Amelia - Next Generation Storage Management Product Architecture Specification (PAS)

Version: A.1

Revised: July 15, 2009

Main Contributors = Scott Hubbard, Bill Hetrick, Ray Jantz, Bill Delaney, and Scott Kirvan

A.1 Review Notes

An effort has been made to limit this specification to the following information.

- Describing the core architectural principles (underlying APIs, basic data flows (i.e. the “plumbing”), and foundational user experience elements).
- Setting the vision for the product solution, and
- Providing enough detailed, fundamental definitions of the understood requirements to drive further definition into the Aspect Architecture Definition (AAD) document.

Note: If there is some material deemed too detailed by the reviewers, that material can be moved into the Aspect Architecture Definition (AAD) document.

TABLE OF CONTENTS

1. Vision and Executive Summary

1.1. Concept Vision & Summary

Amelia is a strategic initiative that focuses on a complete refresh of the storage management software suite produced by LSI-ESG for the DAS, External RAID and DML (StoreAge) product sets. This is clearly a significant undertaking which includes a certain element of risk by going the route of new implementation versus evolutionary, incremental change to existing products. However, the requirements that are driving this initiative simply do not allow for evolutionary product enhancements to address the key product needs. The fundamental requirements are listed below, and are fully described in Section 2.

- Next-Generation Manageability and Ease-of-Use
- Common Management Software Base for All LSI-ESG Devices
- Industry-Standard SMI-S API for All Device Interactions
- Browser-Based Management GUI
- CLI with Industry-Standard Structures for Syntax, Usage and Deployment
- Integration for Open Storage Architecture (OSA) Applications and OEM Frameworks
- Development of Robust Multi-Device Management
- Solutions-Based Focus

When considered in unison, these requirements truly dictate that an entirely new management software package be developed. All of the existing product instances can be loosely described as having three key layers in their core (GUI) elements:

1. The top-level code that exposes a graphical (or component-oriented) representation of the storage system to the user and allows operations/actions to be selected.
2. A middle layer that converts between user-visible representations and operation selections to the underlying device interface model that allows configuration changes to be made, device status to be reflected upward, etc.
3. A bottom-level of code that understands the details of the API for interacting with the device, and manages basic information flow between it and the higher levels of the management application.

The combined need for (1) a radically-improved ease-of-use structure and presentation, (2) a GUI that is capable of running within a Browser window, and (3) a shared software base for all LSI-ESG device management applications, dictates that the top-level implementation of all existing solutions be discarded and re-implemented in a common/shared fashion. Similarly, the need to abandon proprietary APIs in favor of SMI-S (for both internally-developed and partner-developed applications) dictates that the bottom-level implementations of the solutions also be discarded. Because of the changes required for the top and bottom layers, the middle layer also becomes essentially obsolete and unusable as well. Therefore, taken as a whole, these requirements necessitate a more revolutionary approach rather than a set of incremental enhancements to one or more of the existing implementations.

Another critical factor driving the need for a radical departure from the existing solution is the requirement to integrate LSI-ESG-developed management software/capabilities into broader OEM software frameworks, and into higher-level storage applications running within the Open Storage Architecture (OSA) in next-generation LSI-ESG external controllers. OSA, in particular, is a strategic imperative, and it absolutely requires a streamlined mechanism for integrating base block-device management capabilities into the management structures of the higher-level OSA storage applications. The current application structures of the GUIs for SANtricity ES, MSM and SVM are not amenable to such integration. They are each

implemented as monolithic, large-scale applications, with very limited or non-existent support for streamlined, context-oriented integration into higher-level applications. This common flaw of the existing products further drives the need for a radically different application structure that *does* allow for activation of fine-grained device management components, in a context-aware manner, by OSA or OEM-provided applications.

This document provides detailed explanations of the driving requirements, and then outlines the vision by which they can be addressed via a new application structure that emphasizes shared components and user experiences for all LSI-ESG products, while at the same time allowing relevant product variations based on the target customer or market segment. The new structure also readily accommodates the need for context-aware activation of specific management sub-functions by independent applications that are layered on top of the basic block storage capabilities of the underlying devices.

1.2. Initial Implementation Leverage

PAS Note: This section is TBD. Contributors to this section are Scott Kirvan and Bill Hetrick.

This section identifies how the concept under development will build from existing technology or products developed by LSI including existing products or advanced development projects. It will also include other related concepts under development and the inter-related dependencies. External technologies (open source, industry standards, etc…) and products of interest should also be covered.

1.3. Terms to Know

- **AJAX** – Stands for *Asynchronous JavaScript and XML.* Ajax is a web development technique used for creating interactive browser-based applications. The intent is to make web pages feel more responsive and dynamic by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change.
- **CIM** – Stands for *Common Information Model*. CIM provides a common definition of management information for systems, networks, applications and services, and allows for vendor extensions. CIM's common definitions enable vendors to exchange management information between systems throughout the network. The standard language used to define elements of CIM is Managed Object Format (MOF).
- **CIMOM** – Stands for *CIM Object Manager*. This WBEM Services component listens for WBEM client requests for CIM operations. Requests on CIM class definitions are handled directly by the CIMOM. Requests on instances of a CIM class are dispatched to the provider responsible for extracting the dynamic data for that CIM Class.
- **Comet** – Comet is a term used to describe a web application model in which a long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it. Comet is an umbrella term for multiple techniques for achieving this interaction. All these methods rely on features included by default in browsers, such as JavaScript, rather than on non-default plug-ins. In theory, the Comet approach differs from the original model of the web, in which a browser requests a complete web page or chunks of data to update a web page. However in practice, Comet applications typically use Ajax with long polling (i.e. hanging poll) to detect new information on the server.
- **COS** – Stands for *CIM over SCSI*. This is a mechanism for transporting CIM commands over the I/O interface (in-band).
- **DAS** – Direct Attached Storage
- **DOM** – Stands for *Document Object Model*. A language independent API for managing HTML and XML documents allowing navigation of XML and HTML document structure and

content. DOM defines a high-level set of objects that provide an interface between scripting languages and the browser's internal objects.

- **Element Manager (EM)** – The Element Manager is a browser-based application that performs all of the management functions associated with a particular LSI hardware device type. There is a distinct Element Manager for each type of LSI device. An Element Manager manages one LSI device at a time. In SANtricity ES, this was referred to as the Array Management Window (AMW)
- **Enterprise Console (EC)** - The EC is a piece of LSI software used for centralized discovery, monitoring, and *basic* management of multiple LSI devices in a storage environment. In SANtricity, this was referred to as the Enterprise Management Window (EMW). The EC is also used for launching applicable Element Managers for each type of LSI device. Unlike the Enterprise Management Window (EMW) in the SANtricity product, this software is optional. That is, the user can opt to just perform single target-based management by pointing to the LSI device and using the Element Manager. However, for end users with more than just a few LSI devices, the EC provides the level of flexibility in management that they would expect.
- **GWT** – Stands for *Google Web Toolkit*. GWT is an open source AJAX software development framework for developing browser-based AJAX applications. You write your front end code in the Java programming language, and the GWT compiler converts your Java classes to browser-compliant JavaScript and HTML.
- **Host Context Agent (HCA)** – this is a host-based agent that collects host-related topology information (host name, host OS type, and associated HBA world-wide names) and pushes this down to the storage system to automatically configure host topology objects for use in volume/LUN masking operations.
- **HTTP** – Stands for *Hypertext Transfer Protocol*. HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP is a request/response standard of a client and a web server. In our usage of this protocol, a client is the end-user and the server is either the embedded web server on the controller or a web server on a host. For our management requests, we actually use HTTPS (see next definition).
- **HTTPS –** *Stands for Hypertext Transfer Protocol Secure.* HTTPS is a combination of HTTP and a cryptographic protocol. HTTPS connections are often used for sensitive transactions in corporate information systems.

PAS Note: Since we are not using a base Java application for either the Enterprise Console or Element Manager, do we need include the JNLP and JWS terms?

- **JNLP** – Stands for *Java Network Launching Protocol*. JNLP is a specification of protocols and APIs that will enable Java applications to be deployed on the Web.
- **JWS** – Stands for *Java Web Start*. JWS is a framework developed by Sun Microsystems that enables starting Java applications directly from the web using a browser. Unlike Java applets, Web Start applications do not run inside the browser, and the sandbox in which they run does not have to be as restricted, although this can be configured. One chief advantage of Web Start over applets is overcoming many compatibility problems with browsers' Java plug-ins and different JVM versions.
- **Licenses (Viral and Non-Viral)** – A viral license requires users of the resultant software package to disclose source code for all software that runs in conjunction with the open source. A non-viral license does not require that you publish the source code associated with your specific software but may require you to publish any changes (such as a bug fix) you made to the original open source code.
- **OSA –** Stands for *Open Storage Architecture***.** The Open Storage Architecture (OSA) is a software-based architecture and management solution for combining the data protection

layer (DPL) and block virtualization layer (BVL) technology assets of ESG with complementary storage applications to produce new, comprehensive storage solutions. Examples of such solutions include: (1) A unified block and file storage appliance consisting of the integration of ESG technology with an appropriate file-based application, or (2) A unified data replication appliance with services such as de-duplication or virtual tape libraries integrated with ESG technology.

- **RAS** – Reliability, Availability, Serviceability
- **SMASH CLP (SM-CLP)** – Stands for *Systems Management Architecture for Server Hardware (SMASH) Command Line Protocol (CLP)*. The SMASH CLP builds on the DMTF's Common Information Model (CIM) Schema. The SMASH initiative is a suite of specifications that deliver architectural semantics, industry standard protocols and profiles to unify the management of the data center.
- **SMI-S** – Stands for *Storage Management Initiative – Specification*. SMI-S is a storage management standard developed and maintained by the Storage Networking Industry Association (SNIA). SMI-S is based upon the Common Information Model (CIM) and Web-Based Enterprise Management (WBEM) standards defined by the Distributed Management Task Force (DMTF). The objective of SMI-S is to enable broad interoperability among heterogeneous storage vendor systems.
- **SMTP** – Simple Mail Transfer Protocol
- **SNMP** – Simple Network Management Protocol
- **VDS/VSS** – Stands for Microsoft's *Virtual Disk Service (VDS)* and *Volume Shadow Copy Service (VSS)* which are part of Windows Server 2003. LSI has developed a set of VDS/VSS providers which are applied as part of LSI Solutions to improve the efficiency of backup management processes for Microsoft Exchange and Microsoft SQL Server environments.
- **WBEM** – Stands for *Web-Based Enterprise Management*. WBEM is a set of systems management technologies developed to unify the management of distributed computing environments.
- **XML** – Extended Markup Language

1.4. Customer Impact

OEM customers and their respective end users that have had exposure to previous LSI-ESG storage management tools will experience the following impacts.

Element	Impacts
Enterprise Console	• For existing SANtricity ES users, the notion of an Enterprise Console (EC) will be familiar as they will have already been exposed to the EMW. • For existing DAS and SVM users, the EC will be a new piece of the management solution. • The robustness of the new EC will require all users to learn new concepts and operations.
Multi-Device Operations	The Enterprise Console will allow the user to perform selected operations on more than one device at a time thus easing the burden on the user.
CLI	• All existing users will need to transition to the new syntax based on SM-CLP. • All existing scripts will continue to work on legacy storage systems but will NOT work on Amelia-based storage systems.

Element	Impacts
One Externally Exposed API – SMI-S	• The only exposed API for managing LSI storage systems will be SMI-S. • If an OEM has developed their own management tool using existing LSI propriety APIs and SDK, they will not work on Amelia-based storage systems unless they are re-written using the SMI-S API.
Basic Storage Management	Many of the basic storage management high-level concepts and associated tasks will continue to be offered in the storage management UI. However, where possible, a more simplified approach will be taken.
Less Lower-Level Setting Control	Most of the lower-level storage settings will either be abstracted from the end user or be defaulted to appropriate values. A subset of these values may still be offered to more advanced-level users.
Greater User Flexibility	The user will have more flexibility in defining their overall management environment relative to tasks and views shown as well as obtaining more focused reports through pre-defined choices as well as query-based searches.
Storage Management Starting Points	• Depending on the level of application integration, abstractions implemented, and default objects that are created, the starting point for the end user, in many cases, will change to a higher-level entity. • Where appropriate, attributes and settings of the lower-level entities will be migrated to the higher-level entity.
Browser-Based Management & Installation	• Users will now perform all management tasks through a browser-based management interface (both for the Enterprise Console and the Element Managers). • For the Element Manager for External Storage and SVM, an installation of the management software will not be required. The management software will be embedded on the device. For the DAS environment, an installation is required. • For the Enterprise Console, an installation is normally required on a centralized management server. Multiple clients can then access the management server to gain access to the managed devices.
Converged Components	The Element Managers for each LSI device will share common, converged components wherever possible. This will ease the learning curve for end users when managing the spectrum of LSI storage systems offered. This will also streamline the training, documentation and support needed for the storage systems.
New User Interface Behavior and Navigational Schemes	Because of the browser-based and other simplification requirements outlined in this specification, the user interface will be modified to accommodate these changes. Users of existing LSI storage systems must adapt to the new user interface mechanisms. However, standard UI behaviors and navigation schemes appropriate to a browser-based environment will be used.

Element	Impacts
Management of Existing Legacy Storage Systems	• The Enterprise Console (EC) will fully manage only those devices using the Amelia architecture and corresponding APIs. • Any new multi-device operations will not apply to legacy storage systems. • At the very least, the legacy management software and Amelia management software can co-exist on the same client (i.e. for those pieces of the Amelia software that must be installed). • A basic mechanism may be built into the Amelia EC to discover and launch legacy systems. PAS Note: Are we really going to pursue this option? Bill H. needs to add in details in the Devices Supported section.

2. Concept Objectives and Core Strategies

2.1. Next-Generation Manageability and Ease-of-Use

LSI-ESG's current Storage Management Software products (SANtricity ES, MegaRAID Storage Manager (MSM), and SVM) were built around design points that used fairly traditional approaches for configuring and administering storage devices. In particular, the end-user is assumed to be relatively skilled at understanding storage concepts such as RAID levels, availability and performance tradeoffs, and overall device structures. However, this assumption is becoming less and less valid with time. Typical storage administrators are struggling with ever-increasing workloads and time demands, and thereby cannot devote nearly as much time or effort to traditional device management tasks. In addition, LSI-ESG's product set has gained significant traction in entry-level price bands, where dedicated storage administrators typically are not available, and system administrators often have only minimal exposure to even the most basic device management concepts and paradigms. Furthermore, each product iteration introduces new features with new options that, over time, have become an intricate and complex set of choices presented to the user.

To address this significantly changing environment, LSI-ESG's Storage Management Software product set must be revamped to expose device capabilities using high-level abstractions that make sense to even an unskilled administrator. At the same time, the software must employ and/or expose *policy-oriented* device management functions, where basic policy decisions can be made to drive *automatic* lower-level management operations, and thereby increase the productivity of both skilled and unskilled storage administrators.

One of the advantages that LSI has had over its competitors is the ease of use of our management tools. This has been repeatedly stated by our existing OEM customers. They have encouraged us to continue to maintain that advantage. Our user interface is really the portal in which our OEM customers and end users use and judge our product's features and functionality. While our management tools continue to stand up fairly well against our competition, there is increasing pressure from our OEM customers to further enhance the user experience and leapfrog our competition once again. The different factors discussed earlier in this section will continue to erode our competitive advantage in this area unless we continue to enhance the usability of the user experience we offer through our management tools.

In short, the manner in which device management capabilities are exposed to administrators must change in fundamental ways relative to the current product set in order for LSI-ESG's solution to be viable across all targeted price bands and market segments.

2.2. Common Management Software Base for All LSI-ESG Devices

LSI-ESG's device management software has become fragmented. Up until the Aurora release and the introduction of SANtricity ES, the external storage group offered two management solutions: SANtricity and Simplicity. We learned that this two-product set approach was difficult to clearly differentiate the functions needed for each price band environment for our end users and our OEM customers. In addition, despite sharing many common components, they required at least some level of independent and overlapping product development, certification, and maintenance activities, thereby reducing the overall efficiency of the product teams responsible for them.

Likewise, MegaRAID Storage Manager (MSM) is a totally separate application for the DAS product line. It adds a second independent software solution that shares essentially no common foundation with SANtricity ES. The SVM GUI is a third such package, and thus contributes yet another dimension to the efficiency problem.

More importantly, the user experiences within the SANtricity ES, MSM and SVM GUIs are all quite different from each other, so customers are forced to learn to use different tools when they purchase and deploy products from LSI-ESG's DAS, DML and External RAID product lines.

The combination of reduced internal efficiency within the LSI-ESG development organization, and increased complexity for end-users who deploy a range of LSI-ESG products, must be addressed in the Amelia program.

In particular, Amelia must establish common management paradigms, as well as common look-and-feel, for the entire range of products and devices. It must also enable significant reuse of common software components, presumably via a shared code base, for the construction of the device management tools for the DAS products, DML products, and all price-band releases of the External RAID products. If these objectives are met, end users will benefit from a common user experience in managing the entire range of LSI-ESG storage products, and LSI-ESG will benefit from improved internal efficiency (due to reuse and leverage) within its product development organization.

This product convergence has benefits across the various internal organizations and external OEM organizations that support and develop our LSI storage products (testing, publications, marketing, engineering, etc.) by eliminating duplications, consolidating similar functions, and developing commonality (e.g. terminology, training, and publications).

2.3. Industry-Standard SMI-S API for All Device Interactions

PAS Note: Does this section need to say anything about the fact that the external API will be SMI-S but that the internal API will continue to be SYMbol, StoreLib, and Corba?

All of LSI-ESG's core storage management applications were developed long before the SMI-S interface was solidified and standardized, so they all have their own proprietary native APIs for carrying out management and administrative actions. SANtricity ES uses the SYMbol (Sun RPC-based) API; MSM uses StoreLib; SVM GUI uses a CORBA (DCE RPC-based) API.

With the fairly broad adoption of SMI-S as a fundamental industry-standard API, it has become imperative to support this management mechanism in all LSI-ESG products. As is commonly done in the industry, this objective has been met with a stop-gap measure involving the use of Proxy Providers. These Providers can be viewed as plug-ins for CIM servers (i.e. CIM Object Managers, or CIMOMs) running within a selected server in the customer environment. A Proxy Provider translates between the SMI-S world of the CIMOM and the relevant device-specific API, and thereby enables SMI-S management of the device, provided that all management interactions flow through the associated CIMOM.

However, the requirement to have an external host act as the home for a CIMOM that runs Proxy Providers for storage devices is not appealing to our OEMs or their end customers. Such a solution requires that a system administrator select, possibly deploy, and then actively manage that server system in order to maintain administrative access to the physical storage device(s) being supported by the Proxy Provider(s). In addition, the complexity of integrating the Proxy Provider with multiple instances or even platform variations of a CIMOM makes this approach problematic for LSI-ESG in terms of both its development expense and its certification time-line.

Having native support within the device for the SMI-S API (i.e. "Embedded SMI-S Agent" capability) is, therefore, an increasingly urgent product requirement, especially for the External RAID and DML products. Proxy Providers may continue to be acceptable for the DAS product line. However, it is imperative that the management application(s) provided by LSI-ESG utilize SMI-S as the primary device API in all cases, and avoid using any proprietary low-level API (such as StoreLib or SYMbol). Note that this requirement necessitates enhancements to both the management software, which must *use* the SMI-S API in favor of a proprietary one, and to the device firmware, which must *support* the SMI-S API.

2.4. Browser-Based Management GUI

Browser-based technologies are becoming more robust and are getting closer to matching the user experience expected from a traditional desktop application. Browser-based management applications also offer the *Install Once/None, Manage Anywhere* advantage over client-installed applications. This advantage provides the user with easier storage set-up and management. OEM customers are also challenging us to reduce the amount of installable packages required on both the client and the host. OEM customers have recognized the emergence of Browser-based management tools and it has become an expected "check-off" requirement. Both the Enterprise Console (EC) and Element Managers (EMs) (explained in later sections) must be browser-based applications.

All of LSI-ESG's primary storage management applications (SANtricity ES, MSM, and SVM) have been implemented as heavy-weight, monolithic Java applications. They operate much like traditional non-Java applications, in that their binary images, coupled with a variety of support files, libraries, etc., must be installed in the file system of the client where the application is to be executed. This approach has two significant drawbacks that have become serious "pain points" for LSI-ESG customers:

- The additional administrative effort to install the management applications is viewed as a significant usability problem. It clearly adds to the administrative burden on the system where the applications are installed, particularly in terms of disk space provisioning for the application, plus ongoing monitoring for version updates and release currency. Furthermore, if an administrator has several workstations that may need to serve as management consoles, the applications needed to be installed on each of those workstations, thereby multiplying the already undesirable administrative burden.
- Since the management application is clearly an independent entity relative to the device firmware, it is necessary for the administrator to carefully coordinate updates of both the firmware and the application each time an update is applied. Again, the administrative complexity of managing such situations, especially when multiple devices are present in the environment, is simply not acceptable for already-overburdened system and storage administrators.

At various points during the life cycles of several of the LSI-ESG products, it was hoped that the Java foundation of the applications would enable them to run in a Java Virtual Machine within a browser, and thereby achieve the objective of being "install-free". Unfortunately, despite the Java language's significant and ongoing success as an enabler of rich, multi-platform applications, its original promise as a ubiquitous compute engine within all Web browsers remains unrealized, and this is not likely to change in the future.

One technology that *has* materialized as the nearly-ubiquitous compute environment for Web browsers is AJAX (**A**synchronous **J**avaScript **A**nd **X**ML), with JavaScript being the core programming language element of this technology.

In order to address the top-level requirement for browser-based management applications, the following structural objectives must be met:

- Implement the applications in JavaScript, relying on AJAX capabilities within the browser execution environment.
- Rely on the device itself (except, possibly, in DAS deployments) to be the repository of the JavaScript executables that are fed out to the browser for execution. The direct implication from this repository requirement is that the device itself must provide an HTTP server capability, albeit a limited one that is focused on the need to feed out its management application.

2.5. CLI with Industry-Standard Structures for Syntax, Usage and Deployment

The CLI capabilities of the various management packages have typically been deployed as a secondary mechanism, often with limited capabilities relative to the rich functional model of the corresponding GUIs. In addition, the syntax rules employed by LSI-ESG's CLI products are varied, and often highly customized relative to any form of industry standard. This "second-class" status for the CLI no longer aligns with the manner in which many customers wish to manage their devices. In particular, customers are demanding fully functional CLIs whose syntax and usage models conform to evolving industry standards, most notably SM-CLP. In fact, an increasing number of customers (especially within the HPC/HEC market segment) utilize the CLI, rather than the GUI, as the primary interface to the device management capabilities of the product. It is therefore imperative that the Amelia efforts include development of a CLI that is functionally complete, and whose syntax/usage is based on the model set forth in the relevant SM-CLP standards.

2.6. Integration for OSA Applications and OEM Frameworks

A common need for both LSI-ESG's Open Storage Architecture (OSA) and for the broad management framework software suites of key OEMs is for key functional elements of the Amelia application to be "launch-able" from those external applications. For example, in cases where an OEM's management framework is incapable of properly handling a unique feature of an LSI-ESG storage device, the framework developer may elect to invoke the LSI-ESG mechanism (within the Amelia package) for manipulating that feature. Similarly, in an OSA environment, the OSA application (whether it is developed by LSI-ESG, or a partner, or an OEM) may need to invoke the specialized management software for a particular feature of the storage device, rather than re-implement that function within the OSA application. In either case, the primary requirement is that the Amelia management application (i.e. the GUI, in particular) be structured so that all "important" functional capabilities are implemented as components that can be activated within the Amelia application, but also within an external OSA or OEM application, in a *context-oriented* fashion.

The context orientation is a critical aspect of this requirement. In general terms, it means that the launched component immediately present the relevant configuration or status information and option selections to the end-user, rather than requiring the user to re-navigate from a higher-level view to the appropriate dialog or wizard (or other element) of the Amelia application. By properly selecting the right set of components to be implemented in this manner, along with supporting reasonable "context" options, the components can be made to appear much like standard elements of the external application that invokes them. As a result, the OEM or OSA applications can be developed or customized without requiring re-implementation of intricate management interface capabilities that already exist within the Amelia application.

Note that the browser-targeted structure of the Amelia GUI offers two related benefits in terms of easing the integration burden with OEM/OSA applications. First, the prevalent trend in the industry is to develop management applications for use in browser environments, so the browser-based structure of Amelia dovetails perfectly with this paradigm. Second, for cases where the top-level application is not browser-based, the fact that Amelia does run within a browser is still a substantial benefit, since some browser will almost universally be available within the application environment for use by the context-based Amelia component. Admittedly, this latter case suffers from an obvious look-and-feel distinction between the Amelia component and the main application, but in most cases that will be outweighed by the benefits of just being able to leverage the Amelia functional component.

2.7. Development of Robust Multi-Device Management

The need to provide a user with multi-device management functions via an Enterprise Console (EC) has become critical as more users are faced with managing multiple storage devices in their environments. While the current SANtricity ES Enterprise Management Window (EMW) provides basic multi-device discovery and coarse-grain health monitoring, it lacks the type of multi-device functions and information needed by storage administrators. Too often, storage administrators are forced to launch a device's individual storage management tool to complete basic administrative tasks on more than one device. For example, if users want to ascertain the firmware levels of each discovered device or collect support data, they are required to open the individual storage management tool for *each* desired device. This becomes a very time-consuming and repetitive task.

Even though the EC is an optional piece of software, users having more than just a few storage devices to manage will quickly desire the functions offered in the EC and make it be the focal point for their on-going management. Other reasons for developing the EC are as follows:

- Not all of LSI's OEM customers have plans to develop their own Enterprise Frameworks and thus will rely on LSI to provide its own Console.
- While several of our larger OEM customers have plans for developing their own Enterprise Frameworks, many are still in the planning or early development stages. They may need a back-up plan, at least initially, to use our Console.
- Even for those OEM customers that develop their own Enterprise Frameworks, the robustness of these Frameworks may exceed (i.e. not match) the customers' needs or price-sensitivity especially in the lower price bands. Our Enterprise Console would be an excellent, less-expensive alternative for our OEMs to offer.
- At least in the External RAID environment, LSI has provided the Enterprise Management Window (EMW) as a basic component of our overall software management solution. Existing customers will expect a tool that matches or exceeds the current functionality of the EMW.

2.8. Solutions-Based Focus

While a majority of the initiatives for the Amelia project center on providing new and innovative storage management, that aspect is just one piece of the overall solution.

The Amelia project also emphasizes looking at the overall life cycle of a storage system and determining the appropriate solutions (stand-alone tools, information, training, etc) that will aid our end users and sales personnel in reducing the "pain points" associated with each stage. One easy indicator of a "pain point" is to look at the amount of pages devoted to explaining how to complete a given task with the software and hardware. Generally, more difficult and un-usable operations are associated with higher page counts.

A solution-based approach will truly set us apart from the competition. Therefore, this specification also outlines several other important solutions needed for the Amelia project.

3. Common Use Cases

PAS Note: This section is TBD. The main contributor for this section is Bill Hetrick.

4. Life Cycle-Based Solutions

4.1. Definition of Life Cycle Stages

In the life of a storage system, the end user manages the product through the following stages:

- **Plan**– this stage is generally conducted before the product arrives and involves learning about the product's features, determining factors needed for storage allocation and configuration, and having pre-training/demonstrations of the user interface. There may be aspects of planning that are also conducted once the product arrives (i.e. storage planning and basic training).
- **Initial Setup** – this stage involves the actual installation of the hardware itself (cabling, etc.) and setting of any initial software parameters to be able to communicate with the storage product as well as any initial storage-wide global settings. It is imperative that this stage provides the user with "quick successes" to ensure an early favorable impression of the management software and the product.
- **Configure** – this stage includes the provisioning of the storage into base storage and higher-level copy services, as well as other configuration tasks dependent on the capabilities of the storage product (file-based tasks, policy-based tasks, storage tiering, security, user roles, etc.), performing acceptance testing on the configuration, and deploying the configuration into a live environment. While there are many tasks involved in this stage, once the core configuration tasks have been completed, the functions within this stage are infrequently used as users are reluctant to make changes once the product is deployed into a live environment.
- **Monitor/Maintain** – after the storage product has been deployed, a majority of the time spent with the product will be in this stage. This stage is divided into the following areas:

PAS Note: Need input on whether Monitor and Maintain should be one life cycle stage or be broken into two. Currently it is presented as one since the maintenance functions (tune and upgrade) would normally result from a condition detected via the Monitoring.

    - Reports - While the product is in operation, the end user will want to monitor it for problems as well as obtain various reports on its usage, trending data, and performance information.
    - Diagnose & Recover (Service) - when a problem has been detected, the user and support personnel will be provided with the appropriate tools to easily and quickly diagnose and recover from the problem.
    - Tune – based on various monitoring and reporting activity, the end user may need to perform various tasks associated with tuning their storage.
    - Upgrade – this area allows the user to perform tasks associated with upgrades to the storage product (software/firmware upgrades as well as upgrades to the hardware (e.g. storage expansion, etc.)).

Note: Results of monitoring activities may cause actions to be taken in other stages as well.

- **Retire Storage** (Data Migration) – This area involves both the decommissioning of the entire storage product as well as moving portions of the storage onto different tiers of storage relative to various quality-of-service (QoS) attributes based on the relative importance of the storage.

By analyzing and focusing on these life cycle stages, we can ensure that LSI develops comprehensive and innovative product solutions (hardware, software, documentation, and training) to meet the needs of the user for each life cycle stage.

4.2. Benefits of Life Cycle Management

The life cycle management organizational structure benefits both the end user as well as providing benefits for internal organizations.

4.2.1. Intuitive

A product's life cycle stages are an intuitive and easily recognized concept for end users. In a survey commissioned by LSI through the Hattras group, all 20 respondents surveyed acknowledged that these stages accurately reflected the stages they go through when managing storage.

4.2.2. Supports Modular Design

- By designing and organizing the management software (and other solutions such as training and documentation) around these product life cycle stages in a modular fashion, end users are provided with only the appropriate views/objects, tasks, and information they need to effectively manage their storage at the proper stage of the product.
- The life cycle organizational structure is also a natural mechanism for providing modular software components that can be used to…
    - Provide the user with the flexibility to manipulate the management software to fit their particular environments and needs (for example, giving the user the capability to load only specific software life cycle modules). This allows the user to "clean up" the user interface by eliminating views and tasks that may no longer be applicable.
    - Provide the necessary structural organization need for launch-in-context (LiC) requirements. The various LiC tasks will be organized and tagged as belonging to a particular life cycle stage. This gives a higher level application the flexibility to call a single LiC component or a set of LiC components belonging to a particular life cycle stage.
    - Provide self-contained, independent tools that can be either deployed as part of an overall product (normal case) or as a separate application. For example, the tasks and views associated with the Monitor stage could be deployed as a separate application for a mobile device.
    - Aid in designing a flexible solution relative to loading only the necessary data and views into the browser for performance reasons.

4.2.3. Easier Integration

Because of the adaptability of the life cycle stages organizational structure, integration of other features/functions from OSA-based applications can fit seamlessly into these stages. If needed, additional stages can be added.

4.2.4. Product Solution Focus

- The life cycle stages also help developers and product solution architects focus on the best possible solutions for each stage. That is, each stage should be looked at as an independent application with the goal of providing the necessary Views, Objects, and Tasks to support the activities, information, and requirements of that stage.

- Along with the management software being highly modularized based on the life cycle stages, the publications, marketing, and training organizations can use the stages to organize (i.e. modularize) their supporting material giving the user only the information that is needed for a particular life cycle stage.

5. Existing Function Analysis

As part of the Amelia definition, a function analysis must be performed to…

- Determine the core functions to migrate from existing solutions (this must include a list of both retained and deleted functions).
- Indicate which life cycle stage or stages a particular function resides (this is used as both an organizational aid for design and metadata purposes as well as a checkpoint to ensure we have the proper solutions).
- Determine the changes needed, if any, to the retained functions such as
    - Individual choice/parameter assessment. What individual choices/parameters to retain, drop? What appropriate defaults to use? This should be based on the philosophy (discussed later in this specification) that we want to reduce, where possible, the choices that a user must make.
    - Usability assessment. Is the current function and current UI elements appropriate? While the UI look & feel itself is changing, we need an upfront prioritization for each function/UI design to make sure we understand the return on investment relative to time and effort. This is just to make sure that we focus on the most important aspects.
- Determine where the retained functions are exposed to the user (Enterprise Console, Element Manager, Both, or Stand-Alone Tool).
- Determine if there are any in-flight development activities being planned (for example, need to understand if there are enhancements in the various areas (BVL, DPL, or DAS) that would impact the Amelia base architecture or functionality). See also section 26.

Note: The output of this analysis would be a function table matrix capturing this information.

6. Devices Supported

6.1. Device Support Requirements

The storage management solution outlined in this document provides support for any LSI storage device that meets the following requirements:

PAS Note: These requirements need to be updated based on requirements dictated by partnering activities for the Enterprise Console.

- Has full-featured SMI-S support
- Uses new Element Manager for its device management

This includes Amelia-based external storage (including file-based), Direct-attached storage (DAS), External SVM (DPMs) and Application Servers (used for policy management and back-up).

6.2. Continued Legacy Device Support

PAS Note: Need to further investigate what we have planned for legacy support. Bill Hetrick has ideas here (see section 11.3.5).

Legacy LSI devices that don't meet the requirements stated in the previous section will continue to be supported, for a period of time, using their existing APIs and management tools.

Note: At a minimum, the Amelia Enterprise Console and the EMW/AMW from legacy devices must be able to co-existence on the same management client.

7. Global Architectural Principles/Elements

This section describes global principles/elements that need to apply across the various UI designs and core architecture for both the Enterprise Console (EC) and the Element Managers (EM).

7.1. Need to Address Three Types of Users

Amelia needs to address the following users:

- Main user = *Storage generalists*. These individuals have a wide range of responsibilities beyond strictly storage and, unlike in the past, do not want to understand the underlying technology. This user wants to get up and running as quickly and as easily as possible and is more likely to accept the defaults that are provided. This user grew up using the web and various web-based technologies and relate more favorably to information that is presented in a graphical manner.
- Secondary user = *Storage specialist*. These individuals are used to being in more control of the system and want to do more fine tuning than the storage generalists. The environment in which storage specialists work demand that the system is optimized for peak performance (e.g. HPC environment). While we want to minimize the number of tunable parameters presented, we must ensure that we provide the level of control needed for these users because they still exist where our products are being sold.
- Secondary user = *Support personnel*. When an issue arises, support personnel want to quickly and easily diagnose the problem and recover from it using the tools within the storage management tool. Any time a problem occurs with an end user that requires intervention from a support person is normally at level of severity that *demands* quick and accurate diagnosis. End user perception of the quality of the product and company becomes critical during these times.

7.2. User-Centered Tasks & Abstractions

Wherever possible, all tasks provided in the management tool should be driven from the user-perspective (i.e. keeping the user's goals in mind both from a starting point and end result) when designing the different views and tasks for the user to complete. In addition, any objects and/or settings that can be abstracted (removed) so that user manipulation is not required is preferred.

Note: Both the user-centered tasks and the abstractions need to be balanced against the need to provide the agreed-upon full functionality and flexibility offered in our product set (for example, we don't want to present a task or abstract an object that would prevent a user from having access to a particular, needed feature).

Note: It also needs to be kept in mind that the user perspective and abstractions will change depending on the applications that are being offered to the user with our product solution (for example, if we are offering a File-based application, some block-level objects may be abstracted and the starting point for the user may change).

7.3. Automation/Policies

Another method to reduce the workload on the end user is to provide automatic actions to be taken based on pre-set policies. Methods for providing policy-driven actions need to be determined as the various functions/actions are defined. Care should be taken to ensure that the user has control over these policies and understands the results and actions that will be taken.

7.4. User Control

Each user must have the ability to manipulate his/her storage management environment through appropriate user preferences and the ability to manipulate various views and reports. Allowing users to customize the user interface to meet their specific needs is a critical usability differentiator. Each time the user changes a preference or a view, it needs to be persisted so that the next time the user enters the application, the preferences and customizations have been retained. Specific user controls and preferences are defined in the AAD document.

7.5. Experience Level Tiering

Because the storage manager is used in different price band environments and by users with different levels of experience, the following elements are used to provide appropriate user adaptation.

Element	Description
Experience Level Tiering	**Rationale** We're continually faced with the challenge of what tasks and views to present to the user and how to accommodate different levels of user experience. The model described below uses a combination of internal control as well as putting control and flexibility into the user's hands. Many user interfaces today allow the user to control their views and the tasks that are presented. Experience level tiering can affect the actual tasks and views that are displayed. **Description** A 3-tier hierarchy is provided to internally and externally identify the tasks/views. • Tier 1 – these tasks/views are the minimum required to manage a device through the different life cycle stages. • Tier 2 – these tasks/views build upon tier 1 and offer more sophisticated options to the user. • Tier 3 – these tasks/views are advanced and should only be used by experienced users or support personnel. An internal site map is provided that lists all of the actions with characteristic information (metadata) about each action and view (such as tier, keywords, etc.). This site map can be used by LSI and OEMs to define/change the various metadata for each action/view. For details, see section 16. **Note**: It is possible that a tier associated with a particular task/view could vary depending on the price-band at which the application is targeted (for example, task A may be tagged as Tier 3 in an SMB environment and a Tier 2 in a Price-band 4 to 5 environment). The following is an example of how the tiering would work: • An OEM chooses a Standard Feature Control mechanism which specifies a set of functions that are exposed for the product. For these functions, there is a defined set of supported actions and views. Each action and view is tagged with a tier designation as deemed appropriate by internal/OEM organizations. • When the user launches the storage management application for

	the first time, a default tier is shown to the user. The user has the ability at that time (or a later time) to change what tiers are shown by default.
Task Identification	The three levels of tiers are clearly identified throughout the user interface as an aid to the user.

7.6. Minimal Mouse Clicks and Choices

- Emphasis must be placed on using the minimal amount of mouse clicks to complete a particular task. However, care must be taken not to over-analyze this requirement to the point of agonizing over 10 clicks versus 7.
- Emphasis must be placed on displaying only those choices to the user that are absolutely required to provide a rational outcome. The desire is to reduce the number of inputs/choices the user must input and decide on to complete an operation. See section 7.8 for information on Simple and Advanced Paths.
- Where appropriate, default values must be used for choices as well as naming.

7.7. Multi-Object Manipulation

As the number of objects increase, the burden (time and productivity) of performing a common operation on some or all of those objects becomes greater. Therefore, along with the emphasis on minimizing the number of mouse clicks and choices to complete an operation, the user has the ability to select multiple objects to perform certain actions. These actions are normally performed on homogeneous objects, but certain operations can be performed on heterogeneous objects.

7.8. Simple and Advanced Paths

Depending on the particular operation, a user must be presented a simple path and an advanced path.

- These should be two distinct paths. This can originate at the same starting point (with the default being the simple path) but once the selection is made, the two paths should not be intertwined even if certain screens are reused between the two paths. It must be made clear to the user the differences between the two paths and the output from each path.
- The simple path must provide the most straight-forward flow and minimal input to complete the particular operation. However, it needs to be assured that the outcome of the path is providing a viable solution. For example, instead of completely suppressing options, it may be necessary to present the options but then default to the most likely choice. Any suppressed options must be those that we normally don't need to expose to the user except down a more advanced path. The goal must be that 90% of the users can choose the simple path.
- The Advanced Path can assume a more knowledgeable user and present additional control options in more of a worksheet-type of format rather than a lot of segmented screens.

7.9. Highly Graphical

The user interface must be *highly* graphical not only in the main views presented but also in various supporting dialogs and wizards. These graphics must be used as the main mechanism for conveying information and performing object manipulation.

7.10. Initial Usability Perception Awareness

Ease of use during exposure to a product through demo packages or first time usage is critical in helping the user form a favorable usability perception of the product. Unique “wow” factors (such as sound, animation, etc.) and “quick successes” must be incorporated into the user interface to help the user form this early perception. However, care must be taken to not implement UI features that become annoyances after repeated usage. Users must always be given the option to turn off these features. UI features, in the end, must always have a useful function.

7.11. Information Display Optimization/Tiering

Similar to experience levels, information displays must also use a tiering method to ensure that the appropriate amounts and kinds of information are displayed to the user.

- Tier 1 is the most critical and frequently used information. This information must be displayed in the main views and dialogs. Tier 1 information provides a mix of graphical and textual information.
- Tier 2 is supplementary information to Tier 1 and provides the next level of needed information to the user. Tier 2 also provides a mix of graphical and textual information, where appropriate.
- Tier 3 information is detailed, “full disclosure” data that is infrequently used but provides the level needed for detailed analysis and service/support activities. Tier 3 information is generally textual in nature and provided in a format that can be easily parsed and exported to other programs (such as spreadsheets) for further analysis.

7.12. Wizard and Dialog Standardization

- All wizard sequences conform to a standard behavior as outlined by the UX team.
- The user must always be given the option to by-pass introductory and confirmation screens (for non-destructive operations). This is especially important for repetitive types of operations. Once the user understands the purpose of the operation, introductory and confirmation screens become more of a nuisance than a help and just add to the number of clicks it takes to complete an operation.

7.13. Drag-and-Drop Functionality

Drag-and-drop functionality is supported throughout the user interface where deemed appropriate.

7.14. Navigational Methods

Both the Enterprise Console and individual Element Managers must have consistent methods for navigating the user interface and selecting objects and tasks. Later sections provide specific definitions.

7.15. Online Help

All Amelia-based applications must use WebWorks as the common help system.

7.16. On-Screen Text/Labels – Minimalistic Approach

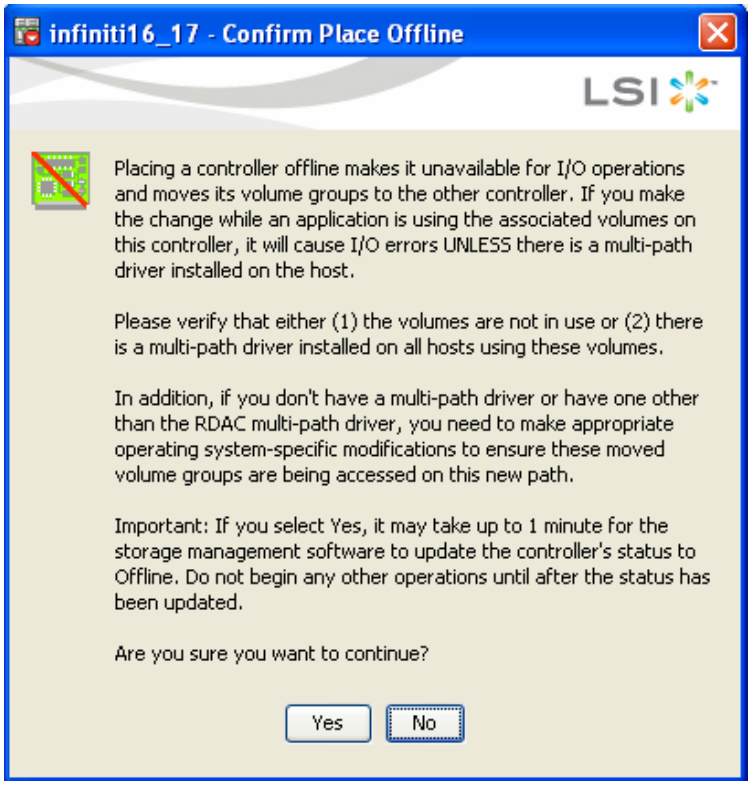
- When first being introduced to a new operation, on-screen text is invaluable in providing appropriate cues for the user to successfully understand the operation and the input that is required. However, once a user has completed an operation just a couple of times, the on-

screen text becomes “noise”. Therefore, a good balance needs to be attained to not overload the screen with un-necessary information. “Anticipatory Contextual” help links on the screen is a good method to provide additional details to the user without cluttering the main screen.

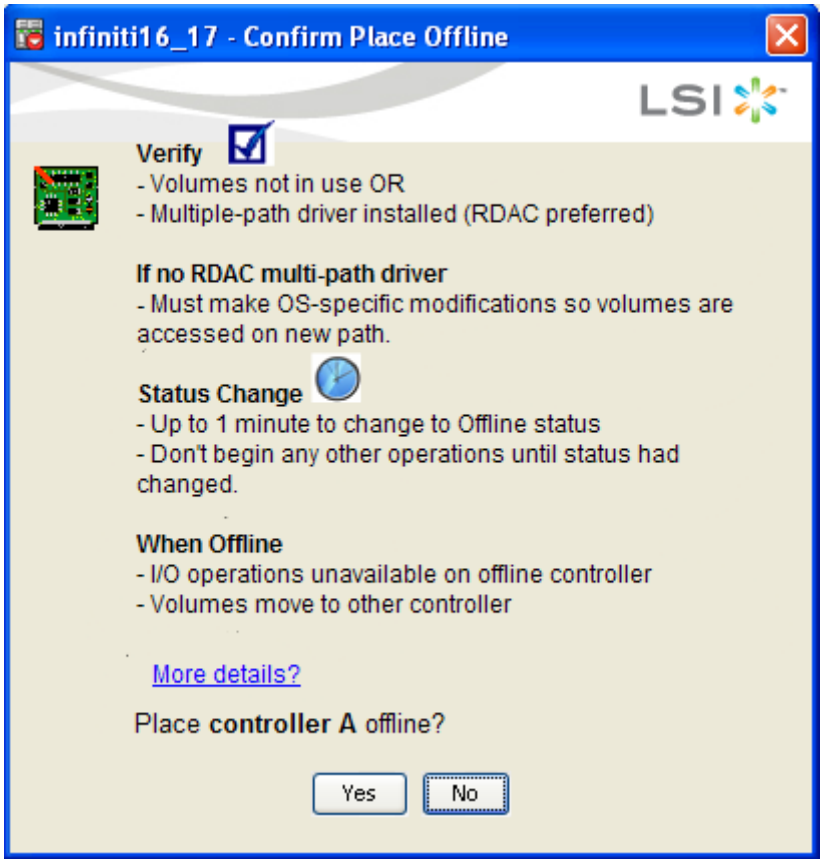
- All on-screen text must be scrutinized to ensure that the fewest number of words are being used to appropriately convey the information needed.

7.17. Messages and Confirmations

- Given the fact that most users don’t pay attention to long messages, a minimalistic approach must be taken relative to messages provided to the user. Instead of using standard prose and paragraphs, the messages must employ a “keyword or idea mapping” approach. This approach must use both graphics and text to convey the necessary information. If needed, a link must be included to provide additional information. For conceptual examples, see http://ideamappingsuccess.com/. For a “before” and “after” example, see the following messages for taking a controller offline.



Current Message



Revised Message (Sample)

- Any destructive operation that will destroy user data must require special acknowledgement by the user. Need to provide a good balance between making the acknowledgement too easy or too cumbersome. Need to ensure that the user doesn't accidentally delete data or important configuration entities.
- In general, it is good practice to dynamically insert the name of the object being acted upon in messages and confirmations to ensure that the user verifies that this is the desired object. This object name must be offset from the rest of the text and highlighted in some fashion. However, care must be taken to not repeatedly display the name in the message or confirmation because it generally distracts from the rest of the information displayed.

7.18. Progress/Feedback Requirements

PAS Note: This is an investigative area with the results being placed into an AAD or specific UX document.

A plan must be developed for determining the proper feedback given to the user during various operations. This plan must state the particular scenario with various time-bound parameters with the appropriate progress/feedback indications.

7.19. Automatic Display Update Requirements

All clients must be notified of system-managed state changes (e.g. a hardware failure) and if one client changes the configuration, other active clients must automatically display the new configuration.

7.20. Accessibility Requirements

PAS Note: Accessibility is an area that needs to be investigated further to determine the exact requirements and impact. The results of the investigation will be included in the AAD and/or a separate UX document.

Both the EC and Element Managers must meet proper accessibility requirements for Web applications. In addition, we need to understand what browser/system specific settings we need to inherit if so desired by the user.

7.21. Customization & Localization

7.21.1. Customization

PAS Note: Need to verify the list below and also see if there are more levels of customization flexibility we need to include. See the next section 7.21.2, for an additional customization scheme.

Customization, or the ability to tailor a software product's appearance to suit the needs of an OEM, is supported for the Enterprise Console and the Element Managers. This means that the customizable elements of the management software consist of the following:

- Client software
    - Nomenclature – e.g., terms like "volume," "drive," "port," etc. can be substituted.
    - Icons
    - Logos
    - About boxes
    - Window titles
- Installers
    - Splash screens
    - Backgrounds
    - Logos
    - Install directory path
    - Individual application selection enabled/disabled

7.21.2. Application Map

While the LSI organization of the user interface will be based on life-cycles, OEM customers may prefer some other categorization of management tasks. As a customization option, not only should we allow an OEM to change the terminology of a category, we should also allow the OEM to change what tasks fall within that category. The 'views' or Launch-in-Context elements will be created and managed by LSI. However, the categorization and labels of the views should be specified in an 'application-map' to specify the layout of the management tasks. The application-map can be a simple XML file with tags to specify categories and the tasks with in those categories.

Additional metadata such as 'user experience level' and keywords about each LiC component can be included in the map to allow an adaptive and flexible user interface. The map must NOT be end-user modifiable, but rather considered part of the OEM Customization parameters of the application. See section 16.2 for additional details on the LiC components and metadata.

7.21.3. Localization

Localization, or the ability to express textual data in a customer's native language, generally applies to output text strings; it is implicit in the software design that input text can be in any character set that can be expressed in Unicode. There are five main areas of output text localization that apply to this strategy:

- Strings that are part of SMI-S responses. These will be localize-able per the CIM standard, on which SMI-S is based.
- Strings displayed by the Enterprise Console and Element Manager. These will be localize-able, following the conventions used in the Google Web Toolkit.

PAS Note: Need to understand if the "partner" framework being considered for the impacts the Enterprise Console localization.

- Strings that are the output of SM-CLP commands. These will be localize-able per the SM-CLP standard. One thing to point out is that the SM-CLP specification states that command line "terms" (e.g., command verbs, etc.) are not subject to translation.
- Strings that are part of a device's proprietary event log. These will always be in English and will not be localize-able.

Supported devices having an embedded SMI-S agent and an embedded Element Manager will be impacted by items (1) and (2) because of the need to store the translated strings on the device. To keep the space requirements to a minimum, the OEM customer will be required to express a language preference as part of placing an order.

7.22. Dashboard/Summary View

Both the Enterprise Console and the Element Managers should have a dashboard/summary view that provides a "quick-look" indication of various aspects of the storage. The groupings (portlets) of information must be organized in such a way to allow the user to customize what information is displayed by re-arranging them, removing them, re-adding them, and minimizing them.

This view can either be considered a life-cycle independent view or could be considered a supporting view of the Monitor life cycle stage since it essentially provides monitoring information.

7.23. Common/Favorites View

PAS Note: Need to get input on the feasibility of this view and the ability to manipulate it.

This view provides the user the ability to add and remove task links of frequently-used or favorite tasks. This view may be initially blank or loaded with a pre-defined set of frequently-used tasks. This view is considered a life-cycle independent view and as such will be included as part of the Navigation strip but not part of a specific life cycle stage.

7.24. User-Defined Folders

The user in both the Enterprise Console and the Elements Managers must have the ability to create user-defined folders for organizational purposes. The user-defined folders must have the following attributes:

- User-defined folder name
- User-defined folder icon – this may not be feasible but thought it would be a nice user option to allow them to select their own icons so long as they met icon size requirements.

- User must be able to move devices into the folders through a move/copy operation (via appropriate menu item or drag-and-drop operation).

8. Enterprise Console (EC) Definition

8.1. Overview/General Functionality

The Enterprise Console inherits the basic functions associated with the SANtricity Enterprise Management Window (EMW) but also provides additional functionality that allows the user to obtain information about multiple devices as well as perform operations on more than one device at a time.

Feedback received indicates that being able to manage multiple devices is critical to users and support personnel from a productivity standpoint. The effort involved in gathering information or performing certain operations is very cumbersome and time-consuming if the user is required to repeat this on each individual device. Mechanisms must be put in place to ease the burden on the user.

The EC is LSI-developed software but is optional. That is, it is not as tightly coupled to the Element Managers as the current EMW/AMW for External RAID. Many OEM customers plan to develop their own full functioned consoles into which our various Element Managers will be provided as plug-in, launch-in-context components. The EC is available for OEM customers that…

- Are ***not*** providing their own Enterprise framework and have a need to supply their end users with a centralized console to manage our LSI devices, or
- Are providing their own Enterprise framework but the robustness of these frameworks may exceed or not match the customers' needs or price-sensitivity, especially in the lower price bands. Our Enterprise Console is an excellent, less-expensive alternative for our OEMs to offer.

Note: We want to ensure that our Enterprise Console implementation doesn't significantly duplicate an OEM console's functionality or impinge in the market space where their full-featured consoles reside, so extensive device management is still relegated to the individual Element Managers. With that in mind, the Storage Console provides the following basic functions:

PAS Note: Not sure how the Tek Tools Profiler plays into the Amelia strategy. Need further investigation.

Basic Function	Description	For Further Details, See Section…
Discovery/Add	Automatically discover or manually add LSI devices	8.6 and 8.7
Removal	Removing a device from the console view.	8.8
Launching	Launching a corresponding Element Manager to manage a specific LSI device.	8.9
Device Display	Providing appropriate views of the discovered devices and allow for the manipulation of those views.	8.10
Device Detailed Property Information	Displaying a summary of the most critical attributes of a device to allow the user to ascertain whether additional actions are needed using either the EC or the Element Manager.	8.11

Basic Function	Description	For Further Details, See Section…
Report/Query Capability	Querying certain attributes of the discovered devices in the console allows the user to quickly create customized reports/queries to gather information needed for follow-on actions.	8.12
Alert and Event Log Configuration	Configuring alerts for one or more like-type devices and to configure certain event log parameters.	8.13
Health Monitor	Obtaining both a coarse-grained device status as well as a summary list of any non-optimal conditions per device (note that troubleshooting and problem resolution still takes place using the appropriate Element Manager functions).	8.14
Single-device Operations	Performing a *limited* set of single device operations. These operations are duplicated from the Element Manager but provide a level of convenience for a user that has the Enterprise Console.	8.15
Multi-device “Mass” Operations	Performing a *limited* set of multi-device operations. These operations can act upon more than one device at a time (normally like-type devices); this is especially important for certain support and upgrade tasks.	8.16
CLP Script File Load/Execution	Loading and executing a CLP script file is provided. The ability to edit the script within the Enterprise Console is not provided (i.e. there is not a script editor pane similar to the EMW).	See CLI section

8.2. Deployment Method

Given that the EC is a browser-based application, there are two options for its deployment (see Section 11.4 for more details).

Note: The first option listed below is the only choice for DAS products and lower-end external products with limited resources.

- Installed on a “management server” – the executable code would be installed on a management server via CD or downloaded through an external web site. The management server must allow multiple clients to connect into it to gain access to the managed devices.
- Embedded on the controllers.

PAS Note: The second deployment option, embedded on the controllers, needs further investigation and definition (see section 11.4.1.1).

8.3. Database Requirement

Because of the requirements for different reporting/querying capabilities within the EC, a light-weight database must be included as part of the EC. Because the EC will have a database, there needs to be a mechanism to ensure availability. A redundant scheme must be developed.

8.4. Data Persistence

Because the EC will have multiple users accessing it, there must be provisions for persistent storage to house individual user preferences. Each user that logs into the EC must be able to have his/her own unique views and preferences. See section 17.2 for more details.

PAS Note: Seems like the best approach is to provide for some kind of user identification (without password) so that we can store information appropriately instead of relying on cookies for this that are only applicable to the particular client.

See section X for more details.

8.5. User Interface Framework Definition & Overall Behavior

PAS Note: Further investigation needs to be done here relative to working with a partner for the EC framework and what the proper UI framework must be to effectively show the necessary views and tasks. The EC framework must consider the life cycle approach defined for the Element Managers.

8.6. Automatic Discovery

PAS Note: This section needs to be investigated assuming we use the “partner” framework and underlying support. Once the investigation has concluded, then appropriate information must be included in this specification and the AAD. The information below provides more definition related to the user interface.

The automatic discovery of the approved devices must match the behavior in the current SANtricity EMW as follows:

- An automatic discovery takes place each time the EC is launched by the user.
- A progress indicator must be shown during the automatic discovery process.

PAS Note: Are we supporting both in-band and out-of-band management?

8.7. Manual Add

PAS Note: This section needs to be investigated assuming we use the “partner” framework and underlying support. Once the investigation has concluded, then appropriate information must be included in this specification and the AAD. The information below provides more definition related to the user interface.

The manual addition of devices into the EC must have the following behavior:

- A manual discovery method must be provided for those devices not able to be discovered via the automatic method.
- Status indication must be shown during a manual add process.
- If desired, a newly discovered device (via the manual add process) can be placed into a user-defined organization folder (the organization folder can either be an existing one or a new one specified by the user during the manual add process).

8.8. Removing a Device

The user can remove a device from their EC views. This removal process must only affect the view for that particular user; other clients accessing the EC must not be affected if they have the same devices in their EC.

PAS Note: If a device becomes obsolete either through a decommissioning or removal from the SAN, should there be a way to remove it from all EC users' views?

8.9. Launching an Element Manager

8.9.1. Launching the Entire Element Manager

The user has the option of launching the element manager into a separate window or embedded within the EC framework. The default is to open the EM in a separate window.

- If the EM is launched in a separate window, then multiple EMs can be open at one time.
- If the EM is embedded in the EC framework, then only one EM can be opened at a time. As one EM is loaded, then the other will close.

8.9.2. Launching an Individual LiC Component

PAS Note: Need to determine how to display individual LiC Components from within the EC (embedded within EC framework or separate window). I think it will depend to some extent on the individual LiC component. Also, need to determine how many different LiC components can be launched at the same time either from the same device or for different devices.

8.10. Device Display

PAS Note: It would be a nice feature to allow the user to group or place discovered devices by geographic location. Don't know if this would be best accomplished through user-defined folders (see below) or some other mechanism – would like to get input from reviewers.

- By default, "like-type" devices must be grouped together in the viewable display of discovered devices.
- There must be a way for the user to quickly determine the number of each type of discovered device as well as a tally of the different device statuses.
- The user must be provided with several different display options to view the discovered devices including…

Note: This would be similar to the different view options available within Windows Explorer.

  - Sortable list display with applicable summary column headings
  - Graphical display with summary information
  - Detailed property information (see next section for details).

- Users must have the option to create user-defined organization folders to group devices in whatever manner they deem appropriate. The folders in the Enterprise Console would replace any default organization (that is, the devices would be moved into the folders rather than just copied).

8.11. Device Detailed Property Information

One of the deficiencies today in the current EMW is that there is very little information displayed for each discovered device. Because of this, the user must always launch the associated Element Manager to obtain this information. In the new Enterprise Console (EC), critical information about each device must be provided. At least the following information must be supplied for each discovered Amelia-based device:

Note: Details of all of the information required will be documented in the Aspect Architecture Definition (AAD) document. Note that the information may vary depending on the specific device discovered. It needs to be assured that only the most critical information is provided in the EC for each device so as to not overload the user with unnecessary information. The information tiering approach outlined in section 7.11 must be used.

- Base Storage Inventory
    - Total capacity
    - Total free capacity
    - Free capacity per storage pool (assuming there is more than one storage pool).
    - Number of each type of logical element
- Base Hardware Inventory
    - Number of controllers and model number
    - Number of each type of drive with model numbers and capacities (for example, Fibre ST-2105, 700 GB – 15 drives and SATA ST-3105, 300 GB – 15 drives).
    - Current firmware versions of all different components (this would just be a line per each type of component not one for every component).
    - Number of hosts attached and associated HBAs
    - HBA utilization – i.e. determine if a particular HBA has been off the fabric for a period of time.
- Basic Monitoring (also, see section 8.14).
    - Problem Summary Synopsis - one-line indications of any non-optimal conditions including when a storage system is running out of free capacity. These summary indications are duplicates of what is found in the Recovery Guru.
    - Indication if a system may require an upgrade.
    - Basic performance numbers
    - Whether any alerts have been set.

PAS Note: DAS and SVM information are TBD.

8.12. Report/Query Capability

PAS Note: Should this be a chargeable feature?

The ability to obtain/locate information across a set of devices is a critical element needed by the end user when attempting to determine additional actions on the various devices under his/her control. The following section outlines the type of information and capabilities that are required.

Note: All reports and output provided by the application must be in XML format and be able to be exported into a spreadsheet compatible format.

Note: These reports and queries must be able to be obtained on one, some, or all devices of the same type.

Note: This capability should be available on a single device as part of the Monitoring Stage of the Element Manager.

- The pre-defined reports and query functions must be based on the detailed property information outlined in the previous section as well as the health status information and any other important information that is displayed in the Enterprise Console.
- The user must be able to display pre-defined reports.
- The user must be able to save the reports.

- The pre-defined reports must include two modes (1) technical and (2) presentation. The technical mode is more for the storage administrator while the presentation mode is for presenting storage statistics to management personnel.
- The user must be able to get current as well as historical/trending information.
- The user must be able to create custom queries/searches and run them to obtain consolidated information on the discovered devices.
- The queries must be able to be saved by the user so that they can be re-run.

Note: The query function within the external SVM should be used as a starting point for the types of queries that can be built. Care must be taken to provide a useful, but limited set of query parameters.

8.13. Alert & Event Log Configuration

Note: The EC needs to be able to inherit any individual alert/event log settings that may have been configured via an individual Element Manager. This covers the use case of where a user initially just uses a solution without an EC (i.e. just using the Element Manager for each device) and then later purchases the EC to manage multiple devices. Because the alerts originate and are stored on the device itself, this shouldn't be an issue. See section 11.9.1 for details.

- The user must be able to perform the alert/event log configuration on one, some, or all discovered devices (EC requirement only, not applicable for the individual Element Managers).
- Must provide a consolidated screen to show all alerts and the corresponding devices and recipients and be able to manage the alerts from one screen. Currently there is no way for the user to ascertain what alerts are set without having to look at each individual device (this is very cumbersome). (EC requirement only; not needed for the individual Element Managers).
- The user must be able to configure alerts for email.
- For email alerts, the user must be able to enter more than one email recipient address at a time using a common delimiter.
- The user must be able to configure SNMP traps.
- The user must be able to indicate that they want event log entries to also be directed to the system log (i.e. Syslog).
- The user must be able to modify the severity levels for the different logged events (note that this option would only be available on like-type devices as the event types will differ per device).

PAS Note: Need to define the different severity levels.

PAS Note: DAS has an option to also display a dialog when an event occurs – do we want to make this be a converged option/behavior or drop this option?

8.14. Health Monitor

The health of each discovered device must be shown as follows:

- A coarse-grain status (optimal, needs attention, fixing, unresponsive, unidentified device, and alarm sounding). The user must be able to obtain this information on one, some, or all devices.
- A consolidated view must be provided to report on which discovered devices are in each of the statuses mentioned above.
- In addition to the coarse-grain status, the user must be able to

- obtain a summary list of problems occurring with a selected device with the ability to launch the Recovery Guru to remedy the problems.
- obtain a summary of any Operations in progress on the device.

8.15. Single Device Operations

PAS Note: This only covers the operations for External Storage. Need to determine actions for DAS and SVM.

The following single-device functions are available in the Enterprise Console. These functions are duplicates of functions available within the Element Manager and will be presented as Launch-in-context components. These functions are provided in the Enterprise Console as (1) a convenience to the end user so that he/she doesn't have to launch a separate application to complete the function, and (2) functions that seem natural to be offered if a user has purchased a centralized Enterprise Console.

- **Locate** – the ability to cause a visual indication (such as an LED) on the device to be active to aid in locating a device.
- **Rename** – the ability to provide a unique user-name to the device to aid in identification.
- **Comment** – the ability to provide a descriptive comment about the device.
- **Configure Alerts** – the ability to configure an alert for an individual device (see section 8.13 for details).
- **Save/Load Configuration** – the ability to save a configuration for replication and recovery purposes and to re-load that configuration on the same device or a device with identical hardware components.
- **Basic Storage Provisioning** –the ability for the user to provide a basic set of input parameters to create a *usable* configuration on the device.

PAS Note: Further details need to be specified as to the extent of provisioning options provided to the user and what level of configuration would be created.

- **Display of Device Profile** – the ability to display the profile of a particular device.
- **Display of Device Summary & Details** – see previous sections (8.10 and 8.11) for details.
- **Display of Device Event Log** – the ability to display the event log for a particular device.
- **Launch of Recovery Guru** – the ability to launch the Recovery Guru for a particular device.

PAS Note: Need to determine if this would just launch the Recovery Guru or the entire EM application. The reason for launching the entire EM would be if there are views/actions the user needs to recover from the problem.

- **Gathering of Support Information** – the ability to gather support information for a particular device.
- **Firmware Upgrade** – the ability to upgrade firmware on the different components (not just the controllers) that comprise a particular device.
- **Report Generation** – the ability to generate and view pre-defined reports such as profile information, storage utilization, etc (see section 8.12).

8.16. Multi-Device "Mass" Operations

Note: Some of these multi-device operations may be offered by just providing a mechanism for a customer to load and execute a CLP script file.

Note: As part of the software solution, we must provide some canned script files for common operations. The user can use these canned scripts for educational purposes to understand the syntax as well as using them as the base for actual executable scripts.

Because the Enterprise Console is providing a view of all managed devices, it makes sense to provide certain operations that act upon more than one device at a time. This is especially critical for support and upgrade scenarios.

Note: Most of these operations would only act upon like-type devices (that is, external RAID devices only, internal RAID devices only, etc.) but not all devices.

- **Configure Alerts/Log Events** – the ability to configure an alert for more than one device (see section 8.13 for details).
- **Save Configuration** – the ability to save the configuration of more than one *like-type* device. Note that a standard filename would be used that includes the name of the device.
- **Load Configuration** – the ability to load a configuration on more than one *like-type* devices with identical hardware components.
- **Gathering of Support Information** – the ability to gather support information for more than one device (this could be homogenous or heterogeneous devices).
- **Firmware Upgrade** – the ability to upgrade firmware on more than one device. Note that the devices must be the same type and at the same firmware level.
- **Report/Query Generation** – the ability to generate and view pre-defined reports such as profile information, storage utilization, etc from information gathered from more than one device (would need to determine if this would work for heterogeneous devices).
- **Remote Replication Activities** – allow the user to perform remote replication tasks (mirroring, volume copy, volume migration) from one storage device to another. The main option would be to replicate all of the source entities from one device to another but also give the option to select the specific source entities.
- PAS Note: Providing remote replication services as part of the EC needs to be investigated further. The ability to do this from the EC seems intuitive and at least one storage competitor offers this feature.

8.16.1. Commands Supporting Multi-Device Operations

Any command that supports a multi-device operation must be structured to pass down a parameterized list of devices on which to perform the operation rather than a series of individual commands. By doing this, we can ensure the desired transactional behavior and recovery methods if not all devices in the list complete the command. These same multi-device commands must be used for single device operations as well with a parameterized list of just one device.

8.17. Exiting the Enterprise Console

The user is given the ability to exit the Enterprise Console.

PAS Note: Do Element Managers continue to display since they are not as closely coupled to the Enterprise Console as before or do they all close?

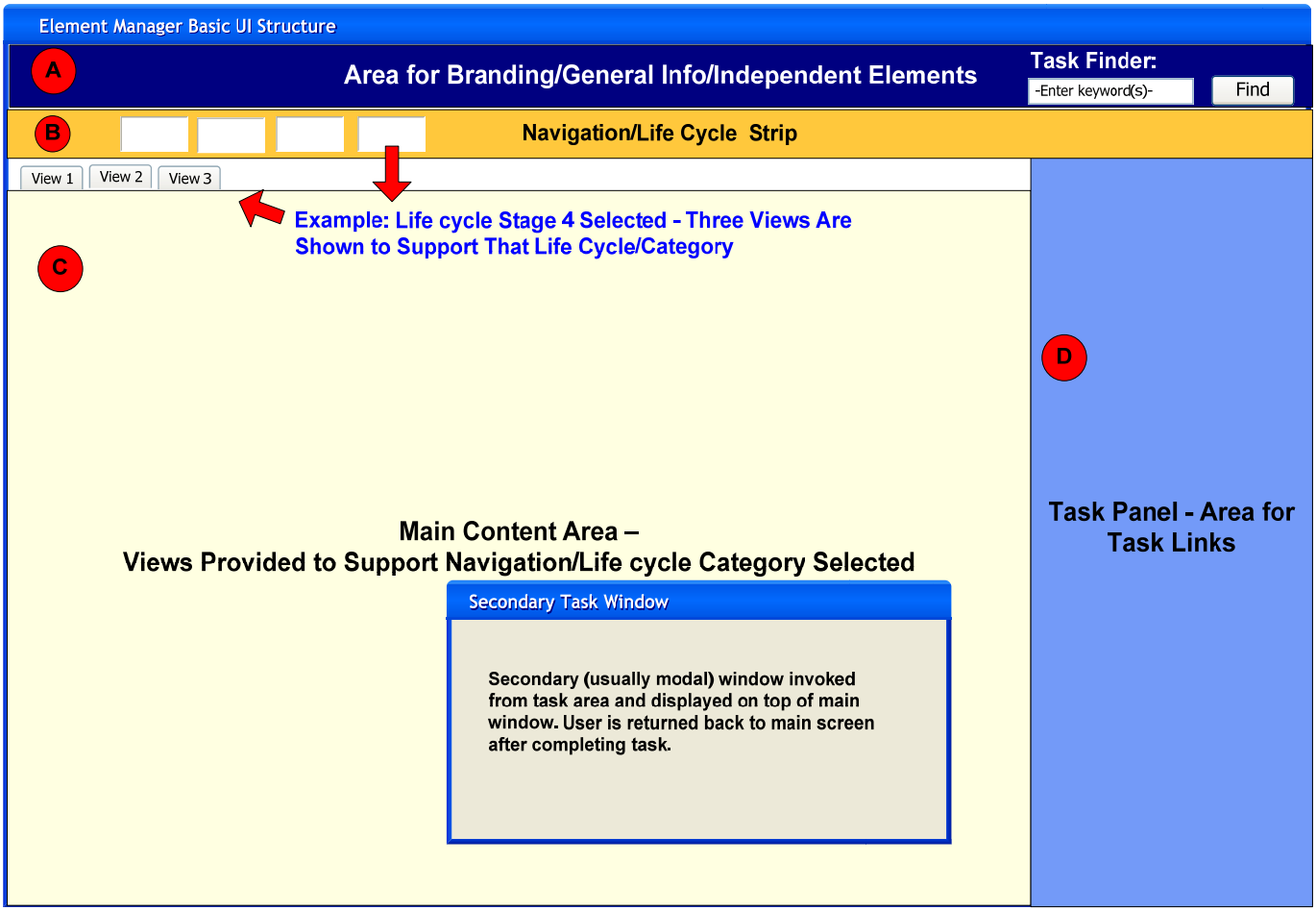
9. Element Manager (EM) Definition

9.1. User Interface Framework Definition & Overall Behavior

The overall UI Framework for single array management within the Element Manager consists of the following areas:

PAS Note: This overall framework may also work for the Enterprise Console (EC). Further investigation needs to occur to determine the EM framework's applicability to the EC.

- A. Branding/General Info Area
- B. Navigation/Life Cycle Strip
- C. Main Content Area
- D. Task Panel



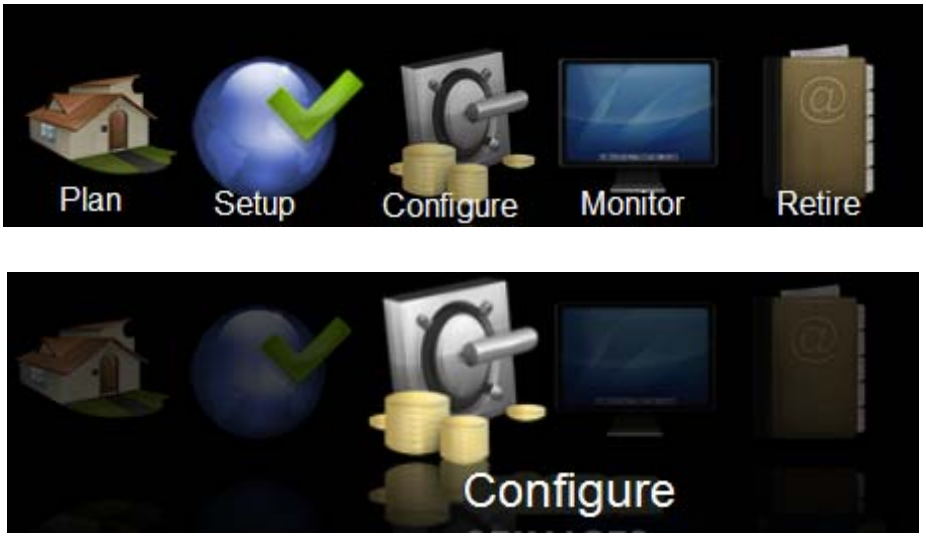
Element Manager Base Framework

9.1.1. A. Branding/General Information Area

The branding/general information area contains the following elements:

- Name of the product
- Company logo
- Any life cycle-independent task/information
    - Task-finder search function
    - Help
    - Etc.

9.1.2. B. Navigation/Life cycle Strip



Navigation/Life Cycle Strip (Sample "Filler" Icons)

The Navigation strip is the main mechanism by which the user navigates to the appropriate life cycle stage to complete tasks within the user interface. The navigation strip is organized by the life cycle stages mentioned in section 4. The tasks/views supporting each life cycle stage are described throughout section 9.

- When a stage is selected in the navigation strip, the main content area would display the appropriate view(s) to support the tasks required for that life cycle stage. In addition, the task panel would provide the appropriate tasks based on the view selected in the main content area and the specific object selected.
- The navigation strip can be docked either horizontally above the main content/task panel area (as shown) or vertically along the left side of main content area. The default location is horizontal but an OEM has the option to choose the default. In addition, the end user has the option to choose the location of the navigation strip.
- Normally, all life cycle stages and their associated software components are loaded when the Element Manager is loaded. However, the user will have the option of defining which stages should be loaded on a subsequent load of the application. Because of this, the navigation strip must provide a visual indication of when a stage is not loaded. Also, if the user selects a life cycle stage and it is currently not loaded, then it will be loaded in response to a user request. On-demand loading provides several advantages to the user (1) only those tasks that are needed for the current life cycle stage that the product is in is included – this allows for a much cleaner interface, and (2) the user may see some performance improvement by not loading all life cycle stages and their corresponding views/objects.

9.1.3. C. Main Content/Views Area

- This area provides the views and objects necessary for the user to complete the tasks for a given life cycle stage displayed in the Navigation strip.
- A life cycle stage may have either one supporting view or multiple supporting views. The main content area is the designated location for showing these views.
- An individual view should be included to (1) provide proper consolidation/segmentation of various similar objects, and (2) to emphasize an important function/aspect for the particular life cycle stage.

  - For example, under the Configure lifecycle stage, you could have separate views for (1) Base Storage, (2) Copy Services, (3) User Role Management, (4) Physical Configuration, etc. For the Monitor lifecycle stage, you could have separate views for (1) Performance Monitoring, (2) Recover, (3) Operations In Progress, (4) Upgrade, etc.
- All Elements Managers, from a convergence standpoint, must use standard views for each life cycle stage. In addition to the standard views, views specific for the particular application within a particular life cycle stage can be used.
- The main content area can consist of views that are read-only as well as views that support interaction (i.e. selecting objects and completing tasks).
- The main content area can be resized.

9.1.4. D. Task Panel Area

PAS Note: Some concern has been raised about the screen real estate that is taken up by the task panel. Alternate approaches may need to be taken to reduce the amount of space required. Also, since the task panel essentially emulates a right-mouse menu item, the “global” view that a pull-down menu structure provides is missing (that is, with a pull-down menu, a user can peruse the functions within the application without having to click on each object). Therefore, a pull-down menu for each life cycle stage along with the applicable right-mouse menus may be a viable alternative. Also, there may be some new innovative approaches for menu item selection to be considered. The following sections describe the behavior assuming we are using the task panel.

The task panel contains a list of general and specific tasks based on the view that is shown in the main content area and the object selected.

- General tasks are those tasks where a specific object does not need to be selected (an example, would be a “Create” task).
- Specific tasks would be displayed as a user selects a particular object in the view.
- The user must be allowed to select more than one object at a time (see section 7.7).
- The user must be able to obtain further information about each task.
- The task panel area can be resized or collapsed/hidden.

9.1.5. Task Finder Search Function

One of the downsides of just providing contextual tasks based on the object or stage/view selected (either through the task panel or through a contextual right-mouse menu) is that the user gets to see only the tasks associated with the object (i.e. a “keyhole” user interface). Unfortunately, if users want to get a global understanding of the product’s overall functions, they are forced to go to every object and every stage/view.

Also, past experience has shown us that no matter how we organize the user interface, users can have a different mental model of where tasks should reside and therefore may have difficulty in locating the stage, view, and object that they should select to start a task.

The task finder essentially flattens out the organization and provides a convenient way for users to just “complete tasks”. A search function toolbar is prevalent in websites and is a familiar and often used tool for end users.

The task finder allows a user to enter a keyword or words and then the associated tasks (results) containing that keyword or an equivalent is provided. The keywords and equivalents are included as part of the metadata associated with each task provided in the application. The keywords provided in the metadata need to accommodate different terms and synonyms (e.g. LUN versus volume)). See section 7.21 for details.

Note: In addition, the user can also input “All” which will essentially return a “site map” of all supported functions organized by each life cycle stage.

Once the user selects a task from the results, then two different paths are possible:

Note: If a particular software module is not loaded at the time, then it would be loaded in order to satisfy the task requested by the user.

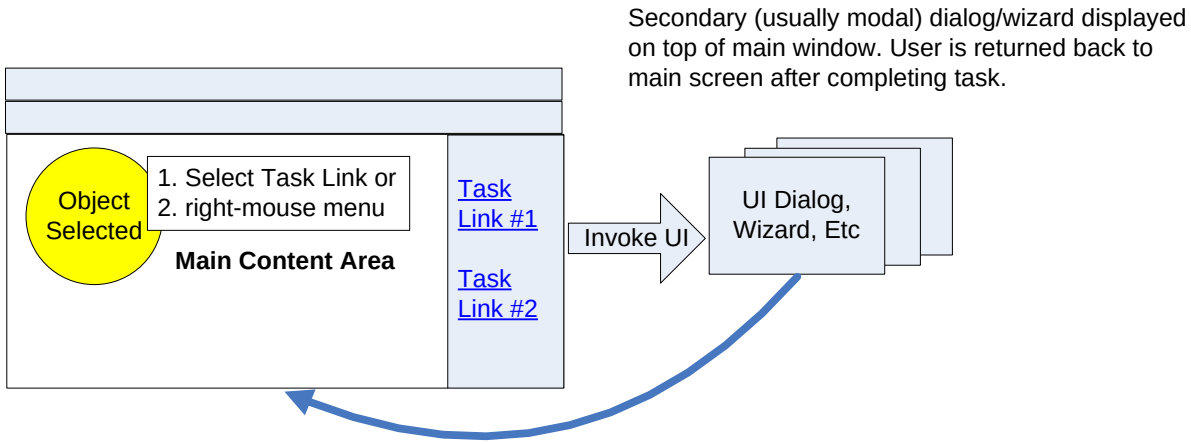
1. If the user selects a task that does NOT require a specific object to be designated, then the standard UI is presented to complete the task.
2. If the user selects a task that does require a specific object to be designated, then an “object selection” dialog is displayed. Once the appropriate object is selected, then the standard UI to complete the task is presented.

Upon completion of the task (regardless of path 1 or 2), the user is given the option to return to whatever view was currently shown or to be taken to the view that best shows the results of the action taken through the task selection.

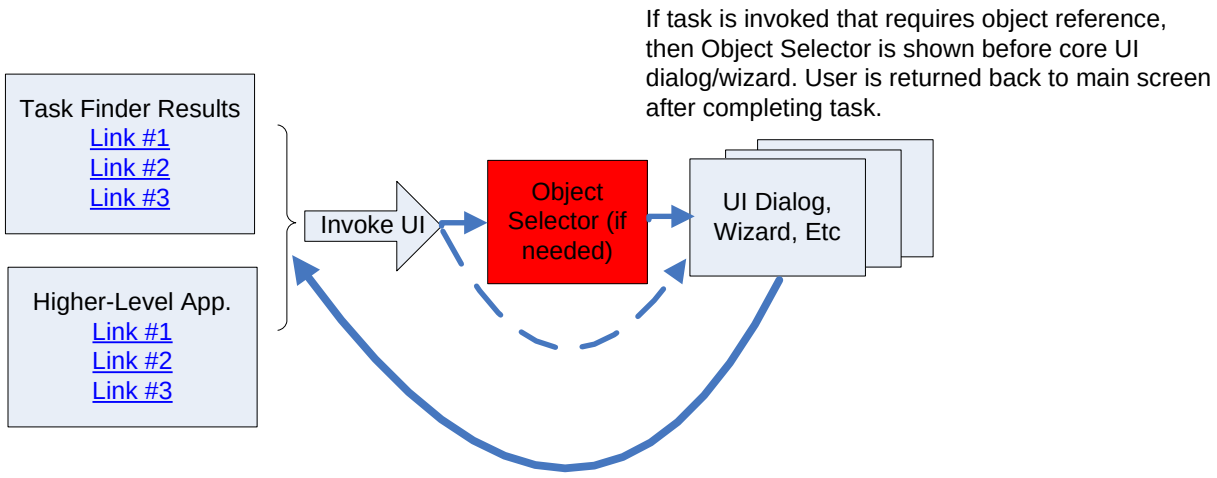
The Task Finder is always available regardless of the current View shown.

9.1.6. Completing Tasks

The Enterprise Console and Element Managers utilize a hybrid model for the user interaction of the management of the LSI devices. The hybrid model uses both object-based (primary method) and task-based (secondary) launch points. Both methods ensure that we address the two major ways in which users complete activities associated with device management



Primary Task Completion Method



Secondary Task Completion Method

9.1.6.1. Primary Method (Object-Task)

The primary method for completing tasks is for the user to use the following sequence:

1. The user selects a life cycle stage in the navigation strip. The main content area displays the appropriate views and objects for the selected stage.
2. The user selects the desired view. The view is displayed with the appropriate objects. The task panel displays the appropriate general tasks and the specific tasks based on the view and the object selected.
3. The user selects the desired task or alternatively can invoke a contextual right-mouse menu and select the desired task. The supporting user interface (dialog, wizard, etc.) is presented on top of the main screen allowing the user to complete the operation. Upon completion of the operation, the user is returned back to the main screen/view from which the task was originally invoked.

9.1.6.2. Secondary Methods (Task-Object)

The secondary method for completing tasks is for the user to use the following sequence:

1. The user selects a task link either (1) from the results of the task finder (see definition in section 9.1.5) or (2) in a higher-level application that uses launch-in-context components from the LSI management application.
2. If the task requires an object reference to complete the task, then an "object selector" dialog is presented first to allow the user to indicate which object to act upon; otherwise, the supporting user interface is presented (see next step).
3. The supporting user interface (dialog, wizard, etc.) is presented on top of the main screen allowing the user to complete the operation. Upon completion of the operation, the user is returned back to the main screen/view from which the task was originally invoked (either the LSI management application or the higher-level application).

9.2. Discovery

PAS Note: This section is TBD pending the answers to the following questions. The main contributor to this section is Scott Hubbard.

- If a user does not have the Enterprise Console, then we need to defined the mechanism for the individual device to be discovered by the user (i.e. what is the URL that they use to contact the device and how is that communicated to the user).
- Also, for some of the things we would like to do, it is advantageous for us to be able to bring up the embedded application without having attached drives. This would allow the user to perform basic, but critical, operations before the drives are attached (for example, being able to be guided in proper cabling of their storage system).

9.3. Deployment Method

PAS Note: This may already been covered sufficiently in the Architecture Section.

The Element Managers (EMs) are web-based applications accessible through a browser. To support this deployment method without sacrificing the client "richness" that SANtricity and MegaRAID users have become accustomed to, the Element Managers are written in Java and compiled into JavaScript using the Google Web Toolkit (GWT). Two distinct deployment scenarios are possible:

1. Device-served. This scenario applies to External RAID and the SVM appliance which has sufficient resources in the device to be able to store the JavaScript code in non-volatile media. Such media is not necessarily disk, depending on factors such as size of application and need to run the EM against a diskless controller module. Under this scenario, three beneficial aspects emerge: (1) a factory-shipped storage system comes "ready-to-manage," i.e., no software installation is required to enable the management user interface; (2) a new release of storage array firmware implicitly includes the management application; and (3) in an environment of multiple storage systems at different firmware versions, there is no possibility of invoking the wrong management application for a particular storage system.
2. Host-served by "application-embedded" web server. This scenario applies to Direct Attach Storage (DAS), which is positioned in the lower-end server markets. In such an environment, the user is apt to not have a web server and to lack knowledge of setting one up. This is addressed by incorporating a light-weight web server into the application, to where its existence is transparent to the customer. All files and binary executables comprising the web server are loaded at application installation time, and all necessary setup is accomplished at that time. In cases where the customer does have an already-existing web server, some provision for co-existence of the two (e.g., port configurability)

would be needed as part of installation. Once the light-weight web server is established, the operation of the application is the same, or nearly the same, as for the device-served scenario.

PAS Note: Not sure we are still considering the incorporation of a light-weight web server into the application. This needs further discussion.

Another aspect of application deployment that fits nicely with the browser-based concept is the idea of deploying the application to a hand-held device such as a mobile cell phone. The increasing support in such devices for web browsers and Javascript makes it realistic to consider serving the application (or portions of it) to such a client, albeit in a “reduced function” form more suitable for a less capable device (for example, the Monitor life cycle stage and supporting views/tasks may be ideal for a mobile device).

9.4. Plan Definition

PAS Note: This section is still TBD. Need to discuss what types of planning or storage allocation aspects would be beneficial to include as part of the Element Manager. The main contributor to this section is Scott Hubbard. Some ideas are included below.

- Storage allocation tools
- Quick learning aspects including demos

9.5. Setup Definition

9.5.1. Essential Setup Tasks/Quick Successes

The initial setup function must provide the following tasks and information.

- An initial View that is very graphical and depicts the current settings (with many settings labeled as “not set”) must be shown first to the user. Then the user is provided with various tasks to configure/change these settings and make sure the system is optimal. Where feasible, the initial setup must be presented as a sequence of related tasks using a wizard rather than discrete steps.
- Tasks that are essential to communicating with the storage system and getting it ready for configuration.
- Tasks that would provide “quick successes” for the user. This is very important to set a favorable initial usability perception for the user. Tasks that are simple in nature and easy for the user to understand must be included.
- Information depicting current critical settings with the ability to configure/change the settings.
- Provide a “What’s Next? Checklist of common tasks that the user will perform using the rest of the management software. It is critical to ensure that the user understands the tasks that need to be done and the proper order. This can either be an interactive checklist or just information. This must be graphical in nature.

Note: Tasks that are repetitive in nature (e.g. those that the user will perform many times such as logical configuration tasks) should NOT be included.

9.5.2. Cable Configurator/Checker/Viewer

PAS Note: Depending on the connectivity requirements to communicate with the storage system, the configurator function may need to be a stand-alone tool.

A task that continues to be troublesome for end users is the proper back-end cabling of the various enclosures that comprise a storage system. This function will aid the user in determining the proper cabling for their enclosure. The cable configurator must have the following attributes:

- User is asked for the minimum input required to determine the proper cabling (some known input parameters are (1) controller enclosure model number, (2) drive enclosure model number(s), and (3) how many of each type of drive enclosure.

PAS Note: It is assumed that if we can communicate with the controller enclosure that we won't need to ask the user for the controller model number (i.e. we can get that information implicitly).

- The output must…
    - Indicate to the user how many cables are needed
    - Provide a graphical representation of the back-end cabling. Methods must be in place to differentiate the different cables required.
    - Allow the cable configuration to be saved into a format that can be printed.
    - Allow the cable configuration to be printed directly from the user interface.
    - Must inform the user that once the configuration has been determined, the controller must be powered off before cabling the drive enclosures.

Once the drive enclosures have been cabled, then the user must be able to (1) invoke a cable checker that validates the cabling and guides the user through any troubleshooting, and (2) view a graphical depiction of the back-end cabling.

9.5.3. Compatibility Checker

Another troublesome area during initial setup is verifying that all of the connected components are compatible with each other.

- At a minimum, the management software must provide a compatibility validation tool that walks the user through the various tasks needed to ensure proper compatibility (somewhat of an online tutorial without much built-in intelligence) and provides a link to a compatibility web site that provides compatibility information.
- Ideally, the compatibility checker would ascertain the various connected components through a discovery/scanning process or by prompting the user and then feed that into the web site to determining if compatibility has been attained.

9.6. Configure – Block Level Definition

9.6.1. Storage Allocation

Users need to be provided with a keen understanding of the amount of storage that is allocated for the various creation/storage provisioning operations that take place. This information may need to be provided in the Planning, Configure, and Monitor stages. Various displays and supporting user interfaces need to be developed to address this area.

9.6.2. Storage Pools & Hot Spares

The Storage Pool is the fundamental storage building block from which host-addressable entities are created. Storage from the pool is used for base entity creation as well as for storage for supporting entities such as point-in-time (PiT) images and temporary volumes. Hot spare drives provided an additional level of protection above and beyond the pool's RAID protection.

Users generally are less concerned with storage pool and hot spare allocation and more focused (at least from the block-level) perspective on provisioning volumes into appropriate entities. With that in mind the principles for storage pool and hot spare creation are as follows

Note: Storage pool and hot spare creation and exposure will fundamentally change with the introduction of CRUSH. See section 26 for further details.

- The notion of storage pools and hot spare drives as an object that is manipulated by the end user is, where possible, abstracted as much as possible from the end user. This can be accomplished through the following methods:

Note: There is probably a combination of these options that would suffice.

  - Pre-configure, at the factory, a storage pool (and hot spare) configuration that is applicable for the storage system's drive configuration. This configuration should be optimized for redundancy, performance, and cost. Users have the option of using this storage pool/hot spare configuration or making modifications as necessary.
  - Auto-configuration (system detected) – instead of doing pre-configuration at the factory, the storage management software/firmware could detect and analyze the current drives available for pool and hot spare configuration and present a default configuration to the user. The user would have the option of accepting the configuration or declining it.
  - Auto-configuration (user-driven) – give the user the ability for the user to allocate the available drives into appropriate pools and hot spare drives. This process would create the necessary pools and hot spare drives based on limited user input. Note that in the past, we've provided an auto-configuration that provided configuration of volumes and their underlying pools (volume groups). The usage of this current implementation has been sparsely used because it doesn't provide the level of flexibility needed. Plus, we are defining multiple volume creation in such a way that will provide the level of flexibility required by the user (see subsequent section for details). Therefore, it makes sense that auto-configuration is focused on drive allocation through the creation of appropriate storage pools/hot spare drives and not volumes and their variants.
  - Manual pool creation (user-driven) – give the user the option to manually configure individual storage pools.
  - Manual hot spare creation (user-driven) – give the user the option of manually specifying how many drives to allocate as hot spare drives.

- The input required from the user for pool creation and the choices presented to the user will be based on user-driven aspects (that is, most of the underlying settings such as RAID level, etc. will be abstracted from the user). These parameters are:
  - Application – i.e. what type of application is the pool going to be used for? Based on the user selection, this will drive underlying pool parameters.

PAS Note: The applications are being investigated. The preliminary list includes the following applications (1) Oracle, (2) MS Exchange, (3) MS SQL Server, (4) Backup/VTL, (5) Generic DB, (6) Generic Email, (7) File System (Generic Storage), (8) SRA/SRM VMware (SRA = Site Recovery Adapter, SRM = Site Recovery Manager), (9) Citrix/Hyper V/VmWare/Virtual Iron and (10) SANboot Volume

PAS Note: Application type also drives certain volume level settings. It will need to be determined how this is handled at the volume level. Especially given the use case of where a user sets up a pool for a certain application but then wants to create a volume from the pool that is targeted towards another application.

  - Pool candidates are presented with various QoS tradeoff parameters. These QoS tradeoff parameters are based on the composition of the storage pool itself (drive type, number of drives, enclosure protection, etc.). Each trade-off parameter for a given pool candidate has a "rating" attached to it. These ratings are presented in such a way to help the user to select the appropriate pool candidate.
  - Once the user chooses the application and the pool candidates, these selections will drive the underlying parameters (RAID level, segment size, etc.).

Note: The rating notion that comes to mind is the Consumer Reports rating system.

- The list of tradeoffs parameters are as follows.
  - Performance
  - Redundancy Level
  - Cost of Redundancy
  - TBD.

- The ability to select specific drives for a storage pools or hot spare drives is no longer supported. Not only is this becoming more difficult on higher-end, large-drive systems but will become an obsolete concept with future technologies such as CRUSH.

Note: A transactional model needs to be implemented to ensure the successful completion of multi-pool creation.

PAS Note: The DAS product line offers the concept of a dedicated hot spare drive for a particular storage pool. Need to decide if we want to inherit this behavior for External storage as well.

9.6.3. Volumes

The creation of volumes must have the following characteristics:

- The user must be allowed to create more than one volume at time with the same attributes (QoS, capacity, etc.).
- The user must be allowed to create more than one volume at a time with different attributes.
- The attributes that the user must provide for volume creation will depend to some extent on the pool that is chosen. The pool itself will already have specific QoS characteristics. Still need to determine whether additional user-driven choices need to be presented to the user (see the previous section).
- Users need to be provided with a better understanding of “Thin Provisioned” volumes versus “Regular Thick” volumes. Need to visually show allocated versus virtual capacity.

Note: A generous upper-bound must be determined relative to how many volumes can be created at one time.

Note: A transactional model needs to be implemented to ensure the successful completion of multi-volume creation.

PAS Note: Seems like it might be best for the user to make all of their volumes “Thin Provisioned”. Would this help us provide any actual storage usage information to the user? Any input here?

9.6.4. Copy Services

PAS Note: This section is TBD. Need to look at our various copy services operations and present them from a user-centered perspective. For example, one competitor has the notion of expiring snapshot volumes with corresponding actions taken once the snapshot expires. The main contributor to this section is Scott Hubbard.

9.7. Configure – File-Based Access Definition

TBD pending selection of file-based application.

9.8. Configure – Alerts Definition

The ability to configure alerts at the individual Element Manager level will essentially mimic the configuration options and requirements that are specified for the EC (with exceptions noted). Therefore, refer to section 8.13 for an explanation of the alert behavior.

9.9. Configure – User Roles Definition

PAS Note: This section is TBD. Need to look at our current user role model (beginning with Flint) and determine if changes are needed from a usability as well as convergence standpoint. We need to ensure that we have consistent user roles defined across the various LSI devices. For other information on Security and User Roles, refer to section 15. The main contributor to this section is Scott Hubbard.

9.10. Configure – Hosts/HBAs Definition

Tasks and appropriate views must be provided to allow the user to configure logged-in HBAs into applicable hosts to be used in gaining access to configured volumes. The user must be able to manually configure hosts as well as obtain them through an automatic detection method (assumed to be similar to the current Host Context Agent).

9.11. Configure – Mappings Definition

Tasks and appropriate views must be provided to allow the user to map hosts to volumes to gain access for read/write activity.

Note: In addition to a sorted list of mapping relationships, a graphical view depicting these relationships would be very beneficial.

9.12. Configure – Miscellaneous

PAS Note: This section needs further definition. The main contributor to this section is Scott Hubbard.

- Ability for the user to create folders to organize various storage entities. This is similar to the current Flint implementation.
- In addition, any configuration tasks/views required for other physical components (storage array, controllers, etc.) must be provided.

9.13. Monitor Definition

PAS Note: This section needs further definition. The main contributor to this section is Scott Hubbard.

PAS Note: The overall service philosophy defined in section 21 will also drive requirements in this area. Section 21 is TBD right now.

Note: The end user/support personnel will spend a majority of the product's life in the monitor stage (monitor, service, recovery). Therefore, it is imperative that we really emphasize this stage and provide the necessary tools and views for an end user and support personnel to quickly and easily diagnose and recover from storage-related conditions.

The monitor stage must provide the following information and tasks to the user/support personnel:

- The ability to monitor the performance of the various source entities (both real-time as well as historical trending data).

- The ability to monitor storage actual usage to obtain real-time data as well as predict growth rates. PAS Note: Not sure we have the mechanisms in place to actually know how when configured storage is actually being used.
- The ability to see and track any current operations in progress in a single consolidated view.
- The ability to access events and save and load them.
- The ability to troubleshoot and recover from problems with the storage via lead-through procedures (i.e. similar to the current Recovery Guru). All tasks associated with solving the problem must be available within the Recovery Guru itself.
- The ability to gather information on components or situations that may be trending toward a non-optimal condition but have not reached that point yet (better predictive failure analysis).
- The ability to upgrade firmware on the various components (not just the controller) that comprise the storage system. Given the importance of upgrades and the reluctance of the user to perform upgrades, this area needs to be really emphasized to provide the appropriate views and tasks to complete this operation. The upgrade design and tasks must be common, where possible, between the Element Manager and the Enterprise Console.
- The ability to tune/reconfigure the storage in response to particular monitoring alerts or corresponding information.
- The ability to query for various items. See section 8.12 for details.
- The ability to gather the required elements needed for a support person to properly diagnose the system needs to be provided (aka support bundle).
- The ability to get an inventory (i.e. Storage Profile) needs to be provided.

PAS Note: It needs to be defined whether the reconfigure tasks are included as part of the Monitor life cycle or the Configure life cycle. Most reconfiguration activities will be the result of some condition detected during monitoring.

9.14. Retire Storage - Solution Definition

PAS Note: This section needs further definition. The main contributor to this section is Scott Hubbard. The views and tasks associated with the retire stage need to focus on both the complete decommissioning of a storage device as well as individual movement (migration) of pieces of the storage from a storage tiering perspective.

10. Stand-Alone Solution Definitions

The following solutions are those that are generally independent of the storage management application but provide the overall solution needed by the various users of our products.

10.1. Simulator/Demo Tool and Sales Package

The demo/simulator tool that we currently provide as part of our current management software continues to be a very valuable tool that is used extensively in the field. It is used to demonstrate our software to aid in sales/planning discussions. In addition, it provides a convenient training tool that an LSI representative can leave behind with the customer.

Making favorable first impressions relative to product features/usability via demonstrations and during sales discussions is critical to the on-going success of our products. The demo/simulator tool for Amelia must have the following capabilities:

- Mimic all of the operations that can be performed with the regular management tool.
- Provide a lead-through wizard that aids the sales person or customer in developing their desired configuration of hardware components and premium features. The lead-through wizard must also include some storage allocation tips to ensure that the user understands how storage is allocated for various operations (especially copy services). This initial configuration would serve as the starting point for the user to create various logical entities.
- In addition to the lead-through wizard mentioned above, provide several “canned” configurations that already come pre-loaded in the simulator.
- Once a configuration has been determined and saved…
    - Provide appropriate planning information back to the user such as power consumption and cooling requirements.
    - Provide at least a rudimentary parts list that would aid in developing a formal quote.
- Provide the capability for the system/sales engineer and customer to not only configure the logical entities of the system through the simulator but be able to save that configuration and load it (i.e. apply it) when the actual system arrives.

The simulator/demo tool should be considered one component of an overall electronic “Sales Package”. The other components of the Sales Package should include materials and information (in an appropriate media form) that highlight the features and concepts of the storage management software and associated hardware. These materials must focus on information needed during the pre-sales/planning stage.

10.2. One-Minute Storage Clinics/Briefs

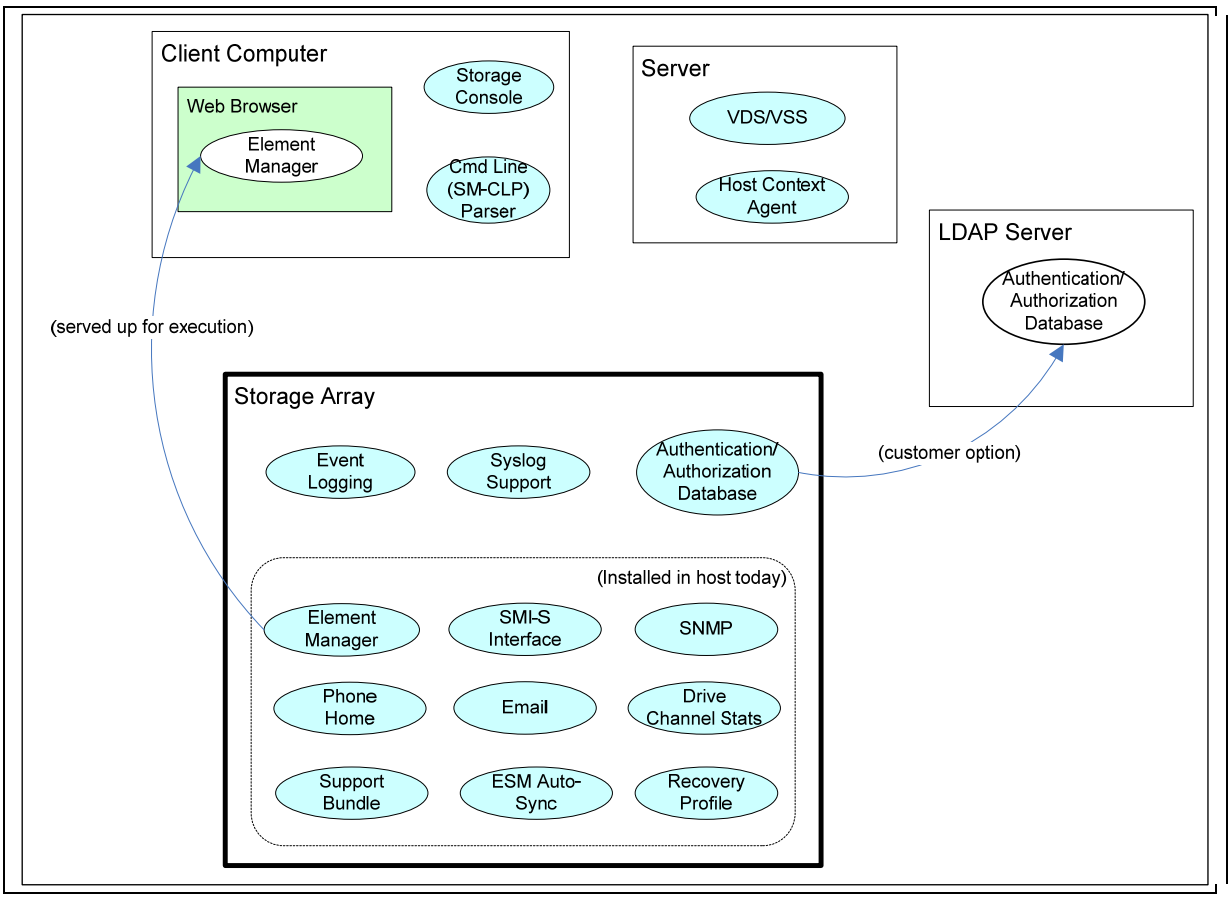
Provide very focused (no more than 1 minute or one-page) clinics/briefs on various aspects of storage including the pros and cons of certain technology. This can be in the form of video and online “Technology 101” briefs.

PAS Note: Other stand-alone solutions are TBD.

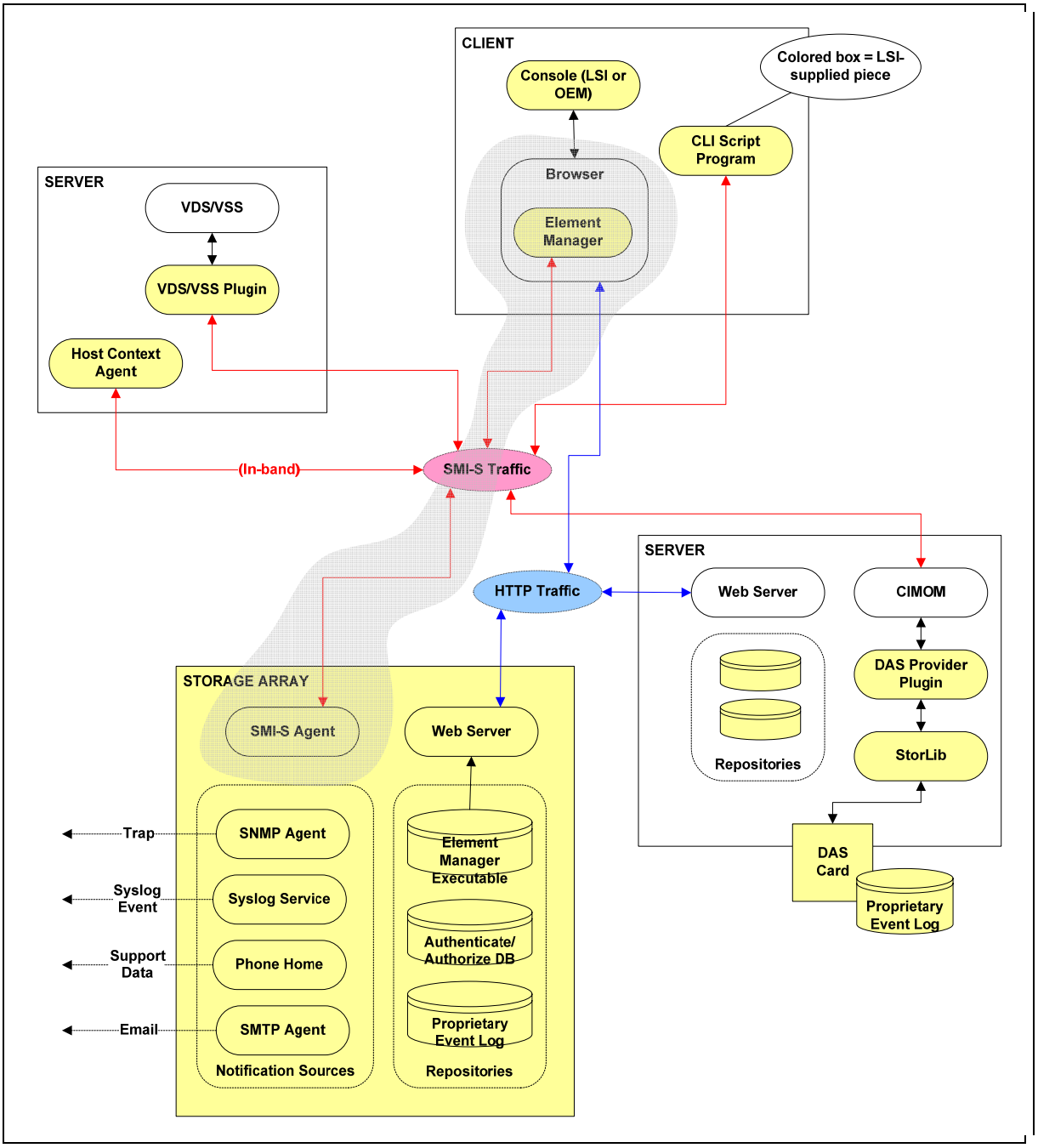
11. SW/FW Architecture Definition

11.1. Overall Architecture Definition

For reference, the diagrams shown in the following figures illustrate the various component locations and interactions, respectively, in a Amelia storage environment. The details of these diagrams are explained in the sections that follow.



Component Locations for External RAID Products in the Amelia Architecture



Component Interactions in the Amelia Architecture

11.2. Device Interactions via SMI-S

A fundamental infrastructure requirement for Amelia is that all interactions between the management application and the device occur using the SMI-S API/protocol. OEMs and end customers are demanding support for and compliance with SMI-S so they can more readily exploit their own customized applications, as well as any 3rd-party storage applications, all of which must be assumed to utilize an SMI-S foundation. Given this requirement, it simply does not make sense for LSI-ESG to build its own storage management application on top of some other (presumably proprietary) API or protocol. Doing so would only necessitate that the proprietary protocol be developed, tested and supported *in addition to* the SMI-S protocol.

While it may be necessary, for transitional purposes, to support both SMI-S and a proprietary protocol for some period of time, the long-term objective must be for LSI-ESG to migrate completely to the SMI-S interaction model for all communications between management application and managed device.

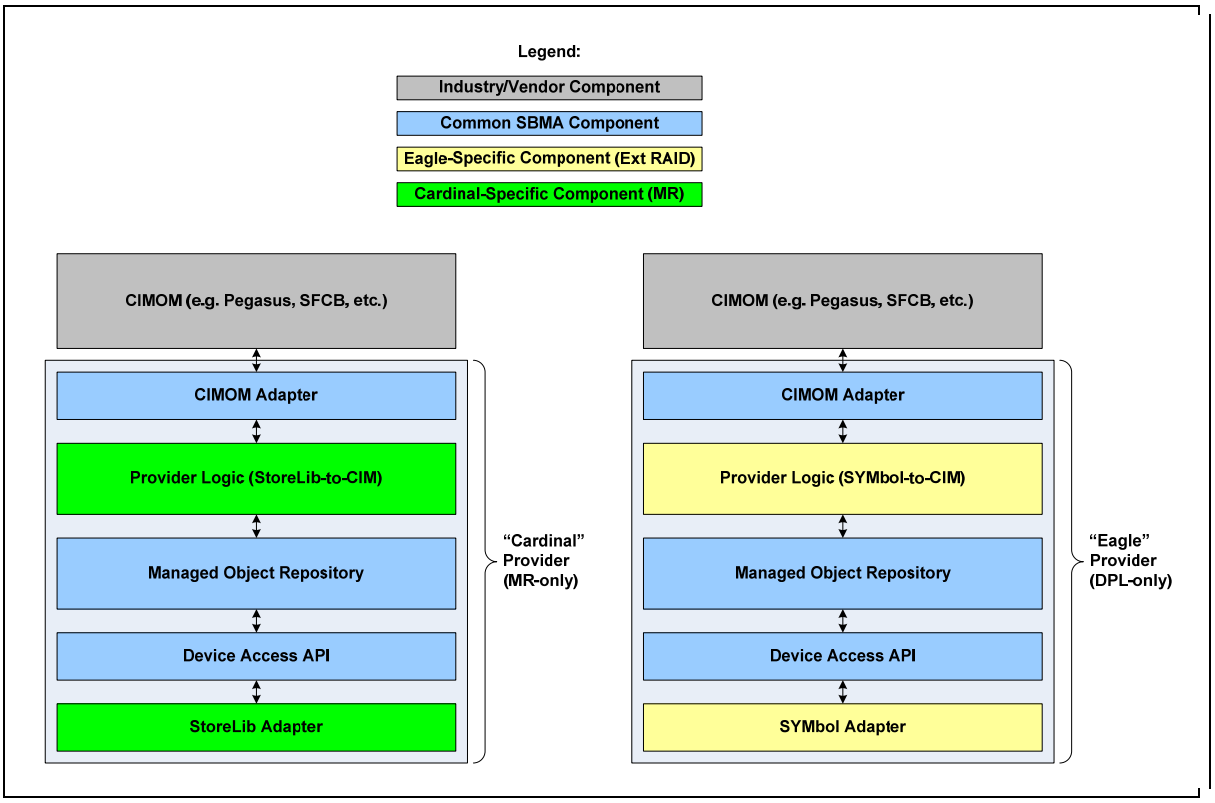
11.2.1. SMI-S Embedded Agent (Device-Resident)

In the case of External RAID devices, the SMI-S capability of the device must be implemented natively within the device, such that no external, host-resident Proxy Provider is required as a gateway between management applications and the device. This is a strong customer demand even today, and the need for this level of function becomes even more critical with the transition to LSI-ESG management applications and tools that utilize SMI-S as the primary interface.

11.2.1.1. Technical Background on Established Products

LSI-ESG has been actively engaged in the development and release of SMI-S support using external Proxy Providers, coupled with host-resident CIMOMs, as an interim means to provide SMI-S access to both DAS and External RAID devices. The current technical foundation for these providers is an internal architecture called SBMA, an acronym for “Standards-Based Management Architecture”. Two implementations of this architecture are shown in the following figure, namely the providers for the Cardinal (MegaRAID) program and for the Eagle (External RAID) program. The components shown in blue are common ones, where the shared implementation is fully leveraged/reused across the programs. The green and yellow components are specific to the MegaRAID and External RAID implementations, and thus vary significantly between the Cardinal and Eagle releases. The CIMOMs, shown in gray, are industry/vendor components that are certified with the LSI-ESG providers as part of the release cycle; these are not developed or customized by LSI-ESG. Note that the SBMA structure utilizes the CMPI (“Common Manageability Programming Interface”) for interactions between Provider and CIMOM, and thereby enables simple integration with any CIMOM that supports CMPI. The two most significant such CIMOMs are Pegasus and SFCB (“Small Footprint CIM Broker”), both of which are available in open-source form.

An important point to note is that the Eagle provider, for External RAID products, implements its SMI-S capabilities by interacting with storage devices using the proprietary SYMbol API.



SBMA Provider Implementations

11.2.1.2. CIMOM Foundation for Embedded SMI-S Agent

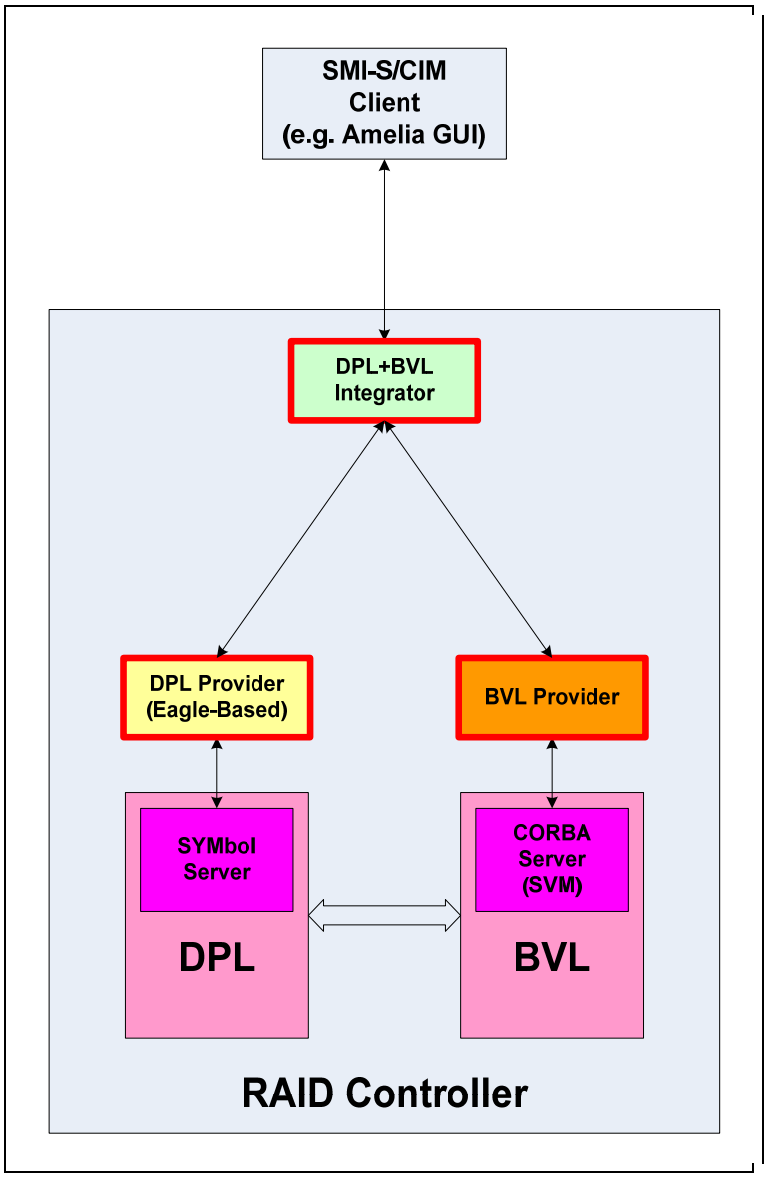
To address the need in Amelia for a device-embedded SMI-S Agent, the Proxy Provider development work will be heavily leveraged. The key to achieving this reuse is to deploy, within the device itself, a CIMOM that supports the CMPI interface, and thereby allows the Eagle Proxy Provider to be plugged into it. With this approach, the CIMOM can expose the SMI-S capabilities of the device to the outside world, while the Proxy Provider establishes the internal linkage to the core device logic, just as it did when deployed in a CIMOM running on an external host system. While it is likely that some adaptation of the Eagle Provider will be required to make it run in this new environment, the vast majority of the core logic can be reused without significant change.

The targeted CIMOM for the embedded implementation is SFCB, which is specifically designed for resource-constrained environments of the type found in the LSI-ESG RAID controllers. In its standard distribution form, the SFCB build and run-time environments are limited to Linux. However, there appear to be no significant technical limitations that would prevent it from being re-targeted for the VxWorks environment of the External RAID controller products.

11.2.1.3. Internal Structure of Provider and Embedded Agent Logic

The SFCB CIMOM and the Eagle-based Proxy Provider components are two critical pieces of the overall SMI-S embedded agent solution, but there are additional elements that are equally important. First, it must be recognized that the Eagle provider logic focuses entirely on the DPL ("Data Protection Layer") feature set within the External RAID products, i.e. the features that are exposed *internally* via the current SYMbol API. However, in the Amelia timeframe, the integrated BVL ("Block Virtualization Layer") capabilities from the StoreAge product set will be included in the RAID firmware, and will provide significantly richer top-level volume functions than what was available prior to that integration. It will, therefore, be necessary to have SMI-S

Proxy Provider capability for monitoring, configuring and managing the functional aspects of the BVL features. Maintaining implementation boundaries between the DPL Provider and the BVL Provider is highly desirable as a means of reducing and compartmentalizing the complexity of the overall solution. Consequently, the structure shown in the following figure will be used to implement the solution. All of the provider logic will be implemented in conjunction with the SFCB CIMOM, as described above. However, there will be separate internal components for the BVL and DPL providers. A third, higher-level entity, the "DPL+BVL Integrator", will then be developed to merge the management capabilities of the underlying components, establish SMI-S associations between their relevant/related objects, and make the system appear, to the external world, as a single, unified device with no obvious DPL/BVL boundaries. Internally, the DPL Provider will interact with the DPL proper using the legacy SYMbol API. Similarly, the BVL Provider will interact with the BVL proper (and the SVM component, in particular) using the legacy CORBA API. However, both the SYMbol and SVM CORBA APIs will be deprecated for external use, and customers wishing to develop applications that interact with the devices via API mechanisms will be guided to use the fully-functional SMI-S APIs.



Composite Provider Structure for DPL+BVL

A few additional notes pertaining to the previous figure are in order. As shown in the diagram, a client application using the SMI-S interface (such as the Amelia GUI, or a customized 3rd-party application) interacts with the storage device through the SFCB CIMOM using capabilities and data supplied by the DPL+BVL Integrator. Those capabilities and data are generated by the DPL+BVL Integrator through interactions with the underlying DPL Provider and BVL Provider. Most importantly, though, it is only the top-level mechanisms provided by the DPL+BVL Integrator that are made visible to the external clients.

In addition, the specific API interface between clients and the CIMOM is encapsulated in the CIM-over-HTTP protocol, as defined by DMTF/SNIA. The SFCB CIMOM implements this protocol, thereby allowing clients to establish network connections directly to the storage device via its integrated Ethernet management port(s).

11.2.1.4. In-Band SMI-S Access

PAS Note: Need weigh in from product management on in-band requirement.

LSI-ESG's External RAID products have traditionally provided both Out-of-Band access (i.e. direct connection from clients to the controller Ethernet port) and In-Band access, where clients connect via standard network protocols to a Proxy Agent running on a host, and the Proxy Agent simply passes the network data through to the controller via a proprietary tunnel established over the I/O path. The tunnel generally uses standard block device READ/WRITE operations to convey data between the device and the Proxy (and thus the remote client).

For Amelia, the In-Band capability is being specifically omitted from the requirement set. The assumption is that typical end-users are familiar enough with standard networking environments, and that low-cost Ethernet switches are prevalent enough, that Out-of-Band connectivity is a much-preferred alternative. In addition, Out-of-Band mechanisms do not require the host-resident Proxy Agent, which aligns with the desire to eliminate all host software installations for the management application.

11.3. GUI Application Structure

The overall GUI architecture is comprised of (1) HTML and JavaScript loaded by the browser client, (2) SMI-S services to supply the data model, and (3) web-server components to adapt SMI construct to web-based application constraints. These components can be located on different systems to allow for a fully embedded service (which is the primary goal), but also to support a partially embedded or a stand-alone solution. The partially embedded and non-embedded structures can be used for MegaRAID devices, SVM appliances, and legacy systems. The following sections describe these structures in detail.

PAS Note: Are we really intending that this applies to legacy devices as stated in the previous paragraph?

11.3.1. JavaScript Executable Format

As noted in Section 2.4, the objective of providing a browser-based storage management solution makes JavaScript an extremely attractive target for the "execution format" of the application. JavaScript capabilities are fundamental to the widely-adopted "AJAX" technology base that is nearly ubiquitous in the rich web-based applications currently available in the market. All leading browsers (IE, Firefox, Safari, etc.) offer full, built-in support for JavaScript capabilities.

Browser-based applications can be built using a number of client-server partitioning structures. One model uses a very thin presentation layer at the browser with most of the logic executed on the server. At the other extreme, the server may only be responsible for providing a few pages and some raw data, with most of the presentation logic is computed in the browser client.

When the Amelia user interface is served from an active storage controller, the objective is to minimize the management compute requirements on the storage controller so that its compute resources are performing I/O operations, optimizing system performance. Therefore the preferred architecture for the Amelia solution is to rely on the storage controller for the SMI-S data, but run as much as possible of the client application in the browser.

11.3.2. GWT Development Environment and Infrastructure

Many Javascript libraries exist to aid in the development of browser user interfaces. Browsers implement Javascript slightly different from each other so an important benefit of these libraries is to provide an abstracted user interface that renders consistently on the set of supported browsers. Abstraction interfaces include AJAX methods to fetch data from servers, common UI elements such as tabbed navigation, calendars, user input forms. Some of the available libraries are Prototype, JQuery, Yahoo UI (YUI), and Dojo.

Building a web-based application without the use of a library is ill advised because these libraries solve most browser compatibility issues and greatly simplify the coding effort. Pragmatically, any of these libraries could be used to implement a user interface using the SMI-S XML data model.

An additional challenge with moving to a JavaScript executable format from the current Java-based technology solution is that the LSI-ESG's engineering organization is heavily focused on Java as its development environment. Independent of that, JavaScript is widely known to offer serious challenges in terms of its direct application to large-scale, sophisticated software packages. As a programming language, it lacks many of the Software Engineering mechanisms that facilitate disciplined development and maintenance of rich applications.

To mitigate the risks associated with the shortcomings of JavaScript technology, the LSI-ESG approach will utilize GWT ("Google Web Toolkit"), which is available in the public domain from Google. GWT, at its heart, is a framework for developing applications using the Java programming language, and all of the rich Software Engineering tool chains that support Java development, but with a target execution environment of JavaScript. The key technology within GWT is a cross-compiler that produces JavaScript executables as the binary image using Java as the source code format. The dominating advantage of this approach is that the developer uses a type-safe language within an integrated development environment with debugging and unit-testing capabilities.

Admittedly, the ancillary libraries and extension packages available for use with GWT are much more limited than those available for true Java applications. However, there are ample extension mechanisms available to enable GWT applications to exploit 3rd-party JavaScript libraries, and even the base GWT libraries appear to be rich enough to satisfy the key requirements of the relevant LSI-ESG applications. We must be judicious in making these choices as using a collection of disparate libraries could create an unmanageable software architecture.

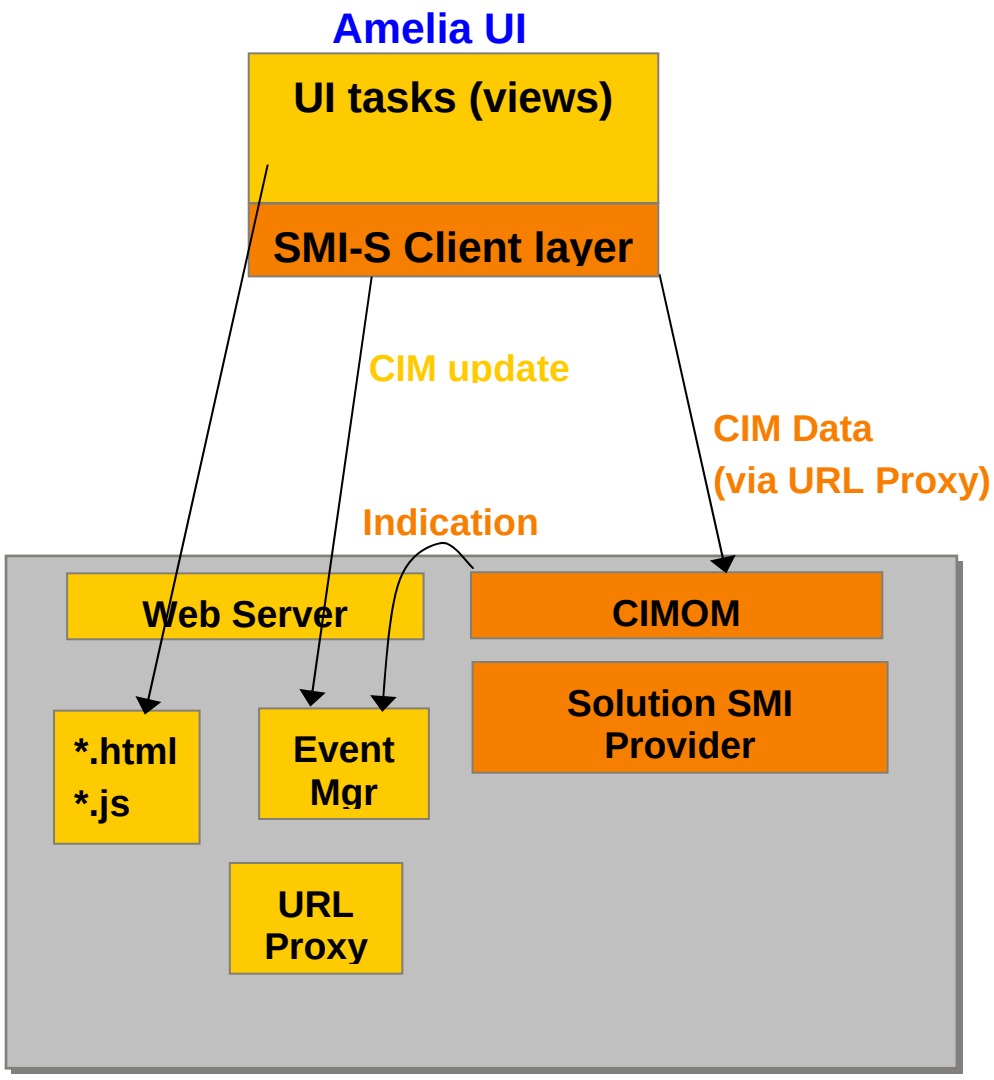
An important benefit to LSI-ESG of using GWT is that the organizational skill sets developed over the past decade (or thereabouts) of Java-oriented development can be retained and heavily exploited/reused, even though the Amelia solution's executable image will be JavaScript. For engineering a feature rich, enterprise class user interface, having a rich development environment with a modern programming language is a substantial benefit and the primary reason for using GWT for the Amelia user interface.

It is important to distinguish between 'development environment' and 'widget library'. GWT does receive some criticism for it rather mundane set of widgets – they are generally just a direct mapping to HTML tags with a little decoration. Other JavaScript libraries may include a widget that perfectly suits our user experience requirements. It is possible (and quite

straightforward) to wrap an externally loaded JavaScript object in a GWT class for inclusion in a GWT user application

11.3.3. Embedded Service

The figure below illustrates the functional components of the management solution when the web server and CIMOM run on the same system. This case applies to when a web server and CIMOM are both embedded on the RAID controller platform as well as for a stand-alone DPM-based SVM deployment where the CIMOM is installed on the SVM management appliance.



11.3.3.1. Web Client Components

Refer to the previous figure.

UI Tasks – the user interface management views rendered by the browser, presented to the system administrator.

SMI-S Client – a client-side JavaScript module that uses the SMI-S protocol to interact with the CIMOM to gather system state information, monitor state changes, and submit configuration change requests. This layer will publish a native JavaScript interface so that OEMs can use this module to create a custom web interface using whatever JavaScript application development strategy they choose. This interface must be compatible with the use of other JavaScript libraries for UI development (such as Dojo).

11.3.3.2. Web Server Components

Refer to the previous figure.

Web Server – a standard web server that responds to http requests for web content pages.

***.html, *.js, etc** – Web content pages that form the basis of the web application including the launch point for the management application.

Event Manager – is an adapter to make standard CIM indications compatible with a web-based client. Normally, a client registers a call-back URL with a CIMOM to receive notifications for state changes. When a state change occurs the CIMOM calls the URL and posts an XML document containing the Indication data. Web clients can only initiate connections – they cannot accept a connection from a server. The Event Manager is a component that can be URL addressed, which registers with the CIMOM for CIM Indications, and also provides an interface for browser clients to request the indications that is compatible with web client rules. See section 11.6 for more information on the event manager's role.

PAS Note: Initialization sequence for the Event Manager is determined but needs to be documented by Bill Hetrick.

URL Proxy – is a component that allows the web application to access pages originating from a different base-URL. The proxy is necessary because of a security feature, the *same origin policy*, built into all current web browsers that requires client-side scripting (such as JavaScript) can only access pages that are served from the same base-URL. However, with standard web servers and standard CIMOMs, even if the URL has the same base address, the port numbers will differ (port 80 for web content, port 5988 for CIM content), and the port number is part of the base-URL of the same origin policy restriction. The URL proxy component is a server-side script or servlet that simply forwards URL requests to another server. This allows the browser client to access the originating server when fetching SMI-S data from the CIMOM by passing the request through the URL proxy.

PAS Note: How to do this with HTTPS – question posed by Bill Hetrick?

11.3.3.3. SMI-S Components

Refer to the previous figure.

CIMOM – standard CIMOM

Solution SMI Provider – and SMI provider, accessed through the CIMOM to provide state and configuration information of the overall solution. The SMI-S solution of the overall storage application solution could be implemented as more than one provider, each of which controls a portion of the overall solution. For example, for a file-based access (FBA) storage solution, one provider could service hardware features, another could service block-storage features (for the underlying file system storage), and another could provide the file system and file-export

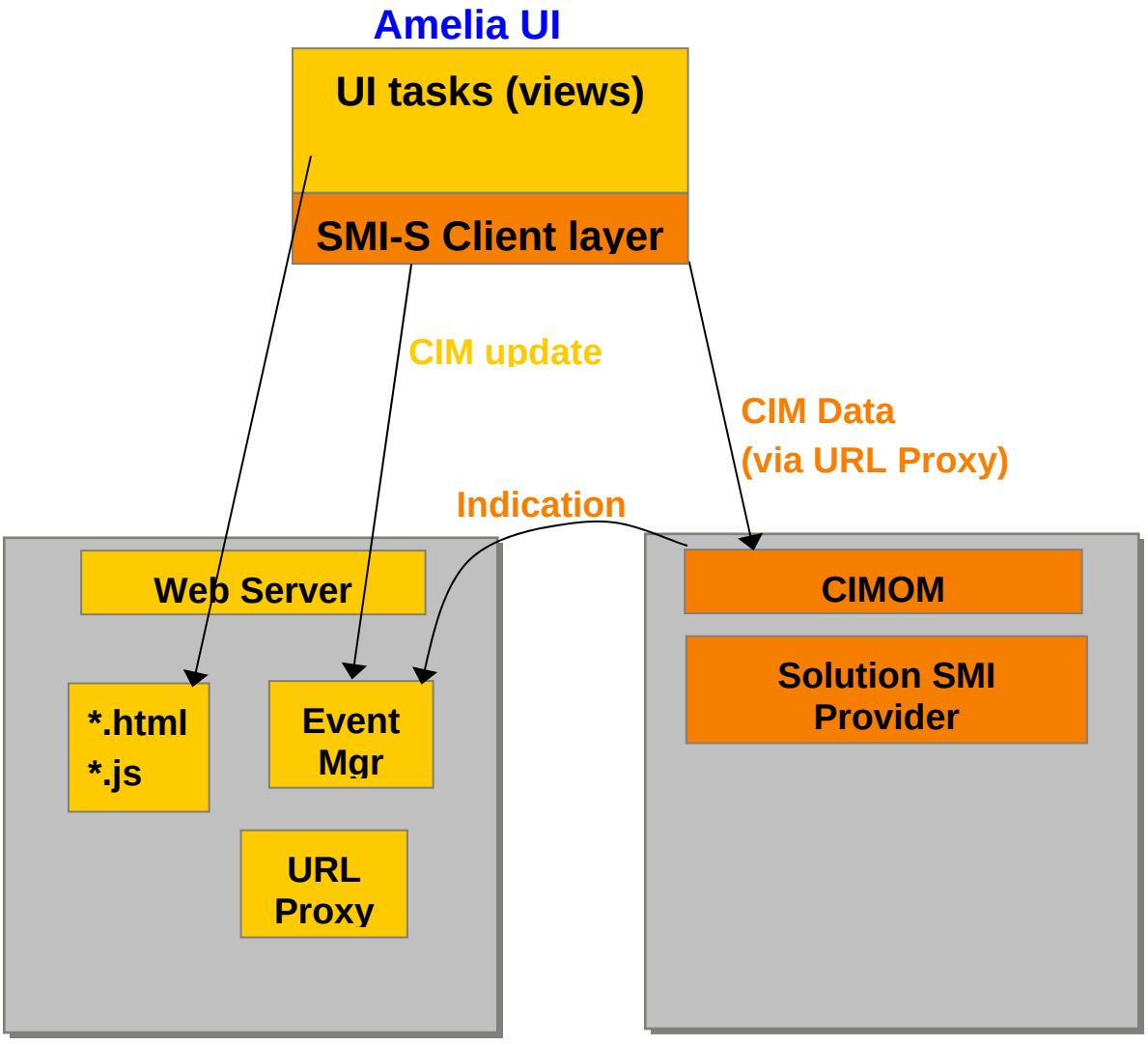
services. The key is that the storage solution must be cohesively managed from the provider(s) through the CIMOM.

Note that the fundamental requirement is that the managed solution provides a compliant SMI-S management interface. The SMI-S components shown in the above diagram are based on the assumption that existing software will be used to implement the SMI solution. However, if we integrate a storage application that provides a compliant SMI interface without a CIMOM or provider, there is no reason to require this structure.

11.3.4. Partially-Embedded

The figure below illustrates the functional components of the management solution when the web service is running on a management server and CIMOM is running on the managed system. This case applies to the device-management of a MegaRAID device when the MegaRAID host machine is not hosting web services for the management application.

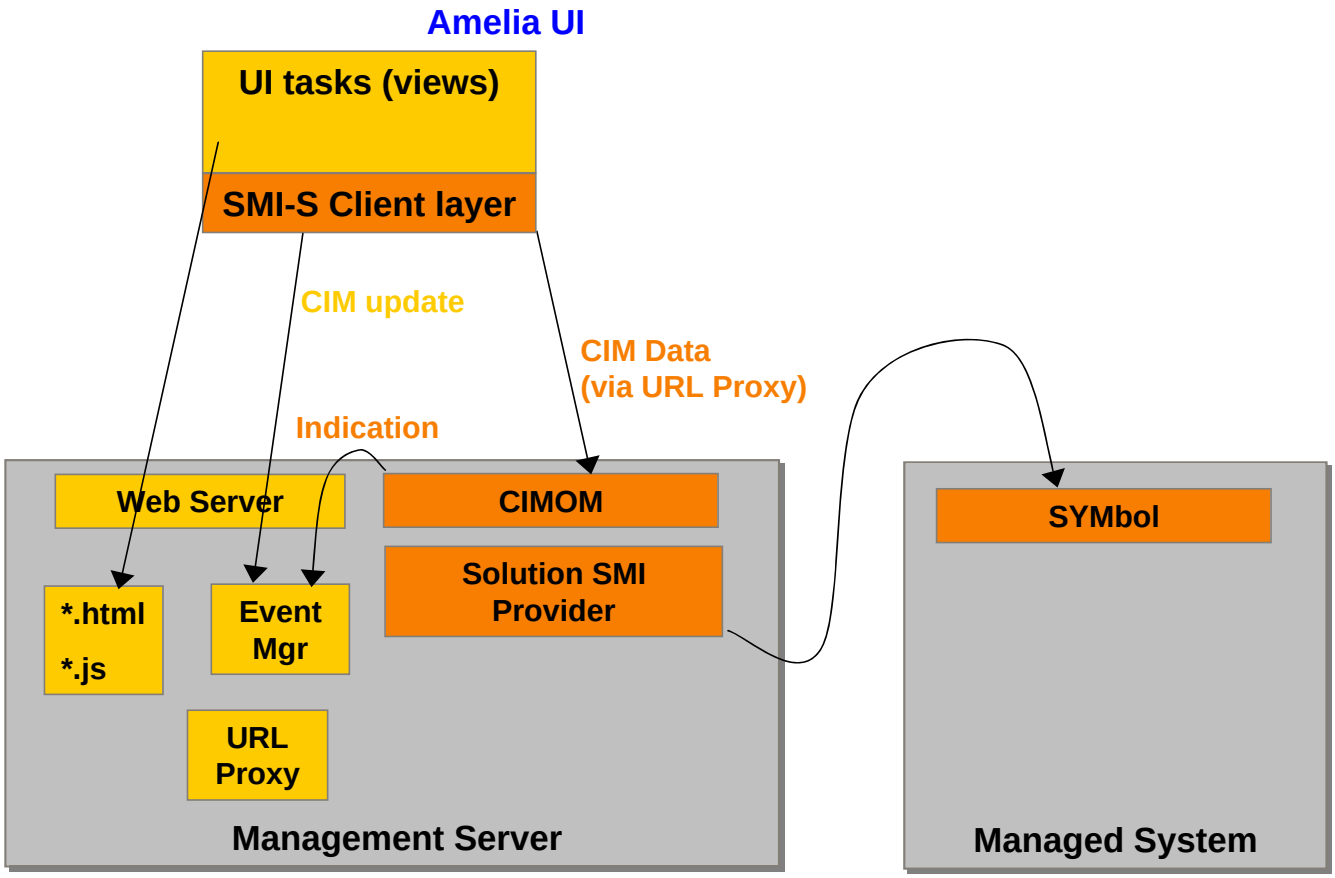
PAS Note: Need to define High-Availability scheme



11.3.5. Non-Embedded

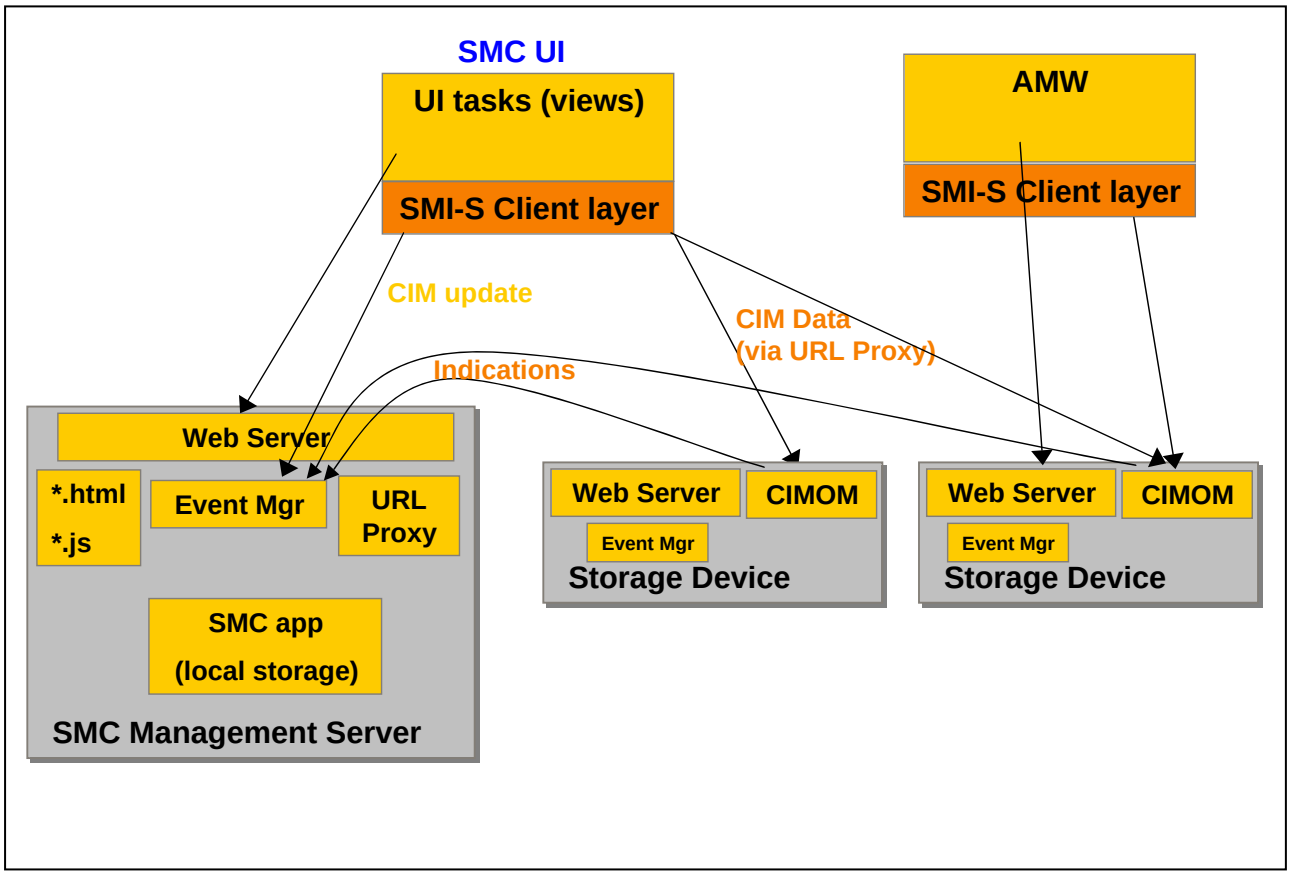
The figure below illustrates the functional components of the management solution when the web service and the CIMOM are running on a management server. This case applies to the providing web-based device-management of legacy systems where embedding these services is not feasible due to embedded system constraints such as memory requirements.

PAS Note: Need to define High Availability scheme.



11.4. Enterprise Console (EC)

The purpose of an Enterprise Console (EC) is to provide an interface to show a set of storage devices with some basic summary information such as status and configure/free capacity. This console is similar to the SANtricity Enterprise Management Window (EMW), but where the EMW generally only displays a Device Management Window launch point with a summary status, the EC (in concept) can extend well beyond launch point functionality with features such as capturing the support bundle from a collection of managed devices or downloading firmware to a set of managed devices or even provisioning storage. Because of the robustness of the EC, it must be an SMI-S client, managing SMI data from multiple storage devices. As a browser-based SMI-S client, the EC can make use of the same Event Manager and URL Proxy that are required for the individual device management function (see the following figure)



PAS Note: SMC should be replaced with EC in the final version

11.4.1.1. Enterprise Console (EC) Application

The core EC application is installed on a management server somewhere in the IT network. (It could be installed on one of the Storage Devices, but that it a special case of the general solution.) The application is browser-based, launched from HTML (and JavaScript, CSS, etc.) pages stored on the management server. A core application module running as a servlet or CGI service provides access to dynamic data for the EC view. Dynamic data includes items such as a list of managed devices, device discovery requests, and sign-on credentials. The server-side application module most likely will require some non-volatile local storage to store and update files.

11.4.1.2. Event Manager

The Enterprise Console (EC) generally provides a list of managed devices in the management domain along with a summary status and statistics for each managed device. This information is gathered from each managed device using the SMI-S protocol. With a generalized implementation, the Event Manager described earlier to support the Device Management interface can be common for the EC application. Specifically, when a client accesses the Event Manager URL, it includes a list of managed devices monitored by the EC. The Event Manager

registers for CIM indications with each managed devices so that it becomes the single source of the event stream for the managed devices. The call-back URL that is registered with each managed device includes a tag that allows the Event Manager to organize the event queues and notify the appropriate clients.

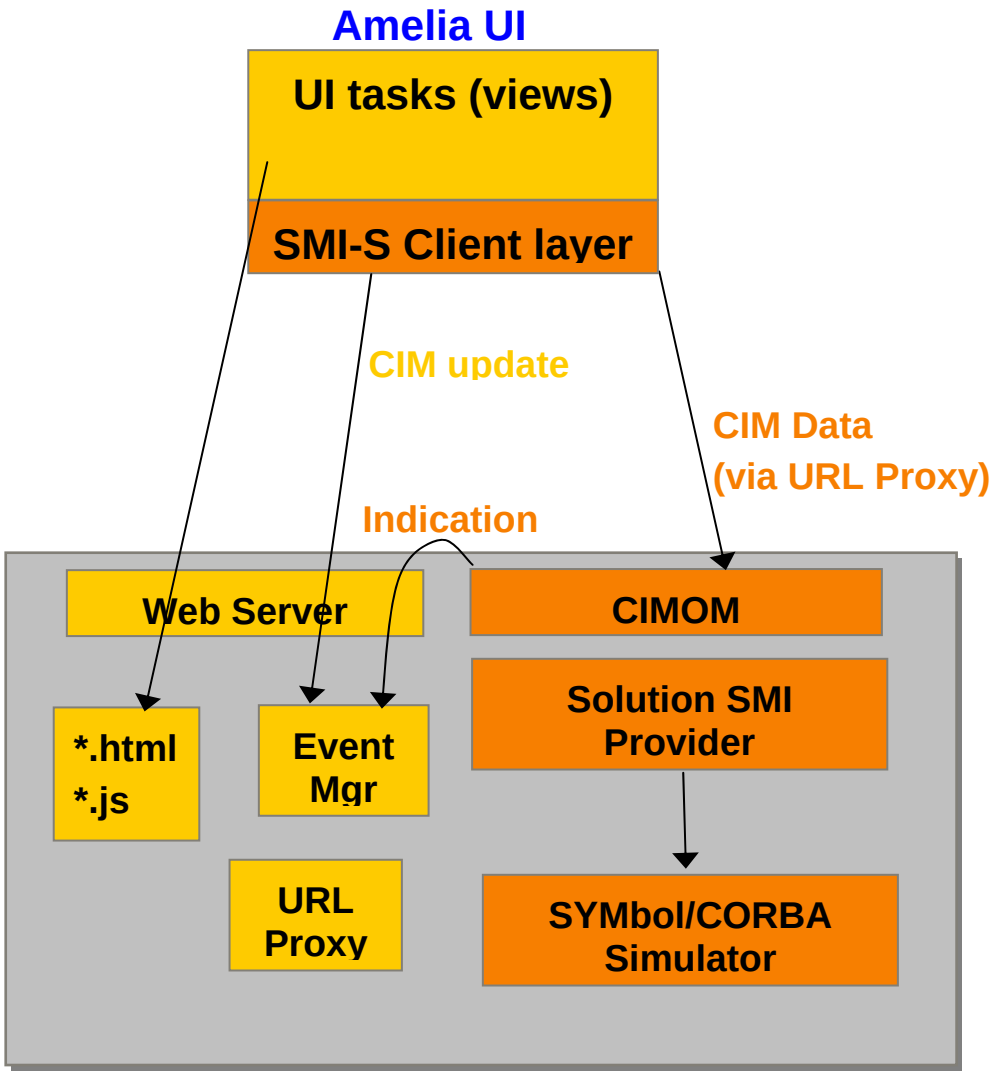
11.4.1.3. URL Proxy

Same proxy as used in the Device Management

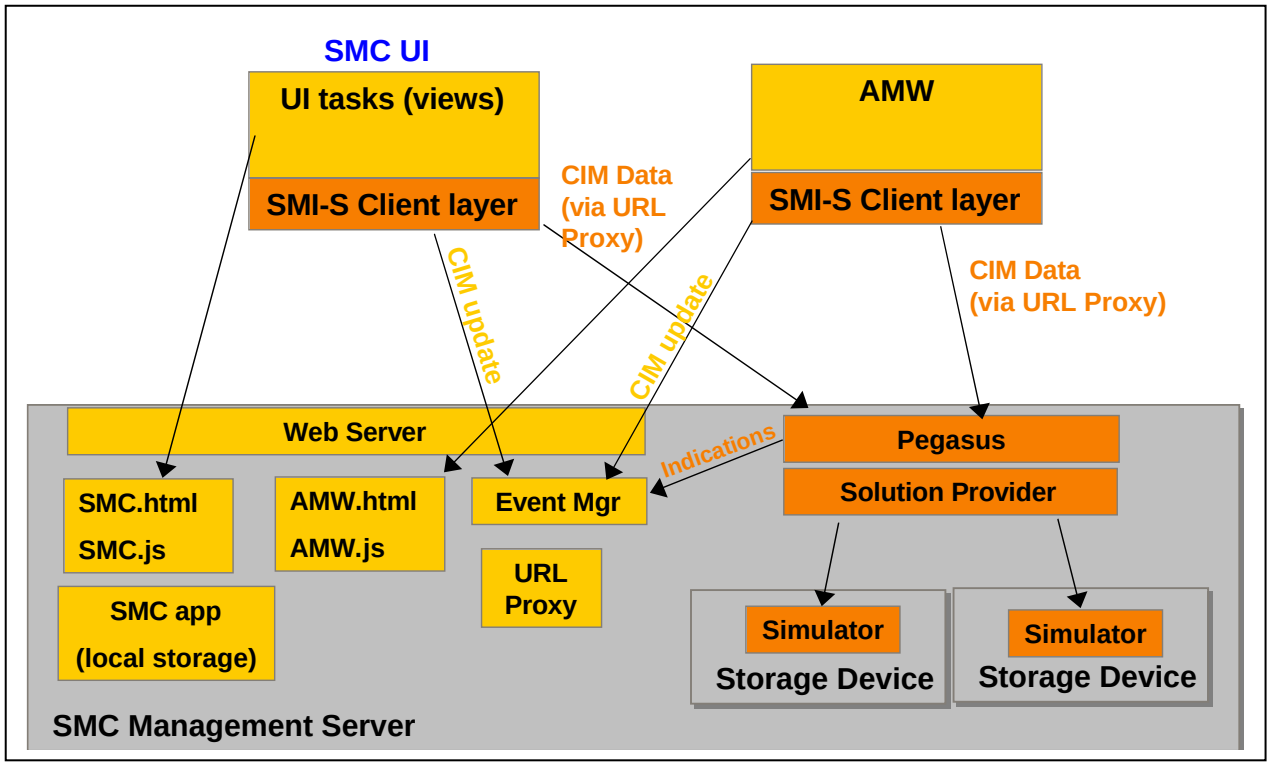
11.5. Simulator

Device Management simulation

Simulate SYMbol/Corba data, while using the standard SMI-S implementation.



Enterprise Console Simulation



PAS Note: SMC should be replaced with EC in the final version

11.6. Automatic State Updates

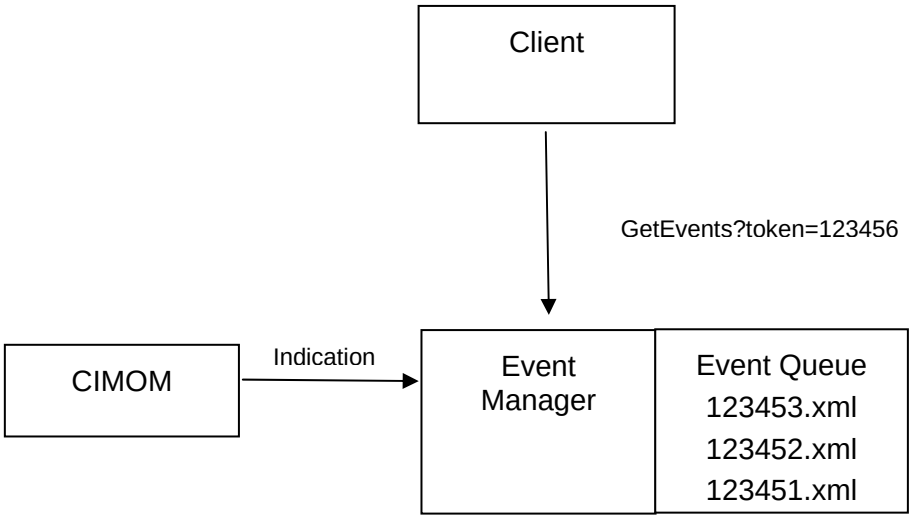
When the state of the managed device changes dynamically, the state change must be reflected in the user interface without user intervention. In contrast, primitive web-based management applications may require the user to 'refresh' the display in order to view the current system state. The two general use cases are 1) when a storage system component (e.g. drive) changes state (e.g. fails) and 2) if one client performs a configuration change any other client monitoring the same system will see the result of the change request.

Configuration changes and state changes are handled in the SMI-S model as 'Indications'. In general, an Indication Client registers with the CIMOM of a managed device as an Indication listener and includes a callback URL (and possibly a filter expression). When the SMI-S provider/CIMOM updates its model, an Indication message is posted to the registered URL. This model is not directly compatible with web-based SMI-S clients because web browsers cannot accept the callback connection as a target.

The Event Manager component in the previous diagrams serves as an adapter from the CIM standard Indication model to a communication method compatible with web-based clients. The Event Manager provides a client-accessible URL to request any outstanding indications. The two arguments to this request are 1) some Event Manager controlled token (e.g. a timestamp), and 2) the address of the managed system. The first time a client calls this URL, the Event Manager registers its callback URL with the CIMOM of the managed device. If the client-provided token matches the current Event Manager token, the client request is held open for up to 30 seconds during which time the Event Manager periodically checks for new events. If the client toke does not match the Event Manager token, the Event Manager returns the set of events back to the client to get it up to date. If the client token is too old so that the Event

Manager cannot provide a full set of events to refresh the client model, the new token is returned, but with no data which triggers the client to re-discover all of the CIM data maintained by the client.

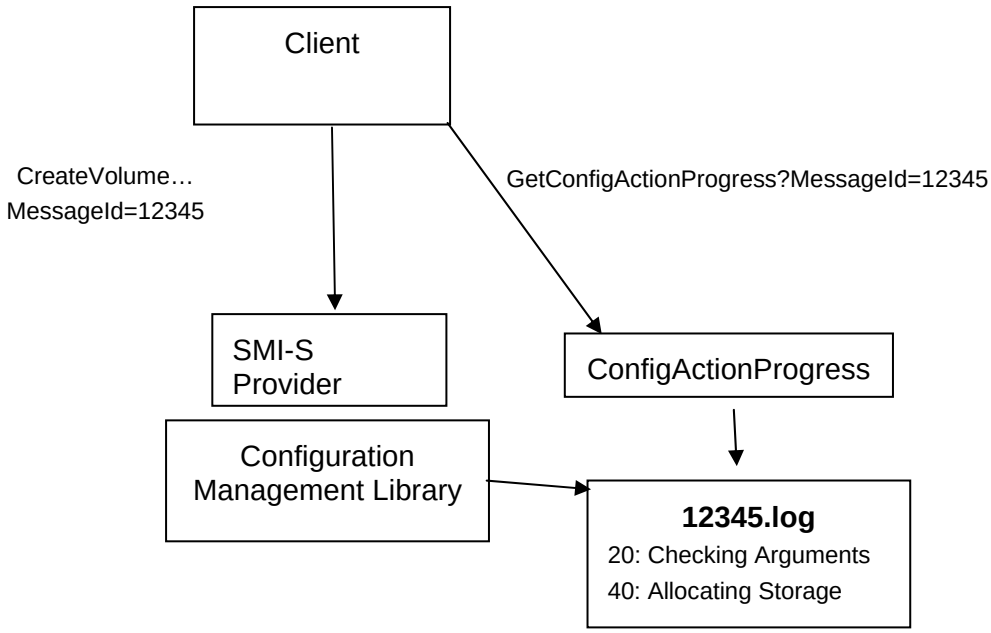
When something changes on the managed device, the CIMOM calls the callback URL and posts the indication. The Event Manager adds the indication to a small queue of indications used to reduce the probability that a client must re-discover. By updating its queue, the event token changes, thereby triggering notification of the indication to all clients.



PAS Note: Error handling – what if the CIMOM fails delivering an event? How does the event manager 'catch up'? How does the event manager get notified? How does the client get notified? Will the client have to rediscover?

11.7. Configuration Request Progress Markers

PAS Note: Doesn't this diagram require some explanation?



11.8. Management Application Feed-Out from Device

As noted in Section 2.4, a key objective of the Amelia architecture is migration to a browser-based management application for which no host-oriented installation process is required. If the device management application (or, more specifically, its executable/binary image) does not reside on a host, then it clearly must reside within the device itself, and be fed out from the device to the browser that is to be the execution home for the application. Having the device act as the repository for its own management application's executable image introduces several important second-level requirements into the overall solution. The most important of these are described in the following sections.

11.8.1. Integrated Packaging of Firmware and Management Application

Clearly, for the device to be able to feed out its management application to client systems for execution, the binary image of the application must reside within the controller firmware image, or at least in a ROM/Flash device on the controller board itself. The option of storing this image on disk is much less appealing, due to the logistics associated with diskless (e.g. factory) operation, cabling problems or malfunctions that prevent disk access, and similar scenarios. Having the application packaged on the controller board will require that the following functional enhancements be made to the controller firmware:

- Introduce a mechanism for updating/downloading *just* the management application image, without changing the device firmware itself. When this operation is being carried out, it will be imperative for the controller firmware to ensure that the version of the downloaded management application is compatible with the current operational version of the device firmware. A relatively flexible and extensible version compatibility management scheme must be included as part of the

packaging and download structure for this new operation. If an update of *just* the management application image occurs, no device reset should be required.

- To simplify the device firmware update/download process, implement a mechanism that requires a device firmware binary image to contain the corresponding image of the management application.

11.8.2. Device-Resident HTTP Server

In order for the device to be able to feed out its management application's binary image to a browser for execution, the device must implement an HTTP server. The server can be highly restricted in its operation, since its sole purpose is to feed out the management application. General-purpose HTTP server capabilities, such as CGI-BIN scripting, etc., can be completely omitted from the server's functional model. If other feature enhancements (beyond those of Amelia) eventually necessitate inclusion of more advanced HTTP server functions, this matter can easily be revisited and revised accordingly.

11.9. Migration of Functions from Host to Device

With the transition to a fully browser-based model, where the management software suite is fed out in a demand-driven fashion from the device to the browser being used by the administrator, there is simply no viable host location at which to run "background monitor" types of operations. In the SANtricity model, numerous such functions are installed on a host system and executed as "daemon" processes. With Amelia, it will be necessary to move these capabilities down to the device itself (at least for External RAID products). The key functions are identified below, along with proposed design directions for the migrated implementations.

11.9.1. Alerts and Notifications

Traditionally, critical events within the device are expected to trigger notification actions in the form of SNMP traps and/or SMTP (e-mail) messages to administrators. The current implementation of these capabilities is within the SANtricity software, where the SYMbol API is used to monitor for the Critical MEL Events that trigger notifications, and then to construct the appropriate form of notification and deliver it out onto the general TCP/IP network using either the SNMP or SMTP protocol. In such scenarios, the source system for the notification is actually the host where the SANtricity monitor code is running.

In order to be able to construct and deliver the relevant notifications, some configuration information must be established and saved for ongoing reference. In particular, the e-mail addresses of the desired SMTP recipients must be tracked, and the SNMP community name and other attributes for constructing the trap messages must also be saved. As the alert/notification mechanism migrates to the device itself, this information will have to be stored on the device and be made accessible via the SMI-S configuration interface for customization, querying, etc.

11.9.2. ESM Firmware Auto-Sync

In the existing SANtricity solution, this feature is implemented within the host-installed Persistent Monitor module. It operates by tracking the "expected" level of ESM firmware in each drive enclosure for a given storage array, and reacting to anomalies by checking to see if a host-resident file exists containing the expected level so that it can be downloaded to the ESM. On the assumption that the required file is present, the host-resident software initiates a standard ESM firmware download sequence to correct the problem.

In moving this function down to the device, one challenge is designing a mechanism for storing the ESM firmware files within the RAID controller so they are available when it becomes necessary to download an ESM firmware image due to a mismatch. The option of having the

RAID controller extract a “standard downloadable image” from one ESM within a pair, for use in downloading it to the other ESM, was considered, but determined to be overly complicated and risky. Rather, a mechanism will be implemented so that ESM firmware images can be deployed to the RAID controller for storage there, such that they are available when needed. There are ample opportunities to build significant RAS improvements on top of this base capability, including the following:

- Implement a version compatibility tracking mechanism so that the RAID controller has the ability to automatically download new firmware to an ESM or ESM pair, even if there is no mismatch condition, solely as a precautionary update to alleviate known incompatibility problems, or even to address critical reliability problems with the currently-installed ESM firmware.
- Institute a mechanism by which a RAID controller can take the initiative to poll a configurable web server to check for updates to ESM firmware images, rather than requiring that such images be provided manually by an administrator.

11.9.3. Support Bundle Collection

In the current SANtricity feature set, collection of a “Support Bundle” can be requested by an administrator so that diagnostic information is assembled for transfer to the LSI-ESG Customer Support organization. Such a request causes the host-resident software to sequence through a set of query operations that extracts various useful batches of information from the controller, assembles them into a compressed ZIP file, and saves them to a location specified by the user. From there, it is relatively easy for the administrator to e-mail (or otherwise transfer) the file to the desired recipient.

Also, it is possible to configure the host software so that a Support Bundle is automatically gathered and saved on the occurrence of any Critical MEL Event. Various throttling mechanisms are part of this feature to ensure that the host system is not overrun with data, to roll off excessive Support Bundle files, etc.

In general, the overall set of Support Bundle features offers many benefits in terms of product supportability and failure diagnosis. These benefits must be retained when the functions are transferred to the device itself. As a starting point, an equivalent set of capabilities must be implemented within the device. This includes:

- The ability to construct a Support Bundle on-demand, presumably as part of a (vendor-unique) SMI-S extension. This capability allows an administrative user to assemble and obtain the desired bundle for transfer to the support team in cases where a problem needs resolution.
- A mechanism for logging numerous Support Bundle files within the device so that they are available for subsequent retrieval by an administrator. The current trigger, namely Critical MEL Events, will still be used internally. Associated methods will be implemented for pruning excessive files for capacity management, throttling, etc.

11.9.4. Recovery Profile Management

The host-based SANtricity package includes a facility for logging RAID volume configuration information to a host-resident file so that it is available for use by the LSI-ESG Customer Support organization in the event of a catastrophic configuration loss by an array. The information in the file reflects key elements of the internal storage layout of each RAID volume, thereby allowing for recreation/recovery if necessary. The saved information on the host system is referred to as the “recovery profile”. Ancillary mechanisms are employed by the host system to ensure that the logged information is pruned in a reasonable fashion, and to thus prevent uncontrolled growth of the profile.

Moving this capability to the device is somewhat problematic, in the sense that a device which has lost some or all of its RAID configuration is also at risk of having lost its recovery profile, as well. However, by saving the recovery profile to a sufficiently distinct persistent storage area on the device, this risk can be mitigated.

As part of the Amelia effort, it will be necessary to address these challenges and move the recovery profile logging mechanism to the device. Additional SMI-S interfaces (presumably vendor-unique) will also be required to allow the recovery profile to be retrieved from the device for analysis and possible recovery action.

11.9.5. Drive-Channel Statistics Gathering (Periodic)

An additional function of the host-based management software is periodic collection and logging of device channel diagnostic statistics (e.g. RLS data for FC arrays). Such data is typically collected once per day and is saved to a log file on the host where the Persistent Monitor function is running. By saving this information, it can be made available to Customer Support personnel to help diagnose problems associated with back-end drive channels on an array.

As with the Recovery Profile functions described in Section 11.9.4, ancillary mechanisms are employed by the host system to ensure that the logged information is pruned in a reasonable fashion, and to thus prevent uncontrolled growth of the profile.

Again, as part of the Amelia effort, it will be necessary to move this logging mechanism to the device itself, and to implement (presumably vendor-unique) SMI-S interfaces to allow the logged information to be retrieved from the device.

11.10. Multi-Threading Model

PAS Note: This section was derived from older material developed by Ray Jantz and needs to be verified by Bill Hetrick and others.

While GWT supports programs written in Java, the language it provides is a subset of the full Java language. One of the more notable missing elements is the Java `Thread` class (because there is no analog to `Thread` in JavaScript). Multi-threading is used freely in SANtricity in order to maintain user interface responsiveness while background activities proceed. While GWT is able to accomplish the same thing, the only concurrency mechanism available to the programmer is callbacks, which basically cause a function that was identified in the remote procedure call to be invoked at the time the call completes. With this kind of an interface, the job of restoring the original context in which the call was made is up to the program/programmer. This can become tedious and complex in callback-intensive programs.

The above remarks suggest that some system or convention for imposing some structure on use of callbacks as an alternative to threading is needed. It appears to be possible to put some experience from SANtricity to good use here. Specifically, SANtricity has relied in some situations on a “task sequencer,” which is a special type of object that is capable of sequencing through a series of “tasks,” where a task is a type of object that can (1) be started, and (2) report a “done” event to its parent sequencer. With this concept, context information can be held in a parent object so that it is available to any of its child tasks, and, when an asynchronous call to the server completes, the parent sequencer can receive that event and continue to the next task. While no substitute for real threads, the task sequencer concept is superior to undisciplined callbacks.

11.11. Extension of Feature Bundle Model

PAS Note: This section was derived from older material developed by Ray Jantz and needs to be verified by Bill Hetrick and others.

With the storage array Element Manager being built and distributed as part of the controller firmware, it makes sense to apply the controller feature bundle and feature keying model to the Element Manager as well as to the firmware. In this way, the storage array management tool can be consistent with the array in terms of which features are firmware-enabled *and* exposed for management administration by applying a single enabling/disabling operation to both parts.

11.12. Prioritization of Management Request

PAS Note: This section was derived from older material developed by Ray Jantz and needs to be verified by Bill Hetrick and others.

The storage management solution should address a long-standing problem concerning prioritization of management requests in relation to other controller activities. In the present design, management requests are treated as low-priority tasks; this can be frustrating to a user when a controller is under load and a user wants to view or change management settings. Management requests typically conform to the dictum that they are not used that often, but require fast response when they are used. This argues for raising the priority of management requests in relation to I/O load.

There should be a default value that is set but changeable by the end user.

11.13. Duplex Controller Operation in Browser-Based Paradigm

When multiple controller nodes are present in the overall system, the management services are virtualized such that the browser-client accesses a single IP address, served by one controller of the managed system. The controller nodes negotiate with each other to determine the active configuration node. The active node responds to requests on the IP addresses for the web service and the CIMOM service. If the active configuration node fails or the inter-node negotiation results in a different active management node, the new-active management node takes over the service at the same IP address. Clients must not need to change URLs or refresh their view to maintain a connection with the management service.

12. APIs and Protocols

12.1. WBEM Services

PAS Note: This section is TBD. The main contributor to this section is Scott Kirvan.

12.2. SMI-S

PAS Note: This section was derived from older material developed by Ray Jantz and needs to be verified by Scott Kirvan and others.

12.2.1. Completeness

Past practice with respect to SMI-S support has reflected a lack of concern about not exposing all device management aspects. This practice developed because (a) it could ease schedule pressure, (2) proprietary interfaces existed to fill in the gaps, and (3) SMI-S was considered not necessarily strategic. The proposed strategy represents a change in thinking to the point of view that SMI-S is the primary management interface for both LSI- and customer-developed management software. Proprietary management interfaces are no longer supported, except as a strictly LSI-internal mechanism for use by an SMI-S provider (e.g. DAS). The implication that this has is the need to express *all* management data and control within the CIM/SMI-S model. One risk in this approach is increased vendor-uniqueness, a subject which is discussed further in section 12.2.3.

12.2.2. Commonality

As the term is applied here, commonality refers to the degree of uniformity in SMI-S interfaces across the various distinct storage products available from LSI. The important principle is that, even though the distinct products have different implementations supported by different organizations, wherever they have externally-visible feature elements in common, the management aspects of those elements must be exposed uniformly across the different products. Another way to put this is that the temptation to diverge a management interface within one of the product lines for sake of convenience or easing of schedule pressure must be resisted. Conformance to the commonality principle will require improved processes and communication within LSI.

12.2.3. Minimal Vendor-Uniqueness

As suggested previously, vendor-unique extensions of the SMI-S interface go hand-in-hand with feature uniqueness of LSI storage products. The challenges inherent in this include:

- Keeping vendor-uniqueness at a minimum. This entails API designers having enough familiarity with the standardized elements of CIM/SMI-S to ensure that vendor-uniqueness is not introduced unnecessarily.
- Avoiding the introduction of vendor-uniqueness for the sake of convenience. This means that developers of management software do not necessarily have as rich of an API as they have had in the past, and it may be necessary to accomplish what used to be done through a single high-level call through a combination of lower-level calls.
- Another factor that comes into play in managing vendor-uniqueness is the need for increased standards participation. Vendor-unique features that prove to have value ultimately become standardized. The challenge for LSI is to minimize the amount of rework necessary when this transition occurs. This underscores the need for two things: (1) a certain amount of agility in the development approach whereby a change of interface does

not entail a rewrite of the feature, and (2) actively promoting vendor-unique elements as candidates for standardization.

PAS Note: Bill Hetrick also included CIM-XML and WSMAN here – not sure what he wants described.

12.3. Command Line Interface (CLI)

PAS Note: The material in this section should be considered preliminary as it was picked up from existing material. Ray Jantz needs to validate this information and add to it as necessary.

In keeping with the goal of moving to complete standards-based management, it is necessary to conform to standards for command-line/scripting modes of array management as well as for conventional APIs. The DMTF has released such a standard called SM-CLP (System Management – Command Line Processing), which is outside of SMI-S, but nonetheless relevant. LSI storage products will support this standard. One of the benefits of doing so, beyond the obvious one of standards-conformance, is the fact that SM-CLP has an option for producing output in XML format, giving users a standard, readily-parsed output structure, which has been lacking in LSI storage products.

The SM-CLP standard defines the concept of a general “CLP Service” and a set of profile-specific “mappings,” which define how command line syntax elements are interpreted for the classes, properties, and methods comprising the profile. The SM-CLP standard effort is presently focused on server management as opposed to storage management; this means that, in order to use the defined CLI interface for storage management, an independent effort is needed to develop the mapping definitions describing how the SM-CLP verbs and keywords are interpreted for storage but it is highly likely that this will be accomplished through a SNIA-initiative.

The main deliverables in regard to SM-CLP support are (1) a C++ implementation of the standard SMI-S and LSI vendor-unique mappings for the supported storage devices, and (2) a set of free-standing executables that implement the SM-CLP “verbs.” The latter deliverable is needed because, as defined, SM-CLP has a shell command processor with no support for control structures (e.g., if-then-else, etc.); this is inadequate for writing meaningful scripts and is a regression compared to the existing SANtricity. Therefore free-standing executables that can run under the user’s shell of choice are necessary. The choice of C++ is based on (1) the belief that it is a better fit with SM-CLP infrastructure than, say, Java, and (2) the performance issues that were encountered with the SANtricity Java-based implementation. The C++ code needs to be highly-portable so that it can run on a variety of operating systems.

12.4. Management API Transport Mechanisms

PAS Note: This was included in the original PAS outline and assigned to Ray Jantz. Ray needs to decide if this has been covered elsewhere or is no longer applicable. Information from section 4 of the Future Storage Management White Paper was targeted for inclusion in this section.

12.5. Software Development Kit (SDK)

PAS Note: This was added by Bill Hetrick – not sure what needs to be put here.

13. Google Web Toolkit (GWT) Overview

PAS Note: This was included in the original PAS outline and assigned to Bill Hetrick. Bill needs to decide if this has been covered elsewhere.

14. Browser Overview

PAS Note: This was included in the original PAS outline and assigned to Bill Hetrick. Bill needs to decide if this has been covered elsewhere. Information from section 8.3.2 of the Future Storage Management White Paper was targeted for inclusion in this section as well as any important browsers considerations that need to be kept in mind.

15. Management Security Overview

PAS Note: This is the straw man version of the Amelia security information. It is an attempt to articulate a cohesive security strategy for all Amelia management features in advance of any deep research. Additional research and prototyping will drive revisions and corrections to this material. In particular I intend to prototype LDAP access from a device and the SSO server. The impact/applicability of the "partner framework" package for the Enterprise Console has not been considered in the current write-up – Scott Kirvan

The Amelia program is to include a comprehensive upgrade to the security of the storage device management interfaces and enterprise management software and utilities. Security updates are focused on creating a more cohesive enterprise security environment. Specifically they include: authenticating users with LDAP directory servers, end-user supplied x.509 certificates, and supplying enterprise wide single sign-on access to LSI storage management software. Note that Amelia does not address the security of user data stored or accessible through a given device. It is strictly concerned with access to the management features of the storage devices. Terms to know in reading this section include:

- **SSO –** Single Sign-On
- **SSL –** Secure Socket Layer
- **LDAP –** Lightweight Directory Access Protocol
- **AD –** Active Directory
- **TPC –**Total Productivity Center

15.1. Authentication & Access Control

Authentication in the context of this document refers to the identification of end-users seeking access to storage management functionality. Access control concerns the allocation of permissions to authenticated users. The availability and form these aspects of security take depends on the specific storage device or application and the management context (i.e. stand-alone or enterprise).

15.1.1. Stand-alone Device

Storage arrays and virtualization appliances currently authenticate users against a device resident authentication repository. Users supply a username and password and are granted a token if their credentials can be validated. The token grants access to management features of the device, eventually timing out after a period of inactivity.

In Amelia the current mechanism is to be maintained along with an extension to allow integration with an LDAP server. With the Amelia extension in place, authentication of an end user takes two forms: local and network. Local authentication proceeds exactly as it does in the pre-Amelia case. Users supply their credentials which are validated against the local authentication repository and a token granted if the credentials are good. Local authentication is used if network login is not configured or as a secondary source of authentication when network authentication fails.

For network login to be available the device must be preconfigured with access information for the LDAP infrastructure. A network login will validate the supplied username and password against the pre-configured LDAP server. The characteristics of network login are given in the list below.

- Users still have local accounts and access control is handled in accordance with local permissions. While LDAP is perfectly capable of and often used to store access control

information, the practice is not standardized and so LDAP is used only to authenticate the user.

- When presented with end-user credentials the CFW network login process is as follows:
    - Verify the user has a local account.
    - Using pre-defined LDAP information, connect to the LDAP server and search for an LDAP user that matches the supplied end-user credentials.
    - If a matching LDAP user is found, attempt to bind to the user with the supplied credentials. If the user can bind to the LDAP server the authentication process is complete. The user is issued a token as with local authentication and has the permissions associated with the local account.
    - If the user can not be found in LDAP or the network authentication fails for other reasons proceed with local authentication for the user.
- Access to a given device via a network login does not imply access to any other device. Access to a second stand-alone device requires that the end-user supply the credentials (in the network case, the same credentials) when attempting to access the new device.

15.1.2. DAS

Historically, access to the management features of a DAS (either MegaRAID or SCG IR) controller are controlled by the host operating system. That is, one must have an administrator account on the host to access or modify the configuration of DAS storage attached to the host. WBEM access to management features of DAS is usually controlled using this same host OS mechanism. Configuration of the authentication mechanism for WBEM access is not part of the installation or configuration of DAS products. While administrator access is the norm, the actual configuration is completely up to the system administrator of the server.

In an Amelia environment gaining access to the DAS element manager will be driven by the configuration of the host environment. If the WBEM service requests a username and password the element manager will in turn request them from the end-user. Authentication of DAS management requests will not (can not) be integrated into the LSI SSO server. Whether the supplied username and password are validated with an LDAP service or some other standard mechanism is controlled by the WBEM service and host operating system. No access control is performed; once logged into the host WBEM service, all management activities are allowed.

15.1.3. Enterprise SSO

PAS Note: Highly speculative material. Enterprise SSO may not be needed or desired in Amelia. The detailed information is suspect as well –there are a number of existing SSO software packages and algorithms to be considered. The mechanism below is a jumble of ideas from Kerberos, CAS, LTPA, JAAS, etc., the details still need to be sorted out – Scott Kirvan

Using LDAP accounts to authenticate storage device users is the first step towards an enterprise authentication environment. It allows users to use the same credentials to access any given storage resource but they must be presented each time they access a new resource (e.g. every time they launch an Element Manager). Single sign-on (SSO) takes that to the next level –users present their credentials once and the SSO service handles individual logins to storage resources.

SSO in Amelia is an optional component integrated with the Enterprise Console (EC), although all solution components require some modification to interact with the SSO service. When the SSO service is present, all (most?) services affiliated with the EC defer authentication to the SSO service. The SSO service has the following general characteristics:

- Attempts to connect to an EC web application by a new user are forwarded to SSO when it is present.

- SSO examines HTTP message headers of forwarded requests; message headers of users already logged into the system contain a “ticket”.
- If no ticket is present the SSO requests credentials from the end-user. Credentials are authenticated using an external authentication mechanism such as LDAP. Once authenticated the user receives a ticket in the HTTP header of the authentication response message.
- Tickets uniquely identify the user, shared secret, life-span of the ticket and other information. The ticket is encrypted and only the SSO can decrypt it.
- Attempts to connect to a EC web application by a user already logged into the SSO service contain a ticket. The web application validates tickets it receives from users by using a validation service of the SSO.
- Tickets are short lived, presenting an invalid ticket results in the user being forwarded to the SSO login mechanism.
- Applications not resident in the EC can be configured to use the SSO as their source of authentication.
- HTTPS is the preferred mechanism for all SSO related communications. It will be possible but not recommended to handle SSO communications using HTTP.

15.1.4. External SSO Integration

Current integration with the TPC SSO implementation is maintained.

15.2. Communication Security

Communication security in Amelia is based on the Secure Socket Layer (SSL). SSL secured communication channels are the preferred access mechanism for all devices and software components in Amelia. While SSL secured channels are preferred, users will not be prevented from configuring their storage management infrastructure from using unsecured channels. SSL does impact management data transfer speeds between components. Whenever possible (i.e. when the communication protocol is HTTPS) communication channels will be configured to remain open for a substantial period of time and support the transfer of multiple messages. This minimizes the impact of SSL on data transfers by limiting the number of times the SSL “handshake” takes place in a given transfer (hopefully to 1).

15.3. Public Key Cryptography

Public key encryption is difficult to deploy safely without requiring substantial interaction from end-users. Attempts to lessen the quantity or complexity of end-user interactions invariably increase security risks. As such there will be two approaches to public key cryptography in Amelia: a default approach requiring little user interaction that supplies reasonable security, and a more secure approach requiring substantial end-user participation.

15.3.1. Default X.509 Certificate & Key Creation

If the end-user declines to configure the public key encryption components the default approach is taken. The default approach is comprised of the following:

- Components acting in a server role (storage arrays, enterprise console, etc.) generate an RSA private key with a length of 2048 bits.
- Sever components use the private key along with the DNS name of the server and other static information to create an X.509 self-signed certificate. This certificate will be presented to clients attempting to make an SSL connection to the server.
- The server’s private key is retained on the device or server and is protected by the device’s authentication (e.g. password required to get into the CFW shell, or administrator account on a server) mechanisms. There is no interface that allows the private key to be retrieved.

- Clients receiving the certificate for the first time may find it objectionable because it is self-signed and in some cases (storage arrays without a DNS client) may not include accurate DNS information. The end-user is required to reconfigure the client to accept the certificate. On a browser this means clicking through a series of ominous dialogs that suggest that the end-user is being asked to do something unreasonable. Some 3rd party clients may not allow the user to accept the certificate and be unable to interact with the server or device at all.

Self-signed certificates with inaccurate DNS information do little to assure a client that they are in-fact communicating with an LSI storage array at a given address. A spoofing/hijacking type attack is possible when default certificate generation is used.

15.3.2. End-User X.509 Certificates & Keys

Because the default array and/or storage server generated X.509 certificates may not be suitable for all security environments, it will be possible for an end-user to upload both a certificate and a private key. An end-user managed certificate environment has the following characteristics:

- Storage devices and storage management severs support an interface that allows an end-user to supply both a certificate and a private key in PEM format. The certificate may (if fact should) be a certificate chain in which case all certificates in the chain are present in the one certificate file.
- Uploading of the private key occurs across a secured or local connection.
- Access to key and certificate related functionality is limited to those with administrative accounts. No mechanism is supplied to retrieve a private key; it can be replaced or deleted but not retrieved.
- Persistent storage of private key values uses the most protected means possible for the given platform.
- Keys stored on a storage management server may be encrypted at the discretion of the end-user. Encrypting keys generally means an administrator must be present to enter a password when the software starts up and so is not appropriate for all environments.
- Once the user supplies a certificate and key the default certificate and key are deleted. The end-user supplied certificate is used in all SSL interactions.
- The end-user is responsible for any actions required to make the certificate acceptable to all relevant clients. This generally means loading the certificate into browser/client trust stores or using a signed certificate whose issuer is already trusted by the client.

End user supplied certificates, done correctly, over come the objectionable qualities of the default certificates. The client is assured that the certificate subject has access to the private key making a spoofing attack unlikely.

15.4. Other Security Considerations

PAS Note: This information was picked up from section 5 of the Future Storage Management White Paper. Ray Jantz and Scott Kirvan need to verify the following information.

Each Element Manager implements security and user-based roles using the SMI-S Security profile (and associated sub-profiles). The Element Managers implement a security and user role model that matches or closely matches the scheme being introduced in the external RAID Flint release as follows:

- Require a user login and password to access the associated Element Manager.
- Provide for assignment and modification of user groups and individual users.
- Provide for assignment and modification of access rights/permissions.

- Provide for assignment and modification of user capacity quotas (i.e., how much capacity a particular user is allowed to configure and manage).

Note: The optional Enterprise Console does not implement its own security model. Rather, if there are any options in the Enterprise Console that create or change attributes on an individual device, then the user is prompted for an appropriate login/password for that particular device/option. ***TBD*** *on how we would handle multi-device operations invoked from the Storage Console or from the CLI.*

Note: The Security profile is currently tagged as *Experimental* meaning that the contents of the profile may change. Issues not currently covered include threat models, protection against specific attack vectors, (such as denial of service, replay, buffer overflow, man in the middle, etc.), topics related to key management, and data integrity. Development of threat models, and specific attack countermeasures required for robust security elements, such as integrity has been left for future work. A description of the Security sub-profiles follows:

- **Authorization Sub-Profile** - this sub-profile extends the Security profile. The Authorization sub-profile specifies base support to enable management of the rights of particular subjects to perform specific operations on selected target elements within a CIM Service.
- **Credential Management Sub-Profile** - this sub-profile provides for management of credentials used by a client to establish its identity to a serving system. An administrator of both the client and server systems establishes an Identity for the client on the server system and creates a credential for the client on the client system.
- **Security Resource Ownership Sub-Profile** - this sub-profile provides the means to model restrictions on CIM operations associated with exclusive use of a resource, For instance, a storage volume in an array. It is intended for environments in which multiple CIM clients may not be completely aware of each other's activities, making it important that use of the resource not be disrupted by a client that is unaware of its use. Specific examples include use of a volume by storage virtualizers and NAS gateways, where attempts to manage the volume by clients not associated with this use could be seriously disruptive. An intended configuration is that a CIM client exists in the cascading device that has exclusive use of the volume, although this is not strictly necessary. *Note that this sub-profile may not be applicable for our needs*.
- **Role-Based Access Control (RBAC) Sub-Profile** - this sub-profile enables management of authorization using RBAC Roles.
- **Identity Management Sub-Profile** - this sub-profile provides support for adding and managing users of a system and for mapping those users to accounts, people and organizations. Users are assumed to have Identity instances to represent their ability to be authenticated. Identity instances may stand alone or may be linked to Accounts, Organizations, OrgUnits, UserContacts, Persons, Groups or Roles. Accounts are used for the purpose of authenticating Identities and may additionally be used as a basis for tracking other information about the use of a system by a particular Identity.
- **3rd Party Authentication Sub-Profile -** This sub-profile extends the Security Identity Management profile by specifying the necessary elements required to manage the relationships between a CIM Server and 3rd party Authentication Servers such as Radius.

16. Launch-in-Context (LiC) Definition

It is imperative that every reasonable functional element of the GUI is an independently launch-able view. This supports the objectives associated with the OSA strategic initiative as well as being able to integrate with other higher-level applications offered by our OEMs. More specifically, it must be possible for all such functions to be activated from a *separate* application (i.e. other than the mainline Amelia GUI) with sufficient means to convey user context information so that the precise operational settings associated with the user's needs can be established by the activated component. This structure will make it possible for independent, higher-level management applications to be adapted to invoke these same functional components, and to do so with sufficient user context that the invocation/activation process is nearly seamless from a usability perspective. As a result, higher-level applications can be modified in very simple ways so that they can offer a fully-integrated solution that encompasses both the higher-level application functions and the lower-level device management functions.

16.1. LiC Overall Structure

The basic foundation of this Launch-in-Context approach is illustrated in the following figure. The block labeled "Top-Level Interface and Task View" represents the mainline GUI, while the various smaller blocks below it show the (hypothetical) set of functional elements that are implemented as Launch-in-Context components. The complete Amelia GUI, i.e. the "LSI Element Manager", therefore would consist of the union of all of these blocks. In this case, any invocations of lower-level components by the main-line GUI could be considered "Internal Launch-in-Context" actions.

The block labeled "Baseline OEM/Partner Storage Application" represents some broader storage application associated with a higher-level function that is integrated into the RAID controller environment. Since such an application would clearly require the ability to manage block storage, the application can be readily adapted to invoke any relevant subset of Launch-in-Context components that are part of the Amelia solution. In this case, such invocations of lower-level components could be considered "External Launch-in-Context" actions. The complete solution, which includes the base storage application *and* the relevant Amelia Launch-in-Context components, is labeled "Composite OEM/Partner Application".

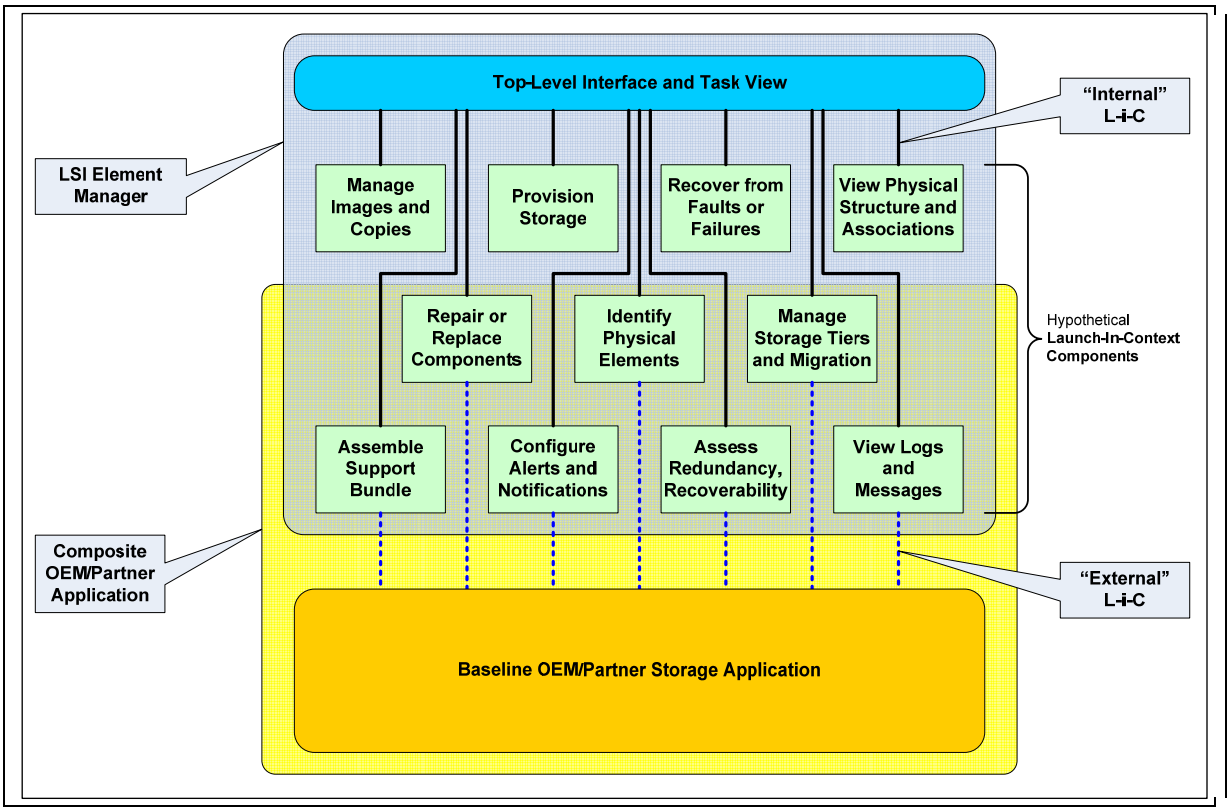


Figure 1: Conceptual View of Launch-in-Context Framework

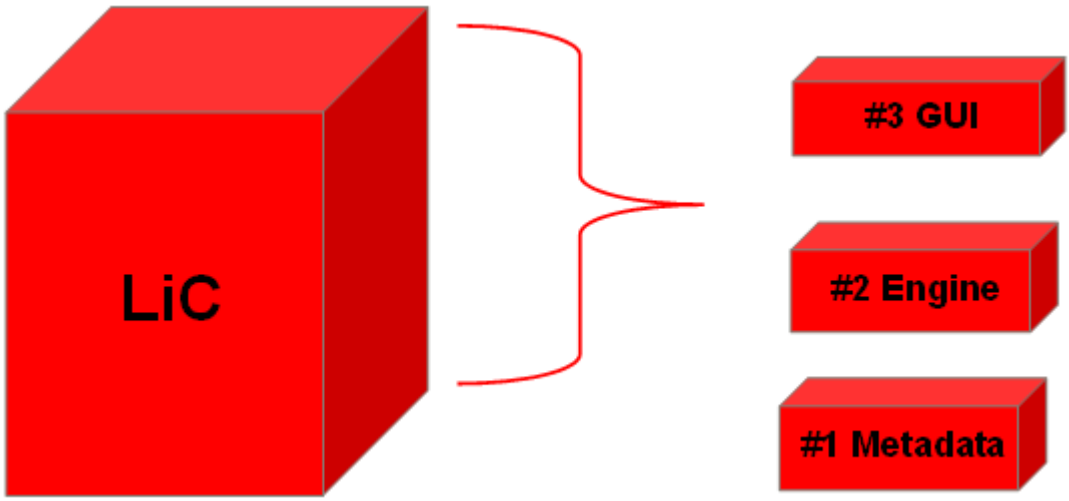
The mechanism for how to build the Amelia application with Launch-in-Context capabilities has been investigated and discussed. The guiding principle is that the Amelia application must be optimized to run as a complete application while simultaneously meeting the Launch-in-Context requirement.

Note: An alternate approach discussed was to optimize for launch-in-context, developing the GUI to build a set of independent management views, stored as separate html files on the server with a framework that ties the views together. However this model becomes cumbersome when dealing with shared SMI-S data and the browser's management of that data because each stand-alone HTML page may need to act as an SMI-S client, collecting the necessary data and maintaining a consistent system state.

Rather than a collection of html files, the application should be architected at the GWT-Java code level as a set of views that are aggregated by a framework, but all views share the same SMI-S data management layer. When the Amelia application is launched, the GWT code determines the views to present based on the URL that was issued. Views register with the client-side SMI-S data management layer such that only the relevant data for the presented views is loaded from the managed device.

16.2. LiC Individual Structure

An individual LiC component is comprised of the following three elements:



- Element #1 (Metadata) – this is descriptor/property information that describes the individual LiC component. This metadata must be able to be customized by OEM partners. The actual mechanism for storing the metadata is described in section 7.21. The metadata provides three key pieces of information for each LiC component:
    - (1) The overall life cycle stage/function group this LiC belongs to. This provides a development organization or higher-level application the flexibility to integrate a single LiC or all of the LiCs related to a life cycle stage.
    - (2) Keywords about the LiC. These keywords are used in search functions to help the user locate tasks. These keywords must include various alternate synonyms to accommodate different user models (for example, a keyword would be volume but an alternate word would be LUN).
    - (3) Indication of the experience level of the function (i.e. level 1 - basic, level 2 - moderate, and level 3 - advanced).
- Element #2 (Function Engine) – provides the core function needed (the API into function is SMI-S).
- Element #3 (GUI) – provides the UI view to allow an end user to manipulate the function.

The LiC component is structured to provide elements #1, #2, and #3 or just elements #1 and #2.

The flexible structure of the LiC allows a development organization (OEM or Internal) the ability to pick up all elements or write their own UI view (i.e. only using elements #1 and #2) to better conform to their specific framework and UI behavior.

16.3. LiC Elements

PAS Note: As part of the AAD development, a list needs to be developed of all LiC components. An LiC is essentially a management interface, so it must be carefully constructed, specified, and considered contractual.

16.4. LiC Behavior

PAS Note: This section is TBD. The main contributor to this section is Bill Hetrick. Rough notes follow.

- OnModuleLoad
- Look at the URL
- Load the SMI-S data appropriate for the requested operation:

- Framework: load all logical, physical
- Share view: load shares, pools
- VHD view: load vhds, pools
- Specify onCancel, onSubmit URLs
- How to return success or failure to calling application? Through URLs

16.5. Variation Control

PAS Note: This section is TBD. The main contributor to this section is Ray Jantz. This section needs to provide a brief explanation of how we will handle variation control relative to building of the various LiC components. The following information is derived from older material and needs to be validated.

It is important that the Element Manager build and runtime infrastructures support flexible variation control mechanisms. An appropriate variation control infrastructure should support a "component assembly" style of application construction using a combination of generalized and specialized components as product requirements dictate. Needs that are driving this sort of infrastructure include:

- Code reuse among the different Element Managers, especially in the RAID space,
- Reduction of storage footprint for device-served Element Managers by excluding code that is irrelevant to the device that is serving it,
- Configurability to different experience levels for different price bands, and
- Better alignment of management software with a Product Lines development approach

In general, the kind of structure that is desirable for the Element Managers is to have a basic core platform that incorporates an infrastructure for SMI-S data and control access, along with a basic GUI "shell." This can form the basis for attaching modularized "feature support" elements. Ideally, the "attachment granularity" would be relatively fine, allowing for attachment of tab panes, menus and toolbars, menu and toolbar items, hyperlinks, etc. While the Google Web Toolkit does not include a ready-made full-featured infrastructure for this sort of thing, it does have some elements that could be helpful, including (1) support for "deferred binding," a variation control mechanism that operates at compile time, and (2) support for modular application assembly through use of jar files.

17. Hardware & Storage Requirements

PAS Note: This section is TBD. The main contributor to this section is Bill Hetrick. This section needs to provide an explanation of hardware requirements needed by Amelia as well as data persistence and storage requirements.

17.1. Hardware Requirements

Memory, Processor Power, HTTP Connections, etc. Take into account both embedded environment for Element Manager as well as management server for Enterprise Console.

17.2. Data Persistence & Storage Requirements

- Use of cookies for non-critical preferences.
- Use of persistent storage for user preferences that need to persistent regardless of the client that the user uses to access the Enterprise Console or Element Manager.

18. Convergence

Wherever possible, the various Element Managers that conform to the Amelia architecture must converge on as many common elements as possible (this will not only aid an end user that moves from one Element Manager to another but also increase the overall efficiencies of the internal development organizations from a re-use perspective).

PAS Note: The following is a preliminary list of common elements. A complete list must be documented in the AAD. The main contributor for the converged elements is Scott Hubbard.

- Main Enterprise Console (EC) and UI behavior
- Dashboard/Summary View Framework
    - Common portlets of information where possible
    - Unique portlets for each platform
- Alert levels and Alert user manipulation
- Recovery Guru framework
- Main Element Manager framework and UI behavior
- Statuses
- Event logging scheme and Event viewer
- User-defined folders
- Wizard sequences and layout
- Browser-based
- Use of GWT
- Modular software structure
- Managing of alarms
- Save/Load configuration operations
- User group definitions
- User permissions
- Hardware operations (locate, prepare for removal, battery settings, etc.).

PAS Note: This section is also supposed to discuss the architectural “plumbing” differences between External, DAS, and DPM. The main contributor for this information is Bill Hetrick.

19. Remote Management

PAS Note: This section is TBD. The main contributor to this section is Ray Jantz. This section covers lights-out management, phone home capabilities and other aspects to allow an end user or service person to perform remote management.

20. Reliability & Availability Architectural Definition

PAS Note: This section is TBD. The main contributors to this section are Ray Jantz, Scott Kirvan, and Bill Hetrick.

20.1. Tracing Mechanisms

20.2. High Availability Requirements

20.3. Configuration Consistency

Note From Bill H. - With Windows Storage Server, file shares are (sometimes) not restored after Windows reboots. We may need to maintain a file that reflects the system configuration and check the system configuration against the file on system boot. (This configuration file could be used to feed the SMI-S data model.)

20.4. Configuration Locks

Notes From Bill H:

- Prevent two clients from making configuration changes where the underlying operations could be interleaved
- Master SMI-S server?
- Management Ethernet address is a cluster resource – owned by one server card at any given time.
- Use read-lock and write lock?
- Multiple readers
- Exclusive writer
- Use virtual configuration service
- HOW CAN THIS FAIL – stuck locks?

20.5. Transactional Processing

20.5.1. Configuration Error Recovery

Notes From Bill H:

- If a configuration request partially completes but fails some operation, the preceding successful operations must be 'undone' to return the system to the original initial condition
- The configuration request could span several applications of an OSA platform e.g. provision storage (storage), create file system (OS), export share (cluster).

21. Overall Service Architectural Definition

PAS Note: This section is TBD. Contributors for this section have not been identified.

21.1. Overall Service Philosophy

21.2. Versioning Strategy

22. Host-Based Utilities & Agents

PAS Note: This section is TBD. Contributors for this section are Ray Jantz and Yanling Qi.

22.1. LSI Utilities

22.2. VDS/VSS Support

PAS Note: Not affected by Amelia???

22.3. Other Plug-Ins

23. Multi-Pathing & Failover

PAS Note: This section is TBD. Contributors for this section are Ray Jantz (with help from Bob Stankey). The operational models may best be put in a different document, but the installation and movement to no LSI componentry must be discussed here.

24. Test Architecture

An effort should be made to describe how the concept should be designed to improve testability effectiveness and efficiency. In most cases, this may involve discussions with LSI and OEM customer test architects.

PAS Note: This section is TBD. Contributors for this section are TBD.

25. Product Roadmap & Phasing

PAS Note: This section is TBD. Contributors for this section are TBD.

26. Future Considerations

PAS Note: This section is TBD. Contributors for this section are TBD. Some rough notes follow.

- BVL storage allocation manager (SAM)
- Web technologies
    - Flash, Sliverlight,
    - Video
- SMI-S
    - backward compatibility (IBM voiced a number of complains about profile changes that affected their applications)
    - vendor unique extensions
- OSA
    - collections of applications
    - virtual machines
- Yuri
- CRUSH
    - The notion of storage pools as an object that is manipulated by the end user is abstracted as much as possible from the end user. For an end user, the starting point is the host-addressable volume. Based on the volume QoS attributes presented and requested by the user, the appropriate storage from a pool is allocated. Even if there are multiple drive types in a storage system (essentially necessitating different CRUSH pools), the user views the available storage as one large pool.
    - Hot spare drives are probably no longer needed.

27. Integration Definition

PAS Note: This section is TBD. The main contributor for this section is Bill Hetrick.

27.1. Framework Integration

27.2. OSA Application Integration

27.3. Pluggability

27.4. Virtual Machine Management