Zones Upgrade Project

Design Specifications

Revision 3.3

Friday, August 5, 2005

1.0 Introduction

1.1 Overview

This document describes a problem the Solaris containers feature imposed on the Solaris upgrade feature starting with Solaris 10 GA (3/05), and details a plan to remedy the situation for Solaris 10 update 1 (12/05) and beyond. This plan is referred to as the *permanent solution* to help distinguish it from other potential short term solutions that are referred to as *interim solutions*.

1.2 Table of contents

This document is divided into the following sections:

1.3 Solaris 10 GA (3/05) introduced *zones*

Solaris 10 GA (3/05) introduced the *containers* feature (also called *zones* - see PSARC 2002/174). A zone provides an environment for running applications that is isolated from all other zones.

As currently implemented, the hardware boots a single Solaris kernel that takes control of the system. This single instance of Solaris is referred to as the *global zone*. The system administrator (also referred to as the *global zone administrator*) has the sole responsibility for configuring, installing, booting, and maintaining any number of independent zones that are referred to as *non-global zones*.

All processes in all zones share the resources of the physical system under control of the single Solaris kernel booted. While all non-global zones are visible from within the global zone, each installed and booted non-global zone appears within itself to be a unique Solaris image completely isolated from all other zones.

Each non-global zone appears to be running in a separate system (physical hardware) with a unique set of configuration information and devices, such as network addresses, host names, users, storage, network connectivity, etc. Any other zone appears as if it is running in different hardware with no way to directly discover the existence or configuration of any other zone on the same system.

1.4 How do zones affect the Solaris *flash* feature?

The flash technology was *not* extended to support zones (see defect 6246943 - flash tools not zone aware create/deploy of flash archives MAY not work if non-global zones installed). While flash archives *might* successfully be created and deployed if non-global zones are installed, under certain circumstances the creation of a flash archive *may not* contain all of the installed non-global zone file systems, and the deployment of a flash archive *may not* result in the complete reconfiguration of all installed non-global zones that might be

necessary to allow the non-global zones to boot and run properly on the target system.

This is a separate issue that is ***not*** addressed by this project.

1.5 How do zones affect the Solaris *upgrade* feature?

1.5.1 What happened?

The Live Upgrade, package, and patch facilities were extended in Solaris 10 GA (3/05) to provide support for *installing* non-global zones, and for *maintaining* software within all installed non-global zones (see PSARC 2003/460).

Support for *upgrading* the installed software on any non-global zone from one release of Solaris to a later release is ***not*** provided with Solaris 10 GA (3/05).

1.5.2 What happens if a system with installed non-global zones is upgraded today?

In Solaris 10 GA (3/05), standard upgrade, Live Upgrade, and `pfinstall(1M)` do not know about non-global zones. The system would be partially upgraded if a Solaris image with installed non-global zones were allowed to be upgraded using the current upgrade mechanisms. The global zone would be properly upgraded, but the non-global zones would be *partially upgraded:*

- The analysis phase of the upgrade process does not take non-global zones into consideration.
- Solaris packages installed in the global zone and in the non-global zone are upgraded.
- Solaris packages installed in non-global zones by the non-global zone admin that are not also in the global zone will remain in their original (non-upgraded) state.
- Non-package operations performed by the upgrade feature are not performed on any non-global zones.

Booting a partially upgraded system results in unpredictable behavior on the part of all the installed non-global zones.

To prevent a Solaris image with installed non-global zones from being damaged by an upgrade attempt, code was included for both standard upgrade and Live Upgrade in Solaris 10 GA (3/05) to refuse to upgrade a system that has installed non-global zones.

As a result, once a system administrator configures and installs the first non-global zone, a Solaris 10 GA (3/05) system can never be upgraded to a later release of Solaris until all non-global zones are effectively destroyed.

The goal of this project is to allow Solaris instances with non-global zones to be upgraded.

1.5.3 Who is impacted by the lack of direct upgrade support?

The following customers are ***not*** impacted by the lack of direct upgrade support for installed non-global zones:

- Upgrading an existing Solaris 7, Solaris 8, or Solaris 9 system to any release of Solaris 10.
- Upgrading an existing installation of any release of Solaris 10 that does not have any zones installed.
- Initial installation of any release of Solaris 10.

The customers that ***are*** impacted by the lack of direct upgrade support for non-global zones are those customers that:

- Have any release of Solaris 10 installed, *and*
- Have installed one or more non-global zones, *and*
- Need the changes present in a later release of Solaris 10 (such as Solaris 10 update 1 (12/05)) than what is already installed on the system, *and*

- That cannot (or do not want to) directly install specific patches to fix the issues that would be resolved by upgrading to a later release of Solaris 10 than what is already installed on the system.

In addition, any customer that needs to upgrade from one release of Solaris to another (such as Solaris 10 to Solaris Nevada) that also has any installed non-global zones is impacted.

1.5.4 What options exist today to workaround this problem?

Once Solaris 10 update 1 (12/05) is available, if the upgrade problem described by this document is not resolved, the following options are available to the customer that has installed Solaris 10 GA (3/05) and has installed non-global zones:

- Install patches for defects resolved in Solaris 10 update 1 (12/05), and install appropriate new packages introduced in Solaris 10 update 1 (12/05) – this preserves the complete system configuration, including the configuration of the global zone and all non-global zones and is referred to as the *interim solution*.
- Delete all installed non-global zones, upgrade the global zone to Solaris 10 update 1 (12/05) (or later), reinstall all non-global zones – this preserves the configuration of the global zone, but requires a complete reinstallation and reconfiguration of all non-global zones.
- Reinstall the system with Solaris 10 update 1 (12/05) – this requires a complete reconfiguration of the entire system, including reinstalling and reconfiguring all non-global zones.

2.0 The solutions

2.1 Overview

A solution for upgrading a system from Solaris 10 GA to Solaris 10 update 1 (12/05) when non-global zones are installed must be available in Solaris 10 update 1 (12/05). The permanent solution described by this document cannot be delivered in time for Solaris 10 update 1 (12/05). An interim solution must be delivered in Solaris 10 update 1 (12/05).

2.2 What is an *upgrade*?

Within a minor release of Solaris (such as Solaris 10), any single update release of Solaris (such as Solaris 10 GA or Solaris 10 update 1 (12/05)) can be upgraded to any future update release of Solaris (such as Solaris 10 update 2 (3/06) (3/06)), or can be upgraded to any future minor release of Solaris (such as Solaris 11).

For example, Solaris 10 GA (3/05) can be upgraded to a later update release (such as Solaris 10 update 1 (12/05) or later) or to a later minor release (such as Solaris 11 or later). Similarly, Solaris 10 update 2 (3/06) can be upgraded to Solaris 10 Update 3 (or later) or to Solaris 11 (or later).

The end result of an *upgrade* of Solaris is the appropriate removal and updating of the packages and patches from the installed release of Solaris to a known set of packages and patches that can be quantified as the specific Solaris release that the system was upgraded to, such as Solaris 10 GA (3/05), Solaris 10 update 1 (12/05) (6/05), Solaris 11, etc.

In general, an *upgrade* of an installed instance of Solaris to a later release of Solaris allows for these types of updates:

- For packages present in the version of Solaris being upgraded that are also updated in the release of Solaris being upgraded to, install the new version of the package on top of the existing package (referred to as an “update in place”) so that existing editable or volatile files have their contents preserved.
- For packages present in the release of Solaris being upgraded to that are not present in the version of Solaris being upgraded, add those new packages that are required because of dependency issues.
- For packages that are present in the version of Solaris being upgraded that are not present in the release of Solaris being upgraded to, remove those packages.

- When going from one minor release of Solaris to a later minor release (such as Solaris 10 to Solaris 11), patches specific to the installed Solaris release may removed.
- When going from one update release of Solaris to a later update release (such as Solaris 10 Update 1 to Solaris 10 Update 2), any installed patches that are not part of the update release being upgraded to are reverted to the version distributed with the update release. For example:
    - Solaris 10 Update 1 is installed on the system.
    - Version 1 of patch 10000-1 for SUNWxyz is delivered into Solaris 10 Update 2.
    - Version 2 of patch 10000-1 for SUNWxyz is delivered into Solaris 10 Update 3.
    - Version 2 of patch 10000-1 is installed on the Solaris 10 Update 1 system.
    - The system is upgraded to Solaris 10 Update 2.
    - The version of patch 10000-1 for SUNWxyz that is installed is now Version 1 (the version distributed with Solaris 10 Update 2).

2.3 The interim solution

The interim solution consists of installing patches for existing packages and installing new packages only. The permanent solution consists of removing obsolete packages, installing new packages, and upgrading existing packages.

This interim solution only works when upgrading within the same release of Solaris (such as going from Solaris 10 GA to Solaris 10 update 1 (12/05)). The interim solution will not work when upgrading to a later release of Solaris (such as upgrading from Solaris 10 to Solaris 11).

This document addresses only the permanent solution. Refer to the design document for the interim solution for details about that project contained in http://installzone.sfbay/projects/zones.

2.4 The permanent solution

2.4.1 Overview

This project is the fourth “phase” of a the multi-phase Admin/Install zones support plan (see PSARC 2003/064). This project was approved for a Minor release only (in this case, Solaris 10 GA). This final phase of the project was not ready for Solaris 10 GA.

PSARC 2003/064 states “For the first release, upgrading a system that has non-global zones installed is only supported through the use of Live Upgrade.”

This is the major goal of this project. The global zone administrator uses Live Upgrade to upgrade Solaris on a system with installed non-global zones. The non-global zone administrator is not provided any mechanism for upgrading Solaris.

2.4.2 Performance goals

Using the existing upgrade framework and methodologies, the performance impact on upgrading a system with installed non-global zones is expected to be similar to the performance impact experienced with the first zone installation code that used the same methodology as standard Solaris install, which basically called pkgadd for each package needed to initially install a zone.

Based on that experience, upgrade performance with zones would be roughly equivalent to sequentially upgrading an equivalent number of similarly configured individual systems. For example, if a standard upgrade of a system with the entire Solaris 10 distribution installed takes 120 minutes to complete, then upgrading a system with zones using the same basic methodology would take approximately the same amount of time scaled up by the number of zones – with 10 zones 1200 minutes; with 50 zones 6000 minutes (4 days); with 110 zones 13200 minutes (over 9 days).

While extending the time to upgrade from 120 minutes for a single zone system to 20 hours for a system with

10 zones might be acceptable, several large customers have over 110 installed non-global zones. It would be unreasonable to expect such customers to wait over a week for an upgrade to complete.

This project has a goal of providing improved performance for the upgrade operation in general, and specifically when upgrading a system with installed non-global zones. For example, it should be possible to optimize the upgrade operation so that packages shared between the global zone and any non-global zones can be upgraded more quickly than the equivalent of upgrading the same packages on individual systems. It should also be possible to optimize the way in which the software contents database is created and updated across multiple zones. There are many other areas that can be optimized.

The minimum performance goal to upgrade a system with installed non-global zones is to be at least as good as the current performance when upgrading an equivalent number of similarly configured stand alone systems.

Put another way, the real time to upgrade a system with "n" installed non-global zones will not be longer than the real time it would take to upgrade "n" individual systems installed with the equivalent software stacks.

2.4.3 Enhance Live Upgrade

This project proposes to first enhance Live Upgrade to support upgrading Solaris instances that have installed non-global zones, and to defer enhancing standard upgrade:

- The effort to get standard upgrade to support installed zones is *more* complex than getting Live Upgrade to support zones.
    - Standard upgrade was never designed to handle the size and complexity that Solaris has evolved to over the past decade, let alone to handle the problems imposed by Solaris zones. The performance of the existing methodology would not scale well if zones were simply introduced into the existing mechanisms.
    - Portions of standard upgrade are used by Live Upgrade – any changes to standard upgrade must preserve Live Upgrade functionality. Making standard upgrade support zones requires more time for development and testing, as it must support use from both the mini-root and from Live Upgrade.
    - When upgrading using multiple CDs or DVDs, standard upgrade requires rebooting the system to continue the upgrade. If the current upgrade methodology were extended:
        - After the reboot following a successful upgrade from the first CD or DVD, the global zone administrator would continue to upgrade the global zone only using the remaining CDs or DVDs.
        - Once the global zone is fully upgraded, then each non-global zone administrator in turn would have to continue the upgrade the respective non-global zone. This would require coordination between all of the various zone administrators so that each zone would have access to the removable media device and that the CD or DVD media would be switched when necessary for each non-global zone.
- Standard install is commingled with standard upgrade. The changes to make standard upgrade support zones will cut across most areas of standard upgrade including those shared with standard install. Extra effort is required to make sure that standard install is not adversely affected by changes to standard upgrade to support non-global zones.
- Since Live Upgrade was introduced into Solaris 8, the long term strategy for upgrade in general is to over time migrate customers from standard upgrade to Live Upgrade and eventually EOL standard upgrade completely. This is not a new strategy introduced by this project, but it has been poorly communicated to the field and to our customers.
- Standard upgrade requires the system to upgrade be taken down for a full backup followed by the upgrade. In case of failure the only fallback is to restore the entire system from backup. The time to backup will be increased based on the storage requirements of all of the installed non-global zones. Since all non-global zones would be off-line during the entire backup and upgrade operation, in the case of server consolidation it would be as though many systems were concurrently off-line for an extended period of time. In the worst case scenario of the customer with over 100 installed non-

global zones, the time to fully backup the system and perform the upgrade using standard upgrade might approach two weeks of system down time.

- Live Upgrade is not currently zone aware.
    - Live Upgrade is missing support for non-global zones not only for upgrade but for creating, booting and administering alternative boot environments as well.
    - Live Upgrade must be enhanced to fully support non-global zones even if upgrade support were not an issue.

After weighing the advantages and disadvantages of the two possible strategies, the conclusion is less overall time will be spent, and long term strategy is better served, by first focusing on fully and properly implementing non-global zone support in Live Upgrade. This solution was approved by Solaris PAC (see PSARC 2003/460). This decision does not preclude making standard upgrade fully and properly support upgrading non-global zones in a follow-on project.

This project delivers several enhancements to Live Upgrade to support installed non-global zones:

- Creation of boot environments (`lucreate(1M)`) is extended by this project to recognize and warn when non-global zones are present in file systems shared between boot environments, and to provide a single "create and upgrade" operation that provides improved performance over the traditional "create" followed by "upgrade" operation.
- Activation and synchronization (`luactivate(1M)`) are extended by this project to properly synchronize all installed non-global zones present on a newly activated boot environment.
- Upgrading of boot environments (`luupgrade(1M)`) is enhanced by this project to properly upgrade a boot environment that has installed non-global zones by using the enhanced upgrade mechanism also delivered by this project.

Given that many customers supposedly wait for the first update before installing Solaris and do a fresh installation at that time, it make sense to explore what effort it would take to have Solaris 10 Update 1 (12/05) automatically reserve the storage space needed for later use by Live Upgrade.

2.4.4 Provide enhanced upgrade mechanism that is zone aware

The existing upgrade mechanism (`pfinstall(1M)`) was adequate for the "one Solaris image to upgrade" role when it was first introduced earlier in Solaris 2. Over time, `pfinstall` has become less adequate as the size and complexity of Solaris has increased. The introduction of Solaris zones requires major changes to support what is in effect a multiple operating system upgrade – a role `pfinstall` was never designed for. Rather than attempt to rework `pfinstall` to address all of these issues, this project implements a new upgrade mechanism that is designed from the ground up for performance and support of zones.

The package and patch tools suffer from existing not zone aware defects (see defects `6275530 and 6275531`). As part of any zone aware upgrade path, the package and patch tools must be enhanced to be zone aware so that any alternative root specified via `-R` is handled properly so that it is treated as a global zone and any installed non-global zones in that global zone are properly updated.

2.5 The future of standard upgrade

When this project is delivered into Solaris 10 Update 2 (3/06), the interim solution introduced into Solaris 10 Update 1 (12/05) will be removed, so that:

- Live Upgrade will successfully upgrade a Solaris image regardless of whether or not there are any installed non-global zones.
- Standard upgrade will successfully upgrade a Solaris image that does not have any installed non-global zones. This is the same behavior as in Solaris 10 GA (3/05) and Solaris 10 Update 1 (12/05).
- Standard upgrade will *not* allow an upgrade of a Solaris image that has any installed non-global zones. This is the same behavior as in Solaris 10 GA (3/05) but is different from Solaris 10 Update 1 (12/05) where the interim solution allowed upgrading of a Solaris image that has installed non-

global zones.

This decision does not preclude making standard upgrade fully and properly support upgrading non-global zones in a follow-on project. If it is decided that standard upgrade must support upgrading a Solaris image with installed non-global zones, the recommended course of action is to revise standard upgrade to use the new upgrade mechanism delivered by this project. Standard upgrade would essentially invoke the same permanent solution used by Live Upgrade that was delivered in Solaris 10 Update 2 (3/05).

At that point in time there would be only one upgrade mechanism used by standard upgrade, jumpstart, and Live Upgrade. This would address the "duplicate upgrade code support" issues that have been raised where standard upgrade and jumpstart would have different upgrade code bases than that used by Live Upgrade.

3.0 Technical details

3.1 Overview

This section provides background information to aid in understanding the feature enhancements proposed later.

3.2 Current upgrade methodology

As of Solaris 10 GA (3/05), upgrading Solaris requires an instance of Solaris to upgrade, and an install image containing the version of Solaris to upgrade *to*. `pfinstall` is the mechanism that performs this upgrade operation. When run, `pfinstall` analyzes and compares both of these images and produces a Bourne shell "script" that performs the upgrade through what is is essentially a set of `pkgrm(1M)` and `pkgadd(1M)` operations that. These features are delivered by the Install consolidation.

Two upgrade mechanisms are currently available: **standard upgrade** and **Live Upgrade**.

3.2.1 Standard upgrade

Standard upgrade is available by booting the Solaris install image containing the version of Solaris to upgrade *to*. For example, to upgrade a system to Solaris 10, the system is booted from a Solaris 10 installation image. In this case, the system is upgraded while the system is running Solaris 10.

An initial scan of the hardware is done looking for attached storage devices that contain installed instances of Solaris that can be upgraded. If upgradable instances of Solaris are found, the system administrator is allowed to choose a Solaris image to upgrade, or to perform an initial installation of Solaris.

If the system administrator chooses to upgrade an installed instance of Solaris, the `pfinstall` executable *contained in the Solaris install imaged that was booted* is invoked. It examines the Solaris image to upgrade and the booted Solaris install image to upgrade to. An upgrade script is generated and executed, and the selected installed Solaris image is upgraded to the version of Solaris to upgrade to.

The system is essentially "off-line" for the duration of the upgrade. This includes first taking the system down and doing a full backup so that in the event of a system or upgrade failure there is a way to recover the original (non-upgraded) system image.

The standard upgrade process includes a primitive "restart" mechanism that (in theory) allows the upgrade to be restarted at the last "remembered" location in the event the system is halted while the upgrade is in progress. This restart mechanism does not always reliably work. For example, if the system was halted during a `pkgadd` operation, the package may be partially installed, and may or may not successfully install when the system is restarted and the `pkgadd` operation is attempted again.

To make this restart mechanism viable, storage information that is cached in system memory must be flushed (written to) the storage devices after each operation is completed. In practice attached disks are synchronized (see `sync(2)`) after each package operation, resulting in a substantial performance impact waiting for all information to be committed to the disks.

3.2.2 Live Upgrade

The Live Upgrade feature (see `live_upgrade(5)`) has been delivered as part of Solaris since Solaris 8 Update 5 (8/01). Live Upgrade allows the system administrator to make a copy of the currently running system (referred to as an **A**lternative **B**oot **E**nvironment or ABE) and to then upgrade the ABE while the currently running system is still fully operational. The system can then be rebooted to the upgraded ABE after the upgrade operation is successful. The only required downtime is the time necessary to take down and reboot the system once the ABE is successfully upgraded. The ABE is upgraded while the system is running the version of Solaris being upgraded.

The system administrator chooses the storage devices to use for the ABE. Once the ABE is created, the system administrator directs the ABE to be upgraded by providing a path to a Solaris install image containing the version of Solaris to upgrade *to*. The ABE file systems are mounted, and then the `pfinstall` executable *contained in the specified Solaris install image to upgrade to* is invoked. An upgrade script is generated by examining both the ABE to be upgraded and the Solaris install image to upgrade to. The upgrade script is run, causing the ABE to be upgraded to the specified version of Solaris to upgrade to.

The system is operational for the duration of the ABE creation and upgrade. The only downtime is the time needed to reboot the system to the ABE following a successful upgrade. If the system is halted during the upgrade, or the upgrade fails, the last booted Solaris image remains untouched (not upgraded) and fully functional. If any problems are discovered once the upgraded ABE is booted, the original (non-upgraded) Solaris image is available for “fallback”, requiring only a system reboot to recover.

3.3 Accessing non-global zones during upgrade

3.3.1 A security issue

Standard upgrade and Live Upgrade both perform the upgrade operation on a non-running instance of Solaris – the *Solaris image to upgrade*. The `-R` option to `pkgadd` and `pkgrm` is used to cause the package operations to operate on the non-running instance of Solaris, thereby preserving the currently running Solaris image. Standard upgrade involves booting the system from a mini-root that is the version of Solaris to upgrade *to*. Live Upgrade involves running the version of Solaris to be upgraded *from*.

Issues in accessing non-global zone file systems:

- To maintain global zone integrity, non-global zone file systems cannot be directly accessed from the global zone.
- To access the file systems of a non-global zone that is not currently booted, the non-global zone must first be booted into a special *single user maintenance mode*.
- Because non-global zone configuration information is based on the global zone's configuration, the global zone must be booted first before any non-global zones can be booted.
- The Solaris image to be upgraded cannot be booted while it is being upgraded, and therefore no zones that are part of the Solaris image to be upgraded can currently be booted during the upgrade process.
- Since non-global zones that are part of the Solaris image to upgrade cannot currently be booted while the Solaris image to upgrade is being upgraded, the non-global zone file systems cannot currently be accessed during an upgrade operation.

Additional work is necessary to allow “booting” or providing a primitive “zone encapsulation” for non-global zones that are not part of the currently running Solaris image so that the non-global zone file systems can be accessed during upgrade operations. See the *scratch zone* enhancement described later in this document.

3.3.2 A matter of trust

The creation of a non-global zone involves processing known global zone data under control of the global zone at the direction of the global zone administrator. A non-global zone's file system space can be “trusted” from the time it is first created (from the global zone) until the non-global zone is first booted. There is no need to take

extra security precautions when accessing a non-global zone's file system space during the initial creation and installation of the non-global zone.

From the moment a non-global zone is first booted, a malicious non-global zone administrator could set up "traps" that are harmless when accessed from within the non-global zone, but that could compromise the security of the global zone if accessed from the global zone. For example, the non-global zone administrator could:

- Modify existing package "remove" scripts that start up a daemon – when pkgrm is used in the global zone using `-R` to refer to the non-global zone's root file system, the daemon is started with root access in the global zone.
- Create a "Trojan horse" directory entries that access the global zone's file system space when accessed directly from the global zone.

3.3.3 The issue

The integrity of the current running system (that might be the mini-root), the global zone and all non-global zones being upgraded is paramount. After the first transition of a non-global zone to a zone state higher than "installed", the non-global zone is forever "not trusted" by the global zone - the non-global zone's file system space may never be trusted outside of the non-global zone's sphere of influence.

For example, from the moment a non-global zone is first booted, in the global zone it is a security risk to:

- use the package and patch commands in the global zone (or from the mini-root) with the `-R` option to perform operations directly on a non-global zone's file system space.
- access any file or directory in any non-global zone's file system space directly from the global zone.

3.3.4 Maintaining security

Maintaining global zone (or mini-root) integrity increases the cost of implementing any operation that must access a non-global zone's file system space. Today to upgrade Solaris a script is created that contains a series of commands that performs the bulk of the *upgrade*. It might seem practical to directly analyze not only the global zone file system space but to also analyze all non-global zones file system space and to then add additional commands to the script that use the `-R` option to directly access and upgrade each non-global zone. This would open up the global zone to an attack by a malicious non-global zone administrator.

3.3.5 Insulating the global zone from non-global zone attacks

A mechanism is available to insulate the global zone from direct attacks and traps from the non-global zone. This allows access to any non-global zone in such a way that no other zone (including the global zone) can be compromised in any way. The mechanism provides a way for the global zone to run a command on the non-global zone and provide the command read-only access to resources that are located outside of the non-global zone.

For example, it is possible to "run the `pkgadd` command under the environment accessible to a particular non-global zone, making available read-only a file on the global zone containing the package stream to add to the non-global zone".

Since the non-global zone's file system space may not be accessed directly from the global zone, use of this mechanism increases the programming effort necessary to perform any operation on any non-global zone's file system space from the global zone.

3.3.6 The bottom line

All actions described in this document that involve updates to a non-trusted non-global zone are required to use this mechanism so that the global zone is never compromised. This is implied even if unspecified elsewhere in this document. The cost of using this mechanism must be taken into account when determining the magnitude of any effort described herein.

3.4 How are failures handled when non-global zones are installed?

3.4.1 Overview

As part of managing the more complex software environment resulting from the introduction of zones, it is necessary to improve the global zone administrator's ability to locate software installation problems. The class of problems this project initially intends to capture and record are those associated with package installation and removal operations, whether resulting from a standalone package operation, a patch operation, or a system installation/upgrade operation.

The high-level requirements are as follows:

- features that perform software maintenance operations must automatically and persistently log failure information so that the failure can be seen and dealt with immediately, or at some future date:
    - a failure to install or remove a package within an OS instance
    - a failure to install or remove a patch within an OS instance
    - a failure to upgrade an OS instance
- the global zone administrator can conveniently view logged failures for selected OS instances on a system: the global zone, non-global zones, diskless clients, or global/non-global zones in alternate boot environments.
- features that log failures must also have the ability to clear failures, and must clear logged failures whenever it is possible.

These requirements have some commonality with the class of problems that the Fault Management Architecture (FMA) was created to solve. At some point we may be able to leverage FMA in this effort, but at this time we do not propose to do so.

3.4.2 Software maintenance failure recording

A new software failure repository is implemented. Features such as the package and patch tools can record failure information in this repository using a standard interface that will be provided. The purpose of recording this information is to alert the global zone administrator that one or more software maintenance operations were not completely successful. It is up to the global zone administrator to determine the cause(s) and to implement the appropriate solution(s) and to then manually reset the condition once the global zone administrator is satisfied that the condition has been resolved.

Mandatory types of information that can be recorded:

- type of operation that failed – referred to as the `failureType` (e.g. package, patch, etc.)
- command that failed (e.g. pkgadd, patchrm, etc.)
- name of primary object the operation failed on (e.g. package name, patch number, etc.)
- date and time that the operation failed
- instance information on where the operation failed – referred to as the `instanceType` (zone name, boot environment name, etc.)

Optional information can be added to the record, such as:

- human readable error message
- command to execute when the failure is cleared

3.4.3 Software maintenance failure recovery

Two methods of clearing logged failures are provided:

- a software maintenance command is able to clear a failure logged by a previously run software

maintenance command:

- pkgadd is used to add a package – the package command fails leaving the package partially installed.
- pkgadd logs a failure to install the specified package.
- pkgrm is used to remove the partially installed package – and is successful.
- pkgrm removes the previously logged failure against the package.

- A special software maintenance command is used to manually clear the logged failure:
  - pkgadd is used to add a package – the package command fails reporting a postinstall script failure.
  - swadm (see section below) is used to report the status of package errors.
  - The global zone administrator manually corrects whatever problem was reported.
  - swadm is used to clear the error.

3.4.4 Failures of interest

The software maintenance and upgrade features can fail as follows:

- the package or patch operations can fail on the current running system, or on an alternative root such as a Live Upgrade alternative boot environment.
- the upgrade operation can fail on the global zone or on one or more non-global zones on an alternative boot environment.

This project is interested in the following software maintenance feature failures:

- a package or patch operations fails on the current running system.
- a package or patch operations fails on an alternative root such as a Live Upgrade alternative boot environment.
- a upgrade operation fails on the global zone on an alternative boot environment.
- a upgrade operation fails on one or more non-global zones on an alternative boot environment.

Other types of failures are possible and can be included in the future if needed.

3.4.5 How are failures logged?

The failures of interest are addressed as follows:

- package or patch tools fail to update the global zone on the current running system: a failure is recorded against the global zone
- package or patch tools fail to update one or more non-global zones on the current running system: a failure is recorded against the appropriate non-global zones
- package or patch tools fail to update the global zone on an alternative boot environment: a failure is recorded against the global zone on the ABE.
- package or patch tools fail to update one or more non-global zones on an alternative boot environment: a failure is recorded against the appropriate non-global zones on the ABE.
- upgrade operation fails to upgrade the global zone on the upgrade boot environment: a failure is recorded against the global zone on the ABE.
- upgrade operation upgrades the global zone successfully, but fails to upgrade one or more non-global zones on the upgrade boot environment: a failure is recorded against the appropriate non-global zones on the ABE.

4.0 Feature enhancements

4.1 Schedule implications

The solutions presented in this document describe all of the feature enhancements necessary to fully implement making several tools "non-global zone aware" with respect to supporting non-global zone upgrading:

- Certain enhancements are absolutely necessary to provide support for upgrading Solaris instances with installed non-global zones; without these enhancements upgrade support for non-global zones is not viable. Such enhancements are marked "***ZONE UPGRADE***".
- Other enhancements are required for completeness of Live Upgrade zone aware functionality; without these enhancements certain Live Upgrade features that should provide support for non-global zones will be lacking such support. Such enhancements are marked "***ZONE AWARE***".

Sections that are subdivided and contain both types of enhancements are marked "**MIXED**".

All of the feature enhancements of this project are ultimately required by the permanent solution. Several of the feature enhancements of this project are required by the interim solution. The design specifications for the interim solution will clarify which pieces of the permanent solution are needed.

4.2 Making Live Upgrade *zone aware [MIXED]*

4.2.1 Summary of work

Live Upgrade does not recognize (support) installed non-global zones. This is an outstanding issue (defect `6264796`) that affects Live Upgrade if any non-global zones are installed:

- software maintenance functions (package, patch, upgrade, flash, etc.) do not update any installed non-global zones.
- boot environment activation and synchronization functions do not synchronize any installed non-global zones when the newly created boot environment is activated and the system rebooted.
- boot environment creation has windows of opportunity that allow installed non-global zones in a newly created boot environment to share file systems with the same installed non-global zones in the currently running boot environment; any software updates performed on the newly created boot environment can cause the current installed non-global zones to be damaged.
- software upgrade functions do not upgrade any installed non-global zones.

Live Upgrade must be enhanced so that all features work properly when the global zone has installed non-global zones.

Summary of work::

- Enhance `lucreate(1M)` to recognize the existence of installed non-global zones, to provide the system administrator with the optional ability to alter the global and non-global zone file system configurations, and to not allow an invalid configuration to be created so that any installed non-global zones are properly replicated in the newly created boot environment.
- Enhance `lucreate` to implement "create new boot environment and upgrade" as a single atomic operation. By providing a new boot environment copy mechanism along with essentially merging "`lucreate`" with "`luupgrade -u`", the speed (real time required) to perform the "copy" and "upgrade" operations is drastically reduced. Among other things, the creation of the new boot environment can be minimized to not copy files and packages that would be deleted or updated during the upgrade process.
- Enhance and adapt the patch tool dependency checking mechanism to provide the difference information necessary to perform an upgrade with non-global zones in the "live upgrade" environment. This functionality will be part of Live Upgrade so that Live Upgrade no longer needs

to rely on `pfinstall`.

- Enhance `luupgrade(1M)` to recognize the existence of installed non-global zones in the boot environment to be upgraded, and to know how to upgrade a boot environment that has installed non-global zones. Use the enhanced and adapted patch tool dependency checking mechanism to perform the upgrade instead of calling `pfinstall`.
- Update `luactivate(1M)` and `lusync(1M)` to know how to activate and synchronize a boot environment that has non-global zones defined.

4.2.2 What about standard upgrade?

By implementing a new zone aware upgrade mechanism in Live Upgrade, there will be two possible upgrade mechanism implemented – the current non-zone aware standard upgrade mechanism implemented by `pfinstall(1M)`, and the new zone aware upgrade mechanism implemented by Live Upgrade.

This project does **not** indefinitely commit to supporting both of these upgrade mechanisms. It is long term strategy to move customers away from standard upgrade to Live Upgrade. By implementing the new upgrade mechanism in Live Upgrade, this is the start of the process of migrating customers away from standard upgrade. Any installation with installed non-global zones will be required to use Live Upgrade.

This decision does not preclude making standard upgrade fully and properly support upgrading non-global zones in a follow-on project. If it is decided that standard upgrade must support upgrading a Solaris image with installed non-global zones, the recommended course of action is to revise standard upgrade to use the new upgrade mechanism delivered by this project. Standard upgrade would essentially invoke the same permanent solution used by Live Upgrade that was delivered in Solaris 10 Update 2 (3/05). At that point in time there would be only one upgrade mechanism used by standard upgrade, jumpstart, and Live Upgrade. This would address the "duplicate upgrade code support" issues that have been raised where standard upgrade and jumpstart would have different upgrade code bases than that used by Live Upgrade.

4.2.3 Enhance `lucreate` *[MIXED]*

4.2.3.1 `lucreate` validate non-global zone file system configuration [ZONE UPGRADE]

Currently, the `lucreate` command allows any mounted file system to be shared between the current boot environment and the boot environment to create. The exceptions are "/", "/usr", "/var" and "/usr/openwin" that are not allowed to be shared. Any other mounted file system is shared by default unless the `lucreate` command is given a -m option to place the file system on a different storage device.

For example, suppose that the current running system has only the file systems "/" and "/export" mounted, and has installed non-global zones with root file systems on "/export/zones".

Given that configuration, this command would create the new boot environment "newbe", place the "/" file system on "c0t0d0s0", and automatically share "/export" between the current and the new boot environment:

```
lucreate -n newbe -m /:/dev/dsk/c0t0d0s0:ufs
```

This would result in both the current and the newly created boot environments sharing "/export/zones" and thereby sharing all of the file systems of the installed non-global zones including the portion that contains installed software.

To prevent this from going unnoticed, the `lucreate` command is enhanced to determine what mounted file systems can (and can not) be shared based on the presence of installed software or non-global zone file systems.

If any portion of an installed non-global zone's file system space that contains installed software would end up being "shared" between the current boot environment and the new boot environment to be created, an appropriate warning message is generated such as:

```
WARNING: zone <x> filesystem </> contains installed software
and is shared between boot environments on </export>
```

The boot environment is successfully created even if these warning messages are displayed. The global zone

administrator can choose to ignore these warnings. When a boot environment that has one or more non-global zone file systems shared is mounted via `lumount(1M)`, the shared file systems are mounted "read-only" within the context of the mounted boot environment. Any operation performed on such a boot environment (such as adding a package or upgrade) will fail with appropriate "read-only" errors and not corrupt the current running system.

The global zone administrator can alternatively delete the boot environment and rerun `lucreate` providing one or more -m options (as necessary) to alter the file system configuration so that no portion of an installed non-global zone's file system space containing installed software is shared.

The same behavior would occur if a non-global zone has other file systems specified with `zonecfg`(1M) "fs" resources. If one of these types of file systems cannot be mounted, the `lucreate` command fails with an appropriate error message.

In the previous example, to create a boot environment where the file systems reported would not be shared, the following command would work:

```
lucreate -n bename -m /:/dev/dsk/c1t0d0s0:ufs \
      -m /export:/dev/dsk/c2t0d0s0:ufs
```

now both "/" and "/export" are placed on separate file systems that are not shared between the current and the new boot environment.

The best practices document for Solaris zones must be updated to include recommendations on how to set up zone file systems so that Live Upgrade can be used without generating a shared file system warning message.

The most basic recommendations are:

- Place the zone root file systems on the same partition that contains the global zone root file system, or alternatively place the zone root file systems on one or more separate partitions that are not shared between the current and the new boot environment. The zone root file systems must contain all installed software, but must not contain any user specific data.
- Place all other zone file systems on the same file system that contains the global zone "/export" file system, or alternatively place all other zone file systems on one or more separate partitions that are shared between the current and the new boot environment.

4.2.3.2 Enhance `lucreate`: "copy and upgrade" atomic operation [ZONE AWARE]

The `lucreate` command is enhanced to implement an operating system upgrade operation as part of the initial boot environment creation process. This will provide significantly improved performance over the separate `lucreate` / `luupgrade` operations when installed non-global zones are present. This type of optimization is compatible with packages that have procedure scripts.

A similar optimization was done in phase one of this project for initial zone creation. The package database is processed and all non editable and volatile files are copied as a single operation. Then the packages are added with pkgadd to deliver the editable and volatile files, run install scripts, and update the package database.

Among other things, this optimization allows:

- For packages that will be removed during the upgrade:
    - Do not copy non-editable and non-volatile files belonging to packages that will be deleted.
    - Clean up the package database in a single operation. Using individual pkgrm commands cause the package database to be read, entries for the package removed, and the package database written back out.
- For packages to be updated during the upgrade:
    - Optimize package database updating so it is done once as opposed to per package.
    - Update all non-editable and non-volatile files in a single operation as opposed to per package.

- For packages to be added during the upgrade:
  - Optimize package database updating so it is done once as opposed to per package.
  - Install all non-editable and non-volatile files in a single operation as opposed to per package.

The following options are added to `lucreate(1M)`:

`-u` – Optional: if specified, request upgrade of the Solaris OS during ABE copy operation

`-s os_image_path` – Required when `-u` specified: path name of a directory containing a Solaris OS image. This can be a directory on an installation medium (such as a CD, DVD, etc.) or can be an path to a directory on a file system (such as NFS, UFS, etc).

`-j profile_path` – Optional when `-u` specified: path to a JumpStart profile that is used to modify the behavior of the upgrade (see `luupgrade(1M)` for details).

If the `-u` option is not provided, `lucreate` behaves as it does currently, creating the new boot environment without any upgrade.

If the `-u` option is specified, `lucreate` causes the new boot environment to be upgraded during the copy operation to the version of the Solaris OS specified by the `-s` option.

For example, the following two commands create the new boot environment "beName" and after the new boot environment is created cause the boot environment to be updated to Solaris 10:

```
lucreate -n beName -m /:/dev/dsk/c?t?d?s?:ufs
luupgrade -n beName -u \
      -s /net/installServerPath/solarisdvd.s10s_dvd/latest
```

would be replaced with this one command that performs both the boot environment creation and the Solaris upgrade:

```
lucreate -n beName -m /:/dev/dsk/c?t?d?s?:ufs -u \
      -s /net/installServerPath/solarisdvd.s10s_dvd/latest
```

4.2.4 Enhance `luupgrade` *[ZONE UPGRADE]*

4.2.5 Upgrade enhanced to work with installed non-global zones

Once the current running Solaris image is copied to a newly created ABE, the `luupgrade` command must recognize the existence of installed non-global zones in the ABE to upgrade, and to properly upgrade an inactive boot environment that has installed non-global zones. The dependency on `pfinstall` is removed and a new upgrade tool is used to perform the Solaris upgrade. Both of these are discussed in detail below.

There is no change to the `luupgrade` user interface to support upgrading an inactive boot environment that has installed non-global zones.

4.2.5.1 `luupgrade` validate boot environment zone file system config [ZONE UPGRADE]

When `luupgrade` is used to upgrade an inactive boot environment, the configuration of any installed non-global zone is checked to determine if any of the non-global zone file systems containing installed software are shared with the currently running system.

If any portion of an installed non-global zone's file system space that contains installed software is "shared" between the current boot environment and the inactive boot environment to be upgraded, an appropriate warning message is generated:

```
WARNING: zone <x> filesystem </> contains installed software
and is shared between boot environments on </export>
```

The global zone administrator has options similar to those for `lucreate` detailed above.

The global zone administrator can choose to ignore these warnings and proceed to use the boot environment being mindful of the fact that the upgrade has failed. When a boot environment that has one or more non-global zone file systems shared is mounted via `lumount(1M)`, the shared file systems are mounted "read-only" within the context of the mounted boot environment. Any operation performed on such a boot environment (such as adding a package or upgrade) will fail with appropriate "read-only" errors and not corrupt the current running system.

The global zone administrator can alternatively delete the boot environment and rerun `lucreate` providing one or more -m options (as necessary) to alter the file system configuration so that no portion of an installed non-global zone's file system space containing installed software is shared. Then reissue the same `luupgrade` command to upgrade the inactive boot environment.

4.2.6 Enhance `luactivate` and `lusync` *[ZONE UPGRADE]*

4.2.6.1 Boot environment activation

Currently when `luactivate` is run, a warning is displayed if any packages remain to be installed as the result of a prior upgrade operation. This is extended to all non-global zones: a warning is displayed if any non-global zone has packages that remain to be installed as the result of a prior upgrade operation.

4.2.6.2 Boot environment synchronization

When an ABE is activated that has installed non-global zones, both `luactivate` and `lusync` must recognize the existence of installed non-global zones in the new boot environment, and must synchronize the new boot environment so that the global zone and all non-global zones are synchronized.

Currently the synchronization is implemented as:

- When a boot environment is created via `lucreate(1M)`, an "initial" synchronization snapshot is made of the current running system that at that instant is the initial contents of the boot environment just created.
- When a boot environment is activated via `luactivate(1M)`:
  - a "source" synchronization snapshot is made of the current running system.
  - A "target" synchronization snapshot is made of the boot environment being activated.
  - The "source" and "target" snapshots are compared and any differences are reported.
- When the system is taken down:
  - a "source" synchronization snapshot is made of the current running system.
  - A "target" synchronization snapshot is made of the boot environment being activated.
- When a newly activated boot environment is first booted:
  - a "source" synchronization snapshot is made of the previous boot environment.
  - A "target" synchronization snapshot is made of the newly activated boot environment.
  - The "source" and "target" snapshots are compared and any differences are reported.
  - The newly activated boot environment is synchronized with the previous boot environment according to the rules described in `synclist`(4).

A "snapshot" consists of a list of all files described by the `synclist`(4) file along with information on the contents and directory information for each file sufficient to allow two snapshots to be compared to determine if a file has been changed in any way. The snapshot is stored in a project private file in the global zone (/etc/lu/ludb.local.xml).

An entry for the "initial" snapshot of "/var/adm/messages APPEND" in /etc/lu/synclist might be:

```
type="initial" action="APPEND" \
```

```
item="/var/adm/messages;root:root:1:100644:REGFIL:1300:819629271:"
```

With the addition of installed non-global zones, the synchronization process must now be extended to all installed non-global zones:

- The same processes that are currently run in the global zone for the various phases of boot environment synchronization are extended by this project to run in each installed non-global zone.
- When a boot environment is activated via `luactivate(1M)`, or when a newly activated boot environment is first booted, any differences for any installed non-global zone are reported to the global zone administrator who has the responsibility to contact the appropriate non-global zone administrators to allow them to correct any issues as necessary.
- Each non-global zone includes:
  - A `synclist`(4) file that allows the non-global zone administrator to control how the zone is synchronized when the system is upgraded using Live Upgrade.
  - A /etc/lu/sync.log file that contains the results of any synchronization that was done on the non-global zone.
  - A /etc/lu/ludb.local.xml file that contains the various snapshots used to perform the synchronization for the non-global zone.

There is no change to the `luactivate` or `lusync` user interfaces. The time required to create the synchronization snapshots will increase depending on the number of installed non-global zones present. The scratch zone feature will be used to implement appropriate portions of the boot environment synchronization.

During normal system operation, non-global zones do not have access to other zones or to other boot environments. When a system is up and running a non-global zone is isolated from all other zones by design. The scratch zone allows mounting any resources that are necessary. The synchronization mechanism is done automatically when the new boot environment is booted. The previous boot environment is made available via the scratch zone mechanism so that the synchronization can be done.

4.2.7 Enhance `lucompare` *[ZONE AWARE]*

The `lucompare(1M)` command, and the generation of the compare database, is enhanced to take the existence of installed non-global zones into account. The `lucompare` command can be used to:

- Compare the current boot environment with any other inactivate boot environment.
- Generate a snapshot of a boot environment.
- Compare the current boot environment with a previously created snapshot.

During normal system operation, non-global zones do not have access to other zones or to other boot environments that are part of the global zone. It is not possible to allow non-global zone administrators to compare their portion of the current boot environment with their portion of inactive boot environments since the non-global zone does not have access to the alternative boot environments that are part of the global zone. In addition, the non-global zones will not even be aware of any alternative boot environments that may be part of the global zone.

Because of this, the `lucompare` command is extended for the global zone administrator only:

- The compare databases are only stored in the global zone's file system space (in /etc/lu/compare).
- When generating a snapshot of a boot environment, the snapshot includes the contents of all non-global zones.
- When comparing the current boot environment with a previously created snapshot, the comparison includes comparing the contents of any installed non-global zones.
- When comparing the current boot environment with an inactive boot environment, the comparison includes comparing the contents of any installed non-global zones.

The generation of the snapshots and the comparisons are done in a zone access secure manner.

4.2.8 Remove dependency on `pfinstall` *[ZONE UPGRADE]*

4.2.9 Overview

Currently `luupgrade` uses `pfinstall` to create an upgrade script. Live Upgrade supports installation on, and upgrading from, up to 3 previous Solaris releases. For Solaris 10, this means Live Upgrade can be installed on Solaris 7, Solaris 8, Solaris 9 or Solaris 10 systems, and be used to upgrade to any release of Solaris 10.

There are existing issues with running a `pfinstall` that is compiled with Solaris 10 on Solaris 7, Solaris 8 or Solaris 9 systems. The Solaris ABI does not support this kind of "forward" compatibility, and it is only by diligent testing that we verify there are no runtime issues. It has long been planned to remove this dependency from Live Upgrade and this project provides the vehicle for doing this.

4.2.10 How pfinstall works today

The `pfinstall` utility can be used to do initial installations, flash installations, and upgrades using a profile which specifies everything `pfinstall` needs to know to carry out its task. The profile contains information such as which root device to install/upgrade, location of the image to install or upgrade to, disk space information, and a slew of other details for `pfinstall` to consume. The use of `pfinstall` by Live Upgrade is bound to a particular upgrade path and many of the operations `pfinstall` executes are time consuming and do need to be done for the purposes of Live Upgrade.

The following are the major operations `pfinstall` currently does for upgrade:

1. Detect hardware capabilities - not done for Live Upgrade: not needed.
2. Detect software capabilities - The software capabilities check was introduced for Live Upgrade because the version of `pfinstall` that `luupgrade` runs will be different than the version of Solaris installed on the system. A user typically upgrades to a version of Solaris that is later than the version where the Live Upgrade packages were installed from, so `luupgrade` has to check what software capabilities are available with the `pfinstall` included in the Solaris version to upgrade to and adjust accordingly.
3. Disk operations:
    a) Root device checking – not done for Live Upgrade: `lucreate` creates and sets up file systems during the creation of boot environments.
    b) Meta device (SVM) initialization – not done for Live Upgrade: the running system already has storage devices initialized and running.
    c) Space calculations – not done for Live Upgrade: requirements are addressed during `lucreate.`
    d) Make system bootable – not done for Live Upgrade: `luupgrade` makes a boot environment bootable.
4. Diskless client functionality - not done for Live Upgrade.
5. Parse jumpstart profile.
6. Analyze software to be upgraded
    a) Load upgrade media.
    b) Load installed media.
    c) Parse pkghistory.
7. Build and run the upgrade script.

When run from Live upgrade, the operations executed by `pfinstall` are a subset of the possible operations that are specific to the upgrade process only.

Analyzing the software to be upgraded is the only major operation that needs to be carried over from `pfinstall` to the new upgrade tool. The new upgrade tool will adhere to the same policies `pfinstall` follows to figure out what needs to be upgraded, however the mechanism by which it does the upgrade will be drastically different as it will make use of new algorithms to optimize performance and support zones.

`pfinstall` configures the software to be upgraded by using three main sources of data:

1. Packaging information from the upgrade media.
2. Packaging information from the installed system.
3. .pkghistory file (and .clusterhistory file if one exists) to determine any special actions needed for certain packages and/or files.

4.2.10.1 Concurrent operation locking

The package and patch zone support introduced a simple concurrency operation lock mechanism. This mechanism prevents concurrent package or patch operations from occurring that could potentially cause data corruption. For example, when a package is to be added to the global zone, all zones are first locked so that no package operations can be performed. If a package operation is currently in operation on any non-global zone, then the global zone waits until all of those operations conclude before proceeding. Once all zones are locked, the global zone package operation proceeds.

Appropriate operations (such as boot environment creation) are extended by this project to use the same locking mechanism. For example, in the case of boot environment creation, a lock is acquired so that no package or patch operation, or any zone creation or destruction operation, can occur while a boot environment is being created.

4.3 New upgrade tool *[MIXED]*

4.3.1 Overview

The new upgrade tool will be used by Live Upgrade in place of `pfinstall` and will be delivered via a Live Upgrade package. The upgrade tool will not have dependencies on any of the libraries that `pfinstall` currently use (e.g. /usr/snadm/libspmi*), and hence Live Upgrade will no longer have this dependency on binaries built on the current release.

The new upgrade tool will be able to analyze and upgrade a non-global zone which may or may not have shared file systems. The changes in the packaging commands will allow pkgadd/pkgrm to be run a mounted non-global zone from an alternate boot environment.

The upgrade tool will be modeled after `pfinstall(1M)`, however it will not do any of the extraneous work that `pfinstall` does that are not necessary for a Live Upgrade. For example the upgrade tool will not do disk operations such as mounting devices, verifying root devices, etc. The upgrade tool relies on Live Upgrade to setup file systems for the alternate root/zone to be upgraded. The `luupgrade` command will mount the alternate root environment/zone on to a local mount point before calling the upgrade tool to do the upgrade.

The upgrade process call the new upgrade tool to analyze and upgrade each zone in the ABE independently starting with the global zone.

4.3.2 Support for 'lucreate -u' *[ZONE AWARE]*

The new upgrade tool will support the new feature in '`lucreate -u`' by having 'copy and upgrade' functionality. This feature will significantly save overall upgrade time because it eliminates the initial copy of the entire boot environment onto the new environment. A comparison is made between the software on the current boot environment against the upgrade media, and the equivalent of copying and upgrading the boot environment is achieved by selectively copying files/packages into the new boot environment (a set that is much smaller than the entire boot environment), and then pkgadd'ing the necessary packages from the upgrade media to complete the upgrade of the new boot environment.

This same basic scheme is already used for zone installation. The file copying can be optimized to a large

degree because the bulk of Solaris is contained in type 'f' (non-volatile and non-editable) files that can be copied directly without worrying about package install scripts. Special handling is used for type 'e' (editable) and type 'v' (volatile) files and for packages with install scripts so that operations are done in the correct order. If a package has a checkinstall script then certain optimizations may not be performed.

A general overview of how this will be achieved:

1. Analyze packages on the current boot environment (the global and all non-global zones) and on the upgrade media to determine affected packages – packages that will be updated via this upgrade process. Use .pkghistory to determine special actions needed on files/packages and deem them affected as well.
2. For packages that are not affected by this upgrade (this includes unbundled packages), add them to the 'copylist'.
3. For packages that are affected by this upgrade:
   a) Add all 'e' and 'v' type files to the 'copylist'
   b) Add all files that need to be removed (this is ascertained from the .pkghistory file), and that have a remove class action script, to the 'copylist'.
   c) Copy modified binaries (type 'f' files that have been modified) to the new boot environment and rename them with a ~$VER extension.
4. Add all files that do not belong to any package to the 'copylist'.
5. Do a mass copy of files in 'copylist' to the new boot environment and generate the contents file for the new boot environment based on what's been copied.
6. pkgrm all necessary packages from the new boot environment.
7. Remove patch information of all patches that have been fresh-bitted from the patch database.
8. pkgadd all necessary packages from upgrade media to the new boot environment.
9. Create special files in the new boot environment

4.3.2.1 Support for 'lucreate -u' – Sparse root non-global zone [ZONE AWARE]

The above algorithm applies to the global zone and to whole root non-global zones. Sparse root non-global zones however, have shared file systems and the above algorithm must be tweaked. In other words, when a sparse root non-global zone is mounted, it will not be empty; it contains a fully populated /usr file system that's already been upgraded, yet the source non-global zone's package database does not reflect this.

The following steps are changed:

Step 3c – Modified binaries that reside on shared file systems will not be copied and renamed on the new boot environment.

Step 5 – We will filter the 'copylist' for any files that reside in a shared file system.

Step 7 – Do not run pkgrm for packages whose contents reside in shared file systems. The software package database is updated, but there is no attempt to remove files that are in read-only areas shared from the global zone.

4.3.3 Support for 'luupgrade -u' *[ZONE UPGRADE]*

The new upgrade tool will support the '`luupgrade -u`' which upgrades a boot environment that has already been copied.

1. Analyze packages on the copied boot environment and upgrade media to determine affected packages – packages that will be updated via this upgrade process. Use .pkghistory to determine special actions needed on files/packages and deem them affected as well.
2. For packages that are affected by this upgrade:

a) Copy modified binaries to the new boot environment and rename them with a ~$VER extension.

3. pkgrm necessary packages from the new boot environment.
4. Remove patch information of all patches that have been fresh-bitted from the patch database.
5. Run necessary pkgadd commands from the upgrade media to the new boot environment.
6. Create special files in the new boot environment.

4.3.3.1 Support for 'luupgrade -u' - Sparse root non-global zone [ZONE UPGRADE]

As in the algorithm for '`lucreate -u`', the above algorithm applies to the global zone and to whole root non-global zones. Sparse root non-global zones have shared file systems and the above algorithm must be tweaked.

The following steps are changed:

Step 2a – Modified binaries that reside on shared file systems will not be renamed.

Step 3 – `Do not` run pkgrm for packages whose contents reside in shared file systems.

4.4 Package & patch tools alternative root zone aware *[ZONE UPGRADE]*

The package and patch tools were enhanced in Solaris 10 GA (3/05) to properly maintain any installed non-global zones. These tools were not enhanced to maintain installed non-global zones in any alternative root configurations (that is, when the `-R` option is used to specify a path to a global zone that is not currently booted). These tools need to be enhanced to properly maintain installed non-global zones for both stand alone use, as well as when used by Live Upgrade to maintain alternative boot environments.

These tools will make use of the scratch zone technology implemented by this project (see below) to operate on installed non-global zones contained in alternative root environments.

4.4.1 How are zones supported in S10 GA?

In order to prevent security issues as described in the previous section “Accessing non-global zones during upgrade”, the package and patch tools cannot directly access non-global zone file system space to perform operations.

The package and patch tools support installed non-global zones in the current running system by running the package or patch operation in a manner similar to running the operation under control of “zlogin”:

- if the zone is booted it it roughly equivalent to running the command as “zlogin -z zone command”.
- if the zone is not booted it is roughly equivalent to:
  - booting the zone into single user mode (`zoneadm -z zone boot -s`).
  - wait for milestone to be online: `svc:/milestone/single-user:default`
  - wait for milestone to be online: `svc:/system/filesystem/local:default`
  - run the package or patch command as “`zlogin zone -s command`”
  - halt the zone (`zoneadm -z zonename halt`).

4.4.2 What is the problem?

When the package and patch commands are run with the -R option, there is no mechanism available today that will allow non-global zones that are not part of the current running system to be booted. Some new mechanism needs to be implemented that allows non-global zones that are part of a non-booted instance of Solaris to be booted and processes run within the context of the appropriate non-global zone.

4.4.3 The scratch zone

As previously described in “accessing non-global zones during upgrade”, to prevent compromising the security

of other zones, any operation performed on a non-global zone must be done within the context of the non-global zone when it is booted. Once the non-global zone is booted, any operation is performed within that non-global zone, thus preserving the integrity of all other zones.

When performing operations on the current running system, the installed non-global zones can be booted using existing mechanisms (e.g. zoneadm). For example, when adding a package in the global zone, once the global zone has been updated, each non-global zone must be updated. If the non-global zone to be updated is not currently booted, it is booted into a special “single user administrative mode”. Then the package operation is run within the context of the booted non-global zone. This preserves the integrity of all other zones.

Any installed non-global zones that are part of an inactive boot environment cannot be directly booted using existing mechanisms. There is no way to run an operation within the context of the non-global zones that are part of an inactive boot environment. A new mechanism is needed that allows a non-global zone that is part of an inactive boot environment to be accessed while maintaining security for all other zones.

The concept of a “scratch zone” is implemented by this project. A scratch zone is a transient non-global zone that can easily be quickly created and destroyed, and that appears (inside the scratch zone) like what a regular non-global zone would look like if installed at the moment the scratch zone is instantiated.

A scratch zone is used as a surrogate for a non-global zone that cannot be booted. To access a non-global zone that cannot be booted, a scratch zone is instantiated. A scratch zone mirrors (to a large extent) what would be seen inside of a "normal" mini-root environment:

- A scratch zone is mounted on a temporary mount point within the global zone. There is no built in limit to the number of scratch zones that can be concurrently instantiated.
- The local tools within the scratch zone's “root” are read-only loopback mounted from the current running system. Since the root file system is read-only, the global zone is protected from any malevolent actions that might come about from accessing the file system of a non-global zone.
- The resources that belong to the non-global zone to be operated on are mounted as an alternative root within the scratch zone (e.g. “/a”). If any resources are currently mounted, then they will be read-only loopback mounted from the current running non-global zone.
- Processes can then be run within the scratch zone that can perform operations on the non-global zone without fear of compromising anything outside of the scratch zone's sphere of influence (e.g. all zones in the current running system remain out of reach of the scratch zone).

The package and patch commands are modified by this project so that when “-R” is used, a scratch zone is used to access any installed non-global zones in the alternative root specified. For each non-global zone to be operated on, a scratch zone is instantiated that has the resources for the non-global zone mounted. Then the operation is performed on the non-global zone within the newly instantiated scratch zone. This isolates the operation to the scratch zone and the resources of the non-global zone mounted, preserving the integrity of all other zones. When all operations are completed, the scratch zone is halted and destroyed.

A detailed design document for the scratch zone will be delivered as a separate document for this project.

4.5 Software failure reporting *[MIXED]*

A new "`swadm`" command is introduced:

```
swadm [ -X ] [ -d ] [ -v ] [ -f ] command
```

This command is used to interrogate and clear failures reported in the software failure repository implemented by this project.

The following `command`s are implemented:.

- `Status [ <instanceSpec>] [ <failureSpec> ]`

  This command is used to display reported failures for the specified instances.

- `Clear [ <instanceSpec> ] [ <failureSpec> ]`

  This command is used to clear failures from the specified instances.

The command syntax is:

```
"swadm" [ "-X" ] [ "-d" ] [ "-v" ] [ "-f" ] <command>

<command> :: = <statusCommand> | <clearCommand>

<statusCommand> ::= "status" <instanceSpec> [ <failureSpec> ]
<clearCommand> ::= "clear" <instanceSpec> [ <failureSpec> ]

<instanceSpec> ::= ( "current" | <instanceType> )
<instanceType> ::=
      [ <bootenvInstance> ] <zoneInstance> [ <bootenvInstance> ]
<zoneInstance> ::= "-z" <zoneName> [ "," <zoneInstance> ]
<bootenvInstance> ::= "-n" <beName> [ "," <bootenvInstance> ]
<zoneName> ::= text representing a zone name
<beName> ::= text representing a boot environment name

<failureSpec> ::= <failureType> [ " " <failureSpec> ]
<failureType> ::= <packageFailures> | <patchFailures>
<packageFailures> ::= <packageFailure> [ " " <packageFailure> ]
<packageFailure> ::= "package" <packageName> [ "," <packageName> ... ]
<patchFailures> ::= <patchFailure> [ " " <patchFailures> ]
<patchFailure> ::= "patch" <patchName> [ "," <patchName> ... ]
<packageName> ::= text representing a package name
<patchName> ::= text representing a patch name
```

The following options are provided:

- -X (optional): causes output from swadm to be in simplified **X**ML format (similar to Live Upgrade's `-X` option as described in PSARC 2001/551 – Live Upgrade XML programmatic interface). This allows the results from the swadm command to be programatically parsed
- -d (optional): causes swadm to run in "no execution" (**d**ry run) mode: swadm outputs what it would do without actually performing any operations. When used with the clear command, the global zone administrator can see what would be cleared by a certain command without actually performing the operation.
- -v (optional): causes swadm to run in "**v**erbose" mode: by default commands output summary information. When the -v option is used, commands output more detailed information.
- -f (optional): causes swadm to not ask for confirmation when performing any operation that could change the logged failures (such as clear).

The following `instanceType`s are implemented:

- `-n <beName>` - refer to the specified **b**oot environment within the current running system. This is consistent with the current live upgrade commands which use -n to specify the boot environment name to use.
- `-z <zoneName>` - refer to specified non-global **z**one within the current running system. This is consistent with the current zone administrative commands which use -z to specify the zone to use.

Notes:

- A boot environment (or zone name) is compared with existing boot environments (or zone names) using file name pattern matching (see `fnmatch`(5)) so that multiple boot environments (or zones) can be specified.
- When matching zone names the global zone is not included; that is, "`-z *`" refers to all installed non-global zones excluding the global zone. The global zone may be referred to only by specifying the name "`global`" directly. To refer to the global zone and all installed non-global zones, specify "`-z global,*`".

The order in which the instanceTypes are specified on the command line is important. The order of

instanceTypes controls the path to the instance to operate on. For example:

- "`-n be1 -z z1`" means "zone z1 on boot environment be1".
- "`-z z1 -n be1`" means "boot environment be1 on zone z1".

- "`current`" – refers to the current running system and is equivalent to "`-z global,*`"

The following `commands` are implemented:.

- `status <instanceSpec> [ "/" <failureSpec> ]`

  This command is used to display reported failures for the specified instances within the current running system (or the boot environment specified by -n or the non-global zone specified by -z).

- `clear <instanceSpec> [ "/" <failureSpec> ]`

  This command is used to clear failures from the specified instances within the current running system (or the boot environment specified by -n or the non-global zone specified by -z).

  The clear command will prompt the user for confirmation unless the -f option is specified.

Examples:

- `swadm status`

  Report all failures for all instances – the global zone, all non-global zones, all boot environments, and all zones within all boot environments.

- `swadm status current`
  - Report all failures for the current running system only – the global zone and all non-global zones – this is equivalent to "swadm status `-z global,*`".
- `swadm status -z global`
  - Report all failures within the global zone.
- `swadm status -z z1`
  - Report all failures within non-global zone z1.
- `swadm status -z *`

  Report all failures for all zones.

- `swadm status -n be1`
  - Report all failures within boot environment be1.
- `swadm status -n *`

  Report all failures for all boot environments.

- `swadm status -n beName -z zone1`
  - Report all failures for zone `zone1` within the specified boot environment.
- `swadm status package pkga`
  - Report failures for package `pkga` for all instances.
- `swadm status current package pkga`
  - Report failures for package `pkga` for the current running system.
- `swadm status -z zone1 package *`

  Report all package failures for zone `zone1 on all instances.`

- `swadm status -z zone1,zone2 package *`

- Report all package failures for zone `zone1` and zone `zone2` on all instances.

Below is a sample of how a failure is generated, listed and cleared:

```
# swadm status
No failures reported

# pkgadd -d /net/path SUNWxxx
[ ... ]
ERROR: unable to boot non-global zone <zonex> into administrative mode
[ ... ]

# swadm status
1 package failure in 1 zone

# swadm -v status
1 package failure in 1 zone
failure <pkgadd> zone <zonex> package <SUNWxxx>: unable to boot non-
global zone into administrative mode

# pkgrm SUNWxxx
[...]

# swadm status
No failures reported

# pkgadd -d /net/path SUNWyyy
[ ... ]
ERROR: postinstall script failed
[ ... ]

# swadm status
1 package failure

# swadm -v status
1 package failure
failure <pkgadd> package <SUNWyyy>: postinstall script failed

# swadm clear
1 package failure
Clear failures? (y/n) y
```

5.0 Feature exceptions

5.1 Diskless client upgrade support

Diskless client support was officially EOLed when Adminsuite support was EOLed in Solaris 2.7. As part of that process, the diskless client upgrade code was disabled in Solaris 2.7. The diskless client support was brought back for Solaris 2.8 1/01 and Solaris 9; however, the group that did this work was not part of the Solaris Install group and did not have the resources to make upgrade work.

Their decision was to not enable the upgrade code and to instead state that the only way to upgrade a system with diskless client support was to uninstall all diskless clients, perform the OS upgrade, and reinstall all diskless clients. Currently if an upgrade is done of a Solaris 9 FCS system with diskless clients configured to Solaris 10 GA, none of the diskless clients or services are upgraded.

It would require a major effort to get the diskless client upgrade code working again with Solaris 10 even without this project. Diskless client upgrade support is beyond the scope of this project. There will be no attempt made by this project to take diskless client into account.

5.2 Solaris product registry support

The first three phases of PSARC 2003/460 (admin/install zones support) delivered into S10 GA (3/05) do not provide support for the Solaris product registry. Several defects are outstanding, including:

```
6295318 - prodreg dumps core on non-global zone when running Studio 9/10 installers
6220284 - zone is missing /var/sadm/install/productregistry and prodreg
      will core dump
```

This project cannot provide non-global zone support for the Solaris product registry until the zone installation, package and patch tools are enhanced to provide such support. Because of that, this project does not provide any additional support for the Solaris product registry. A separate follow-on project needs to be funded that provides full support for the Solaris product registry with installed non-global zones.

The Purple Haze installer project will obsolete the current product registry. The idea is to move the very limited amount of useful information in the current product registry into the package database. This means that the current (and future) code to maintain the package database across all non-global zones will automatically take care of the "registry" data as well.

5.3 The `lu` interface (`/usr/sbin/lu`)

The `lu` feature is a Command User Interface – a text based interface that is implemented using `curses(1)` and `fmli(1)`. The `lu` feature cannot be internationalized due to limitations in `fmli(1)` that prevent 8-bit characters from being used. There is no future plan to internationalize the `fmli` interface.

The plan of record is the `lu` feature will be replaced in the future with a more up to date Graphical User Interface. A G11N waiver is in effect for `lu` until such time as a replacement internationalizable GUI solution can be delivered.

Since the year 2000, only one new feature in Live Upgrade have been made available in the `lu` feature (ability to install flash archive in alternative boot environment). All other new Live Upgrade features are implemented in the Live Upgrade command-line utilities only. Because of this, the current `lu` feature must not be depended on for critical functionality. This fact is documented in the various Solaris installation guides and the `lu(1M)` manual page.

This project provides for no additional work to be done to specifically support the `lu` feature. Any updates to any Live Upgrade command line utilities proposed by this project will **not** be reflected by any changes to the `lu` feature.

6.0 User experience

6.1 Current install

As of Solaris 10 GA (3/05), upgrading Solaris requires an instance of Solaris to upgrade, and an install image containing the version of Solaris to upgrade *to*. `pfinstall` is the mechanism that performs this upgrade operation. When run, `pfinstall` analyzes and compares both of these images and produces a Bourne shell "script" that performs the upgrade.

Two upgrade mechanisms are currently available: **standard upgrade** and **Live Upgrade**. The results of an upgrade are identical regardless of which mechanism is used because `pfinstall` is called in both cases to perform the actual upgrade work.

When an upgrade is completed, summary results are displayed for the system administrator running the upgrade tool that indicate overall success or failure of the upgrade operation. Detailed results of the upgrade operation are stored in several files in the upgraded Solaris image:

- `/var/sadm/system/logs/upgrade_log` – the overall results of the upgrade operation are stored in this file. See Appendix A for the typical contents stored in this file.
- `/var/sadm/system/data/upgrade_cleanup` – if present contains a list of files on the

upgraded system that may need to be manually modified after the upgrade. Typically the files in this list are files that were modified since their original installation. See Appendix B for the typical contents stored in this file.

6.2 Live Upgrade

This project enhances Live Upgrade to support upgrading a boot environment that has non-global zones installed. If the system administrator uses Live Upgrade, non-global zones can be present on the Solaris image to upgrade. The user experience is fundamentally the same whether non-global zones are present or not.

The Live Upgrade feature enables you to maintain multiple operating system images on a single system. An image (called a boot environment) represents a set of operating system and application software packages. The boot environments might contain different operating system and/or application versions.

Live Upgrade is extensively documented:

- There are section 1M manual pages for each Live Upgrade command line interface utility, including detailed information on how to upgrade a system using Live Upgrade.
- There are section 4 manual pages on Live Upgrade file formats.
- There is a section 5 manual page on the Live Upgrade standard and environment.
- The Solaris installation guide has detailed information on how to use Live Upgrade.

On a system with the Solaris Live Upgrade software, your currently booted OS environment is referred to as your active, or current boot environment. You have one active, or current boot environment; all others are inactive. You can perform any number of modifications to inactive boot environments on the same system, then boot from one of those boot environments. If there is a failure or some undesired behavior in the newly booted boot environment, Live Upgrade software makes it easy for you to fall back to the previously running boot environment.

Live Upgrade software includes a full suite of commands which implement all of the Live Upgrade features and functions, including:

- The lucreate command offers a set of command line options that enable you to create a new (or inactive) boot environment based on the current boot environment. The system administrator uses lucreate to create a new boot environment that is a copy of the current running system. See Appendix C for an example of how to use `lucreate` to create a new boot environment that is a copy of the currently running system.
- The luupgrade command enables you to upgrade an operating system image on an inactive boot environment. The source for the image can be any valid Solaris installation medium, including a Solaris Flash archive. The system administrator uses the luupgrade command to upgrade an inactive boot environment to a later release of Solaris. See Appendix D for an example of how to use `luupgrade` to upgrade an existing inactive boot environment (created previously by lucreate) to Solaris 10.
- The luactivate command causes a specified inactive boot environment to be the next boot environment that the system boots. Activating an inactive boot environment is done by making the boot environment's root partition bootable so that the next time the system is booted, the that boot environment is the one the system boots from.

6.3 How does this project affect the user experience?

6.3.1 Command line interfaces

The command line interface for `lucreate` and `luupgrade` are not changed. The same commands and options that the system administrator would have used in previous releases of Solaris are used in the global zone to create a new boot environment and then upgrade that boot environment to Solaris 10 Update 2 (3/06). The use of `lucreate` and `luupgrade` does not need to be changed if installed non-global zones are present in the

global zone.

6.3.2 Log files

Currently several log files are generated which contain the bulk of the upgrade "results". Besides looking at the output displayed by the upgrade mechanism (summary success or failure), the system administrator is accustomed to looking at the contents of the upgrade log files to determine what kind of problems were encountered, and what kinds of cleanup are necessary following an upgrade:

- `/var/sadm/system/logs/upgrade_log` - the global zone continues to have the upgrade log for the global zone placed in this file. The contents of this file are identical to that described in Appendix A.
- `/var/sadm/system/data/upgrade_cleanup` – the global zone continues to have the upgrade cleanup log for the global zone placed in this file. The contents of this file are identical to that described in Appendix B.

Because the non-global zones do not have access to the newly created boot environment, to facilitate access to the various log and upgrade files generated, those files are placed on both the current boot environment and the new boot environment. This way, the non-global zone administrators will have access to the log and upgrade files without having access to the inactive boot environment.

The following new log files are introduced by this project:

- `/var/sadm/system/logs/upgrade_log_zone_<zonename>` - each non-global zone will have an upgrade log specific to the non-global zone placed in this file. The contents of this file are identical to that described in Appendix A.
- `/var/sadm/system/logs/upgrade_log_zone_<zonename>` - the global zone will have have an upgrade log specific to the indicated non-global zone placed in this file. There will be one file created for each installed non-global zone. The contents of this file are identical to the file by the same name stored in the indicated non-global zone.
- `/var/sadm/system/logs/upgrade_log_summary` - the global zone will have a new upgrade summary log file placed in this file. The global zone administrator can examine this one file to determine if any failures or warnings were generated during the upgrade. This file and the contents are new and are described in Appendix E.
- `/var/sadm/system/logs/upgrade_cleanup_zone_<zonename>` - each non-global zone will have an upgrade cleanup log specific to the non-global zone placed in this file. The contents of this file are identical to that described in Appendix B.
- `/var/sadm/system/logs/upgrade_cleanup_zone_<zonename>` - the global zone will have have an upgrade cleanup log specific to the indicated non-global zone placed in this file. There will be one file created for each installed non-global zone. The contents of this file are identical to the file by the same name stored in the indicated non-global zone.
- `/var/sadm/system/logs/upgrade_cleanup_summary` - the global zone will have a new upgrade summary log file placed in this file. The global zone administrator can examine this one file to determine if any failures or warnings were generated during the upgrade. The contents of this file are an amalgam of the contents of the individual upgrade cleanup log files for all zones.

6.3.3 What are the flow of events?

Before the introduction of Solaris zones, the system administrator would coordinate the upgrade activities with the users of the system. If Live Upgrade were used, the users would be notified when the new boot environment would be created, upgraded, and activated. If standard upgrade were used, the users would be notified when the system would be down for the backup, upgrade and reboot operations. The system administrator would take care of upgrading the system and any cleanup or other issues that need to be addressed before making the system generally available again.

With the advent of zones, the global zone administrator takes on an expanded role of the same basic actions that

system administrators have taken in the past. The global zone administrator continues to communicate with the users of the global zone, but must now also communicate and coordinate the upgrade operation with the various non-global zone administrators that would be affected by any upgrade operation. The global zone administrator still handles synchronization and upgrade issues for the global zone, but must now also be concerned about those same issues for the installed non-global zones. The global zone administrator must notify the non-global zone administrators when the upgrade is going to occur, and involve them as appropriate in the process.

The non-global zone administrators perform the system administration role for their respective zones. They must notify their user base as necessary about when the system will be down and why, and handle the synchronization and upgrade issues that might be relevant to their individual non-global zones even though they do not perform the upgrade operation itself. This requires coordination with the global zone administrator as the upgrade process proceeds.

A general flow of events might be:

- The global zone admin decides the system is going to be upgraded.
- The global zone admin notifies all non-global zone admins that the system is going to be upgraded
    - The global zone admin discusses the reasons for the upgrade, and any other issues that might need to be addressed, with the non-global zone admins.
    - The global zone admin requests that all non-global zone admins become familiar with the boot environment synchronization process, and make sure that the synchronization control file in their respective zones is updated as necessary.
    - All parties agree on the time frame for performing the various upgrade tasks.
- All zone admins notify their user base of the upgrade plans and the resulting outages that will occur.
- The global zone admin uses `lucreate` to create a new boot environment which is a copy of the currently running system.
- The global zone admin uses `luupgrade` to upgrade the newly created inactive boot environment to a later release of Solaris.
- The global zone admin notifies all non-global zone admins that the upgrade is completed.
    - The global zone admin examines the upgrade and cleanup log files and resolves any issues relating only to the global zone. The global zone admin must work with the non-global zone admins as necessary to resolve issues that involve any of the non-global zones.
    - If there are serious issues that affect the upgrade in general, the global zone admin communicates this with the non-global zone admins. If necessary, the global zone admin can choose to abandon the current upgrade, and attempt the upgrade again in the future after the issues have been addressed.
    - The global zone admin requests that the non-global zone admins review their respective upgrade and cleanup log files. The global zone admin works with each non-global zone admin as appropriate to resolve any issues.
- When all of the zone administrators believe that the upgrade is successful, they agree on the time to transition the system to the upgraded boot environment.
- The global zone admin uses `luactivate` to activate the upgraded boot environment, and then causes the system to be rebooted.
- Once the system is rebooted, all of the zone administrators verify that the upgraded system has no issues and makes their respective zones generally available.
- If there are any issues discovered with the upgraded system that cannot be resolved, the zone administrators can decide whether or not the previous (non-upgraded) boot environment should be reactivated and the system rebooted.

7.0 References

7.1 Arc Cases

- LSARC 2005/070 – Unified installer (project Purple Haze)
- PSARC 2003/460 – Admin/Install Zones Support
- PSARC 2002/174 – Virtualization and Namespace Isolation
- PSARC 2002/064 – Live Upgrade Boot Environment Enhancements
- PSARC 2001/551 – Live Upgrade XML Programmatic Interface
- PSARC 2001/084 – Live Upgrade FCS
- PSARC 1997/318 – Solaris/NCR Online Upgrade
- PSARC 1993/553 – Extended System Configuration

7.2 Defects

7.2.1 Defects associated with this project

These defects are directly associated with this project:

```
6275530 - package commands do not process configured zones when -R is given path to global zone
6275531 - patch commands do not process configured zones when -R is given path to global zone
6264796 - live upgrade does not process configured zones when creating and processing BEs
6246943 - flash tools not zone aware create/deploy of flash archives MAY not work if non-global
```

7.2.2 Defects related to this project

These defects are related to this project and should be considered when implementing new code:

```
4909825 - prevent downgrade of unbundled packages that are also bundled
4840294 - Lockhart 2.0 packages get downgraded when upgrading OS to Solaris 10.
unknown – Upgrade needs to handle when a file switches types between "e" and "v"
```

7.3 RFEs

- 4976161 – PSARC 2003/460 Admin/Install Zones Support (Phase IV: luupgrade zone aware)

7.4 URLs

- ➔ `http://zones.eng` - zones project main page
- ➔ `http://installzone.sfbay/projects/zones` - Solaris install consolidation zones project page
- ➔ `http://installzone.sfbay/purple` - Unified software install platform
- ➔ http://docs.sun.com/app/docs/coll/1236.1 - Solaris 10 Release and Installation Collection
  - ➢ http://docs.sun.com/app/docs/coll/1236.1?q=live+upgrade - Live Upgrade documentation

7.5 Document revision history

Revision	Date	Author	Notes
1.0	6/02/2005	Gary Gere	Initial draft
1.1	6/03/2005	Gary Gere	Revised lucreate to include notion of “create and upgrade” transaction
1.2	6/14/2005	Gary Gere	Updated damaged state and lucreate sections
1.3	6/20/2005	Dave Miner	Updated damaged state section
2.0	6/29/2005	Gary Gere	Major revisions
2.1	7/01/2005	Gary Gere	Revised section 3.5 (failure handling)
2.2	7/05/2005	Gary Gere	Revised section 4.5 (software failure reporting)
2.3	7/08/2005	Gary Gere	Added details to lustatus command (section 4.2.4)
2.4	7/11/2005	Gary Gere	Removed schedule information, revised lucreate -m command (section 4.2.1.2), added information on scratch zone (section 4.4)
2.5	7/13/2005	Gary Gere	Changes based on discussions with Dave Miner and James Carlson.
2.6	7/14/2005	Gary Gere	Included pfinstall/lucreate/luupgrade methods and algorithms to support lucreate -u and luupgrade -u when non-global zones are installed.
3.0	7/15/2005	Gary Gere	Design document finalized - ready for review.
3.1	7/22/2005	Gary Gere	Incorporated comments from David Powell
3.2	7/25/2005	Gary Gere	Incorporated comments from Mary Ding
3.3	8/04/2005	Gary Gere	Incorporated comments from David Powell, Dan Price, Rich McAllister and David Comay
Friday, August 5, 2005 01:45 pm			Date and time this document was prepared

Appendix A - /var/sadm/system/logs/upgrade_log

The results of the upgrade operation performed by the existing tools in Solaris 10 GA (3/05) are stored in this file. The contents are typically:

```
Starting upgrade:
Removing obsolete packages and saving modified files

Removing package SUNWcsu:

Removal of <SUNWcsu> was successful.
Removing package SUNWkvm:

[...]

Removal of <SUNWsogm> was successful.
Removing patch 113887-27 from the system.
Removing patch 113886-27 from the system.

Installing new packages

Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Doing pkgadd of SUNWocfd to /.
603 blocks

Installation of <SUNWocfd> was successful.
Doing pkgadd of SUNWcsu to /.
24838 blocks

[...]

Installation of <SUNWsra> was successful.
Doing pkgadd of SUNWsrh to /.
282 blocks

Installation of <SUNWsrh> was successful.

The messages printed to the screen by this upgrade have been saved to:

      /a/var/sadm/system/logs/upgrade_log

After this system is rebooted, the upgrade log can be found in the file:

      /var/sadm/system/logs/upgrade_log

Please examine the file:

      /a/var/sadm/system/data/upgrade_cleanup

It contains a list of actions that may need to be performed to complete
the upgrade.  After this system is rebooted, this file can be found at:

      /var/sadm/system/data/upgrade_cleanup

After performing cleanup actions, you must reboot the system.
```

Appendix B - /var/sadm/system/data/upgrade_cleanup

When an upgrade is performed by the existing tools in Solaris 10 GA (3/05), this file contains a list of files on the upgraded system that may need to be manually modified after the upgrade. Typically the files in this list are files that were modified since their original installation.

Entries can be one of:

```
<file1>: existing file renamed to <file2>
<file1>: existing file preserved, the new version was installed as <file2>
<file>: had been deleted and has now been restored
<file>: file type was changed from <type1> to <type2>
<file>: target of symbolic link was changed from <target1> to <target2>
<file1>: target of hard link was changed from <file2>
```

Typical contents:

```
/var/apache/tomcat: had been deleted and has now been restored.
/etc/gconf/2/path: existing file renamed to /etc/gconf/2/path~11
/etc/snmp/conf/snmpd.conf: existing file renamed to /etc/snmp/conf/snmpd.conf~11
/a/etc/.login: existing file preserved, the new version was installed \
      as /a/etc/.login.new
/a/etc/pam.conf default entries updated, please examine/update \
      customized entries
/a/etc/pam.conf updating pam_unix with default PAM entries please \
      examine/update any new entries
EXISTING_FILE_SAVED_TO_OLD: /a/etc/usb/config_map.conf.old /
a/etc/usb/config_map.conf.old.13145032
'/net' entry in /a/etc/auto_master map was not updated to include \
      '-nobrowse' option.

Sendmail has been upgraded to version 8.13.4 .
After you reboot, you may want to run
/usr/sbin/check-hostname
and
/usr/sbin/check-permissions ALL
These two shell-scripts will check for common
misconfigurations and recommend corrective
action, or report if things are OK.
```

Appendix C – Using lucreate to create a new boot environment

Below is an example of the simplest use of `lucreate` to create a new boot environment (named "newBe") that is a copy of the currently running boot environment (named "currentBe") placing the "/" filesystem on "/dev/dsk/c1d0s4":

```
# lucreate -n newBe -m /:c1d0s4:ufs -m /usr:c1d0s5:ufs
Discovering physical storage devices
Discovering logical storage devices
Cross referencing storage devices with boot environment configurations
Determining types of file systems supported
Validating file system requests
The device name <c1d0s4> expands to device path </dev/dsk/c1d0s4>
Preparing logical storage devices
Preparing physical storage devices
Configuring physical storage devices
Configuring logical storage devices
Analyzing system configuration.
Comparing source boot environment <currentBe> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Searching /dev for possible boot environment filesystem devices
Updating system configuration files.
The device </dev/dsk/c1d0s4> is not a root device for any boot environment.
Creating configuration for boot environment <newBe>.
Creating boot environment <newBe>.
Creating file systems on boot environment <newBe>.
Creating <ufs> file system for </> on </dev/dsk/c1d0s4>.
Mounting file systems for boot environment <newBe>.
Calculating required sizes of file systems for boot environment <newBe>.
Populating file systems on boot environment <newBe>.
Checking selection integrity.
Integrity check OK.
Populating contents of mount point </>.
Copying.
Creating shared file system mount points.
Making boot environment <newBe> bootable.
Population of boot environment <newBe> successful.
Creation of boot environment <newBe> successful.
```

Appendix D – Using luupgrade to upgrade an existing inactive boot environment

Below is an example of using `luupgrade` to upgrade the boot environment created in Appendix C to Solaris 10:

```
# luupgrade -n newBe -u -s /
net/installsvr/export/solarisdvd.10x_dvd/latest
Validating the contents of the media
</net/installsvr/export/solarisdvd.10x_dvd/latest>.
The media is a standard Solaris media.
The media contains an operating system upgrade image.
The media contains <Solaris> version <10>.
Constructing upgrade profile to use.
Locating the operating system upgrade program.
Checking for existence of previously scheduled Live Upgrade requests.
Creating upgrade profile for BE <newBe>.
Determining packages to install or upgrade for BE <newBe>.
Performing the operating system upgrade of the BE <newBe>.
CAUTION: Interrupting this process may leave the boot environment
unstable
or unbootable.
Upgrading Solaris: 100% completed
Installation of the packages from this the media is complete.
Adding operating system patches to the BE <newBe>.
The operating system patch installation is complete.
ABE boot partition backing deleted.
INFORMATION: The file </var/sadm/system/logs/upgrade_log> on boot
environment <newBe> contains a log of the upgrade operation.
INFORMATION: The file </var/sadm/system/data/upgrade_cleanup> on boot
environment <newBe> contains a log of cleanup operations required.
INFORMATION: Review the files listed above. Remember that all of the
files are located on boot environment <newBe>. Before you activate boot
environment <newBe>, determine if any additional system maintenance is
required or if additional media of the software distribution must be
installed.
The Solaris upgrade of the boot environment <newBe> is complete.
```

Appendix E - /var/sadm/system/logs/upgrade_log_summary

This file is new with this project. A summary of the upgrade is stored in this file. The results of a successful upgrade are typically:

```
Upgrade of boot environment <newBe> on <Thu Aug  4 13:01:58 PDT 2005>
Upgraded from media </net/installsvr/export/solarisdvd.10x_dvd/latest>
The media is a standard Solaris media.
The media contains an operating system upgrade image.
The media contains <Solaris> version <10>.
There are <24> installed non-global zones configured.
No failures were reported.
No warnings were reported.
The file </var/sadm/system/logs/upgrade_log> in the global zone on boot
environment <newBe> contains a log of the upgrade operation.
The file </var/sadm/system/data/upgrade_cleanup> in the global zone on boot
environment <newBe> contains a log of cleanup operations required.
Review the files listed above.
Before you activate boot environment <newBe>, determine if any additional system
maintenance is required or if additional media of the software distribution must
be installed.
The Solaris upgrade of the boot environment <newBe> is complete.
```

The results of an upgrade with problems are typically:

```
Upgrade of boot environment <newBe> on <Thu Aug  4 13:01:58 PDT 2005>
Upgraded from media </net/installsvr/export/solarisdvd.10x_dvd/latest>
The media is a standard Solaris media.
The media contains an operating system upgrade image.
The media contains <Solaris> version <10>.
There are <24> installed non-global zones configured.
There are <3> failures reported against <6> zones
There are <2> warnings reported against <13> zones
The file </var/sadm/system/logs/upgrade_log> in the global zone on boot
environment <newBe> contains a log of the upgrade operation.
The file </var/sadm/system/data/upgrade_cleanup> in the global zone on boot
environment <newBe> contains a log of cleanup operations required.
Review the files listed above.
Before you activate boot environment <newBe>, determine if any additional system
maintenance is required or if additional media of the software distribution must
be installed.
Failure <1>: package <SUNWxyz> failed to install when added to zones <z1, z2, z5>
Failure <2>: package <SUNWabc> failed to install when added to zones <z2, z3, z5>
Failure <3>: package <SUNWfgh> failed to uninstall when removed from zone <z6>
Warning <1>: package <SUNWuio> generated warnings when added to zones <z3, z6>
Warning <2>: package <SUNWwer> generated warnings when added to zones <z1, z5>
The file </var/sadm/system/logs/upgrade_log_zone_z1> in the global zone on boot
environment <newBe> contains a log of failures and warnings for zone <z1>
The file </var/sadm/system/logs/upgrade_log_zone_z2> in the global zone on boot
environment <newBe> contains a log of failures and warnings for zone <z2>
The file </var/sadm/system/logs/upgrade_log_zone_z3> in the global zone on boot
environment <newBe> contains a log of failures and warnings for zone <z3>
The file </var/sadm/system/logs/upgrade_log_zone_z5> in the global zone on boot
environment <newBe> contains a log of failures and warnings for zone <z5>
The file </var/sadm/system/logs/upgrade_log_zone_z6> in the global zone on boot
environment <newBe> contains a log of failures and warnings for zone <z6>
The Solaris upgrade of the boot environment <newBe> is complete.
```