



Part I: Minimizing Domains for Sun Fire™ V1280, 6800, 12K, and 15K Systems

By Nicholas O'Donnell, Enterprise Server Products

Alex Noordergraaf, Enterprise Server Products

Sun BluePrints™ OnLine—August 2003



<http://www.sun.com/blueprints>

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 817-3340-10
Revision 1.0, 8/29/03
Edition: August 2003

Copyright 2003 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, California 95045 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Sun BluePrints, Sun Fire, JumpStart, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the US and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights—Commercial use. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the Far and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, Californie 95045 Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à <http://www.sun.com/patents> et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats-Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Sun BluePrints, Sun Fire, JumpStart, Java, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

LA DOCUMENTATION EST FOURNIE "EN L'ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Please
Recycle



Adobe PostScript

Part I: Minimizing Domains for Sun Fire™ V1280, 6800, 12K, and 15K Systems, Overview

This article is the first part of a two-part series that provides information and recommendations for minimizing domains for Sun Fire™ V1280, 6800, 12K, and 15K systems. This part provides background information, describes the concept of qualifying a minimized Solaris™ configuration for an application, covers how to automate installations using JumpStart™ technology, and details a recommended methodology for minimizing a system. Part II describes the package configurations needed for various applications, describes the profiles produced for performing JumpStart installations of domains, and provides a case study as an example of applying minimization methodology to an application.

This document contains the following topics:

- “Enhancing Security” on page 2
- “Background” on page 4
- “Qualifying a Solaris Configuration” on page 7
- “Automating Domain Installations” on page 13
- “Using Scripts to Qualify a Solaris Configuration” on page 16
- “Minimization Methodology” on page 21
- “About the Authors” on page 38
- “Related Resources” on page 39
- “Ordering Sun Documents” on page 41
- “Accessing Sun Documentation Online” on page 41

Enhancing Security

Securing computer systems against unauthorized access is one of the most pressing issues facing today's data center administrators. Recent studies suggest that the number of unauthorized access continues to rise, as do the monetary losses associated with these security breaches.

One way to reduce system vulnerabilities is to minimize the amount of software on a server. Fewer software components on a server means fewer security holes to detect and close. A vast number of system intrusions are accomplished through exploiting security holes in unnecessary operating system (OS) components. Minimizing the quantity of OS and other software components installed on a server can greatly improve overall system security by reducing the potential vulnerabilities.

This technique is referred to as minimization and is based on the premise that only software components for which there is a specific requirement or business need should be installed on a system.

Recommendations and Methodologies for Minimization

Recommendations and methodologies for minimizing systems are in a variety of forums, including Sun BluePrint OnLine articles, most notably the article titled "Minimizing the Solaris Operating Environment for Security: Updated for Solaris 9 Operating Environment." That article focuses on how to minimize the Solaris Operating Environment (OE) for a single-purpose, appliance-like application; a web server. Many functions and tools are not installed because they are not required for an appliance-like installation on which:

- Minimal debugging takes place
- Additional software would never be installed without re-installing the entire system
- The entire system is treated like a single field-replacable unit (FRU)

In this two-part article series for minimizing domains, we extend the recommendations made in the “Minimizing the Solaris Operating Environment for Security” article to address some of the fundamental challenges with minimized systems, including the following:

- Difficulty in adding software packages to a system after it is deployed and in production.

Adding additional software or hardware, whether OE or third-party, is difficult for many reasons. The primary difficulty is determining which Solaris OE packages (such as packages containing libraries, drivers, or software) might be required to make additional software function properly. After identifying required additional components, other challenges are typically encountered. This article addresses those challenges by proposing mechanisms to determine package dependencies as well as provide a baseline configuration that includes commonly used system components.

- Systems that cannot be treated as single FRUs require diagnostic tools for use by the customer as well as Sun’s support organization to diagnose and root cause problems with the system. This article covers these issues by proposing system configurations that include software packages and commands typically used by Sun’s support organization either directly or indirectly to address system problems.
- Package dependencies for advanced features in the Sun Fire V1280, 6800, 12K, and 15K systems. To date, these have not been identified in Sun product documentation. This article provides a configuration where the package dependencies of these advanced features are identified, documented, and tested.
- Required packages for Sun remote monitoring solutions.
- Required packages for popular third party applications such as Oracle and Sun ONE Application Server products.
- Validation that the defined packages incorporate all required components for proper function. This article addresses this issue by recommending rigorous quality assurance mechanisms and techniques to validate that the software load, as defined, is as correct and complete as possible.

As with all well-documented configurations, the implementation of these recommendations should be as automated as possible. We address automation in this article by defining JumpStart profiles that incorporate the recommendations and address the fundamental challenges listed previously. These profiles are available electronically and can serve as a starting point for developing profiles for your systems.

Background

This section describes the Sun Fire systems covered by this article, defines the extent of minimization on these systems, and describes the system configurations used for testing.

This section contains the following topics:

- “Assumptions and Limitations” on page 4
- “Hardware Overview” on page 5
- “Domains and System Controllers” on page 5
- “Hardware Differences” on page 5
- “Domain Differences” on page 6
- “Domain Installations” on page 6
- “Hardware Configurations” on page 7

Assumptions and Limitations

Sun Fire V1280, 6800, 12K, and 15K domains are Solaris systems that can be accessed either through a network or from a built-in system controller (SC). The methodology described in this article addresses new automated OE domain installations using JumpStart technology. Minimization of already deployed or installed domains is neither recommended nor supported.

This article is focused on developing minimized systems for deploying customer applications. The general methodology can be applied to determine the packages required for components in the Solaris Operating Environment (Solaris OE).

One of the most important details to gain from this article is the technique used to define a minimized system for a user application. This technique allows customers to apply a similar strategy to define a minimized system for their own applications. The example applications detailed in this article serve as a guide to this technique.

Hardware Overview

This article focuses on minimizing Sun Fire V1280, 6800, 12K, and 15K domains only. It does not cover minimizing SCs. The following subsections provide brief descriptions of the important differences of this hardware.

Additional information on these products is available in their product documentation manuals at <http://docs.sun.com>. Additionally, security best practice recommendations are available in the following Sun BluePrint OnLine articles available at <http://sun.com/security/blueprints>:

- “Securing Sun Fire 12K and 15K System Controllers”
- “Securing Sun Fire 12K and 15K Domains”
- “Securing the Sun Fire Midframe System Controller”

Domains and System Controllers

A domain is a virtual, installable Solaris system within a Sun Fire 6800, 12K, and 15K chassis. More than one domain might be defined for a single chassis and each of the domains are independent and isolated from one another.

Note – The Sun Fire V1280 does not support domains. It has a single Solaris system that is described in “Domain Differences” on page 6 of this article.

The SC is a separate system built into the chassis that defines, controls, and monitors domains contained within the chassis. A domain is defined through the system controller by assigning appropriate hardware components to it.

Hardware Differences

The Sun Fire hardware is divided into two groups defined by differences in the SC.

- The Sun Fire 12K and 15K systems use two independent UltraSPARC® II systems built into the chassis to provide failover capability. These both run the Solaris OE and have custom-written software to remain synchronized.
- The V1280 and 6800 chassis SCs are not Solaris based. They run the VxWorks operating system (OS) on the SC; the OS has the software for configuring domains implemented in firmware. Due to the limited nature of these SCs, it is often a requirement to have an additional Solaris based machine to act as a monitoring system for the domains.

Note – The Sun Fire V1280 and Netra 1280 server are functionally equivalent. Their main physical difference is that the Netra 1280 server is powered by DC power supplies and is certified to be NEBS-compliant security hardware. This article refers specifically to the Sun Fire V1280, but the software minimization aspects are equivalent on both.

Domain Differences

Once configured, the individual domain differences are minor. The following are the possible configurations:

- The Sun Fire 15K can be configured into a maximum of 18 domains.
- The Sun Fire 12K can accommodate a maximum of 9 domains.
- The Sun Fire 6800 can be configured as a maximum of 4 domains in the 6800, and 2 domains in 3800 and 48x0.
- The Sun Fire V1280 server consists of a single, default Solaris system that can be configured as 4, 8, or 12 UltraSPARC III CPUs. Although the Sun Fire V1280 does not support domains, the Solaris instance is viewed as a single domain instance for the purposes of this article.

Note – The Sun Fire V1280 server has dedicated serial and Ethernet ports for remote monitoring and administration capabilities by the Lights Out Management (LOM) module, independent of the Solaris OE, running on the SC.

Domain Installations

Automating a domain installation requires a Solaris system to be configured to act as a JumpStart server. The Sun Fire 12K and 15K SC can be used for this function. However, configuring a JumpStart server is beyond the scope of this article.

Because the Sun Fire V1280 and 6800 SCs are not Solaris based, they require an external Solaris based system to act as the JumpStart server. For more information on JumpStart technology and how it can be used to configure servers, refer to the *Advanced Installation* guide available at <http://docs.sun.com> or the Sun BluePrints book titled *JumpStart™ Technology—Effective Use in the Solaris™ Operating Environment*.

Hardware Configurations

In this article, the configurations used for testing are a Sun Fire 15K, a Sun Fire 6800 (midframe), and a Sun Fire V1280. Testing makes use only of two domains on these systems. The other domains are separate and independent.

The Sun Fire 15K is running Solaris 9 (12/02) OE and SMS 1.3 on the SC. The Sun Fire 15K has a Capacity On Demand 2.0 software CPU board available with a pre-installed right-to-use (RTU) license. The CPU board is configured into the test domain when needed.

The Sun Fire 6800 is running firmware 5.14.0 on the SC. The Sun Fire 6800 has a Capacity On Demand 2.0 CPU board, consisting of four CPUs. The CPU board is configured into the test domain when needed.

The Sun Fire V1280 server contains a single CPU board consisting of four CPUs.

Qualifying a Solaris Configuration

This section defines what qualifying a Solaris configuration is, describes the tested and supported software chosen in this article for minimization, and defines what is not supported.

This section contains the following topics:

- “What Is Qualification?” on page 8
- “What Software Is Supported?” on page 8
- “Solaris OE Versions” on page 8
- “Solaris Secure Shell” on page 8
- “Domain-Specific Components” on page 9
- “Support Software Components” on page 10
- “Additional Applications” on page 12

What Is Qualification?

Qualification is the process of identifying packages needed for an application by applying the minimization methodology to it. (See “Minimization Methodology” on page 21.)

The software and software versions in this article were chosen as a representative sample, meeting many customer needs based on typical usage of Sun Fire V1280, 6800, 12K, and 15K domains.

What Software Is Supported?

The software mentioned in this article is provided as an example of minimizing a Solaris domain and does not imply that Sun provides support for issues with third-party applications. For example, although ORACLE 9i is used as an example in this article, Sun does not provide support for the ORACLE software or attempt to produce a minimized configuration for it.

Sun provides support only for a minimized domain the software runs on; providing that the minimized configuration is based on one of the defined profiles in this article and the configuration is not missing any of the required packages.

Sun does support other Sun applications such as Sun ONE Application Server, Sun Validation Test Suite software, and so on, described in this document.

Solaris OE Versions

We selected Solaris OE versions 8 and 9 as the baseline to use for domain minimization. For UltraSPARC III based Sun Fire V1280, 6800, 12K, and 15K systems, you can use Solaris OE versions 8 or 9. If you use Solaris 8 OE, we recommend Solaris 8 (2/02) OE. For Solaris 9 OE, we used the Solaris 9 (12/02) OE.

All software testing was performed on both Solaris OE versions 8 and 9, unless noted.

Solaris Secure Shell

Solaris Secure Shell is a replacement for unsecured services such as Telnet, FTP, and `r*(rcp, rsh, etc)` commands. Solaris Secure Shell provides a medium for secure communication between networked hosts, and it avoids the legacy security problems unsecured services had, such as sending passwords unencrypted and session data being prone to snooping network traffic.

Solaris 9 OE incorporates bundled and supported Secure Shell software. Solaris 8 OE does not include supported Secure Shell software.

This article uses OpenSSH in its Solaris 8 OE domain testing; however, it is not a supported element in the configuration. OpenSSH can be downloaded at <http://www.sunfreeware.com>. For more information about OpenSSH, refer to <http://www.openssh.org> and to the Sun BluePrints book titled *Secure Shell in the Enterprise*.

Domain-Specific Components

Sun Fire systems have critical components in them to increase reliability, availability, and serviceability (RAS). The software is specialized for the hardware it supports and is redundant on other systems.

Dynamic Reconfiguration

Dynamic reconfiguration (DR) is the process of modifying a running domain by adding or removing hardware components, such as CPU or IO boards, without bringing the domain down.

Capacity On Demand 2.0

Capacity on Demand 2.0 software allows enforcement of policies regarding usage of additional CPU boards. It is a facility that allows adding new boards to domains by installing RTU license keys. It provides an ability to monitor capacity-on-demand resources through commands executed on the SC.

Capacity on demand is a CPU-only transaction. The software is part of Sun Management Services 1.3 on the Sun Fire 12K and 15K, and part of the firmware on the Sun Fire 6800 SC.

Capacity On Demand 2.0 software uses the License Processing System (LPS) on the Sun Fire 6800, 12K, and 15K SC for tracking usage. LPS is stored in the non-volatile RAM (NVRAM) on Sun Fire 6800 SCs and stored in the file system of the Sun Fire 12K and 15K systems.

Note – The Sun Fire V1280 firmware 5.13.13 does not support Capacity On Demand 2.0 software.

Sun Management Services 1.3

Sun Management Services 1.3 is the latest version of the Sun Fire 12K and 15K domain management software. It allows administrators to perform configuration and administrative tasks to domains; allows dynamic reconfiguration events to be scheduled; allows monitoring of running domains; and allows diagnostics to be performed on domains.

Sun Management Center 3.5 Domain Agent Software

Sun Management Center 3.5 is the standard GUI interface that Sun uses for configuring and monitoring system resources.

Additional products build on top of the Sun Management Center framework and use its plug-in architecture to expose their functionality. A good example of this is the SMS 1.3, which uses the plug-in architecture of Sun Management Center to provide the ability to execute dynamic reconfiguration from it.

Sun Management Center 3.5 server is installed on an external Solaris system. Software has to be installed on the domain for Sun Management Center 3.5 to be able to monitor it. Only the domain agent software needs to be installed.

Note – Some of the examples in this article refer to Sun Management Center 3.0. Note that Sun Management Center 3.5 became available during the writing of this article, and it is that version which is supported on minimized configurations defined in this article. The most important thing to gain from the examples that refer to Sun Management Center 3.0 is the problem being addressed, rather than the fact it occurred with a specific version of the software.

Support Software Components

The following applications allow users to maintain the integrity of their systems by keeping up-to-date with system patches. Also, these applications allow Sun service personnel to quickly and efficiently gather valuable information about system state.

Solaris Patch Manager 1.0

Solaris Patch Manager is a patch management system that allows easy analysis of patches required for a domain; provides an ability to automatically keep patch levels current; and can validate the authenticity of signed patches that are downloaded.

Sun Explorer 4.1

Sun Explorer data collector is a collection of scripts that gathers system information and generates a report.

Note – Generated reports can be automatically emailed to Sun, if outbound Simple Mail Transport Protocol (SMTP) is enabled.

Sun Explorer data collector is included with Sun Remote Services Net Connect 3.0. The most current version as of this article is version 4.1.

Sun Remote Services Net Connect 3.0

Sun Remote Services Net Connect is a web-based, remote monitoring, data collection, and system configuration reporting service.

It provides an ability for an administrator to self-monitor Sun systems. For customers with support contracts, such as SunSpectrum Gold or SunSpectrum Platinum, Sun Remote Services Net Connect data output can be sent to Sun for analysis.

Note – Sun Remote Services Net Connect 3.0 currently does not work on the Sun Fire V1280. Check for software patches or updates after the release of this article.

Sun Remote Services Net Connect 3.0.x should be installed on each domain of a system and configured according to the *Customer Installation Guide*. The security implications for installing and configuring Sun Remote Services Net Connect are beyond the scope of this article.

Note – Sun Remote Services Net Connect 3.0 installs Sun Explorer data collector as part of the installation. It inspects the system for any installed versions of Sun Explorer data collector. If a different release is present, such as v4.0, then Sun Remote Services Net Connect 3.0 installation routine asks to uninstall the other version and install the version included with Sun Remote Services Net Connect 3.0. If the user does not select uninstall, then the Sun Remote Services Net Connect 3.0 installation aborts.

Sun Validation Test Suite 5.1

Sun Validation Test Suite 5.1 performs diagnostic tests to validate Sun hardware by checking the connectivity and functionality of hardware devices, controllers, and peripherals.

Additional Applications

The following applications are often bundled with the Solaris 9 OE. These applications are compatible with Solaris 8 OE and are tested on both versions.

ORACLE 9i

ORACLE 9i is a fully featured version of Oracle's RDBMS. This application is popular for use on domains.

Sun ONE Software

Sun ONE (formerly iPlanet™) Application Server is a subset of the components that make up the Sun Open Network Environment suite of applications.

The components on which this article focuses are the following:

- Sun ONE Web Server 6.0.
- Sun ONE Application Server Enterprise Edition 6.5.
- Sun ONE Application Server Platform Edition 7.0.

Note – Sun ONE Application Server Enterprise Edition 7.0 was not available at the time of writing this article.

Automating Domain Installations

The JumpStart framework is used in this article to install the OS and packages required to support an application. Installing and qualifying a Solaris configuration of a domain for an application is a repetitive task, so the automation that JumpStart technology provides is very desirable; it simplifies the duplication of effort and reduces user errors.

This section describes how JumpStart technology is used for automating minimized installations and applying critical patches.

This section contains the following topics:

- “Configure the JumpStart Server” on page 13
- “Configure Minimal Domain Install Profile” on page 14
- “Install Patches” on page 15

Configure the JumpStart Server

The following list briefly summarizes the process of setting up a JumpStart server for installing a domain after a minimal profile is defined.

1. Verify the JumpStart infrastructure.
 - a. Build and configure a JumpStart environment (not covered in this article, see “Related Resources” on page 39 for other sources).
 - b. Configure the domain to use the JumpStart environment (not covered in this article).
 - c. Verify that the JumpStart environment has the appropriate Solaris OE versions available.
2. Configure minimal domain install profiles.
3. Install most recent patch clusters on JumpStart server.

Note – Installing the Solaris Security Toolkit software on the JumpStart server is the recommended practice. This allows minimized profiles to be configured with ease, allows patches to be easily installed using pre-existing finish scripts within the Solaris Security Toolkit software, and allows automated installation of packages that are not part of the Solaris OE. For more information about Solaris Security Toolkit features, refer to the Sun BluePrint book titled *Securing Systems with the Solaris Security Toolkit*.

Refer to the Sun BluePrints book *JumpStart™ Technology—Effective Use in the Solaris™ Operating Environment* for instructions on configuring the JumpStart server and verifying the software.

Configure Minimal Domain Install Profile

Minimized domains are defined by specifying a JumpStart profile in the `rules` file on the JumpStart server.

Profiles are located on the JumpStart server. They consist of the packages to be added and removed from a metacluster.

A metacluster is a collection of packages that `suninstall` (the subsystem for installing the Solaris OE) uses to specify what to install on the system. The usual metacluster installed to a domain is `SUNWCXall` and contains every package in the OS, but this is mostly unnecessary. The minimization process identifies the packages within metacluster `SUNWCXall` that are needed and separates them from those that are not.

A minimized profile is specified as a subset of the smallest metacluster available, which is currently metacluster `SUNWCreq` in Solaris OE versions 8 and 9. Within this metacluster are packages that are not needed. Also, some additional packages are needed that are not in `SUNWCreq` but are in `SUNWCXall`.

To delete unwanted packages, use the `delete` keyword in the profile for the package. To add needed packages, use the `add` keyword in the profile for the package.

Additional keywords are available that can be added to a profile. These are beyond the scope of this article.

A minimized profile might look like the following.

```
install_type    initial_install
cluster SUNWCreq
package SUNWxwdvx delete
package SUNWxwmox delete
package SUNWadmfu add
package SUNWadmfw add
[...]
```

Each application to be installed is likely to contain additional Solaris packages that must be installed. These are added to the minimal profile. The process for discovering these is covered in “Minimization Methodology” on page 21.

Install Patches

Installing patches is critical to the security of any minimized system. Download the latest Solaris Recommended Patch Cluster from SunSolveSM and install it on the Jumpstart server.

For minimized systems, only patches that contain some or all of the packages already installed on a minimized system are patched.

The patch cluster install script uses `patchadd` to install patches. The return code given by `patchadd` can often be misleading. The `patchadd` command reports return “code 8” when it can only partially install a patch. Also, it reports return code 8 if no patches are added, because none of the packages are installed on the minimized system.

To see what `patchadd` modified, type something similar to the following.

```
# patchadd -p | grep 111293-04
Patch: 111293-04 Obsoletes: 111052-01 Requires: Incompatibles:
Packages: SUNWcsl SUNWcslx SUNWcsr
```

The patching of a system occurs by executing a finish script specified in the JumpStart install server configuration. The finish script runs the cluster patch install script, which installs the patches in the patch cluster. The patch cluster should be copied to the JumpStart server. The finish script is executed by specifying it in the `rules` file. This is the same file that specifies the profile to install during a JumpStart installation.

Note – Applying the patch cluster might install new packages if they form part of a patch. For example, when installing the patch cluster, after a JumpStart of the profile for X application support under Solaris 8 OE occurred, the patching process installed new package SUNWxcu4.

Using Scripts to Qualify a Solaris Configuration

The general strategy for producing a minimized configuration for an application requires examining all files that make up the application installation. The idea is to identify dependencies these files have in Solaris OE such that only the required parts of Solaris OE are kept. The reason for this is that applications do not uniformly specify the packages in Solaris OE that they have dependencies on, so these have to be determined manually. Even Solaris OE packages suffer from this deficiency.

This operation uses packages because they are the units by which an installed Solaris system is defined and controlled. Removing individual files from a Solaris OE package is not a supported method of minimization.

To reduce the burden of finding the needed Solaris OE packages, several Perl and Korn shell scripts were developed. This section describes these scripts. (See the “Related Resources” on page 39 for the Web address from which to obtain these scripts.)

These scripts use techniques for finding packages by identifying the dynamic dependencies an executable or dynamic library has. A dynamic library can have dependencies on other dynamic libraries. The scripts make use of these properties to provide a systematic way of identifying the packages that executables and dynamic libraries have dependencies on, and which packages these dynamic libraries are in.

This section contains the following topics:

- “Find Dependent Files Using `truss` Command” on page 17
- “Find Dynamic Library Dependencies Using `ldd` Command” on page 18
- “Find Package Dependencies” on page 18
- “Find Multiple Packages and Files” on page 19

Find Dependent Files Using `truss` Command

Run all executables in an application with `truss` to find all dynamic libraries, configuration files, and data files opened successfully. When an executable command from the application is run, various libraries are opened by the runtime linker at startup. Additionally, configuration and data files are opened as the process is in execution. An executable is made up of dynamic libraries that are members of Solaris OE packages. Configuration and data files are members of Solaris OE packages. These relationships are captured by using the `truss` command to track system calls.

The `scan.pl` script takes the output of running `truss` on an executable and analyzes the output to find which dynamic libraries, configuration files, and data files were successfully opened at runtime.

These files are then cross-referenced with the Solaris OE package management system to find which packages these files belong to.

The following is an example of running the `truss` command on an executable to produce output, and `scan.pl` processing the output to find required Solaris OE packages.

```
# truss -f -o /tmp/truss.out -topen,open64,stat,exec /bin/ls \  
/tmp/truss.out  
# scan.pl -p < /tmp/truss.out  
Needed packages:  
    SUNWcsl  
    SUNWcsu  
    SUNWkvm
```

Note – The `truss` command is used here instead of `pldd` because `pldd` can only provide a snapshot of the process in execution at a given time. Because the process must be running, processes that execute for short durations, such as `/bin/ls`, would not be guaranteed to return correct results. Additionally, there is an issue when the snapshot is taken `pldd` would not know that a `dlopen` and `dlclose` might have occurred. The `truss` command overcomes these limitations by logging the calls over the lifetime of the process from startup to termination.

Find Dynamic Library Dependencies Using `ldd` Command

Run `ldd` against all executables in the application to find dynamic library dependencies that the runtime linker links in when running the executable. Next, find the packages that the dynamic libraries are in. An application often has its own dynamic libraries. These libraries reference other dynamic libraries in Solaris OE. Normally, dynamic libraries are linked in at runtime when an application executable starts up. This process is not always guaranteed to happen because the application might `dlopen` the library instead, and only when it's required.

The `ldd.ksh` script takes one or more executables or libraries and finds which packages contain the dynamic libraries. It does this by cross-referencing the `ldd` command output with a call to `pkgchk` to identify the package a dynamic library is in.

The `ldd.ksh` script also finds libraries that might be missing from an installation by processing `ldd` output. The `truss` command has limited ability to achieve this task, and provides incomplete information.

The following is an example of running `ldd.ksh` on an executable. The `-s` flag suppresses detailed output.

```
# ldd.ksh -s /opt/SUNWsrspcx/bin/srsproxy
Couldn't find package for
      libaio.so.1 => (file not found)
SUNWcsl SUNWlibC SUNWlibms
```

Find Package Dependencies

An executable (or dynamic library) typically requires multiple dynamic libraries to be able to execute. The chances of all the dynamic libraries being contained in the same package as the executable is small, which means there is a relationship (a dependency) between the packages containing the dynamic libraries and the package containing the executable. Solaris OE tracks this dependency relationship using dependency lists in the Solaris OE package management system.

Any package normally has a relationship back to mandatory core Solaris OE packages. The `depend-tree.pl` script constructs these dependencies from a starting package all the way back to the core Solaris OE packages.

It is always important to maintain dependencies among packages, because these are critical to Solaris OE. Also, it reduces the risk of packaging problems that might occur when patching a system.

Minimization has some requirements that do not fit well when following dependency links. The main ones are from following dependencies for `SUNWnamos` and `SUNWdtdst`. Issues arising from these are covered in the next section and in “Removing Unneeded Dependent Packages” on page 24.

The `depend-tree.pl` script works out the dependency tree that a package has. The `depend-tree.pl` script pulls details of dependencies all the way to the root packages. Root packages do not depend on anything else. This dependency information is kept in the Solaris OE package management system.

The following is an example of running `depend-tree.pl` to find package dependencies that it has on other Solaris OE packages.

```
# depend-tree.pl -Hu SUNWesu
SUNWcar SUNWkvm SUNWcsr SUNWcsu SUNWcsd SUNWcsl SUNWp15u SUNWp15v
```

The command line argument `-HuN` means suppress the header; display unique packages only (no duplicates); and ignore `SUNWnamos` package dependencies.

The `-N` option is supplied because some executables open locales in `SUNWnamos`, despite `LANG_ALL` being set to `C`. This action causes the dependencies for `SUNWnamos` to be included. This action is undesirable because a lot of these are X fonts packages and are simply not needed. Without the `-N` option, the output for `SUNWnamos` would be as follows.

```
# depend-tree.pl -Hu SUNWnamos
SUNWcar SUNWkvm SUNWcsr SUNWcsu SUNWcsd SUNWi15cs SUNWi15rf
SUNWeurf SUNWilcs SUNWxwplt SUNWesu SUNWcsl SUNWp15u SUNWp15v
SUNWxwdv SUNWxwfmt SUNWxwice SUNWdtdcor SUNWlibms SUNWcpp SUNWzlib
```

Find Multiple Packages and Files

The scripts described in previous sections all operate on a single executable, a dynamic library, or a package. To maximize the ability to operate at a more granular level, two wrapper functions, `meta.ksh` and `pkg.ksh`, are used to operate on multiple packages and files. This operation allows a list of packages or directories to be given to `pkg.ksh`.

meta.ksh

This script is a wrapper around `scan.pl`, `ldd.ksh`, and `depend-tree.pl`. It takes an executable or a shared library as an argument and executes the first two scripts to get the Solaris OE packages required. These are then passed into `depend-tree.pl` to get dependencies, and formatted for output.

The `-q` option tells the script to suppress detailed output. An example of running `meta.ksh` is as follows.

```
# meta.ksh -q /usr/openwin/bin/xterm
Package:          SUNWxwopt
truss: SUNWcsd SUNWcsl SUNWcsr SUNWcsu SUNWkvm SUNWlccom SUNWlibms
SUNWxwice SUNWxwopt SUNWxwplt
ldd: SUNWcsl SUNWlibms SUNWxwice SUNWxwplt
depend: SUNWcar SUNWkvm SUNWcsr SUNWcsu SUNWcsd SUNWxwplt SUNWesu
SUNWcsl SUNWpl5u SUNWpl5v SUNWxwdv SUNWxwfmt SUNWxwice SUNWdtcor
SUNWlibms SUNWcpp SUNWzlib SUNWzlib
```

The `meta.ksh` script will kill `-9` the executable within a few seconds after it has the information it requires, if the executable does not gracefully exit.

pkg.ksh

This script is the top-level script that operates on packages. If the `-d` option is specified, it operates on directories.

It takes a package or a directory and iterates over its contents to find all executables and shared libraries. When it finds one, it calls `meta.ksh` and processes the output. It produces a list of package dependencies as output.

Specifying the `-t` option takes a profile template, merges in application packages needed from Solaris OE, and writes the merged file to stdout. Adding the `-o` option writes the template to the specified file.

If the `-s` option is specified, `pkg.ksh` searches the directories specified by the `-d` option or the packages specified by the `-s` option for dynamic libraries and configures the system to point to their location temporarily.

See the Case Study in Part II of this article for examples of `pkg.ksh` in use.

At this time, the usage template for `pkg.ksh` is as follows.

```
# pkg.ksh
Usage: pkg.ksh [-s] [-t <profile template>] [-o <new profile>] \
[[-d <dir>...<dir>] [-p <pkg>...<pkg>] | [<pkg>...<pkg>]]
-s means search for *.so* dynamic libraries and set LD_LIBRARY_PATH
and LD_LIBRARY_PATH_64 as appropriate.
```

Minimization Methodology

The section defines a reusable methodology to produce a minimized domain configuration for an application by defining the packages needed to make up the minimized domain. This section describes common problems and solutions found when applying the minimization methodology to chosen applications.

The procedure requires the domain first to be installed with the Solaris OE meta-cluster `SUNWCXall`, to identify packages required, and to establish package dependencies.

The `SUNWCXall` meta-cluster of packages contains all Solaris OE and OEM software included in the Solaris OE distribution. This meta-cluster contains all the Solaris OE packages needed for an application to create the minimized profile. Other packages might need to be downloaded and installed, but these must be installed after the JumpStart installation has occurred.

Once the minimized profile is defined for the application, a JumpStart install operation is executed on the domain with the minimized profile to check that the install and main functionality of the application is error-free.

Having a second domain available, which is installed with `SUNWCXall`, can reduce the time needed to produce minimized configurations for multiple applications. Also, it can reduce time if references are made to the fully installed system during the minimization tasks.

Note – It is assumed that the locale settings for applications installed are in the default language, U.S. English. This setting requires setting all environmental variables `LC_*` to `C` in your shell before running `pkg.ksh`. The `pkg.ksh` has an option to do this when it is run. Localization is not supported for the applications in this article other than for the default language. Setting the locale environmental variables to other settings might include additional packages in the minimized configuration such as locale library packages, additional font packages, X packages, and locale packages.

To apply the minimization methodology, see the following topics:

- “Review Application Documentation” on page 22
- “Install the Application” on page 23
- “Identify Packages Needed by an Application” on page 24
- “Build Minimized Profile” on page 27
- “Use JumpStart With Minimized Profile” on page 27
- “Install Software” on page 27
- “Resolve Application Install Errors” on page 28
- “Run the Primary Application and Look for Errors” on page 38

Review Application Documentation

Often the install documentation accompanying an application provides information about where application files are installed.

Install documentation might also define if the software has any package prerequisites. These can be found during the install process but, with some large applications (such as ORACLE 9i), this can be a lengthy process, so reviewing documentation can be a time saver.

Identify any Solaris OE packages required by an application so you can add them to the minimized JumpStart profile.

Documentation provides important information for setting the location of dynamic libraries. Two methods are available for specifying the location of dynamic libraries to the system: `crle(1)` and the environmental variable pair `LD_LIBRARY_PATH` and `LD_LIBRARY_PATH_64`. The recommended method is to use `crle(1)` wherever possible, but the documentation might require using the environmental variable pair method.

The location of the dynamic libraries should be configured before running `pkg.ksh`. Note that `pkg.ksh` has the ability to try to determine the location of the dynamic libraries, but this method is not fool proof.

Install the Application

The installation operation is a multistep process. Installing an application on a full system and minimized system should be exactly the same. Any differences that occur might cause the installation to take a different path through the installation process, which could have unexpected side effects. It is important that the install should be a typical installation, causing as much of the application to be installed as possible. This approach allows the minimization process to be as effective as possible.

▼ To Install an Application

1. Run `pkginfo` to capture the “before” installation snapshot, using the `script` command.
2. Log the install using the `script` command, so the process can be reviewed later.
3. Extract the application contents and examine the directory structure for packages that might be installed.
4. Install the application on the system.
If the application comes as a compressed `tar` file, then it might be possible to simply extract to a directory and perform subsequent steps in this methodology there, because no packages need to be installed.
5. Run `pkginfo` to capture the “after” installation snapshot, using the `script` command.
6. Run `diff` on the two snapshots to identify packages installed by the application.
7. After installing the application, make sure the location of application dynamic libraries is configured according to the documentation.

Note – Set `LC_*` to `C` to prevent locale packages from being included.

Failure to set the application dynamic libraries location properly can cause the minimization call to `pkg.ksh` to yield unwanted packages. For example, installing Sun Explorer without configuring where its libraries are located (the default location is `/opt/SUNWexplo/lib`) causes `pkg.ksh` to require `SUNWfruid`. However, the dynamic libraries that Sun Explorer uses are a subset of `SUNWfruid`, with each library needed available. So, configuring dynamic libraries properly prevents `SUNWfruid` from being needed on a minimized system. The `pkg.ksh` can attempt to guess what the dynamic library directories are by searching for them. Specifying this option can cause a lengthy wait for `pkg.ksh` to complete.

8. Identify where all the packages and files from the install are located in the file system.

For completeness of minimizing an application, identification is needed of where all the packages and files from an install are located in the file system. This task is often simple, but might be complex, depending on how the install works. It's completely application dependent.

For example, installing the Sun Management Center 3.5 agent software produces an install log that contains the names of all packages installed. An important distinction here is the difference between Solaris OE packages and application packages that are part of Sun Management Center 3.5. Application packages should not be added to the minimized profile.

Identify Packages Needed by an Application

Executables and dynamic libraries within an application make use of dynamic libraries and other files, such as configuration files, startup scripts for daemons, and so forth, within the Solaris OE.

These dynamic libraries and other files are contained within Solaris OE packages. Identify the dynamic libraries by running `pkg.ksh` against the installed application files or packages. Identify the other files from package dependencies.

The packages required must have their dependency relationships to other packages maintained by following the dependency lists for each of them.

These packages and their dependencies need to be added to the minimized profile so they are installed as part of the minimized package configuration.

Removing Unneeded Dependent Packages

A general rule for package dependencies is that they should be satisfied whenever they exist. This rule is mostly the case with our minimization methodology, however, there are times when extra packages required to satisfy dependencies are simply not needed and are redundant.

The situations where dependencies are not needed is a shortcoming of how packages are sometimes specified in Solaris OE. A trade-off needs to occur with respect to stringently adhering to the package dependencies and weighing this with the goal of achieving a secure system through minimization.

A good example of unwanted package dependencies is seen by reviewing the install log output by the `script` command when installing the Sun Management Center 3.0 agent software. This troubleshooting requires using commands like `ldd` and `pkgchk` to find what is causing unexpected package requirements and whether their inclusion is fully justified.

After running `pkg.ksh` against the install on a fully loaded system, reviewing the output log shows the following lengthy dependency list for the library `libcalendarservice.so.1.0`.

```
/opt/SUNWsymon/base/lib/sparc-sun-solaris2.8/  
libcalendarservice.so.1.0: ELF 32-bit MSB dynamic lib SPARC  
Version 1, dynamically linked, not stripped  
Package:          SUNWesaes  
truss: SUNWesaes  
ldd: SUNWcsl SUNWtdtst  
depend: SUNWcar SUNWkvm SUNWcsr SUNWcsu SUNWcsd SUNWdtbas SUNWcsl  
SUNWxwdv SUNWxwplt SUNWesu SUNWpl5u SUNWpl5v SUNWxwftnt SUNWxwice  
SUNWdtcor SUNWlibms SUNWcpp SUNWzlib SUNWxwft SUNWxwopt SUNWmfrun  
SUNWctpls SUNWctpls SUNWtdmn SUNWtdte SUNWocf SUNWj3rt SUNWlibc  
SUNWocfr SUNWocfr SUNWdticn SUNWtltk SUNWdtjxt SUNWdtscm
```

Investigating this library further, we see it has `truss` and `ldd` dependencies on three packages directly. One of these packages is creating the unexpected dependency list. Running `depend-tree.pl` on `SUNWesaes` and `SUNWcsl` yields nothing substantial, but running against `SUNWtdtst` shows the cause.

Manual investigation of `libcalendarservice.so.1.0` using the `ldd` command shows that the library `/usr/dt/lib/libcsa.so.0` looks most interesting because its path name contains “dt.”

```
# ldd /opt/SUNWsymon/base/lib/sparc-sun-solaris2.8/  
libcalendarservice.so.1.0  
libcsa.so.0 => /usr/dt/lib/libcsa.so.0  
libdl.so.1 => /usr/lib/libdl.so.1  
libnsl.so.1 => /usr/lib/libnsl.so.1  
libc.so.1 => /usr/lib/libc.so.1  
libmp.so.2 => /usr/lib/libmp.so.2  
/usr/platform/SUNW,Ultra-2/lib/libc_psr.so.1
```

The `libcsa.so` is a member of the package `SUNWdtdst`, as speculated, which is the CDE desktop applications package, as shown by the following.

```
# pkgchk -l -p /usr/dt/lib/libcsa.so.0
Pathname: /usr/dt/lib/libcsa.so.0
Type: regular file
Expected mode: 0755
Expected owner: root
Expected group: bin
Expected file size (bytes): 190800
Expected sum(1) of contents: 27313
Expected last modification: Mar 13 08:46:49 PM 2002
Referenced by the following packages:
    SUNWdtdst
Current status: installed
```

If the dependencies for `SUNWdtdst` are followed to their logical conclusion, then the following additional packages are also required.

```
# depend-tree.pl -u SUNWdtdst
Unique packages
  SUNWcar SUNWkvm SUNWcsr SUNWcsu SUNWcsd SUNWdtbas SUNWcsl
  SUNWxwdv SUNWxwplt SUNWesu SUNWpl5u SUNWpl5v SUNWxwft SUNWxwice
  SUNWdtcor SUNWlibms SUNWcpp SUNWzlib SUNWxwft SUNWxwopt SUNWmfrun
  SUNWctpls SUNWctpls SUNWtdmn SUNWtdte SUNWocf SUNWj3rt SUNWlibC
  SUNWocfr SUNWocfr SUNWdticn SUNWtltk SUNWdtjxt SUNWdtscm
```

So, to satisfy the library requirement, the dependencies require Java™, X library, and CDE packages to be installed.

This requirement is not necessary because only a single-dynamic library from this package is needed, and this fact is the reason that dependency chains for shared libraries are not followed when `pkg.ksh` is run. We confirm this by searching the script output file for other dependencies on `SUNWdtdst`. There are none.

Without these shared library dependencies, only a few packages are needed to get Sun Management Center 3.0 running on a domain. With them, you have a potential security problem with redundant package inclusion.

Build Minimized Profile

A minimized profile comprises input from three areas:

- Minimal bootable Solaris OE profile based on `SUNWCreq`, with required additional packages.
- Installation package recommendations in documentation of the application, if applicable.
- The package list produced by running `pkg.ksh` against application packages or directories.

To the minimal bootable Solaris OE profile, append the additional packages needed using the keyword `add`.

Use JumpStart With Minimized Profile

Add the minimized profile for the application to the JumpStart server and perform a domain install using it.

Only those packages specified will be installed and the system will be patched with the recommended cluster patch.

The installation logs for the JumpStart server will highlight any errors that occur during installation. These are located in `/var/sadm/system/logs` on the domain. If any errors occur, correct them and repeat the install procedure.

Install Software

Some software such as Solaris Secure Shell, which is now part of Solaris 9 OE, can be installed automatically during a JumpStart installation, by specifying it in the JumpStart profile.

Software such as Sun Validation Test Suite, Sun Explorer data collector, and so on, are added to a domain after JumpStart installations by using `pkgadd`. This process can be automated with an appropriate response file and finish script. The package to be installed should be stored on the JumpStart server.

Other software such as ORACLE 9i and Sun Management Center 3.5 have manual install requirements. The best you can do is use a JumpStart finish script to extract their contents to the `/opt` directory during the JumpStart installation, and manually run the install after JumpStart installation is completed.

The value of doing so needs to be evaluated in the context of how often domain installation is likely to occur and the storage space requirements of the application. ORACLE 9i comes on three CDs, which is almost 2 gigabytes of disk space and might be considered excessive if installation occurs only once.

Note – It is possible to make custom packages for an application using the command `pkgmk` command. This command allows the deployment of the applications with a minimum of effort after a JumpStart install has occurred.

Resolve Application Install Errors

Errors that occur during application install are presented in different ways. The exact nature of these is hard to define, but relatively obvious when they occur because they typically lead to an aborted install or display a message on the console.

Generally, there are two states that should be achieved and be free of error before testing any application:

- Install the application successfully without error.
- Run `pkg.ksh` on application directories, as done in Step 3 of the minimization methodology, without any missing dynamic libraries. If any are missing, then this should be consistent with installation on the full system.

The following sections highlight and describe some of the problems encountered when using the minimization methodology and how the issues are resolved.

Quick Method of Installing Needed Packages on a Minimized System

If the installation of an application fails on a minimized system, then the missing packages need to be identified on a fully installed system.

Once these have been found, by NFS mounting the directory containing the image of Solaris, it is possible to add the missing packages directly to the minimized system by changing to the package directory of the NFS mounted Solaris install image, and using the `pkgadd -d . <missing packages>` command.

This command installs missing packages to the minimized JumpStart installation. Next, reinstall the application and check for further errors. If successful, add the missing packages to the minimized JumpStart profile, perform a new minimized JumpStart install, and reinstall the application and check for errors. This process ensures that no unexpected errors or inconsistencies occur during JumpStart installation.

Note – By default, NFS is an unsecured service, but is a reasonable choice for this operation. For more information about securing the use of NFS, refer to the Sun BluePrints OnLine article titled “Solaris Operating Environment Security - Updated for Solaris 9 Operating Environment.”

Dealing With Missing Packages

Installing an application on a minimized system can sometimes install more software than was added for installation on a full system. This happens because the application might install pre-requisite packages that are missing on the minimized system. These are already installed on the full system, so they do not need to be added.

When installing Patch Manager on a full system, the installer examines the system to see if packages are already installed. If so, then it does not install the Java Runtime Environment, as can be seen from the following Patch Manager install output.

```
Beginning installation ...
Now installing supporting packages...
    SUNWjhrt
    == installed version is newer
    SUNWsdb
    == installed successfully
    SUNWapcy
    == installed successfully
    SUNWppmn
    == installed successfully
    SUNWcert
    == installed successfully
    SUNWpmgr
    == installed version is newer
    SUNWppro
    == installed successfully

Installation is complete and verified. Be sure to:
[...]
```

However, if Patch Manager is installed on the minimized system, the Java™ Runtime Environment and its companion packages are installed, as the following shows:

```
Beginning installation ...
Now installing supporting packages...
  SUNWj3rt
  == installed successfully
  SUNWlj3rt
  == installed successfully
  SUNWj3dev
  == installed successfully
  SUNWjhrt
  == installed successfully
  SUNWjsse
  == installed successfully
  SUNWsdb
  == installed successfully
  SUNWapcy
  == installed successfully
  SUNWppmn
  == installed successfully
  SUNWcert
  == installed successfully
  SUNWpmgr
  == installed successfully
  SUNWppro
  == installed successfully

Installation is complete and verified. Be sure to:
[...]
```

Note – If packages are added in this manner that are part of the Solaris OE, then it is recommended practice to reapply the Solaris Recommended Patch Cluster. See “Install Patches” on page 15 for more information about patches.

Subsequently, running `pkg.ksh` on the minimized system shows many shared libraries are missing as, in the Patch Manager example, the X shared libraries were not installed to satisfy Java dependencies. Running `pkg.ksh` on the fully installed system did not highlight these, due to incorrect dependencies in Patch Manager.

The extracted `tar` file for installing Patch Manager should have been examined before starting the minimization process. That would have shown the packages this application is most likely interested in. The following directory listing shows the packages Patch Manager could potentially install.

```
# ls pproSunOSsparc5.9jre2.1
README      SUNWcert    SUNWj3rt    SUNWjsse    SUNWpmgr    SUNWppro
patchlist  SUNWapcy    SUNWj3dev    SUNWjhrt    SUNWlj3rt    SUNWppmn
SUNWsdb     setup
```

If Patch Manager dependencies were specified correctly for `SUNWppro` then executing `pkg.ksh` would yield all the packages required by following the dependency chain in the Solaris OE package management system.

To address this short coming, examine the application's directory structure to give hints of what to expect. Next, compare the `diff` output of the before and after for `pkginfo` on both installs of the fully installed and minimized system to spot inconsistencies.

In the Patch Manager case, the directory structure shows 11 packages, but installing on the full system only highlighted 7 packages, so this is something that raises a flag for further investigation.

Dealing With Java Runtime Environment and X Dependencies

The Patch Manager case mentioned in the previous section raises an interesting dilemma for performing minimization. For many applications, the use of the Java Runtime Environment is to display graphical applications, and it needs the X libraries available to produce these displays. But there is a further point; not every Java application is graphical. They can be console-only applications built to use the rich API that Java offers.

In the Patch Manager example, this application is a console-only application for updating patches on the system. It does not display a GUI, so although package dependencies show it requires various X libraries, these are not used, based on our experience of finding the minimal package configuration for the Patch Manager 1.0 application.

Note – Exercise caution when faced with the decision of removing required packages, even if you believe that the contents of the packages might not be used by the application. The ability to support such a configuration can be adversely impacted.

For Solaris 9 OE, the fact that Patch Manager is run console-only allows Patch Manager to be installed free of X library requirements. Solaris 8 OE is more restrictive when it comes to leaving out X packages, due to mandatory packages included in metacluster `SUNWCreq`.

For other applications, it requires a judgement call by the user, who has experience with the application, to decide if it is graphical or not and if it needs X libraries packages installed, as indicated by `pkg.ksh`.

A side effect of having no X libraries is that the Java Runtime Environment does not install due to a dependency that the Motif RunTime Kit has on the `SUNWmfrun` package.

This package must be installed or the following message is displayed when Patch Manager installs `SUNWj3rt`.

```
WARNING:
  The <SUNWmfrun> package "Motif RunTime Kit" is a
  prerequisite package and should be installed.

Installation of <SUNWj3rt> was suspended (interaction required).
No changes were made to the system.
```

Finding Missing Packages After Minimized Install

A lot of applications do not specify their package dependencies properly. These missing dependencies are often exposed when a shell script executes as part of the install process. The script tries running a missing command and fails because the package containing the executable is not installed.

The application install script making the assumption that the executable will be installed is not realistic for a minimized system.

During the install of Sun Remote Services Net Connect 3.0 on a minimized system, this type of dependency problem became apparent because install complains that `showrev` is missing, as is shown in the following install output.

```
--- user/cron interaction test passed successfully.

InstallNetConnect.003.000.000.sh: showrev: not found

Processing package instance <SUNWsrspcx> from </tmp/
NetConnectInstall/content/packages/SRSproxy.001.000.000.INSTALL/
content/packages/SUNWsrspcx>
```

showrev is often needed for installations and should always be installed as part of any minimized profile. The following shows which packages are needed when it is executed on a full system.

```
# pkg.ksh -d `which showrev`
list is /usr/bin/showrev
/usr/bin/showrev:      ELF 32-bit MSB executable SPARC Version 1,
dynamically linked, stripped
Package:              SUNWadmc
truss: SUNWadmc SUNWadmfw SUNWcsl SUNWcsr SUNWkvm SUNWlibC
SUNWlibms
ldd: SUNWadmc SUNWadmfw SUNWcsl SUNWlibC SUNWlibms
depend: SUNWcar SUNWkvm SUNWcsr SUNWcsu SUNWcsd SUNWcsd SUNWadmc
SUNWesu SUNWcsl SUNWcsl
Unique packages needed:
SUNWadmc
SUNWadmfw
SUNWcar
SUNWcsd
SUNWcsl
SUNWcsr
SUNWcsu
SUNWesu
SUNWkvm
SUNWlibC
SUNWlibms
```

This problem was not highlighted on the fully installed system because it was masked by the fact that SUNWadmc and SUNWadmfw, which contain showrev, were already installed.

Most of the packages identified by pkg.ksh are mandatory for a Solaris OE install. Those that are not mandatory need to be added to the minimized profile to install showrev. Make sure that Sun Remote Services Net Connect 3.0 installs properly. The packages needed by Sun Remote Services Net Connect 3.0 are listed in the following table.

TABLE 1 Mandatory Packages

Package Type	Description
SUNWadmc	System administration core libraries
SUNWadmfw	System and network administration framework
SUNWlibC	Sun Workshop compilers bundled libc
SUNWlibms	Forte Developer bundled shared libm

Dealing With Unresolved Dependencies

The methodology presented in this article uses a systematic way of going through each file of an installed application and figuring out the dependencies, if it is an executable or dynamic library.

A problem can occur where an executable is compatible with a specific version of the Solaris OE only. There may be libraries that cannot be resolved on a different version of Solaris OE, because the shared library versions might be different.

An example similar to this occurred when installing Sun Management Center 3.0 agent software on Solaris 9 OE.

Running `pkg.ksh` on the full system after installing the application shows missing libraries that clearly are not right, because no packages were removed.

```
# pkg.ksh SUNWeswga SUNWessta SUNWesspa SUNWesmod SUNWesmcp
SUNWescom SUNWesasm SUNWesamn SUNWesagt SUNWesaes SUNWesaem
SUNWesae
[...]
/opt/SUNWsymon/util/bin/sparc-sun-solaris2.8/snmp_edit: ELF 32-
bit MSB executable SPARC Version 1, dynamically linked, not
stripped
Package:          SUNWesagt
truss: SUNWesagt
ldd: Couldn't find package for:
      libXm.so.2 => (file not found)
      libXext.so.5 => (file not found)
      libXmu.so.5 => (file not found)
[...]
```

The same issue occurs running `pkg.ksh` on the minimized system. This is expected, because the minimized system is a subset of the fully installed system.

In the previous case, it turns out that `snmp_edit` is a little used, undocumented GUI wrapper program within Sun Management Center 3.0. Typically, this program is not required and, because it is the only executable that uses X libraries within Sun Management Center 3.0, is the reason that this application has no X package dependencies.

Items that are learned from the `snmp_edit` example are as follows:

- Not every executable or dynamic library is needed by an application. Where an executable has unusual dependencies, for example, with the `snmp_edit` executable being the only one that depends on X packages, it is always worth investigating further to understand if it is a main program. This judgement call differs from application to application. It is worth noting that this is a deficiency of automating minimization; this is not something that can be easily programmed to take account of. What the script tools can do is attempt to highlight rarely needed packages by specific dynamic libraries or executables based on frequency of use.

Note – By using Solaris Basic Security Module auditing, it is possible to log audit records for any executable that is launched. This is done when calls are made to the `execve()` kernel system call. By exercising the application functionality over a period of time, it is possible to analyze audit records to determine if a program is used. For more information, refer to the Solaris Security Auditing Web site at <http://www.sun.com/software/security/audit>.

- Dynamic libraries might not be installed on a system by default. If an executable cannot resolve dynamic libraries on a fully installed system, then one of the following might be the cause:
 - The command is not compatible with the installed Solaris version.
 - A custom package requires a special software load, for example, `SUNWlxml` and `SUNWlxmlx` on Solaris 8 OE.
- Dynamic libraries might be installed on the system yet not found. If an executable has unresolved dynamic libraries, then the cause might be one of the following:
 - An incorrectly configured application dynamic library directory, configured through `crle(1)` or `LD_LIBRARY_PATH` and `LD_LIBRARY_PATH_64`. An example of this is given in the next section.
 - A missing symbolic link for a library version. The likelihood of this cause is low, and its occurrence unusual, but the version discrepancy in `snmp_edit` could be this.

Resolving Installed Shared Library Configuration Errors

Before executing an application for testing purposes, it is important to examine the environment and confirm that no problems occurred.

The most significant check is validating that all libraries the installed application might use are installed. This check can be performed by running `pkg.ksh` against the packages or directories, as done on the fully installed domain.

One important point is that libraries not found might indicate a library path problem with the `crle(1)` setting or `LD_LIBRARY_PATH` and `LD_LIBRARY_PATH_64` environmental variable settings, rather than a package installation issue.

An example of this problem is found when applying the minimization methodology to Sun Validation Test Suite 5.1. After running `pkg.ksh` on the minimized system, a lot of shared libraries were installed, but their packages could not be identified.

```
# pkg.ksh SUNWvts SUNWvtsx
[...]
Unique packages needed:
Couldn't find package for /usr/lib/64/libc.so.1
Couldn't find package for /usr/lib/64/libdevinfo.so.1
Couldn't find package for /usr/lib/64/libdl.so.1
Couldn't find package for /usr/lib/64/libCrun.so.1
Couldn't find package for /usr/lib/64/libX11.so.4
Couldn't find package for /usr/lib/64/libaio.so.1
Couldn't find package for /usr/lib/64/libc.so.1
Couldn't find package for /usr/lib/64/libdl.so.1
SUNWbip
SUNWcar
[...]
```

Examining the script log produced when installing Sun Validation Test Suite 5.1 shows an executable with the problem libraries to focus on.

```
/opt/SUNWvts/bin/sparcv9/afbttest:      ELF 64-bit MSB executable
SPARCV9 Version 1, UltraSPARC1 Extensions Required, dynamically
linked, not stripped
Package:      SUNWvtsx
truss: SUNWcsu SUNWvtsx SUNWxwplx
ldd: Couldn't find package for /usr/lib/64/libCrun.so.1
Couldn't find package for /usr/lib/64/libaio.so.1
Couldn't find package for /usr/lib/64/libc.so.1
Couldn't find package for /usr/lib/64/libdl.so.1
[...]
```

More clearly, running `ldd` shows the libraries are resolved, indicating they are installed. The directory of `/usr/lib/64` is non-standard for 64-bit libraries in Solaris OE. The directory containing the libraries turns out to be a symbolic link to the traditional `sparcv9` directory for 64-bit libraries.

```
# ldd /opt/SUNWvts/bin/sparcv9/afbtest
libX11.so.4 => /usr/openwin/lib/sparcv9/libX11.so.4
libnsl.so.1 => /usr/lib/64/libnsl.so.1
libdl.so.1 => /usr/lib/64/libdl.so.1
libsocket.so.1 => /usr/lib/64/libsocket.so.1
libdga.so.1 => /usr/openwin/lib/sparcv9/libdga.so.1
libvtstest.so.1 => /opt/SUNWvts/lib/sparcv9/
libvtstest.so.1
libCrun.so.1 => /usr/lib/64/libCrun.so.1
libm.so.1 => /usr/lib/64/libm.so.1
[...]
# ls -lt /usr/lib/64
lrwxrwxrwx 1 root root 7 Feb 7 11:43 /usr/lib/64
-> sparcv9
```

Setting `LD_LIBRARY_PATH_64` to the `sparcv9` directory path resolves the problem and allows finding package dependencies to succeed.

```
# echo $LD_LIBRARY_PATH_64
/usr/lib/sparcv9
# ldd /opt/SUNWvts/bin/sparcv9/afbtest
libX11.so.4 => /usr/lib/sparcv9/libX11.so.4
libnsl.so.1 => /usr/lib/sparcv9/libnsl.so.1
libdl.so.1 => /usr/lib/sparcv9/libdl.so.1
libsocket.so.1 => /usr/lib/sparcv9/libsocket.so.1
libdga.so.1 => /usr/openwin/lib/sparcv9/libdga.so.1
libvtstest.so.1 => /opt/SUNWvts/lib/sparcv9/libvtstest.so.1
libCrun.so.1 => /usr/lib/sparcv9/libCrun.so.1
[...]
# pkg.ksh SUNWvts SUNWvtsx
[...]
Unique packages needed:
SUNWbip
SUNWcar
[...]
```

Run the Primary Application and Look for Errors

Run the application and test it to make sure basic functionality is operational. Error checking should be carried out continuously during this process. The output to the console, to `/var/adm/messages`, and to the application's own logs should be constantly monitored.

About the Authors

Nicholas O'Donnell has worked in the Enterprise Server Products (ESP) group at Sun for three years, developing software on the HPC ClusterTools products and, in the last year, working on Sun Fire server security.

He has over eight years experience with administration and development in such diverse areas as hand-written recognition systems for mail transports, web development at an Internet startup, and working for several major merchant banks on Wall Street.

Alex Noordergraaf has over ten years experience in the areas of computer and network security. As the Security Architect of the Enterprise Server Products (ESP) group at Sun Microsystems, he is responsible for providing technical leadership to define the security of Sun's next generation servers while addressing security for current products. He is the driving force behind the very popular freeware Solaris Security Toolkit. Prior to his role in ESP, he was a Senior Staff Engineer in the Enterprise Engineering (EE) group of Sun Microsystems, where he developed, documented, and published security best practices through the Sun BluePrints program. Published topics include: Sun Fire 15K system security, secure N-tier environments, Solaris OE minimization, Solaris OE network settings, and Solaris OE security. He has co-authored the following Sun BluePrint books: *Securing Systems with the Solaris Security Toolkit*, *Enterprise Security Solaris Operating Environment*, *Security Journal*, and *Jumpstart Technology-Effective Use in the Solaris Operating Environment*.

Prior to his role in EE, he was a Senior Security Architect with Sun Professional Services where he worked with many Fortune 500 companies on projects that included security assessments, architecture development, architectural reviews, and policy/procedure review and development. He developed and delivered an enterprise security assessment methodology and training curriculum to be used worldwide by SunPS. His customers included major telecommunication firms, financial institutions, ISPs, and ASPs. Before joining Sun, Alex was an independent contractor specializing in network security. His clients included BTG, Inc. and Thinking Machines Corporation.

Acknowledgements

The authors would like to recognize Martin Englund for writing the `scan.pl` script.

Related Resources

Publications

- Howard, John S. and Noordergraaf, Alex. *JumpStart™ Technology: Effective Use in the Solaris™ Operating Environment*, The Official Sun Microsystems Resource Series, Prentice Hall, October 2001.
- Noordergraaf, Alex. “Solaris Operating Environment Security: Updated for the Solaris 9 Operating Environment,” Sun BluePrints OnLine, December 2002, <http://www.sun.com/solutions/blueprints/1202/816-5242.pdf>.
- Noordergraaf, Alex and Benson, Tony M. “Securing the Sun Fire Midframe System Controller: Updated for SCapp 5.13, Solaris 8 (2/02), and Solaris 9,” June 2002, <http://www.sun.com/blueprints/0602/816-4940-10.pdf>.
- Noordergraaf, Alex and Brunette, Glenn. *Securing Systems with the Solaris Security Toolkit*, Sun Microsystems, Prentice Hall Press, ISBN 0-13-141071-7, June 2003.
- Noordergraaf, Alex and Nimeh, Dina. “Securing the Sun Fire 12K and 15K Domains,” Sun BluePrints OnLine article, February 2003, <http://www.sun.com/solutions/blueprints/0203/817-1357.pdf>.
- Noordergraaf, Alex and Nimeh, Dina. “Securing the Sun Fire 12K and 15K System Controllers,” Sun BluePrints OnLine article, February 2003, <http://www.sun.com/solutions/blueprints/0203/817-1358.pdf>.
- Noordergraaf, Alex and Watson, Keith. “Solaris Operating Environment Security - Updated for Solaris 9 Operating Environment,” Sun BluePrints OnLine, December 2002, <http://www.sun.com/solutions/blueprints/1202/816-5242.pdf>.
- Reid, Jason. *Secure Shell in the Enterprise*, Sun Microsystems, Prentice Hall Press, ISBN 0-13-142900-0, June 2003.

- *Solaris 8 Advanced Installation Guide, February 2002*, <http://docs-pdf.sun.com/816-2411/816-2411.pdf>.
- *Sun Fire V1280/Netra 1280 System Administration Guide*, <http://www.sun.com/products-n-solutions/hardware/docs/html/817-0509-10>.

Web Sites

Note – Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

- Solaris Security Toolkit software:
<http://www.sun.com/solutions/blueprints/tools>
- Solaris Security Auditing:
<http://www.sun.com/software/security/audit>
- Sun Fire 15K server:
<http://www.sun.com/servers/highend/sunfire15k/index.xml>
- Sun Fire 12K server:
<http://www.sun.com/servers/highend/sunfire12k/index.xml>
- Midframe and Midrange servers:
<http://www.sun.com/servers/midrange>
- Sun Fire V1280 server:
<http://www.sun.com/servers/midrange/sunfirev1280>
- Highend servers, dynamic reconfiguration:
http://www.sun.com/servers/highend/dr_sunfire
- Dynamic reconfiguration for Sun Fire 3800, 4800, 4810 and 6800 Midframe servers: http://www.sun.com/servers/midrange/dr_sunfire
- Capacity on Demand software: <http://www.sun.com/datacenter/cod>
- Sun Fire 15K/12K Servers System Management Services:
<http://www.sun.com/servers/highend/sms.html>
- Sun Management Center software:
<http://www.sun.com/software/solaris/sunmanagementcenter/index.html>
- Solaris Patch Manager software:
http://www.sun.com/service/support/sw_only/patchmanager.html

- **Sun Explorer data collector software:**
<http://sunsolve.sun.com/pub-cgi/show.pl?target=explorer/explorer>
- **Sun Remote Services Net Connect software:**
<http://www.sun.com/service/support/srs/netconnect>
- **Sun Validation Test Suite software:**
<http://www.sun.com/oem/products/vts>
- **Sun ONE Advantage software:**
<http://www.sun.com/software/solaris/licensing/advantage.html>
- **Recommended Patch Clusters software:**
<http://sunsolve.sun.com/pub-cgi/show.pl?target=patchpage>
- **Domain minimization profiles and scripts:**
<http://www.sun.com/solutions/blueprints/tools/>
- **Supplement CD: Lights Out Management 2.0. software:**
<http://docs.sun.com/db/doc/816-2583-10/6m8u2cujl?a=view>

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The docs.sun.com web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is <http://docs.sun.com/>

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: <http://www.sun.com/blueprints/online.html>