

# Windows Server 2008 Code Integrity (ci.dll) Security Policy

## For FIPS 140-2 Validation

v 3.5  
04/21/08

<b>1</b>	<b>INTRODUCTION</b>	<b>2</b>
1.1	Cryptographic Boundary for CI.DLL	2
<b>2</b>	<b>SECURITY POLICY</b>	<b>2</b>
2.1	CI.DLL Security Policy Elements	2
2.2	Code Integrity security policy	3
<b>3</b>	<b>CI.DLL PORTS AND INTERFACES</b>	<b>4</b>
3.1	CI.DLL export functions	4
3.1.1	CiInitialize()	4
3.1.2	CiValidateImageHeader()	4
3.1.3	CiValidateImageData()	4
3.2	CI.DLL Data Input Interface	5
3.3	CI.DLL Data Output Interface	5
3.4	CI.DLL Control Input Interface	5
3.5	CI.DLL Status Output Interface	5
<b>4</b>	<b>SPECIFICATION OF ROLES</b>	<b>5</b>
4.1	Maintenance Roles	5
4.2	Multiple Concurrent Interactive Operators	5
<b>5</b>	<b>CRYPTOGRAPHIC KEY MANAGEMENT</b>	<b>5</b>
<b>6</b>	<b>CI.DLL SELF TESTS</b>	<b>5</b>
<b>7</b>	<b>ADDITIONAL DETAILS</b>	<b>6</b>

# 1 Introduction

Code Integrity (CI.DLL) is a Windows Server 2008 feature that verifies the integrity of some key Windows Server 2008 binary image files as they are loaded into memory from the disk.

CI.DLL is not a general purpose cryptographic module. It is validated under FIPS 140-2 because it implements cryptographic algorithms and provides the integrity checks of the Windows Server 2008 general purpose cryptographic modules.

## 1.1 Cryptographic Boundary for CI.DLL

The Windows Server 2008 CI.DLL consists of the dynamically-linked library (CI.DLL, versions 6.0.6001.18000) and the ancillary files, ntph.cat and ntpe.cat. The cryptographic boundary for CI.DLL is defined as the enclosure of the computer system, on which CI.DLL is to be executed. The physical configuration of CI.DLL, as defined in FIPS 140-2, is multi-chip standalone.

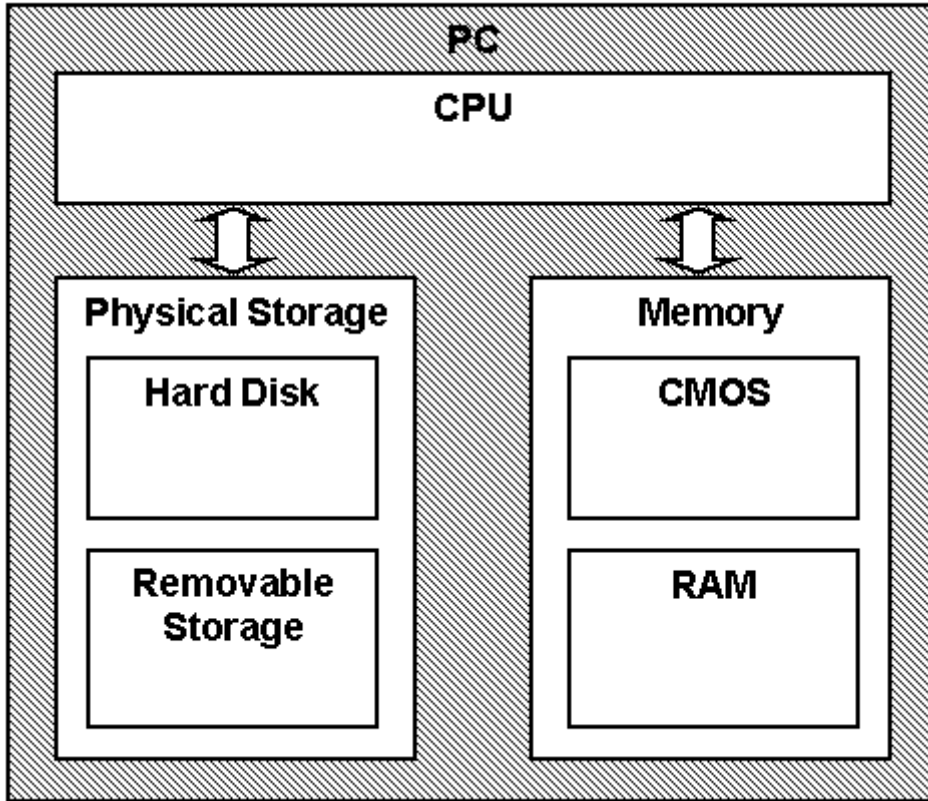
# 2 Security Policy

## 2.1 CI.DLL Security Policy Elements

CI.DLL is considered to be in a FIPS mode of operation when the following rules are followed.

- CI.DLL is supported on Windows Server 2008.
- Windows Server 2008 is an operating system supporting a "single user" mode where there is only one interactive user during a logon session.
- CI.DLL is only in the "Approved mode of operation" when Windows is booted normally, meaning Debug mode has not been enabled and Driver Signing enforcement has not been disabled.
- CI.DLL operates in FIPS mode of operation only when used with the FIPS approved version of Winload OS Loader (winload.exe) (FIPS 140-2 Cert. #1005) operating in FIPS mode

The following diagram illustrates the master components of the CI.DLL module



## 2.2 Code Integrity security policy

- CI.DLL's main service is to verify the integrity of digitally signed drivers and components within the computer (such as ksecdd.sys, bcrypt.dll, rsaenh.dll, dssenh.dll). In addition to this service, WINLOAD.EXE is also provides status services. These status services indicate whether the aforementioned integrity checks passed.
- All services implemented within CI.DLL are available to the User and Crypto officer roles. The User and Crypto officer roles are assumed by the operating system or application processes that will invoke binary image verification in CI.dll. The Window Memory Manager is an example.
- CI.DLL implements the following FIPS-140-2 Approved algorithms.
  - RSA PKCS#1 (v1.5) digital signature verification (Cert. #355)
  - SHA-1 hash (Cert. #753)
- CI.DLL performs the following power-on (start up) self-tests.
  - SHS (SHA-1) Known Answer Test
  - RSA verify using a verify test with a Known Signature of the PKCS#1 v1.5 format
- CI.DLL verifies the integrity of the Server 2008 general purpose cryptographic modules using the following FIPS-140-2 Approved algorithms.
  - RSA PKCS#1 (v1.5) verify with public key
  - SHA-1 hash

Cryptographic bypass is not supported by CI.DLL.

CI.DLL was tested using the following machine configurations:

x86	Microsoft Windows Server 2008 (x86 version) – Dell SC440 (Intel Pentium D 1.8GHz)
x64	Microsoft Windows Server 2008 (x64 version) – Dell SC440 (Intel Pentium D 1.8GHz)
IA64	Microsoft Windows Server 2008 (IA64 version) – HP zx2000 (Intel Itanium2 900 MHz)

### 3 CI.DLL Ports and Interfaces

#### 3.1 CI.DLL export functions

The following list contains all the functions exported by CI.DLL to its callers. Note that CI.DLL is not callable outside the kernel. They are explained further in the subsequent subsections.

- CiInitialize()
- CiValidateImageHeader()
- CiValidateImageData()

CiInitialize() is directly callable by a kernel mode caller. However, the other two are not.

##### 3.1.1 CiInitialize()

CiInitialize() is the function exported by CI.DLL for initializing the image file integrity validation capability of CI.DLL.

As the power-on (start up) function of CI.DLL, CiInitialize() conducts the following power-on (start up) self-tests.

- SHS (SHA-1) Known Answer Test
- RSA verify using a verify test with a Known Signature of the PKCS#1 v1.5 format

If a self test fails, CiInitialize() returns STATUS\_UNSUCCESSFUL. On the other hand, after the successful initialization, CiInitialize() returns a callback structure consisting of the following two functions. A caller subsequently can use these two functions to obtain the image file integrity validation service from CI.DLL.

- CiValidateImageHeader()
- CiValidateImageData()

##### 3.1.2 CiValidateImageHeader()

When a caller (such as the Memory Manager) wants to obtain the set of trusted per-page SHA-1 hashes of an image file, it calls CiValidateImageHeader().

In the case of a Server 2008 general purpose cryptographic module (namely, bcrypt.dll, rsaenh.dll, or dssenh.dll), if CiValidateImageHeader() does not find the set of trusted per-page SHA-1 hashes for the cryptographic module, then the integrity of the cryptographic module is not valid. Subsequently, the Server 2008 Memory Manager does not load any page of the cryptographic module.

Ksecdd.sys is a kernel mode driver, and is loaded into memory all at once (not in a per-page fashion as the other general purpose crypto modules). As a result, when CiValidateImageHeader() is called by the memory manager, the CI\_VALIDATE\_DRIVER\_IMAGE flag is set, and the entire image is validated by verifying an RSA digital signature.

##### 3.1.3 CiValidateImageData()

After calling CiValidateImageHeader() to obtain the set of trusted per-page SHA-1 hashes of an image file, a caller (such as the Memory Manager) would want to know the integrity of a page of the image file. The caller uses CiValidateImageData() to check the page integrity.

If the computed hash matches the identified trusted hash, then CiValidateImageData confirms the integrity of the page. Otherwise, CiValidateImageData() returns STATUS\_INVALID\_IMAGE\_HASH. The Server 2008 Memory Manager does not load invalid pages.

### **3.2 CI.DLL Data Input Interface**

The Data Input Interface for CI.DLL consists of the three CI export functions. Data and options are passed to the interface as input parameters to the CI export functions. Data Input is kept separate from Control Input by passing Data Input in separate parameters from Control Input.

### **3.3 CI.DLL Data Output Interface**

The Data Output Interface for CI.DLL also consists of the three CI export functions. For CiInitialize(), data is returned to its caller as the Callbacks output parameter. For CiValidateImageHeader(), data is returned to its caller as the SePool output parameter.

### **3.4 CI.DLL Control Input Interface**

The Control Input Interface for CI.DLL also consists of the three CI export functions. Options for control operations are passed as input parameters to the CI export functions. The SecureRequired parameter in CiValidateImageHeader() is the only control option provided by CI.DLL in the Control Input Interface.

### **3.5 CI.DLL Status Output Interface**

The Status Output Interface for CI.DLL also consists of the three CI export functions. For each function, the status information is returned to the caller as the return value (eg STATUS\_SUCCESS, STATUS\_UNSUCCESSFUL, STATUS\_INVALID\_IMAGE\_HASH) from the function.

## **4 Specification of Roles**

CI.DLL supports both User and Cryptographic Officer roles (as defined in FIPS-140-2). Both roles have access to all services implemented in CI.DLL through a caller component running in the kernel mode. The module does not provide authentication, as such both roles are implicitly assumed when the services exported by the module are invoked.

### **4.1 Maintenance Roles**

Maintenance roles are not supported by CI.DLL.

### **4.2 Multiple Concurrent Interactive Operators**

There is only one interactive operator during a logon session. Multiple concurrent interactive operators sharing a logon session are not supported.

## **5 Cryptographic Key Management**

CI.DLL does not handle security-relevant information such as secret and private cryptographic key, authentication data or any other protected information. Hence, there is no operation related to any of the below.

- Key generation
- Key output
- Key storage

The only CSPs the module supports are the RSA PKCS#1 public keys used to verify integrity. These CSPs are accessible by both approved roles. The CSPs are stored on the hard-drive. Zeroization is performed by deleting the CI.DLL module, ntph.cat, and ntpc.cat files.

## **6 CI.DLL Self Tests**

CI.DLL performs the following power-on (start up) self-tests when a caller calls its CiInitialize().

- SHS (SHA-1) Known Answer Test

- RSA verify using a verify test with a Known Signature of the PKCS#1 v1.5 format

## 7 Additional details

For the latest information on Windows Server 2008, check out the Microsoft web site at <http://www.microsoft.com>.

<b>CHANGE HISTORY</b>			
<b>AUTHOR</b>	<b>DATE</b>	<b>VERSION</b>	<b>COMMENT</b>
Stefan Santesson	2/14/2008	3.2	Updated for SP1 and merged CMVP review changes

