# Athena Layering Requirements

Mark Rosenstein

March 17, 1993

This document is presented as the requirements for the Layered Athena project. This introduction explains a bit more about what the project is, since the Opportunity Evaluation was insufficiently clear. The idea of layering Athena on vendor operating systems is not new, but until now we have not really addressed it. Now is the time to do this as an independent project. If we don't take this opportunity, we will be forced into it by the small decisions that we make as we port the environment to new platforms.

**What is Layered Athena?** By any account, Layered Athena is a clean separation between the vendor operating system and Athena's value added software.

The Layered Athena project has three motivations:

1. To make it easier for private workstation owners to use Athena,
2. To make it easier for us to port Athena to new platforms and release new and updated software.
3. To do good computer science.

These varying viewpoints, which are described more fully below, make it difficult to get the idea across and make sure we are all thinking of the same thing. While these motivations are not mutually exclusive, they do suggest different approaches to the problem.

The two primary motivations of Layered Athena suggest different tradeoffs and focus to this project. We realize that at this point both of these approaches are strategically important to Athena. For now we will assume that simplifying our release engineering process is the primary goal, with making Athena more accessible to private workstations secondary. Hopefully we will be able to do both, but that focus gives us a better chance of completing the first phase of this project and positioning DCNS for further work.

1. Cleanly layering Athena on the vendor operating system will give the private workstation owner more flexibility. Making it easier for private workstation owners to use Athena will in turn make Athena more available in laboratories and departmental computing centers. Layering will allow us to add Athena functionality to an existing workstation and have everything continue to work, even if we have not previously tested that exact hardware configuration. We should also be able to remove the layered Athena product and return the workstation to a working configuration. Layering enables us to keep up with new hardware

as fast as the vendor supports it. If the workstation owner needs software support from the vendor, he can temporarily remove Athena to show whether the problem is in our or their software.

2. A clean layering can also make it easier to maintain the Athena environment. By not releasing the vendor OS and Athena software in one huge combined package, it is likely that ports of the environment will be easier and faster. Currently, when porting the Athena environment to a new platform, actually getting the bulk of the code to compile and run is a small part of the effort. A larger part of the effort is integrating everything with the vendor OS and properly configuring it. Two pieces are particularly difficult to port: login and the installation process. Layered Athena can help simplify a lot of the integration/configuration work. By partitioning the environment into packages, it becomes possible to release a port before the difficult parts like login are done[1]. Layering would simplify regular releases on existing platforms as well.

3. Finally, layering is just plain good computer science. This was the original motivation for the developers suggesting it several years ago. Making a clear boundary between a standard (the vendor operation system) and a local modification (the Athena changes) is a good idea that makes an aesthetically better system. It also makes it easier to support and maintain that system. Changing the architecture solely for aesthetics is insufficient justification for the work required. However combined with the benefits of easier porting, easier maintenance, easier use by private workstation owners, aesthetics provide compelling justification.

**Are the goals exclusive or can they work together?** This project can have such diverse goals because it is a basic reorganization of past work which can have a lot of impact on the system. A number of other major features of Layered Athena fall out from the above goals. One is actually identifying what Athena is, what software does this apply to and what must be present on a workstation to consider it to be running Athena. Another is partitioning Athena into pieces that can be loaded on a workstation or ported to a new platform separately from the rest of the suite. An important variation on the basic goal of being able to layer Athena on the vendor OS is for the end user to be able to use Athena services on a workstation that the system manager has not already setup for this. Finally, a long-term goal for this whole project is for public workstations to be a special case of Layered Athena, rather than built with a different scheme.

**Deliverables.** The primary deliverable from this project will be a framework for packaging the Unix Athena release. Specifically, we will produce a sample port for one platform with a well-known vendor OS, such as Ultrix 4.3 on the DECstation, along with the necessary tools to install, update, and remove that software.

Due to the past difficulty in gathering requirements and communicating with customers for similar projects, we are taking a different approach with these requirements. This document was produced internally without further data gathering. We are now seeking comments on it from the potential customers of the layered Athena: both our own operations groups

[1] One can work around not having an Athena login by using the vendor login program and running an Athena startup script once you are in.

and users in the wider MIT community. The requirements presented here attempt to cover both the technical and support sides of the project.

1. Divisibility: the Athena release will be divided up into smaller pieces, herein referred to as subsets. See example list of subsets at the end of this document.

   (a) Subsets are logical units which may be a single program or may include many directories of fonts, libraries, utility programs, and other support files.

   (b) Use of some subsets may require the inclusion of others. These dependencies must be made explicit, and form a hierarchy of subsets.

   (c) For convenience, some popular groupings of common subsets may be created.

   (d) The number of different layers and subsets must be small enough that it can be easily comprehended by a user whose primary interest is not computers. We expect about ten subsets to be defined.

   (e) It is not necessary that all subsets be available on all platforms. While it is desirable to have them the same, this makes it possible to begin using a port which is not yet complete.

2. Placement: the workstation owner should have the choice of placement of the system software.

   (a) Not having a given subset available at all.

   (b) Being able to use a subset that resides on the fileserver. This may require placing symbolic links or configuration files on the local disk, while leaving as much as possible behind on the file server. The fileserver being depended upon may be a centrally managed Athena server or a local departmental server.

   (c) Copying all parts of the subset to the local machine.

   Note that some subsets, such as those affecting basic networking on the workstation, may only make sense copied to the local disk. We also may want to limit some of these choices to make Layered Athena easier to support.

3. Updates: as with the AUTOUPDATE variable in the current workstation scheme, the workstation owner must have the option of:

   (a) His workstation is updated automatically whenever a newer version of a subset is available.

   (b) He is notified that new software is available, but must take manual action.

   It should be possible to back out of an update or even remove all of the layered athena products from a workstation. We may want to limit some of these options to make it easier to support this software and encourage users to stay up to date.

4. Conflicts with Vendor operating system. Whenever possible, we should avoid conflicting with similar systems provided by the vendor.

   (a) Use of Layered Athena on a workstation must not preclude the loading of any third party software on that workstation.

   (b) If we provide an enhanced version of a program the vendor also provides, we must not remove the vendor version from the system, merely rename it or put ours earlier in the users' search path.

   (c) If our enhanced version lacks some feature that third-party software may rely on, then we cannot move the vendor version at all.

   (d) Wherever possible, locally developed programs, as well as enhanced versions of vendor programs, should not depend on availability of vendor sources.

5. Tracking: we need to know what people are using so that we can provide better service and know when it is safe to decommission old versions.

   (a) Optional real time monitoring such as SNMP should be able to tell us what subsets are loaded. While enabling this is optional, it must be done in such a way that most people opt to do it.

   (b) The install and update tools should record what is on a workstation so that recreating that workstation or cloning a configuration on many workstation is easier.

   (c) A method must exist to locally determine how a workstation is configured.

   (d) We should encourage, but not require, the registration with I/S of those workstations using Layered Athena.

6. Scale: if someone manages a number of similar machines, it should be possible to configure one, then duplicate that configuration on the others. If a central database is used to store configurations, it must easily handle tens of thousands of workstations. Similarly, any automatic update mechanism must be able to handle tens of thousands of workstations updating at the same time.

7. Security

   (a) Since someone with root privileges can do whatever they want to a workstation, security really only applies to the stable storage of configurations.

   (b) An unprivileged user may be allowed to temporarily use some Athena software. While the system manager may not want to Athenize a workstation, the users may still be able to use some parts of the Athena environment without any privileged or permanent changes. Such a change must be simple and quick to install and deinstall.

8. Licensed Software: Important parts of the Athena software suite are licensed third party software. It is important that making layered Athena available not easily allow people to copy licensed software off campus or otherwise violate licensing terms. We also charge for access to this software. We must make sure that people pay when they "Athenize" a workstation.

   For now we will move forward with the Layered Athena project without concern for technical solutions to enforce software licensing. A team will be formed to examine the policy issues, and determine if it is purely policy or if technical support is necessary. In any case, such a technical solution would be outside the scope of Layered Athena, as it would apply to regular Athena installations as well.

9. Customization: The workstation owner may want a particular subset with a simple local customization. There must be provision for running a local customization script at install and update time to handle changes.

10. User Interface: a user interface must be supplied so that the average workstation owner can customize their workstation without calling for help from I/S. This interface must not make any assumptions about what is already on the workstation (i.e. X Windows) so that it can run on the raw vendor's software. The interface should attempt to verify that the person running it is the owner of the workstation and they are authorized to put "Athena" on this workstation, then let them configure it however they want.

11. Support: we have to be prepared to support these. To be able to provide reasonable support, we need:

    (a) Explicit written policy. With layered Athena it will become much more fuzzy what is and isn't an Athena workstation. We need Service Level Agreements with private workstation owners so that they know what they can expect from us, and what we require of them to provide service. This must be clearly communicated with the workstation owners. We also must consider how the Athena Rules of Use apply to Layered Athena.

    (b) A way to load software. If layered Athena requires the vendor OS to already be installed, then we need to be able to help workstation owners install the vendor OS as well.

    (c) Problem resolution. How do we deal with finger pointing between vendor customer support and our software, especially when billable calls are involved?

    (d) Training for support personnel. Because this makes it easier and faster to port and change the Athena environment, the developers have to be especially sensitive to keeping the support organizations informed and trained in a timely manner.

    (e) Identifying the configuration. A simple method must exist for support personnel (e.g. OLC consultants) to know the machine configuration when they attempt to help a user.

(f) Documentation. We need to prepare documentation how to install and update systems for the end user, a description of each of the subsets, and an administrator's guide for Athena Operations personnel and others who want a lot of detail.

(g) Training. No end-user training courses will be necessary for installing and updating Layered Athena, although this may increase interest in how to use Athena.

As is probably clear from reading the requirements, the solution is both technical and procedural. In addition to programming the pieces necessary to pull this together, we will have to document it and identify how we deal with special cases, how we support customers using this, and many other things.

## Appendix: Subsets

This is a proposed list of subsets to support.

**Kerberos** Support to run kerberos clients. This includes a couple of configuration files and a small set of utility programs.

**AFS** Transarc's AFS file service. This one will be tricky, as it consists of kernel extensions, daemons, modifications to the system startup scripts, several configuration directories, and allocation of a large amount of disk space. It requires the kerberos subset to function.

**Basic Athena Services** These are the core services that the rest of Athena depends upon: hesiod, zephyr, moira, attach, and email. This requires several configuration files, a couple of daemons, and a number of utility programs. Email here means the ability to send outgoing mail, and possibly POP support in some simple mail readers. It requires the kerberos subset to function. Note that at this point, people can attach lockers and run most third party applications.

**X** This is the locally built clients, and on some platforms, server as well.

**Login** This is the usual Athena login program, which will allow anyone with an Athena account to use the workstation. It requires the Basic Athena Services, and X if the workstation does not already have an acceptable X implementation.

**Help** Online documentation and consulting. This includes olc, olh, the man pages, and whatever other help facilities we have available. This requires the Basic Athena Services.

**Tex** This includes the TeX and LaTeX formatters, support programs, fonts and macro libraries. It does not depend on anything else.

**Andrew** This is the editor EZ and the other programs in that suite. It includes a number of config files, fonts and utilities as well. It also does not depend on anything else.

**Athena Applications** These are the rest of the Athena applications. Notably, it includes emacs, discuss, delete, the Athena csh, mh, and many more. No other subsets are required, although some applications will require Basic Athena Services.

**Public** This includes the remaining configuration files and pieces necessary to turn a workstation into a public Athena workstation. It requires all of the other subsets.