Sun Crypto Accelerator 6000 Board

User's Guide for Version 1.0

Part No. 819-5536-12
May 2010, Revision A

Please
Recycle

Contents

Declaration of Conformity

To receive a copy of the latest Declaration of Conformity (DoC) for the product, create an online request at (https://www2.sun.de/dct/forms/reg_us_1607_755_0.jsp), or send email to: compliance_request_ww@oracle.com.

Regulatory Compliance Statements

Your Sun product is marked to indicate its compliance class:

- Federal Communications Commission (FCC) — USA
- Industry Canada Equipment Standard for Digital Equipment (ICES-003) — Canada
- Voluntary Control Council for Interference (VCCI) — Japan
- Bureau of Standards Metrology and Inspection (BSMI) — Taiwan

Please read the appropriate section that corresponds to the marking on your Sun product before attempting to install the product.

FCC Class B Notice

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference.
2. This device must accept any interference received, including interference that may cause undesired operation.

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/television technician for help.

Shielded Cables: Connections between the workstation and peripherals must be made using shielded cables in order to maintain compliance with FCC radio frequency emission limits. Networking connections can be made using unshielded twisted pair (UTP) cables.

Modifications: Any modifications made to this device that are not approved by Sun Microsystems, Inc. may void the authority granted to the user by the FCC to operate this equipment.

ICES-003 Class B Notice - Avis NMB-003, Classe B

This Class B digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe B est conforme à la norme NMB-003 du Canada.

VCCI 基準について

クラス A VCCI 基準について

クラス A VCCI の表示があるワークステーションおよびオプション製品は、クラス A 情報技術装置です。これらの製品には、下記の項目が該当します。

この装置は、情報処理装置等電波障害自主規制協議会 (VCCI) の基準に基づくクラス A 情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。

クラス B VCCI 基準について

クラス B VCCI の表示  があるワークステーションおよびオプション製品は、クラス B 情報技術装置です。これらの製品には、下記の項目が該当します。

この装置は、情報処理装置等電波障害自主規制協議会 (VCCI) の基準に基づくクラス B 情報技術装置です。この装置は、家庭環境で使用することを目的としていますが、この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。取扱説明書に従って正しい取り扱いをしてください。

BSMI Class A Notice

The following statement is applicable to products shipped to Taiwan and marked as Class A on the product compliance label.

警告使用者：
這是甲類的資訊產品，在居住的環境中使用
時，可能會造成射頻 干擾，在這種情況下，
使用者 會被要求採取某些適當的對策。

Using This Documentation

The *Sun Crypto Accelerator 6000 Board User's Guide for Version 1.0* lists the features, protocols, and interfaces of the Sun Crypto Accelerator 6000 Board from Oracle and describes how to install, configure, and manage the board in your system.

This user's guide assumes that you are a network administrator with experience configuring one or more of the following:

- Oracle Solaris operating system (OS)
- Sun platforms with PCI I/O cards
- Sun Java Web System Servers and Apache Web Servers
- IPsec
- SunVTS software
- Certification authority acquisitions

Note – In this document these x86 related terms mean the following:
– "x86" refers to the larger family of 64-bit and 32-bit x86 compatible products.
– "x64" points out specific 64-bit information about AMD64 or EM64T systems.
– "32-bit x86" points out specific 32-bit information about x86 base systems. For supported systems, see "Hardware and Software Requirements" on page 10.

Documentation, Support, and Training

Sun Function	URL
Documentation	http://docs.sun.com
Support	http://www.sun.com/support/
Training	http://www.sun.com/training/

Related Documentation

The documents listed as online are available at:

http://docs.sun.com/app/docs/prod/*filename*#hic

Application	Title	Part Number	Format	Location
Latest information	*ABC Release Notes*	801-1234-*xx*	Printed PDF	Shipping kit Online
Service	*ABC System Service Manual*	801-1234-*xx*	PDF HTML	Documentation CD and online Online

The following table lists the documentation that is related to this product.

Application	Title	Part Number	Format	Location
Latest information	*ABC Release Notes*	801-1234-*xx*	Printed PDF	Shipping kit Online at: http://docs.sun.com/app/docs/prod/*filename*
Service	*ABC System Service Manual*	801-1234-*xx*	PDF HTML	Documentation CD and online Online at: http://docs.sun.com/app/docs/prod/*filename*

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in this document, go to:

http://www.sun.com/service/contacting

Documentation Feedback

Submit comments about this document by clicking the Feedback[+] link at http://docs.sun.com.

Include the title and part number of your document with your feedback:

Sun Crypto Accelerator 6000 Board User's Guide for Version 1.0, part number 819-5536-12.

CHAPTER 1

Product Overview

This chapter provides an overview of the Sun Crypto Accelerator 6000 Board. Sections include:

Product Features

The Sun Crypto Accelerator 6000 Board is an 8 lane PCI Express based host bus adapter (HBA) that combines IPsec and SSL cryptographic acceleration with Hardware Security Module (HSM) features. The board provides improved performance, additional security features, and support for new Oracle Solaris OS on SPARC and x86 and x86 AMD Opteron platforms running Linux. The combination of a dedicated HSM, advanced cryptographic security, and secure key management specifically meets the security and performance needs for financial services.

Once installed, the board is initialized and configured with the `scamgr` utility, which manages the keystore and user information and determines the level of security in which the board operates. Once a keystore and security officer account are configured, Java and PKCS#11 applications such as Sun Java System Application Server software, and OpenSSL applications such as Apache can be configured to use the board for cryptographic acceleration.

Key Features

- Low profile, half length PCI-express, 8 lane (bi-8)
- Support for Oracle Solaris Cryptographic Framework
- Accelerates IPsec and SSL cryptographic functions
- Session establishment rate – up to 13,000 operations per second
- Bulk encryption rate – up to 1 Gbps
- Provides up to 2048-bit RSA encryption
- Provides tamper-resistant, centralized security key and certificate administration for Sun Java System Web Server for increased security and simplified key management
- FIPS 140-2 Level 3 certification
- Low CPU utilization – frees up server system resource and bandwidth
- Secure private key storage and management
- Dynamic reconfiguration (DR), and redundancy and failover support on Sun's midframe and high-end servers
- Support for Oracle Solaris 10 and future compatible releases
  - Support for Oracle Solaris Cryptographic Framework
- Support for Linux Red Hat EL 4.0, SuSE Enterprise 9.0
  - Support for openCryoptoki software
- Support for Sun's Predictive Self-Healing
- Support for the Service Management Facility (SMF), which is an improved mechanism for controlling system startup and the relationship between services.
- Multi-Admin keystore security, supporting the requirement of multiple security officers to authenticate keystore backup and restore operations.
- Serial port for direct input adminstration interface
- USB port for keystore backup/restore to USB mass storage devices

Financial Services Support

The board supports PIN and credit card related functionality, ensuring the security of sensitive customer data by performing the entire operation within the secure cryptographic boundary of the board. Specialized key management capabilities and a new user library (`libfinsvcs.so`) and associated application interface are provided to support this feature. See Chapter 4 for details.

Supported Applications

- Java application
- Sun Java System servers
- Apache Web Server
- PKCS11 applications

Supported Cryptographic Protocols and Algorithms

The board supports the following protocols:

- SSLv2
- SSLv3
- TLSv1

The board accelerates the following IPsec encapsulating security payload (ESP) algorithms.

TABLE 1-1 IPsec ESP Cryptographic Algorithms

Type	Algorithm
Symmetric	DES, 3DES, AES (encryption and decryption)

The board accelerates the following SSL algorithms.

TABLE 1-2 SSL ESP Cryptographic Algorithms

Type	Algorithm
Symmetric	DES, 3DES, AES (encryption and decryption)
Asymmetric	Diffie-Hellman and RSA (up to 2048 bit key), DSA

The board accelerates the following SSL functions:

- Secure establishment of a set of cryptographic parameters and secret keys between a client and a server.
- Secure key storage on the board – keys are encrypted if they leave the board.

Diagnostic Support

- SunVTS diagnostic tests
- Security officer initiated diagnostics (`scadiag` and `scamgr`)

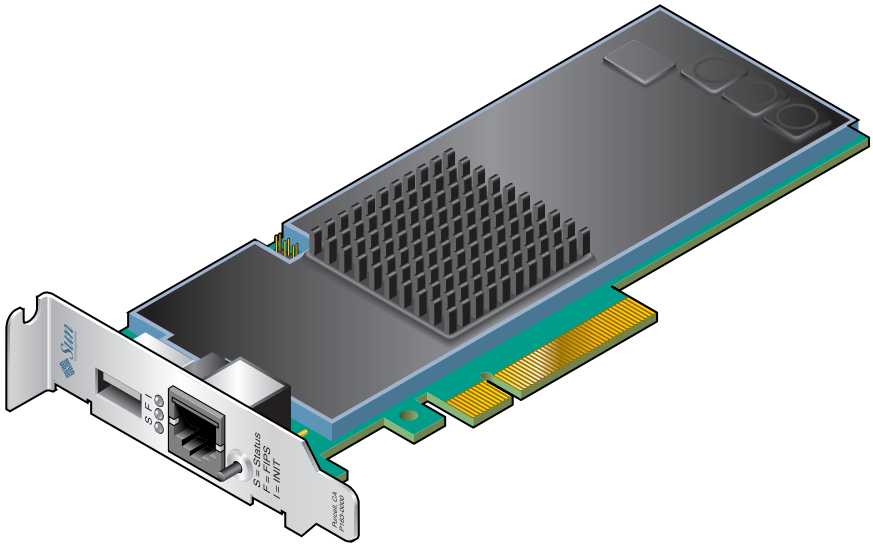
Cryptographic Algorithm Acceleration

Together with the Oracle Solaris Cryptographic Framework, the board accelerates cryptographic algorithms in both hardware and software. The reason for this complexity is that the cost of accelerating cryptographic algorithms is not uniform across all algorithms. Some cryptographic algorithms were designed specifically to be implemented in hardware, others were designed to be implemented in software. For hardware acceleration, there is the additional cost of moving data from the user application to the hardware acceleration device, and moving the results back to the user application. Note that a few cryptographic algorithms can be performed by highly tuned software as quickly as they can be performed in dedicated hardware.

Hardware Overview

The Sun Crypto Accelerator 6000 hardware is a low profile, half length (6.6 inches [1.67.64 mm] by 2.54 inches [64.41 mm]) 8 lane PCI-Express based HBA that enhances the performance of IPsec and SSL and provides robust security features. FIGURE 1-1 provides an illustration of the board.

FIGURE 1-1 Sun Crypto Accelerator 6000 Board



LED Displays

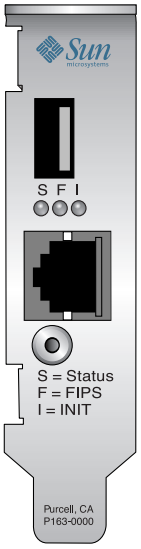
TABLE 1-3 describes the LED displays.

TABLE 1-3 Front Panel LEDs

Label	Color	Indication
STATUS	Green/Red	• Off when bootstrap firmware executes • Green in POST, and DISABLED states (driver not attached) • Flashing green in IDLE, OPERATIONAL, and FAILSAFE states (heart beat) • Red when board is in the HALTED (fatal error) state or when a low-level hardware initialization failure occurs • Flashing red if an error occurrs during the boot process
FIPS	Green/Yellow	• Off in non-FIPS mode • Green when operating in FIPS mode • Flashing yellow when zeroize jumper is present
INIT	Green/Yellow	• Off if the board has not been initialized • Green if the card has been initialized by a security officer • Yellow in POST, DIAGNOSTICS, and FAILSAFE (firmware not upgraded) states • Flashing yellow when running DIAGNOSTICS

FIGURE 1-2 shows the location of the LEDs.

FIGURE 1-2 LED Locations



Direct Input Devices

The Sun Crypto Accelerator 6000 Board has three direct input devices: an RJ-11 serial port, a USB port, and a point of presence button.

Serial Port

The six wire RJ-11 port connector enables direct input adminstration. The port operates at a baud rate of 9600-8N1. The pinout specifications are described in TABLE 1-4 and shown in FIGURE 1-3.

TABLE 1-4 RJ-11 Port Connector Pins and Signals

Pin	Signal	Definition
1	PWR	5 volt DC power
2	NC	Not connected
3	NC	Not connected

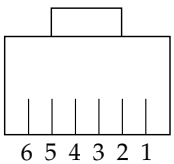
TABLE 1-4 RJ-11 Port Connector Pins and Signals *(Continued)*

Pin	Signal	Definition
4	XMIT	Transmit data
5	RECV	Data receive
6	GND	Signal ground

Suggested Serial Device

Any device with a properly configured serial port and cable can be used for direct input administration the device. However, for maximum security it is suggested that a stateless hand-held device be used to ensure sensitive information and keying material are not compromised. One such device tested and suggested is the Termiflex OT/30 Hand-held Terminal from Warner Power. A Termiflex OT/30 Terminal has been configured specifically for use with the board and can be ordered directly from Warner Power using part number 99-3619-04001 (http://ec.transcendusa.com/)

FIGURE 1-3 RJ-11 Port Connector Pins



USB Port

The standard size USB connector enables you to backup and restore the onboard keystore. The port is USB 1.1 compliant and is compatible with standard USB mass storage devices (bulk-only).

Suggested USB Device

Although other USB mass-storage devices work, only the JetFlash 2.0 USB Flash drive from Transcend has been fully tested and qualified for use with the board. Before using another device for backup and restore operations, verify that

diagnostics run successfully with the USB device installed. Choose devices with high transfer speeds and quick response times for the best compatibility with the board (http://www.termiflex.com/).

Point Of Presence Button

The point of presence button provides physical presence verification when pressed. The physical pressing of this button cannot be emulated remotely.

Dynamic Reconfiguration and High Availability

The Sun Crypto Accelerator 6000 hardware and associated software provide the capability to work effectively on SPARC platforms supporting dynamic reconfiguration (DR) and hot-plugging. During a DR or hot-plug operation, the Sun Crypto Accelerator 6000 software layer automatically detects the addition or removal of a board, and adjusts the scheduling algorithms to accommodate the change in hardware resources.

Note – DR is supported on SPARC platforms only.

For High Availability (HA) configurations, multiple Sun Crypto Accelerator 6000 boards can be installed within a system or domain to insure that hardware acceleration is continuously available. In the unlikely event of a Sun Crypto Accelerator 6000 hardware failure, the software layer detects the failure and removes the failed board from the list of available hardware cryptographic accelerators. Sun Crypto Accelerator 6000 software adjusts the scheduling algorithms to accommodate the reduction in hardware resources. Subsequent cryptographic requests are scheduled to the remaining boards.

Note that the Sun Crypto Accelerator 6000 hardware provides a source for high-quality entropy for the generation of long-term keys. If all the Sun Crypto Accelerator 6000 boards within a domain or system are removed, long-term keys are generated with lower-quality entropy.

Load Sharing

The Sun Crypto Accelerator 6000 software enables the distribution of load across as many boards as are installed within the Oracle Solaris domain or system. Incoming cryptographic requests are distributed across the boards based on fixed-length work queues. Cryptographic requests are directed to the first board, and subsequent requests stay directed to the first board until it is running at full capacity. Once the

first board is running at full capacity, further requests are queued to the next board available that can accept the request of this type. The queueing mechanism optimizes throughput by facilitating request coalescing at the board.

Hardware and Software Requirements

TABLE 1-5 provides a summary of the hardware and software requirements for the Sun Crypto Accelerator 6000 Board.

TABLE 1-5 Hardware and Software Requirements

Hardware and Software	Requirements
Hardware	• Sun Fire T1000, T2000, x2100 • Sun Ultra 40, 20
Operating System	Oracle Solaris 10, Red Hat EL 4.0*, SuSE Enterprise 9.0*, and future compatible releases of these operating systems.

***Note –** 1 Gbyte of memory is suggested for Linux operating systems.

Oracle Solaris 10 on SPARC and x86 Platforms

The board supports the Oracle Solaris 10 operating system on both SPARC and x86 AMD Opteron Linux platforms. The board acts as a cryptographic service provider to the Oracle Solaris Cryptographic Framework, allowing applications to access the board's functionality with PKCS11, OpenSSL, and JAVA (J2SE).

Linux on x86 AMD Opteron Linux Platforms

The openCryptoki software interface is used based on support by both the Red Hat EL 4.0 and SuSE Enterprise 9.0 Linux distributions. openCryptoki provides a user level interface that allows selecting specific cryptographic providers.

Note – IPsec cryptographic hardware acceleration is not supported on the current Linux distributions.

Required Patches

Refer to the *Sun Crypto Accelerator 6000 Board Product Notes for Version 1.0* for detailed required patch information.

CHAPTER 2

Installing the Sun Crypto Accelerator 6000 Board

This chapter describes how to install the Sun Crypto Accelerator 6000 Board hardware in both Oracle Solaris OS and Linux environments, and also how to install and remove the software. Sections include:

Once you have installed the hardware and software of the board, you need to initialize the board with configuration and keystore information. See "Initializing the Board With `scamgr`" on page 39 for information on how to initialize the board.

Handling the Board

Each board is packed in a special antistatic bag to protect it during shipping and storage. To avoid damaging the static-sensitive components on the board, reduce any static electricity on your body before touching the board by using one of the following methods:

- Touch the metal frame of the computer.
- Attach an antistatic wrist strap to your wrist and to a grounded metal surface.

Caution – To avoid damaging the sensitive components on the board, wear an antistatic wrist strap when handling the board, hold the board by its edges only, and always place the board on an antistatic surface (such as the plastic bag it came in).

Installing the Board on Oracle Solaris Platforms

Installing the board involves inserting the board into the system and loading the software tools. The hardware installation instructions include only general steps for installing the board. Refer to the documentation that came with your system for specific installation instructions.

▼ Install the Hardware

1. **As superuser, follow the instructions that came with your system to shut down and power off the computer, disconnect the power cord, and remove the computer cover.**
2. **Locate an unused PCI slot (preferably an x8 PCI-Express slot).**
3. **Attach an antistatic wrist strap to your wrist, and attach the other end to a grounded metal surface.**
4. **Using a Phillips screwdriver, remove the screw from the PCI slot cover.**

   Save the screw to hold the bracket in Step 5.

5. **Holding the board by its edges only, take it out of the plastic bag and insert it into the PCI slot.**

6. **Secure the screw on the rear bracket.**

7. **Replace the computer cover, reconnect the power cord, and power on the system.**

8. **For Oracle Solaris SPARC platforms, verify that the board is properly installed by entering the** `prtdiag` **command from a terminal:**

```
% prtdiag
========================= IO Configuration =========================

           IO
Location   Type Slot Path                                 Name           Model
---------- ---- ---- ---------------------------------- ------------- ---------
IOBD/NET0  PCIE IOBD /pci@780/pci@0/pci@1/network@0      network-pciex8086,105e
IOBD/NET1  PCIE IOBD /pci@780/pci@0/pci@1/network@0,1    network-pciex8086,105e
IOBD/PCIE0 PCIE 0    /pci@780/pci@0/pci@8/pci@0/pci108e,5ca0@e  pci108e,5ca0
IOBD/PCIX  PCIX IOBD /pci@7c0/pci@0/pci@1/pci@0/isa@2    isa
IOBD/PCIX  PCIX IOBD /pci@7c0/pci@0/pci@1/pci@0/usb@5    usb-pciclass,0c0310
IOBD/PCIX  PCIX IOBD /pci@7c0/pci@0/pci@1/pci@0/usb@6    usb-pciclass,0c0310
IOBD/PCIX  PCIX IOBD /pci@7c0/pci@0/pci@1/pci@0/ide@8    ide-pci10b9,5229
IOBD/PCIX  PCIX PCIX /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2      LSILogic,sas-
pci1000,50 LSI,1064
IOBD/NET2  PCIE IOBD /pci@7c0/pci@0/pci@2/network@0      network-pciex8086,105e
IOBD/NET3  PCIE IOBD /pci@7c0/pci@0/pci@2/network@0,1    network-pciex8086,105e
```

In the preceding example, the `/pci@780/pci@0/pci@8/pci@0/pci108e,5ca0@e` identifies the device path to the board. There is one such line for each board in the system.

9. **For Oracle Solaris x86 platforms, verify that the board is properly installed by entering the** `scanpci` **command from a terminal:**

```
# /usr/X11/bin/scanpci
...
pci bus 0x0082 cardnum 0x0e function 0x00: vendor 0x108e device
0x5ca0
  Sun Microsystems Computer Corp.  Device unknown
```

Installing the Sun Crypto Accelerator 6000 Board Software With the `install` Script

There are two methods to install the software, manually or with the `install` script. This section describes how to install the software with the `install` script. To install the software manually, refer to "Installing the Software on Oracle Solaris Platforms Without the Installation Script" on page 21.

The `install` script identifies which platform you are installing on (Oracle Solaris SPARC or x86, Linux x86 or x64) and calls the appropriate installation scripts for your platform. The `install` script also automatically installs the required patches before installing the software.

In addition to the software provided on the product CD, required software is provided at the Sun Download Center (`http://www.sun.com/download/`).

The `install` script path is as follows:

`/cdrom/cdrom0/Sun_Crypto_Acc_6000`

▼ Install the Software With the `install` Script

1. **Insert the Sun Crypto Accelerator 6000 CD into a CD-ROM drive that is connected to your system.**

- If your system is running Sun Enterprise Volume Manager, the system should automatically mount the CD-ROM to the `/cdrom/cdrom0` directory.
- If your system is not running Sun Enterprise Volume Manager, mount the CD-ROM as follows:

```
# mount -F hsfs -o ro /dev/dsk/c0t6d0s2 /cdrom
```

You see the following files and directories in the `/cdrom/cdrom0/Sun_Crypto_Acc_6000` directory:

TABLE 2-1 Files in the `/cdrom/cdrom0/Sun_Crypto_Acc_6000` Directory

File or Directory	Contents
`README`	
`Copyright`	U.S. copyright file
`FR_Copyright`	French copyright file
`install`	Script that installs the Sun Crypto Accelerator 6000 packages for both Oracle Solaris SPARC and x86 systems, and for Linux x86 or x64 systems
`Solaris/sparc`	Contains the Oracle Solaris SPARC software packages: • `SUNWmcact` – Activation file • `SUNWmcadevfw` – Development firmware • `SUNWmcaf` – FMA support • `SUNWmcafw` – Firmware • `SUNWmcamn` – Manual pages • `SUNWmcar` – Drivers • `SUNWmcau` – User components • `SUNWscafsu` – Financial services (`usr`) • `SUNWscafsm` – Financial services manual pages • `SUNWscamga` – Administration client • `SUNWscamgm` – Administration manual pages • `SUNWscamgr` – Administration (`root`) • `SUNWscamgu` – Administration (`usr`)
`Solaris/i386/`	Contains the Oracle Solaris i386 software packages: • `SUNWmcact` – Activation file • `SUNWmcaf` – FMA support • `SUNWmcafw` – Firmware • `SUNWmcamn` – Manual pages • `SUNWmcar` – Drivers • `SUNWmcau` – User components • `SUNWscafsu` – Financial services (`usr`) • `SUNWscafsm` – Financial services manual pages • `SUNWscamga` – Administration client • `SUNWscamgm` – Administration manual pages • `SUNWscamgr` – Administration (`root`) • `SUNWscamgu` – Administration (`usr`)
`Solaris/install`	Script that installs the software packages for both Oracle Solaris SPARC and x86 systems. This script is normally called by the main `install` script.
`Solaris/remove`	Script that removes the software packages for Oracle Solaris SPARC and x86 systems.

TABLE 2-1 Files in the `/cdrom/cdrom0/Sun_Crypto_Acc_6000` Directory *(Continued)*

File or Directory	Contents
`Linux/`*supported-kernel*	Contains the Linux x86 or x64 software `rpm` packages: • `sun-sca6000` – software and drivers • `sun-sca6000-admin` – administration utilities • `sun-sca6000-config` – configuration files for administration and keystore I/O services • `sun-sca6000-man` – user documentation • `sun-sca6000-var` – variable length files
`Linux/install`	Script that installs the Sun Crypto Accelerator 6000 packages for Linux systems. This script is normally called by the main `install` script.
`Linux/remove`	Script that removes the Sun Crypto Accelerator 6000 packages for Linux x86 Systems.
`docs`	Contains the PDF pointer document that links to the required software and the latest user's guide (this document) and product notes (819-5537).

2. Install the required software by typing:

```
# cd /cdrom/cdrom0/Sun_Crypto_Acc_6000
# ./install
```

The `install` script analyzes the system to identify the system architecture and the required patches. The `install` script then installs those patches, and installs the main software appropriate for your system. The following is an example of running the `install` script on an Oracle Solaris SPARC system.

Note – The copyright and license information is omitted from the following example. Refer to Appendix C for copyright and software licenses.

```
# ./install
This program installs the software for the Sun Crypto Accelerator
6000


This script is about to take the following actions:
- Install Sun Crypto Accelerator 6000 support for Solaris 10

To cancel installation of this software, press 'q' followed by a Return.
         **OR**
Press Return key to begin installation:

*** Installing Sun Crypto Accelerator 6000 software for Solaris 10...
Installing required packages:
```

```
SUNWmcaf SUNWmcact SUNWmcafw SUNWmcamn SUNWmcar SUNWmcau SUNWscafsu SUNWscamga
SUNWscamgm SUNWscamgr SUNWscamgu


Installation of <SUNWmcaf> was successful.

Installation of <SUNWmcact> was successful.

Installation of <SUNWmcafw> was successful.

Installation of <SUNWmcamn> was successful.

Installation of <SUNWmcar> was successful.

Installation of <SUNWmcau> was successful.

Installation of <SUNWscafsu> was successful.

Installation of <SUNWscafsm> was successful.

Installation of <SUNWscamga> was successful.

Installation of <SUNWscamgm> was successful.

Installation of <SUNWscamgr> was successful.

Importing Keystore I/O Service to SMF
Starting Keystore I/O Service
Importing Administration Service to SMF
Starting Administration Service

Installation of <SUNWscamgu> was successful.

*** Installation complete.

To remove this software, use the 'remove' script on this CDROM, or
the following script:

        /var/tmp/crypto_acc.remove

A log of this installation can be found at:
        /var/tmp/crypto_acc.install.2006.02.28.0833
```

Directories and Files for Oracle Solaris Platforms

TABLE 2-2 shows the directories created by the default installation of the Sun Crypto Accelerator 6000 software.

TABLE 2-2 Sun Crypto Accelerator 6000 Board Directories and Files for Solaris Platforms

Directory	Contents
`/kernel/drv`	Driver configuration files
`/kernel/drv/sparcv9`	64-bit SPARC drivers
`/kernel/drv/amd64`	64-bit AMD drivers
`/opt/SUNWsca/include`	Financial services header files
`/opt/SUNWsca/lib`	Financial services libraries
`/opt/SUNWsca/lib/sparcv9`	Financial services libraries
`/opt/SUNWsca/lib/amd64`	Financial services libraries
`/opt/SUNWsca/man`	Financial services man pages
`/usr/lib/crypto`	Services
`/usr/lib/crypto/firmware/sca`	Firmware files
`/usr/man`	Man pages
`/usr/sbin`	Administration utilities
`/var/sca/keydata`	Keystore files (encrypted)
`/var/sca/log`	Service log files
`/var/svc/manifest/device`	Service manifests

Note – Once you install the Sun Crypto Accelerator 6000 hardware and software, you need to initialize the board with configuration and keystore information. See "Initializing the Board With `scamgr`" on page 39 for information on how to initialize the board.

Removing the Sun Crypto Accelerator 6000 Software on Oracle Solaris Platforms With the `remove` Script

There are two methods to remove the software, the `remove` script on the CD-ROM, or the `pkgrm` command. Use the `remove` script described in this section if you used the `install` script to install the software. If you installed the software without the `install` script, see "Removing the Software on Oracle Solaris Platforms Without the remove Script" on page 23.

▼ Remove the Software With the `remove` Script on the CD-ROM

- **Type the following with the Sun Crypto Accelerator 6000 CD-ROM inserted:**

```
# cd /cdrom/cdrom0/Sun_Crypto_Acc_6000/
# ./Solaris/remove
```

Installing the Software on Oracle Solaris Platforms Without the Installation Script

This section describes how to install the software manually without using the installation script provided on the product CD.

Refer to the latest version of the *Sun Crypto Accelerator 6000 Board Product Notes for Version 1.0* (819-5537) for a list of the required patches. You must install all of the required patches before installing the main software. The latest product notes are available at: `http://docs.sun.com`

Note – The `/cdrom/cdrom0/Sun_Crypto_Acc_6000/install` script automatically identifies your system architecture, installs the required patches, and installs the main software appropriate for your system.

In addition to the software provided on the product CD, required software is provided at the Sun Download Center (http://www.sun.com/download/).

▼ Install the Software Without the `install` Script

1. **Insert the Sun Crypto Accelerator 6000 CD into a CD-ROM drive that is connected to your system.**

- If your system is running Sun Enterprise Volume Manager, the system should automatically mount the CD-ROM to the `/cdrom/cdrom0` directory.
- If your system is not running Sun Enterprise Volume Manager, mount the CD-ROM as follows:

```
# mount -F hsfs -o ro /dev/dsk/c0t6d0s2 /cdrom
```

The required packages must be installed in a specific order and must be installed before installing any optional packages. Once the required packages are installed, you can install and remove the optional packages in any order.

2. **Install the required software packages by typing:**

```
# cd /cdrom/cdrom0/Sun_Crypto_Acc_6000/Packages
# pkgadd -d . SUNWmcaf SUNWmcact SUNWmcafw SUNWmcamn SUNWmcar
SUNWmcau SUNWscafsu SUNWscafsm SUNWscamga SUNWscamgm SUNWscamgr
SUNWscamgu
```

3. **(Optional) To verify that the software is installed properly, run the `pkginfo` command.**

```
# pkginfo SUNWmcaf SUNWmcact SUNWmcafw SUNWmcamn SUNWmcar SUNWmcau SUNWscafsu
SUNWscafsm SUNWscamga SUNWscamgm SUNWscamgr SUNWscamgu
system      SUNWmcact  Sun Crypto Accelerator 6000 Activation File
system      SUNWmcaf   Sun Crypto Accelerator 6000 FMA Support
system      SUNWmcafw  Sun Crypto Accelerator 6000 Firmware
system      SUNWmcamn  Sun Crypto Accelerator 6000 Manual Pages
system      SUNWmcar   Sun Crypto Accelerator 6000 Drivers
system      SUNWmcau   Sun Crypto Accelerator 6000 User Components
system      SUNWscafsu Sun Crypto Accelerator Financial Services
system      SUNWscafsm Sun Crypto Accelerator Financial Services Man Pages
system      SUNWscamga Sun Crypto Accelerator Administration Client
system      SUNWscamgm Sun Crypto Accelerator Administration Man Pages
system      SUNWscamgr Sun Crypto Accelerator Administration (root)
system      SUNWscamgu Sun Crypto Accelerator Administration (usr)
```

4. **(Optional for Oracle Solaris SPARC platforms) To ensure that the driver is attached, use the `prtdiag` command.**

```
# prtdiag -v
```

Refer to the `prtdiag`(1m) online manual pages.

5. **(Optional for Oracle Solaris x86 platforms) To ensure that the driver is attached, use the `scanpci` command.**

```
# /usr/X11/bin/scanpci
...
pci bus 0x0082 cardnum 0x0e function 0x00: vendor 0x108e device
0x5ca0
  Sun Microsystems Computer Corp.  Device unknown
```

6. **(Optional) Use the `modinfo` command to see that modules are loaded.**

```
# modinfo | grep Crypto
62   1317f62  20b1f 198   1  mca (MCA Crypto 1.0)
197  136d5d6   19b0 199   1  mcactl (MCA Crypto Control 1.0)
```

See "Directories and Files for Oracle Solaris Platforms" on page 20 for a description of the directories and files in the default installation.

Removing the Software on Oracle Solaris Platforms Without the `remove` Script

Note – Remove the Sun Crypto Accelerator 6000 Board software manually only if you did *not* use the `install` script to install the software. If you installed the software with the installation script, to remove the software, see "Removing the Sun Crypto Accelerator 6000 Software on Oracle Solaris Platforms With the remove Script" on page 21.

If you have created keystores (see "Managing Keystores With `scamgr`" on page 43), you must delete the keystore information that the Sun Crypto Accelerator 6000 Board is configured with before removing the software. The `zeroize` command removes all key material, but does not delete the keystore files that are stored in the filesystem of the physical host in which the board is installed. See the "Perform a Software Zeroize on the Board" on page 62 for details on the `zeroize` command. If you have not yet created any keystores, you can skip this procedure.

▼ Delete Existing Keystores

1. **Become superuser.**
2. **Remove the keystore files with the `rm` command.**

Caution – Do not delete a keystore that is currently in use or that is shared by other users and keystores. To free references to keystores, you might have to shut down the web server, administration server, or both

▼ Remove the Software Without the `remove` Script

Caution – Before removing the Sun Crypto Accelerator 6000 Board software disable any web servers you have enabled for use with the board. Failure to do so leaves those web servers nonfunctional.

- **As superuser, use the `pkgrm` command to remove only the software packages you installed.**

Caution – Installed packages must be removed in the order shown. Failure to remove them in this order could result in dependency warnings and leave kernel modules loaded.

If you installed all the packages, you would remove them as follows:

```
# pkgrm SUNWscamgu SUNWscamgr SUNWscamgm SUNWscamga SUNWscafsm
SUNWscafsu SUNWmcau SUNWmcar SUNWmcamn SUNWmcafw SUNWmcact
SUNWmcaf
```

Installing the Sun Crypto Accelerator 6000 Board on Linux Platforms

openCryptoki software is required for the board on Linux platforms. It must be installed before installing the software. Refer to Appendix B to install the openCryptoki software.

Installing the hardware on Linux platforms is the same as with Oracle Solaris platforms. After the system is up, type the following command to verify the board is installed properly:

```
% lspci
```

The output of the previous command should contain the following line:

```
Network and computing encryption device: Sun Microsystems Computer
Corp.: Unknown device 5ca0
```

Installing the Sun Crypto Accelerator 6000 Software on Linux Platforms With the `install` Script

Type the following command:

```
% ./install
Do you agree to the above license terms? [ACCEPT or DECLINE]
ACCEPT
Installing required packages:
     sun-sca6000-admin-1.0-1.x86_64.rpm
     sun-sca6000-var-1.0-1.x86_64.rpm
     sun-sca6000-config-1.0-1.x86_64.rpm
     sun-sca6000-1.0-1.x86_64.rpm
     sun-sca6000-man-1.0-1.x86_64.rpm
     sun-sca6000-firmware-1.0-1.x86_64.rpm
```

Installing the Sun Crypto Accelerator 6000 Software on Linux Platforms Without the `install` Script

The packages for SuSE Linux Enterprise Server 9 Service Pack 3 are in the `2.6.5-7.244-smp-x86_64` directory. The packages for Red Hat Enterprise Linux 4.0 Update 2 are in the 2.6.9-22.ELsmp-x86_64 directory. The packages are as follows:

```
sun-sca6000-1.0-1.x86_64.rpm
sun-sca6000-man-1.0-1.x86_64.rpm
sun-sca6000-admin-1.0-1.x86_64.rpm
sun-sca6000-var-1.0-1.x86_64.rpm
sun-sca6000-config-1.0-1.x86_64.rpm
sun-sca6000-firmware-1.0-1.x86_64.rpm
```

▼ Install the Software Without the `install` Script

1. **Change to the appropriate directory for your platform and enter the following command:**

```
% rpm -i sun-sca6000-man-1.0-1.x86_64.rpm sun-sca6000-admin-1.0-
1.x86_64.rpm sun-sca6000-var-1.0-1.x86_64.rpm sun-sca6000-config-
1.0-1.x86_64.rpm sun-sca6000-1.0-1.x86_64.rpm sun-sca6000-
firmware-1.0-1.x86_64.rpm
```

2. **(Optional) To ensure that the driver is attached, use the `scanpci` command.**

```
# /usr/X11R6/bin/scanpci
...
pci bus 0x0082 cardnum 0x0e function 0x00: vendor 0x108e device
0x5ca0
  Sun Microsystems Computer Corp.  Device unknown
```

Directories and Files for Linux Platforms

TABLE 2-3 shows the directories created by the default installation of the Sun Crypto Accelerator 6000 software.

TABLE 2-3 Sun Crypto Accelerator 6000 Board Directories and Files

Directory	Contents
`/etc/init.d`	Start/stop scripts (links)
`/etc/rc5.d`	Service config files
`/etc/opt/sun/sca6000`	Daemon configuration files
`/opt/sun/sca6000/bin`	Application executables, drivers, and the `scamgr` utility
`/opt/sun/sca6000/bin/drv`	Driver files
`/opt/sun/sca6000/firmware`	Firmware files
`/opt/sun/sca6000/lib`	OpenCryptoki plug-ins and application libraries
`/opt/sun/sca6000/man`	Man pages
`/opt/sun/sca6000/sbin`	Administration utilities and services and daemon executables
`/usr/local/lib/opencryptoki/stdll/`	openCryptoki plugin files
`/var/opt/sca6000/keydata`	Keystore files (encrypted)
`/var/opt/sca6000/lock`	Service lock files
`/var/opt/sca6000/log`	Service log files

Note – Once you install the Sun Crypto Accelerator 6000 hardware and software, you need to initialize the board with configuration and keystore information. See "Initializing the Board With `scamgr`" on page 39 for information on how to initialize the board.

Removing the Sun Crypto Accelerator 6000 Software on Linux Platforms Without the `remove` Script

Any applications, such as SunJava System and Apache webservers, that are using the board must be stopped before uninstalling the Sun Crypto Accelerator 6000 software.

▼ Remove the Software

Change to the appropriate directory for your platform and enter the following command:

```
% ./remove
```

Alternatively, you can enter the following command on one line:

```
% rpm -e sun-sca6000-1.0-1.x86_64.rpm sun-sca6000-man-1.0-
1.x86_64.rpm sun-sca6000-admin-1.0-1.x86_64.rpm sun-sca6000-var-
1.0-1.x86_64.rpm sun-sca6000-config-1.0-1.x86_64.rpm sun-sca6000-
firmware-1.0-1.x86_64.rpm
```

CHAPTER 3

Administering the Sun Crypto Accelerator 6000 Board

This chapter provides an overview of administering the board on both Oracle Solaris and Linux platforms with the `scamgr` and `scadiag` utilities, and the `scad` and `scakiod` service daemons. Additional instructions for Linux platforms are included in the last two sections. Sections include:

Using the scamgr Utility

The scamgr utility offers a command-line interface to the board. Only users designated as security officers are permitted to use the scamgr utility. When you first connect to a board with scamgr, you are prompted to create an initial security officer and password.

The scamgr command-line syntax is:

- scamgr [-?]
- scamgr [-H]
- scamgr [-V]
- scamgr [-y] [-h *hostname*] [-p *port*] [-d *device*] [-f *filename*]
- scamgr [-y] [-h *hostname*] [-p *port*] [-d *device*] *command*

Note – When using the -d attribute, mca*N* is the board's device name, where the *N* corresponds to the Sun Crypto Accelerator 6000 device instance number.

Note – Certain shells interpret the ? character when using -? option on the command line. To avoid this, use the escape character (\) directly in front of the ?. For example in the C shell, the command is changed to scamgr -\?.

TABLE 3-1 shows the options for the scamgr utility.

TABLE 3-1 scamgr Options

Option	Meaning
-?	Displays help files for scamgr commands and exits.
-H	Displays help files for scamgr commands and exits.
-d mca*N*	Connects to the board that has *N* as the driver instance number. For example, -d mca1 connects to device mca1, where mca is a string in the board's device name and 1 is the instance number of the device. This value defaults to mca0 and must be in the form of mca*N*, where *N* corresponds to the device instance number.
-f *filename*	Interprets one or more commands from *filename* and exits.
-h *hostname*	Connects to the board on *hostname*. The value for *hostname* can be a host name or an IP address, and defaults to the loopback address.

TABLE 3-1 `scamgr` Options

Option	Meaning
`-p` *port*	Connects to the board on *port*. The value for *port* defaults to 6870.
`-V`	Displays version information for `scamgr`.
`-y`	Forces a yes answer to any command that would normally prompt for a confirmation.

Note – The variable *sec-officer* is used throughout this document as an example security officer name.

Modes of Operation

`scamgr` can run in one of three modes. These modes differ mainly in how commands are passed into `scamgr`. The three modes are Single-Command mode, File mode, and Interactive mode.

Note – To use `scamgr`, you must authenticate as security officer. How often you need to authenticate as security officer is determined by which operating mode you are using.

Single-Command Mode

In Single-Command mode, you must authenticate as security officer for every command. Once the command is executed, you are logged out of `scamgr`.

When entering commands in Single-Command mode, you specify the command to be run after all the command-line switches are specified. For example, in Single-Command mode, the following command would show all the users in a given keystore and return the user to the command shell prompt.

```
$ scamgr show user
Security Officer Name: sec-officer
Security Officer Password:
```

All output from Single-Command mode goes to the standard output stream. This output can be redirected using standard UNIX shell-based methods.

File Mode

In File mode, you must authenticate as security officer for every file you run. You are logged out of `scamgr` after the commands in the command file are executed.

To enter commands in File mode, you specify a file from which `scamgr` reads one or more commands. The file must be ASCII text, consisting of one command per line. Begin each comment with a hash (#) character. If the File mode option is set, `scamgr` ignores any command-line arguments after the last option. The following example runs the commands in the `deluser.scr` file and answers all prompts in the affirmative:

```
$ scamgr -f deluser.scr -y
```

Interactive Mode

In Interactive mode, you must authenticate as security officer every time you connect to a board. This is the default operating mode for `scamgr`. To log out of `scamgr` in Interactive mode, use the `logout` command. Refer to "Logging In and Out With `scamgr`" on page 32.

Interactive mode presents the user with an interface similar to `ftp`(1), where commands can be entered one at a time. The `-y` option is not supported in Interactive mode.

Logging In and Out With `scamgr`

When you use `scamgr` from the command line and specify *host*, *port*, and *device* using the `-h`, `-p`, and `-d` attributes respectively, you are immediately prompted to log in as security officer if a successful network connection was made.

The `scamgr` utility establishes an encrypted network connection (channel) between the `scamgr` application and the Sun Crypto Accelerator 6000 firmware running on a specific board.

During setup of the encrypted channel, boards identify themselves by their hardware serial ID address and an RSA public key. A trust database (`$HOME/.sunw/sca/trustdb`) is created the first time `scamgr` connects to a board. This file contains all of the boards that are currently trusted by the security officer.

Note – The board is preprogrammed with a unique remote access key for connecting to an uninitialized board. The fingerprint for this remote access key is printed on the board and must be verified when logging into a board for the first time to ensure a secure channel is established with the correct board.

`scamgr` Prompt

The `scamgr` prompt in Interactive mode is displayed as follows:

```
scamgr{mcaN@hostname, sec-officer}> command
```

The following table describes the `scamgr` prompt variables:

TABLE 3-2 `scamgr` Prompt Variable Definitions

Prompt Variable	Definition
`mca`*N*	`mca` is a string that represents the board. *N* is the device instance number (unit address) that is in the device path name of the board.
hostname	The name of the host for which the board is physically connected. *hostname* may be replaced with the physical host's IP address.
sec-officer	The name of the security officer that is currently logged in to the board.

Logging In to a Board With `scamgr`

If the security officer connects to a new board, `scamgr` notifies the security officer and prompts with the following options:

```
1. Abort this connection
2. Trust the board for this session only
3. Trust the board for all future sessions
```

If the security officer connects to a board that has a remote access key that has been changed, `scamgr` will notify the security officer and prompt the following three options:

```
1. Abort this connection
2. Trust the board for this session only
3. Replace the trusted key with the new key
```

Logging In to a New Board

Note – The remaining examples in this chapter were created with the Interactive mode of `scamgr`.

When connecting to a new board, `scamgr` must create a new entry in the trust database. The following is an example of logging in to a new board.

```
# scamgr -h hostname
Warning: Serial ID and Public Key Not Found
------------------------------------------------------------
The Serial ID and public key presented by this board were
not found in your trust database.

Serial ID: 36:30:30:30:30:33
Key Fingerprint: baa4-17f8-1128-1c6a-9a18-3719-988f-64a0-a4a5-
f72f
------------------------------------------------------------
Please select an action:

1. Abort this connection
2. Trust the board for this session only.
3. Trust the board for all future sessions.

Your Choice -->
```

Logging In to a Board With a Changed Remote Access Key

When connecting to a board that has a changed remote access key, `scamgr` must change the entry corresponding to the board in the trust database. The following is an example of logging in to a board with a changed remote access key.

```
# scamgr -h hostname
Warning: Public Key Conflict
------------------------------------------------------------
The public key presented by the board you are connecting
to is different than the public key that is trusted for
this Serial ID.

Serial ID: 36:30:30:30:30:33
New Key Fingerprint: baa4-17f8-1128-1c6a-9a18-3719-988f-64a0-
a4a5-f72f
Trusted Key Fingerprint: e207-6ff7-41f4-3766-bafd-5910-973d-a32b-
46e8-6e73
------------------------------------------------------------
Please select an action:

1. Abort this connection
2. Trust the board for this session only.
3. Replace the trusted key with the new key.

Your Choice -->
```

Logging Out of a Board With `scamgr`

If you are working in Interactive mode, you might want to disconnect from one board and connect to another board without completely exiting `scamgr`. To disconnect from a board and log out, but remain in Interactive mode, use the `logout` command:

```
scamgr{mcaN@hostname, sec-officer}> logout
scamgr>
```

In the previous example, notice that the `scamgr>` prompt no longer displays the device instance number, hostname, or security officer name. To log in to another device, type the `connect` command with the following optional parameters.

TABLE 3-3 `connect` Command Optional Parameters

Parameter	Meaning
`dev mca`*N*	Connect to the board with the driver instance number of *N*. For example `-d mca1` connects to the device `mca1`; this defaults to device `mca0`.
`host` *hostname*	Connect to the board on *hostname* (defaults to the loopback address). *hostname* may be replaced with the physical host's IP address.
`port` *port*	Connect to the board on port *port* (defaults to 6870).

Example:

```
scamgr{mcaN@hostname, sec-officer}> logout
scamgr> connect host hostname dev mca2
Security Officer Login: sec-officer
Security Officer Password:
scamgr{mca2@hostname, sec-officer}>
```

`scamgr` does not let you issue the `connect` command if you are already connected to a board. You must first log out and then issue the `connect` command.

Each new connection causes `scamgr` and the target board firmware to renegotiate new session keys to protect the administrative data that is sent.

Entering Commands With `scamgr`

This section lists the available `scamgr` commands and describes their usage.

scamgr Commands

TABLE 3-4 lists the scamgr commands.

TABLE 3-4 scamgr Commands

Command	Description
backup	Backs up the master key.
connect	Begins an admin session with the firmware.
create	Creates users and accounts.
delete	Deletes users and accounts.
diagnostics	Performs diagnostic tests.
disable	Disables users, modes, and options.
enable	Enables users, modes, and options.
exit	Exits the scamgr utility.
load	Loads data input items.
logout	Logs you out of the current session.
quit	Exits the scamgr utility.
rekey	Generates new system keys.
reset	Resets the hardware.
set	Sets operating parameters.
show	Shows system settings.
zeroize	Deletes all keys and resets the board.

The scamgr utility has a command language that must be used to interact with the board. Commands are entered using all or part of a command (enough to uniquely identify that command from any other command). Entering sh instead of show would work, but re is ambiguous because it could be reset or rekey.

The following example shows entering commands using entire words:

```
scamgr{mcaN@hostname, sec-officer}> show user
User                                          Status
------------------------------------------------------
web-admin                                     Enabled
Tom                                           Enabled
------------------------------------------------------
```

The same information can be obtained in the previous example using partial words as commands, such as sh us.

An ambiguous command produces an explanatory response:

```
scamgr{mcaN@hostname, sec-officer}> re
Ambiguous command: re
```

Getting Help for Commands

`scamgr` has built-in help functions. To get help, you must enter a question mark (?) character following the command you want more help on. If an entire command is entered and a "?" exists anywhere on the line, you get the syntax for the command, for example:

```
scamgr{mcaN@hostname, sec-officer}> create ?
Sub-Command                    Description
-------------------------------------------------------
so                             Create a new security officer
user                           Create a new user

scamgr{mcaN@hostname, sec-officer}> create user ?
Usage: create user [<username>]

scamgr{mcaN@hostname, sec-officer}> set ?
Sub-Command                    Description
-------------------------------------------------------
lock                           Lock master key (Prevents key backup)
multiadmin                     Configure Multi-Admin mode
passreq                        Set password security level
password                       Change password for security officer
timeout                        Set firmware auto-logout timer
```

You can also enter a question mark at the `scamgr` prompt to see a list of all of the `scamgr` commands and their description, for example:

```
scamgr{mcaN@hostname, sec-officer}> ?
Sub-Command                     Description
-----------------------------------------------------------------
backup                          Backup master key
connect                         Begin admin session with firmware
create                          Create users and accounts
delete                          Delete users and accounts
diagnostics                     Run diagnostic tests
disable                         Disable users, modes or options
enable                          Enable users, modes or options
exit                            Exit scamgr
load                            Load data items
logout                          Logout current session
quit                            Exit scamgr
rekey                           Generate new system keys
reset                           Reset the hardware
set                             Set operating parameters
show                            Show system settings
zeroize                         Delete all keys and reset board
```

Note – When not in `scamgr` Interactive mode, the "?" character could be interpreted by the shell in which you are working. In this case, ensure that you use the command shell escape character before the question mark. For example in the C shell, you would need to type: `\?`

Quitting the `scamgr` Utility

Two commands allow you to exit from `scamgr`, `quit` and `exit`. The Ctrl-D key sequence also exits from `scamgr`.

Initializing the Board With `scamgr`

The first step in configuring a board is to initialize it. When you initialize a board it is necessary to create a keystore. (See "Web Server Concepts and Terminology" on page 110.) When you first connect to a board with `scamgr`, you are prompted to

initialize the board with a new keystore or an existing keystore, which is stored in a backup file. `scamgr` prompts you for all the required information for either type of board initialization.

Initializing the Board With a New Keystore

This section describes how to initialize the board with a new keystore.

▼ Initialize the Board With a New Keystore

1. Initialize the board with the `scamgr` command.

- If the board is installed locally, enter `scamgr` at the system prompt.
- If the system is remote, enter `scamgr -h` *hostname*

2. Enter 2 then 1 as shown in the following example:

```
# scamgr -h hostname
Please select an action:

1. Abort this connection
2. Trust the board for this session only.
3. Replace the trusted key with the new key.

Your Choice --> 2
This board is uninitialized.
You will now initialize the board. You may either
completely initialize the board and start with a new
keystore or initialize the board to use an existing
keystore, providing a backup file in the process.

1. Initialize the board with a new keystore
2. Initialize the board to use an existing keystore

Your Choice (0 to exit) --> 1
```

3. Create a keystore name.

See "Naming Requirements" on page 44.

```
Keystore Name: keystore-name
```

4. **Select FIPS 140-2 mode or non-FIPS mode.**

   When in FIPS mode the board is FIPS 140-2, level 3 compliant. FIPS 140-2 is a Federal Information Processing standard that requires tamper-resistance and a high level of data integrity and security. Refer to the FIPS 140-2 document located at:
   http://www.nist.gov/dmvp

```
Run in FIPS 140-2 mode? (Y/Yes/N/No) [No]: y
```

5. **Create an initial security officer name and password.**

   See "Naming Requirements" on page 44.

```
Initial Security Officer Name: sec-officer
Initial Security Officer Password:
Confirm Password:
```

Note – Before an essential parameter is changed or deleted, or before a command is executed that might have drastic consequences, `scamgr` prompts you to enter `Y`, `Yes`, `N`, or `No` to confirm. These values are not case sensitive; the default is `No`.

---

6. **Verify the configuration information:**

```
Board initialization parameters:
-----------------------------------------------------
Initial Security Officer Name: sec-officer
Keystore name: keystore-name
Run in FIPS 140-2 Mode: Yes
-----------------------------------------------------

Is this correct? (Y/Yes/N/No) [No]: y
Initializing crypto accelerator board... This may take a few
minutes...Done.
```

Initializing the Board to Use an Existing Keystore

If you are adding multiple boards to a single keystore, you might want to initialize all of the boards to use the same keystore information. In addition, you might want to restore a board to the original keystore configuration. This section describes how to initialize a board to use an existing keystore which is stored in a backup file.

You must first create a backup file of an existing board configuration before performing this procedure. Creating and restoring a backup file requires a password to encrypt and decrypt the data in the backup file. (See "Back Up the Master Key" on page 49.)

Note – To initialize a board from a previous backup, both the master key backup file and the encrypted keystore data files are required. The encrypted keystore files must exist in the keystore directory (`/var/sca/keydata` by default). There are three files that must be placed in the top level keystore directory on the machine to which the keystore is being restored. The first fle is the `config` file for the keystore, which has the filename *keystore-name.serial-number.*{*keystore-id*}`.conf`. The second and third are the *user*`.db` and *object*`.db` files, which are located in the subdirectory under the top level keystore directory named *keystore-name.serial-number.*{*keystore-id*}

▼ Initialize the Board to Use an Existing Keystore

1. Initialize the board with the `scamgr` command.

- If the board is installed locally, enter `scamgr` at the system prompt.
- If the system is remote, enter `scamgr -h` *hostname*

1. Enter 2 as shown in the following example:

```
# scamgr -h hostname
This board is uninitialized.
You will now initialize the board.  You may
either completely initialize the board and
start with a new keystore or restore the board
using a backup file.

1. Initialize the board with a new keystore
2. Initialize the board to use an existing keystore

Your Choice (0 to exit) --> 2
```

2. Enter the path and password to the backup file:

Note – If the backup file was created in Multi-Admin mode, authentication is required by multiple security officers assigned the Multi-Admin role.

```
Enter the path to the backup file: /tmp/board-backup
Password for restore file:
```

3. Verify the configuration information:

```
Board restore parameters:
-----------------------------------------------------------------
Path to backup file: /tmp/board-backup
Keystore name: sca6000-keystore
Requires Multi-Admin auth: No
-----------------------------------------------------------------

Is this correct? (Y/Yes/N/No) [No]: y
Restoring data to crypto accelerator board...
```

Managing Keystores With `scamgr`

A keystore is a repository for key material. Associated with a keystore are security officers and users. Keystores not only provide storage, but a means for key objects to be owned by user accounts. This enables keys to be hidden from applications that do not authenticate as the owner. Keystores have three components:

- **Key objects –** Long-term keys that are stored for applications such as the Sun Java System Web Server.
- **User accounts –** These accounts provide applications a means to authenticate and access specific keys.
- **Security officer accounts –** These accounts provide access to key management functions through `scamgr`.

Note – A single board must have exactly one keystore. Multiple boards can be configured to collectively work with the same keystore to provide additional performance and fault tolerance.

Naming Requirements

Security officer names, user names, and keystore names must meet the following requirements:

TABLE 3-5 Security Officer Name, User Name, and Keystore Name Requirements

Name Requirement	Description
Minimum length	At least one character
Maximum length	63 characters for security officer names and user names. 32 characters for keystore names
Valid characters	Alphanumeric, underscore (_), dash (-), and dot (.)
First character	Must be alphabetic

Password Requirements

Password requirements vary based on the current `set passreq` setting (`low`, `med`, or `high`).

▼ Set the Password Requirements

1. **Start the `scamgr` utility.**
2. **Type `set passreq`.**

   This command sets the password character requirements for any password prompted by `scamgr`. There are three settings for password requirements, as shown in the following table:

TABLE 3-6 Password Requirement Settings

Password Setting	Requirements
`low`	Does not require any password restrictions. This is the default while the board is in non-FIPS mode.
`med`	Requires six characters minimum. Three characters must be alphabetic and one character must be nonalphabetic. This is the default setting while the board is in FIPS 140-2 mode and is the minimum password requirement allowed in FIPS 140-2 mode.
`high`	Requires eight characters minimum. Three characters must be alphabetic, and one character must be nonalphabetic. This is not a default setting and must be configured manually.

▼ Change Password Requirements

1. **Start the** `scamgr` **utility.**
2. **Type the** `set passreq` **command followed by** `low`**,** `med`**, or** `high`**.**

   The following commands set the password requirements for a board to `high`:

```
scamgr{mcaN@hostname, sec-officer}> set passreq high

scamgr{mcaN@hostname, sec-officer}> set passreq
Password security level (low/med/high): high
```

▼ Change Passwords

Only security officer passwords may be changed with `scamgr`. Security officers can change their own password.

1. **Start the** `scamgr` **utility.**
2. **Type** `set password`**.**

   For example:

```
scamgr{mcaN@hostname, sec-officer}> set password
Enter new security officer password:
Confirm password:
Security Officer password has been set.
```

User passwords may be changed through the PKCS#11 interface with the Sun Java System Web Server `modutil` utility. Refer to the Sun Java System Web Server documentation for details.

Managing Security Officers and Users

This section describes how to populate and list security officers and users and how to enable, disable, and delete them.

▼ Populate a Keystore With Security Officers

There might be more than one security officer for a keystore. Security officer names are known only within the domain of the board and do not need to be identical to any user name on the host system.

1. **Start the** `scamgr` **utility.**

2. **Type** `create so`.

   When creating a security officer, the name is an optional parameter on the command line. If the security officer name is omitted, `scamgr` prompts you for the name. (See "Naming Requirements" on page 44.) For example:

```
scamgr{mcaN@hostname, sec-officer}> create so Alice
Enter new security officer password:
Confirm password:
Security Officer Alice created successfully.

scamgr{mcaN@hostname, sec-officer}> create so
New security officer name: Bob
Enter new security officer password:
Confirm password:
Security Officer Bob created successfully.
```

▼ Populate a Keystore With Users

User names are known only within the domain of the board and do not need to be identical to the UNIX user name for the web server process.

1. **Start the** `scamgr` **utility.**

2. **Type** `create user` ***user-name*.**

When creating a user, the user name is an optional parameter on the command line. If the user name is omitted, `scamgr` prompts you for the user name. (See "Naming Requirements" on page 44.) For example:

```
scamgr{mcaN@hostname, sec-officer}> create user web-admin
Enter new user password:
Confirm password:
User web-admin created successfully.

scamgr{mcaN@hostname, sec-officer}> create user
New user name: Tom
Enter new user password:
Confirm password:
User Tom created successfully.
```

Users must use this password when authenticating during a web server startup.

Caution – Users must remember their password so they can access their keys. There is no way to retrieve a lost password.

Note – The user account is logged out if no commands are entered for more than five minutes. This is a tunable option. See "Set the Auto-Logout Time" on page 58 for details.

▼ List Users

You can list users associated with a keystore.Start the `scamgr` utility.

1. **Type the `show user` command. For example:**

```
scamgr{mcaN@hostname, sec-officer}> show user
User                                          Status
------------------------------------------------------
web-admin                                     Enabled
Tom                                           Enabled
------------------------------------------------------
```

▼ List Security Officers

You can list security officers associated with a keystore.

1. **Start the `scamgr` utility.**
2. **Type the `show so` command. For example:**

```
scamgr{mcaN@hostname, sec-officer}> show so
Security Officer                          Multi-Admin Role
------------------------------------------------------------------
sec-officer1                              Enabled
sec-officer2                              Enabled
sec-officer3                              Enabled
sec-officer4                              Disabled
------------------------------------------------------------------
```

▼ Disable Users

Note – Security officers cannot be disabled. Once a security officer is created, it is enabled until it is deleted.

Users and Security officers are enabled by default. Users may be disabled. Disabled users cannot access their key material with the PKCS#11 interface. Enabling a disabled user restores access to all of that user's key material.

1. **Start the** `scamgr` **utility.**
2. **Type** `disable user` ***user-name*.**

   When enabling or disabling a user, the user name is an optional parameter on the command line. If the user name is omitted, `scamgr` prompts you for the user name. For example:

```
scamgr{mcaN@hostname, sec-officer}> disable user Tom
User Tom disabled.
scamgr{mcaN@hostname, sec-officer}> disable user
User name: web-admin
User web-admin disabled.
```

▼ Enable Users

1. **Start the** `scamgr` **utility.**
2. **Type the** `enable user` ***user-name*** **command. When enabling a user, the user name is optional. For example:**

```
scamgr{mcaN@hostname, sec-officer}> enable user Tom
User Tom enabled.

scamgr{mcaN@hostname, sec-officer}> enable user
User name: web-admin
User web-admin enabled.
```

▼ Delete Users

1. **Start the** `scamgr` **utility.**

2. **Type** `delete user` ***user-name*.**

   When deleting a user, the user name is an optional parameter on the command line. If the user name is omitted, `scamgr` prompts you for the user name. For example:

```
scamgr{mcaN@hostname, sec-officer}> delete user web-admin
Delete user web-admin? (Y/Yes/N/No) [No]: y
User web-admin deleted successfully.

scamgr{mcaN@hostname, sec-officer}> delete user
User name: Tom
Delete user Tom? (Y/Yes/N/No) [No]: y
User Tom deleted successfully.
```

▼ Delete Security Officers

1. **Start the** `scamgr` **utility.**

2. **Type delete so so-name.**

   When deleting a security officer, the security officer name is an optional parameter on the command line. If the security officer name is omitted, `scamgr` prompts you for the security officer name. For example:

```
scamgr{mcaN@hostname, sec-officer}> delete so Bob
Delete Security Officer Bob? (Y/Yes/N/No) [No]: y
Security Officer Bob deleted.

scamgr{mcaN@hostname, sec-officer}> delete so
Security Officer name: Alice
Delete Security Officer Alice? (Y/Yes/N/No) [No]: y
Security Officer Alice deleted.
```

▼ Back Up the Master Key

Keystores are stored on the disk and encrypted in a master key. This master key is stored in the board firmware and can be backed up by a security officer.

1. **Start the** `scamgr` **utility.**

2. **Type** `backup` ***path-name*.**

   The path name can be placed on the command line or if omitted, `scamgr` prompts you for the path name.

Note – If the following command is executed in Multi-Admin mode, authentication is required by multiple security officers assigned the Multi-Admin role.

```
scamgr{mcaN@hostname, sec-officer}> backup /opt/backup-directory-name/bkup.data
Enter a password to protect the data:
Confirm password:
Backup to /opt/SUNWconn/mca/backups/bkup.data successful.
```

3. **Set a passphrase for the backup data.**

   This passphrase encrypts the master key in the backup file.

Caution – Choose a passphrase that is very difficult to guess when making backup files, because this password protects the master key for your keystore. You must also remember the password you enter. Without the password, you cannot access the master key backup file. There is no way to retrieve the data protected by a lost password.

Note – To initialize a board from a previous backup, both the master key backup file and the encrypted keystore data files are required. The encrypted keystore files must exist in the keystore directory (`/var/sca/keydata` by default). There are three files that must be placed in the top level keystore directory on the machine to which the keystore is being restored. The first fle is the `config` file for the keystore, which has the filename *keystore-name.serial-number.*{*keystore-id*}`.conf`. The second and third are the *user*`.db` and *object*`.db` files, which are located in the subdirectory under the top level keystore directory named *keystore-name.serial-number.*{*keystore-id*}

▼ Lock the Keystore to Prevent Backups

A site might have a strict security policy that does not permit the master key for a board to leave the hardware.

Caution – Once this command is issued, all attempts to back up the master key will fail. This lock persists even if the master key is rekeyed. The only way to clear this setting is to zeroize the board with the `zeroize` command. (See "Perform a Software Zeroize on the Board" on page 62.)

1. **Start the `scamgr` utility.**

2. **Type** `set lock`**. For example:**

```
scamgr{mcaN@hostname, sec-officer}> set lock
WARNING: Issuing this command will lock the
         master key.  You will be unable to back
         up your master key once this command
         is issued.  Once set, the only way to
         remove this lock is to zeroize the board.
Do you wish to lock the master key? (Y/Yes/N/No) [No]: y
The master key is now locked.
```

Multi-Admin Authentication

The `scamgr` utility includes a special mode of operation called Multi-Admin mode. In this mode, certain commands require multiple security officers to authenticate and approve the command before it can complete successfully. Security officers must be in the Multi-Admin role before they can authenticate Multi-Admin commands.

When a Multi-Admin command is issued, no other general administration on the board can take place until either the command times out, is cancelled by the security officer who started the command, or the command completes successfully. A timeout from 1 to 15 minutes must be set at or before Multi-Admin mode is enabled. See "Set a Multi-Admin Command Timeout" on page 53 for more information. Also security officers must set the number of Multi-Admin role members required to authenticate any Multi-Admin command.

When a Multi-Admin command is initiated, the `scamgr` session from which it is started will wait until one of three conditions occur: The command completes successfully, the command fails, or the command times out. Other Multi-Admin role members will log in to the device using their respective `scamgr` sessions. During Multi-Admin mode commands, these role members can only authenticate the command in progress. If the initiating security officer's `scamgr` session terminates unexpectedly, the security officer can log back in to the device and cancel the command. Otherwise, the board cannot be administered normally until the command times out.

Managing Multi-Admin Mode With `scamgr`

This section describes how to configure and manage Multi-Admin mode with the `scamgr` utility. First, you must identify your security officers and place them in the Multi-Admin role. You must have enough security officers in that role to satisfy the

minimum number set with the `set multiadmin minauth` command. If the number of Multi-Admin role members is below the minimum threshold, you cannot enable Multi-Admin mode.

▼ Assign Security Officers the Multi-Admin Role

1. **Start the `scamgr` utility.**
2. **Type `enable authmember` *so-name*.**

   If executed in Multi-Admin mode, this command requires authentication by multiple security officers assigned the Multi-Admin role. The following command assigns a security officer the Multi-Admin role.

```
scamgr{mcaN@hostname, sec-officer}> enable authmember sec-officer
Added multi-admin role to Security Officer sec-officer.
```

▼ Remove a Security Officer From the Multi-Admin Role

1. **Start the scamgr utility.**
2. **Type `disable authmember` *so-name*.**

   If executed in Multi-Admin mode, this command requires authentication by multiple security officers assigned the Multi-Admin role. For example:

```
scamgr{mcaN@hostname, sec-officer}> disable authmember sec-officer
Removed multi-admin role from Security Officer rew.
```

This command removes security officers from the Multi-Admin role only if they are in addition to the minimum required. This command exits only if a minimum number of security officers are assigned the Multi-Admin role. See "Set the Minimum Number of Security Officers Required to Authenticate Multi-Admin Commands" on page 52.

▼ Set the Minimum Number of Security Officers Required to Authenticate Multi-Admin Commands

1. **Start the `scamgr` utility.**

2. **Type** `set multiadmin minauth` *minimum-role-members*.

   The *minimum-role-members* value must be at least two and less than or equal to the total number of security officers on the system. In addition, if Multi-Admin mode is already enabled the new value cannot exceed the number of members with the Multi-Admin role. If executed in Multi-Admin mode, this command requires authentication by multiple security officers assigned the Multi-Admin role.

   For example, the following command sets the minimum number of required security officers to authenticate Multi-Admin commands.

```
scamgr{mcaN@hostname, sec-officer}> set multiadmin minauth 3
Multi-admin mode now requires 3 security officers to authenticate.
```

▼ Set a Multi-Admin Command Timeout

1. **Start the** `scamgr` **utility.**

2. **Type** `set multiadmin timeout` *minutes.*

   The *minutes* value must be between 1 and 1440 minutes (1 day). If a value larger than 1440 is specified, the value will be set to 1440. If executed in Multi-Admin mode, this command requires authentication by multiple security officers assigned the Multi-Admin role.

   For example, the following command changes the timeout for commands that require Multi-Admin mode authentication.

```
scamgr{mcaN@hostname, sec-officer}> set multiadmin timeout 3
New multi-admin timeout value is 3 minutes.
```

▼ Enable Multi-Admin Mode

1. **Start the** `scamgr` **utility.**

2. **Type** `enable multiadmin`**.**

   When enabled, certain commands require multiple security officers to authenticate before the command can complete successfully. When this command is executed, the security officer is presented with the current Multi-Admin mode settings and is given the opportunity to change these settings before the command completes. This command does not accept the `-y` (yes to all) flag.

   For example, the following command enables Multi-Admin mode.

```
scamgr{mcaN@hostname, sec-officer}> enable multiadmin
WARNING: This command will place the device in multi-
         admin mode.  This mode will require multiple
         security officers to authenticate for certain
         commands to be executed.

Enable Multi-Admin Mode? (Y/Yes/N/No) [No]: y

Multi-Admin mode parameters:
------------------------------------------------------------------
Minimum number of admins: 3
Multi-Admin command timeout: 3 minutes
------------------------------------------------------------------

Is this correct? (Y/Yes/N/No) [No]: y
The board is now in multi-admin mode.
```

Disable Multi-Admin Mode

1. **Start the** `scamgr` **utility.**

2. **Type** `disable multiadmin`**.**

   This command requires authentication by multiple security officers assigned the Multi-Admin role.

   For example, the following command disables Multi-Admin mode.

```
scamgr{mcaN@hostname, sec-officer}> disable multiadmin
```

Add Additional Security Officers to the Multi-Admin Role

1. **Start the** `scamgr` **utility.**

2. **Type enable authmember** `sec-officer`***N*.**

   Where *N* is the number of the security officer.

   With the minimum number of required security officers set to three, adding additional security officers requires the authorization of three different security officers, including the initiating security officer, to authenticate before this command can complete.

   Execute the following command on the initiating security officer's `scamgr` session.

```
scamgr{mca0@localhost, sec-officer1}> enable authmember sec-officer4
NOTICE: Please wait while the other required 2 administrators
        authenticate this command.  This command will time out
        in 3 minutes.

Update: Authenticated security officers: sec-officer1
Update: Authenticated security officers: sec-officer1 sec-officer3
Update: Authenticated security officers: sec-officer1 sec-officer3 sec-officer2
Added multi-admin role to Security Officer sec-officer4.
```

3. **Ask other security officers to log in from their respective `scamgr` sessions and authorize the command.**

```
# scamgr
Security Officer Login: sec-officer3
Security Officer Password:
NOTICE: A Multi-Admin command is currently in progress.
        You are a member of the Multi-Admin role and
        may approve this command.
Command: enable authmember sec-officer4
Initiating SO: sec-officer1

Authorize this command? (Y/Yes/N/No) [No]: y
Authorization successful
```

```
# scamgr
Security Officer Login: sec-officer2
Security Officer Password:
NOTICE: A Multi-Admin command is currently in progress.
        You are a member of the Multi-Admin role and
        may approve this command.
Command: enable authmember sec-officer4
Initiating SO: sec-officer1

Authorize this command? (Y/Yes/N/No) [No]: y
Authorization successful
```

▼ Cancel a Multi-Admin Command Originated by the Initiating Security Officer

1. **Start the** `scamgr` **utility.**
2. **Type** `disable authmember sec-officer`***N.***

   Where *N* is the number of the security officer.

   For example, the following command is cancelled. This command must be entered on the initiating security officer's `scamgr` session.

```
scamgr{mca0@localhost, sec-officer1}> disable authmember sec-officer4
NOTICE: Please wait while the other required 2 administrators
        authenticate this command.  This command will time out
        in 3 minutes.

Update: Authenticated security officers: sec-officer1
```

To cancel the command, the initiating security officer must either close the current `scamgr` session or log in with a second `scamgr` session.

```
# scamgr
Security Officer Login: sec-officer1
Security Officer Password:
NOTICE: A Multi-Admin command is currently in progress.
        Since you are the admin that initiated this
        command, you have the option of cancelling it.
        If you choose not to cancel the command, you
        will be logged out and the board will continue
        with the command.

Cancel this command? (Y/Yes/N/No) [No]: y
Authorization successful
```

If the `scamgr` session from which the command was initiated is still active, the following message is displayed.

```
Failed to remove role from Security Officer sec-officer4: Command cancelled
```

▼ Allow a Multi-Admin Command to Time Out

1. **Start the** `scamgr` **utility.**
2. **Type a command.**

3. **Ensure other security officers do not authenticate the command.**

   For example, the following command is issued by security officer.

```
scamgr{mca0@localhost, sec-officer1}> disable authmember sec-officer4
WARNING: Issuing this command will remove the
         multi-admin role from this security
         officer.  Once complete, this security
         officer will not be able to validate multi-
         admin commands.

Proceed with change? (Y/Yes/N/No) [No]: y
NOTICE: Please wait while the other required 2 administrators
        authenticate this command.  This command will time out
        in 3 minutes.

Update: Authenticated security officers: sec-officer1
Update: Authenticated security officers: sec-officer1 sec-officer2
Failed to remove role from Security Officer sec-officer4: Multi-Admin command
timeout
```

▼ Log In to a Board During a Multi-Admin Command as a Security Officer Not in the Multi-Admin Role

- **Log in as a non-multi-admin security officer.**

```
# scamgr
Security Officer Login: new-sec-officer
Security Officer Password:
You have authenticated successfully but this board is
currently waiting for all needed approvals for a
Multi-Admin mode command.  Since you are not a member
of the Multi-Admin role, you will not be able to
administer this board until this command has completed.

Connection closed.
```

▼ Attempt to Execute a Multi-Admin Command Without Multi-Admin Role Permissions

1. **Start the** `scamgr` **utility.**

2. **Type a command as a security officer without Multi-Admin role permissions.**

   The command fails. For example:

```
scamgr{mca0@localhost, new-so}> disable multiadmin
WARNING: Issuing this command will take the board
         out of multi-admin mode and return it to the
         single-administrator mode of authentication.

Proceed with change? (Y/Yes/N/No) [No]: y
Failed disabling Multi-admin mode: Unauthorized command
```

Managing Boards With `scamgr`

This section describes how to manage boards with the `scamgr` utility.

▼ Set the Auto-Logout Time

1. **Start the `scamgr` utility.**
2. **Type `set timeout` *N*.**

   Where *N* is the number of minutes before a security officer is automatically logged out. A value of 0 disables the automatic logout feature. The maximum delay is 1,440 minutes (1 day). A newly initialized board defaults to 5 minutes.

The following command changes the auto-logout time for a security officer to 10 minutes:

```
scamgr{mcaN@hostname, sec-officer}> set timeout 10
```

▼ Display Board Status

1. **Start the `scamgr` utility.**

2. **Type** `show status`**.**

   This command displays the hardware and firmware versions for that board, the MAC address of the network interface, the status (Up, Down, speed, duplex, and so on) of the network interface, and the keystore name and ID. For example:

```
scamgr{mcaN@hostname, sec-officer}> show status
Board Status
-------------------------------------------------------------
Version Info:
* Hardware Version: 1.2
* Firmware Version: 1.0
* Serial Number: 36:30:30:30:30:33

Keystore Info:
* Keystore Name: sca6000-keystore.600003
* Keystore ID: c3270900c3270900
* Keystore Lock: Disabled
* FIPS 140-2 Mode: Disabled

Security Settings:
* Login Session Timeout (in minutes): 5
* Password Policy Security Level: LOW
* Number of Master Key Backups: 0
* Multiadmin Mode: Enabled
* Minimum Number of Authenticators: 2
* Multiadmin Timeout: 5 Minutes
-------------------------------------------------------------
```

▼ Load New Firmware

You can update the firmware for the board as new features are added.

1. **Start the** `scamgr` **utility.**

2. **Type load** `firmware` ***path-name*****.**

   Where *path-name* is the path to the firmware file.

A successful update of the firmware requires you to manually reset the board with the `reset` command. When you reset the board, the currently logged in security officer is logged out.

```
scamgr{mcaN@hostname, sec-officer}> load firmware /usr/lib/crypto/firmware/sca
Security Officer Login: sec-officer
Security Officer Password:
WARNING: This command will load new firmware onto the
          the target device.  You must issue a reset
          command and log back into the target device in
          order to use the new firmware.

Proceed with firmware update? (Y/Yes/N/No) [No]: y
```

▼ Reset the Board

In certain situations, it might be necessary to reset the board. To do this, you must issue the `reset` command. You are asked if this is what you wish to do. Resetting a board might temporarily cease the acceleration of cryptography on the system unless there are other active board boards able to take over the load. Also, this command automatically logs you out of `scamgr`, so you must reconnect to the device by logging back into `scamgr` if you wish to continue administering it.

1. **Start the `scamgr` utility.**
2. **Type `reset`.**
3. **Type `y` to proceed, type `n` to cancel.**

   For example:

```
scamgr{mcaN@hostname, sec-officer}> reset
WARNING: Issuing this command will reset the
         the board and close this connection.

Proceed with reset? (Y/Yes/N/No) [No]: y
Reset successful.
```

▼ Rekey the Board

If your security policy changes, you might want to use new keys as the master key or remote access key. The `rekey` command enables you to regenerate either of these keys, or both.

Rekeying the master key also causes the keystore to be reencrypted under the new key, and invalidates older backed up master key files with the new keystore file. Make a backup of the master key whenever it is rekeyed. If you have multiple boards using the same keystore, you need to back up this new master key and restore it to the other boards.

Rekeying the remote access key logs the security officer out, forcing a new connection that uses the new remote access key.

1. **Start the `scamgr` utility.**
2. **Type rekey.**
3. **Specify which keys to rekey.**

   You may specify one of three key types when issuing the `rekey` command:

TABLE 3-7 Key Types

Key Type	Action
`master`	Rekeys the master key.
`remote`	Rekeys the remote access key. Logs the security officer out.
`all`	Rekeys both master and remote access keys.

The following is an example of entering a key type of `all` with the `rekey` command:

```
scamgr{mcaN@hostname, sec-officer}> rekey

Key type (master/remote/all): all
WARNING: Rekeying the master key will render all old board backups
         useless with the new keystore file. If other boards use this
         keystore, they will need to have this new key backed up and
         restored to those boards. Rekeying the remote access key will
         terminate this session and force you to log in again.

Rekey board? (Y/Yes/N/No) [No]: y
Rekey of master key successful.
Rekey of remote access key successful.  Logging out.
```

▼ Perform a Software Zeroize on the Board

There are two methods of clearing a board of all its key material. The first method is with a hardware jumper (shunt). This form of zeroizing returns the board to its original factory state (Failsafe mode). (See "Zeroizing the Sun Crypto Accelerator 6000 Hardware to the Factory State" on page 167.) The second method is to use the `zeroize` command.

Note – The `zeroize` command removes the key material, and leaves any updated firmware intact. This command also logs the security officer out upon successful completion.

---

1. **Start the `scamgr` utility.**
2. **Type zeroize.**
3. **Confirm the command.**

   For example:

```
scamgr{mcaN@hostname, sec-officer}> zeroize
WARNING: Issuing this command will zeroize all keys
         on the board.  Once zeroized, these keys
         cannot be recovered unless you have
         previously backed up your master key.

Proceed with zeroize? (Y/Yes/N/No) [No]: y
All keys zeroized successfully.
```

▼ Use the `scamgr diagnostics` Command

Diagnostics can be performed from the `scamgr` utility and from the SunVTS software. The `diagnostics` command in `scamgr` covers three major categories in the board hardware – general hardware, cryptographic subsystem, and network subsystem. Tests for general hardware cover DRAM, flash memory, the PCI bus, the DMA controller, and other hardware internals. Tests for the cryptographic subsystem cover random number generators and cryptographic accelerators. Tests on the network subsystem cover the `sca` device.

1. **Start the `scamgr` utility.**

2. **Type** `diagnostics`.

   For example:

```
scamgr{mcaN@hostname, sec-officer}> diagnostics
Performing diagnostic tests.  This may take a few minutes...Done.
Diagnostic Results
------------------------------------------------------------
General Hardware:                PASS
Cryptographic Subsystem:         PASS
USB Hardware:                    PASS
------------------------------------------------------------
```

Using the `scadiag` Utility

The `scadiag` utility provides a command-line interface to the board that enables superusers to perform administrative tasks without authenticating as security officer. Command-line options determine the actions that `scadiag` performs.

To access the `scadiag` utility easily, place the Sun Crypto Accelerator 6000 tools directory in your search path, for example:

```
$ PATH=$PATH:/usr/sbin/
$ export PATH
```

The `scadiag` command-line syntax (described in TABLE 3-8) for both Oracle Solaris and Linux is as follows:

- `scadiag [-?]`
- `scadiag [-d]` mca*N*
- `scadiag [-f]` mca*N*
- `scadiag [-k]` mca*N*
- `scadiag [-l]` mca*N* (device name is optional)
- `scadiag [-r]` mca*N*
- `scadiag [-u]` *fw-file device*
- `scadiag [-V]`
- `scadiag [-z]` mca*N*

The Oracle Solaris specific command-line syntax for `scadiag` is as follows:

- `scadiag [-m] [online|offline|diag]` mca*N*
- `scadiag [-s]` mca*N*

The Linux specific command-line syntax for `scadiag` is as follows:

- `scadiag [-m] [online|offline] mcaN`

Note – In the `scadiag` option examples in this section, `mcaN` is the board's device name where the *N* corresponds to the Sun Crypto Accelerator 6000 Board device instance number.

Note – Certain shells interpret the `?` character when using `-?` option on the command line. To avoid this, use the escape character (`\`) directly in front of the `?`. For example in the C shell, the command is changed to `scadiag -\?`.

TABLE 3-8 describes the supported options for the `scadiag` utility.

TABLE 3-8 `scadiag` Options

Option	Meaning
Oracle Solaris and Linux	
`-?`	Displays help files for scadiag commands.
`-d mcaN`	Performs diagnostics on the board.
`-f mcaN`	Displays the public key fingerprint used by the board for securing administration sessions.
`-k mcaN`	Displays the public key and the public key fingerprint used by the board for securing administration sessions.
`-l mcaN`	Lists devices. The device name (`mcaN`) is optional. Without the device name specified, this option lists all devices. With the device name specified, this option shows the mode of the device.
`-r mcaN`	Resets the board.
`-u` *fw-file device*	Loads the firmware file *fw-file* onto device. This command works only when the board is uninitialized. To upgrade firmware on an initialized board, use the `scamgr`(1m) command.
`-V`	Displays the version information for `scadiag`.
`-z mcaN`	Zeroizes the board.
Oracle Solaris Only	

TABLE 3-8 `scadiag` Options *(Continued)*

Option	Meaning
`-m online|offline|diag mca`*N*	Changes the mode of the device. The mode should be one of the following: • `online` – normal mode of operation (default) • `diag` – the device is offlined from keystore slot, and it can be accessed directly from an application (Oracle Solaris only) • `offline` – the device cannot be accessed from an application
`-s mca`*N*	Checks device status for possible DR. This option verifies whether the board is in use as a crypto service provider only.
Linux Only	
`-m online|offline mca`*N*	Changes the mode of the device. The mode should be one of the following: • `online` – normal mode of operation (default) • `offline` – the device cannot be accessed from an application

The following is an example of the `-d` option:

```
# scadiag -d mca0
Running mca0 on-board diagnostics.
Diagnostics on mca0 PASSED.
```

The following is an example of the `-f` option:

```
# scadiag -f mca0
b605-c285-392c-1c8f-5cc6-ec61-e617-1b7f-4ded-71b0
```

The following is an example of the `-k` option:

```
# scadiag -k mca0
Device: mca0
Key Length: 1024 bits
Key Fingerprint: b605-c285-392c-1c8f-5cc6-ec61-e617-1b7f-4ded-
71b0
Modulus:
        e4df259c 4725367a 3070ddff d78c4225 bf9a755c
        6d084667 fa043dd1 207595fb 4afdbe95 c9cca1ab
        f2a525ca 348cfffe 9c635056 94523f08 f7941797
        32d79603 3acf96c9 29c6b9ac d3f064ee 7c3a4790
        d06bf143 ce36a467 5f30332b b7782d93 17fc064b
        14438df6 679684ca afc599dc 3d1b2f87 30da4dc1
        63db86b7 48b1a29d
Public Exponent:
        00010001
```

The following is an example of the `-l` option:

```
# scadiag -l
mca/0
mca/1
# scadiag -l mca0
Device mca1:
State : Online
Status : Initialized
```

The following is an example of the `-m` option (note that the `diag` option is Oracle Solaris only):

```
# scadiag -m diag mca0
Device mca0 is now in diagnostic mode.
% scadiag -l mca0
Device mca0
State : Diag
Status: Initialized
```

The following is an example of the `-r` option:

```
# scadiag -r mca0
Resetting device mca0, this may take a minute.
Please be patient.
Device mca0 reset ok.
```

The following is an example of the `-s` option:

```
# scadiag -s mca0
Device mca0 free.
```

The following is an example of the `-u` option:

```
# scadiag -u fw-file mca0
Updating firmware on mca0, this may take a few minutes.
Please be patient.
Firmware update on mca0 complete.
Reset required to activate new firmware.
```

The following is an example of the `-V` option:

```
# scadiag -V
scadiag (Sun Crypto Accelerator 6000)
Copyright 2006 Sun Microsystems, Inc.
All rights reserved.
Use is subject to license terms.
```

The following is an example of the `-z` option:

```
# scadiag -z mca0
Zeroizing device mca0, this may take a few minutes.
Please be patient.
Device mca0 zeroized.
```

Managing Services for Oracle Solaris Platforms

Two service daemons are provided for the board: `scad` and `scakiod`. The `scad` service performs administrative I/O functions between the `scamgr`(1m) utility and the firmware. The `scakiod` service performs keystore I/O services. The Fault Management Resouce Identifiers (FMRIs) for these two services are `svc:/device/scad` and `svc:/device/scakiod`.

▼ Start and Stop the Services

- **Use the `svcadm`(1m) command to start and stop the services. For example:**

```
# svcadm enable scad
# svcadm enable scakiod
# svcadm disable scad
# svcadm disable scakiod
```

You can specify both services in a single command to start both simultaneously.

```
# svcadm enable scad scakiod
```

Service Configuration Parameters

TABLE 3-9 lists and describes the service parameters of the `scad` service.

TABLE 3-9 `scad` Service Parameter Descriptions

Parameter	Description
`debuglevel`	Sets the default log mask for events. By default, the level is NOTICE. The other accepted values, in order of increasing verbosity are INFO and DEBUG.
`log_file`	Takes a path name to a file specifically for logging data. This service sends log entries to `syslog` whether or not a log file is specified.
`hostbind`	Tells the service to listen explicitly on the IP address to which the specified *host-name* is resolved. Alternately, you may specify an IP address as the value for this property. If this property is undefined, the service listens on all interfaces.
`port`	Specifies the port that the service listens on for incoming connections. The default port is 6871.
`timeout`	Sets a timer for reads and writes of administrative data between clients and the service. The value is in seconds, and the default is to 300 seconds (five minutes).
`maxdata`	Sets a limit on the amount of data a client can send to the card in a single command. The value is in bytes and the default is 4 MB (4194304 bytes). Command data sizes that exceed this amount are rejected and the connection to the client is closed.

TABLE 3-10 lists and describes the service parameters of the `scakiod` service.

TABLE 3-10 `scakiod` Service Parameter Descriptions

Parameter	Description
`debuglevel`	Sets the default log mask for events. By default, the level is NOTICE. The other accepted values, in order of increasing verbosity are INFO and DEBUG.
`keystore_dir`	Sets an alternate directory for keystore files. The default value is `/var/sca/keydata`. Any alternate location must have read, write, and execute permissions for the user that the service runs as. Do not allow any permissions for any other user to this directory.
`log_file`	Takes a path name to a file specifically for logging data. This service sends log entries to `syslog` whether or not a log file is specified.
`hostbind`	This property is undefined by default, which makes the default behavior for this service is to bind to all interfaces. To bind the service to a specific hostname or IP address, you must define the hostbind property. See the following command.

▼ List Service Configuration Parameters

Configuration parameters are under the SMF property group `config`.

- **Use the `svccfg`(1m) command to list service properties. For example:**

```
# svccfg -s service-name listprop
```

The following is an example of listing the service properties for the `scakiod` service:

```
# svccfg -s scakiod listprop
general                         framework
general/action_authorization  astring  solaris.smf.manage.sca
general/entity_stability      astring  Unstable
general/single_instance       boolean  true
fs                            dependency
fs/entities                  fmri     svc:/system/filesystem/local
fs/grouping                   astring  require_all
fs/restart_on                 astring  none
fs/type                       astring  service
start                         method
start/exec                    astring  /usr/lib/crypto/scakiod
start/group                   astring  :default
start/limit_privileges        astring  :default
start/privileges              astring  :default
start/project                 astring  :default
start/resource_pool           astring  :default
start/supp_groups             astring  :default
start/timeout_seconds         count    30
start/type                    astring  method
start/use_profile             boolean  false
start/user                    astring  daemon
start/working_directory       astring  :default
stop                          method
stop/exec                     astring  :kill
stop/timeout_seconds          count    30
stop/type                     astring  method
config                        application
config/debuglevel             astring  NOTICE
config/keystore_dir           astring  /var/sca/keydata
config/log_file               astring  /var/sca/log/scakiod.log
config/value_authorization    astring  solaris.smf.manage.sca
tm_common_name                template
tm_common_name/C              ustring  "Sun Crypto Accelerator"
tm_man_scakiod                template
tm_man_scakiod/manpath        astring  /usr/share/man
tm_man_scakiod/section        astring  1M
tm_man_scakiod/title          astring  scakiod
```

▼ Modify Service Configuration Parameters

Use the `svccfg`**(1m)** command to modify a property as follows:

```
# svccfg -s service-name setprop config/property-name=value
```

For example, the following command defines the `hostbind` property:

```
# svccfg -s scad setprop config/hostbind=host: host1 host2 ...
```

Additional Instructions for Administering the Board on Linux Platforms

Administering the board on Linux platforms is similar to Oracle Solaris. The differences are given in this section.

The `scamgr` program is installed in the `/opt/sun/sca6000/bin` directory. You can put this directory in your path or directly launch the program with the following command:

```
% /opt/sun/sca6000/bin/scamgr
```

A newly installed board must be initialized before using (see "Initializing the Board With `scamgr`" on page 39). In addition, the board must be re-initialized after performing a zeroize (see Appendix E) with the `scamgr` program (see "Using the `scamgr` Utility" on page 30).

The board must be stopped and restarted after initialization or a zeroize. Use the following commands to stop and start the board:

```
% /etc/init.d/sca stop
% /etc/init.d/sca start
```

The `scadiag` program is installed in the `/opt/sun/sca6000/sbin` directory. You can place this directory in your path or directly launch the program with the following command:

```
% /opt/sun/sca6000/sbin/scadiag
```

The `scad` and `scakiod` daemons are installed in `/opt/sun/sca6000/sbin` directory. Do not start or stop these daemons manually. Stop and start the board to stop and start these daemons.

CHAPTER 4

Financial Services

Note – The financial services features described in this chapter are supported for the Oracle Solaris OS on both SPARC and x86 platforms. These features are not currently supported for the Linux OS.

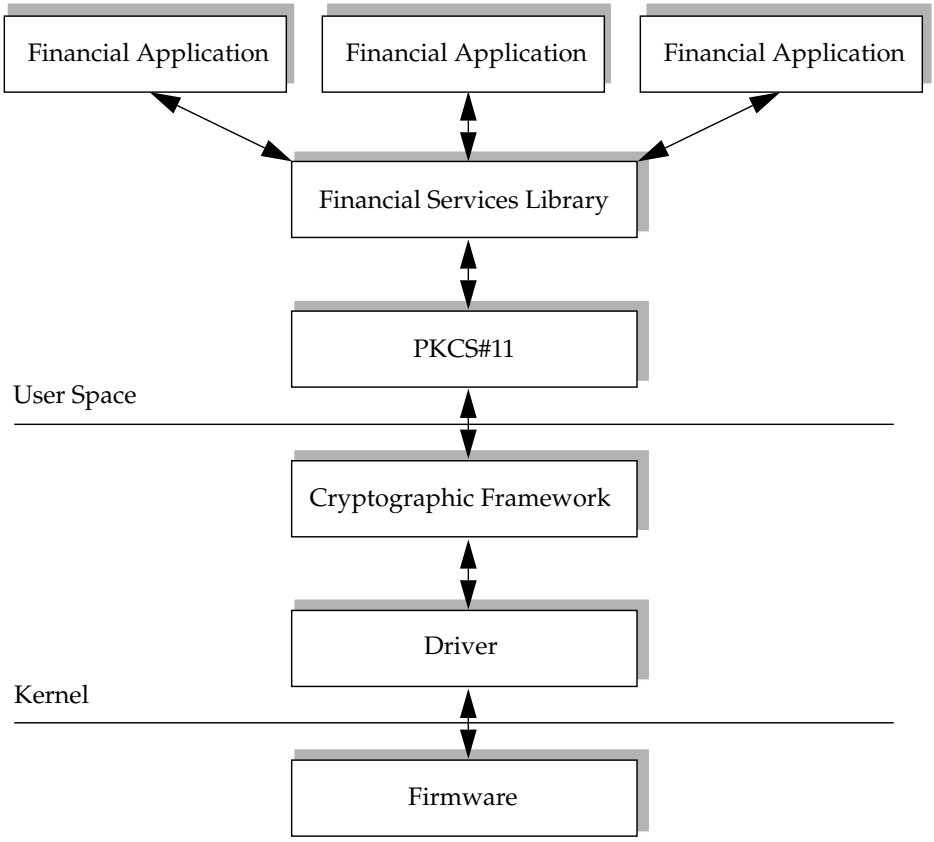
The Sun Crypto Accelerator 6000 Board supports PIN and credit card related functionality, ensuring the security of sensitive customer data by performing the entire operation within the secure cryptographic boundary of the board. Specialized key management capabilities and a new user library (`libfinsvcs.so`) and associated application interfaces are provided to support this feature. Data types referenced in this chapter are defined in the `/opt/SUNWsca/include/finsvcs.h` header file, which is included in Appendix F.

This chapter describes the application programming interface (API) to enable developers to use this new functionality. Basic familiarity with PIN and credit card processing and the associated standards is assumed. Sections include:

Financial Service Components Overview

The new financial service library uses the underlying PKCS#11 infrastructure to tunnel complex commands to the board resident firmware. Financial applications are not required to interpret the PKCS#11 interface, however, because this interprettation is handled by the financial services library. A high level overview of the financial services components is depicted in FIGURE 4-1:

FIGURE 4-1 Financial Services High Level Architecture



There are three core components that comprise the board financial services functionality:

- Key Management
- PIN processing
- Card processing

These core components are described in the following sections.

Enabling the Financial Services Feature

The financial services functionality is currently disabled by default due to an issue in the Oracle Solaris Cryptographic Framework. Enabling financial services in a redundant hardware configuration might cause errors under heavy loads due to this issue. In a single card configuration, these errors do not occur.

▼ Enable Financial Services

- **Make the following change in the** `/kernel/drv/mca.conf` **file:**

```
enable-finsvcs=1;
```

Financial Services Library Initialization

This section describes the functions used to initialize the financial services library.

Library Open Function `fs_lib_open()`

Financial services applications must issue the `fs_lib_open()` function to initialize the financial services library. This function locates the desired PKCS#11 provider and verifies that it supports the financial services mechanism. The `fs_lib_open()` function returns a handle that must be used in subsequent financial services library calls.

The syntax for the `fs_lib_open()` function is as follows:

```
fsLibHandle_t fs_lib_open(char *tokenName, fsReturn_t *err)
```

TABLE 4-1 lists the parameters for the `fs_lib_open()` function.

TABLE 4-1 `fs_lib_open()` Function Parameters

Parameter	Description
`tokenName`	Name of the desired token
`err`	Error value

TABLE 4-2 lists the return values for the `fs_lib_open()` function.

TABLE 4-2 `fs_lib_open()` Function Return Values

Return Value	Description
`fsOK`	Successfully initialized, desired token found, and support for financial services verified
`fsNotFound`	Token not found
`NULL`	Error occured
`Non-NULL`	Valid library handle returned when successfully initialized
`err`	`err` is set as follows: • `fsOK` – Successfully initialized, the desired token found, and support for financial services verified • `fsNotFound` – Token not found • `fsError` – Unable to initialize libary

Library Shutdown Function `fs_lib_close()`

Applications can close the financial services library services when the services are no longer required.

The syntax for the `fs_lib_close()` function is as follows:

```
fsReturn_t fs_lib_close(fsLibHandle_t handle)
```

TABLE 4-3 lists the parameters for the `fs_lib_close()` function.

TABLE 4-3 `fs_lib_close()` Function Parameters

Parameter	Description
`handle`	Financial services library handle returned from the `fs_lib_open` function

TABLE 4-4 lists the return values for the `fs_lib_close()` function.

TABLE 4-4 `fs_lib_close()` Function Return Values

Return Value	Description
`fsOK`	Successfully closed the financial services library
`fsInvalidHandle`	Library handle invalid

Session Establishment Function `fs_session_open()`

Users can establish multiple financial services sessions, thus allowing multithreaded access to the financial services capabilities. Sessions can be created only after you have initialized the financial services library with the `fs_lib_open()` function. A unique session handle is returned and must then be used for all financial service requests for that specific session.

The syntax for the `fs_session_open()` function is as follows:

```
fsSessHandle_t fs_session_open(fsLibHandle_t handle)
```

TABLE 4-5 lists the parameters for the `fs_session_open()` function.

TABLE 4-5 `fs_session_open()` Function Parameters

Parameter	Description
`handle`	Library handle obtained from the `fs_lib_open()` function

TABLE 4-6 lists the return values for the `fs_session_open()` function.

TABLE 4-6 `fs_session_open()` Function Return Values

Return Value	Description
`Non-NULL`	Session handle upon success
`NULL`	On error

Session Shutdown Function `fs_session_close()`

Once a user has completed financial operations, they can dissolve the session by issuing the `fs_session_close()` function. The session handle obtained from the `fs_session_open()` function must be used with this function.

The syntax for the `fs_session_close()` function is as follows:

```
fsReturn_t fs_session_close(fsSessHandle_t handle)
```

TABLE 4-7 lists the parameters for the `fs_session_close()` function.

TABLE 4-7 `fs_session_close()` Function Parameters

Parameter	Description
`handle`	Session handle from previous `fs_session_open()` function

TABLE 4-8 lists the return values for the `fs_session_close()` function.

TABLE 4-8 `fs_session_close()` Function Return Values

Return Value	Description
`fsOK`	Session closed successfully
`fsInvalidHandle`	Session handle invalid

Financial Services Data Types

The financial services API requires the use of new data types defined in the `finsvcs.h` header file. Appendix F provides the `finsvcs.h` header file.

Key Management Overview

To meet the strict key management requirements of financial institutions, the board adheres to the following essential financial key management principles.

Key Separation and Compartmentalization of Risk

Keys must be used for specifically defined functions only. This requirement limits potential damage from a key compromise. To meet this requirement, functional key type information is associated with each financial key. The board allows generating and importing the types of keys defined in the following list and enforces that they are used for specific operations only.

The following types of financial keys are supported:

- ***Master File Key (MFK)***

  The board is a dedicated hardware security module (HSM). The MFK never leaves the secure HSM and encrypts other operational keys when they leave the HSM. An MFK can be used only on the encrypting HSM. MFKs are entered into the board in component form with the direct input device.

- ***Key Encryption Key (KEK)***

  Encrypts other keys for key exchange operations. The KEKs are entered into the board in component form with the direct input device.

- ***PIN Encryption Key (PEK)***

  Encrypts PINs. There are two types of supported PEKs:

  - Terminal PIN Key (TPK) – Encrypts PINs on the terminal side of the transaction (ATM, POS device).
  - Zone Working Key (ZWK) – Encrypts PINS when transferring between different financial institutions.

- ***PIN Verification Key (PVK)***

  Verifies PIN operations.

- ***Card Verification Key (CVK)***

  Verifies card operations.

Allowed Key Forms

The following key form requirements are enforced by the board.

- Cleartext keys must stay within the board, with the exception of existing as at least two separate components, each under the control of a different security officer.
- When not stored in the board or when in component form, all keys must be enciphered with a key of equal or greater cryptographic strength.

Direct Key Loading

For security, the MFK and KEKs are entered directly into the board with the direct input device connected to the board's serial port. These keys are entered in component form by unique security officers. This extra security step is required to meet the following key management requirements:

- *Split knowledge* – No single user can know the entire key.
- *Dual control* – The component and a valid user name and password are required to enter a key component.

Loading the MFK

The MFK is loaded with the direct input device. The MFK is entered as a series of key components that are combined to create the MFK. Each component is entered by a different security officer, preventing any single user from knowing the MFK. The MFK is the only financial services key maintained in the board. This key never leaves the board.

The command syntax for loading the MFK is as follows:

```
sca6000, so}> load mfk
```

Enabling the MFK

Once the MFK components are loaded, the MFK must be enabled with the direct input device by a valid security officer.

The command syntax for enabling the MFK is as follows:

```
sca6000, so}> enable mfk
```

Loading the KEKs

The KEKs are also entered with the the direct input device. However, unlike the MFK, these keys are extracted from the board by financial applications. The extracted keys are encrypted with the MFK. The KEKs are never in the clear. Similar to the MFK, KEKs are entered in component form preventing any single individual from knowing the actual KEK. A label must also be entered with the direct input device and is used programmatically by an application to retrieve the desired KEK.

The command syntax for loading the KEKs is as follows:

```
sca6000, so}> load kek
```

Change the MFK

Financial applications require their keys be encrypted using the MFK, thus changing the MFK is a complex process. After entering the new MFK, the applications must request that all of their keys be reencrypted using the new MFK with the `fs_translate_key`() function. The board must maintain the old MFK until this process is complete. Once this process is complete, a security officer can use the new MFK by typing the `enable mfk` command.

Key Management Functions

Financial applications require key management related functionality from the HSM. The following basic capabilities are required:

- Generate Key
- Import Key
- Export Key
- Translate Key (from the old MFK to the new MFK)
- Retrieve KEK

Generate Key Function `fs_generate_key()`

Applications generate different types of financial keys. DES keys are primarily used for these functions. Along with the key type, key usage information is required to limit when the key can be used. The following types of key uses are supported:

- PIN Encryption Keys
- Terminal PIN Key (TPK)
- Zone Working Key (ZWK)
- PIN Verification Key (PVK)
- Card Verification Key (CVK)

Once generated these keys are encrypted by the MFK and returned in the user provided buffer upon success.

The syntax for the `fs_generate_key()` function is as follows:

```
fsReturn_t fs_generate_key(fsSessHandle_t handle, fsKeyType_t
type, fsKeyUsage_t usage, fsKey_t *key)
```

TABLE 4-9 lists the parameters for the `fs_generate_key()` function.

TABLE 4-9 `fs_generate_key()` Function Parameters

Parameters	Description
`handle`	Session handle returned by `fs_session_open()`
`type`	Key algorithm type: DES, DES2, DES3
`usage`	Intended financial key usage
`key`	Buffer for the generated key, encrypted with the MFK or a derivative

TABLE 4-10 lists the return values for the `fs_generate_key()` function.

TABLE 4-10 `fs_generate_key()` Function Return Values

Return Value	Description
`fsOK`	Key generated
`fsInvalidKeyType`	Invalid key type specified
`fsBufferTooSmall`	Provided buffer is too small for the key
`fsInvalidState`	Device not in proper state to handle command

Import Key Function `fs_import_key()`

To interoperate with peer nodes, the board must be able to import keys from these peers. Applications can import existing keys from peer nodes with the import function.

The syntax for the `fs_import_key()` function is as follows:

```
fsReturn_t fs_import_key(fsSessHandle _t handle, fsKeyUsage_t
usage,  fsKey_t *KEK, fsKey917_t *iKey, fsKey_t *oKey, BOOLEAN
useVariants)
```

TABLE 4-11 lists the parameters for the `fs_import_key()` function.

TABLE 4-11 `fs_import_key()` Function Parameters

Parameters	Description
`handle`	Session handle returned by the `fs_session_open()` function
`usage`	Type of intended key usage: TPK, ZWK, PEK, CVK, and so on
`kek`	KEK key shared with the peer node with which the imported key was encrypted
`ikey`	Imported key – this is an ANSI X9.17 formatted key
`oKey`	Key returned and translated by the MFK
`usevariants`	True if the imported key is an Atalla variant

TABLE 4-12 lists the return values for the `fs_import_key()` function.

TABLE 4-12 `fs_import_key()` Function Return Values

Return Value	Description
`fsOK`	The `oKey` is filled in for this case if the key is successfully imported
`fsInvalidKeyType`	Invalid import key type
`fsInvalidKeyUsage`	Unsupported key usage type
`fsInvalidKEK`	Invalid KEK
`fsInvalidKey`	Import key is invalid
`fsInvalidState`	Device is not in proper state to handle command
`fsError`	Processing error

Export Key Function `fs_export_key()`

The board must allow users to move keys from one device to another. Additionally, the peer device might not be a board. The export key function enables this feature and allows users to export keys from the board. How the exported keys are transported to the peer is determined by the application developer.

The syntax for the `fs_export_key()` function is as follows:

```
fsReturn_t fs_export_key(fsSessHandle_t handle, fsKeyUsage_t
usage, fsKey_t *KEK, fsKey_t *iKey, fsKey917_t *oKey, boolean_t
useVariants);
```

TABLE 4-13 lists the parameters for the `fs_export_key()` function.

TABLE 4-13 `fs_export_key()` Function Parameters

Parameter	Description
`handle`	Session handle returned by the `fs_session_open()` function
`usage`	Type of intended key usage: TPK, ZWK, PEK, CVK, and so on
`KEK`	Key shared with the peer node with which the exported key is encrypted
`iKey`	Input key encrypted with the MFK
`oKey`	Exported key in ANSI 9.17 format
`useVariants`	True if the exported key is an Atalla variant

TABLE 4-14 lists the return values for the `fs_export_key()` function.

TABLE 4-14 `fs_export_key()` Function Return Values

Return Value	Description
`fsOK`	The `oKey` is filled in for this case if key successfully exported
`fsInvalidKeyType`	Export key type invalid
`fsInvalidKeyUsage`	Key usage type invalid
`fsInvalidKey`	Key for export invalid
`fsInvalidKEK`	Encryption key invalid
`fsInvalidState`	Device is not in proper state to handle command
`fsError`	Processing error

Translate Key Function `fs_translate_key()`

When the MFK is updated, users must convert all of their keys using the new MFK.

The syntax for the `fs_translate_key()` function is as follows:

```
fs_return_t fs_translate_key(fsSessHandle_t handle, fsKey_t *iKey,
fsKey_t *oKey)
```

TABLE 4-15 lists the parameters for the `fs_translate_key()` function.

TABLE 4-15 `fs_translate_key()` Function Parameters

Parameter	Description
`handle`	Session handle returned by the `fs_session_open()` function
`iKey`	Input key encrypted with the old MFK
`oKey`	Output key encrypted with the new MFK

TABLE 4-16 lists the return values for the `fs_translate_key()` function.

TABLE 4-16 `fs_translate_key()` Function Return Values

Return Value	Description
`fsOK`	Key converted with the new MFK
`fsInvalidKey`	Input key invalid
`fsInvalidState`	Device is not in proper state to handle command

Retrieve Object Function `fs_retrieve_object()`

Applications can retrieve select objects from the board. For security reasons, there are two types of objects that are input with the direct input device that can be retrieved by applications:

- KEKs
- Decimalization tables used in IBM-3624 PIN verification operations

  When the object is entered, a unique label is also entered. This label is used to locate the object.

The syntax for the `fs_retrieve_object()` function is as follows:

```
fsReturn_t fs_retrieve_object(fsSessHandle_t handle,
fsObjectType_t type, char *label, fsObjectData_t *obj)
```

TABLE 4-17 lists the parameters for the `fs_retrieve_object()` function.

TABLE 4-17 `fs_retrieve_object()` Function Parameters

Parameter	Description
handle	Session handle returned by the `fs_session_open()` function
label	Byte string identifier for the object
obj	Output buffer where the object is returned
type	Type of object to retrieve: KEK or decimalization table

TABLE 4-18 lists the return values for the `fs_retrieve_object()` function.

TABLE 4-18 `fs_retrieve_object()` Function Return Values

Return Value	Description
`fsOK`	Object located and retrieved
`fsNotFound`	Object not located
`fsBufferTooSmall`	Output buffer too small to hold object
`fsInvalidState`	Device is not in proper state to handle command

Status Function `fs_status()`

An application can query the board for its current status. The following board status values are supported:

- `fsStateUninit` – Device not initialized for financial services.
- `fsStateNormalMode` – Core functionality enabled. User's cannot import or export keys in this mode.
- `fsStateSensitiveMode` – Only key import and export requests allowed.
- `fsStateMfkChange` – Change of the MFK pending – only key translations done in this mode.

The syntax for the `fs_status()` function is as follows:

```
fsReturn_t fs_status(fsStatusBuff_t *status)
```

The parameter for the `fs_status()` function is as follows:

- `status` – Status buffer

PIN Processing Functions

The Sun Crypto Accelerator 6000 Board supports PIN verification and translation functionality. The interface ensures that sensitive customer data is exposed only within the secure HSM. This section describes the capabilities and interfaces supported.

PIN Block Formats

The board supports *ANSI/ISO Format 0* and *ISO Format 1* PIN-blocks described in this section.

ANSI/ISO Format 0

The ANSI/ISO Format 0 is an 8-byte block constructed with the combining of two 64-bit components: The *clear text PIN*, and the *cleartext account number field*.

The cleartext PIN, represented in hexadecimal characters, appears as follows:

```
C N P P P P P/F P/F P/F P/F P/F P/F P/F P/F F F
```

The cleartext PIN hexadecimal characters are defined in TABLE 4-19.

TABLE 4-19 ANSI/ISO Format 0 Cleartext PIN Hexadecimal Characters

Field	Name	Value
C	Control field	4-bit field with binary value of 0000
N	PIN length	4-bit field with binary value between 0x4 and 0xc
P	PIN digit	4-bit field with binary value of 0000 to 1001
P/F	PIN/filler	4-bit field with binary value determined by PIN length
F	Filler	4-bit field with binary value of 1111

The cleartext account number field, represented in hexadecimal characters, appears as follows:

```
C C C C A A A A A A A A A A A A
```

The cleartext account number field hexadecimal characters are defined in TABLE 4-20.

TABLE 4-20 ANSI/ISO Format 0 Cleartext Account Number Field Hexadecimal Characters

Field	Name	Value
C	Control field	4-bit field with binary value of 0000
A	Primary account number (PAN)	12 right most digits of the PAN represented as 4-bit binary numbers with values of 000 to 1001 (0 to 9)

ISO Format 1

ISO Format 1 is the supported ISO-1 PIN block. This format supports a PIN length between 4 and 12 digits. PINs longer than 12 digits are truncated.

The ISO-1 PIN-block format, represented in hexadecimal characters, appears as follows:

```
 1 L P P P P P/R P/R P/R P/R P/R P/R P/R P/R R R
```

The ISO-1 PIN-block format hexadecimal characters are defined in TABLE 4-21.

TABLE 4-21 ISO Format 1 Hexadecimal Characters

Field	Name	Value
1	Fixed data	4-bit binary value of 0x1
L	PIN length	4-bit binary value between 0x4 and 0xc
P	PIN digit	4-bit binary value between 0x0 and 0x9
R	Random Digit	4-bit binary value between 0x0 and 0xf
P/R	PIN digit or random digit	4-bit binary value determined by the PIN length – P or R

PIN Calculation Methods

The Sun Crypto Accelerator 6000 Board supports the Visa PIN Validation Value (PVV) and the IBM-3624 methods for calculating PINs.

Visa PVV

PVV is a calculation and verification method specified by Visa. The credit card issuer or a designated agent provides a PIN Verification Service (PVS). This service compares the cardholder's PIN to a cryptographic transformation of the PIN.

The PVV method is a two step process:

1. When a PIN related credit card is issued, the issuer derives a 4-digit PVV. The PVV and the PIN Verification Key Index (PVKI) are either encoded on the credit card or registered in an online database. The stored PVV is called the reference PVV.
2. When a cardholder enters their PIN at the point of service, a transaction PVV is generated. The transaction PVV is compared to the reference PVV by the issuer or their agent. If the two PVVs match, the cardholder is authenticated.

For PVV, the following input is required:

- The customer's primary account number (PAN)
- The customer's PIN
- A one digit PVKI

IBM-3624

The following input is required for the IBM-3624 PIN calculation method:

- The customer’s PIN
- The PIN verification key (PVK) used in the PIN calculation algorithm and encrypted with the MFK
- Validation information for identifing the customer, this is typically the customer’s account number
- Check length, this is the number of PIN digits to check
- Reference offset data
- Decimalization table used to convert algorithm output into decimal digits

Personal Account Number (PAN)

The right most 11 digits of the PAN, excluding the mod-10 check digit that must be used to generate the PVV – for example:

PAN	PAN Digits Selected

4839 1234 5678 9019	1234 5678 901

PIN

The PIN associated with the PAN must be used to generate the PVV. Regardless of the length of the PIN (4 to 12 digits), only the left most four digits are used.

PVKI

The PVKI is a one digit value that identifies which PIN verification key (PVK) to use for the PVV calculation. The PVKI is a single digit value from 0 to 6. A PVKI of 0 indicates that the PIN cannot be verified through PVS.

PIN Verify Function `fs_pin_verify()`

The PIN verify operation is executed by the credit card issuer or their agent to authenticate a cardholder's transaction. The Sun Crypto Accelerator 6000 board supports two types of PIN verification, Visa PVV and IBM-3624. Additionally, the board supports two types of PIN block formats, ANSI/ISO Format 0 and ISO Format 1.

The syntax for the `fs_pin_verify()` function is as follows:

```
fsReturn_t fs_pin_verify(fsSessHandle_t handle, fsPinAlg_t alg,
fsKey_t *PEK, fsKey_t *PVK,  fsPAN_t *PAN, fsPIN_t *iPIN,
fsPinData_t *data)
```

TABLE 4-22 lists the parameters for the `fs_pin_verify()` function.

TABLE 4-22 `fs_pin_verify()` Function Parameters

Parameter	Description
`handle`	Session handle returned by `fs_session_open()` function
`alg`	PIN algorithm: Visa PVV or IBM-3624
`PEK`	PIN encryption key encrypted with the HSM's MFK – the PIN has been encrypted with this key
`PVK`	Key encrypted with the MFK and used in the PIN verification computation

TABLE 4-22 `fs_pin_verify()` Function Parameters *(Continued)*

Parameter	Description
`PAN`	Personal account number
`iPIN`	Encrypted input PIN
`data`	PIN algorithm specific data For Visa PVV `data` consists of: • PVKI • Reference PVV For IBM-3624 `data` consists of: • Decimalization table • Validation data • Check length • Offset data

TABLE 4-23 lists the return values for the `fs_pin_verify()` function.

TABLE 4-23 `fs_pin_verify()` Function Return Values

Return Value	Description
`fsOK`	PIN was verified
`fsOK`	PIN failed verification
`fsInvalidPEK`	PEK invalid
`fsInvalidPinType`	PIN block format invalid
`fsInvalidPVK`	PVK invalid
`fsInvalidPVKI`	PVKI invalid (0 < PVKI `||` PVKI > 6)
`fsInvalidState`	Device not in correct state to process command
`fsInvalidDectbl`	Invalid decimalization table
`fsError`	Processing error

PIN Translate Function `fs_pin_translate()`

This function translates a PIN from one encryption key to another. This function is typically done at banking transactions. An example is when a cardholder uses their ATM card at a different bank than the one that issued the card. At the transaction, the PIN comes in encrypted using a PIN encryption key (PEK) specified by the point of service bank. To route the transaction to the credit card issuing bank, the

transaction decrypts the PIN using the transaction originator's PEK and then reencrypts it using the credit card issuing bank's PEK. The PIN block format can be requested to be translated (from ISO Format 0 to ISO Format 1 for example).

The syntax for the `fs_pin_translate()` function is as follows:

```
fsReturn_t fs_pin_translate(fsSessHandle_t handle, fsKey_t *iPEK,
fsKey_t *oPEK, fsPIN_t *iPIN, fPIN_t *oPIN, fsPAN_t *PAN)
```

TABLE 4-24 lists the parameters for the `fs_pin_verify()` function.

TABLE 4-24 `fs_pin_verify()` Function Parameters

Parameter	Description
`handle`	Session handle returned by the `fs_session_open()` function
`iPEK`	Input PEK used to encrypt the PIN – this key is encrypted with the MFK
`oPEK`	Output PEK encrypted with the MFK
`iPIN`	Encrypted input PIN
`oPIN`	Buffer for encrypted output PIN
`PAN`	Personal account number

TABLE 4-25 lists the return values for the `fs_pin_verify()` function.

TABLE 4-25 `fs_pin_verify()` Function Return Values

Return Value	Description
`fsOK`	Operation successful
`fsInvalidKey`	Source or destination PEK invalid
`fsInvalidKeyUsage`	Key usage type invalid
`fsInvalidPinType`	Source or destination PIN block format invalid
`fsInvalidPin`	PIN invalid or corrupt – PIN must be decimal digits
`fsInvalidPan`	PAN invalid or corrupt
`fsInvalidState`	Device not in proper state to handle command

Credit Card Processing Overview

The Sun Crypto Accelerator 6000 Board board supports credit card verification processing for the major types of credit cards, Visa, MasterCard, and American Express. The interface ensures that sensitive customer data is only exposed within the HSM. This section describes the capabilities and interfaces supported.

Credit Card Processing Functions

The card verification (CV) is a cryptographic checksum of the data stored on a magnetic card. The Sun Crypto Accelerator 6000 board supports CV operations for Visa, MasterCard, and American Express credit cards.

Credit Card Verification Methods

Credit card verification is performed during normal ATM and POS transactions to verify the magnetic card data. The following algorithms are supported:

- CVV – Visa
- CVC – MasterCard
- CSC – American Express

Administering Financial Services

This section describes the financial services administrative features and commands.

Financial Services Security Officers (FSSO)

FSSOs have specific financial services permissions. Each FSSO requires a unique security officer account created with the `scamgr` utility, described in Chapter 3. Only security officers can create and delete FSSO accounts. FSSO authentication is required to input keys and certain commands with the direct input device. These accounts can be configured to require single or multiple FSSO authentication for

certain commands. To require multiple FSSOs per board to authenticate commands, a security officer must enable Multi-Admin mode, which is described in "Multi-Admin Authentication" on page 51.

Direct Input Device

A direct input device is required to load critical security parameters into the board. Only FSSOs can use the direct input device. See "Direct Input Devices" on page 7.

The direct input device is required to import the MFK and the KEKs. An FSSO must log in to the board with the direct input device and then initiate the `key input` command to import these keys.

Setting Financial Services Mode (`fsmode`)

To enable financial services features, an FSSO must place the board in one of two modes:

- Normal mode – Enables all functions except importing and exporting keys.
- Sensitive mode – Enables importing and exporting keys.

Administrative Commands

TABLE 4-26 describes the financial services administrative commands.

Note – Only FSSOs can initiate the commands listed in TABLE 4-26, and each command must be entered with a direct input device.

TABLE 4-26 Financial Services Administrative Commands

Command	Description
`key input`	Enters the MFK or the KEKs – the direct input device must be used to enter this command.
`load mfk`	Initiates the MFK key installation. After issuing this command, you can enter the respective key component and log out. Subsequent FSSOs can then log in and enter this command and enter their key component. Once the minimum number of components (default 2) have been entered, the key is considered pending and the device is disabled for everything other than key translation requests. Unique FSSOs must enter each component, otherwise an error is reported.
`enable mfk`	Activates a new MFK and deletes the old one. Use this command after all applications have translated their keys under the new MFK.
`cancel mfk`	Cancels the MFK. Must be initiated before entering all of the MFK components.
`delete mfk`	Deletes the MFK. Must be done before enabling a pending MFK. If there is a previously enabled MFK, the board reverts to it.
`load kek`	Installs a KEK. You are prompted for a key label to associate with the key. The KEK is installed in component form similar to the MFK, so after entering the first component, you can log off. Additional security officers can then log in and enter their respective components, then log off. Unique FSSOs must enter each component otherwise an error is returned.
`cancel kek`	Cancels a KEK. Must be done while entering a KEK and before all components are entered. Note that KEKs are only temporarily stored on the board. Once an application retrieves the object, it is deleted. Additionally, KEKs are not preserved during a board reset.
`delete kek`	Deletes a KEK. Only authorized security officers can delete a previously entered KEK. The label used when the KEK was entered must be specified to locate the KEK.
`load decimalization table`	Loads a decimalization table, which is required for IBM-3624 PIN verification. You are prompted for a label to associate with the decimalization table. The entered decimalization table is encrypted with the MFK and can be retrieved by an application with the `fs_retrieve_object`() function.
`delete decimalization table`	Deletes a decimalization table.

CHAPTER 5

Building PKCS#11 Applications for Use With the Sun Crypto Accelerator 6000 Board

This chapter describes the board's implementation of the PKCS#11 interface and describes how to build customized PKCS#11 applications to be used with the board. Additional instructions for Linux platforms are included in the last section. Sections include:

The Sun Crypto Accelerator 6000 Board is registered in the Oracle Solaris Cryptographic Framework as a hardware provider. Thus, the board can be administered using the system commands. Refer to the *Oracle Solaris 10 System Administration Guide: Security Services* document.

The Oracle Solaris Cryptographic Framework provides a PKCS#11 library through which the board is accessed. For Oracle Solaris SPARC platforms, the default location for this library is `/usr/lib/` for 32-bit mode and `/usr/lib/sparcv9/` for 64-bit mode.

```
/usr/lib/libpkcs11.so
/usr/lib/sparcv9/libpkcs11.so
```

For Oracle Solaris x64 platforms, the default location for this library is `/usr/lib/64/`:

```
/usr/lib/64/libpkcs11.so
```

Note – The `64` directory is a softlink that resolves to `sparcv9` or `amd64` depending on your architecture.

Board Administration

PKCS#11 has a limited administrative facility with just two functions – `C_InitToken`, which initializes the token, and `C_InitPin`, which sets user PINs. The board does not use this facility, and instead uses the `scamgr` utility. See Chapter 3.

When the board is first initialized, `scamgr` prompts you to set up a security officer account. This security officer is not related to the PKCS#11 security officer, and cannot authenticate to a board through the PKCS#11 interface.

Also during board initialization, `scamgr` prompts for the keystore name. The keystore name is used as the slot description and the token label for the Keystore slot. See "Keystore Slot" on page 99.

After the board is initialized, the security officer can create one or more users using the `scamgr` utility. Users created by the security officer authenticate to a board through the PKCS#11 interface. Since PKCS#11 is designed for a single-user system, the `C_Login` entry point does not take the username as a parameter. To differentiate users, a PIN must be given as a string of the form `username:password`. For example, if the password of user `webserv` is `abc123`, the PIN used through the PKCS#11 `C_Login` entry point is `webserv:abc123`.

Slot Descriptions

There are four kinds of slots available through the Oracle Solaris PKCS#11 library.

- Keystore slot

  The Keystore slot groups together the multiple hardware providers that share a common keystore to support availability and load balancing. The Keystore slot description and the token label for the board are made up of the keystore name padded with spaces.

- Sun Metaslot

  The Sun Metaslot uses all of the cryptographic engines on the system, including the board; thus, it provides the maximum functionality. By default, Sun Metaslot uses the Oracle Solaris Softtoken keystore; however it can be configured to use the board keystore. See "Sun Metaslot" on page 100.

- Hardware slot

  The Hardware slot is bound to and dedicated to a hardware device. These slots are directly accessible when the device is uninitialized or when it is in diagnostic mode. There should be three hardware slots per board. These slots are useful for diagnosis because they are directly associated with a board. The hardware slot description and the token label for the board are in the following format: `mca/`*N* `Crypto Accel 1.0. [CB|CA|OM]`. Where *N* is the instance number.

- Sun Softtoken slot

  The Sun Softtoken slot is a software cryptographic provider with an on-disk keystore.

The following subsections provide details on the Keystore slot, Sun Metaslot, and Hardware slot.

Keystore Slot

The Keystore slot has the advantage of hardware redundancy and load balancing when there are more than one board on the system with the same keystore. For example, when there are two boards with the same keystore with the name of `ks`, a slot with the slot description and token label of `ks` is used as the Keystore slot.

When the Keystore slot is used, a crypto job may be sent to either board based on the board state. If one board is fully tasked, the job is sent to the other board. Also, if one board is not available due to a hardware failure, the job is sent to the other board.

With Keystore slot, both sensitive session keys and sensitive token keys are kept secure on the board. Thus, the secure key value is never revealed clear on the host memory. If the security of sensitive session keys are required, the Keystore slot is preferred over Sun Metaslot.

Sun Metaslot

The Sun Metaslot takes advantage of the board for cryptographic acceleration along with all other cryptographic providers available on the system. The Sun Metaslot uses the board for the mechanisms it supports, and it uses other slots, including the Oracle Solaris software implementation, for the mechanisms not supported by the board. The Sun Metaslot also supports failover. For more details, please refer to the Sun Metaslot documentation.

Configuring Sun Metaslot to Use the Sun Crypto Accelerator 6000 Board Keystore

Through Sun Metaslot, only one keystore can be accessed. By default Sun Metaslot uses the Oracle Solaris Softtoken keystore. To access the Sun Crypto Accelerator 6000 keystore through Sun Metaslot, you must use one of the following configurations.

- Configure Sun Metaslot to use the board keystore system-wide using `cryptoadm`(1M).

  Enter the following command to use the board keystore. For the example in this section, *ks* is the name of the board keystore.

```
% cryptoadm enable metaslot token=ks
```

  This command forces a global change throughout the system, which causes all applications on the system to use the board keystore by default.

- Configure Sun Metaslot to use the board keystore with an environment variable.

  Sun Metaslot can be configured to use the board's keystore on a per application basis by setting an environment variable. The variable should be set to the name of the board keystore.

```
% METASLOT_OBJECTSTORE_TOKEN=ks
% export METASLOT_OBJECTSTORE_TOKEN
```

  The environment variable overwrites the system-wide configuration.

Configuring Secure Failover for Sun Metaslot

Sun Metaslot supports fail over by automatically migrating keys from the board keystore to other slots. By doing so, the keys securely stored on the board might be revealed on the host memory. To protect the secure keys, enter the following command:

```
% cryptoadm disable metaslot auto-key-migrate
```

The auto-key migration can also be disabled on a per application basis by setting the following environment variable.

```
% METASLOT_AUTO_KEY_MIGRATE=false
% export METASLOT_AUTO_KEY_MIGRATE
```

When the auto key migration is disabled, sensitive token keys are not automatically migrated to other slots. With this configuration, if an operation with a sensitive token key fails on the board, the request does not failover to other slots, and the operation fails.

When this variable is not set, the sensitive token key is migrated to other slots that support the operation, and the request is processed in a failover slot. If the job fails over to a software slot, such as Sun Softtoken, the key could be revealed on the host memory.

Note – This configuration applies to the sensitive token keys only. Other keys, such as nonsensitive keys and sensitive session keys are still automatically migrated for failover.

To verify the current system-wide configuration, enter the following command:

```
% cryptoadm list -v metaslot
```

The following output shows that the Sun Metaslot is enabled, the automatic key migration is disabled, and the keystore slot, `ks`, is used for the persistent object store.

```
% cryptoadm list -v metaslot
System-wide Meta Slot Configuration:
------------------------------------
Status: enabled
Sensitive Token Object Automatic Migrate: disabled
Persistent object store token: ks

Detailed Meta Slot Information:
-------------------------------
actual status: enabled.
Description: Sun Metaslot

Token Present: True
Token Label: Sun Metaslot
Manufacturer ID: Sun Microsystems, Inc.
Model: 1.0
Serial Number:
Hardware Version: 0.0
Firmware Version: 0.0
UTC Time:
PIN Length: 0-253
Flags: CKF_RNG CKF_LOGIN_REQUIRED CKF_USER_PIN_INITIALIZED
CKF_TOKEN_INITIALIZED CKF_SO_PIN_LOCKED
```

Hardware Slot

When the board has not been initialized or when the board is in the diagnostic mode, the device can be directly accessed with the hardware slots. There are three hardware slots (CB, CA, and OM) per board.

- The CB Hardware slot accelerates pure bulk operations such as DES, 3DES, and AES. This slot supports session keys only.
- The CA Hardware slot accelerates asymmetric operations such as RSA, DSA, and DH.
- The OM Hardware slot allows key management operations such as key generation and key creation. However, the keys created on the OM slot cannot be used until the board is initialized and online.

Hardware slot is dedicated to a single board and thus does not allow hardware redundancy or load balancing. For a typical application, you might want to use either the Keystore slot or Sun Metaslot. The Hardware slot, however, is useful for diagnosis.

PKCS#11 and FIPS Mode

When put in FIPS mode by the SO (using `scamgr`), the board is compliant with Federal Information Processing Standard FIPS 140-2 level 3. Detailed information on FIPS 140-2 can be found at: `http://www.nist.gov/dmvp`

Operating the board in FIPS mode causes the following changes in the board's operation:

- Only FIPS-approved mechanisms are made available by the board itself.
- All keys and critical security parameters cross the PCI bus in encrypted form.
- Certain additional integrity checks are done at startup, and when keys and random numbers are generated.
- Random numbers are generated by a FIPS-approved algorithm that combines saved state and true random data (entropy) from a thermal-noise-based generator using hashing and arithmetic. 512 bits from the thermal-noise-based generator are used for every 160 bits of output data. (In non-FIPS mode, 512 bits from the thermal-noised-based generator are SHA-1 hashed to 160 bits.)

FIPS mode applies only to the board itself. As stated above, when the board is put in FIPS mode, only FIPS-approved mechanisms are provided by the board. Notably, MD5, and RC2 are not FIPS-approved.

However, because the FIPS regulations apply only to the hardware, software implementation of the non-FIPS-approved mechanisms will still be available through the Sun Metaslot.

Developing Applications to Use PKCS#11

The necessary header files are in `/usr/include/security`; add this directory to the include path and include `cryptoki.h`. The lower-level include files, `pkcs11.h`, `pkcs11f.h`, and `pkcs11t.h` are also available in the directory. These files are identical to those available at the PKCS#11 web site `http://www.rsasecurity.com/rsalabs/PKCS`.

The PKCS#11 libraries are `/usr/lib/libpkcs11.so` (32-bit mode) and `/usr/lib/sparcv9/libpkcs11.so` (64-bit mode).

The Oracle Solaris PKCS#11 library can be linked as an ordinary library, or it can be dynamically opened with `dlopen` (3DL).

When linking as an ordinary library, use the following command:

```
% cc [flags] files... -L /usr/lib -R /usr/lib -lpkcs11 [other libraries...]
```

Sun Crypto Accelerator 6000 Board PKCS#11 Implementation Specifics

The PKCS#11 administrative functions `C_InitToken` and `C_InitPin` are not implemented. The `C_Login` function with the `CKU_SO` (security officer) flag is rejected.

Token Objects

In PKCS#11, public token objects are token objects that are visible and deletable without authentication. Because the users known by the board software are unrelated to Oracle Solaris users, and because the software does not ascertain user identity until `C_Login` succeeds, these objects would need to be globally visible to all users, and therefore deletable by any user. Because this behavior is not acceptable, public token objects are not allowed. Any attempt to create a public token object will fail.

The number of session objects is limited by virtual memory only. Token objects must all fit in the RAM on the board, and the driver limits the size of the keystore to 16 Mbytes. However, the fields of the `CK_TOKEN_INFO` structure (returned by the

C_GetTokenInfo function) that indicate maximum memory sizes are all set to CK_EFFECTIVELY_INFINITE. The C_GetObjectSize function is not implemented.

Supported and Unsupported Functions

The optional dual operation functions (C_DigestEncryptUpdate, C_DecryptDigestUpdate, C_SignEncryptUpdate, and C_DecryptVerifyUpdate) are not implemented, and the CKF_DUAL_OPERATIONS_FLAG in the flags field returned by C_GetTokenInfo is false.

C_GetOperationState and its companion function C_SetOperationState are not supported.

Since the board can only operate SHA-1 and MD5 in a single part and the PKCS#11 interface requires both single part and multipart for the hash operations, CKM_SHA_1 and CKM_MD5 are not available from the user level of the PKCS#11 application. However, those mechanisms are available for the kernel consumers, such as IPsec.

The tokens provided by the board system are considered unremovable. Thus the CKF_REMOVABLE_DEVICE flag returned by CK_GetSlotInfo is false. However, the board can be dynamically reconfigured when there is no PKCS#11 application that has an active session on the board.

The C_WaitForSlotEvent function is not implemented, and the board system never calls the callback function passed as the Notify parameter to C_OpenSession. The software never surrenders control back to the calling application with the pApplication parameter of C_OpenSession.

Random Number Generator

The board contains a high-quality true random number generator. It does not need to be seeded, and in fact, C_SeedRandom will be rejected.

Software Attributes

The Sun Crypto Accelerator 6000 software defines the default values for some attributes as listed in the following table. Some permission flags such as CKA_LOCAL and CKA_ALWAYS_SENSITIVE are not implemented or enforced as noted.

TABLE 5-1 PKCS#11 Attributes and Default Values

Attribute	Value
CKA_AC_ISSUER	empty string
CKA_ALWAYS_SENSITIVE	always false
CKA_APPLICATION	empty string
CKA_ATTR_TYPES	empty string
CKA_AUTH_PIN_FLAGS	false
CKA_DECRYPT	true (not enforced)
CKA_DERIVE	false (not enforced)
CKA_ENCRYPT	true (not enforced)
CKA_END_DATE	empty string
CKA_EXTRACTABLE	true
CKA_HAS_RESET	false
CKA_ID	empty string
CKA_ISSUER	empty string
CKA_LABEL	empty string
CKA_LOCAL	always false
CKA_MODIFIABLE	true
CKA_NEVER_EXTRACTABLE	always false
CKA_OBJECT_ID	empty string
CKA_OWNER	empty string
CKA_PRIVATE	same as CKA_TOKEN
CKA_RESET_ON_INIT	false
CKA_SECONDARY_AUTH	false
CKA_SENSITIVE	opposite of CKA_EXTRACTABLE
CKA_SERIAL_NUMBER	empty string
CKA_SIGN	true (not enforced)
CKA_SIGN_RECOVER	true (not enforced)

TABLE 5-1 PKCS#11 Attributes and Default Values *(Continued)*

Attribute	Value
CKA_START_DATE	empty string
CKA_SUBJECT	empty string
CKA_TOKEN	false
CKA_TRUSTED	false
CKA_UNWRAP	true (not enforced)
CKA_VERIFY	true (not enforced)
CKA_VERIFY_RECOVER	true (not enforced)
CKA_WRAP	true (not enforced)

The CKA_TOKEN attribute defaults to false. The CKA_PRIVATE attribute defaults to the same value as CKA_TOKEN. An attempt to set both CKA_TOKEN and CKA_PRIVATE to false will fail since the board does not support public token objects.

The CKA_EXTRACTABLE attribute defaults to true. The CKA_SENSITIVE attribute defaults to the opposite of CKA_EXTRACTABLE. An attempt to set both CKA_SENSITIVE and CKA_EXTRACTABLE to false will fail with CKR_TEMPLATE_INCONSISTENT.

Inconsistent attributes are generally not detected. For example, even if CKA_VALUE_LENGTH is specified in the template when the CKK_DES key is created with C_CreteObject, Sun Crypto Accelerator 6000 software will not return an error code. The inconsistent attribute CKA_VALUE_LENGTH is simply ignored by the software.

Software Error Codes

The error codes returned by the software are not always as specific as what might be expected. In particular, CKR_MECHANISM_INVALID is returned for many errors where other values might seem more appropriate. The return code CKR_HOST_MEMORY usually means that an internal call to the malloc(3c) command failed. After this error is returned, an important state has probably not been properly saved, and attempting to continue, except by calling C_Finalize, could be ineffective.

The Mutex callback function pointers that can be passed to C_Initialize are ignored.

Token Object Handles

As required by the PKCS#11 standard, all token object handles become invalid when the user calls the `C_Logout` function or closes the last PKCS#11 session. The software purges the token objects from the software's cache. A subsequent successful `C_Login` function brings in all the then-current token objects. Note that this login could be for a different user and thus bring in a different set of token objects. However, even if this login is for the same user, the token objects might not get the same handles as they had before.

Building PKCS#11 Applications for Use With the Sun Crypto Accelerator 6000 Board on Linux Platforms

The openCryptoki software is used as the PKCS#11 framework. See Appendix B for details on openCryptoki software.

If the softtoken slot of the openCryptoki software is not installed on the system, you see only the Sun Crypto Accelerator 6000 (`SCA`) slot as follows:

```
Linux 2.6.5-7.139-smp Linux (SCA)
```

If the softtoken slot of the openCryptoki software is installed, you see the Sun Crypto Accelerator 6000 (`SCA`) slot first followed by the softtoken slot as follows:

```
Linux 2.6.5-7.139-smp Linux (SCA)
Linux 2.6.5-7.139-smp Linux (soft)
```

The first part of the slot description is from the operating system. The second part denotes whether it is the Sun Crypto Accelerator 6000 (`SCA`) slot or the softtoken slot (`soft`).

The Sun Crypto Accelerator 6000 (`SCA`) slot is a general PKCS#11 slot. The previous descriptions in this chapter are applicable to this slot.

CHAPTER 6

Installing and Configuring Sun Java System Application Server Software

This chapter describes how to configure the Sun Crypto Accelerator 6000 Board for use with Sun Java System servers. Additional instructions for Linux platforms are in the last section. Sections include:

Note – The Sun Java System servers described in this manual were previously named iPlanet Servers.

Note – All Sun Java System server software is supported for use with the board. The example in this section covers configuring the Sun Java System Web Server only. Refer to the Sun Java System documentation for details on how to install and configure Sun Java System server software.

Administering Security for Sun Java System Web Servers

This section provides an overview of the security features of the board as it is administered with Sun Java System applications.

Note – To manage keystores, you must have access to the system administrator account for your system.

Web Server Concepts and Terminology

Keystores and users must be created for applications that communicate with the board through a PKCS#11 interface, such as the Sun Java System Applications.

Note – The Apache Web Server (Chapter 7) does not use the keystore or user account features described in this chapter.

Users

Within the context of the board, users are owners of cryptographic keying material. Each key is owned by a single user. Each user may own multiple keys. A user might want to own multiple keys to support different configurations, such as a production key and a development key (to reflect the organizations the user is supporting).

Note – The term user or user account refers to Sun Crypto Accelerator 6000 users created in `scamgr`, not traditional UNIX user accounts. There is no fixed mapping between UNIX user names and Sun Crypto Accelerator 6000 user names.

Keystores

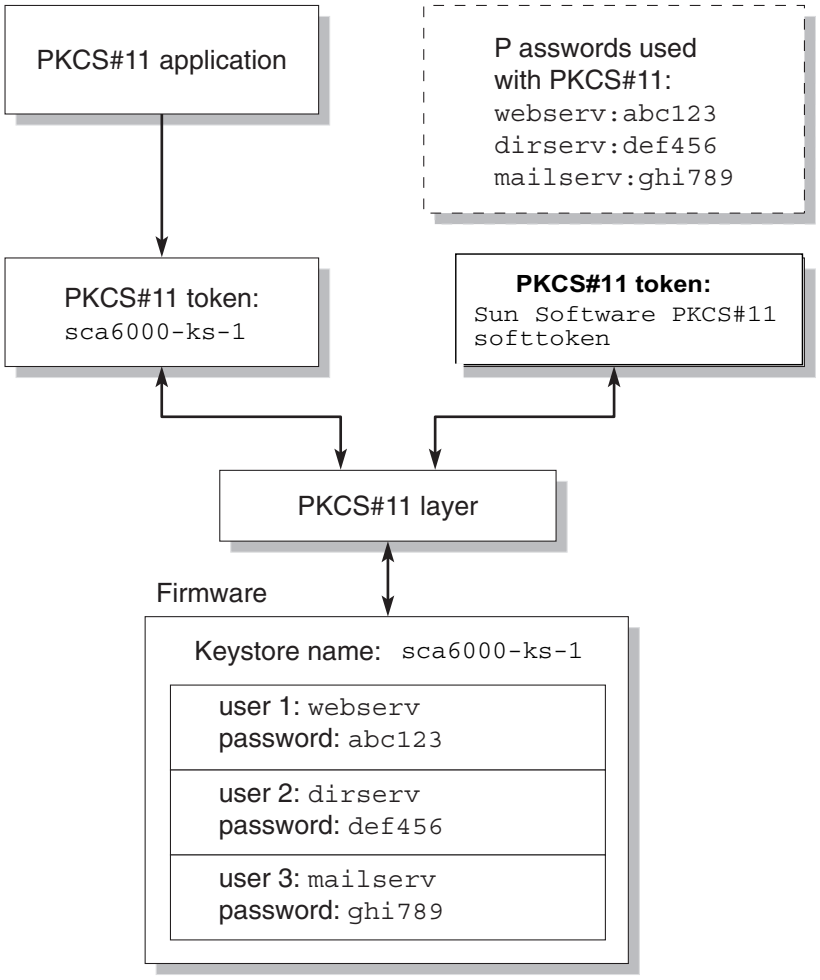
A keystore is a repository for key material. Associated with a keystore are security officers and users. Keystores provide not only storage, but a means for key objects to be owned by user accounts. This enables keys to be hidden from applications that do not authenticate as the owner. Keystores have three components:

- Key objects – Long-term keys that are stored for applications such as the Sun Java System Web Server.
- User accounts – Accounts that provide applications a means to authenticate and access specific keys
- Security officer accounts – Accounts that provide access to key management functions through `scamgr`.

Note – A single board must have exactly one keystore. Multiple boards can be configured to collectively work with the same keystore to provide additional performance and fault-tolerance.

A typical installation contains a single keystore with three users. For example, such a configuration could consist of a single keystore *keystore-name* and three users within that keystore, `webserv`, `dirserv`, and `mailserv`. This would enable the three users to own and maintain access control of their server keys within that single keystore. FIGURE 6-1 illustrates an overview of a typical installation.

FIGURE 6-1 Keystore and Users Overview



An administrative tool, `scamgr`, is used to manage Sun Crypto Accelerator 6000 keystores and users. See "Managing Keystores With `scamgr`" on page 43.

Slots and Tokens

As discussed in Chapter 5, there are four kinds of slots presented through the Oracle Solaris Cryptographic Framework's PKCS#11 interface.

The Sun Crypto Accelerator 6000 Keystore slot can also be used for Sun Java System applications. Through a Keystore slot, asymmetric operations are the only mechanisms accelerated by the Sun Crypto Accelerator 6000 board. When there are more than two boards using the same keystore, Keystore slot provides additional performance and fault-tolerance.

Example:

If there are two boards, `mca0` and `mca1`, each is assigned a keystore name (`engineering` and `finance`), three slots are presented to the Sun Java System application.

- `engineering`
- `finance`
- Sun Software PKCS#11 softtoken

If the server certificate resides in the `finance` keystore, the possible slots to be used for the Sun Java System application is as follows:

1. `metaslot`
2. `finance` (the Keystore slot)

Preparing to Configure Sun Java System Web Servers

This section describes assigning passwords, how to populate a keystore, and how to enable the Sun Java System Web Server.

You are asked for several passwords in the course of enabling a Sun Java System Web Server, all of which are described in TABLE 6-2. These passwords are referred to throughout this chapter.

TABLE 6-1 Passwords Required for Sun Java System Web Servers

Type of Password	Description
Sun Java System Web Server Administration Server	Required to start up the Sun Java System Web Server Administration Server. This password was assigned during the Sun Java System Web Server setup.
Web Server Trust Database	Required to start the internal cryptographic module when running in secure mode. This password was assigned when creating a trust database through the Sun Java System Web Server Administration Server. This password is also required when requesting and installing certificates into the internal cryptographic module.
Security Officer	Required when performing `scamgr` privileged operations.
username:password	Required to start the Sun Crypto Accelerator 6000 Board module when running in secure mode. This password is also required when requesting and installing certificates into the internal cryptographic module *keystore-name*. This password consists of the *username* and *password* of a keystore user that was created in `scamgr`. The keystore *username* and *password* are separated by a colon (:).

Populating a Keystore

Before you can enable the board for use with a Sun Java System Web Server, you must first initialize the board and populate the board's keystore with at least one user. The keystore for the board is created during the initialization process. You can also initialize Sun Crypto Accelerator 6000 boards to use an existing keystore. See "Initializing the Board With `scamgr`" on page 39.

Note – Only one keystore per Sun Crypto Accelerator 6000 board can be configured and you must configure one keystore per board. You can configure multiple Sun Crypto Accelerator 6000 boards to collectively work with the same keystore to provide additional performance and fault-tolerance.

▼ Populate a Keystore

1. **Access the `scamgr` utility with the `scamgr` command or enter `scamgr -h` *hostname* to connect `scamgr` to a board on a remote host.**

   See "Using the `scamgr` Utility" on page 30.

```
$ scamgr -h hostname
```

2. **Populate the board's keystore with users.**

These user names are known only within the domain of the Sun Crypto Accelerator 6000 board and do not need to be identical to the UNIX user name that the web server process is using. Before attempting to create the user, remember that you must first log in as a `scamgr` security officer.

3. **Create a user with the `create user` command.**

```
scamgr{mca0@hostname, sec-officer}> create user username
Initial password:
Confirm password:
User username created successfully.
```

The username and password created here collectively make the *username:password* (See TABLE 6-1). You must use this password when authenticating during a web server startup. This is the keystore password for a single user.

Caution – Users must remember this *username:password*. Without this password, users cannot access their keys. There is no way to retrieve a lost password.

4. **Exit `scamgr`.**

```
scamgr{mca0@hostname, sec-officer}> exit
```

Installing and Configuring Sun Java System Web Server 6.1

This section describes how to install and configure Sun Java System Web Server 6.1 to use the board. You must perform these procedures in order. Refer to the Sun Java System Web Server documentation for more information about installing and using Sun Java System Web Servers. This section includes the following procedures:

1. Install the Sun Java System Web Server.
2. Create a trust database.
3. Request a certificate.
4. Install the certificate.
5. Configure the Sun Java System Web Server.

Caution – These procedures must be followed in the order given. Failure to do so could result in an incorrect configuration.

Caution – The Sun Java System Web Server Administration Server must be up and running during the configuration process.

Note – The example in this section uses the Keystore slot.

▼ Install Sun Java System Web Server 6.1

1. **Download the Sun Java System Web Server 6.1 software.**

   You can find the web server software at the following URL:
   `http://www.sun.com`

2. **Change to the installation directory and extract the web server software.**

3. **Install the web server with the setup script from the command-line.**

   The default path name for the server is: `/opt/SUNWwbsvr/`.

   This chapter refers to the default paths. If you decide to install the software in a different location, be sure to note where you installed it.

```
% ./setup
```

4. **Answer the prompts from the installation script.**

   Except for the following prompts, you can accept the defaults:

   a. **Agree to accept the license terms by typing** `yes`**.**

   b. **Enter a fully qualified domain name.**

   c. **Enter the Sun Java System Web Server 6.1 Administration Server password twice.**

d. Press Return when prompted.

The Sun Java System Web Server Administration Server must be up and running during the configuration process.

▼ Create a Trust Database

1. **Start the Sun Java System Web Server 6.1 Administration Server.**

   Use the following command (instead of running `startconsole` as `setup` requests):

```
% /opt/SUNWwbsvr/https-admserv/start
Sun Java System Web Server 6.1 B08/22/2003 12:37
info: CORE3016: daemon is running as super-user
info: CORE5076: Using [Java HotSpot(TM) Server VM, Version
1.4.1_03] from [Sun Microsystems Inc.]
info: WEB0100: Loading web module in virtual server [vs-admin] at
[/admin-app]
info: HTTP3072: [LS ls1] http://hostname.domain:8888 ready to
accept requests
startup: server started successfully
```

   The response provides the URL for connecting to your servers.

2. **Start the Administration GUI by opening up a web browser and typing:**

```
http://hostname.domain:admin-port
```

   In the Authentication dialog box, enter the Sun Java System Web Server 6.1 Administration Server user name and password you selected while running `setup`.

Note – If you used the default settings during Sun Java System Web Server setup, enter `admin` for the User ID or the Sun Java System Web Server 6.1 Administration Server user name.

---

3. **Click OK.**

   The Sun Java System Web Server 6.1 Administration Server window is displayed.

4. **Create the trust database for the web server instance.**

   You might want to enable security on more than one web server instance. If so, repeat the following Step a through Step d for each web server instance.

Note – If you want to run SSL on the Sun Java System Web Server 6.1 Administration Server as well, the process of setting up a trust database is similar. Refer to the Sun Java System documentation at `http://docs.sun.com` for more information.

a. **Click the Servers tab in the Sun Java System Web Server 6.1 Administration Server dialog box.**

b. **Select a server and click the Manage button.**

c. **Click the Security tab near the top of the page and click the Create Database link.**

d. **Enter a password in the two dialog boxes and click OK.**

See TABLE 6-1 **for the web server trust database password information.**

Choose a password of at least eight characters. This will be the password used to start the internal cryptographic modules when the Sun Java System Web Server runs in secure mode.

▼ Register the Board With the Web Server

1. **Register the Oracle Solaris PKCS#11 library in the security module database of the Sun Java System Web Server using the `modutil` utility.**

Note – `modutil` is a utility developed by Mozilla and is available with the Sun Java System distribution. By default, the `modutil` is located at `/opt/SUNWwbsvr/bin/https/admin/bin` directory. It uses the NSS libraries located at `/opt/SUNWwbsvr/bin/https/lib`, and should be included in the environment variable, `$LD_LIBRARY_PATH`.

```
% modutil -dbdir /opt/SUNWwbsvr/alias -nocertdb -add "Solaris
Cryptographic Framework" -libfile /usr/lib/libpkcs11.so
```

2. **To limit the slots presented to those required to start the web server, disable all slots, except for one slot used by the Sun Java System application.**

If the application asks for a password for every known PKCS#11 token, do not provide one.

```
% modutil -dbdir /opt/SUNWwbsvr/alias -nocertdb -disable "Solaris Cryptographic
Framework"
% modutil -dbdir /opt/SUNWwbsvr/alias -nocertdb -enable "Solaris Cryptographic
Framework" -slot "keystore-name"
```

▼ Generate a Server Certificate

1. **Restart the Sun Java System Web Server 6.1 Administration Server by typing the following commands:**

```
% /opt/SUNWwbsvr/https-admserv/stop
% /opt/SUNWwbsvr/https-admserv/start
```

   The response provides the URL for connecting to your servers.

2. **Start the Administration GUI by opening up a web browser and typing:**

```
http://hostname.domain:admin-port
```

   In the Authentication dialog box, enter the Sun Java System Web Server 6.1 Administration Server user name and password you selected while running `setup`.

Note – If you used the default settings during Sun Java System Web Server setup, enter **`admin`** for the user ID or the Sun Java System Web Server 6.1 Administration Server user name.

---

3. **Click OK.**

   The Sun Java System Web Server 6.1 Administration Server window is displayed.

4. **To request the server certificate, select the Servers tab near the top of Sun Java System Web Server 6.1 Administration Server window.**

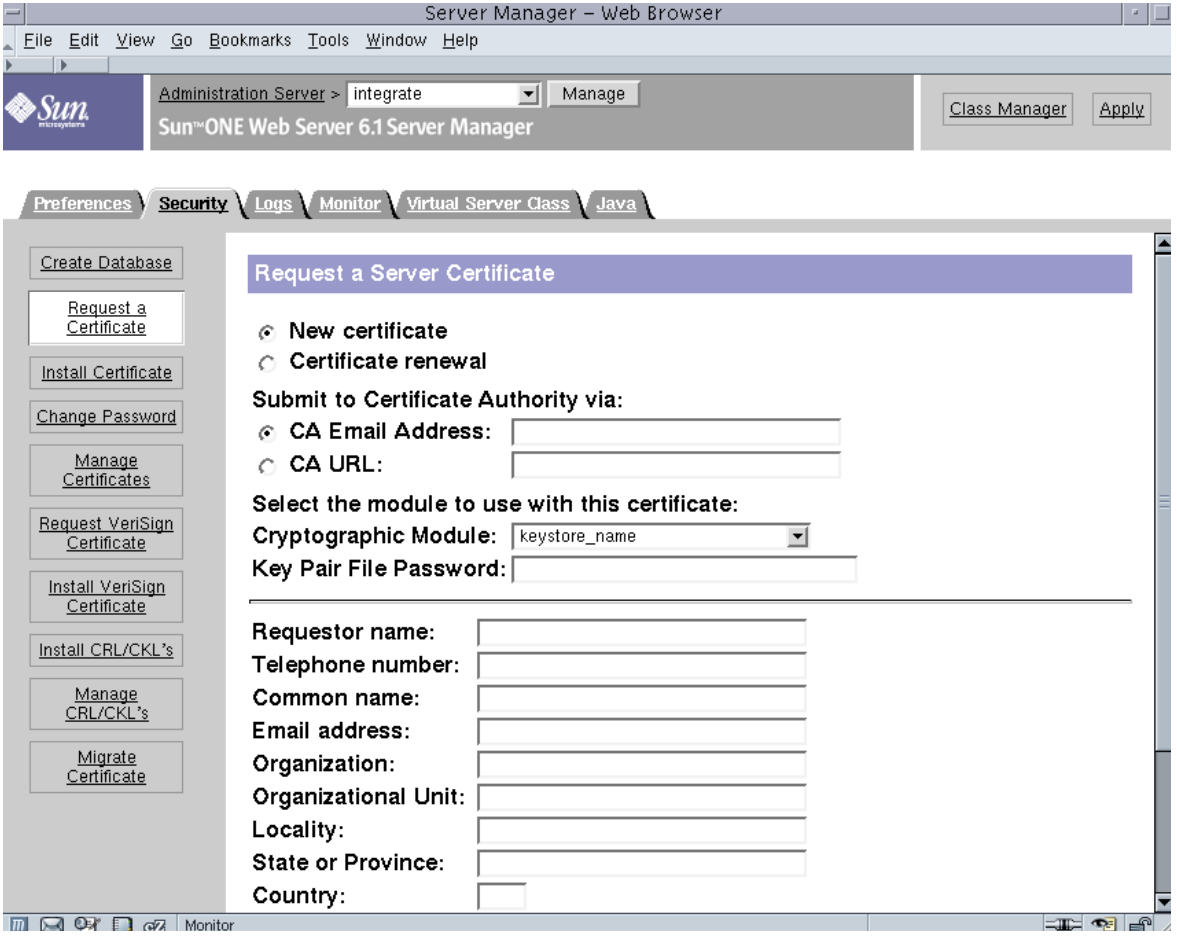
5. **Select a server from the drop-down menu and click the Manage button.**

   The Sun Java System Web Server 6.1 Server Manager window is displayed.

6. **Select the Security tab near the top of the Sun Java System Web Server 6.1 Server Manager window.**

7. **Click the Request a Certificate link on the left panel.**

FIGURE 6-2 Sun Java System Web Server 6.1 Administration Server Request a Server Certificate Dialog Box With *keystore-name* Selected



8. **Fill out the form to generate a certificate request, using the following information:**

   a. **Select a New Certificate.**

      If you can directly post your certificate request to a web-capable certificate authority or registration authority, select the CA URL link. Otherwise, select CA Email Address and enter an email address where you would like the certificate request to be sent.

   b. **Select the Cryptographic Module you want to use.**

      Each slot has its own entry in this pull-down menu. For this example, the *keystore-name* is chosen.

c. **In the Key Pair File Password dialog box, provide the password for the user that will own the key.**

   This password is the *username:password* (See TABLE 6-1).

d. **Type the appropriate information for the requestor information fields in TABLE 6-2.**

TABLE 6-2 Requestor Information Fields

Field	Description
Requestor Name	Contact information for the requestor
Telephone Number	Contact information for the requestor
Common Name	Web site domain that is typed in a visitor's browser
Email Address	Contact information for the requestor
Organization	Company name
Organizational Unit	(Optional) Department of the company
Locality	(Optional) City, county, principality, or country
State	(Optional) Full name of the state
Country	Two-letter ISO code for the country (for example, the United States is US)

e. **Click OK to submit the information.**

9. **Use a certificate authority to generate the certificate.**

- If you choose to post your certificate request to a CA URL, the certificate request is automatically posted there.
- If you choose the CA Email Address, copy the certificate request that was emailed to you with the headers and hand it off to your certificate authority.

10. **Once the certificate is generated, copy it, along with the headers, to the clipboard.**

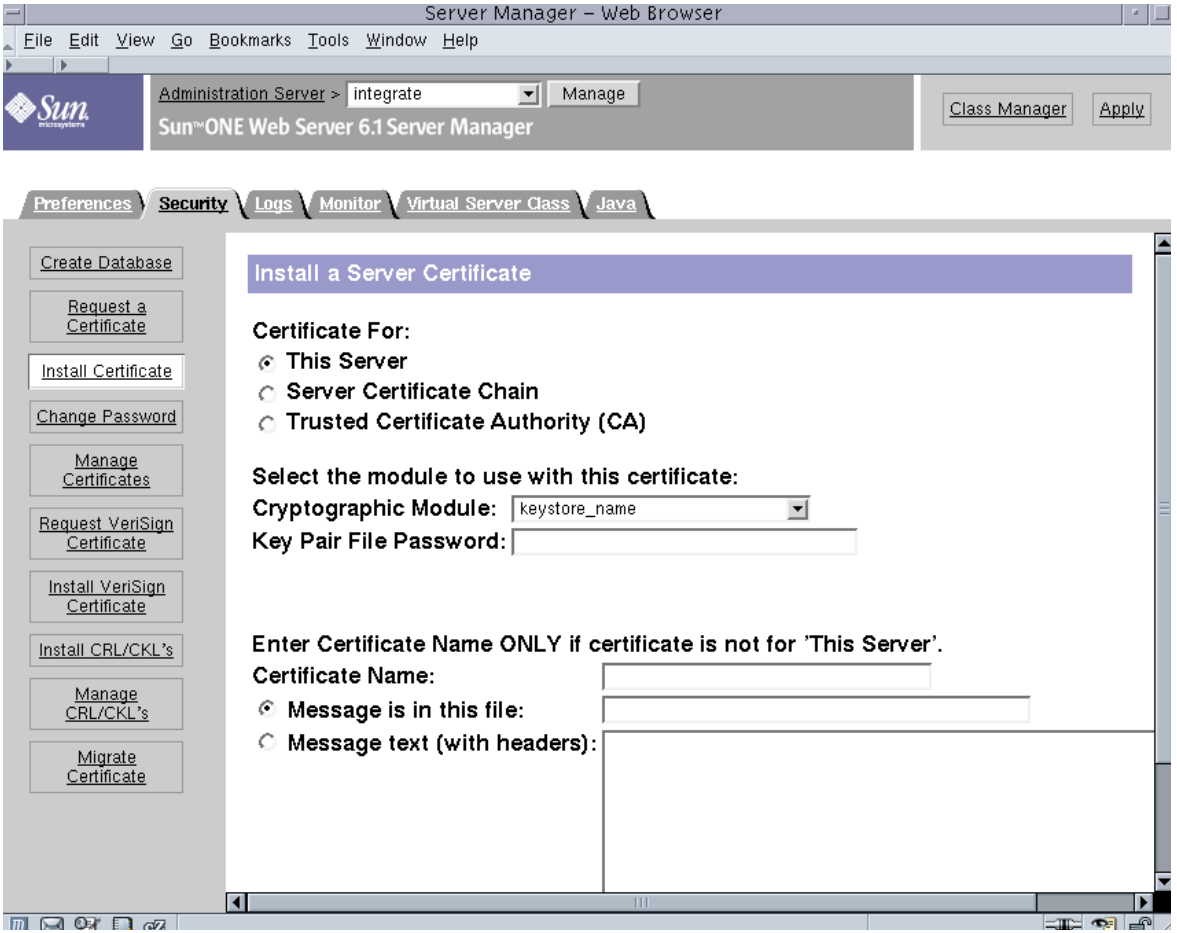
Note – The certificate is different from the certificate request and is usually presented to you in text form. Keep this data on the clipboard for Step 4 of "Install the Server Certificate" on page 122.

▼ Install the Server Certificate

Once your request has been approved by a certificate authority and a certificate has been issued, you must install the certificate in the Sun Java System Web Server.

1. **Click the Security tab near the top of the Sun Java System Web Server 6.1 Server Manager window.**
2. **On the left panel, click the Install Certificate link.**

FIGURE 6-3 Sun Java System Web Server 6.1 Administration Server Install a Server Certificate Dialog Box



3. **Fill out the form to install your certificate:**

TABLE 6-3 Fields for the Certificate to Install

Fields	Description
Certificate For	This server
Cryptographic Module	Each slot has its own entry in this pull-down menu. Ensure that you select the correct slot name. For this example, use *keystore-name*.
Key Pair File Password	This password is the *username:password* (TABLE 6-1)
Certificate Name	In most cases, you can leave this blank. If you provide a name, it alters the name the web server uses to access the certificate and key when running with SSL support. The default for this field is `Server-Cert`.

4. **Paste the certificate you copied from the certificate authority (in Step 10 of the "Generate a Server Certificate" on page 119) into the Message text box.**

   You are shown some basic information about the certificate.

5. **Click OK.**

6. **If everything looks correct, click the Add Server Certificate button.**

   On-screen messages tell you to restart the server. This is not necessary because the web server instance has been shut down the entire time.

You are also notified that in order for the web server to use SSL, the web server must be configured to do so. Use the following procedure to configure the web server.

Now that your web server and the Server Certificate are installed, you must enable the web server for SSL.

▼ Enable the Web Server for SSL

1. **Select the Preferences tab near the top of the page.**

2. **Select the Edit Listen Sockets link on the left panel.**

   The main panel lists all the listen sockets set for the web server instance.

   a. **Click the link under Listen Socket ID for the listen socket you wish to configure.**

   b. **Alter the following fields:**

   - **Port** – Set to the port on which you will be running your SSL-enabled web server (usually this is port 443).

- **Security –** Set to Enabled.

c. Click OK to apply these changes.

3. **Click the link under Listen Socket ID again for the listen socket you wish to configure.**
4. **Enter the** *username:password* **to authenticate to the keystore on the system.**
5. **If you want to change the default set of ciphers, select the cipher suites under the Ciphers heading.**

   A dialog box is displayed for changing the cipher settings. You can select either Cipher Default settings, SSL2, or SSL3/TLS. If you select the Cipher Default, you are not shown the default settings. The other two choices require you to select the algorithms you want to enable in a pop-up dialog box. Refer to your Sun Java System documentation on cipher selection.

6. **Select the certificate for the keystore followed by**: `Server-Cert` **(or the name you chose).**

   Only keys that the appropriate keystore user owns appear in the Certificate Name field. This keystore user is the user that is authenticated with the *username:password*.

7. **When you have chosen a certificate and confirmed all the security settings, click OK.**
8. **Select the Apply link in the far upper right corner to apply these changes before you start your server.**
9. **Select the Load Configuration Files link to apply the changes.**

   You are redirected to a page that allows you to start your web server instance.

   If you click the Apply Changes button when the server is off, an authentication dialog box prompts you for the *username:password*. This window is not resizable, and you might have a problem submitting the change.

   There are two workarounds for this problem:

- Select Load Configuration Files instead.
- Start up the web server first, and click Apply Changes.

10. **In the Sun Java System Web Server 6.1 Administration Server window, select the On/Off link on the left side of the window.**
11. **Enter the passwords for the servers and click Server On.**

    You are prompted for one or more passwords. At the Module Internal prompt, provide the password for the web server trust database.

At the Module *keystore-name* prompt, enter the *username:password*.

Enter the *username:password* for other keystores as prompted.

12. **Verify the new SSL-enabled web server at the following URL:**

    `https://`*hostname.domain:server-port*`/`

Note – The default *server-port* is 443.

Configuring Sun Java System Web Servers to Start Up Without User Interaction on Reboot

You can enable the Sun Java System Web Servers to perform an unattended startup at reboot with an encrypted key.

▼ Create an Encrypted Key for Automatic Startup of Sun Java System Web Servers on Reboot

1. **Navigate to the `config` subdirectory for your Sun Java System Web Server instance – for example,** `/opt/SUNWwbsvr/https-webserver-instance-name/config`**.**

2. **Create a** `password.conf` **file with only the following lines (See** TABLE 6-1 **for password definitions):**

```
internal:trust-db-password
token-label:username:password
```

3. **Set the file ownership of the password file to the UNIX user ID that the web server runs as, and set the file permissions to be readable only by the owner of the file:**

```
# chown web-server-UNIX-user-ID password.conf
# chmod 400 password.conf
```

Installing and Configuring Sun Java System Web Server on Linux Platforms

The Sun Java System Web Server for Red Hat Enterprise Linux 4.0 is downloadable at: `http://www.sun.com/download/products`. The latest release is 6.1 Service Pack 5. Both RHEL 4.0 and SuSE 9 are supported with the Sun Java Web Server software.

The installation and configuration of Sun Java System Web Server on Linux is similar to that on Oracle Solaris. The only difference is the registration of the board with the web server. Refer to "Register the Board With the Web Server" on page 118 in this chapter for details.

On Linux platforms, the PKCS#11 library is `/usr/local/lib/libopencryptoki.so`. Thus, use the following command to register the board with the Sun Java Web Server:

```
% modutil -dbdir /opt/SUNWwbsvr/alias -nocertdb -add
"openCryptoki" -libfile /usr/local/lib/libopencryptoki.so
```

Use the following commands to disable the openCryptoki slots other than the Sun Crypto Accelerator 6000 slot:

```
% modutil -dbdir /opt/SUNWwbsvr/alias -nocertdb -disable
"openCryptoki"
% modutil -dbdir /opt/SUNWwbsvr/alias -nocertdb -enable
"openCryptoki" -slot "slot-name"
```

For example, for SuSE 9 SP1 the "*slot-name*" is as follows:

```
"Linux 2.6.5-7.139-smp Linux (SCA)"
```

Use the following command to check whether the other slots are disabled:

```
% modutil -dbdir /opt/SUNWwbsvr/alias -nocertdb -list
"openCryptoki"
```

The output of this command should be similar to the following:

```
Using database directory /opt/SUNWwbsvr/alias...
-----------------------------------------------------------
Name: openCryptoki
Library file: /usr/local/lib/libopencryptoki.so
Manufacturer: IBM
Description: Meta PKCS11 LIBRARY
PKCS #11 Version 2.11
Library Version: 2.2
Cipher Enable Flags: None
Default Mechanism Flags: None

  Slot: Linux 2.6.5-7.139-smp Linux (SCA)
  Slot Mechanism Flags: None
  Manufacturer: Linux 2.6.5-7.139-smp
  Type: Hardware
  Version Number: 0.0
  Firmware Version: 1.1
  Status: Enabled
  Token Name: apisclan1
  Token Manufacturer: SUNWmca
  Token Model: sca6000
  Token Serial Number:
  Token Version: 0.0
  Token Firmware Version: 0.0
  Access: NOT Write Protected
  Login Type: Login required
  User Pin: Initialized

  Slot: Linux 2.6.5-7.139-smp Linux (Soft)
  Slot Mechanism Flags: None
  Manufacturer: Linux 2.6.5-7.139-smp
  Type: Software
  Version Number: 0.0
  Firmware Version: 1.1
  Status: DISABLED (user disabled)
  Token Name: IBM OS PKCS#11
  Token Manufacturer: IBM Corp.
  Token Model: IBM SoftTok
  Token Serial Number: 123
  Token Version: 1.0
  Token Firmware Version: 1.0
  Access: NOT Write Protected
  Login Type: Login required
  User Pin: Initialized
-----------------------------------------------------------
```

Notice the `Status: Enabled` for `Linux 2.6.5-7.139-smp Linux (SCA)` slot and `Status: DISABLED (user disabled)` for `Linux 2.6.5-7.139-smp Linux (Soft)`. If the `Linux 2.6.5-7.139-smp Linux (Soft)` slot is not disabled (Java System Web Server 6.1 SP4 and SP5 might have this behavior), do the following to remove the `Linux 2.6.5-7.139-smp Linux (Soft)` slot:

```
% cd /usr/local/lib/opencryptoki/stdll
% mkdir tmp
% mv libpkcs11_sw.* PKCS11_SW.so tmp
```

CHAPTER 7

Installing and Configuring Apache Web Server Software

This chapter explains how to configure and enable the Sun Crypto Accelerator 6000 Board for use with Apache Web Servers on both Oracle Solaris and Linux platforms. Sections include:

Installing and Configuring Apache Web Server on Oracle Solaris Platforms

This section provides instructions specific to Oracle Solaris platforms.

Creating a Private Key and Certificate

The following procedure describes how to create the private key and certificate required to enable Apache Web Servers to use the Sun Crypto Accelerator 6000 Board. If you already have a private key and certificate, go to "Enabling Apache Web Server" on page 131.

▼ Create a Private Key and Certificate

1. Generate an RSA private key in Privacy-Enhanced Mail (PEM) format.

```
% ./openssl genrsa -des3 -out  /usr/local/apache2/conf/server.key 1024
```

2. Create your PEM passphrase.

This passphrase protects the key material. Be sure to select a strong passphrase, but one that you can remember. If you forget the passphrase, you will be unable to access your keys.

```
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
```

Caution – You must remember the passphrase you enter. Without the passphrase, you cannot access your keys. There is no way to retrieve a lost passphrase.

3. Create a certificate request using the keys you just created.

```
% ./openssl req -new -key /usr/local/apache2/conf/server.key  -out /crtreq.csr
```

You must first enter the passphrase to access your keys. Then provide the appropriate information for the fields in TABLE 7-1:

TABLE 7-1 Certificate Field Descriptions

Certificate Field	Description
Country Name	The two-letter ISO code for the country, which is asserted on the certificate and is a required field (for example, the United States is US).
State or Province Name	(Optional) The full name of the state in this field (or type "." and press Return).
Locality	(Optional) City, county, principality, or country, which is also asserted on the certificate if provided.

TABLE 7-1 Certificate Field Descriptions *(Continued)*

Certificate Field	Description
Organization Name	A value for the Organization to be asserted on the certificate.
Organizational Unit Name	(Optional) A value for the Organizational Unit that will be asserted on the certificate.
SSL Server Name	Web site Domain that is typed in a visitor's browser.
Email Address	Contact information for requestor.

The following is an example of how the certificate fields are entered:

```
Enter PEM pass phrase:
You are about to be asked to enter information that will be incorporated into
your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:US
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:.
Organization Name (eg, company) []:Fictional Company, Inc.
Organizational Unit Name (eg, section) []:Online Sales Division
Common Name (eg, YOUR name) []:www.fictional-company.com
Email Address []:admin@fictional-company.com

Please enter the following 'extra' attributes to be sent with your certificate
request
A challenge password []:
An optional company name []: Fictional Comany, Inc.
```

4. Hand off the `certreq.csr` file to your certificate authority.

Enabling Apache Web Server

Apache Web Server and `mod_ssl` are provided with the Oracle Solaris 10 OS. The following instructions are for these specific releases of Apache Web Server. Refer to the Apache Web Server documentation for more information.

▼ Enable the Apache Web Server

1. **Create an `httpd` configuration file.**

   For Oracle Solaris systems, the `httpd.conf-example` file is usually in `/etc/apache`. You can use this file as a template and copy it as follows:

```
% cp /etc/apache/httpd.conf-example /etc/apache/httpd.conf
```

2. **Replace `ServerName` with your server name in the `http.conf` file.**

3. **Find you private key and certificate.**

- If you have a private key and certificate, go to Step 4.
- If you do not have a private key and certificate, go to "Creating a Private Key and Certificate" on page 130.

4. **Rename the private key as `server.key` and place it in the `/etc/apache/ssl.key` directory.**

5. **Rename the private certificate as `server.crt` and place it in the `/etc/apache/ssl.crt` directory.**

6. **Start the Apache Web Server.**

   This example assumes the Apache binary directory is `/usr/apache/bin`. If this is not the Apache binary directory, type in the correct directory.

```
% /usr/apache/bin/apachectl startssl
```

7. **Enter you PEM passphrase if prompted for it.**

8. **Verify the SSL enabled web server with a browser pointing to the following URL:**

```
https://ServerName:ServerPort/
```

Note – The default port is 443.

---

9. **Verify that the Sun Crypto Accelerator 6000 board is being used.**

```
% kstat -n sca0
```

   Verify that the `rsaprivate` field is being incremented in the statistics.

Installing and Configuring Apache Web Server on Linux Platforms

The Apache web server included in the Linux installation does not have the appropriate plugins. This section describes how to prepares the Apache Web Server with appropriate plugins to use the Sun Crypto Accelerator 6000 board for SSL acceleration.

Note – On Oracle Solaris platforms, the OpenSSL executable is in the `/usr/sfw/bin/` directory. On Linux platforms, the OpenSSL executable is in the `/usr/bin/` directory.

Preparing OpenSSL Libraries

Download the following files from the OpenSSL web site:

- `http://www.openssl.org/source/openssl-0.9.7d.tar.gz`
- `http://www.openssl.org/contrib/pkcs11_engine-0.9.7d.patch.2006-04-17.gz`

Choose a directory to uncompress the OpenSSL software (`/var/tmp/` is used in this example). Type the following command:

```
% tar -zxvf openssl-0.9.7d.tar.gz
% gunzip pkcs11_engine-0.9.7d.patch.2006-04-17.gz
```

Change to the new `/var/tmp/openssl-0.9.7d` directory and install the patch with the following command:

```
% patch -p1 < ../pkcs11_engine-0.9.7d.patch.2006-04-17
```

The following is an example of the output:

```
patching file Configure
patching file Makefile.org
patching file README.pkcs11
patching file crypto/engine/Makefile.ssl
patching file crypto/engine/cryptoki.h
patching file crypto/engine/eng_all.c
patching file crypto/engine/engine.h
patching file crypto/engine/hw.ec
patching file crypto/engine/hw_pk11.c
patching file crypto/engine/hw_pk11_err.c
patching file crypto/engine/hw_pk11_err.h
patching file crypto/engine/hw_pk11_pub.c
patching file crypto/engine/pkcs11.h
patching file crypto/engine/pkcs11f.h
patching file crypto/engine/pkcs11t.h
```

Note – Check the *Sun Crypto Accelerator 6000 Board Product Notes for Version 1.0* for any additional required patches. You must install all of the required patches before configuring OpenSSL.

Use the following command to configure and compile OpenSSL. Refer to the `README.pkcs11` and `INSTALL` file for more information.

```
% ./config --pk11-libname=/usr/lib64/pkcs11/PKCS11_API.so
% make
```

Compiling Apache Web Server

Download Apache 2.2.0, `httpd-2.2.0.tar.gz`, from `http://www.apache.org`. Choose a directory to uncompress the Apache software (`/var/tmp` is used in this example). Type the following command:

```
% tar -zxvf httpd-2.2.0.tar.gz
```

Change to the new `/var/tmp/httpd-2.2.0` directory and type the following command to configure the Apache Web Server. Refer to the `INSTALL` file for more information.

```
% ./configure --enable-ssl --with-ssl=/var/tmp/openssl-0.9.7d
```

There are many other options to configure Apache. The `--enable-ssl --with-ssl=/var/tmp/openssl-0.9.7d` options are the minimum required. These options provide the location of the OpenSSL libraries.

Finally, use the following commands to compile and install Apache. Refer to the `INSTALL` file for more information:

```
% make
% make install
```

Note – Using Apache 2.2.0 or 2.2.2 on SuSE with the x86_x64 architecture, `make` could fail with an error message similar to the following: `/usr/lib/libexpat.la: could not read symbols: Invalid operation` If this error occurs, change the `/usr/lib/libexpat.la` entry to `/usr/lib64/libexpat.la` in the `srclib/apr-util/Makefile`.

By default, Apache is installed in the `/usr/local/apache2` directory.

Configuring and Starting Apache Web Server

The Apache software is installed in the `/usr/local/apache2` directory in this example.

Edit the `/usr/local/apache2/conf/httpd.conf` file and change the following line to enable SSL:

```
#Include conf/extra/httpd-ssl.conf
```

to:

```
Include conf/extra/httpd-ssl.conf
```

To enable the PKCS#11 OpenSSL engine, edit the `/usr/local/apache2/conf/extra/httpd-ssl.conf` file to add the following line:

```
SSLCryptoDevice pkcs11
```

just before the following line:

```
#   Pass Phrase Dialog:
```

In the same file, also change the following line:

```
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
```

to:

```
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL:
!DHE-RSA-AES256-SHA:!DHE-DSS-AES256-SHA:!AES256-SHA:!DHE-RSA-
AES128-SHA:!DHE-DSS-AES128-SHA:!RSA-AES128-SHA
```

This change eliminates the strong ciphers that do not work well with OpenSSL. Save the change and exit editing.

Prepare a certificate request and a certificate as described in the previous sections of this chapter.

Note – Use `/usr/bin/openssl` for the OpenSSL command, `/usr/local/apache2/conf/server.key` and `/usr/local/apache2/conf/server.crt` for the key and certificate files for Apache 2.x.

Put the private key in the `/usr/local/apache2/conf/server.key` file and the certificate in the `/usr/local/apache2/conf/server.crt` file.

Use the following command to start the Apache Web Server:

```
% /usr/local/apache2/bin/apachectl start
```

Note – Apache could fail to start with an error message similar to the following:
`Syntax error on line 52 of /usr/local/apache2/conf/extra/httpd-ssl.conf: SSLCryptoDevice: Invalid argument; must be one of: 'builtin' (none), ...`
If this error occurs, verify that:
`--pk11-libname=/usr/lib64/pkcs11/PKCS11_API.so` is used for the OpenSSL configuration and also that `/usr/lib64/pkcs11/PKCS11_API.so` is a link to the 64-bit openCryptoki PKCS#11 library with the `file` command.

Test the Apache Web Server as described in the previous sections of this chapter. Verify that the Sun Crypto Accelerator 6000 board is being used with the following command:

```
% cat /proc/driver/mca0
```

Verify that the `rsaprivate` field is being incremented in the statistics.

CHAPTER 8

Diagnostics and Troubleshooting

This chapter describes diagnostic tests and troubleshooting for the Sun Crypto Accelerator 6000 Board software. Additional instructions for Linux are in the last section. Sections include:

Diagnostic Software

The Sun Crypto Accelerator 6000 software provides three interactive utilities for running diagnostics on the board. The first of these utilities, SunVTS, focuses on the system-level network and cryptographic functionality of the Sun Crypto Accelerator 6000 subsystem (driver, firmware, and hardware). The other two utilities, `scamgr` and `scadiag`, perform low-level diagnostics on individual hardware components of the Sun Crypto Accelerator 6000 board.

Performing SunVTS Diagnostics

SunVTS is Sun Validation Test Suite software. The core SunVTS wrapper provides test control and a user interface to a suite of system level tests. These tests are delivered with packages `SUNWvts` and `SUNWvtsts` to make up a bundle that is contained on the Oracle Solaris 10 Software DVDs, and also available for download at: `http://www.sun.com/oem/vts`

The Sun Crypto Accelerator 6000 board can be tested with SunVTS 6.2 software that is released with the Oracle Solaris 10 6/06 OS. The SunVTS test, `cryptotest`, provides diagnostics of the cryptographic circuitry of the board.

Refer to the SunVTS 6.2 test reference manuals (x86 or SPARC), user's guide, and quick reference card for instructions on how to perform and monitor this diagnostic test. These documents are available at: `http://docs.sun.com`.

Performing `scamgr` Diagnostics

The `scamgr` utility is used by a security officer to test an initialized card and is the recommended interactive diagnostic application. Both `scamgr` and `scadiag` invoke the same diagnostics routines on the card, but the `scamgr` utility provides more information regarding any failures encountered. Details on how to run the `scamgr` utility are provided in Chapter 3 of this document, and an example of how to run diagnostics using `scamgr` is provided in "Use the `scamgr diagnostics` Command" on page 62.

Performing `scadiag` Diagnostics

The `scadiag` interface allows the security administrator to perform diagnostics on both an initialized and uninitialized board. The `scadiag` interface provides less information regarding diagnostic failures then the `scamgr` interface and is primarily intended to provide a general pass/fail status to someone other than a board security officer. To run `scadiag` diagnostics, the user invokes the `scadiag` command with the `-D` parameter. Details on how to run the `scadiag` utility are provided in Chapter 3, and an example of how to run diagnostics using `scadiag` is provided in "Using the `scadiag` Utility" on page 63.

Disabling Crypto Traffic on Other Hardware Providers in Your System

Sun Metaslot chooses the first hardware slot available in the system for crypto operations. For a system with a crypto chip built into the main CPU, such as the Sun Fire T1000/T2000, the crypto chip often becomes the first hardware slot. In this case, most crypto jobs except for the sensitive token key operation are sent to that crypto chip until the main CPU becomes 100 percent utilized. To avoid this congestion, such

hardware providers can be disabled with the `cryptoadm`(1M) utility. This utility can also direct Sun Metaslot to use the Sun Crypto Accelerator 6000 board for all crypto operations.

▼ Disable Other Hardware Providers

- **Type the following command:**

```
% cryptoadm disable provider=provider-name mechanism=all
```

Use the `kstat`(1M) command to verify that the crypto jobs are being processed by the Sun Crypto Accelerator 6000 board.

▼ Reenable Other Hardware Providers

- **Type the following command:**

```
% cryptoadm enable provider=provider-name mechanism=all
```

Refer to `cryptoadm`(1M) man page for details.

Using the `kstat` Utility

The `kstat`(1m) utility examines and reports available kernel statistics. The following is an example of using the `kstat` utility with the board:

```
root@cas1# kstat -n mca1
module: mca                             instance: 1
name:   mca1                            class:    misc
        3desbytes                       0
        3desjobs                        7
        aesbytes                        0
        aesjobs                         0
        caflowctl                       0
        cahiwater                       124
        calowater                       123
        caringsize                      132
        casubmit                        27
        cbflowctl                       0
```

```
        cbhiwater                       124
        cblowater                       123
        cbringsize                      132
        cbsubmit                        0
        crtime                          158.452327536
        dhderive                        0
        dhkeygen                        0
        dsasign                         0
        dsaverify                       0
        fsbytes                         0
        fsjobs                          0
        keygenjobs                      0
        md5bytes                        0
        md5jobs                         0
        mode                            standard
        omflowctl                       0
        omhiwater                       124
        omlowater                       123
        omringsize                      132
        omsubmit                        7
        rngbytes                        540
        rngjobs                         27
        rsaprivate                      0
        rsapublic                       0
        sha1bytes                       0
        sha1jobs                        0
        snaptime                        77620.93607806
        status                          online
        unwrapjobs                      0
        wrapjobs                        0
```

Determining Cryptographic Activity With the `kstat` Utility

The Sun Crypto Accelerator 6000 board does not contain lights or other indicators to reflect cryptographic activity on the board. To determine whether cryptographic work requests are being performed on the board, use the `kstat`(1M) command to display the device usage. The following excerpt shows the various `kstat` options that can be used to determine cryptographic activity.

Note – The following output has noncryptographic activity omitted.

```
# kstat mca:0
module: mca                             instance: 0
name:   mca0                            class:    misc

3desbytes                               0
3desjobs                                7
aesbytes                                32
aesjobs                                 1
rsaprivate                              0
rsapublic                               1
dsasign                                 0
dsaverify                               0
dhderive                                0
dhkeygen                                0
md5bytes                                0
md5jobs                                 0
sha1bytes                               0
sha1jobs                                0
fsbytes                                 0
fsjobs                                  0
rngbytes                                60
rngjobs                                 3
keygenjobs                              0
wrapjobs                                0
unwrapjobs                              0
```

Note – In the previous example, `0` is the instance number of the `mca` device. This number should reflect the instance number of the board for which you are performing the `kstat` command.

Displaying the `kstat` information indicates whether cryptographic requests or "jobs" are being sent to the Sun Crypto Accelerator 6000 Board. A change in the *jobs* values over time indicates that the board is accelerating cryptographic work requests sent to the Sun Crypto Accelerator 6000 Board. If cryptographic work requests are not being sent to the board, verify your web server configuration per the web server specific configuration.

Determining Cryptographic Activity On Linux Platforms

The Sun Crypto Accelerator 6000 board does not contain lights or other indicators to reflect cryptographic activity on the board. To determine whether cryptographic work requests are being performed on the board, you must use the `/proc` file system

▼ Determine Cryptographic Activity On Linux Platforms

- **Use the following command to display the device usage:**

```
% cat /proc/driver/mca0
```

The following excerpt shows the various statistics that can be used to determine cryptographic activity:

```
3desbytes                   0
3desjobs                    7
aesbytes                    32
aesjobs                     1
rsaprivate                  0
rsapublic                   1
dsasign                     0
dsaverify                   0
dhderive                    0
dhkeygen                    0
md5bytes                    0
md5jobs                     0
sha1bytes                   0
sha1jobs                    0
fsbytes                     0
fsjobs                      0
rngbytes                    60
rngjobs                     3
keygenjobs                  0
wrapjobs                    0
unwrapjobs                  0

mode                        FIPS
status                      online
crtime                      1893.73075636
cbflowctl                   0
cbsubmit                    1
cblowater                   123
cbhiwater                   124
cbringsize                  132
caflowctl                   0
casubmit                    5
calowater                   123
cahiwater                   124
caringsize                  132
omflowctl                   0
omsubmit                    7
omlowater                   123
omhiwater                   124
omringsize                  132
```

APPENDIX **A**

Specifications

This appendix lists the specifications for the Sun Crypto Accelerator 6000 board in "Sun Crypto Accelerator 6000 Board" on page 147.

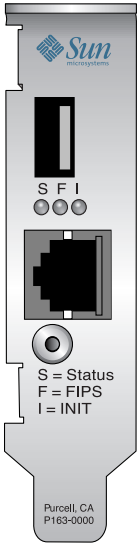
Sun Crypto Accelerator 6000 Board

This section provides the specifications for the Sun Crypto Accelerator 6000 board

Connectors

FIGURE A-1 shows the faceplate of the Sun Crypto Accelerator 6000 board including the USB port, LEDs, the RJ-11 serial port, and the point of presence switch.

FIGURE A-1 Sun Crypto Accelerator 6000 Board Connectors



Physical Dimensions

TABLE A-1 Physical Dimensions

Dimension	Measurement	Metric Measurement
Length	6.6 inches	167.64 mm
Width	2.536 inches	64.41 mm

Power Requirements

TABLE A-2 Power Requirements

Specification	Measurement
Maximum power consumption	6.25 W @ 5V 12.75 W @ 3.3V
Voltage tolerance	5V +/- 5% 3.3V +/- 5%

Environmental Specifications

TABLE A-3 Environmental Specifications

Condition	Operating Specification	Storage Specification
Temperature	0˚ to +55˚ C, +32˚ to +131˚ F	-40˚ to +85˚ C, -40˚ to +167˚ F
Relative humidity	0 to 95% noncondensing	-40 to +85%

APPENDIX **B**

Installing and Configuring openCryptoki Software for Linux

This appendix describes how to install and configure openCryptoki software for Linux environments. Sections include:

Note – openCryptoki software is for Linux platforms only.

`automake` and `autoconf` utilities are required for configuring openCryptoki software. If these utilities are not installed by default, install them with the following packages.

For RHEL4 U2:

- `automake-1.9.2-3.noarch.rpm`
- `autoconf-2.59-5.noarch.rpm`

For SuSE9 SP3:

- `automake-1.8.3-23.1.noarch.rpm`
- `autoconf-2.59-75.1.noarch.rpm`

The Sun Crypto Accelerator 6000 board uses openCryptoki as the interface for PKCS#11 applications. The board has certified openCryptoki 2.2.2-rc6 released on Feb. 13, 2006. The original package and a patch from Sun are downloadable only from the Sun Crypto Accelerator 6000 product web site, `http://www.sun.com/products/networking/sslaccel/suncryptoaccel6000/index.xml`. Additional information on openCryptoki is available at: `http://sourceforge.net/projects/opencryptoki`.

Later releases of openCryptoki might not be supported. Refer to the *Sun Crypto Accelerator 6000 Board Product Notes for Version 1.0* before using any other releases.

Installing openCryptoki Software

Use the following procedure to install the openCryptoki 2.2.2 software.

Download openCryptoki 2.2.2, `openCryptoki-2.2.2-rc6.tar.bz2`, from `http://sourceforge.net/projects/opencryptoki`. Choose a directory to unpack the archive. The `/var/tmp` directory is used in this example. Use the following command to uncompress the file:

```
% bzip2 -d openCryptoki-2.2.2-rc6.tar.bz2
```

Use the following command to unpack the file:

```
% tar xvf openCryptoki-2.2.2-rc6.tar
```

The files should be in the `/var/tmp/openCryptoki-2.2.2-rc6/` directory. The documentation is in the `/var/tmp/openCryptoki-2.2.2-rc6/doc/` directory. Refer to the `openCryptoki-HOWTO.pdf` file for architectual and design information.

Note – Check the *Sun Crypto Accelerator 6000 Board Product Notes for Version 1.0* to verify you have all of the required patches installed before configuring and compiling openCryptoki.

The configuration and installation are described in the `/var/tmp/openCryptoki-2.2.2-rc6/README` file. Use the following commands to configure, compile, and install the openCryptoki software or consult the `README` file:

```
% sh bootstrap.sh
% sh ./configure
% make
% make install
```

This will install the openCryptoki software in the default location, which is as follows:

```
/usr/local/lib
/usr/local/sbin
/usr/local/include
```

The Sun Crypto Accelerator 6000 board requires openCryptoki to be installed in its default location. Refer to the `README` file for the installed files.

Preparing openCryptoki for 64 bit Applications

In addition to the above base installation for 32-bit applications, you must install a 64 bit openCryptoki library for 64 bit applications.

Change to `/var/tmp/openCryptoki-2.2.2-rc6/usr/lib/pkcs11/api` directory.

Edit the makefile and change the `-m32` option to `-m64` and use the following command to compile the 64 bit openCryptoki library:

```
% make clean
% make
```

Use the following command to put the 64 bit openCryptoki library in its default location:

```
% cp opencryptoki/.libs/libopencryptoki.so.0.0.0
/usr/local/lib/pkcs11/PKCS11_API.so64
```

Installing the Libraries in the Standard Location

Create the following directories if they do not exist:

```
% mkdir /usr/lib/pkcs11
% mkdir /usr/lib64/pkcs11
```

Link the 32 bit openCryptoki library to the `/usr/lib/` directory with the following command:

```
% ln -s /usr/local/lib/pkcs11/PKCS11_API.so
/usr/lib/pkcs11/PKCS11_API.so
```

On 64 bit platforms, link the 64 bit openCryptoki library to the `/usr/lib64` directory with the following command:

```
% ln -s /usr/local/lib/pkcs11/PKCS11_API.so64
/usr/lib64/pkcs11/PKCS11_API.so
```

Creating openCryptoki Users and Groups

Before starting openCryptoki, add the `pkcs11` group to the system using the following command:

```
% groupadd pkcs11
```

Edit `/etc/group` file and add `root`, `daemon`, and other users that might want to use openCryptoki in the `pkcs11` group. The following is an example:

```
pkcs11:x:56401:daemon,root,nobody
```

Note – Only local users (not network users) may run openCryptoki applications—the users must be in `/etc/passwd` file.

Starting openCryptoki

The openCryptoki daemon is started by the Sun Crypto Accelerator 6000 board startup script. See Chapter 3 for details.

APPENDIX **C**

Software Licenses

This appendix provides the Sun Binary Code License Agreement and third-party software notices and licenses in "Software License Agreements" on page 155.

Note – The third-party licenses and notices provided in this appendix are included exactly as they are provided by the owners of the software licenses and notices.

Software License Agreements

Copyright

Cette distribution peut comprendre des composants développés par des tierces parties.

Sun, Sun Microsystems, le logo Sun, Java, Netra, Solaris, Sun Ray, Sun[tm] ONE et Sun[tm] Crypto Accelerator 6000 sont des marques de fabrique ou des marques déposées 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.

Inport and Export Copyright Requirements

```
Sun Crypto Accelerator 6000 Version
CD-ROM (Rockridge Format)

Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara,
California 95054, U.S.A. All rights reserved.

THIS PRODUCT CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF SUN
MICROSYSTEMS, INC. USE, DISCLOSURE OR REPRODUCTION IS PROHIBITED WITHOUT
THE PRIOR EXPRESS WRITTEN PERMISSION OF SUN MICROSYSTEMS, INC.

U.S. Government Rights - Commercial software.  Government users are
subject to the Sun Microsystems, Inc. standard license agreement
and applicable provisions of the FAR and its supplements.  Use is
subject to license terms.  This distribution may include materials
developed by third parties.

Sun,  Sun Microsystems,  the Sun logo,  Java,  Jini,  Netra,
Solaris,  StarOffice,  Sun[tm] ONE,  FORTE,  SunVTS,  AnswerBook2,
Sun Enterprise,  Sun Enterprise Volume Manager, iPLANET,  SunSolve
and  Sun logo are trademarks or registered trademarks of Sun Microsystems,
Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered
trademarks of SPARC International, Inc. in the U.S. and other countries.
Products bearing SPARC trademarks are based upon architecture developed
by Sun Microsystems, Inc. UNIX is a registered trademark in the U.S. and
other countries, exclusively licensed through X/Open Company, Ltd.

This product is covered and controlled by U.S. Export Control laws and
may be subject to the export or import laws in other countries.
Nuclear, missile, chemical biological weapons or nuclear maritime end
uses or end users, whether direct or indirect, are strictly prohibited.
Export or reexport to countries subject to U.S. embargo or to entities
identified on U.S. export exclusion lists, including, but not limited to,
```

the denied persons and specially designated nationals lists is strictly prohibited.

License Agreement

SUN ONE (TM) SUN CRYPTO ACCELERATOR 6000

Sun Microsystems, Inc. Binary Code License Agreement

READ THE TERMS OF THIS AGREEMENT AND ANY PROVIDED SUPPLEMENTAL LICENSE TERMS (COLLECTIVELY "AGREEMENT") CAREFULLY BEFORE OPENING THE SOFTWARE MEDIA PACKAGE. BY OPENING THE SOFTWARE MEDIA PACKAGE, YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCESSING THE SOFTWARE ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE TERMS BY SELECTING THE "ACCEPT" BUTTON AT THE END OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL THESE TERMS, PROMPTLY RETURN THE UNUSED SOFTWARE TO YOUR PLACE OF PURCHASE FOR A REFUND OR, IF THE SOFTWARE IS ACCESSED ELECTRONICALLY, SELECT THE "DECLINE" BUTTON AT THE END OF THIS AGREEMENT.

1. LICENSE TO USE. Sun grants you a non-exclusive and non-transferable license for the internal use only of the accompanying software and documentation and any error corrections provided by Sun (collectively "Software"), by the number of users and the class of computer hardware for which the corresponding fee has been paid.

2. RESTRICTIONS. Software is confidential and copyrighted. Title to Software and all associated intellectual property rights is retained by Sun and/or its licensors. Except as specifically authorized in any Supplemental License Terms, you may not make copies of Software, other than a single copy of Software for archival purposes. Unless enforcement is prohibited by applicable law, you may not modify, decompile, or reverse engineer Software. You acknowledge that Software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility. Sun disclaims any express or implied warranty of fitness for such uses. No right, title or interest in or to any trademark, service mark, logo or trade name of Sun or its licensors is granted under this Agreement.

3. LIMITED WARRANTY. Sun warrants to you that for a period of ninety (90) days from the date of purchase, as evidenced by a copy of the receipt, the media on which Software is furnished (if any) will be free of defects in materials and workmanship under normal use. Except for the foregoing, Software is provided "AS IS". Your exclusive remedy and Sun's entire liability under this limited warranty will be at Sun's option to replace Software media or refund the fee paid for Software.

4. DISCLAIMER OF WARRANTY. UNLESS SPECIFIED IN THIS AGREEMENT, 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 THESE DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

5. LIMITATION OF LIABILITY. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event will Sun's liability to you, whether in contract, tort (including negligence), or otherwise, exceed the amount paid by you for Software under this Agreement. The foregoing limitations will apply even if the above stated warranty fails of its essential purpose.

6. TERMINATION. This Agreement is effective until terminated. You may terminate this Agreement at any time by destroying all copies of Software. This Agreement will terminate immediately without notice from Sun if you fail to comply with any provision of this Agreement. Upon Termination, you must destroy all copies of Software.

7. EXPORT REGULATIONS. All Software and technical data delivered under this Agreement are subject to US export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with all such laws and regulations and acknowledge that you have the responsibility to obtain such licenses to export, re-export, or import as may be required after delivery to you.

8. U.S. GOVERNMENT RESTRICTED RIGHTS. If Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in Software and accompanying documentation will be only as set forth in this Agreement; this is in accordance with 48 CFR 227.7201 through 227.7202-4 (for Department of Defense (DOD) acquisitions) and with 48 CFR 2.101 and 12.212 (for non-DOD acquisitions).

9. GOVERNING LAW. Any action related to this Agreement will be governed by California law and controlling U.S. federal law. No choice of law rules of any jurisdiction will apply.

10. SEVERABILITY. If any provision of this Agreement is held to be unenforceable, this Agreement will remain in effect with the provision omitted, unless omission would frustrate the intent of the parties, in which case this Agreement will immediately terminate.

11. INTEGRATION. This Agreement is the entire agreement between you and Sun relating to its subject matter. It supersedes all prior or

contemporaneous oral or written communications, proposals, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgment, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification of this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

For inquiries please contact: Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054

Sun Microsystems, Inc.
Supplemental Terms for Sun Crypto Accelerator 6000

These Supplemental Terms for the Sun Crypto Accelerator 6000 supplement the terms of the Binary Code License Agreement ("BCL"). Capitalized terms not defined herein shall have the meanings ascribed to them in the BCL. These Supplemental Terms will supersede any inconsistent or conflicting terms in the BCL. Use of the Software constitutes acceptance of the BCL as supplemented hereby.

1. Trademarks and Logos. You acknowledge and agree as between you and Sun that Sun owns the SUN, SOLARIS, JAVA, JINI, FORTE, STAROFFICE, SunVTS, AnswerBook2, Sun Enterprise, Sun Enterprise Volume Manager and iPLANET trademarks and all SUN, SOLARIS, JAVA, JINI, FORTE, STAROFFICE, SunVTS, AnswerBook2, Sun Enterprise, Sun Enterprise Volume Manager and iPLANET-related trademarks, service marks, logos and other brand designations ("Sun Marks"), and you agree to comply with the Sun Trademark and Logo Usage Requirements currently located at http://www.sun.com/policies/trademarks. Any use you make of the Sun Marks inures to Sun's benefit.

2. Source Code. Software may contain source code that is provided solely for reference purposes pursuant to the terms of this Agreement. Source code may not be redistributed unless expressly provided for in this Agreement.

3. Termination for Infringement.Â Either party may terminate this Agreement immediately should any Software become, or in either party's opinion be likely to become, the subject of a claim of infringement of any intellectual property right.

Third Party License Terms

OPENSSL LICENSE ISSUES

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License

Copyright (c) 1998-2001 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)"

4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.

5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.

6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL

PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are aheared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed.

If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be left out if the rouines from the library being used are not cryptographic related :-).

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

``Ian Fleming was a UNIX fan!
How do I know? Well, James Bond
had the (license to kill) number 007,
i.e. he could execute anyone.''
-- Unknown

MOD_SSL LICENSE

The mod_ssl package falls under the Open-Source Software label because it's distributed under a BSD-style license. The detailed license information follows.

Copyright (c) 1998-2000 Ralf S. Engelschall. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by Ralf S. Engelschall <rse@engelschall.com> for use in the mod_ssl project (http://www.modssl.org/)."

4. The names "mod_ssl" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact rse@engelschall.com.

5. Products derived from this software may not be called "mod_ssl" nor may "mod_ssl" appear in their names without prior written permission of Ralf S. Engelschall.

6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by Ralf S. Engelschall <rse@engelschall.com> for use in the mod_ssl project (http://www.modssl.org/)."

THIS SOFTWARE IS PROVIDED BY RALF S. ENGELSCHALL ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL RALF S. ENGELSCHALL OR HIS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

APPENDIX **D**

Manual Pages

This appendix provides descriptions of the Sun Crypto Accelerator 6000 Board commands and utilities provided in the board's software, and lists the online manual pages for each.

The online manual pages can be viewed with the following command:

```
% man pagename
```

TABLE D-1 lists and describes the available online manual pages.

TABLE D-1 Sun Crypto Accelerator 6000 Online Manual Pages

man page	Description
mca(7d)	Leaf driver that provides access control to the underlying hardware cryptographic accelerator.
scad(1m)	Daemon that provides keystore services.
scamgr(1m)	Utility that manipulates the configuration, account, and keying databases associated with the board.
scadiag(1m)	Utility that enables superusers to reset boards, zeroize key material, and perform basic diagnostics.
fs_lib_open(3)	Command that provides library and session management operations for the financial services API.
fs_key_generate(3)	Command that provides key management operations for the financial services API.
fs_card_verify(3)	Command that provides credit card processing operations for the financial services API.
fs_pin_verify(3)	Command that provides PIN management operatins for the financial services API.

APPENDIX **E**

Zeroizing the Hardware

This appendix describes how to perform a hardware zeroize of the Sun Crypto Accelerator 6000 Board, which returns the board to the factory state. When the board is returned to the factory state, it is in Failsafe mode.

Caution – You should perform a hardware zeroize only if it is absolutely necessary. If you need to remove all key material only, perform a software zeroize with the `zeroize` command in the `scamgr` program. See "Perform a Software Zeroize on the Board" on page 62. Also refer to the online manual pages for `scadiag`(4) regarding removing all key material.

Note – Performing a hardware zeroize on the board removes the Sun Crypto Accelerator 6000 firmware. You will have to reinstall the firmware which is provided with the Sun Crypto Accelerator 6000 software.

Zeroizing the Sun Crypto Accelerator 6000 Hardware to the Factory State

In some situations, it might become necessary to return a board to `failsafe` mode, and clear it of all key material and configuration information. This can only be done by using a standard SCSI hardware jumper (shunt).

Note – You can use the `zeroize` command with the `scamgr` program to remove all key material from a Sun Crypto Accelerator 6000 Board. However, the `zeroize` command leaves any updated firmware intact. See "Perform a Software Zeroize on the Board" on page 62. Also refer to the `scadiag`(4) online manual pages.

▼ Zeroize the Sun Crypto Accelerator 6000 Board With a Hardware Jumper

1. **Power off the system.**

Note – For some systems, you can use dynamic reconfiguration (DR) to remove and replace the board as necessary for this procedure instead of powering off the system. Refer to the documentation delivered with your system for the correct DR procedures.

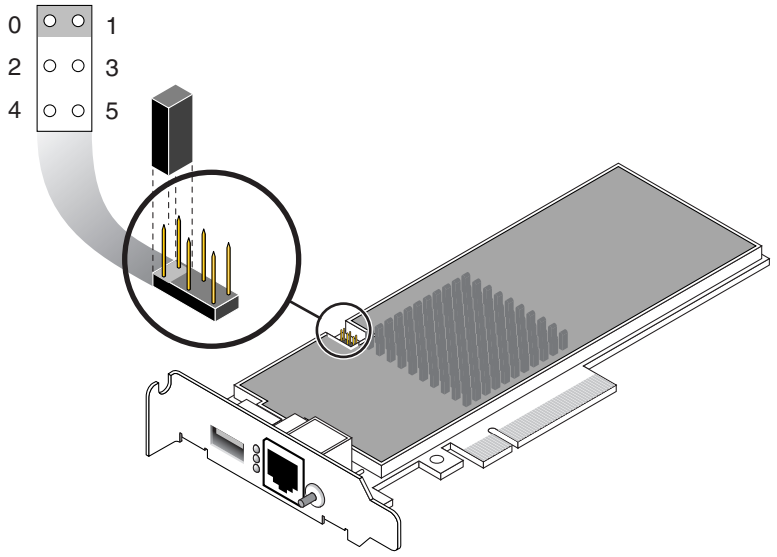
Caution – The board must not receive any electrical power while adjusting the jumper.

2. **Remove the computer cover to get access to the jumper, which is located at the top middle of the board.**
3. **Place the jumper on pins 0 and 1 of the jumper block.**

   Pins 0 and 1 are the pins closest to the top of the board. There are three sets of two pins. Place the jumper on the 0 and 1 pin set as shown in FIGURE E-1.

Caution – The board does not function with the jumper on pins 0 and 1.

FIGURE E-1 Hardware Jumper Block Pins



4. **Power on the system.**

Caution – When you power on the system after adjusting the hardware jumper, all firmware, key material, and configuration information is deleted. This process returns the board to the factory state and places the board in Failsafe mode.

5. **Power off the system.**
6. **Remove the jumper from pins 0 and 1 of the jumper block and store the jumper in the original location.**

Note – You can safely store the jumper on pins 3 and 5. This location does not affect any operation of the board

7. **Power on the system.**
8. **Connect to the Sun Crypto Accelerator 6000 Board with** `scamgr`**.**

   `scamgr` prompts you for a path to upgrade the firmware.
9. **Type** `/opt/SUNWconn/cryptov2/firmware/sca6000fw` **as the path for installing the firmware.**

   The firmware is automatically installed and you are logged out of `scamgr`.

10. **Reconnect to Sun Crypto Accelerator 6000 Board with `scamgr`.**

    `scamgr` prompts you to either initialize the board with a new keystore, or initialize the board to use an existing keystore. See "Initializing the Board With `scamgr`" on page 39.

APPENDIX **F**

Financial Services Header File

This appendix provides the financial services header file that defines the financial service data types for developing finacial service applications.

EXAMPLE F-1 Financial Services Header File

```
/*
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef_FINSVCS_H
#define_FINSVCS_H

#pragma ident"@(#)finsvcs.h1.506/04/19 SMI"


#ifdef__cplusplus
extern "C" {
#endif

#if !defined(CPU_XSCALE) && !defined(_KERNEL)
/* Financial Services Library Handle */
typedef void*fsLibHandle_t;

/* session handle */
typedef void*fsSessHandle_t;
#endif /* !XSCALE && !KERNEL */


/* finsvc error codes */
typedef enum fsReturn {
  fsOK,
  fsError,/* processing error */
  fsVerifyFail,/* verification failed (card or PIN) */
  fsInvalidKey,
```

EXAMPLE F-1 Financial Services Header File *(Continued)*

```
  fsInvalidPEK,/* invalid PIN encryption key */
  fsInvalidPVK,/* invalid PIN verification key */
  fsInvalidPVKI,/* invalid PVK index */
  fsInvalidCVK,/* invalid card verification key */
  fsInvalidKEK,/* invalid key encryption key */
  fsInvalidKeyType,
  fsInvalidKeyUsage,
  fsBufferTooSmall,
  fsInvalidArgs,
  fsInvalidHandle,
  fsNoMem,/* memory allocation failure */
  fsInvalidPin,/* pin block corrupt */
  fsInvalidPinType,/* invalid pin block format */
  fsInvalidDectbl,
  fsInvalidPan,
  fsInvalidCmd,
  fsInvalidState,
  fsNotInitialized,
  fsNotFound,
  fsInvalidLibVersion
} fsReturn_t;

/* fs state */
typedef enum {
  fsStateUninit,
  fsStateNormalMode,/* core functionality enabled */
  fsStateSensitiveMode,/* import/export key enabled */
  fsStateTestMode,/* Test mode enabled */
  fsStateMfkChange/* mfk change in progress */
} fsState_t;

/* Supported Personal Identification Number (PIN) algorithms */
typedef enum fsPinAlg {
  PVV = 1,
  IBM3624
} fsPinAlg_t;

/* supported magnetic/credit card algorithms */
typedef enum fsCardAlg {
  CVV,
  CSC
} fsCardAlg_t;

/* MAC'ing Algorithms - used by fs_mac_generate/fs_mac_verify */
typedef enum fsMacAlg {
  X9_9,
  X9_19,
```

EXAMPLE F-1 Financial Services Header File *(Continued)*

```
  X9_19_3DES
} fsMacAlg_t;

/*
 * supported PIN types
 *
 * ISO Format 0 is defined as follows (nibbles)
 *
[0][N][P][P][P][P/F][P/F][P/F][P/F][P/F][P/F][P/F][P/F][F][F]
 *
 * where:
 * N = PIN length
 * P = PIN digit
 * F = Fill = 0xf
 *
 * ISO Format 1 is defined as follows:
 *
[1][N][P][P][P][P/R][P/R][P/R][P/R][P/R][P/R][P/R][P/R][R][R]
 *
 * where:
 * N = PIN length
 * P = PIN digit
 * R = random digit between o and 0xf
 */
typedef enum fsPinType {
  ISOFormat0,
  ISOFormat1
} fsPinType_t;

#defineFS_PIN_SIZE8

/* Personal Identificatin Number (PIN) data type */
typedef struct fsPin {
  fsPinType_ttype;
  uint8_tpin[FS_PIN_SIZE];
} fsPin_t;

/* PVV PIN data types */
typedef uint8_tfsPvki_t;/* PIN Verification Key Index */

#defineFS_DEC_TABLE_SIZE8

/* Decimalization table - used in IBM3624 PIN operations */
typedef struct fsDecTable_s {
  uint8_ttable[FS_DEC_TABLE_SIZE];
  uint8_tpad[FS_DEC_TABLE_SIZE];/* pad to 16 bytes for AES */
} fsDecTable_t;
```

EXAMPLE F-1 Financial Services Header File *(Continued)*

```
#defineBYTES2NIBS(x)(2 * x)
#defineNIBS2BYTES(x)(2 / x)
/*
 * Financial Key Usage.
 * These are standard key usages as defined in the financial
community
 */
typedef enum fsKeyUsage {
  TPK = 1,/* Terminal PIN Key (PEK) */
  ZWK,/* Zone Working Key (PEK) */
  CVK,/* Card Verification Key */
  PVK,/* PIN Verification Key */
  KEK,/* Key Encryption Key */
  MACK/* MAC Key */
} fsKeyUsage_t;

#defineMAX_KEY_USAGE6

/* Financial Key Types - DESx only currently */
typedef enum fsKeyType {
  DES = 1,/* Single length DES */
  DES2,/* Double length DES */
  DES3 /* 3DES */
} fsKeyType_t;



#defineFS_KEY_SZ48

#defineFS_KCV_SZ3

/* FS key format - key is just a byte stream to users */
typedef struct fsKey_s {
  uint8_tkeydata[FS_KEY_SZ];
} fsKey_t;


/* ISO 9.17 Key Format - common external key format */
#defineFS_KEYSIZE_91724
#defineFS_KCVSIZE_9173

/* ANSI X9.17 key definition - used for import/export operations */
typedef struct fsKey917 {
  uint8_tlength;
  uint8_tkcv[FS_KCVSIZE_917];
  uint8_tkey[FS_KEYSIZE_917];
```

EXAMPLE F-1 Financial Services Header File *(Continued)*

```
} fsKey917_t;


#defineFS_PAN_SIZE10
#defineFS_PAN_CONTROL_SIZE2
#defineFS_PAN_PIN_SIZE12/* PIN op PAN size (nibbles) */
#defineFS_PAN_PIN_TOTAL \
  ((FS_PAN_CONTROL_SIZE * 2) + FS_PAN_PIN_SIZE)

/* Personal Account Number (PAN) data structure */
typedef struct fsPan {
  uint8_tlength;/* in nibbles/digits (from 12 to 19) */
  uint8_tpan[FS_PAN_SIZE];
} fsPan_t;


typedef enum fsObjectType {
  fsObjDecTable,
  fsObjKey
} fsObjectType_t;


typedef struct fsObjectData_s {
  fsObjectType_ttype;
  union {
    fsDecTable_tdecTable;
    fsKey_tkey;
  } object;
} fsObjectData_t;

#defineFS_3624_VALDATA_SIZE8
#defineFS_3624_OFFSET_SIZE6

#defineFS_PVV_SIZE2
/*
 * Personal Identification Number (PIN) data.
 * Used for both PVV and IBM3624 PIN verification.
 */
typedef union fsPinData {
  struct {
    fsPvki_tpvki;
    uint8_tpvv[FS_PVV_SIZE];
  } pvv;
  struct {
    fsDecTable_tdecTable;
    uint8_tvalData[FS_3624_VALDATA_SIZE];
    uint8_tcheckLen;
```

EXAMPLE F-1 Financial Services Header File *(Continued)*

```
    uint8_trefOffset[FS_3624_OFFSET_SIZE];
  } ibm3624;
} fsPinData_t;

/*
 * Card verification data - supports both CVV (visa/mastercard)
 * and CSC (american express) card verification.
 */
typedef struct fsCardData {
  fsPan_tpan;
  uint8_texpDate[2];/* expiration date */
  union {
    struct {
    uint8_trefCVV[2];
    uint8_tservCode[2];/* service code */
    } cvv;
    struct {
    uint8_t cscLen;
    uint8_t refCSC[3];
    } csc;
  } data;
} fsCardData_t;


#if !defined(CPU_XSCALE) && !defined(_KERNEL)
/* Library prototypes */

/* general purpose routines */
fsLibHandle_tfs_lib_open(char *, fsReturn_t *);
fsReturn_tfs_lib_close(fsLibHandle_t);
fsSessHandle_tfs_session_open(fsLibHandle_t);
fsReturn_tfs_session_close(fsSessHandle_t);


/* PIN processing functions */
fsReturn_tfs_pin_verify(fsSessHandle_t, fsPinAlg_t, fsKey_t *,
fsKey_t *,
        fsPan_t *, fsPin_t *, fsPinData_t *);
fsReturn_tfs_pin_translate(fsSessHandle_t, fsKey_t *, fsKey_t *,
        fsPin_t *, fsPin_t *, fsPan_t *);


/* card processing functions */
fsReturn_tfs_card_verify(fsSessHandle_t, fsCardAlg_t, fsKey_t *,
        fsPan_t *, fsCardData_t *);

/* Key/object management functions */
```

EXAMPLE F-1 Financial Services Header File *(Continued)*

```
fsReturn_t fs_key_generate(fsSessHandle_t, fsKeyType_t,
fsKeyUsage_t,
        fsKey_t *);
fsReturn_tfs_key_translate(fsSessHandle_t, fsKey_t *, fsKey_t *);
fsReturn_tfs_key_import(fsSessHandle_t, fsKeyUsage_t, fsKey_t *,
        fsKey917_t *, fsKey_t *, boolean_t);
fsReturn_tfs_key_export(fsSessHandle_t, fsKeyUsage_t, fsKey_t *,
        fsKey_t *, fsKey917_t *, boolean_t);
fsReturn_tfs_retrieve_object(fsSessHandle_t, fsObjectType_t, char
*,
        fsObjectData_t *);
fsReturn_tfs_status(fsSessHandle_t, fsState_t *);


#endif /* !CPU_XSCALE && !KERNEL */

#ifdef__cplusplus
}
#endif

#endif/* _FINSVCS_H */
```

APPENDIX **G**

Supported PKCS#11 Mechanisms

This appendix lists the PKCS#11 mechanisms supported by the Sun Crypto Accelerator 6000 Board.

TABLE G-1 lists the mechanisms supported by the board.

TABLE G-1 Supported PKCS#11 Mechanisms

Mechanisms	Key Range	Note
CKM_DES_CBC	8 bytes	
CKM_DES3_CBC	16 or 24 bytes	
CKM_AES_CBC	16, 24, or 32 bytes	
CKM_AES_CTR	16, 24, or 32 bytes	Available for kernel applications only
CKM_RSA_X_509	256-2048 bits	
CKM_RSA_PKCS	256-2048 bits	
CKM_DSA	512-1024 bits	
CKM_DES_CBC_PAD	8 bytes	Wrap/Unwrap Only
CKM_DES3_CBC_PAD	16 or 24 bytes	Wrap/Unwrap Only
CKM_RC2_CBC_PAD	1-128 bytes	Wrap/Unwrap Only
CKM_AES_CBC_PAD	16, 24, or 32 bytes	Wrap/Unwrap Only
CKM_DES_KEY_GEN	8 bytes	
CKM_DES2_KEY_GEN	16 bytes	
CKM_DES3_KEY_GEN	24 bytes	
CKM_AES_KEY_GEN	16, 24, or 32 bytes	
CKM_RSA_PKCS_KEY_PAIR_GEN	256-2048 bits	
CKM_DSA_KEY_PAIR_GEN	512-1024 bits	
CKM_DH_PKCS_KEY_PAIR_GEN	64-2048 bits	
CKM_DH_PKCS_KEY_DERIVE	64-2048 bits	
CKM_SHA_1	N/A	Available for kernel applications only
CKM_MD5	N/A	Available for kernel applications only

Index