UEFI
Self-Certification Test (SCT)
Version 2.1

Getting Started Guide

Revision 2.1
May, 2009

Revision History

Revision	Revision History	Date
2.1	Initial release; release revision number matches UEFI Specification number.	5/12/09

Contents

Tables

1 Introduction

1.1 Summary

This document provides detailed instructions for building the UEFI SCT and creating a UEFI SCT Package including UEFI SCT agent and EMS. UEFI SCT is backward compatible with EFI SCT; however, it introduces new UEFI Management Side (EMS) application on the operating system to perform unified remote management for SCT testing. In order to build the UEFI SCT, the EFI Developer Kit (EDK) and the EFI Shell are required. The EDK and the EFI Shell can be downloaded from www.TianoCore.org.

1.2 Overview

This document does not try to explain the UEFI SCT, the EFI Shell, or the EDK. That information is available at www.TianoCore.org and in other material available at the Web site. Instead, this document specifies how to download the code, build it and run it.

- Users installing pre-built binaries may skip ahead to Chapter 5.
- Users who want to use the EMS or who want to add debug information to the SCT or IHV-SCT will need to follow the full instructions to debug and build the tools.

Note: *Changing the actual functionality of the test could prevent accurate and consistent results.*

Other documentation is available at www.TianoCore.org and www.uefi.org. The UEFI SCT includes the following additional documentation:

- *UEFI SCT User Guide* - A user's guide detailing how to use the UEFI SCT application on an IA32, EM64T or Intel® Itanium Family processor target test platform.
- *UEFI SCT Case Specification* - The descriptions for all checkpoints and assertions in the UEFI SCT. This document can be used in conjunction with the GUID definitions and the assertions of the UEFI SCT.

1.3 UEFI SCT Build Tips

1.3.1 IA32 – UefiSct\Platform\IntelTest\UEFI\IA32\

For this example the "UefiSct" is the root directory where the source for the UEFI SCT is unzipped. "nmake uefi21" or "nmake uefi" can be used to make the selection to

build UEFI2.1 SCT or UEFI2.0 SCT. The IA32 Build tip will generate a directory named SctPackage which includes test cases and UEFI SCT Applications. The SctPackage is located at uefi21/uefi directory depending on your build selection.

1.3.2 EM64T – UefiSct\Platform\IntelTest\UEFI\X64\

For this example the “UefiSct” is the root directory where the source for the UEFI SCT is unzipped. “nmake uefi21” or “nmake uefi” can be used to make the selection to build UEFI2.1 SCT or UEFI2.0 SCT. The X64 Build tip generates a directory named SctPackage which includes test cases and UEFI SCT Applications. The SctPackage is located at uefi21/uefi directory depending on your build selection.

1.3.3 Itanium® Family Processor – UefiSct\Platform\IntelTest\UEFI\IPF\

For this example the “UefiSct” is the root directory that the source for the UEFI SCT is unzipped. The IPF Build tip exists to allow the common components to be compiled with a 64-bit compiler. “nmake uefi21” or “nmake uefi” can be used to make the selection to build UEFI2.1 SCT or UEFI2.0 SCT. The IPF Build tip generates a directory named SctPackage which includes test cases and EFI SCT Applications. The SctPackage is located at uefi21/uefi directory depending on your build selection. This Build tip is useful since compiling with a 64-bit compiler helps find non-portable code. It also ensures that any assembly language components have a corresponding Itanium build option. In addition, it helps ensure that non-portable extensions such as _asm {} are not used in common code.

1.4 Conventions Used in This Document

This document uses the typographic and illustrative conventions described below.

1.4.1 Pseudocode Conventions

Pseudocode is presented to describe algorithms in a more concise form. None of the algorithms in this document are intended to be compiled directly. The pseudocode is presented at a level corresponding to the surrounding text.

In describing variables, a *list* is an unordered collection of homogeneous objects. A *queue* is an ordered list of homogeneous objects. Unless otherwise noted, the ordering is assumed to be First in First out (FIFO).

Pseudocode is presented in a C-like format, using C conventions where appropriate. The coding style, particularly the indentation style, is used for readability and does not necessarily comply with an implementation of the *Extensible Firmware Interface Specification* or any of the architecture specifications that are associated with the EDK.

1.4.2 Typographic Conventions

Plain text The normal text typeface is used for the vast majority of the descriptive text in a specification.

Plain text (blue) In the online help version of this specification, any plain text that is underlined and in blue indicates an active link to the cross-reference. Click on the word to follow the hyperlink. Note that these links are *not* active in the PDF of the specification.

Bold In text, **Bold** typeface identifies a processor register name. In other instances, **bold** typeface can be used as a running head within a paragraph.

Italic In text, *Italic* typeface can be used as emphasis to introduce a new term or to indicate a manual or specification name.

BOLD Monospace Computer code, example code segments, and all prototype code segments use a **BOLD Monospace** typeface with a dark red color. These code listings normally appear in one or more separate paragraphs, though words or segments can also be embedded in a normal text paragraph.

Italic Monospace In code or in text, words in *Italic Monospace* indicate placeholder names for variable information that must be supplied (i.e., arguments).

Plain Monospace In code, words in a Plain Monospace typeface that is a dark red color but is not bold or italicized indicate pseudocode or example code. These code segments typically occur in one or more separate paragraphs.

See the master Framework glossary in the Framework Interoperability and Component Specifications help system for definitions of terms and abbreviations that are used in this document or that might be useful in understanding the descriptions presented in this document.

See the master Framework references in the Interoperability and Component Specifications help system for a complete list of the additional documents and specifications that are required or suggested for interpreting the information presented in this document.

The Framework Interoperability and Component Specifications help system is available at the following URL:

http://www.intel.com/technology/framework/spec.htm

2 Quick Start Guide

2.1 Setup Development System

This section describes the steps that are necessary initially to set up the local system in preparation for building a platform (build tip) in the UEFI SCT source tree.

In general we tried to make the project as compiler-neutral as possible. We would like to add support for more compiler types and also support a Linux-hosted development environment in the future, but we need your help.

2.1.1 Install Tools

Several Microsoft tools are required to build the EDK and the UEFI SCT source tree. The following tools must be installed on the development system:

- Microsoft Windows 2000® or Microsoft Windows XP® operating system

For 32-bit Intel architecture (IA32) platform development:

- Microsoft Visual Studio .NET® 2003 Professional (7.1)

For Intel Itanium® processor family development:

- Microsoft® C/C++ Optimizing Compiler Version 13.10.2240.8 for IA64 in the Microsoft Windows Server® 2003 DDK (Build 3790).

If you are compiling for targets based on Itanium architecture, use **`NMAKE.EXE`**, **`LINK.EXE`** and **`LIB.EXE`** from the Microsoft Windows Server 2003 DDK (Build 3790).

2.2 Download Source Code

2.2.1 Downloading UEFI SCT

The first step is to download the source code from www.uefi.org. In order to access the source code, you must register and use the log in information you receive. Don't worry, everyone is welcome to join. To build the UEFI SCT, the EDK and the EFI Shell source code will also need to be downloaded.

2.2.1.1 Downloading Using a Link

Directions for downloading UefiSctPackAll source using a link:

1. Login to www.uefi.org/specs/ and download the SCT source packages.
2. Under Supplemental Test Tools, select the most recent Self Certification Test (SCT)the latest date.
3. Extract the Zip file to a location on your system.

2.3 Downloading Third-party Libraries

To implement remote validation, EMS uses three third-party libraries such as the following: Tcl-Tk, Winpcap and Libnet. The Linet library is released together with UEFI SCT. The ActiveTcl version you use should be 8.4.13-threaded or higher, and Winpcap should be 3.0. You can get third-party Tcl-Tk and Winpcap libraries by following the steps below.

2.3.1 Downloading the Tcl-Tk Library

1. Download the Tcl-Tk installation file from the official website http://www.tcl.tk. Below is a recommended download link from the official site: http://downloads.activestate.com/ActiveTcl/Windows/8.4.13/ActiveTcl8.4.13.0.261555-win32-ix86-threaded.exe
2. Install the downloaded setup file into your OS (Microsoft Windows 2000 or Microsoft Windows XP). For example, you can install ActiveTcl into your C:\Tcl. In this example we will assume the setup folder is C:\Tcl. You can choose any location you like.
3. Copy all content under C:\Tcl\include into C:\Test\Ems\Lib\Tcl\Include\.
4. Copy the .lib files under C:\Tcl\lib into C:\Test\Ems\Lib\Tcl\Lib\.

2.3.2 Downloading the Winpcap Library

1. Download Winpcap installation file and Winpcap Developer's Pack from the official website http://www.winpcap.org/. Below are recommended download links from the official site: http://www.winpcap.org/archive/3.0-WinPcap.exe and http://www.winpcap.org/archive/3.0-WpdPack.zip
2. Install the downloaded setup file into your OS (Microsoft Windows 2000 or Microsoft Windows XP).
3. Extract the 3.0-WpdPack.zip file. For example, you can extract the file into C:\WpdPack. In this example we will assume the extraction folder is C:\WpdPack. You can choose any location you like.
4. Copy all content under C:\WpdPack\Include into C:\Test\Ems\Lib\WpdPack\Include\.
5. Copy the .lib files under C:\WpdPack\Lib into C:\Test\Ems\Lib\ WpdPack\Lib\.

2.3.3 Downloading EDK and EFI Shell

In order to build the EFI SCT, the EDK and the EFI Shell will also need to be downloaded from www.TianoCore.org.

2.3.3.1 Directions for Downloading EDK Source Using the Web Site:

1. Click the Projects tab near the upper left corner of the page.
2. Click on edk in the Name column of the Projects folder.
3. In the Project tools box (upper left of the page), click on Documents & files.
4. Under Documents and files > edk, click on Releases. Two subdirectories display: Development Snapshots and Official Releases.
5. Click Official Releases
6. Click on the file corresponding to the version specified in the Release Notes.
7. Extract the Zip file to the UEFI SCT directory. For this example, extract to C:\Test\UefiSct.

2.3.3.2 Directions for Downloading EFI Shell Source Using the Web Site:

1. Click the Projects tab near the upper left corner of the page.
2. Click on efi-shell in the Name column of the Projects folder.
3. In the Project tools box (upper left of the page), click on Documents & files.
4. Under Documents & files > edk, click on Releases. Two subdirectories display: Development Snapshots and Official Releases.
5. Click Official Releases
6. Click on the file corresponding to the version specified in the Release Notes.
7. Extract the Zip file to the EDK\Other\Maintained\Application directory of UEFI SCT. For this example, extract to C:\Test\UefiSct\Edk\Other\Maintained\Application.

3 Building the UEFI SCT Agent

3.1 Building the UEFI SCT Agent

This section describes the steps that are necessary to set up and build the UEFI SCT Agent. The SCT Package will be produced when building the UEFI SCT Agent. This package will need to be installed onto the Target test platform that is UEFI or "Framework" based. The directory SctPackage will be created under the IA32, the X64, or the IPF efi/uefi build directories.

The UEFI SCT supports three build tips. Each is given a hosting directory within the build tree. Table 1 lists these build tips.

The UEFI SCT also supports a compatible EFI build for all three build tips.

Table 1. Build Tips in the UEFI SCT Source Tree

Build Tip	Description
Platform\IntelTest\UEFI\IA32	The UEFI build environment for IA32.
Platform\IntelTest\UEFI\X64	The UEFI build environment for EM64T-based platforms.
Platform\IntelTest\UEFI\IPF	The UEFI build environment for Itanium Architecture-based platforms.

3.1.1 IA32 Build Tip

Run Visual Studio .NET 2003 Command Prompt to go to the command line environment. The following commands can be used to build the UEFI SCT IA32 tip. If the build is successful, an executable image file **`*.efi`** will be created in the Platform\IntelTest\UEFI\IA32\[uefi21 | uefi]\IA32 directory.

1. cd c:\test\uefisct\platform\inteltest\uefi\ia32\
2. set efi_source=c:\test\uefisct
3. nmake uefi21

3.1.2 X64 Build Tip

To build the UEFI SCT X64 tip, you just need to change the directory from IA32 to X64. The executable image file ***.efi** will be created in the Platform\IntelTest\UEFI\X64\[uefi21 | uefi]\X64 directory.

1. cd c:\test\uefisct\platform\inteltest\uefi\x64\
2. set efi_source=c:\test\uefisct
3. nmake uefi21

3.1.3 IPF Build Tip

To build the UEFI SCT IPF tip, you just need to change the directory from IA32 to IPF. The executable image file ***.efi** will be created in the Platform\IntelTest\UEFI\IPF\[uefi21 | uefi]\IPF directory.

1. cd c:\test\uefisct\platform\inteltest\uefi\ipf\
2. set efi_source=c:\test\uefisct
3. nmake uefi21

3.1.4 EFI Compatibility

For a compatible EFI SCT build, change "nmake uefi21/uefi" to "nmake efi".

3.2 Build Configuration Files

One important feature of the UEFI SCT source tree is that only one configuration file primarily is used to describe the local system for building the various build tips. Table 3 lists the configuration files that can be used.

Table 2. Build Configuration Files

Type of Configuration File	Description
PlatformTools.env	Defines platform-specific requirements. It's located in the platform build directory.

3.3 Environmental Variables

3.3.1 EDK Platform Builds

The only environmental variable that must be defined for building the IA32, X64 or the IPF build tip is **EFI_SOURCE**. **EEI_SOURCE** must be defined to point to the base of the EFI source tree (e.g., **set EFI_SOURCE=C:\Test\UefiSct**).

3.3.2 Microsoft Tools Environment Variables

In order to build IA32 or the IPF Test platform tips, there are specific environment variables that must be set for Visual C++ to work properly. **VSVARS32.BAT** must be run for the C compiler to build the selected tip properly. If the Visual C++ installation was not allowed to update environmental variables, an EDK build may terminate because the C compiler is not in the path. If this scenario occurs, then **VSVARS32.BAT**, which was created when Visual C++ was installed, must be manually run to set up environmental variables for using Visual C++. This file is typically located in the **\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools** directory.

To set up Microsoft Tools Environment:

Launch a “Visual Studio .Net 2003 Command Prompt”. The "Visual Studio .Net 2003 Command Prompt" can be launched from [Start]->[All Programs]->[Microsoft Visual Studio .Net 2003]->[Visual Studio .Net Tools]->[Visual Studio .Net 2003 Command Prompt].

For a system without this option, configure it manually by performing the following steps.

1. Open Windows command line shell (**cmd.exe**) and perform the following operations.
2. Click Start->Run and type in **cmd.exe**. You can also usually find it under Start->All Programs->Accessories->Command Prompt.
3. Before building the EDK you verify that the VC++ compiler is in the path.

   VC++ has a batch file called vcvars32.bat that you can execute

Ensure the VC++ tools are in their path by following the steps below:

1. Right click on My Computer icon
2. Select Properties
3. Select the Advanced tab
4. Click the Environment Variables button near the bottom of the tab
5. Under System variables double click Path. You may have to scroll down to find Path
6. Edit “Variable value:” to include the path for VC++: C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\bin; (This is an example. The path may differ in your system.)

4 Building the EMS

4.1 Building the EMS

This section describes the steps that are necessary to set up and build the EMS (uEfi Management Side). The Ems package will be produced when the UefiSctPackAll source package is downloaded and extracted. The Ems package will need to be built based on three third-party libraries: Tcl-Tk, Winpcap and Libnet. Default Ems packages use ActiveTcl8.4.13-threaded, winpcap3.0 and co-released libnet for building. The corresponding Makefile library includes files and linked .lib files that need to be changed if higher version libraries are used.

Run Visual Studio .NET 2003 Command Prompt to go to the command line environment. The following commands can be used to build the EMS OS application. If the build is successful, an Ems.exe executable file will be generated under the bin directory. Then you can start the application by running "Ems.exe Main.tcl" under bin directory.

1. cd c:\test\ems\src
2. set ems_dir=c:\test\ems
3. nmake
4. cd ..\bin
5. Ems Main.tcl

4.2 Runtime Configuration File

EMS provides one configuration file primarily used by the Ems application to configure the test environment and describe the test methodology. Table 4 lists the configuration files that can be used.

Table 3. Build Configuration Files

Type of Configuration File	Description

Init.conf	Defines default configuration for EMS OS application.
	Below is an example of the configuration item:
	`CASE_ROOT_DIR = c:\`
	`VERBOSE_LEVEL = DEFAULT`
	`COMMUNICATION_TYPE = MNP`
	`TARGETMAC = 00:00:00:00:00:00`
	It's located in the bin directory.

5 Installing the UEFI SCT Agent

5.1 Overview

UEFI SCT contains UEFI SCT agent in EFI side and EMS in OS side. This section describes the steps that are necessary to install UEFI SCT in the EFI side.

5.2 Installing the UEFI SCT Agent

UEFI SCT agent is a shell application, so the EFI Shell environment is a must to run UEFI SCT agent. For EFI1.10 sample code or Tiano implementation, you can setup the shell environment following the below steps in case you don't have built-in shell:

Follow the steps in the EFI Shell Getting Started document to build the image file shell.efi.

1. Copy the shell.efi to the target machine.
2. Add a boot option to the shell.efi just added.
3. Boot to the specified shell environment, and to do the following installation steps according to different target platforms.

The UEFI SCT Agent can be installed on the following platforms:

- IA32 Platform
- Itanium-Based Platform
- EM64T-Based Platform

5.2.1 Installing the UEFI SCT Agent on an IA32 Platform

1. Copy the contents of the IA32 build directory SctPackage to a USB device or IDE-CD.
2. Put the USB or IDE-CD into the USB port or the IDE-CD drive and boot the system to the EFI Shell environment.
3. In EFI Shell environment, change the current drive and directory to the installation CD or USB device drive and root directory.
4. Run installIA32.efi and follow the instructions on the screen.

5.2.2 Installing the UEFI SCT Agent on an Itanium-Based Platform

To install the UEFI SCT Harness from the installation CD:

1. Copy the contents of the IPF build directory SctPackage to a USB device or IDE-CD.
2. Put the USB or IDE-CD into the USB port or the IDE-CD drive and boot the system to the EFI Shell environment.
3. In EFI Shell environment, change the current drive and directory to the installation CD or USB device drive and root directory.
4. Run install64.efi and follow the instructions on the screen.

5.2.3 Installing the UEFI SCT Agent on an EM64T-Based Platform

To install the UEFI SCT Harness from the installation CD:

1. Copy the contents of the X64 build directory SctPackage to a USB device or IDE-CD.
2. Put the USB or IDE-CD into the USB port or the IDE-CD drive and boot the system to the EFI Shell environment.
3. In EFI Shell environment, change the current drive and directory to the installation CD or USB device drive and root directory.
4. Run **`installX64.efi`** and follow the instructions on the screen.