

**Western Kentucky University**  
**TopSCHOLAR®**

---

Masters Theses & Specialist Projects

Graduate School

---


12-1-2013

# A Problem Solving Approach to Enterprise FileVault 2 Management and Integration

Nicholas Cobb

Western Kentucky University, [nicholas.cobb328@topper.wku.edu](mailto:nicholas.cobb328@topper.wku.edu)

Follow this and additional works at: <http://digitalcommons.wku.edu/theses>

 Part of the [Business Administration, Management, and Operations Commons](#), [Computer Engineering Commons](#), and the [Technology and Innovation Commons](#)

---

## Recommended Citation

Cobb, Nicholas, "A Problem Solving Approach to Enterprise FileVault 2 Management and Integration" (2013). *Masters Theses & Specialist Projects*. Paper 1296.

<http://digitalcommons.wku.edu/theses/1296>

This Thesis is brought to you for free and open access by TopSCHOLAR®. It has been accepted for inclusion in Masters Theses & Specialist Projects by an authorized administrator of TopSCHOLAR®. For more information, please contact [connie.foster@wku.edu](mailto:connie.foster@wku.edu).



A PROBLEM SOLVING APPROACH TO ENTERPRISE FILEVAULT 2  
MANAGEMENT AND INTEGRATION

A Thesis  
Presented to  
The Faculty of the Engineering Technology Management Graduate Program  
Western Kentucky University  
Bowling Green, Kentucky

In Partial Fulfillment  
Of the Requirements for the Degree  
Master of Science in Engineering Technology Management

By  
Nicholas Cobb

December 2013

A PROBLEM SOLVING APPROACH TO ENTERPRISE FILEVAULT 2  
MANAGEMENT AND INTEGRATION

Date Recommended 11/1/2013

Mark Doggett  
Dr. Mark Doggett, Director of Thesis

James Kanan  
Dr. James Kanan

Daniel Jackson  
Dr. Daniel Jackson

Carl A. Fero 11-11-13  
Dean, Graduate Studies and Research Date

## ACKNOWLEDGEMENTS

Dedicated to my supportive family, friends, and loved ones. Special recognition for elements of this paper to Patrick Gallagher, Kevin Gautreaux, Matthew House, Joshua Levitsky, Christopher Silvertooth, Rich Trouton, Ronnie Turner, and Josh Vanderpool as well as the members of the vibrant and intellectual MacEnterprise support community. I would also like to thank the members of my graduate committee, Dr. Mark Doggett, Dr. James Kanan, and Dr. Daniel Jackson for their guidance in pursuit of furthering my education.

## CONTENTS

Chapter 1 – Introduction .....	1
Statement of the Research Problem .....	2
Significance of the Research.....	2
Hypothesis.....	4
Limitations .....	5
Delimitations.....	5
Assumptions.....	6
Chapter 2 – Review of Literature.....	7
Disk Management Background.....	7
Consumer-Based Logic.....	8
Industry and Community Solutions .....	9
Facing Integration Challenges .....	11
Issues with Existing Workflows .....	12
EscrowToEPO: Bridging the Gaps .....	14
Chapter 3 – Research Methodology.....	16
Research Design.....	16
Participants and Data Sets.....	16
Data Collection .....	16
Instruments.....	17
Procedures.....	17
Data Analysis .....	18

Chapter 4 – Data Collection.....	20
Preparing for Development.....	20
Proof of Concept Development .....	21
Results of the Concept .....	24
Application Development .....	25
Results of Application Development and Testing .....	33
Enterprise Adoption .....	40
Chapter 5 – Conclusion.....	43
Industry Implications .....	44
Development Implications and Future Maintenance .....	45
Project Summary.....	48
References.....	50
Glossary .....	56

## LIST OF FIGURES

Figure 1. Waterfall model of development .....	20
Figure 2. Confirmed Proof of Concept Initial Synchronization TS-TEST-AIR.....	22
Figure 3. Initial Text Contents of Proof of Concept Script.....	23
Figure 4. Reading the Recovery Key, Setting as Custom Property, and Sync .....	24
Figure 5. Proof of Concept Custom Property Recovery Validation for TS-TEST-AIR...	25
Figure 6. EscrowToEPO Interface Elements .....	27
Figure 7. Initial Synchronization and Log File Code Syntax .....	28
Figure 8. Enable FileVault 2 and Display the Recovery Key.....	29
Figure 9. The encSuccess method in code (Part 1).....	30
Figure 10. The encSuccess method in code (Part 2).....	31
Figure 11. Disk Status-Checking Script Initialization Code.....	32
Figure 12. Single line code changes example; OS version check.....	32
Figure 13. EscrowToEPO launchdaemon plist file syntax .....	33
Figure 14. Updated interface view after initial synchronization.....	35
Figure 15. McAfee EPO validation of initial synchronization for TS-TEST-AIR.....	36
Figure 16. McAfee EPO validation of initial synchronization for TS-TEST-AIR2.....	36
Figure 17. EscrowToEPO post-encryption interface view for TS-TEST-AIR.....	37
Figure 18. EscrowToEPO post-encryption interface view for TS-TEST-AIR2.....	38
Figure 19. McAfee EPO post-encryption report for TS-TEST-AIR .....	39
Figure 20. McAfee EPO post-encryption report for TS-TEST-AIR2 .....	39
Figure 21. EscrowToEPO success rate of adoption in the large shipping enterprise .....	41
Figure 22. Three scenarios in status script initialization.....	46



LIST OF TABLES

Table 1. Encryption status of Apple laptops in the large shipping enterprise .....42

A PROBLEM SOLVING APPROACH TO ENTERPRISE FILEVAULT 2  
MANAGEMENT AND INTEGRATION

Nicholas Cobb

December 2013

58 Pages

Directed by: Dr. Mark Doggett, Dr. James Kanan, and Dr. Daniel Jackson

Department of Engineering Technology Management      Western Kentucky University

Consumer technology adoption into large enterprise environments is occurring at an unprecedented rate. Employees require the flexibility and efficiency of using operating systems, computers, and mobility products they are familiar with and that enable their productivity. Due to this industry phenomenon, one large shipping enterprise must work to create solutions to integrate Apple's OS X operating system into its traditional Windows-based operating environment. This level of integration must take place carefully to enable usability and foster the continued data security of enterprise assets.

This paper describes the steps and methodology taken, as well as the rationale used, to accomplish the task of integrating Apple's FileVault 2 full disk encryption technology into existing McAfee management infrastructure and traditional deployment and support workflows. Using a combination of industry and community solutions and techniques, a low-cost software solution named EscrowToEPO is created to facilitate the secure and user-friendly adoption of FileVault 2 as a full disk encryption solution. This paper also includes the success/failure rate of adoption and implications as to how the adoption of similar solutions can occur to support future operating systems or other environments.

## **Chapter 1 - Introduction**

Information technology is a complex, ever-changing field that often perpetuates its own development. Once software creation takes place, testing occurs, a version releases, users become acquainted, and the product life cycles continue. Feature recommendations become acceptable suggestions, improving both content and functionality, ultimately causing the release of new products. The dynamic need of users to utilize a variety of evolving technology applications requires information technology professionals' full acclimation to constantly shifting support paradigms and platforms. For the large enterprise environment of a for-profit shipping company operating in two hundred twenty countries worldwide, an adapted learning curve must be especially acute in order to create and manage solutions to adapt deployment and support of changing technology advancements into existing resources for client management and usage.

Based on prior needs of the large shipping enterprise, client management operations currently take place to facilitate the administration of Windows-based PC clients through security applications for more than one hundred fifty thousand systems. These applications include, but are not limited to, group policy enforcement, software delivery and license management, and security adaptations and appliances such as McAfee's full disk encryption. Given the previous statement, it is important to note the company chooses to rely primarily on the Windows operating system platform for the large majority of corporate technology needs. As such, McAfee solutions adopted for the management of client machines primarily focus on Windows-based management only, and as client management needs arise, must undergo modification to fit other platforms

even in cases where McAfee offers little to no support for non-Windows platforms. This particular enterprise, like many other companies, is able to maintain a platform-specific management infrastructure for its business needs, primarily because those needs fall specifically on a single platform for 95% of its endpoints. Due to emerging industry trends this company, as well as many other companies, is working to adopt support for increased usage of mobile technologies and to embrace the allowance of computing solutions not currently part of the approved technology standards. Existing systems must integrate with custom solutions to facilitate client management of the expanding platforms due to factors such as a younger generation of information technology employees, a changing emphasis on preferred technology, and the need to meet business obligations of external clients.

### **Statement of the Research Problem**

The purpose of this project was to implement a software application solution to integrate Apple full disk encryption technology native to the OS X operating system into traditional McAfee resources and management methodologies used in the large enterprise. The specific objectives of the project have shown (1) existing management and security infrastructure utilization was sufficient to manage and deploy native FileVault 2 full disk encryption technology in OS X Mountain Lion for satisfactory, low-impact machine performance, and (2) maximized enterprise cost savings through utilization of this solution.

### **Significance of the Research**

The changing technology context of corporate standards described above fits firmly into the context of IT industry trends. Over time, trends of the consumer market

have slowly shifted enterprise support paradigms to include support for both consumer-grade and privately owned devices in use for business purposes (Neihaves, Koffer & Ortbach, 2013, p. 39). Specifically, many administrators refer to the adoption of Apple devices and operating platforms into the enterprise as a “consumerization of IT” (Stagliano, DiPaolo & Coonnelly, 2013, p.1). In addition, Apple, as well as other mobile computing companies, makes products specifically geared at consumers, not enterprise businesses. According to Moore, the last decade has seen a reverse phenomenon in the adoption of new technologies and innovations. Typically led by the business industry, technology innovations and adoptions are occurring at the consumer level first, causing businesses to lag in adoption rate of these platforms (Moore, 2011, p.2). For this example, the adoption of Apple’s devices and operating system into the enterprise is occurring due to the needs from the bottom of the employment chain, as opposed to the typical strategic planning, integration, and innovation from the top, as many standard corporate technology adoptions occur (Moore, 2011, p.2). In addition, Apple laptops in the large shipping enterprise currently exist outside of the approved standard computing list; however, due to their relatively small number, these devices continue to be allowed as exceptions to the standards. Due to this phenomenon, enterprise administrators and integration teams must examine the differing behaviors of the devices and the implications to security and determine whether existing contracts with suppliers and vendors will enable integration and support of the Apple operating system or not (Mahesh & Hooter, 2013, p.3).

With integration efforts underway, it becomes important for the previously mentioned enterprise professionals to consider the work others have done to accomplish

the same task. In some cases, integration with aforementioned security infrastructure such as McAfee has taken place to various degrees, both in the corporate enterprise and educational environments. Integration projects from various Apple Certified Technical Coordinators of the widespread Apple support community primarily utilize the communication abilities of McAfee's EPO agent to facilitate policy enforcement on Mac OS X clients. Patrick Gallagher is currently Deputy Technical Lead at Emory University. Patrick, as well as various other administrators, has worked to create installation scripts within a package that facilitates communication or machine information from Mac clients back to Emory's enterprise McAfee infrastructure using custom properties at the time of install (Gallagher, 2013). Other community projects work to utilize existing Active Directory infrastructure to keep track of and store machine-specific information. In one case, Christopher Silvertooth, stores a FileVault 2 encryption key within his enterprise's Active Directory infrastructure for each Apple machine object bound to the domain (Silvertooth, 2012). Rich Trouton is currently Lead Help Desk technician at Howard Hughes Medical Institute and holds Apple's Certified Technical Coordinator certifications for the last five versions of OS X (10.4-10.8). Rich has created a disk status-checking script that determines and reports the encryption status of a disk remotely (Trouton, 2013). Considering these examples, integration work of Apple devices into existing corporate infrastructure is one of ongoing challenge and moderate success. These examples; however, demonstrate the importance of utilizing community resources and investigating various existing options to minimize integration challenges with the accomplishments of others for the same purpose.

## **Hypothesis**

This thesis includes the hypothesis statement that a software solution can be created for the large shipping enterprise to leverage existing McAfee security infrastructure to integrate and manage the native FileVault 2 full disk encryption solution on the Mac OS X platform. Given proof of this hypothesis, other organizations with similar management infrastructure could leverage this solution to integrate management of FileVault 2 into their environment.

### **Limitations**

1. There is no budget for additional client management infrastructure or software.
2. The solution must utilize, communicate and exhibit compatibility with existing systems of client management.
3. The proposed software solution must not interfere with existing security software required for deployment to Apple computers, but can utilize security software where applicable.
4. Due to the limitations of previous Apple operating systems, the solution proposed must fit the desired corporate standard of 10.8.x OS X Mountain Lion.
5. Due to the current low rate of enterprise ownership and lack of fully approved corporate standard for Apple computers, development resources are restricted to one development computer (MacBook Pro) and two test computers (MacBook Airs). No other resources are available for development and testing due to the pending adoption of Apple computing platforms based on available solutions for integration.

### **Delimitations**

1. The project data collection regarding success of software adoption is delimited to a specific set of machines, running 10.8.x OS X Mountain Lion operating system in the large shipping enterprise.
2. The project data collection will be delimited to the management system in which the proposed software solution exhibits communication and compatibility with McAfee EPO infrastructure, data collection, and the reporting resident on this system.

### **Assumptions**

1. Project assumes corporate technology standard will allow for the support of OS X computers once the encryption solution reaches implementation.
2. Project assumes application will function within pre-defined compatibility requirements based on Apple's FileVault 2 product.
3. Project assumes standards-based course of development and testing of final product.
4. Project assumes the solution meets general deployment requirements based on the enterprise standards.
5. Project assumes McAfee will continue to develop and provide support for the ePolicy Orchestrator (EPO) software.



## **Chapter 2 – Review of Literature**

Due to the user experience described above, FileVault 2 is the preferred disk encryption solution for Apple laptops in the large shipping enterprise for many reasons. With native integration into the Mac OS X operating system, Apple allows users the ability to protect their files without the need to worry about managing the security of data. By keeping the emphasis on the simplicity of their computing environment, Apple's encryption solution gives the user the ability to leverage the highest system performance achievable when using a full disk encryption product. Without this level of integration, an encryption product can be obtrusive or even counterproductive to the user, as the pre-boot file system may not directly interact with the single-sign on login mechanism of the Apple computer (McAfee Inc., 2013). For this reason, the large shipping enterprise, already faced with a high cost of client management integration, prefers not to deploy an obtrusive login-based product to users who adopt the Apple platform, especially in the case where external business requirements demand this adoption.

### **Disk Management Background**

The integration of FileVault 2 does not only extend benefits to the user of the Apple computer system. In addition to providing full disk encryption, Apple created a volume manager called Core Storage, which creates the encrypted volume on the disk, and allows for the management of the CoreStorage volume through the existing command line disk utility (diskutil) (Apple Inc, 2012, p.8). Due to the ease at which FileVault 2 can be supported natively on each Apple computer system, as well as the extensive documentation provided by Apple and other community resources, administrators at the large shipping enterprise would highly prefer to integrate FileVault

2 into their encryption compliance strategies as the preferred solution for encrypting Apple computer systems.

### **Consumer-Based Logic**

Despite the desire to settle on FileVault 2 as an encryption solution, challenges of Apple's design logic precluded the adoption of FileVault 2 as an enterprise standard for encryption compliance. With the release of FileVault 2 on OS X Lion, Apple gave the user the ability to encrypt the whole system disk. Previous versions of the operating system only allowed for encryption of the user's home folder, or profile where the user data was stored, not the whole disk. At the time of enabling FileVault 2, users have the opportunity to store their recovery key with Apple or to keep track of it themselves, by writing the key or taking a screenshot. This option is sufficient for consumer-based users to manage but by Apple's own wording, "Corporate regulations may require that all encryption and recovery key storage be maintained inside the corporate controlled infrastructure" (Apple Inc., 2012, p. 8). With this in mind, Apple built a mechanism to manage FileVault 2 recovery keys from an institutional level, which allows organizations to preset the recovery key and deploy it via pre-existing deployment mechanisms. For many companies, this workflow is less than sufficient as it makes every machine with FileVault enabled utilize the same institutional recovery key. To this end, organizations without existing management or integrated deployment systems for the Apple platform are unable to leverage the pre-built FileVault 2 management strategies and any security policies revolving around its enablement as designed by Apple.

Perhaps one of the biggest integration challenges faced by Apple and enterprise administrators is the ability to integrate Apple's consumer-based logic into enterprise-

based systems, specifically related to enabling and managing FileVault 2 encryption. Originally designed for consumers to store a single recovery key in iCloud, Apple's FileVault 2 management logic is not exactly enterprise-friendly. When users store their recovery key in iCloud, it is bound to their (sometimes) personally affiliated Apple ID. In many corporate environments, storing corporate information personally, or on another company's server architecture is a violation of many enterprise security policies (Apple, 2012, p.8).

### **Industry and Community Solutions**

Google Inc. places a strong emphasis on the usage of Apple computers by its employees, primarily for security reasons (Gelles & Waters, 2010). With this emphasis, Google struggled with the ability to integrate FileVault 2 encryption into their enterprise specifically due to the consumer focus of the FileVault 2 product (Google Inc., 2012). With no ability to internally manage FileVault 2 outside of the graphical user interface pane on the Mac, Google created 'Cauliflower Vest (csfde)', a command-line interface to enable and manage FileVault 2 encryption and recovery keys using a Google application server (Google Inc., 2012). This project, created to serve an internal enterprise need at Google, paved the way for a variety of changes in the FileVault 2 management realm.

After Cauliflower Vest, Apple released OS X Mountain Lion in the summer of 2012. Included in OS X Mountain Lion was a mechanism similar to Google's creation, but this time officially supported and integrated into the operating system by Apple. After the release of this command-line utility, called 'fdsetup' the number of community solutions to storing and managing FileVault 2 recovery keys multiplied. For example, Graham Gilbert, Lead Engineer for pebble.it, created a software solution for FileVault 2

recovery key escrow that builds on the functionality of Cauliflower Vest and fdesetup (Gilbert, 2013). Crypt, the project's name, allows administrators to setup a web application server to manage the recovery keys, and a client to enable encryption on each OS X endpoint (Gilbert, 2013).

Despite the motivations of community developers mentioned throughout this paper, software manufacturers have been slow to include Apple's fdesetup utility in their products, and some vendors – including McAfee – chose not to adopt 'fdesetup' at all. One such company is Symantec, which markets the PGP encryption product as a solution for full disk encryption. Symantec's PGP encryption software includes a proprietary pre-boot authentication screen that is not native to the Mac OS X operating system (Symantec, 2012, p. 1). Additionally, single sign on options from this product are only supported for Windows-based computers (Symantec, 2012, p. 1). Meanwhile, McAfee and other vendors continued to develop and support their own workflows. These deployment and configuration workflows traditionally work well for the management of Windows-based endpoints, but do not exhibit reliable performance on Apple hardware and operating systems (McAfee Inc., 2013). Given these industry and community trends from full-disk encryption software vendors, the large enterprise detailed above needed a solution that would integrate into both the McAfee security infrastructure and the OS X operating system, and would not require additional funding to setup and maintain.

However, some client management software manufacturers fully support the management of FileVault 2 on Mac OS X computers. Joshua Levitsky, a senior technical consultant at Absolute Software, authored the FileVault 2 management guide utilized by administrators of Absolute Manage client management software. Absolute Manage,

which runs a software agent on each Mac OS client, allows administrators to deploy packages and scripts to enable FileVault 2 in both the institutional, personal, and hybrid topologies as designed by Apple (Levitsky, 2013, p. 5). The solution from Absolute Manage also allows for compliance reporting from the console through the execution of scripts on the managed Apple computer. Encryption status, recovery information, and the users authorized to login are some of the supported reporting features in the guide (Levitsky, 2013, p. 12). Similarly, JAMF Software's Casper Suite also offers FileVault 2 management. The Casper Suite product also uses a client agent to deploy encryption configurations built inside the JAMF Software Server to managed Apple machines (JAMF Software LLC., 2013, p. 7). Like Absolute Manage, Casper Suite can also deploy personal, institutional, or hybrid FileVault 2 configurations and report on the status of encrypted systems for recovery and compliance (JAMF Software LLC., 2013, p. 15). While both solutions offer the ability to manage FileVault 2 and other aspects of Mac OS X client computers, the large shipping enterprise has neither of these systems present in its environment.

### **Facing Integration Challenges**

For the large shipping enterprise, the challenges of implementing a FileVault 2 solution are evident for a variety of reasons. To start, the enterprise has already absorbed the high cost of ownership of management infrastructure for Windows-based PCs. Due to costs of this implementation, any solution considered for encryption management must specifically integrate with the existing infrastructure.

As another deployment challenge, the main vendors of that infrastructure, Computer Associates and McAfee, do not offer flexible or timely methods of support for

platforms outside of the traditional Windows base (Computer Associates, 2009, p. 12). In particular, products created by these vendors for the Apple platform generally release as a “non-Windows” (Linux) product due to the Unix-base of the Apple OS X operating system. In this manner, McAfee created an installation script to facilitate manual deployment of their EPO software, instead of following the standard package deployment method of the OS X operating system (McAfee Inc., 2011, p.34). This is another challenge to overcome, as system administrators in the large enterprise must create a proprietary EPO package in order to facilitate ease of deployment to user machines. For the large shipping enterprise, this undermines the ease with which EPO can be deployed. With regard to McAfee’s full disk encryption solution, ease of deployment of McAfee’s EPO software is very important, as the encryption software deploys to the machine once the EPO client has been previously installed (McAfee Inc., 2012, p.29).

### **Issues with Existing Workflows**

The process for enabling McAfee’s full disk encryption is complex and inconsistent. As detailed above, in order to enable full disk encryption on the Mac McAfee requires installation of the EPO software to take place. In addition to the lack of deployment package provided by McAfee, the EPO software requires that a manual synchronization take place in order to establish a communicated object on the EPO server. The manual synchronization process also requires command-line interaction, as McAfee provides no user interface for EPO. After that has occurred, an administrator or technician sets an encryption policy for that system on the server. Once the policy has been set, another manual synchronization is required to communicate the newly assigned policy to the machine. After the successful synchronization, the encryption software

deploys to the machine through an array of scripted tasks, and the user receives a prompt to reboot the system. The process for enabling encryption becomes increasingly complex at this point. After a reboot and another successful synchronization, the machine should begin encryption. However, the product does not always activate as advertised. In some cases, the product will reach encryption activation after a number of subsequent successful manual synchronizations; in other cases, the product may never activate (McAfee Inc., 2013, p.1).

In addition to the sometimes-flawed activation process, there are other specific issues with the McAfee encryption solution for Mac OS X that makes it an unfavorable solution. First, the encryption software's authentication management system corrupts itself on a consistent basis. This prevents users from being able to authenticate and use their Apple computer system. While the system lockout exhibited is frustrating for the user, the authentication corruption can cause the need for manual decryption in order to gain access to the operating system or user data. For users, the downtime could be significant depending on the hardware specifications of the computer, as the speed of encryption or decryption is dependent on processor speed, the type of drive, or utilization of the system (McAfee Inc., 2013, p.2). Second, McAfee maintains a segmented approach to the allowance of system updates or new hardware released by Apple. In the case of firmware updates, McAfee sometimes advises not to apply firmware updated without first decrypting the machine (McAfee Inc., 2012, p.1). For support groups in the large shipping enterprise, this presents an unacceptable challenge. Compliance policies in the enterprise require that updates to device operations or security, whether at the operating system or firmware level, occur in a timely fashion. If the large enterprise is to

maintain adherence to compliance requirements, each machine would have to be manually decrypted, system updates applied, and re-encrypted again at the expense of user down time. Due to these issues, the high level of integration of FileVault 2 into the OS X operating system is ideal as is a software solution that can leverage the enterprise infrastructure and manage FileVault 2 encryption.

### **EscrowToEPO: Bridging the Gaps**

Due to the issues mentioned above with the existing disk encryption product, the large enterprise requires a software solution to integrate FileVault 2 into its OS X user base and existing back end McAfee systems. While many of the community solutions listed above would allow the user experience to benefit from FileVault 2, they do not integrate with McAfee systems, as the community solutions alone often exhibit proprietary functionality for each environment they represent. In addition, many open-source projects are prone to negative security reviews in large enterprises due to the ease of availability of the project code (Hewlett Packard Development Company, LP., 2011, p.3). Conversely, the commercial solution provided by McAfee requires the usage of their encryption product, which is unfavorable due to the factors mentioned above.

Examining the challenges presented by the above, it became evident that each community project referenced in this paper carried a problem-solving approach for FileVault 2 management integration in the large shipping enterprise. Creation of a software solution that combined McAfee EPO synchronization, disk encryption status checking, and EPO custom property assignment would provide an answer to this challenge while enabling the best experience for OS X users. A software application, called EscrowToEPO, first enabled encryption for the corporate standard 10.8 systems



using `fdsetup`. Next, the application assigned McAfee EPO custom properties using the FileVault 2 personal recovery key, the machine's serial number and timestamp to identify the synchronization period, and the encryption status of the disk. Using EPO, EscrowToEPO synchronized these variables back to the McAfee management server for storage and compliance reporting on a regular interval. In this manner, EscrowToEPO adoption as a solution provides the large shipping enterprise with several benefits. First, users were able to take advantage of the positive experience and high level of operating system integration that Apple's FileVault 2 solution provides. Additionally, EscrowToEPO allowed the large shipping enterprise to utilize the existing McAfee backend to manage the compliance reporting and storage required of encryption management standards.

## **Chapter 3 - Research Methodology**

### **Research Design**

The research design of this thesis examined quantitative data based on the successful synchronization of the clients that were running the proposed software solution. To represent failures, reporting showed Mac OS X laptop clients with the McAfee EPO software installed, but not reporting encryption compliance data. As the software solution was manually deployed to a list of pre-existing Mac users, the success or failure data of the software solution was validated upon installation. Due to existing compliance requirements, the data collected was considered final once those requirements had been met. The quantitative measure of success versus failure ratio was sufficient to validate the data based on the pre-existing compliance requirements.

### **Participants and Data Sets**

Participants in the overall project were limited to employees using Mac OS X based laptops in the large shipping enterprise described throughout this thesis. Their participation in the project facilitated compliance with enterprise requirements that all laptops enable full-disk encryption. The participants' participation in the project was reflected in non-identifiable data provided by the McAfee EPO server infrastructure and only served the purpose of success/failure auditing. No personal information was used or made available.

### **Data Collection**

The central focus of this project was to determine the success variables of the development and integration of EscrowToEPO. Data collection of integrated systems took place from the McAfee ePolicy Orchestrator server console chosen for compliance

reporting due to its existence as the in-place endpoint security infrastructure managing encrypted client systems.

### **Instruments**

The instruments used for development included TextEdit, Terminal, and Xcode. Instruments used for data collection included McAfee's ePolicy Orchestrator reporting tools, to report on successfully integrated and encrypted systems. Reported data was analyzed and represented using Microsoft Excel.

### **Procedure**

Step 1: A proof of concept script was created that utilized various aspects of several community projects in this paper. For the proof of concept creation to take place, resources from those community projects were combined to represent the theoretical implementation of this solution in script form. After these elements were combined in a script, execution of the script determined the success of the EscrowToEPO solution. Initial design of the proof of concept took place using a text-editing application native to the OS X operating system called TextEdit. The resulting text file was exported as an executable shell script that ran on a test machine to verify proper communication of machine custom properties took place using the problem-solving approach of the combined solutions.

Step 2: After verification that the script successfully communicated with the EPO server, Apple's integrated development environment application, called Xcode, was used to develop EscrowToEPO that facilitated enabling FileVault 2 and communication of encryption recovery data back to the McAfee ePO server infrastructure. EscrowToEPO

was validated based on the successful communication of the FileVault 2 personal recovery key to the McAfee management server.

Step 3: After successful validation took place, EscrowToEPO was put up for adoption in the large shipping enterprise by deploying manually to a pre-determined list of OS X 10.8 computers in need of FileVault 2 encryption.

### **Data Analysis**

Step 1: A working proof of concept script as detailed above used existing community projects. The script's effectiveness was judged on a success/failure scale in a test environment. Success was measured based on whether or not the script successfully established and completed transferred communication of Apple machine custom properties, including an encryption recovery key and timestamp, with the McAfee EPO server infrastructure.

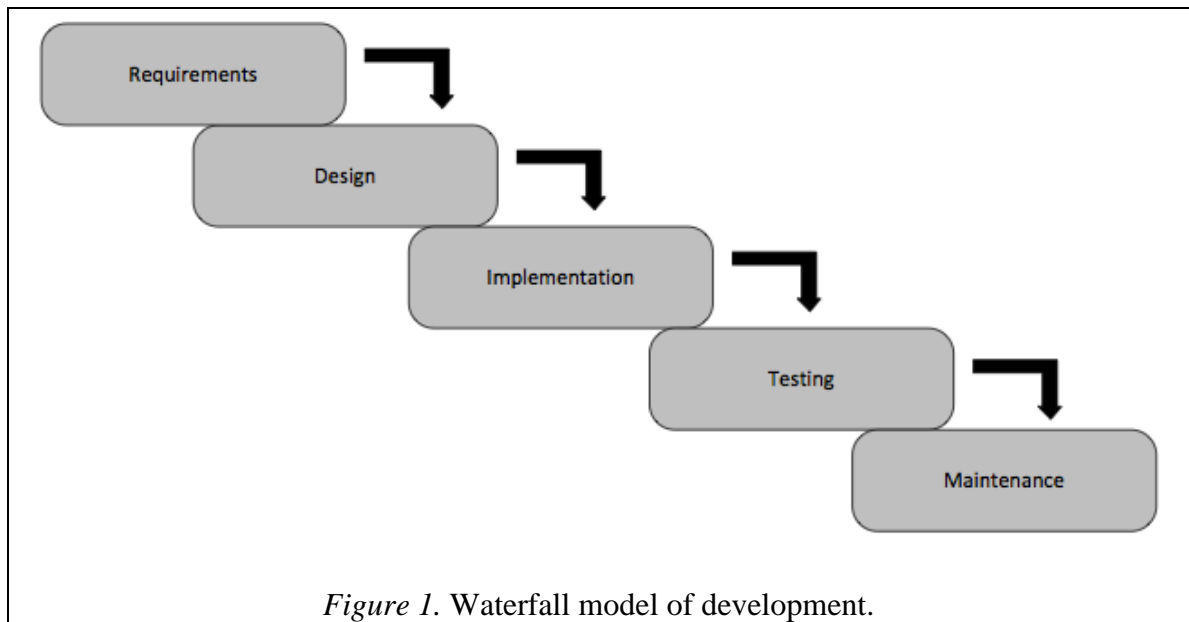
Step 2: After the proof of concept script was verified as successful via the data analysis described above, the software solution was developed and implemented in both test and production environments. The software solution included a system-level monitoring protocol to provide compliance data indicating the status of FileVault 2 encryption on the user's system drive, facilitated by a launchdaemon. Success of the software solution in test was measured on whether or not the application successfully established and completed transferred communication of Apple machine custom properties, including an encryption recovery key and timestamp, with the McAfee EPO server infrastructure. The test systems did not show any signs of failure and were always communicating the encryption and recovery variables except when information security staff applied an update to the EPO server infrastructure.

Step 3: With successful validation of the software solution within the test environment, the software was manually deployed to production user systems throughout the enterprise. As software was deployed, user systems were validated at the time of installation and encryption being enabled. If a system showed signs of failure or was not communicating, the variables causing failure were noted, however once the application was manually deployed in production, no failures were encountered.

## Chapter 4 – Data Collection

### Preparing for Development

To begin the standards-based development of the EscrowToEPO application solution, an application development process, or software development life cycle was selected to adhere to the development standards required of the project. EscrowToEPO development took place under the guidance and in adherence to general principles of the waterfall model of software life cycle development. The waterfall model for application development focuses on a sequential development process that is utilized to allow software development to flow through the steps of establishing and completing Requirements, Design, Implementation, Verification, and Maintenance (Huo, Verner, Zhu & Babar, 2004, p.2). Figure 1 displays a representation of the waterfall model.



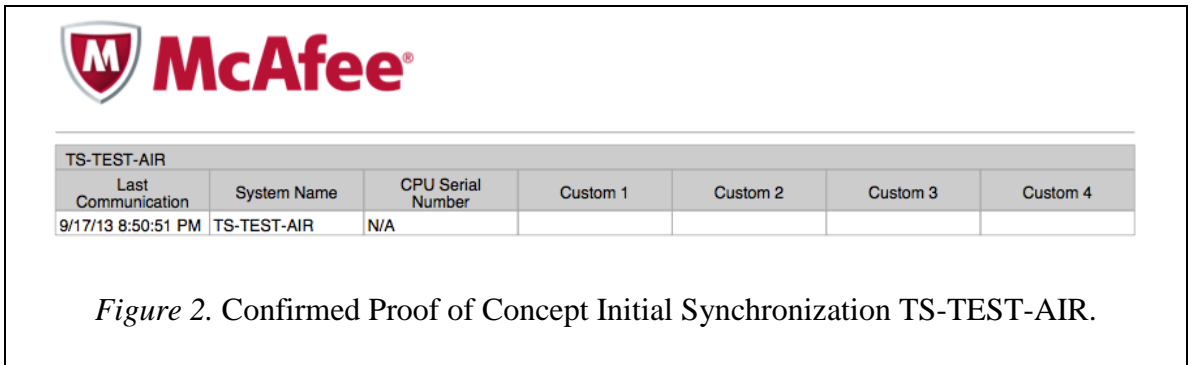
Adhering to the waterfall software life cycle, the EscrowToEPO application development requirements dictated that facilitation of encryption management must utilize an existing enterprise system and should not interfere with existing security

software required to run on the Mac OS X platform. Given these requirements, the design phase of the waterfall cycle worked to base the proof of concept and the EscrowToEPO application solution on the existing required installation of the McAfee EPO software. During the implementation phase, the proof of concept and EscrowToEPO application was developed to adhere to the basic requirements and design aspects set forth in the previous stages. In this stage, the code for the script and the application was completed and the interface was designed, in two separate phases. Once completed, the script and application entered individual testing phases in sequential order beginning with the proof of concept script, based on the design of the overall study. Here, the interaction with the EPO server was validated against the hypothesis that the existing McAfee infrastructure could be utilized to manage FileVault 2 encryption. Lastly, the waterfall method included a maintenance phase that dictated how the EscrowToEPO project should continue once it was ready to deploy.

### **Proof of Concept Development**

In order to complete the testing of the proof of concept script, a MacBook Air from the test pool was preconfigured with Apple's 10.8 OS X Operating System, the McAfee EPO software, and has a computer name set to "TS-TEST-AIR". Due to EscrowToEPO's dependence on the McAfee EPO software, an initial synchronization of the test MacBook Air created a managed computer object within the EPO server environment. The EPO server established the managed computer object using the computer name from above, "TS-TEST-AIR". The process of initial synchronization confirmed successful EPO communication between the client and server as well as validated that testing of the proof of concept script could be initiated. The initial

synchronization is shown as confirmed by the date/time entry in the “Last Communication” field in Figure 2.



After synchronization the development process began with the creation of the proof of concept script, which combined several of the community solutions documented above. Figure 3 displays the script’s original syntax and initial text contents described here as used to enable encryption.



```
#!/bin/bash

# Ensure that script is now running as root.
RunAsRoot()
{
    if [ ! $( id -u ) -eq 0 ]; then
        echo
        echo "*This application must be run as root. Re-running with admin privileges.*"
        exec sudo su root -c "bash \"$workingDirectory/$scriptName\"" # Call this prog as root
        exit $? # since we're 'execing' above, we wont reach this exit
        # unless something goes wrong.
    fi
}

RunAsRoot

# Create the directory for storing the RecoveryKey.plist
echo "Creating directory for Recovery Key plist file..."
if [ ! -d /usr/local/key ]; then
    mkdir -p /usr/local/key
fi
echo "plist Directory created successfully."

# Prompt for Username
echo "Enter the username that is used for login on this computer: "
read -s $currentUser

# Prompt for Password
echo "Enter the password for the user $currentUser: "
read -s Password

if [ -z "$Password" ]; then
    echo -e "\nYou must provide a password."
    exit 1
fi

# Enable FileVault 2 for the User that is interacting
fdesetup enable -user $currentUser -password $Password -outputplist > /usr/local/key/RecoveryKey.plist
```

Figure 3. Initial Text Contents of Proof of Concept Script.

The first task of the script was to establish the directory to store the recovery key plist file. This was performed using the /usr/local/ directory, as it is a hidden directory for the standard OS X operating system configuration. Referencing Christopher Silvertooth's FileVault 2 Active Directory management project, the script's interaction began by prompting the user for an administrative, or root level, password to obtain access to make changes to the system (Silvertooth, 2012). Next, the user received sequential prompts to enter the username and password of a local administrator account (Silvertooth, 2012). After entry, the username and password data were stored as variables within the script for

use during the process to enable encryption. The script then executed the 'fdsetup' command included in version 10.8 of Apple's OS X operating system to manage FileVault 2 (Apple, 2012). During this execution, an additional argument, called 'outputplist', told the system to export the recovery information into a plist file for temporary storage on the system (Apple, 2012).

Once the script successfully enabled encryption and stored the recovery data within the plist file, the script then attempted to read the recovery key data from the plist file and additionally stored it as a variable. The script syntax utilized for performing this task and additional commands used to set the recovery key string as an EPO custom property for sync back to the server were similar to the processes utilized by Patrick Gallagher's custom EPO installer scripts (Gallagher, 2013). After the FileVault 2 personal recovery key was set as a custom EPO property, the script then executed the command to synchronize the computer back to its managed object on the EPO server. This portion of the proof of concept is represented in Figure 4.

```
# Read the recovery key
echo "Reading Recovery Key"
recoveryKey=`defaults read /usr/local/key/RecoveryKey.plist RecoveryKey`
echo $recoveryKey

# Set ePO custom property for Recovery Key
/Library/McAfee/cma/bin/msaconfig -CustomProps1 "$recoveryKey"

# Perform a sync
/Library/McAfee/cma/bin/cmdagent -P

exit 0
```

*Figure 4. Reading the Recovery Key, Setting as Custom Property, and Sync.*

## Results of the Concept

Figure 5 notes the change in synchronization time, as well as the updated “Custom 1” field with a FileVault 2 personal recovery key, for the TS-TEST-AIR managed object.



---

Criteria: System Name Equals \*TS-TEST-AIR\*

TS-TEST-AIR					
Last Communication	System Name	Custom 1	Custom 2	Custom 3	Custom 4
9/20/13 10:13:13 PM	TS-TEST-AIR	ZARO-5UWO-H4U6-HB28-LFH9-E7JQ			

*Figure 5. Proof of Concept Custom Property Recovery Validation for TS-TEST-AIR.*

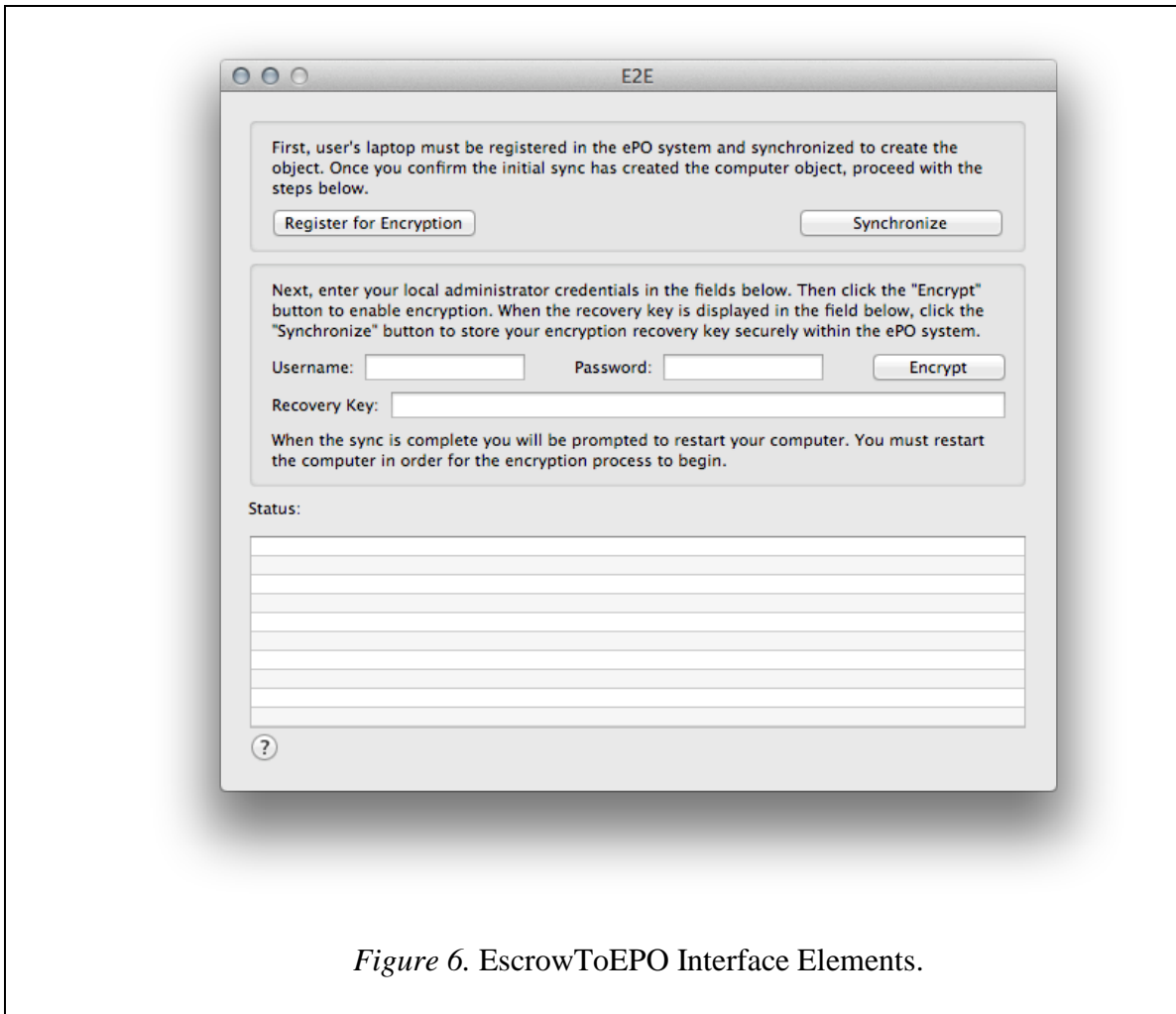
Both the updated synchronization time and presence of the FileVault 2 personal recovery key in the “Custom 1” field indicated a successful synchronization took place at the time the proof of concept script was run on the TS-TEST-AIR test computer. With the proof of concept script validated as successful, the development of the EscrowToEPO application was initiated.

### **Application Development**

In order to build the EscrowToEPO application and represent the workflow and logic of the encryption process, interface items were required to accurately represent the functionality, logic, and workflow of the process validated by the proof of concept script. The required functionality consisted of the ability to perform an initial synchronization with the EPO server in order to create the managed computer object. As a primary task the application stored the username and password as temporary variables, enabled FileVault 2 encryption, and secured storage and synchronization of recovery information

with the EPO server. Additionally, the application also reported the status of the disk encryption and set that as a custom property while maintaining the reporting needs of the large shipping enterprise.

The interface consisted of simple fields to enter the username and password, an encrypt button that initiated the enabling 'fdsetup' command and exported the recovery information, a synchronize button, and a log file view to be able to troubleshoot the synchronization task should communication issues arise. Several other interface elements were created to assist the user with enabling encryption and managing the intended workflow of the large shipping enterprise. A help button that pointed to the company helpdesk page, as well as some instructional dialog text was also present to assist the user with completing the encryption process. Lastly, a registration button provided the user with the intended workflow for enabling FileVault 2 encryption in the large shipping enterprise. Figure 6 shows the interface elements contained within the EscrowToEPO application interface.



*Figure 6. EscrowToEPO Interface Elements.*

For the application development and testing, code was written using Xcode and the Applescript-Objective-C programming language. In this development phase, both MacBook Airs were used for the test runs of the application workflow and status checking as the application was developed. For the testing of the application, the initial tasks of EscrowToEPO included the preliminary synchronization and refresh of the log file into the log view area after each machine was registered in the encryption recovery system. In the large shipping enterprise, the initial synchronization took place manually after the user pressed the registration button and registered their machine at the

corporation's encryption recovery website. In the same manner as the proof of concept script, the initial synchronization established the managed computer object in the EPO server. Figure 7 below shows the code required to establish a successful synchronization from the client to the EPO server as well as the code involved in loading the log file into the EscrowToEPO user interface log view. Both of these functions are grouped to provide the user with multiple functionality and efficiency, and are connected to the "Synchronize" interface button shown previously in Figure 5.

```

on syncButton_(sender)
    log "User selected Synchronize"
    spinnerBro's startAnimation_(spinnerBro)
    theWindow's displayIfNeeded()
    fvStatus's setTextColor_(current application's NSColor's blackColor)
    fvStatus's setStringValue_("Initiating sync with the server...")
    try
        tell current application
            do shell script "/Library/McAfee/cma/bin/cmdagent -P" with administrator privileges
        end tell
        current application's NSThread's sleepForTimeInterval_(3)
        try
            -- Load ePO Log File
            log "Collect and Send - Loading ePO Log"
            set unixpath to "/Library/McAfee/cma/scratch/etc/log"
            set UTF8StringEncoding to current application's NSUTF8StringEncoding
            set {txt, theError} to current application's NSString's
                stringWithContentsOfFile_encoding_error_(unixpath, UTF8StringEncoding, reference)
            if txt = missing value then
                log theError
                spinnerBro's stopAnimation_(spinnerBro)
            else
                set newlineCharacterSet to current application's NSCharacterSet's newlineCharacterSet
                set textParagraphs to txt's componentsSeparatedByString_("\n")
                set my epoLog to textParagraphs
                epoAC's rearrangeObjects()
                epoTable's scrollRowToVisible_(epoAC's arrangedObjects()'s |count|() - 1)
                theWindow's displayIfNeeded()
                fvStatus's setTextColor_(current application's NSColor's blackColor)
                fvStatus's setStringValue_("Synchronize task complete!")
            end if
        end try
        spinnerBro's stopAnimation_(spinnerBro)
        on error
            log "Collect and Send - Error Occurred"
            tell current application's NSAlert to set theAlert to alloc()'s init()
            tell theAlert
                setMessageText_("EscrowtoEPO Encountered An Error")
                setInformativeText_("EscrowtoEPO was unable to perform the task requested. Refer to the
                    logs in Console for troubleshooting, or contact your system administrator.")
                addButtonWithTitle_("OK")
                setAlertStyle_(2)
                set theResult to runModal()
            end tell
            spinnerBro's stopAnimation_(spinnerBro)
        end try
    end syncButton_

```

Figure 7. Initial Synchronization and Log File Code Syntax.

After the initial synchronization, the user enabled encryption by entering the administrator username and password in the interface's text fields and clicking the "Encrypt" button. When the user clicked the "Encrypt" button, the code shown in Figure 8 enabled FileVault 2 disk encryption, output the recovery plist file to a hidden directory as a hidden file, read the recovery key and displayed it in the interface, and finally alerted the user that a restart is necessary to finalize the process.

```

try
  log "Encryption being enabled for user -- " & (stringValue() of userEntry)
  -- Do shell script to enable encryption
  do shell script "/usr/bin/fdesetup enable -user " & (stringValue() of userEntry) & "
    -password " & (stringValue() of passEntry) & " -outputplist > " & plistDir with
    administrator privileges
  current application's NSThread's sleepForTimeInterval_(1)
  theWindow's displayIfNeeded()
  my fvStatus's setTextColor_(current application's NSColor's blackColor)
  my fvStatus's setStringValue_("Getting recovery key...")
  current application's NSThread's sleepForTimeInterval_(2)
  -- Task to read the Recovery Key
  tell current application's NSPipe to set outPipe to pipe()
  set outFileHandle to outPipe's fileHandleForReading()
  tell current application's NSTask to set theTask to alloc()'s init()
  tell theTask
    setLaunchPath_("/usr/bin/defaults")
    setArguments_({"read", plistDir, "RecoveryKey"})
    setStandardOutput_(outPipe)
    |launch|()
  end tell
  tell outFileHandle to set theData to readDataToEndOfFile()
  set theResult to current application's NSString's alloc()'s initWithData_encoding_(
    theData, current application's NSUTF8StringEncoding)
  set theKey to theResult as text
  if theTask's terminationStatus() as integer is not 0 then set theResult to "There was
    an error"
  my recoveryKey's setStringValue_(theKey)
  if theResult = "There was an error" then
    theWindow's displayIfNeeded()
    my fvStatus's setTextColor_(current application's NSColor's redColor)
    my fvStatus's setStringValue_("There was an error enabling encryption!")
  else
    theWindow's displayIfNeeded()
    my fvStatus's setTextColor_(current application's NSColor's blackColor)
    my fvStatus's setStringValue_("Recovery Key active, initiating the sync
      process...")
    -- INSERT THE COMMAND FOR TASKS HANDLER TO SET CUSTOM PROP/SYNC/DELETE
    encSuccess_(me)
    theWindow's displayIfNeeded()
    my fvStatus's setTextColor_(current application's NSColor's blackColor)
    my fvStatus's setStringValue_("Encryption has now been enabled! Please restart
      the computer.")
    spinnerBro's stopAnimation_(spinnerBro)
    tell current application's NSAlert to set theAlert to alloc()'s init()
    tell theAlert
      setMessageText_("EscrowtoEPO Requires A Restart!")
      setInformativeText_("Please restart the computer to enable encryption.")

```

Figure 8. Enable FileVault 2 and Display the Recovery Key.

In addition to enabling FileVault 2 and prompting the user for a restart, the code in Figure 8 also checked that encryption was enabled successfully and initiated the reading and custom property setting of the recovery key and synchronization of the recovery information back to the McAfee EPO server by calling the “encSuccess” method in code. The encSuccess method also contains code to notify and update the user of the progression of tasks through the encryption and synchronization process. Figures 9 and 10 show the encSuccess method’s code to enable these tasks.

```

-- Error-checked Processes to Set Recovery Key as Custom Property, Sync, Remove the Recovery plist
file
on encSuccess_(sender)
  theWindow's displayIfNeeded()
  my fvStatus's setTextColor_(current application's NSColor's blackColor)
  my fvStatus's setStringValue_("Preparing data for sync...")
  current application's NSThread's sleepForTimeInterval_(1)
  try
    -- Set the FV2 Recovery Key as an EPO custom property
    tell current application's NSPipe to set outPipe to pipe()
    set outFileHandle to outPipe's fileHandleForReading()
    tell current application's NSTask to set theTask to alloc()'s init()
    tell theTask
      setLaunchPath_("/Library/McAfee/cma/bin/msaconfig")
      setArguments_({"-CustomProps1", theKey})
      setStandardOutput_(outPipe)
      |launch|()
    end tell
    tell outFileHandle to set theData to readDataToEndOfFile()
    set taskResult to current application's NSString's alloc()'s initWithData_encoding_(theData,
      current application's NSUTF8StringEncoding)
    if theTask's terminationStatus() as integer is not 0 then set taskResult to "There was an
      error"
    if taskResult = "There was an error" then
      do shell script "cp -R " & plistDir & " " & secureDir with administrator privileges
      theWindow's displayIfNeeded()
      current application's NSThread's sleepForTimeInterval_(1)
      log "There was an error setting the recovery key as a custom property. The ePO agent may
        need to be re-installed."
      tell current application's NSAlert to set theAlert to alloc()'s init()
      tell theAlert
        setMessageText_("EscrowtoEPO Encountered An Error")
        setInformativeText_("EscrowtoEPO was unable to perform the escrow task because of a
          problem with the ePO agent. The key has been securely exported.")
        addButtonWithTitle_("OK")
        setAlertStyle_(2)
        set theResult to runModal()
      end tell
      syncButton_(me)
      theWindow's displayIfNeeded()
      my fvStatus's setTextColor_(current application's NSColor's blackColor)
      my fvStatus's setStringValue_("Finalizing the process...")
      current application's NSThread's sleepForTimeInterval_(1)
      do shell script "rm -rf " & plistDir with administrator privileges
    end try
  end

```

Figure 9. The encSuccess method in code (Part 1).



```

else
    theWindow's displayIfNeeded()
    my fvStatus's setTextColor_(current application's NSColor's blackColor)
    my fvStatus's setStringValue_("Synchronizing with the server...")
    -- Perform a Collect and Send Properties sync to send the key to the EPO server
    syncButton_(me)
    -- Remove the Recovery plist from the machine for security
    theWindow's displayIfNeeded()
    my fvStatus's setTextColor_(current application's NSColor's blackColor)
    my fvStatus's setStringValue_("Finalizing the process...")
    current application's NSThread's sleepForTimeInterval_(1)
    do shell script "rm -rf " & plistDir with administrator privileges
end if
on error
    theWindow's displayIfNeeded()
    my fvStatus's setTextColor_(current application's NSColor's redColor)
    my fvStatus's setStringValue_("An error occurred during the sync!")
    tell current application's NSAlert to set theAlert to alloc()'s init()
    tell theAlert
        setMessageText_("EscrowtoEPO Encountered An Error")
        setInformativeText_("EscrowtoEPO was unable to perform the escrow task as requested.
            Refer to the logs in Console for troubleshooting, or contact your system
            administrator.")
        addButtonWithTitle_("OK")
        setAlertStyle_(2)
        set theResult to runModal()
    end tell
    log "An error here indicates something happened during the escrow process. It could be the
        EPO agent's fault, given this point in the application."
end try
end encSuccess_

```

*Figure 10.* The encSuccess method in code (Part 2).

In addition to the programmatic features detailed above, EscrowToEPO utilized a status-checking script to check and report on the status of encryption on the system disk. This script, originally created by Rich Trouton of the Howard Hughes Medical Institute, uses command-line tools native to Apple's OS X operating system to verify and report on the status and percentage of FileVault 2 encryption being enabled on the disk (Trouton, 2013). In order to make Trouton's script work to maintain both recovery and status custom properties, an initializing section was added to prime the script to utilize McAfee EPO custom properties when reading recovery info and checking the status of the disk. Figure 11 shows the section in code that was added to Rich's script to enable the EscrowToEPO workflows:

```

#!/bin/sh -

# Credit to Rich Trouton for developing the initial FV2 Status checking script

# Change the permissions on /Library/McAfee applicable directories
sudo chmod 777 /Library/McAfee/cma/scratch/
sudo chmod 755 /Library/McAfee/cma/scratch/etc

# set plist directory as variable
secureDir="/usr/local/.InfoSec.plist"
customDir="/Library/McAfee/cma/scratch/CustomProps.xml"

# set the date for timestamp
timeStamp=`date`

if [ -f $secureDir ]; then
    echo "The recovery key file exists."
    echo "Now taking care of business."
    recoveryKey=`defaults read $secureDir RecoveryKey`
    serialNumber=`defaults read $secureDir SerialNumber`
elif [ -f $customDir ]; then
    echo "Custom Props xml is being used."
    echo "Grabbing the key."
    recoveryKey=`grep -o "\(\w\{4\}-\)\{5\}\(\w\{4\}\)" /Library/McAfee/cma/scratch/CustomProps.xml`
else
    echo "\n\nThe Recovery plist was not present in the secure directory."
    echo "The recovery key should have already been uploaded by e2e,"
    echo "but if not there has been an error."
    recoveryKey="n/a"
fi

```

Figure 11. Disk Status-Checking Script Initialization Code.

In addition to the changes made to initialize Trouton's script to use McAfee EPO, several single line code enhancements were made based on status checks to already existing syntax. Those lines executed the setting of the disk status script variable as a custom property and synchronization back to EPO, and are noted in Figure 12.

```

if [[ ${osvers} -lt 7 ]]; then
    encStatus="FileVault 2 Encryption Not Available For This Version Of Mac OS X"
    echo "$encStatus"
# Set ePO custom property for Recovery Key and Encryption Status and Sync
/Library/McAfee/cma/bin/msaconfig -CustomProps1 "$recoveryKey" -CustomProps2 "$encStatus" -CustomProps3 "$timeStamp" -CustomProps4 "$serialNumber"
sudo /Library/McAfee/cma/bin/cmdagent -P
fi

```

Figure 12. Single line code changes example; OS version check.

The modifications to the disk status script allow it to be used with McAfee EPO custom properties and synchronization tasks, but the script needed to be run periodically

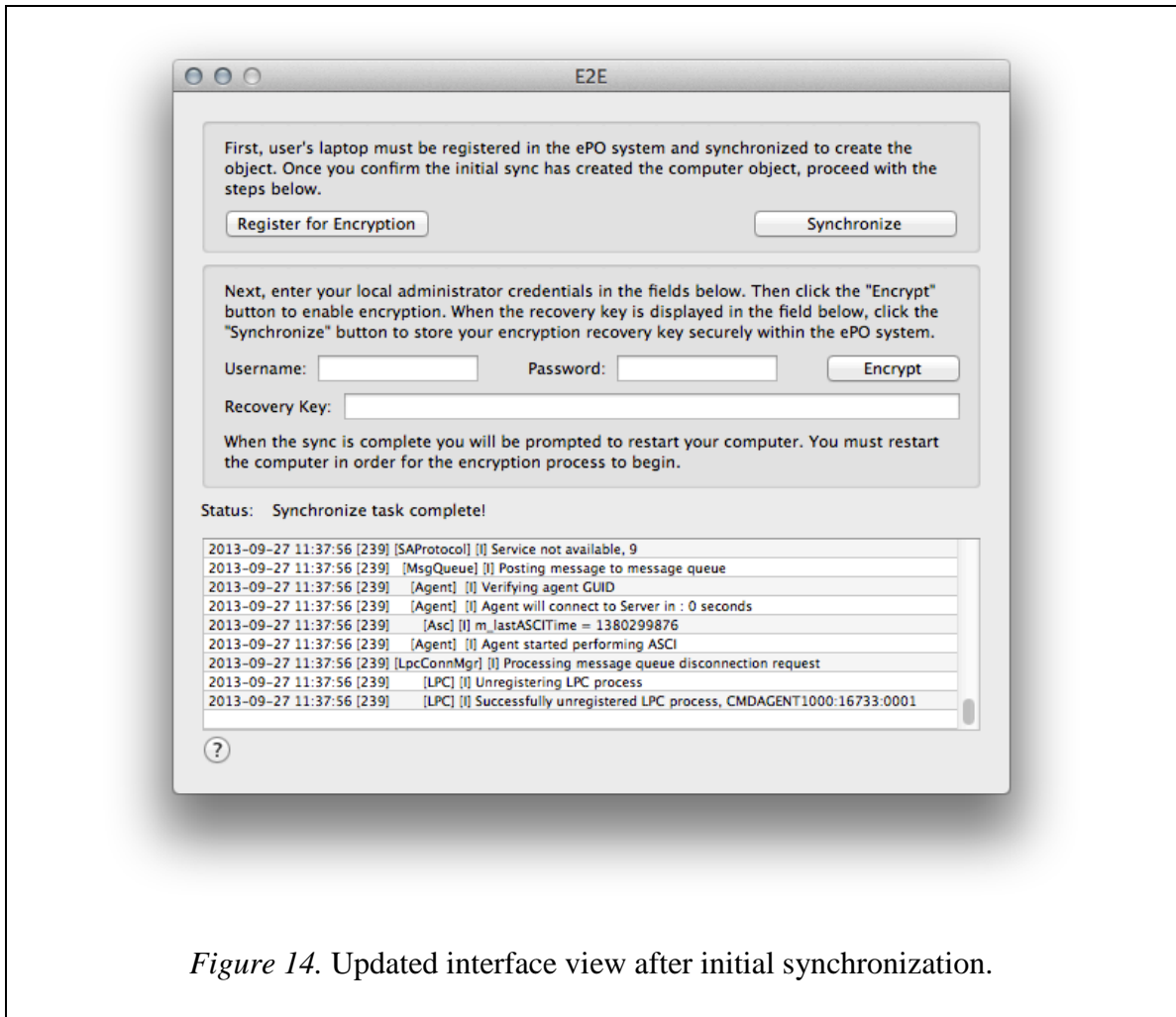
in the background in order to allow for accurate compliance reporting to the McAfee EPO server system. In order to set the script to run periodically, a system-level task executed the running of the disk status script on an interval of six hours. Apple allows the configuration of system-level tasks to be executed within OS X without interaction from the user when configured using a launchdaemon, or an advanced plist file (Apple, 2007). The launchdaemon for EscrowToEPO allowed the disk status script to be executed in the background every six hours and synchronized the status of FileVault 2 encryption on the system disk, as well as the recovery information back to the McAfee EPO server. The script was called by the launchdaemon and was included in the EscrowToEPO application bundle. Both the application (EscrowToEPO) and the launchdaemon were included in the packaged installation of the EscrowToEPO application installer. Figure 13 shows the syntax of the launchdaemon plist file used to accomplish this periodic reporting and synchronization task.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>Label</key>
    <string>com.yourcompany.e2estatus</string>
    <key>ProgramArguments</key>
    <array>
      <string>/Applications/EscrowToEPO.app/Contents/Resources/e2estatus.sh</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>StartInterval</key>
    <integer>21600</integer>
    <key>StartOnMount</key>
    <true/>
  </dict>
</plist>
```

*Figure 13.* EscrowToEPO launchdaemon plist file syntax.

## Results of Application Development and Testing

Due to the previous testing performed, the machine TS-TEST-AIR was decrypted in order to be re-used for the test. Additionally, a second MacBook Air allocated for development testing, named “TS-TEST-AIR2”, was included to show proof of consistency of the EscrowToEPO solution and ran the 10.8.5 version of Apple’s OS X Mountain Lion operating system (latest version available at test time). Both test machines also ran the latest version of McAfee’s EPO software, (version 4.6 available at test time) which facilitated the communication of the machine back to the McAfee EPO server. Finally, both machines were registered for enterprise recovery per the requirements of the enterprise’s intended encryption workflow. This exemplified the recovery process for administrators and ensures the successful communication to those managed objects on the server take place. Each machine, now ready to test EscrowToEPO, was loaded with the EscrowToEPO application via the packaged installer, and an initial synchronization was performed using the “Synchronize” interface button. Once the initial synchronization occurred, the interface for both machines reflected that the synchronization had taken place for that device. Figure 14 shows the updated user interface of EscrowToEPO on TS-TEST-AIR and TS-TEST-AIR2 after the initial synchronization process has taken place.



Notably, each managed object report generated by the McAfee server showed that there was successful communication from the background process. This communication verified that the disk encryption status and timestamp variables were reported successfully. This is evident with the fields “Custom 2” and “Custom 3” being updated with results from the successful background communication performed by the status script. With both machines having performed their initial synchronization, Figures 15 and 16 show the established initial communication for each device.



TS-TEST-AIR					
Last Communication	System Name	Custom 1	Custom 2	Custom 3	Custom 4
10/9/13 4:13:43 PM	TS-TEST-AIR	n/a	FileVault 2 Encryption Not Enabled	Wed Oct 9 11:10:56 CDT 2013	

Figure 15. McAfee EPO validation of initial synchronization for TS-TEST-AIR.



TS-TEST-AIR2					
Last Communication	System Name	Custom 1	Custom 2	Custom 3	Custom 4
10/9/13 3:48:22 PM	TS-TEST-AIR2	n/a	FileVault 2 Encryption Not Enabled	Wed Oct 9 00:45:25 CDT 2013	

Figure 16. McAfee EPO validation of initial synchronization for TS-TEST-AIR2.

After the initial communication was verified, the actual enabling of FileVault 2 encryption on the test computers could then take place. Using local administrator account credentials, both machines were enabled for encryption using the EscrowToEPO interface. Once the encryption process was completed, both machines were prompted for a restart by the EscrowToEPO application, and then restarted for the FileVault 2 encryption process to begin. Figures 17 and 18 show the EscrowToEPO interface view after encryption has been enabled and a prompt for restart is pending for both TS-TEST-AIR and TS-TEST-AIR2.

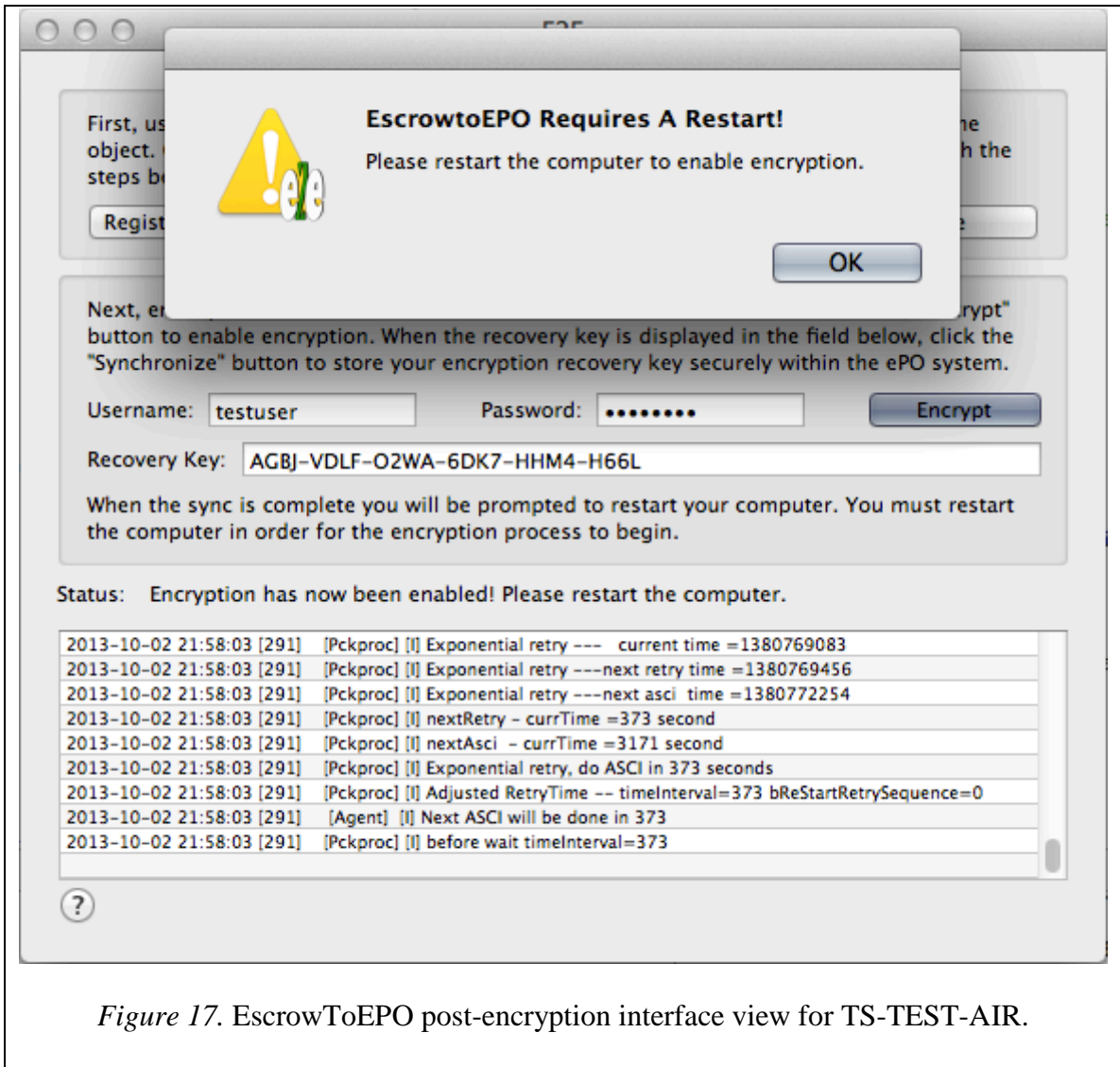


Figure 17. EscrowToEPO post-encryption interface view for TS-TEST-AIR.

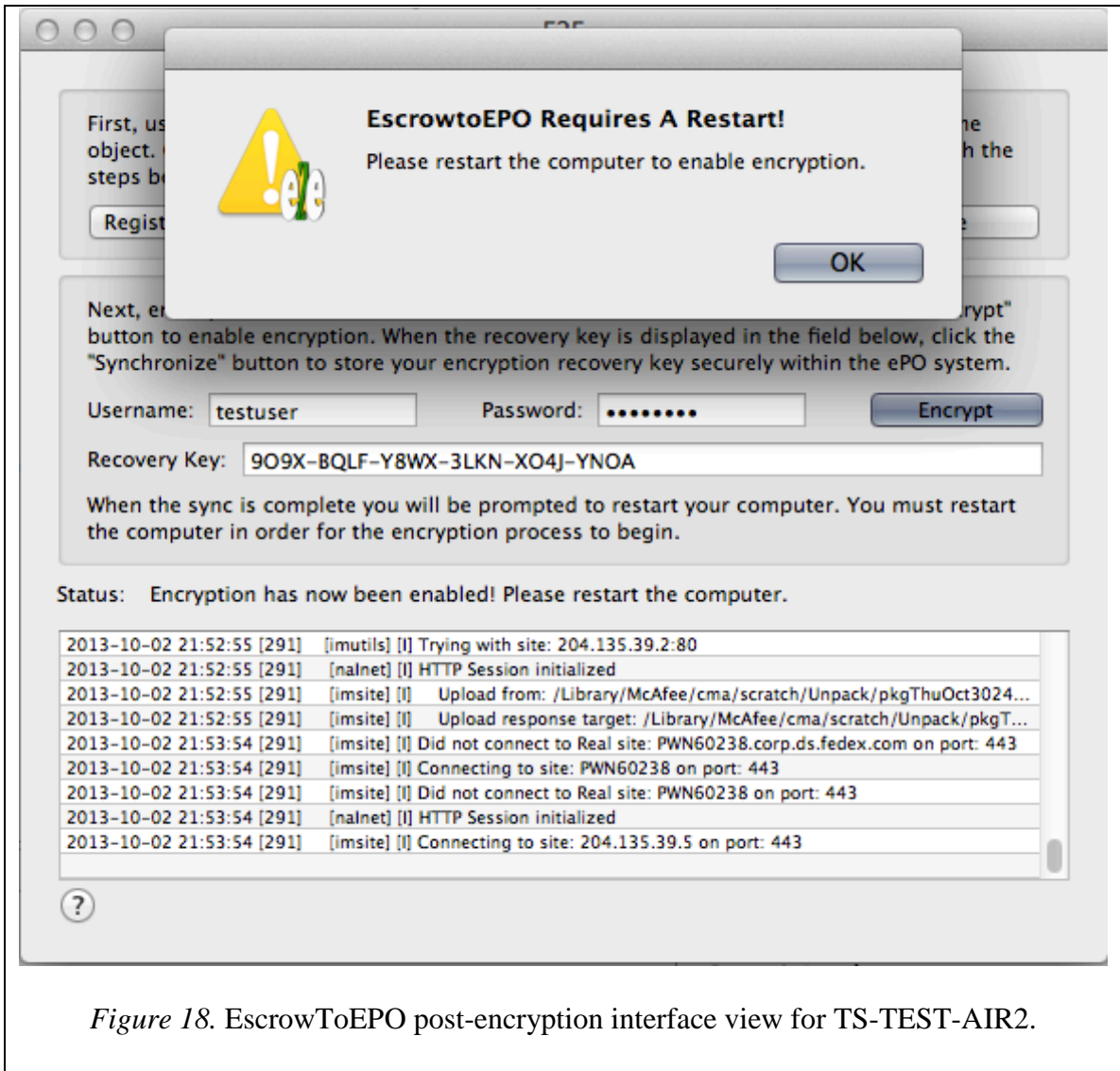


Figure 18. EscrowToEPO post-encryption interface view for TS-TEST-AIR2.

At this point in the testing, both machines were rebooted and authenticated to via the pre-boot encryption screen presented to the user by Apple’s FileVault 2 implementation.

Once the machines loaded the usable desktop interface of the OS X operating system, the McAfee reports were obtained from the EPO server. Figures 19 and 20 show the McAfee EPO reports as they exist post-encryption using EscrowToEPO.





TS-TEST-AIR					
Last Communication	System Name	Custom 1	Custom 2	Custom 3	Custom 4
10/3/13 6:47:03 PM	TS-TEST-AIR	AGBJ-VDLF-O2WA-6DK7-HHM4-H66L	FileVault 2 Encryption Complete	Thu Oct 3 10:30:39 CDT 2013	C02JDCBKDRVC

Figure 19. McAfee EPO post-encryption report for TS-TEST-AIR.



TS-TEST-AIR2					
Last Communication	System Name	Custom 1	Custom 2	Custom 3	Custom 4
10/3/13 2:31:10 AM	TS-TEST-AIR2	9O9X-BQLF-Y8WX-3LKN-XO4J-YNOA	FileVault 2 Encryption Complete	Wed Oct 2 17:31:02 CDT 2013	VMWVvk2KGj7O3kbqzD ZaY4Ahlg

Figure 20. McAfee EPO post-encryption report for TS-TEST-AIR2.

As indicated in the reports for both TS-TEST-AIR and TS-TEST-AIR2, all four “Custom” fields are populated with the information pertinent to the escrow, recovery, and compliance reporting of FileVault 2 full disk encryption management in the enterprise. Additionally, the post-encryption reports validated the recovery keys of TS-TEST-AIR and TS-TEST-AIR2 shown in Figures 19 and 20 for their respective managed objects on the EPO server. In this manner, EscrowToEPO has validated that the existing McAfee EPO management and server infrastructure was sufficient to manage and deploy native FileVault 2 full disk encryption technology in OS X Mountain Lion. Due to the confirmation of EscrowToEPO as a working solution to integrating FileVault 2 management into the enterprise’s existing environment, the roll out and adoption of

EscrowToEPO to the exempted pool of existing Apple laptop computers in the enterprise began to take shape.

### **Enterprise Adoption**

With the successful validation of the EscrowToEPO application workflows through development testing and successful replication on a second test machine, the information security organization of the enterprise decided to deploy EscrowToEPO as a solution to its encryption management integration challenge. As a result, the information security group of the enterprise reported a one hundred percent success rate in compliance reporting on the status of encrypted, managed objects in McAfee's EPO system. All of the Apple machines that received the EscrowToEPO manual deployment via packaged installation successfully communicated and reported the status of disk encryption to the McAfee EPO server. Figure 21 shows the success rate of adoption on machines to which EscrowToEPO was deployed.



Macs with FileVault Recovery Keys in CustomProp1	
Last Communication	Custom 2
9/27/13 9:28:39 PM	FileVault 2 Encryption Not Enabled
9/27/13 9:23:27 PM	FileVault 2 Encryption Complete
9/27/13 9:22:22 PM	FileVault 2 Encryption Complete
9/27/13 9:18:23 PM	FileVault 2 Encryption Complete
9/27/13 9:16:30 PM	FileVault 2 Encryption Complete
9/27/13 9:10:52 PM	FileVault 2 Encryption Complete
9/27/13 9:04:32 PM	FileVault 2 Encryption Complete
9/27/13 8:07:17 PM	FileVault 2 Encryption Complete
9/27/13 7:53:44 PM	FileVault 2 Encryption Complete
9/27/13 7:19:44 PM	FileVault 2 Encryption Complete
9/27/13 6:42:54 PM	FileVault 2 Encryption Complete
9/27/13 6:33:30 PM	FileVault 2 Encryption Complete
9/27/13 5:29:47 PM	FileVault 2 Encryption Complete
9/27/13 3:47:44 PM	FileVault 2 Encryption Complete
9/27/13 2:19:55 PM	FileVault 2 Encryption Complete
9/27/13 12:53:47 PM	FileVault 2 Encryption Complete
9/25/13 6:52:33 PM	FileVault 2 Encryption Complete
9/23/13 9:14:31 PM	FileVault 2 Encryption Complete
9/23/13 7:30:34 PM	FileVault 2 Encryption Not Enabled
9/23/13 5:14:33 PM	FileVault 2 Encryption Complete
9/23/13 2:17:55 PM	FileVault 2 Encryption Complete
9/18/13 5:45:38 PM	FileVault 2 Encryption Complete
9/9/13 8:53:14 PM	FileVault 2 Encryption Complete
8/28/13 8:19:51 PM	FileVault 2 Encryption Proceeding. of (248.7 GB) Encrypted
8/27/13 4:57:18 PM	FileVault 2 Encryption Complete
8/21/13 6:04:33 PM	FileVault 2 Encryption Complete
8/18/13 5:01:52 AM	FileVault 2 Encryption Complete
7/23/13 1:22:41 PM	FileVault 2 Encryption Complete
7/11/13 1:47:21 PM	FileVault 2 Encryption Complete
7/10/13 7:01:42 PM	FileVault 2 Encryption Not Enabled
7/2/13 4:34:24 PM	FileVault 2 Decryption Completed

Figure 21. EscrowToEPO success rate of adoption in the large shipping enterprise.

Despite the successful adoption of EscrowToEPO as an enterprise solution to FileVault 2 encryption management for Apple machines, several machines reported a “Not Enabled” status regarding FileVault 2 full disk encryption. Overall, some machines had installed the application, but users or technicians had not enabled FileVault 2 full disk encryption using the EscrowToEPO application. Due to the fact that the status script checked for the presence of CoreStorage volumes on a disk, Apple’s file system implementation for encrypted volumes, these systems were reporting correctly that they are not encrypted

(Apple Inc., 2012, p.23). Table 1 displays the percentage of various encryption systems in the enterprise.

Table 1

*Encryption status of Apple laptops in the large shipping enterprise*

Encryption Status	Number of Machines	FileVault 2 Enabled Percentage
FileVault 2 Enabled	27	87.10%
FileVault 2 Not Enabled	3	9.70%
FileVault 2 In Progress (Proceeding)	1	3.20%
Total Apple laptops receiving EscrowToEPO deployment	31	100.00%

## Chapter 5 – Conclusion

The successful development of EscrowToEPO and adoption of FileVault 2 in the large shipping enterprise signaled the importance of utilizing community work to enhance and solve integration challenges of both the FileVault 2 and McAfee EPO products. Further, the EscrowToEPO solution encourages this type of investigation to take place to facilitate resolutions to challenges with other products as well. With the successful validation and initial deployment, the EscrowToEPO application worked to further simplify the integration of existing enterprise infrastructure with new products and devices being evaluated and used by employees in the enterprise and beyond, specifically the Mac OS X operating system. The testing also validated the hypothesis that a software solution could be created for the large shipping enterprise to leverage its existing McAfee security infrastructure to integrate and manage the native FileVault 2 full disk encryption solution on the Mac OS X platform.

In addition, leveraging the EscrowToEPO solution provided the large shipping enterprise with widely documented procedures to troubleshoot and manage Apple's FileVault 2 full disk encryption product. To that end, future FileVault 2 integration and deployment challenges encountered by the enterprise can now be fully supported by Apple and McAfee. This is due to the fact that EscrowToEPO application uses supported methodologies of both products as their respective vendors designed them. Perhaps most importantly, EscrowToEPO has enabled a cost savings, as the information security group is no longer required to purchase and integrate a third-party solution.

By providing adoption benefits directly to the enterprise, EscrowToEPO also challenged the traditions of product design used by McAfee by utilizing the EPO

software in a much different way than the EPO documentation reflects (McAfee Inc., 2011). Throughout its various product guides, it is clear that McAfee designed the EPO software to be remotely deployed in an ideal network configuration where client management is pre-existing for all supported systems (McAfee Inc., 2011). However, EscrowToEPO and its background reporting features may have influenced McAfee to make drastic changes to their product offering for endpoint disk encryption on Mac OS X systems.

### **Industry Implications**

After testing and validation was completed, the EscrowToEPO project was provided not only to the enterprise, but also to the community that generated many of the solutions the applications utilizes to perform its functions. On September 11<sup>th</sup>, 2013, a private community group appeared on the McAfee community support site (McAfee Inc., 2013). This private community group, named “Management of Native Encryption – Preview Build”, advised that:

McAfee® Management of Native Encryption (MNE) is a management product that allows McAfee® ePolicy Orchestrator® (McAfee ePO™) administrators to manage Apple FileVault. McAfee Management of Native Encryption provides an easy-to-use administrative interface to manage and report on FileVault. This preview build is available for customers to deploy to a test environment only and should not be deployed to production systems. (McAfee Inc., 2013)

Due to the fact that the large shipping enterprise is highly invested in McAfee’s security products as an enterprise customer, it is possible that the EscrowToEPO application

influenced McAfee to investigate adopting FileVault 2 management into their EPO product.

The choice to incorporate FileVault 2 management into its EPO product could open the doors to more business for McAfee. As a security vendor, McAfee has now added client management functionality to their security product offering. The ability to manage FileVault 2 may be enough to retain the business of companies, like the large shipping enterprise, that originally invested in McAfee software for its Windows-based management capabilities and security configurations. With the additional feature of managing FileVault 2 disk encryption, McAfee can now offer companies the ability to report on various types of encryption systems across an enterprise spectrum of varying machine types. Now able to use EPO infrastructure to manage encryption compliance of multiple machine types, McAfee enterprise customers are able to integrate additional operating systems such as Apple's OS X Mountain Lion into their approved corporate standards.

### **Development Implications and Future Maintenance**

Despite the success of the project and validation of the hypothesis, there were some issues that arose during development of the EscrowToEPO application. The first issue occurred when the information security group applied an update to the McAfee EPO server infrastructure during application development. The security update changed the way EPO custom properties were retained by the server, and changed EPO policies controlling the behavior of the custom properties client file after a synchronization. As a result, the status-checking script initialization syntax was edited to allow for three different recovery scenarios.

In the first scenario, the EscrowToEPO has successfully enabled FileVault 2 and has exported the recovery information to a secure location without error. The background status script can easily read the recovery information and report the data to EPO without error if the first scenario is evident. In the second scenario, the background status script pulls the recovery information from the EPO custom properties file if the recovery information is missing from the secure location. This scenario occurs when EscrowToEPO encounters an issue exporting the recovery file, but can also occur if there is an error synchronizing the information during the enabling of FileVault 2. The third scenario occurred when both the recovery file and the custom property file are missing. In this scenario, EscrowToEPO may have uploaded the recovery information to the EPO server; however, both of the files may have become damaged, lost, or had their permissions changed, thus preventing the successful reporting from taking place. Scenario 1 is represented first in an “if” stanza, followed by scenario 2 in an “elif” (else if) stanza, and finally scenario 3 in an “else” stanza. Figure 22 represents the code syntax of the initialization section of the disk status script.

```
if [ -f $secureDir ]; then
    echo "The recovery key file exists."
    echo "Now taking care of business."
    recoveryKey=`defaults read $secureDir RecoveryKey`
    serialNumber=`defaults read $secureDir SerialNumber`
elif [ -f $customDir ]; then
    echo "Custom Props xml is being used."
    echo "Grabbing the key."
    recoveryKey=`grep -o "\(\w\{4\}-\)\{5\}\(\w\{4\}\)" /Library/McAfee/cma/scratch/CustomProps.xml`
else
    echo "\n\nThe Recovery plist was not present in the secure directory."
    echo "The recovery key should have already been uploaded by e2e,"
    echo "but if not there has been an error."
    recoveryKey="n/a"
fi
```

Figure 22. Three scenarios in status script initialization



In addition to the problems caused by the EPO update, another issue exists that will eventually impact how the large shipping enterprise maintains the EscrowToEPO application in the future. EscrowToEPO was written in the Applescript-Objective-C programming language. Applescript is an English-based scripting language native to the Apple operating system. Applescript can be used to call Objective-C methods and frameworks, but utilizes the deprecated garbage collection method of memory management for OS X applications (Stanley, 2012, p.217). In OS X Mountain Lion, Apple deprecated garbage collection, a method of managing memory based on periodic cleanup of the memory by a background task (Stanley, 2012, p. 217). Due to Apple's deprecation of garbage collection in favor of newer, more efficient memory management methodologies, the EscrowToEPO application may not function or be supported on versions of Apple's OS X operating system past 10.8 (Stanley, 2012, p. 217). This problem could be avoided in the future by continuing to develop the EscrowToEPO solution using only Objective-C code. This would allow the enterprise using the application to benefit from the most updated methodologies of memory management supported by Apple.

EscrowToEPO also carries implications to further research. When EscrowToEPO is compared to community solutions such as Christopher Silvertooth's Active Directory escrow scripts and Graham Gilbert's Crypt project, it becomes evident that Apple system administrators are looking for products or management solutions that will enable secure server management of FileVault 2 encryption and its recovery data. Further inference to FileVault 2 management can be

made when those projects are examined under the context of enterprise environments of various sizes. As referenced above, Apple's own FileVault 2 management guide acknowledges the emphasis placed on internal enterprise management of encryption recovery data to maintain corporate information security policies (Apple, 2012, p. 8). EscrowToEPO contributes to the community of FileVault 2 management solutions by adding a new workflow to a proven security solution produced and distributed by McAfee. Utilizing both an internal enterprise environment as well as existing security infrastructure, EscrowToEPO creates avenues for research with other security software produced by vendors other than McAfee and enterprises operating with similar environments.

### **Project Summary**

Building on existing solutions from community experts, EscrowToEPO was able to leverage supported methodologies from Apple and McAfee that can continue to be utilized by the large shipping enterprise to enable the low cost initiatives for encryption and compliance reporting. In the context of FileVault 2 support and troubleshooting, EscrowToEPO also enabled the large shipping enterprise to leverage well-documented procedures from Apple and its large community of system administrators. Through a standardized development process, the application development, testing, and validation ensured adequate alignment to the project requirements set forth by the needs of the large shipping enterprise. In conclusion, the EscrowToEPO application was able to provide the large shipping enterprise with a solution that integrated FileVault 2 management into their existing environment.

With the user experience also improved, the enterprise has acquired the usability and management components it needs to further adopt the OS X Mountain Lion operating system as an enterprise standard.

## References

- Apple, Inc., (2007, November 05). *Technical note tn2083: Daemons and agents*. Retrieved from [https://developer.apple.com/library/mac/technotes/tn2083/\\_index.html](https://developer.apple.com/library/mac/technotes/tn2083/_index.html)
- Apple Inc., (2010). *Property list programming guide* [White paper]. Retrieved from [https://developer.apple.com/library/mac/documentation//cocoa/conceptual/PropertyLists/AboutPropertyLists/AboutPropertyLists.html#//apple\\_ref/doc/uid/10000048i-CH3-SW2](https://developer.apple.com/library/mac/documentation//cocoa/conceptual/PropertyLists/AboutPropertyLists/AboutPropertyLists.html#//apple_ref/doc/uid/10000048i-CH3-SW2)
- Apple Inc., (2012). *Best practices for deploying filevault 2: Deploying OS X full disk encryption technology* [White paper]. Retrieved from [http://training.apple.com/pdf/WP\\_FileVault2.pdf](http://training.apple.com/pdf/WP_FileVault2.pdf)
- Apple Inc., (2012). *OS X Mountain Lion: Core technologies overview* [White Paper]. Retrieved from [http://movies.apple.com/media/us/osx/2012/docs/OSX\\_MountainLion\\_Core\\_Technologies\\_Overview.pdf](http://movies.apple.com/media/us/osx/2012/docs/OSX_MountainLion_Core_Technologies_Overview.pdf)
- Computer Associates., (2009). *CA IT client manager: Release notes and known issues r12 sp1* [White paper]. Retrieved from [https://supportcontent.ca.com/phpdocs/0/8176/8176\\_r12sp1\\_relnote.pdf](https://supportcontent.ca.com/phpdocs/0/8176/8176_r12sp1_relnote.pdf)

- Choudary, O., Grobert, F., & Metz, J. (2012). Infiltrate the vault: Security analysis and decryption of lion full disk encryption. *Cryptology ePrint Archive*, 374, Retrieved from <http://eprint.iacr.org/2012/374.pdf>
- Davisson, G., & White, K. (2013). *Os x support essentials*. United States: Peachpit Press.
- Gallagher, P. (2013, January 6). [Web log message]. Retrieved from [https://github.com/patgmac/scripts/blob/master/bash/Emory McAfee Mac Install/Emory McAfee MSM/Emory Install McAfee MSM.command](https://github.com/patgmac/scripts/blob/master/bash/Emory%20McAfee%20Mac%20Install/Emory%20McAfee%20MSM/Emory%20Install%20McAfee%20MSM.command)
- Gelles, D., & Waters, R. (2010, May 31). Google ditches windows on security concerns. *Financial Times*. Retrieved from <http://www.ft.com/cms/s/2/d2f3f04e-6ccf-11df-91c8-00144feab49a.html>
- Gilbert, G. (2013, January 18). [Web log message]. Retrieved from <http://grahamgilbert.com/blog/2013/01/18/crypt-a-filevault-2-escrow-solution/>
- Google, Inc. . (2012, November 29). *csfde: Cli tool to enable filevault 2 on a disk* [Google Project Hosting]. Retrieved from <http://code.google.com/p/cauliflowervest/wiki/Csfde>
- Hewlett Packard Development Company, LP. (2011). Open source security study: How are open source development communities embracing security best practices?. HP Enterprise Security, 1-6. Retrieved from: [http://www.hpenterprisesecurity.com/collateral/whitepaper/HPEnterpriseSecurity\\_WP\\_HPFortifyOpenSourceSecurityStudy.pdf](http://www.hpenterprisesecurity.com/collateral/whitepaper/HPEnterpriseSecurity_WP_HPFortifyOpenSourceSecurityStudy.pdf)

- Huo, M., Verner, J., Zhu, L., & Babar, M. (2004). Software quality and agile methods. IEEE 28th annual international computer software and applications conference. Retrieved from <http://utopia.csis.pace.edu/dps/2008/mcapellan/projects/SW Quality and Agile Methods - 04.pdf>
- JAMF Software, LLC,. (2010). *Administering filevault 2 on OS X Mountain Lion with the Casper Suite v9.0* [White paper]. Retrieved from [http://www.jamfsoftware.com/libraries/pdf/technical\\_papers/Administering-FileVault-2-on-OS-X-Mountain-Lion-with-the-Casper-Suite-v9.0.pdf](http://www.jamfsoftware.com/libraries/pdf/technical_papers/Administering-FileVault-2-on-OS-X-Mountain-Lion-with-the-Casper-Suite-v9.0.pdf)
- Levitsky, J. (2013). Mac OS filevault 2 management guide. Absolute Manage, 1.2, 1-21. Retrieved from <https://amrc.absolute.com>
- Mahesh, S., & Hooter, A. (2013). *Managing and securing business networks in the smartphone era*. (Master's thesis, University of New Orleans) Retrieved from [http://scholarworks.uno.edu/mgmt\\_facpubs/5](http://scholarworks.uno.edu/mgmt_facpubs/5)
- McAfee, Inc.,. (2011). *Product guide: McAfee agent 4.6.0* [White Paper]. Retrieved from [https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT\\_DOCUMENTATION/23000/PD23185/en\\_US/McAfee\\_Agent\\_4\\_6\\_Product\\_Guide\\_en\\_us.pdf](https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT_DOCUMENTATION/23000/PD23185/en_US/McAfee_Agent_4_6_Product_Guide_en_us.pdf)
- McAfee, Inc.,. (2012). *Product guide: McAfee Endpoint Encryption 6.2* [White Paper]. Retrieved from

[https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT\\_DOCUMENTATION/23000/PD23743/en\\_US/ee\\_620\\_product\\_guide\\_en-us.pdf](https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT_DOCUMENTATION/23000/PD23743/en_US/ee_620_product_guide_en-us.pdf)

McAfee, Inc., (2013). Technical articles – *Mac OS* unavailable after a thunderbolt firmware upgrade when endpoint encryption for mac is installed, encrypted, and active. [Technical Article]. Retrieved from

<https://kc.mcafee.com/corporate/index?page=content&id=KB72520>

McAfee, Inc., (2013). *Technical articles – Endpoint Encryption for Mac menulet continually shows 'Waiting for information' (after an OS upgrade while EEMac status is Active)*. [Technical Article]. Retrieved from

<https://mysupport.mcafee.com/Eservice/Article.aspx?id=KB72265>

McAfee, Inc., (2013). *Technical articles – FAQs for endpoint encryption for pcs*. [Technical Article]. Retrieved from

<https://kc.mcafee.com/corporate/index?page=content&id=KB72568>

McAfee, Inc., (2013). *Technical articles - endpoint encryption for mac client always shows the system state as inactive (troubleshooting article)*. [Technical Article].

Retrieved from

<https://kc.mcafee.com/corporate/index?page=content&id=KB68933>

McAfee Inc. (2013, September 11). Management of native encryption – preview build [Online community group]. Retrieved from

<https://community.mcafee.com/groups/mne-v10-preview-release>

Moore, G. 2011. Systems of Engagement and The Future of Enterprise IT - A Sea Change in Enterprise IT [White paper]. Silver Spring, Maryland, USA: AIIM.

Retrieved from

<http://www.aiim.org/~media/Files/AIIM%20White%20Papers/Systems-of-Engagement-Future-of-Enterprise-IT.ashx>

Neihaves, B., Koffer, S., & Ortbach, K. (2013, March). *The effect of private it use on work performance - towards an it consumerization theory*. 11th international conference on wirtschaftsinformatik, Leipzig, Germany. Retrieved from

<http://www.tecnostress.it/wp-content/uploads/2013/03/The-Effect-of-Private-IT-Use-on-Work-Performance-Towards-an-IT-Consumerization-Theory.pdf>

Scarfone, K., Souppaya, M., & Sexton, M. (2007). Guide to storage encryption technologies for end user devices. *Special Publications 800-111*, 1-40. Retrieved from <http://csrc.nist.gov/publications/nistpubs/800-111/SP800-111.pdf>

Silvertooth, C. (2012, April 09). [Web log message]. Retrieved from

<http://musings.silvertooth.us/2012/04/filevault-key-escrow-utilizing-active-directory-or-open-directory/>

Stagliano, T., DiPaolo, A., & Coonnelly, P. (2013). The consumerization of information technology. *Graduate Annual, 1*, Retrieved from

<http://digitalcommons.lasalle.edu/graduateannual/vol1/iss1/10>



Stanley, S. (2012). *Applescriptobjc explored*. (4 ed., pp. 216-219). Parkdale, Victoria, Australia: Myriad Communications Pty Ltd.

Symantec Corporation,. (2012). *Symantec encryption solutions for endpoints, powered by PGP™ Technology* [White paper]. Retrieved from [http://www.symantec.com/content/en/us/enterprise/fact\\_sheets/b-Encryption-Solutions-for-Endpoints\\_DS\\_21276725.pdf](http://www.symantec.com/content/en/us/enterprise/fact_sheets/b-Encryption-Solutions-for-Endpoints_DS_21276725.pdf)

Symantec Corporation,. (2012). *How drive encryption works* [White paper]. Retrieved from [http://www.symantec.com/content/en/us/enterprise/white\\_papers/b-how-drive-encryption-works\\_WP\\_21275920.pdf](http://www.symantec.com/content/en/us/enterprise/white_papers/b-how-drive-encryption-works_WP_21275920.pdf)

Trouton, R. (2013, January 28). [Web log message]. Retrieved from <http://derflounder.wordpress.com/2013/01/28/updated-filevault-2-status-scripts-now-available-now-handles-unencrypted-fusion-drives/>

## GLOSSARY

*Active Directory* – network operating system software manufactured by Microsoft. Widely used throughout enterprise/corporate IT infrastructure for authentication, file sharing, and security policies. Computers in an organization are bound or joined to Active Directory so that network services can be provided and computers can be managed (working definition).

*Apple ID* – Unique account, associated with an email address, that is used as an identifier by customers and developers using Apple services for file sharing, media and application purchase, security, and other various services (working definition).

*Applescript* – English-based automation scripting language native to the Mac OS X operating system environment. Applescript has the ability to call Objective-C, to facilitate programming an application through the use of this English-based programming language (working definition).

*End user* - An individual that does not exist in a technology needs-fulfillment role, or is an employee whose position does not exist to provide technology support, design, or troubleshooting. An end user is classified as any non-information technology faculty or staff member that uses technology goods or services under the support or guidance of information technology personnel (working definition).

*Fdesetup* – command-line utility released with OS X Mountain Lion to support and manage FileVault 2 (working definition).

*FileVault 2* – second-generation, full-disk standard XTS-AES-128 encryption product at the disk level designed for data at rest protection (Apple Inc., 2012, p. 3).

*Full-disk encryption* – the process of encrypting all the data on a hard drive or system disk that is used to boot a computer, including the operating system data. Access to the data is only permitted after successful authentication to the encryption management product or software (Scarfone, Souppaya & Sexton, 2007, p. 3-1).

*iCloud* – Apple’s free hosted (cloud) storage and communication service (Davisson & White, 2013, p. 50).

*McAfee ePolicy Orchestrator (EPO)* – McAfee security infrastructure software that allows for policy enforcement and computer security remote management. The software consists of a server architecture product and clients for multiple platforms that report to the server on the status and information of computer (working definition).

*Objective-C* – the primary programming language used for native application development in Apple’s OS X operating system. Objective-C contains several code bridges, including Applescript-Objective-C and Py-Objective-C (Python) (working definition).

*OS X Lion* – OS X (10.7) Apple operating system version released in Summer 2011. OS X Lion contained the first implementation of FileVault 2 (working definition).

*OS X Mountain Lion* – OS X (10.8) Apple operating system version released in Summer 2012. OS X Mountain Lion contained the first native implementation of the command-line utility to manage FileVault 2 (working definition).

*Personal Recovery Key* – Randomly-generated 120-bit alphanumeric code that can be stored with Apple or stored by the user. Using ‘*fdsetup*’, the key can

potentially be escrowed to a third-party system (Choudary, Grobert & Metz, 2012, p. 9).

*Property List (plist)* – A plist is a file containing a structured data representation used by Mac OS X for storage, access, and organization of standard types of data (Apple, 2010).

*Shell script* – a shell script is a text-based script written to interface and perform commands with the command line interpreter or interface of an operating system (working definition).

*Terminal* – primary UNIX command-line environment interface built into Mac OS X (Davisson & White, 2013, p.91).

*Xcode* – Apple’s integrated development environment software required for development of Mac OS X applications (working definition).

