THALES

ProtectToolkit 5.9.1
ProtectServer HSM

MIGRATION GUIDE

Document Information

Last Updated	2024-04-18 12:25:13-04:00

that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service, or loss of privacy.

CONTENTS

PREFACE: About the ProtectServer HSM Migration Guide

This document describes the key differences between the ProtectServer Network and PCIe HSMs and their legacy counterparts. It also provides instructions for migrating Functionality Modules (FMs) to the new environment. It contains the following chapters:

> "HSM Migration" on page 8

> "FM Migration" on page 12

This preface also includes the following information about this document:

> "Document Conventions" below

> "Support Contacts" on page 7

For information regarding the document status and revision history, see "Document Information" on page 2.

Document Conventions

This document uses standard conventions for describing the user interface and for alerting you to important information.

Notes

Notes are used to alert you to important or helpful information. They use the following format:

> **NOTE** Take note. Contains important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss. They use the following format:

> **CAUTION!** Exercise caution. Contains important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury. They use the following format:

> ****WARNING** Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury.**

Command Syntax and Typeface Conventions

Format	Convention
bold	The bold attribute is used to indicate the following: > Command-line commands and options (Type **dir /p**.) > Button names (Click **Save As**.) > Check box and radio button names (Select the **Print Duplex** check box.) > Dialog box titles (On the **Protect Document** dialog box, click **Yes**.) > Field names (**User Name:** Enter the name of the user.) > Menu names (On the **File** menu, click **Save**.) (Click **Menu > Go To > Folders**.) > User input (In the **Date** box, type **April 1**.)
italics	In type, the italic attribute is used for emphasis or cross-references to other documents in this documentation set.
<variable>	In command descriptions, angle brackets represent variables. You must substitute a value for command line arguments that are enclosed in angle brackets.
[**optional**] [<optional>]	Represent optional **keywords** or <variables> in a command line description. Optionally enter the keyword or <variable> that is enclosed in square brackets, if it is necessary or desirable to complete the task.
{**a**|**b**|**c**} {<a>|<b>|<c>}	Represent required alternate **keywords** or <variables> in a command line description. You must choose one command line argument enclosed within the braces. Choices are separated by vertical (OR) bars.
[**a**|**b**|**c**] [<a>|<b>|<c>]	Represent optional alternate keywords or variables in a command line description. Choose one command line argument enclosed within the braces, if desired. Choices are separated by vertical (OR) bars.

Support Contacts

If you encounter a problem while installing, registering, or operating this product, please refer to the documentation before contacting support. If you cannot resolve the issue, contact your supplier or Thales Customer Support.

Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable database of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone

The support portal also lists telephone numbers for voice contact (Contact Us).

CHAPTER 1: HSM Migration

The ProtectServer Network and PCIe HSMs are direct replacements for the legacy PSE and PSI-E HSMs, which have been declared end of sale, and are no longer available for purchase.

Although the ProtectServer Network and PCIe HSMs are functionally equivalent to their legacy counterparts, the underlying hardware is significantly different. The major hardware change is to the embedded cryptographic engine used on the HSMs. The legacy PSE/PSI-E HSMs incorporate the K5 cryptographic engine. The ProtectServer Network and PCIe HSMs incorporate the more modern K6 cryptographic engine.

Although every effort has been made to mitigate the impact of these hardware changes, the introduction of a new cryptographic engine impacts the following:

- Functionality modules (FMs). The processor used on the ProtectServer Network and PCIe HSMs is different from the processor used on the legacy ProtectServer HSMs. As a result, you must rebuild your FMs to run on the new hardware.
- Serial devices. The serial port on the ProtectServer HSMs has been replaced on the ProtectServer Network and PCIe HSMs with a USB port and a USB-to-serial cable. Any serial devices that were previously attached to a ProtectServer HSM will continue to work on the ProtectServer Network and PCIe HSMs.

In addition to these changes, the SafeNet ProtectToolkit also includes some software fixes/enhancements, as described in "Software Changes" below.

Functionality Modules

The K5 cryptographic engine is based on the ARM processor. The K6 cryptographic engine is based on the PowerPC processor. As a result, any FMs built for the PSE/PSI-E (K5) HSMs will not run on the ProtectServer Network and PCIe HSMs. You must rebuild your existing FMs to run on the PowerPC (K6) platform. This requires a Linux machine or VM and some changes to your source files, as described in "FM Migration" on page 12.

Software Changes

The software changes introduced in this release primarily affect the FM SDK, as detailed in "FM Migration" on page 12. Any additional changes are described in the following sections.

FM SDK (formerly PPO) is now included with the ProtectToolkit software

The latest versions of the client software and HSM firmware can be found on the Thales Technical Support Customer Portal. See "Support Contacts" on page 7 for more information.

Installation Directories

The installation directories have been modified to conform to SafeNet standard conventions, as follows:

Linux	/opt/safenet/protecttoolkit5/opt/safenet/fm-toolchain
Windows	C:\Program Files\SafeNet\Protect Toolkit 5

Environment Variables

Environment configuration for the ProtectToolkit-C SDK and FM SDK has been simplified in this release as follows. Manual setting of environment variables is no longer required.

Linux	A configuration script (**setvars.sh**) is now included with ProtectToolkit-C to configure your development environment. You would typically run this script each time you open a new shell. See the installation documentation for more information.
Windows	The runtime environment is automatically configured as part of the installation process. The FM SDK installation directory includes a configuration batch (**fmsdkvars.bat**) file to configure your FM development environment. You would typically run this batch file each time you open a new shell. See the installation documentation for more information.

Installer Directory Structure

```
├──<part_number>_sw_license_agreement.pdf
├──<part_number>_sw_license_agreement.txt
├──autorun.inf
│
├──firmware
│   └──<firmware_upgrade_files>
│
└──SDKs
    ├── safeNet-install.sh
    │
    ├── AIX
    │   ├── PTKcprt
    │   │   └── PTKcprt.bff
    │   ├── PTKcpsdk
    │   │   └── PTKcpsdk.bff
    │   ├── PTKjprov
    │   │   └── PTKjprov.bff
    │   ├── PTKjpsdk
    │   │   └── PTKjpsdk.bff
    │   └── PTKnethsm
    │       └── PTKnethsm.bff
    │
    ├── HP-UX
    │   ├── PTKcprt
    │   │   └── PTKcprt.depot
    │   ├── PTKcpsdk
    │   │   └── PTKcpsdk.depot
    │   ├── PTKjprov
    │   │   └── PTKjprov.depot
    │   ├── PTKjpsdk
    │   │   └── PTKjpsdk.depot
    │   └── PTKnethsm
    │       └── PTKnethsm.depot
```

```
│
├── Linux
│   ├── fm_sdk
│   │   └── PTKfmsdk-<version>.i386.rpm
│   ├── fm_toolchain
│   │   └── fm-toolchain-ppc440e-<version>.i686.rpm
│   ├── hsm_net_server
│   │   └── PTKnetsrv-<version>.i386.rpm
│   ├── network_hsm_access_provider
│   │   └── PTKnethsm-<version>.i386.rpm
│   ├── pci_hsm_access_provider
│   │   └── PTKpcihsmK6-<version>.i386.rpm
│   ├── ptkc_runtime
│   │   └── PTKcprt-<version>.i386.rpm
│   ├── ptkc_sdk
│   │   └── PTKcpsdk-<version>.i386.rpm
│   ├── ptkj_runtime
│   │   └── PTKjprov-<version>.i386.rpm
│   └── ptkj_sdk
│       └── PTKjpsdk-<version>.i386.rpm
│
├── Linux64
│   ├── fm_sdk
│   │   └── PTKfmsdk-<version>.x86_64.rpm
│   ├── fm_toolchain
│   │   └── fm-toolchain-ppc440e-<version>.i686.rpm
│   ├── hsm_net_server
│   │   └── PTKnetsrv-<version>.x86_64.rpm
│   ├── network_hsm_access_provider
│   │   └── PTKnethsm-<version>.x86_64.rpm
│   ├── pci_hsm_access_provider
│   │   └── PTKpcihsmK6-<version>.x86_64.rpm
│   ├── ptkc_runtime
│   │   └── PTKcprt-<version>.x86_64.rpm
│   ├── ptkc_sdk
│   │   └── PTKcpsdk-<version>.x86_64.rpm
│   ├── ptkj_runtime
│   │   └── PTKjprov-<version>.x86_64.rpm
│   └── ptkj_sdk
│       └── PTKjpsdk-<version>.x86_64.rpm
│
├── Solaris
│   ├── PTKcprt
│   │   └── PTKcprt.pkg
│   ├── PTKcpsdk
│   │   └── PTKcpsdk.pkg
│   ├── PTKjprov
│   │   └── PTKjprov.pkg
│   ├── PTKjpsdk
│   │   └── PTKjpsdk.pkg
│   └── PTKnethsm
│       └── PTKnethsm.pkg
│
├── SolarisX86
│   ├── PTKcprt
│   │   └── PTKcprt.pkg
│   ├── PTKcpsdk
│   │   └── PTKcpsdk.pkg
```

```
│   ├── PTKjprov
│   │   └── PTKjprov.pkg
│   ├── PTKjpsdk
│   │   └── PTKjpsdk.pkg
│   └── PTKnethsm
│       └── PTKnethsm.pkg
│
├──Win32
│   ├──fm_sdk
│   │   └── PTKfmsdk.msi
│   ├──hsm_net_server
│   │   └── PTKnethsm.msi
│   ├──network_hsm_access_provider
│   │   └── PTKnethsm.msi
│   ├──pci_hsm_access_provider
│   │   └── PTKpcihsmK6.msi
│   ├──ptkc_runtime
│   │   └── PTKcprt.msi
│   ├──ptkc_sdk
│   │   └── PTKcpsdk.msi
│   ├──PTKJ_Runtime
│   │   └──PTKjprt.msi
│   ├──PTKJ_SDK
│   │   └── PTKjpsdk.msi
│   └──Ptk-M
│       └── SafenetKSP32.msi
│
└──Win64
    ├──fm_sdk
    │   └── PTKfmsdk.msi
    ├──hsm_net_server
    │   └── PTKnethsm.msi
    ├──network_hsm_access_provider
    │   └── PTKnethsm.msi
    ├──pci_hsm_access_provider
    │   └── PTKpcihsmK6.msi
    ├──ptkc_runtime
    │   └── PTKcprt.msi
    ├──ptkc_sdk
    │   └── PTKcpsdk.msi
    ├──PTKJ_Runtime
    │   └──PTKjprt.msi
    ├──PTKJ_SDK
    │   └── PTKjpsdk.msi
    └──Ptk-M
        ├── PTKmprt64.msi
        └── SafenetKSP64.msi
```

CHAPTER 2: FM Migration

This chapter describes the changes and enhancements to the FM development process for the ProtectServer Network and PCIe HSMs. It provides guidance and recommends best practices for developing new FMs or migrating your existing FMs to work on the ProtectServer Network and PCIe HSMs. This chapter contains the following sections:

Supported Hardware and Software

FM, HOST or Cryptoki applications built or recompiled using the Release 5 (or higher) FM SDK are supported on the ProtectServer Network and PCIe HSMs.

Recompiling your existing FM, HOST, and Cryptoki application source code should require **makefile** modifications only. FMs must be compiled on Linux instead of Windows.

Platform support is as follows:

Embedded FM development	Linux only
FM HOST applications	Windows and Linux

The minimum required installation for building FM HOST applications with ProtectToolkit 5 on Windows is Microsoft Visual C++ (the 2005, 2008, and 2010 runtime redistributables must all be installed), **PTKcpsdk**, and **PTKfmsdk**. Compilers other than Microsoft Visual C++ may work, but our examples and configuration **makefile (<FMDIR>\cfgbuild.mak)** are built for Microsoft Visual C++.

Summary of Changes and Enhancements

This section provides a summary of the changes and enhancements made to the FM SDK and toolchain. In most cases, your existing FMs should continue to work after recompiling on the new platform.

The most significant change to the FM development process in Release 5 (or higher) is that FMs must now be compiled on Linux instead of Windows. This will require moving your source to a Linux machine or VM. Recompiling in the new environment should not require any source changes.

If your existing FMs use the makefile syntax used in the samples provided in previous ProtectServer releases, you must update your makefiles to use the syntax defined in the samples provided with the Release 5 (or higher) ProtectToolkit.

Some of the APIs used in previous ProtectServer releases have been changed or deprecated, however the header files provided with Release 5 (or higher) ProtectToolkit include backwards compatibility for the old APIs. Warnings for the deprecated APIs will be displayed at compile time.

See descriptions of the following changes below:

FM SDK Toolkit

The FM SDK cross-compiler is now Linux-only.

There are no longer separate FM and HOST toolkits. All FM-SDK toolkits are the same, except that the cross compiler and embedded libraries will only be available on Linux hosts, and are no longer supported on Windows XP.

Toolchain

The major changes to the toolchain are as follows:

- **GCCFMDIR** is no longer required
- **gnumake** is now **make**, and you can now use the native system **make**
- C99 support
- additional function support
- the tool naming convention has changed
- some of the tools in the toolchain use newer versions

$GCCFMDIR No Longer Required

$GCCFMDIR is not required for building FMs on the ProtectServer PCIe HSM since the toolchain is non-relocatable.

> **NOTE** For best practice, ensure that **$GCCFMDIR** is not defined.

Builds Use make Instead of gnumake

In Release 5 (or higher), **gnumake** becomes simply **make**, and the host's native make can be used. If the host's native **make** is older than 3.82, **make** can also be used from the **fm-toolchain** from the following location:

/opt/safenet/fm-toolchain/fmsdk-ppc440e-1.0/sysroots/i686-fmsdk-linux/usr/bin/make

In previous releases, we provided GNU make (renamed to **gnumake**). The **gnumake** command was used to build FMs. This has changed as follows:

Linux	The **gnumake** command is no longer available. If you use an automated build that calls **gnumake** you must update your build scripts to call **make** instead of **gnumake**.
Windows	The **gnumake** command is provided in **<fm_install_dir>/bin**. It is simply a renamed version of the **make** command.

C99 Support

The FM SDK supports a subset of the ISO C 99 standard library as defined by ISO/IEC 9899:1999. In general, floating point math, multibyte characters, localization, and I/O APIs are not supported. **printf** and **vprintf** are exceptions, and are redirected to the logging channel.

In addition to the standard library, you can also use C99 language features not present in ANSI C (C89/90). C99 stdint.h types are now supported by the FM SDK toolchain, however sized types from the proprietary Integers.h remain used for FM SDK published APIs to maintain continuity and compatibly with pre-C99 versions.

Due to the change to C99, and the default C99 locale, the strftime %x result is a different format than in PPO3. It is now mm/dd/yy instead of Ddd Mmm dd yyy.

See the FM SDK Programming Guide for more information.

> **NOTE** For best practice, update your code to take advantage of the features offered by C99.

Functions added to the FM SDK in ProtectServer 2

The following functions have been added:

- **ctype.c**

  isblank

- **stdio.h**

  printf, vprintf, vsscanf

  snprintf (moves from non-standard to C99 variant).

- **stdlib.h**

  atoll, llabs, lldiv, strtoll, strtoull

Updated Tool Versions

Tool	PPO 3.0 version	FM SDK 5.0 version
gcc	2.95.3	4.6.1
gnu make	3.78.1	3.82
binutils	2.11.2	2.21.1
C standard	C89	C99

New Tool Naming Convention

The tool naming convention changes from **<toolname>-fm** to **<arch>-fm-<toolname>**. For example **gcc-fm** becomes **powerpc-fm-linux-gcc**.

Makefiles

Makefiles

The emulation mode makefile syntax used in Release 5 (or higher) is different than in previous releases. See the makefiles in the **samples** directory for the new syntax. You must change your existing makefiles to conform to the new syntax.

If your existing makefiles include **cfgbuild.mak**, you should be able to recompile existing non-emulation code without making any changes, other than changing the path separators, as described below. If you only used **cfgbuild.mak** as an example, changes will be required in your makefiles for the new toolchain. See the makefiles in the samples directory, and the **cfgbuild.mak** file.

Path Separators

When building for ProtectServer PCIe HSM on Linux hosts, all path separators in your makefiles must use forward slashes (/). Windows-style backslashes (\ ) will not work.

> **NOTE** Change all backslashes (\ ) to forward slashes (/) in your makefiles.

The cfgbuild.mak File

For both FMs and HOST applications, **cfgbuild.mak** is now a supported file and not a simply an example. The **cfgbuild.mak** file has been moved to **$(FMDIR)/** from **$(FMDIR)/samples**. A wrapper is provided in **$(FMDIR)/samples** for compatibility.

The default libraries provided by the FM SDK are now automatically included by **cfgbuild.mak** and should not be specified within the FM's makefile directly.

> **NOTE** For best practice, include **cfgbuild.mak** in all of your makefiles to set up the compiler, link flags, and link the FM toolchain.

Memory Endian Issues

The processors on the ProtectServer Network and PCIe HSMs are big endian, whereas the PSI-E and PSG processors are little endian. In Release 5 (or higher) all methods of endian byte order manipulation are consolidated into the **endyn.h** header file.

> **NOTE** For best practice, use the endian macros provided in the ProtectToolkit-C header file **endyn.h** to encode all messages in network byte order. By using the endian macros on both host and FM, endian differences between host and HSM are not an issue.

FM_MAKE_VERSION Macro

The samples included in previous ProtectServer releases did not use the FM_MAKE_VERSION(major,minor) macro. The samples in Release 5 (or higher) use this macro. Not using the macro may result in FM version numbers being displayed reversed in **ctconf** and **ctfm**.

> **NOTE** For best practice, use the FM_MAKE_VERSION(major,minor) macro to define the version number passed to DEFINE_FM_HEADER().

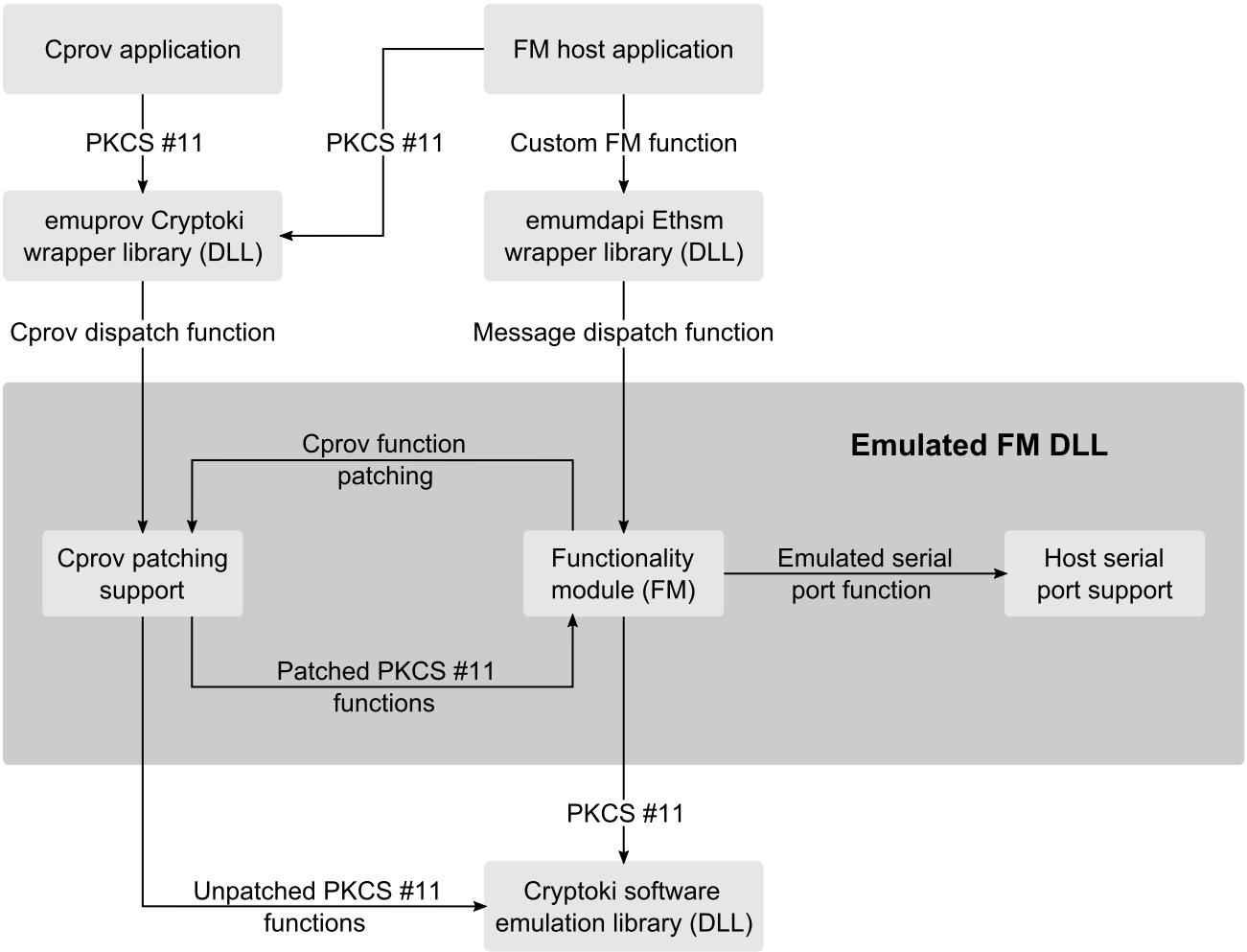
Emulation Mode Enhancements

Emulation mode has been enhanced to support C99 and Cryptoki (**Cprov**) function patching of any application that is run against the emulated cryptoki wrapper built with the emulated FM.

Unlike Protect Processing 3.0 and earlier releases, applications do not have to be recompiled with the emulated libraries, they simply have to have the **emucprov** and **emumdapi** wrapper libraries (with standard **cryptoki.so** and **ethsm.so** naming) in the library search path ahead of the real cryptoki library.

> **NOTE** For best practice, ensure that the **emucprov** and **emumdapi** wrapper libraries appear before the cryptoki library in your library search path.

The following diagram illustrates the functionality provided by the FM SDK in emulation mode. See the *ProtectToolkit FM SDK Programming Guide* for more information.



FM Certificates

By default, MKFM will not sign with a 512-bit certificate. It is recommended that you create your FM certificates using RSA 2048 instead of RSA512. For example:

ProtectServer	**ctcert c -s0 -k -trsa -z512 -lfm**
ProtectServer 2	**ctcert c -s0 -k -trsa -z2048 -lfm**

MKFM now uses SHA-512 instead of SHA-1. To continue using a legacy 512-bit certificate for signing with a SHA-1 hash, you can use the -3 option of the MKFM command, although this is not recommended.

> **NOTE** For best practice, create your FM certificates using RSA 2048.

FM Debug Logging Using **printf**

Historically, debug logging has been via a simulated serial port 0 and the **dbgprint** routines. These methods are maintained for backwards compatibility.

As a simpler alternative to these methods, ProtectServer 2 adds support for standard C **printf** to write debug messages to the **hsmtrace** log.

> **NOTE** For best practice, update your code to use **printf** instead of serial port 0 and the **dbgprint** routines.

Compile-Time Checking

The _SFNT_FM_ compiler define has been added to enable a compile time check for an FM build.

_SFNT_FM_ is also set for emulation builds, which also have FM_EMU and EMUL set.

The **clock()** Function Standard

The **clock()** function now follows the ANSI/ISO standard of returning CPU time. This differs from Microsoft Windows **clock()** which returns wall time. Elapsed time checks should now use the more accurate THR_BeginTiming and THR_UpdateTiming APIs. Although these APIs existed in PPO, **clock()** on the PSG and K5 behaved like Windows, returning wall time instead of CPU time.

> **NOTE** For best practice, use the THR_BeginTiming and THR_UpdateTiming APIs to perform elapsed time checks.

The **integers.h** Header File Removed From **$(FMSDK)/include**

The **integers.h** header file is no longer included in both **$(FMSDK)/include** and **$(CPROVDIR)/include**. It is now only included in **$(CPROVDIR)/include** since ProtectToolkit-C is required in order to use the FM SDK.

MKFM and CTFM Disable (d) Flag is Now Delete

The tools used to load and manage your FMs, such as MKFM and **ctfm** are the same as in previous releases with the exception of the "Disable" (d) flag. In previous releases, this flag disabled, but did not remove the FM. In this release, this flag will fully remove an FM rather than simply disabling it.

Obsolete FM SDK Copy Function Removed

The **Copy** function, which is obsolete and previously deprecated, has been removed from both the **Cipher** object and the **Hash** object.

> **NOTE** For best practice, remove all instances of the **CipherObj copy** and **HashObj copy** functions from your code.

Support Removed for libfmhost

The **libfmhost** library is no longer supported for HOST applications. The **fmdisp.h** header remains and has wrappers around the supported MD APIs for migration purposes. If your application uses these legacy APIs, your compile output will include deprecated warnings.

Failed FMs

Failed FMs are now deleted instead of being disabled.

Migrating Your FMs

Migrating your existing FMs should require only makefile changes and recompilation. It is recommended that you take advantage of the enhancements to the FM SDK introduced in Release 5 (or higher) when migrating your source.

> **NOTE** FM's compiled using the ProtectToolkit FM SDK with firmware 5.01.00 or newer do not load correctly into HSMs using firmware 5.00.xx.

To migrate your FMs

1. Install the ProtectToolkit-C SDK and FM SDK on a machine or VM running a supported Linux OS.

   > **NOTE** By default, the FM SDK is set to operate in emulation mode. You cannot install FMs while in emulation mode. To install an FM you must change the operating mode to hardware mode. The operating mode is specified when you run the installation script, and can be changed by re-running the script.

2. Update your makefiles, as outlined in "Makefiles" on page 15
3. Review the list of changes and enhancements introduced in the Release 5 (or higher) FM SDK, as detailed in "Summary of Changes and Enhancements" on page 12, to determine whether any additional changes are required. You may also want to update your code at this time to implement some of the enhancements.
4. Build your source.
5. If necessary, fix any compile errors. See "Summary of Changes and Enhancements" on page 12 for a list of the changes introduced in Release 5 (or higher) that may be the source of the errors.

Glossary

A

Adapter

The printed circuit board responsible for cryptographic processing in a HSM

AES

Advanced Encryption Standard

API

Application Programming Interface

ASO

Administration Security Officer

Asymmetric Cipher

An encryption algorithm that uses different keys for encryption and decryption. These ciphers are usually also known as public-key ciphers as one of the keys is generally public and the other is private. RSA and ElGamal are two asymmetric algorithms

B

Block Cipher

A cipher that processes input in a fixed block size greater than 8 bits. A common block size is 64 bits

Bus

One of the sets of conductors (wires, PCB tracks or connections) in an IC

C

CA

Certification Authority

CAST

Encryption algorithm developed by Carlisle Adams and Stafford Tavares

Certificate

A binding of an identity (individual, group, etc.) to a public key which is generally signed by another identity. A certificate chain is a list of certificates that indicates a chain of trust, i.e. the second certificate has signed the first, the

third has signed the second and so on

CMOS

Complementary Metal-Oxide Semiconductor. A common data storage component

Cprov

ProtectToolkit C - SafeNet's PKCS #11 Cryptoki Provider

Cryptoki

Cryptographic Token Interface Standard. (aka PKCS#11)

CSA

Cryptographic Services Adapter

CSPs

Microsoft Cryptographic Service Providers

D

Decryption

The process of recovering the plaintext from the ciphertext

DES

Cryptographic algorithm named as the Data Encryption Standard

Digital Signature

A mechanism that allows a recipient or third party to verify the originator of a document and to ensure that the document has not be altered in transit

DLL

Dynamically Linked Library. A library which is linked to application programs when they are loaded or run rather than as the final phase of compilation

DSA

Digital Signature Algorithm

E

Encryption

The process of converting the plaintext data into the ciphertext so that the content of the data is no longer obvious. Some algorithms perform this function in such a way that there is no known mechanism, other than decryption with the appropriate key, to recover the plaintext. With other algorithms there are known flaws which reduce the difficulty in recovering the plaintext

F

FIPS

Federal Information Protection Standards

FM

Functionality Module. A segment of custom program code operating inside the CSA800 HSM to provide additional or changed functionality of the hardware

FMSW

Functionality Module Dispatch Switcher

H

HA

High Availability

HIFACE

Host Interface. It is used to communicate with the host system

HSM

Hardware Security Module

I

IDEA

International Data Encryption Algorithm

IIS

Microsoft Internet Information Services

IP

Internet Protocol

J

JCA

Java Cryptography Architecture

JCE

Java Cryptography Extension

K

Keyset

A keyset is the definition given to an allocated memory space on the HSM. It contains the key information for a specific user

KWRAP

Key Wrapping Key

M

MAC

Message authentication code. A mechanism that allows a recipient of a message to determine if a message has been tampered with. Broadly there are two types of MAC algorithms, one is based on symmetric encryption algorithms and the second is based on Message Digest algorithms. This second class of MAC algorithms are known as HMAC algorithms. A DES based MAC is defined in FIPS PUB 113, see http://www.itl.nist.gov/div897/pubs/fip113.htm. For information on HMAC algorithms see RFC-2104 at http://www.ietf.org/rfc/rfc2104.txt

Message Digest

A condensed representation of a data stream. A message digest will convert an arbitrary data stream into a fixed size output. This output will always be the same for the same input stream however the input cannot be reconstructed from the digest

MSCAPI

Microsoft Cryptographic API

MSDN

Microsoft Developer Network

P

Padding

A mechanism for extending the input data so that it is of the required size for a block cipher. The PKCS documents contain details on the most common padding mechanisms of PKCS#1 and PKCS#5

PCI

Peripheral Component Interconnect

PEM

Privacy Enhanced Mail

PIN

Personal Identification Number

PKCS

Public Key Cryptographic Standard. A set of standards developed by RSA Laboratories for Public Key Cryptographic processing

PKCS #11

Cryptographic Token Interface Standard developed by RSA Laboratories

PKI

Public Key Infrastructure

ProtectServer

SafeNet HSM

ProtectToolkit C

SafeNet's implementation of PKCS#11. Protecttoolkit C represents a suite of products including various PKCS#11 runtimes including software only, hardware adapter, and host security module based variants. A Remote client and server are also available

ProtectToolkit J

SafeNet's implementation of JCE. Runs on top of ProtectToolkit C

R

RC2/RC4

Ciphers designed by RSA Data Security, Inc.

RFC

Request for Comments, proposed specifications for various protocols and algorithms archived by the Internet Engineering Task Force (IETF), see http://www.ietf.org

RNG

Random Number Generator

RSA

Cryptographic algorithm by Ron Rivest, Adi Shamir and Leonard Adelman

RTC

Real-Time Clock

S

SDK

Software Development Kits Other documentation may refer to the SafeNet Cprov and Protect Toolkit J SDKs. These SDKs have been renamed ProtectToolkit C and ProtectToolkit J respectively. The names Cprov and ProtectToolkit C refer to the same device in the context of this or previous manuals. The names Protect Toolkit J and ProtectToolkit J refer to the same device in the context of this or previous manuals.

Slot

PKCS#11 slot which is capable of holding a token

SlotPKCS#11

Slot which is capable of holding a token

SO

Security Officer

Symmetric Cipher

An encryption algorithm that uses the same key for encryption and decryption. DES, RC4 and IDEA are all symmetric algorithms

T

TC

Trusted Channel

TCP/IP

Transmission Control Protocol / Internet Protocol

Token

PKCS#11 token that provides cryptographic services and access controlled secure key storage

TokenPKCS#11

Token that provides cryptographic services and access controlled secure key storage

U

URI

Universal Resource Identifier

V

VA

Validation Authority

X

X.509

Digital Certificate Standard

X.509 Certificate

Section 3.3.3 of X.509v3 defines a certificate as: "user certificate; public key certificate; certificate: The public keys of a user, together with some other information, rendered unforgeable by encipherment with the private key of the certification authority which issued it"