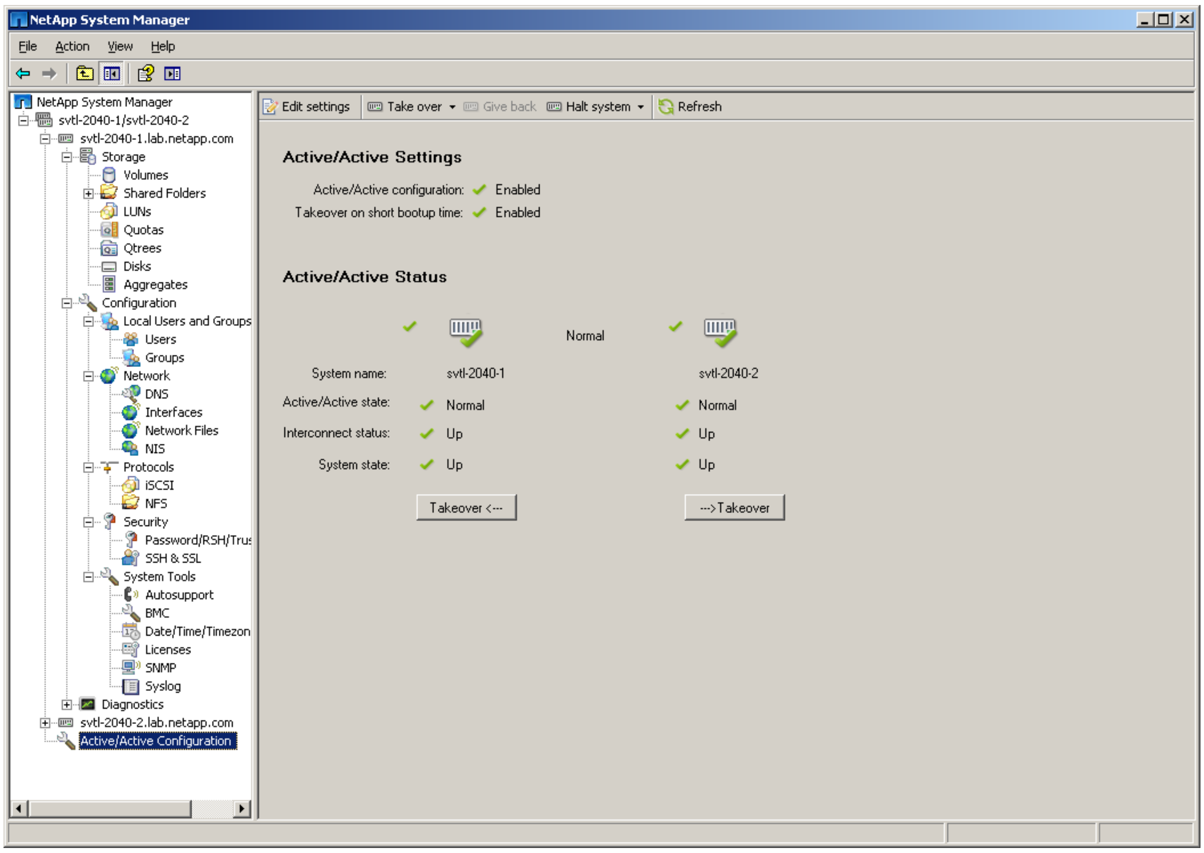
Disks:

Name	RAID Type	RAID group
0c.00.11	data	rg0
0c.00.8	dparity	rg0
0c.00.7	parity	rg0

Figure 4. Aggregate View

3.1.4 Cluster Component

The Cluster view (shown in Fig. 5) provides controls for takeover and giveback between the nodes of a high availability (HA) system. This view also provides access to enabling and disabling the HA abilities.



Active/Active Settings

Active/Active configuration:	✓ Enabled
Takeover on short bootup time:	✓ Enabled

Active/Active Status

	✓	Normal ✓
System name:	svtl-2040-1	svtl-2040-2
Active/Active state:	✓ Normal	✓ Normal
Interconnect status:	✓ Up	✓ Up
System state:	✓ Up	✓ Up

Figure 5. Cluster View

3.2 Mobile Application UI Requirements

Working from the analysis of System Manager described above, the UI requirements for the mobile application will be limited to the Disk, Aggregate, and Volume functions. The use cases described above include functions such as adding disks to an aggregate; modifying the size of a FlexVol to resolve an out of space condition and examining the state of the aggregates, volumes and disks in the system.

The mobile system screen is limited to the point where each table presented in System Manager in different areas of the view will need to be a separate view in the mobile application. The data will also need to be reorganized into views that will provide useful information to the user at each level but may not be able to show all elements of each table in the equivalent table on the mobile UI. An example of this reduction is in the aggregate summary table. The System Manager aggregate summary table presents multiple fields with different information. The mobile application summary table will present the name, and state of the aggregate and leave the other elements to be displayed by the detailed view.

The mobile application UI will need a way to select which area to examine and administer. Each area can initially present a list similar to the summary list of data that is common for each area of the System Manager that is being modeled. The summary list can be used to select a detailed view. The detailed view can provide access to information that is similar in purpose to the information System Manager presents in the lower half of the views for each area.

3.3 Mobile Application Functional Requirements

The functional requirements for the mobile application are based on the functionality described above in the System Manager analysis. The functions to be implemented are listed here in summarized form to simplify development and verification of completeness.

3.3.1 Disk Functions

Examine the disks in the system and the current state of the disks

Add selected disk to an aggregate

3.3.2 Volume Functions

Create a volume within an aggregate

Offline a volume

Online a volume

Destroy a volume

Modify size of a volume within the constraints of the aggregate

3.3.3 Aggregate Functions

Create an aggregate with a specified number of spare disks

Offline an aggregate

Online an aggregate

Destroy an aggregate

3.3.4 Cluster Management

Initiate a takeover from either node

Initiate a giveback

3.4 ManageONTAP API Analysis

The ManageONTAP API is a collection of web service interfaces provided by the DataONTAP operating system on NetApp storage systems. The API provides interfaces to perform almost every function that is provided by the DataONTAP CLI. Using the functional requirements as a guide it is possible to identify which API calls will be needed by the various components of the application.

3.4.1 Disk Functions

The disk-list-info API provides a collection of data for each disk in the system. The level of detail provided by this function is exceeds the requirement to display summary data about the disks.

In order to add a disk to and aggregate, aggr-add is used. This function requires a disk name and an aggregate name.

3.4.2 Volume Functions

Volume information is provided by volume-list-info. The information returned by this command includes the information that is needed for the summary as well as additional information that could be used in future development.

The volume-create, volume-offline, volume-online and volume-destroy commands provide the functionality to create, offline, online and destroy volumes as the names imply.

The volume-size API is a setter/getter function and can be used to retrieve the current size or change the size of the volume depending on the usage.

3.4.3 Aggregate Functions

The aggregate functionality specified in the Functional Requirements above is provided by the APIs aggr-create, aggr-online, aggr-offline, aggr-destroy, aggr-list-info. These commands are similar in behavior to the volume commands.

3.4.4 Cluster Functions

The cluster functionality to provide information about the current state of the HA configuration is provided by cf-status API. The cf-takeover and cf-giveback APIs provide the functionality to initiate the takeover and giveback between the nodes of an HA pair.

As is implied by the complete feature set of System Manager, there are many more APIs available from ManageONTAP. These APIs provide a rich environment to implement additional features in the iPhone application if they are needed.

4. System Architecture

This system is divided into several components. Each component addresses one of the areas that were selected for requirements. The interface of the application to the NetApp system being managed is through the CocoaONTAP library that is provided by NetApp. The application consists of several packages that encapsulate a specific feature set. In addition to the CocoaONTAP library which provides access to the storage system and handles the communication between the storage system and the application, there is one other set of infrastructure components. The System Selection component will present systems that have been used previously and allow for them to be selected for management. The login package will verify that filer, username and password match and store that data for future use by the System Selection component. The Main Menu is the last piece of infrastructure provided that allows for the selection areas of management. The application contains a set of features and presents a list of the features as the main menu once a node has been selected. Each item on the main menu will load the associated package which contains the entire Model-View-Controller pattern for that feature. Each package will be passed the node information needed to communicate with the node and will instantiate the data model objects that are required for the feature that it supports. Each package will be responsible for handling any errors that are returned from the node through the CocoaONTAP library.

4.1 Architecture Subsystems

This section describes the roles of the system components shown in Fig. 6. These components show the organization of the iPhone application based on the functional requirements determined from the analysis of System Manager.

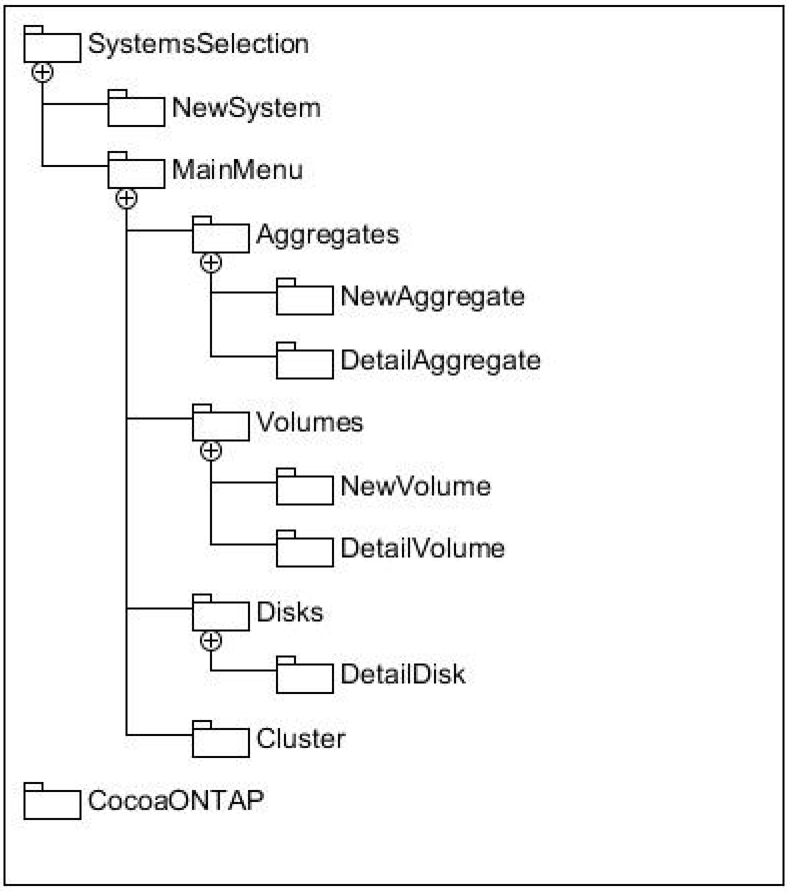


Figure 6. System Architecture

CocoaONTAP represents the supporting library provided by NetApp to develop automation for system administration. This element is not strictly part of the system but it provides the necessary interface between the iPhone application code and the storage system.

The NewSystem component gathers the credentials and system information from the user that is needed to communicate with the storage system. This component also verifies the credentials are correct and saves the system credentials for future use. The SystemSelection component is the starting point for the application after launch. The List is populated from a file of saved system credentials. Selection of a system from the list will load the MainMenu Component.

The Aggregate, Volume, and Disk components perform the management tasks associated with those parts of the storage system. Each component contains the data model that represents the state of the respective part of the storage system.

The Aggregate component deals with the configuration and maintenance of aggregates which are related to RAID groups.

The Volume component deals with the configuration and maintenance of volumes which are the storage components that are exported as NFS mount points and CIFS shares.

The Cluster component allows for the verification of the current state of a clustered system and control of node takeover and giveback.

4.1.1 Aggregates Subsystem

The Aggregates subsystem is made up of a data model that provides the required information about the aggregates in the storage system, and several views and their associated view controllers that allow the user to see different parts of the data model and perform certain operations on the storage system. The views provided by the subsystem provide a summary of all aggregates on the node as the initial view, a detailed view of a specific aggregate and a view that provides the ability to create new aggregates. The detailed view also provides functionality to offline, online and destroy the aggregate.

The data model for the aggregates subsystem is made up of an array of Aggregate objects. The Aggregate object is a custom object that contains the required data for each aggregate in the storage system.

4.1.2 Volumes Subsystem

The Volumes subsystem is very similar to the Aggregates subsystem. The primary difference is the set of information provided by the Volume object.

4.1.3 Disks Subsystem

The Disks subsystem contains a data model that is different from the Aggregates and Volumes subsystems. The data model for disks consists of several lists that are mutually exclusive. Multiple lists are used to simplify the development of the views and views controllers. Each list represents a different collection of disks. For example, there is a list of disks for each aggregate, separate lists for spare, broken and partner disks. The increased complexity of the data model provides a simpler mechanism for grouping the disks according to their current usage in the views.

4.1.4 Cluster Subsystem

The Cluster subsystem is simpler than the other subsystems because it has only a single view and view controller and the data model only requires minimal information about the current state of the storage system.

5. System Technology

The technologies used in this system work are primarily client side technologies, Cocoa and iOS [11]. The interface to the storage system leverages a library developed by NetApp, Inc. called CocoaONTAP that provides a set of classes for communicating commands to the storage system and returning the results from the storage system to the other areas of code. The mechanism used by CocoaONTAP to communicate with the storage system is XML over HTTP or HTTPS depending on the settings selected by the user of the CocoaONTAP library.

The Cocoa Touch environment provided by Apple provides a wide array of libraries that simplifies design and development of applications for the iPhone environment. The iPhone Human Interface Guidelines describes the differences between the iPhone user interface and describes some of the design options that should be considered when building applications for the iPhone platform. iPhone OS offers the TableView and NavigationController which together present an environment that allows for the translation of and application like System Manager to the iPhone [12]. NavigationController helps manage a set of views, in this case TableViews, which present a hierarchy of data. This application model is well defined in the Table Views Programming Guide [2] [11] [13].

The Manage ONTAP API that CocoaONTAP uses to communicate with the filer is well documented and used both internally by System Manger as well as custom tools and automation developed by NetApp customers. Manage ONTAP API provides programmatic interfaces to almost every feature of the Storage System. This API will be used within the application to collect data from the storage system and send commands. CocoaONTAP provides the infrastructure for building Manage ONTAP commands, sending them over the network, and parses the response into Objective-C objects [6] [13] [14].

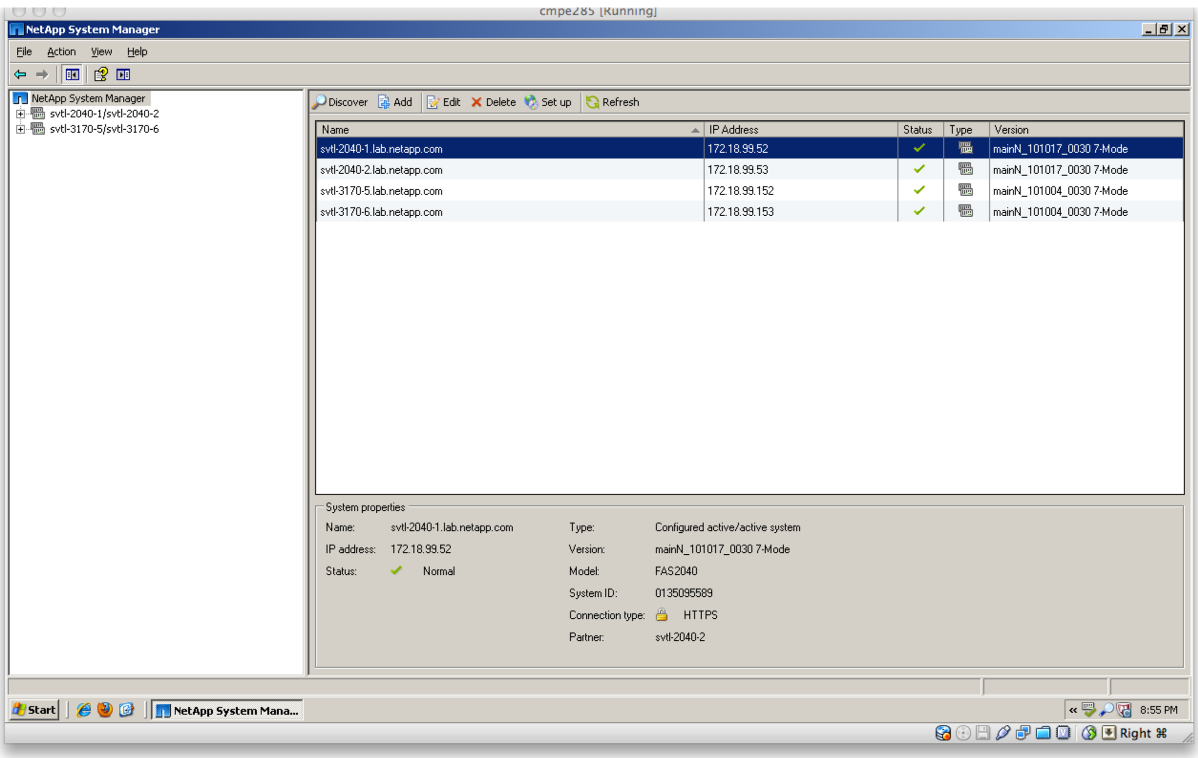
The design of the system is loosely structured based on the functionality presented by the System Manager tool provided by NetApp. The relationship between System Manager and the iPhone application is described in Section 3.

The NetApp storage systems consist of one or two controllers which can be arranged in an Active/Active cluster for high availability and a set of disk shelves. Aggregates are a collection of disks which can be divided into volumes which can be exported by the system and mounted by clients. There are many features of the NetApp storage system that provide features for disaster recovery and data protection. This application will focus on basic storage administration for aggregates, volumes and disks and the cluster behavior of the storage system.

6. System Design

The goal is to design a system that can run on the constrained screen and computing platform of the iPhone or iPod Touch that provides similar functionality to the administrative tools provided by NetApp. The functionality of System Manager is the model for the application design in terms of functionality. The application will be designed using

Model-View-Controller paradigm preferred by Apple in the Cocoa development environment. The applications design is driven by the limitations of the user interface provided by the iOS environment [2] [13], specifically a very limited screen; the features set desired for the final product; and the user interface provided by System Manager that is the inspiration for this system. The initial window presented by System Manager provides several pieces of information. In Fig. 7 the initial windows is shown.



Name	IP Address	Status	Type	Version
svtl-2040-1.lab.netapp.com	172.18.99.52	✓		mainN_101017_0030 7-Mode
svtl-2040-2.lab.netapp.com	172.18.99.53	✓		mainN_101017_0030 7-Mode
svtl-3170-5.lab.netapp.com	172.18.99.152	✓		mainN_101004_0030 7-Mode
svtl-3170-6.lab.netapp.com	172.18.99.153	✓		mainN_101004_0030 7-Mode

System properties

Name:	svtl-2040-1.lab.netapp.com	Type:	Configured active/active system
IP address:	172.18.99.52	Version:	mainN_101017_0030 7-Mode
Status:	✓ Normal	Model:	FAS2040
		System ID:	0135095589
		Connection type:	HTTPS
		Partner:	svtl-2040-2

Figure 7. System Manager Main Window

Along the left side of the window is a list of systems for which administrative credentials are known. The upper right window shows a list of nodes that belong to the systems listed on the left including information about the current state of the system, the IP address and the version of DataONTAP that is currently running on the node. The lower right portion of the window shows more information about the node selected in the upper right portion of the window.

6.1 View Design

The primary challenge with translating presentation information between a computer screen and an iPhone is selecting a set of data that is appropriate for the smaller display. The iPhone interface lends itself to presenting only a limited set of information at any given time. In the case of translating the rich desktop interface to the iPhone, partitioning the application into specific views is the simplest approach. Because the items displayed on the right side of the System Manager screen are primarily informational and aside from selecting different nodes to see detailed information are not interactive, the left hand view is of primary interest. Fig. 8 shows the System Selection screen from the application.

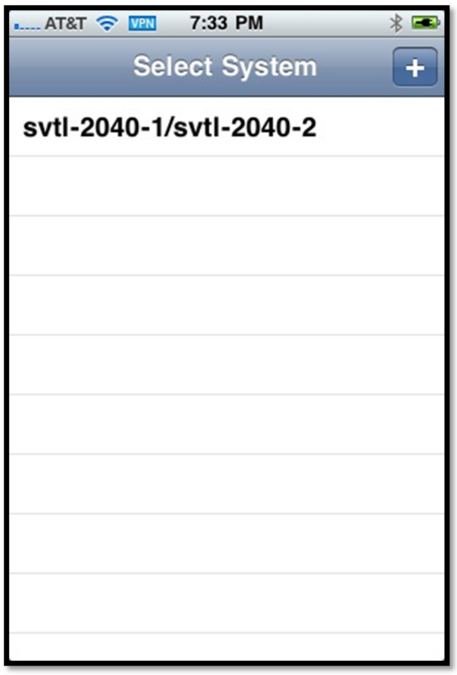


Figure 8. System Selection iPhone Screen

Fig. 9 shows the Main Menu view from the application that can be compared to Fig. 10 which shows the content of the left side of System Manager when the system hierarchy is expanded. Differences here demonstrate the subset of features and administrative functions that have been implemented in the application as compared to the full feature set provided by System Manager.

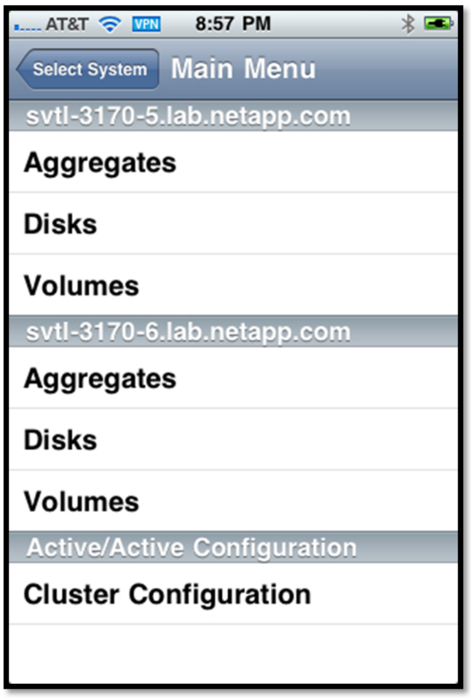
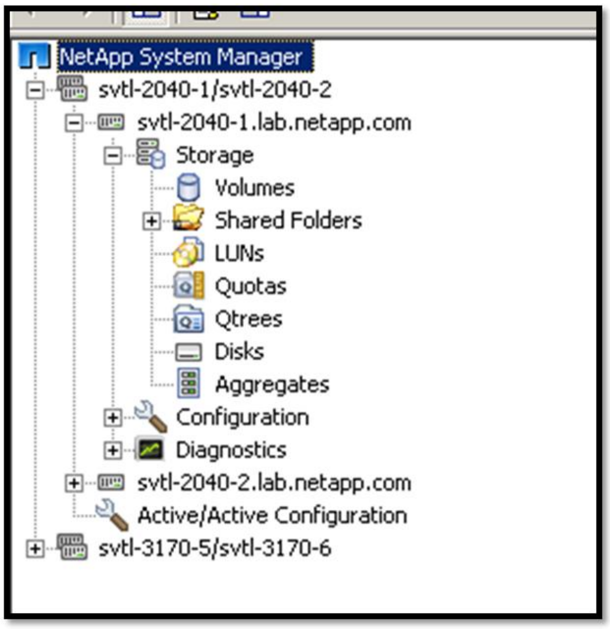


Figure 9. Main Menu

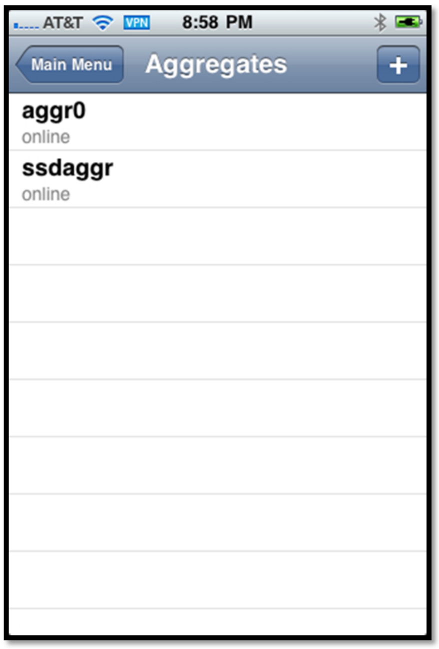
Each item on the list presents a new view (Fig. 10) that exposes more information and functionality to the user based on the features selected.



```
NetApp System Manager
  svtl-2040-1/svtl-2040-2
    svtl-2040-1.lab.netapp.com
      Storage
        Volumes
        Shared Folders
        LUNs
        Quotas
        Qtrees
        Disks
        Aggregates
      Configuration
      Diagnostics
    svtl-2040-2.lab.netapp.com
    Active/Active Configuration
  svtl-3170-5/svtl-3170-6
```

Figure 10. System Manager Expanded Tree

As the user navigates through the tree of views in the iPhone the views are analagous to the various levels of the hierarchy presented in System Manager for each node. When a leaf elemet is reached the source of the mapping moves from the left side of the screen to the right side. Aggregates in System Manager opens a list on the right side that shows a list of aggregates on the node with a summary of information about the aggregates. This information maps the the Aggregate list shown in Fig. 11.



aggr0
online
ssdaggr
online

Figure 11. Aggregate List

While the Microsoft Windows UI has certain accepted paradigms for UI layout and certain accepted button designs and behaviors, these standards are not the same on iOS. iOS has its own models and standards [13]. When examining the System Manager UI for features to implement in the iPhone application, certain conversions became obvious. In each table that allowed for the addition or creation of a new entry or element there was an Create button in the toolbar. This Create button becomes a "+" button in the navigation bar for each analogous view. The Edit button in the System Manager toolbar does not have an analogue in the iOS application. Because of the touch nature of selections editing must be selected from the detailed view that is presented when items are selected from the list of summary tables.

6.2 Controller Design

The controller objects contain the code that handles presenting the views and providing them with the data from the model objects. Each view will have a controller class that handles the creation of the model classes and the collection of data from the storage system to populate the model objects. The controller design is partially driven by the decision to use UITableView classes to develop this application. UITableViewControllers have certain required interfaces that provide the data that is required for the view.

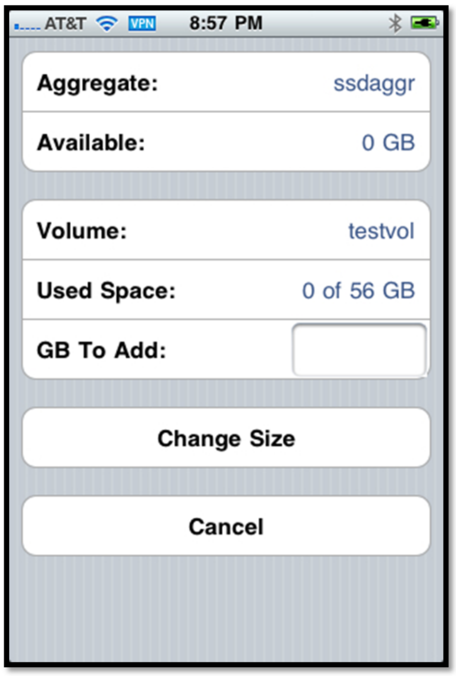
6.3 Model Design

The model components of the application depend on the information that is provided by the ONTAPI API for the commands that control the features and by the design decisions made around what features to support in each area. For example, aggregates are represented by an array of objects that contain a subset of the information available about the aggregate. The subset of data is determined by the selection of data that will be presented to the user. It would be possible to add additional fields to this object to provide additional information to the user. This information could be presented in additional views that could be added to the application as enhancements.

The aggregate model is representative of the various data models. The aggregate information is fetched from the storage system when the aggregate selection is made at the Main Menu. The data is stored in a custom object which contains a list of objects that represent each aggregate on the storage system. The custom Aggregate object contains the data that the application will use to present the detailed aggregate view.

In order to minimize the network activity required for the iPhone application, the data models are use a smaller set of data than what is presented by System Manager. The Aggregates view in System Manager presents both summary information about the aggregates as well as disk information for the selected aggregate. The ManageONTAP API that provides the summary information presented by System Manager, aggr-list-info, does not provide disk information. System Manager must make use of a second API call to disk-list-info to gather the information about the disks contained in the aggregate. The iPhone application does not collect this additional information in the Aggregate model and therefore lacks the ability to add disks to an aggregate from the Aggregates Views.

The iPhone application models do make use of multiple calls when it improves the user experience by providing critical information to aid the user in making decisions. This is demonstrated in the use of Aggregate models in the Volume component to provide information about the available space in an Aggregate when creating a volume or increasing the size of the volume. Fig. 12 shows the aggregate data that is presented when attempting to add space to a volume.


AT&T VPN 8:57 PM

Aggregate: ssdaggr
Available: 0 GB

Volume: testvol
Used Space: 0 of 56 GB
GB To Add:

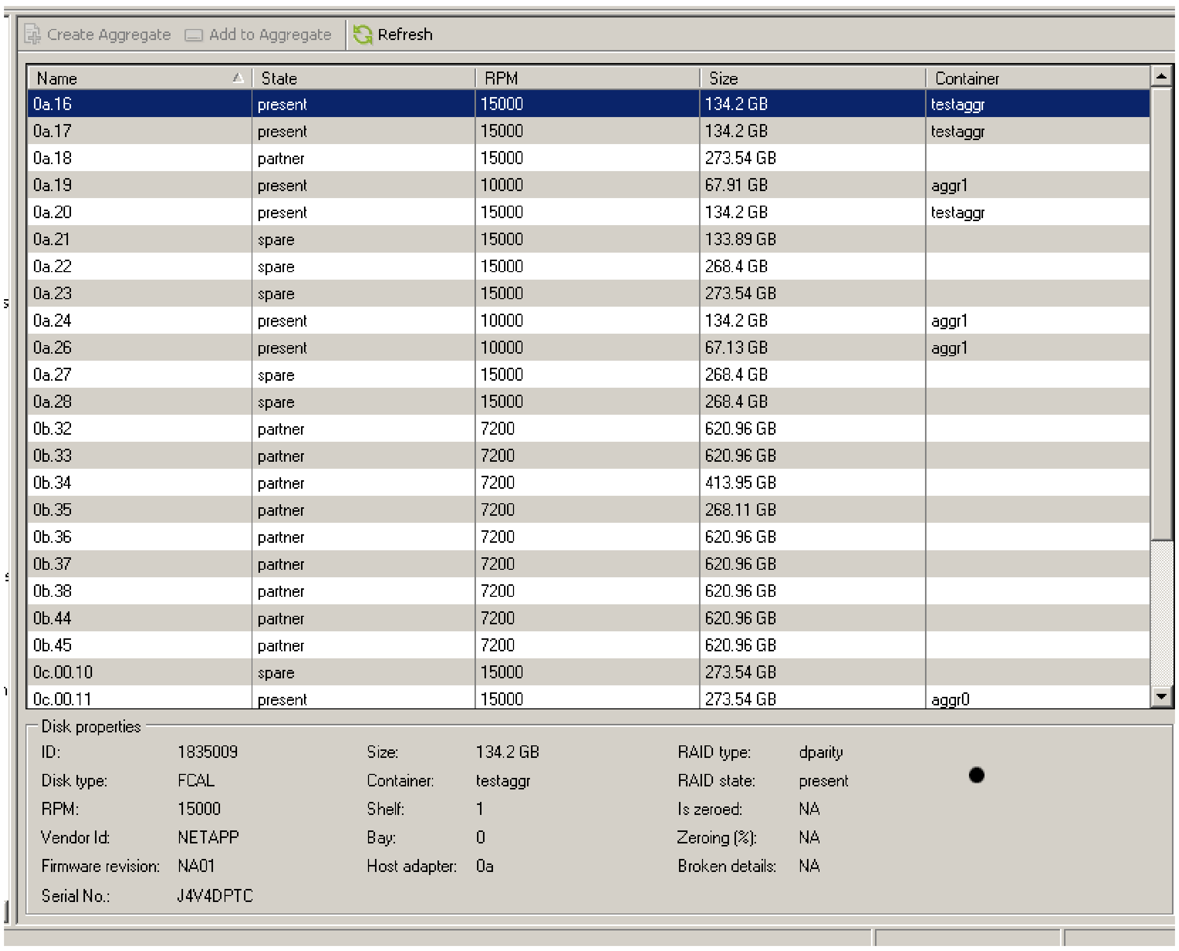
Figure 12. Volume Size Increase View

7. System Implementation

The procedures for converting System Manager functionality to a mobile application are primarily based on the differences in the user interfaces between Microsoft Windows and a touch based mobile device. The UI design of System Manager consists primarily as a series of tables showing different pieces of information from the storage system and several sets of dialog boxes that allow for the modification of the storage system components. The implementation of tables in iOS is a widely used UI paradigm. The TableView is used in many of the built in iOS applications as well as many third party applications. Mapping between the tables of System Manger and the TableView in iOS is straight forward. The mapping of dialog boxes to the iOS application requires more thought about what the dialog is doing and where it would fit into the iOS UI model.

7.1 Table Translation

The tables displayed by System Manger present multiple pieces of information in each view and are divided between a summary table of all elements and a detailed table of a single selected element. In iOS the limitations of screen size require that each of the tables be presented by separate views. Compare the Disk View in System Manger in Fig. 13. This view lists all of the disks in the system and some of the data about the disks. The detailed view in the lower part of the window displays additional information about the selected disk. Converting this information to iOS, the summary table becomes a list of disks grouped by current usage type which is the most important data point from the summary table for the use cases defined for the mobile application. Selection of a disk from this table then presents an additional view that shows the detailed data for the selected disk. Fig. 14 shows the two iOS views that show the data provided by the single System Manager View.

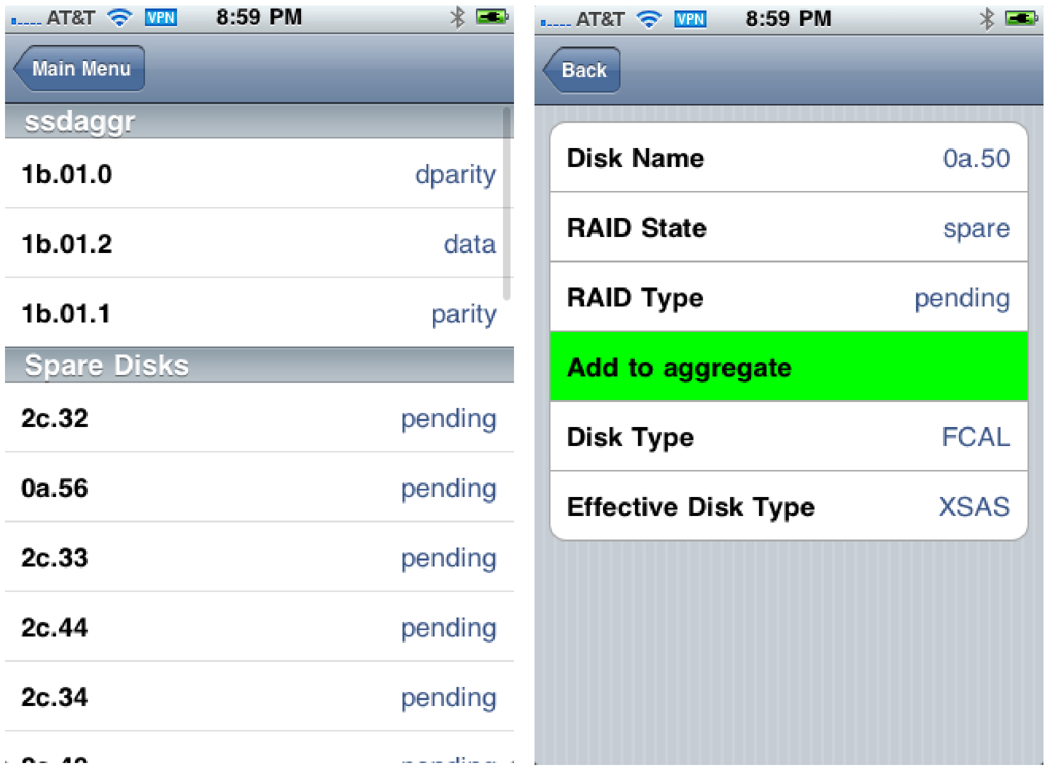


Name	State	RPM	Size	Container
0a.16	present	15000	134.2 GB	testaggr
0a.17	present	15000	134.2 GB	testaggr
0a.18	partner	15000	273.54 GB	
0a.19	present	10000	67.91 GB	aggr1
0a.20	present	15000	134.2 GB	testaggr
0a.21	spare	15000	133.89 GB	
0a.22	spare	15000	268.4 GB	
0a.23	spare	15000	273.54 GB	
0a.24	present	10000	134.2 GB	aggr1
0a.26	present	10000	67.13 GB	aggr1
0a.27	spare	15000	268.4 GB	
0a.28	spare	15000	268.4 GB	
0b.32	partner	7200	620.96 GB	
0b.33	partner	7200	620.96 GB	
0b.34	partner	7200	413.95 GB	
0b.35	partner	7200	268.11 GB	
0b.36	partner	7200	620.96 GB	
0b.37	partner	7200	620.96 GB	
0b.38	partner	7200	620.96 GB	
0b.44	partner	7200	620.96 GB	
0b.45	partner	7200	620.96 GB	
0c.00.10	spare	15000	273.54 GB	
0c.00.11	present	15000	273.54 GB	aggr0

Disk properties

ID:	1835009	Size:	134.2 GB	RAID type:	dparity
Disk type:	FCAL	Container:	testaggr	RAID state:	present
RPM:	15000	Shelf:	1	Is zeroed:	NA
Vendor Id:	NETAPP	Bay:	0	Zeroing (%):	NA
Firmware revision:	NA01	Host adapter:	0a	Broken details:	NA
Serial No.:	J4V4DPTC				

Figure 13. Disk View



AT&T VPN 8:59 PM

Main Menu

ssdaggr

1b.01.0	dparity
1b.01.2	data
1b.01.1	parity

Spare Disks

2c.32	pending
0a.56	pending
2c.33	pending
2c.44	pending
2c.34	pending

AT&T VPN 8:59 PM

Back

Disk Name	0a.50
RAID State	spare
RAID Type	pending
Add to aggregate	
Disk Type	FCAL
Effective Disk Type	XSAS

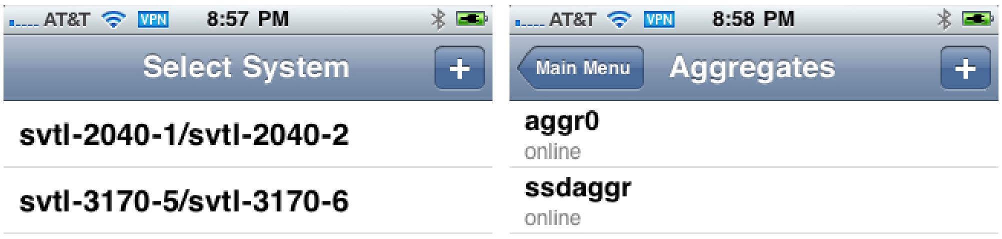
Figure 14. iOS Disk View

The translation of table views throughout the application focused on providing the most import single piece of data from the summary table in System Manager in the first level view of the iOS. In the case of disks the use information is most important. For volumes and aggregates, the state, whether online or offline was the most important piece of information to provide in the initial view.

7.2 Functional Elements Translation

For functional elements there are two types identified within the System Manager application, create and modify. The availability of these functional elements depends on the part of System Manager that is being used. For example there is no create function for disks.

The create functionality in System Manager relates to the element types being managed. In the initial System Manager view there is an Add Systems function, in Aggregates and it is a Create function. In iOS there is a fairly common model of adding a “+” button to the navigation bar when it is possible to add an entry to the displayed list. This same model is used in the iOS application to allow the user to access the functionality to create volumes and aggregates and add system credentials to the list of systems. Fig. 15 shows some of these items.


AT&T VPN 8:57 PM

Select System +

svtl-2040-1/svtl-2040-2

svtl-3170-5/svtl-3170-6

AT&T VPN 8:58 PM

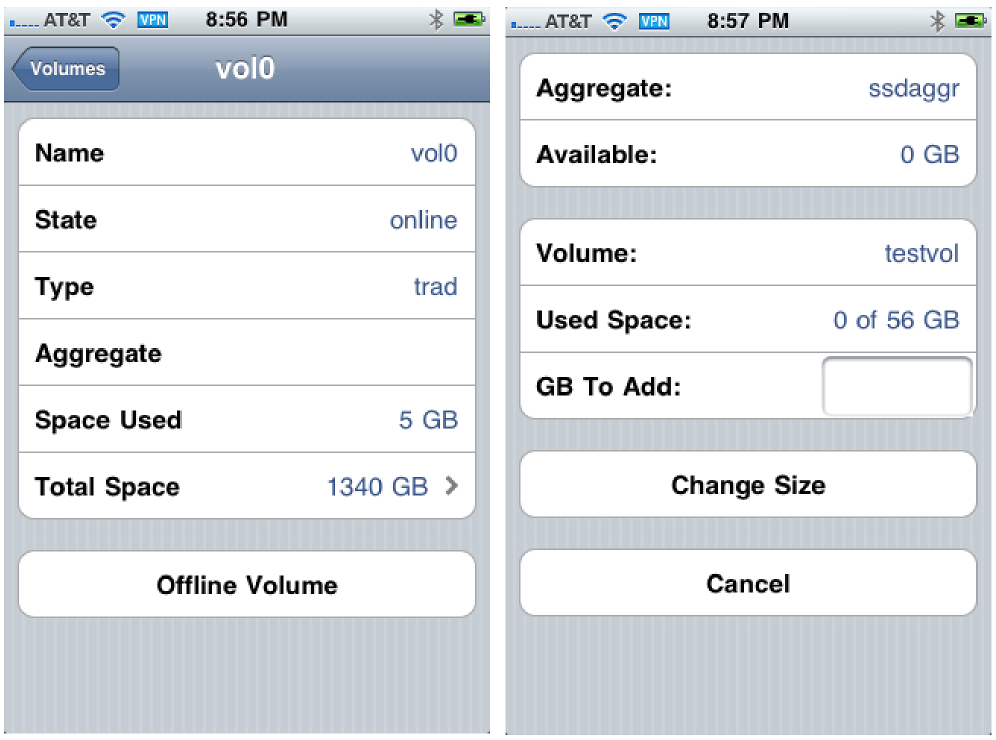
Main Menu Aggregates +

aggr0
online

ssdaggr
online

Figure 15. Create Buttons

The editing functionality provided by System Manager does not translate directly to the iOS application. The first thing to consider is that the item to be edited is the item selected in the summary table. In the iOS application selection from the summary table opens the detail view. The detail view must be the start point for edit functionality. For volumes the use case dictates that the user must be able to increase the size of a volume. The detailed view for volumes provides size information for the volume. In many iOS applications based on TableViews selection of a cell opens additional information about that content of the cell. In this application the selection of the volume size cell opens a view that provides the implemented edit functionality. Fig. 16 shows the detailed view and edit functions for the volume component of the application.


AT&T VPN 8:56 PM

Volumes vol0

Name	vol0
State	online
Type	trad
Aggregate	
Space Used	5 GB
Total Space	1340 GB

Offline Volume

AT&T VPN 8:57 PM

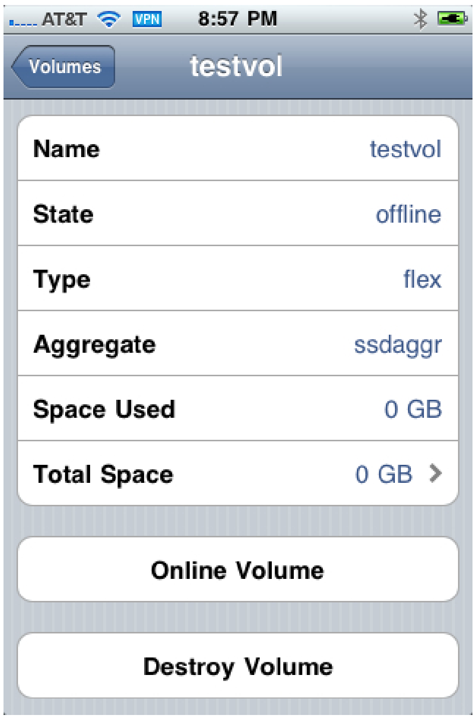
Aggregate:	ssdaggr
Available:	0 GB
Volume:	testvol
Used Space:	0 of 56 GB
GB To Add:	

Change Size

Cancel

Figure 16. Edit Functions

In addition to the ability to modify the size, the volume detail page also provides buttons to change the state of the volume and if the volume is offline, destroy it. The aggregate views provide the same features. Fig. 17 shows these features.

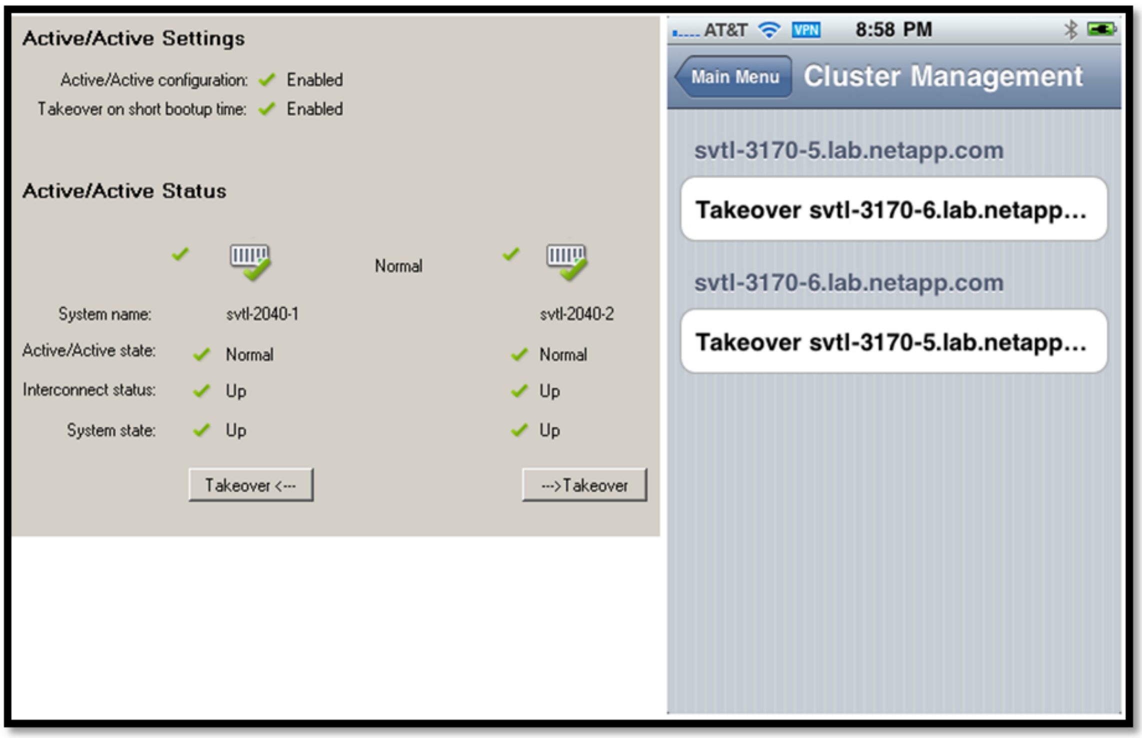


Name	testvol
State	offline
Type	flex
Aggregate	ssdaggr
Space Used	0 GB
Total Space	0 GB

Figure 17. Offline Volume View

7.3 Cluster Management Translation

The characteristics of System Manager’s view for Active/Active Configuration are very different from the other views that are implemented by the application. This view consists of some informational text and buttons to initiate a takeover of each node by the partner. Because of the simplicity of this view, the translation to iOS is almost direct. Fig. 18 shows the details of the System Manager window and the equivalent iOS application Window.


Active/Active Settings

Active/Active configuration: ✓ Enabled
Takeover on short bootup time: ✓ Enabled

Active/Active Status

Normal

System name:	svtl-2040-1	svtl-2040-2
Active/Active state:	✓ Normal	✓ Normal
Interconnect status:	✓ Up	✓ Up
System state:	✓ Up	✓ Up

Takeover <--- ---> Takeover

AT&T VPN 8:58 PM

Main Menu Cluster Management

svtl-3170-5.lab.netapp.com

Takeover svtl-3170-6.lab.netapp...

svtl-3170-6.lab.netapp.com

Takeover svtl-3170-5.lab.netapp...

Figure 18. Cluster Views

The performance of this application is not easily measured in an objective way. In qualitative terms the performance of the application is adequate to perform the basic functionality that is desired for the application. There is noticeable degradation of performance when using the application over 3G when compared to WIFI.

The iPhone and other iOS devices in their default state require that all applications be either loaded through a development system or installed through the iTunes App Store which is run by Apple. The deployment of this application if it were to be released by NetApp would by necessity be through the iTunes App Store. Most likely this application would be offered under similar terms to other system management tools and it would be up to the administrators of the storage system to acquire the application and configure the storage systems that they manage to support the application.

Maintenance, like deployment, is accomplished through the iTunes App Store. When a new or updated version of the application becomes available the device on which it is installed will indicate that updates are available in the store.

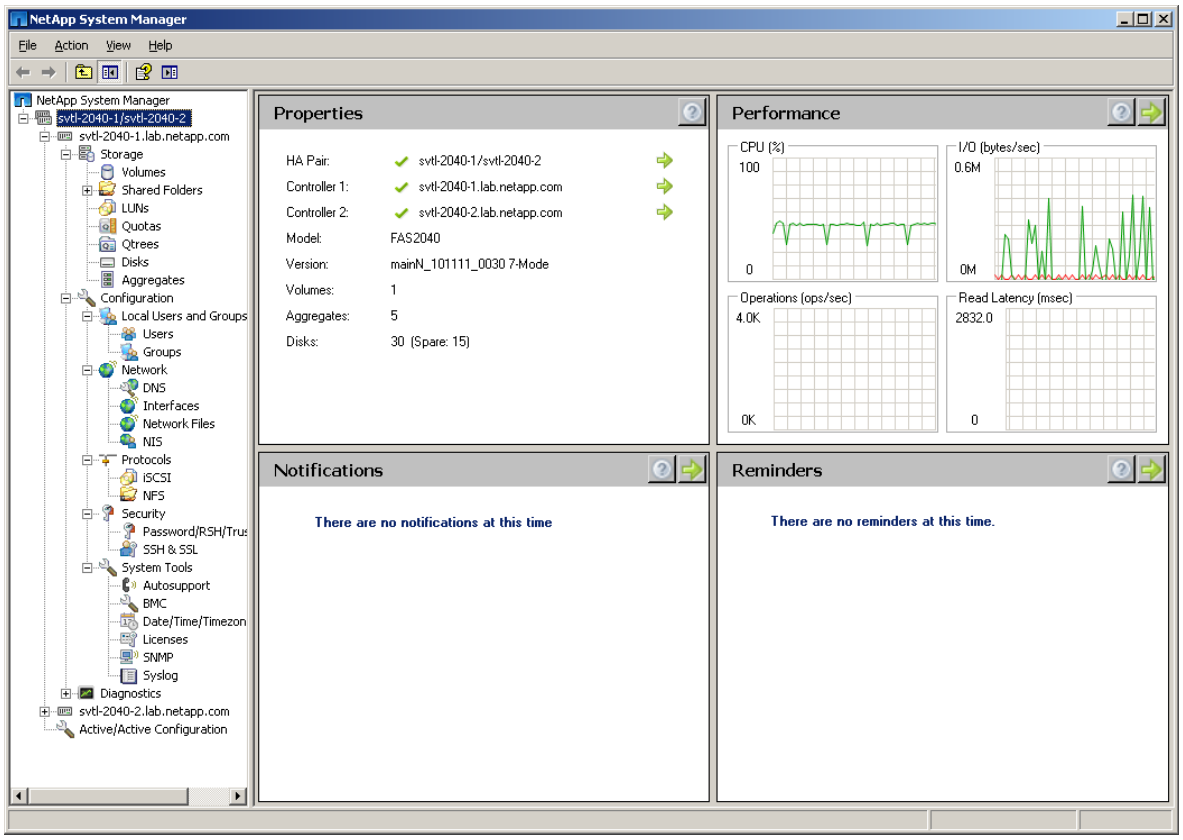
8. Summary

While developing this system, there were two application systems that we looked at for user interface models. FilerView is a web based management system that manages a specific node for a DataONTAP system. System Manager is a windows based system that allows the management of several nodes from the same interface as well as management of the high availability relationship between the nodes. While these interfaces differ in the tools the presentation of the data, there are some underlying characteristics. Each application provides a hierarchy of data in one or areas of the screen. The System Manager system has a tree like view of functions in the left frame of the window. This view is similar to the way Windows Explorer presents the directory structures on a disk. While the most direct process of mapping this type of data to an iPhone application would be to present a single view for each level of the hierarchy, this would produce a very fragmented application. It is useful to consider the likely use case of the iPhone application. It is not likely that the user is going to use the iPhone to configure every part of the storage system. The much more likely use case is a storage administrator gets a notification at night or on a weekend that a volume is reaching capacity or an aggregate is out of space. The administrator will want to quickly add a disk to the aggregate or increase the size of the volume to resolve the immediate issue until they can get to a computer at a more convenient time to resolve the matter in a more complete fashion. The other possible event that could trigger a notification is a controller takeover. The administrator may want to initiate the giveback operation to bring the node that failed back into service. Given the use cases described above a subset of the features provided by System Manager can be selected to provide the administrator with the tools to quickly resolve these issues and return to their other activities.

In addition to these activities, a well-designed system can be expanded incrementally based on user feedback. The application implemented in this system could be easily expanded to offer any of the other features that are offered by System Manager without compromising the existing functionality. As each function is added it would be necessary to evaluate the layout of the main menu to determine if further partitioning is needed.

9. Conclusion

The translation of applications from the desktop to a mobile device requires an understanding of both the functionality to be provided and the usage patterns of the mobile device [15] [16]. In this case, the goal of the desktop application is to manage the configuration and monitoring of a NetApp storage system from a desktop application. The monitoring component shown in the right frame of Fig. 19 is something that does not translate to the usage model for an iPhone. It is also highly improbable that a storage administrator is going to configure every aspect of the storage system from a mobile device. The long list of capabilities can be trimmed to a few and those can be flattened into a single list of capabilities in the main menu.



HA Pair:	✓ svtl-2040-1/svtl-2040-2
Controller 1:	✓ svtl-2040-1.lab.netapp.com
Controller 2:	✓ svtl-2040-2.lab.netapp.com
Model:	FAS2040
Version:	mainN_101111_0030 7-Mode
Volumes:	1
Aggregates:	5
Disks:	30 (Spare: 15)

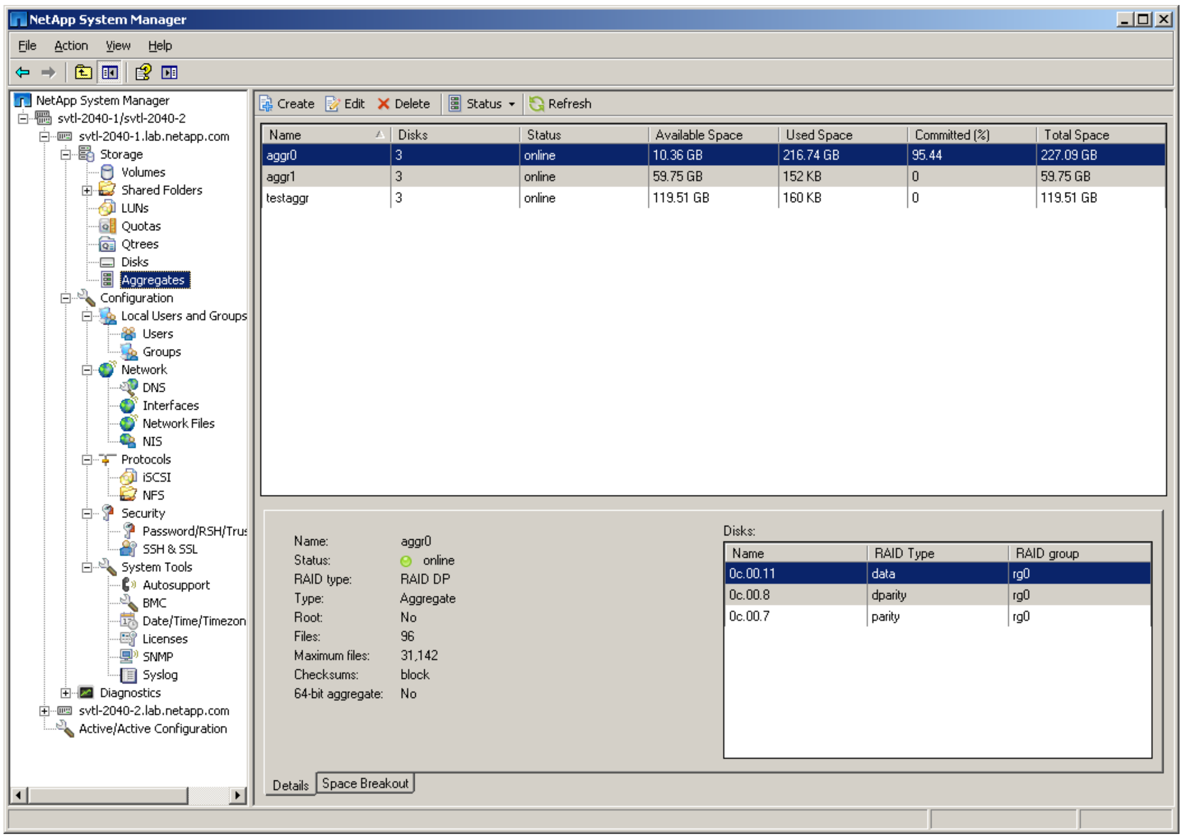
There are no notifications at this time

There are no reminders at this time.

Figure 19. System Manager Monitoring Screen

Early in the development, the iPhone application presented information from both nodes at each level of the hierarchy when lists were presented. Making the system division at the top level as it is in System Manager greatly simplifies the application because there is only one system to talk in each component. This division also improves the response time reducing the number of interactions between the application and the storage system.

When considering the right frame of the application, most of the areas of interest tended to be tables with buttons that provided access to functionality. The conversion from large tables as presented in System Manager to the single column table views used in iPhone applications it is necessary to convert the tables into a series of views. The top level view is a list of the rows of the table with a possible secondary element added to the cell. For an example compare Fig. 20 the Aggregate window from System Manager to Fig. 21 the top level Aggregate view from the iPhone application and Fig. 22 the Aggregate Detail view from the iPhone manager. Another view could be added to show the disk roles used in the aggregate but that level of detail has not been implemented. There is space in the detail view to provide additional information mapped from the System Manager table. This view also shows the button implemented for changing the state of the aggregate. An additional button will appear when the aggregate being viewed is offline that enables destroying the aggregate.



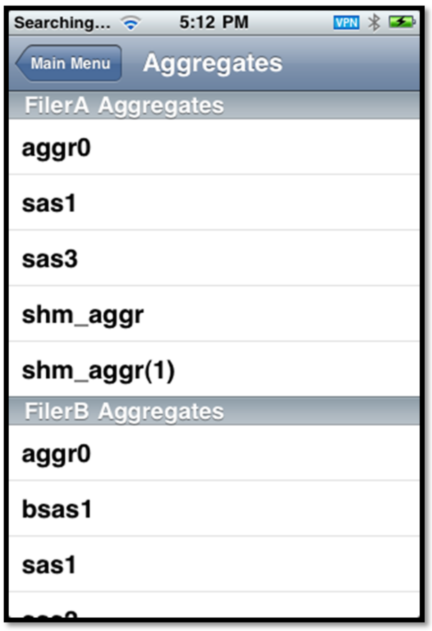
Name	Disks	Status	Available Space	Used Space	Committed (%)	Total Space
aggr0	3	online	10.36 GB	216.74 GB	95.44	227.09 GB
aggr1	3	online	59.75 GB	152 KB	0	59.75 GB
testaggr	3	online	119.51 GB	160 KB	0	119.51 GB

Name:	aggr0
Status:	online
RAID type:	RAID DP
Type:	Aggregate
Root:	No
Files:	96
Maximum files:	31,142
Checksums:	block
64-bit aggregate:	No

Disks:

Name	RAID Type	RAID group
0c.00.11	data	rg0
0c.00.8	dparity	rg0
0c.00.7	parity	rg0

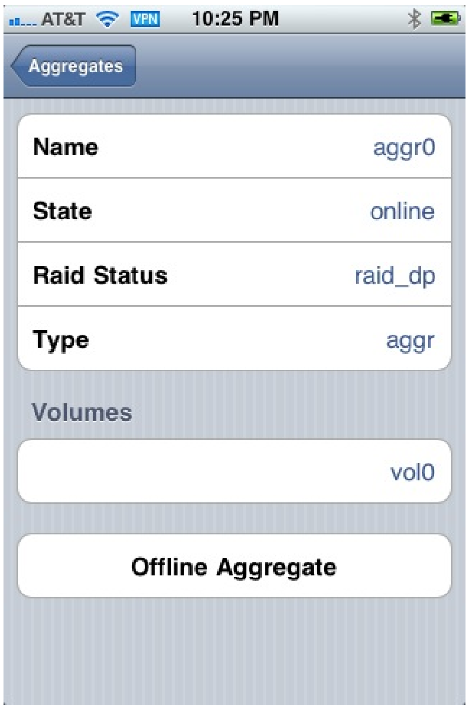
Figure 20. System Manager Aggregate View



FilerA Aggregates
- aggr0
- sas1
- sas3
- shm_aggr
- shm_aggr(1)

FilerB Aggregates
- aggr0
- bsas1
- sas1

Figure 21. iPhone Application Aggregate List



Name	aggr0
State	online
Raid Status	raid_dp
Type	aggr

vol0

Figure 22. iPhone Application Aggregation Detail

These views show the general guidelines used in the conversion of System Manager functionality to the iPhone interface. The guidelines are generally useful for mapping similar applications to iPhone but would not be appropriate for all types of applications. A word processing application, for example, has a very different interface and goal and would not be able to use the same types of translations.

Acknowledgment

The authors would like to thank NetApp, Inc, Francis Tang, David Halliwell and the San Jose State University graduate student Foram Patel for their excellent support of this work of the paper.

References

[1] Sunday, J. RE. iPhone apps. (2008, December 15). Retrieved March 14, 2010, from http://communities.netapp.com/message/5811

[2] About *Table* Views in *iOS* Apps http://developer.apple.com/library/ios/#documentation/windowsviews/conceptual/viewpg_iphoneos/CreatingViews/CreatingViews.html. Retrieved November, 22, 2012.

[3] NetApp, Inc. Management Software – System Manager. Retrieved March 14, 2010, from http://www.netapp.com/us/products/management-software/system-manager.html

[4] NetApp, Inc. Management Software – Operations Manager. Retrieved March 14, 2010, from http://www.netapp.com/us/products/management-software/operations-manager.html

[5] NetApp, Inc. Manage ONTAP SDK Introduction and Download Information. Retrieved March 14, 2010, from http://communities.netapp.com/docs/DOC-1110

[6] NetApp, Inc. Objective-C ManageONTAP SDK. Retrieved March 14, 2010, from http://communities.netapp.com/docs/DOC-1984

[7] View Programming Guide for *iOS: About Windows and Views* http://developer.apple.com/library/ios/#documentation/windowsviews/conceptual/viewpg_iphoneos/Introduction/Introduction.html. Retrieved November, 22, 2012.

[8] Lam, S.C.K. A smartphone-centric platform for personal health monitoring using wireless wearable biosensors. *7th International Conference on Information, Communications and Signal Processing, (ICICS 2009)*, Macau, 8-10 December 2009. http://dx.doi.org/10.1109/ICICS.2009.5397628

[9] Zhang, L., Liu, Y., & Guo, W. Research on Diversified Designing Methods and User Evaluation of Smartphone Interface. *International Symposium on Computational Intelligence and Design (ISCID 2010)*, Hangzhou, China, 29-31 October 2010. http://dx.doi.org/10.1109/ISCID.2010.89

[10] Yun, M., Lee, J., & Kim, S., Downloadable User Interface for Mobile Devices. *Fourth International Conference* on *Networked Computing and Advanced Information Management, (NCM 2008)*, Gyeongju (South Korea), 2-4 September 2008. http://dx.doi.org/10.1109/NCM.2008.32

[11] Conway, J., Hillegass, A.; *IOS Programming, The Big Nerd Ranch Guide*, Big Nurd Ranch, Inc., 2nd Edition, 2011.

[12] Apple iPhone Features OS X http://web.archive.org/web/20080111051348/http://www.apple.com/iphone/features/index.html#macrox. Retrieved June 15, 2010.

[13] About iOS App Programming http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/Introduction/Introduction.html. Retrieved November, 22, 2012

[14] Liu, Q., Wang, G., & Wu, J., Efficient Sharing of Secure Cloud Storage Services. *10th IEEE International Conference* on *Computer and Information Technology (CIT 2010).* Bradford (UK), June 29 - July 1, 2010. http://dx.doi.org/10.1109/CIT.2010.171

[15] Teng, C., Mobile Application Development: Essential New Directions for IT *Seventh International Conference Information Technology: New Generations (ITNG 2010)*, April 12-14, 2010. Las Vegas (USA). Pp. 471-475. http://dx.doi.org/10.1109/ITNG.2010.249

[16] Yu, W. D., Le, K., Towards a Secure Software Development Lifecycle with SQUARE+R. *IEEE 36th International Conference on Computer Software and Applications Workshops (COMPSACW 2012)*, Izmir, Turkey, 16-20 July 2012. Pp. 565 – 570. http://dx.doi.org/10.1109/COMPSACW.2012.104