IQVIA Java Crypto Module

Software Version: 1.0

FIPS 140-2 Non-Proprietary Security Policy

Friday, November 6, 2020

FIPS Security Level: 1
Document Version: 0.3

IQVIA Inc.

6700 Century Avenue, Suite 300

Mississauga, Ontario L5N 6A4

Canada

Phone: +1 (905) 816-5000

Email: appnucleussupport@iqvia.com

http://www.iqvia.com/

Revision History

Version	Modification Date	Modified By	Description of Changes
1.0	2020-06-12	Jonathan Kean	Initial draft.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

This section provides an introduction to the Non-Proprietary Security Policy. It is assumed that readers of this document are familiar with the IQVIA Java Crypto Module from IQVIA Inc. A bulleted list of documents describing the IQVIA Java Crypto Module product is provided below in Section 1.3.

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the IQVIA Java Crypto Module from IQVIA Inc. This Security Policy describes how the IQVIA Java Crypto Module meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-2, which details the U.S. and Canadian Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) website at http://csrc.nist.gov/groups/STM/cmvp.

This document also describes how to run the module in a secure FIPS-Approved mode of operation. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the module. The IQVIA Java Crypto Module is referred to in this document as the IQVIA Java Crypto Module, the crypto-module, or the module.

1.2 References

This document deals only with operations and capabilities of the module in the technical terms of a FIPS 140-2 cryptographic module security policy. More information is available on the module from the following sources:

- The IQVIA website (www.iqvia.com) contains information on the full line of products from IQVIA.
- The CMVP website (http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm) contains contact information for individuals to answer technical or sales-related questions for the module.

1.3 Document Organization

The Security Policy document is one document in a FIPS 140-2 Submission Package. In addition to this document, the Submission Package contains:

- Vendor Evidence document
- Finite State Model document

Other supporting documentation as additional references. This Security Policy and the other validation submission documentation were produced by IQVIA Inc. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Submission Package is proprietary to IQVIA and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact IQVIA.

2 IQVIA Java Crypto Module

2.1 Overview

The IQVIA AppNucleus® Platform enables enterprises to rapidly deploy HIPAA1-compliant mobile health applications to anyone, anywhere, on any mobile device.

The AppNucleus® Platform offers convenient and secure management of critical data and personal information, from a mobile phone, using SMS2, or a tablet or PC3, directly over the Internet and wireless networks, utilizing SOAP4. Furthermore, it provides organizations the necessary tools to deploy customized mobile applications that meet their business and brand needs. The AppNucleus® Platform uses two-factor authentication technology for user and device authentication and state-of-the-art encryption techniques for data protection. Its uniqueness and strengths lies in the ability for organizations to securely publish content to a wide range of different mobile phones. In addition, it enables the customization of both the application configuration as well as the user interface over-the-air without the need of re-deployment. Figure 1, below, illustrates the AppNucleus® Platform.

The security of the AppNucleus® Platform is formed around in-depth security design principles that provide controls at multiple levels of data storage, access and transfer. Although the focus and expertise of IQVIA's AppNucleus team is in designing and implementing high security standards at the software architecture, application and cryptographic levels, The IQVIA AppNucleus team works with its customers to define the overall security strategy, which includes operations security; communication and network security; access controls; business continuity and disaster recovery; and compliance with privacy regulations.

[1] HIPAA – Health Insurance Portability and Accountability Act
[2] SMS – Short Message Service
[3] PC – Personal Computer
[4] SOAP – Simple Object Access Protocol

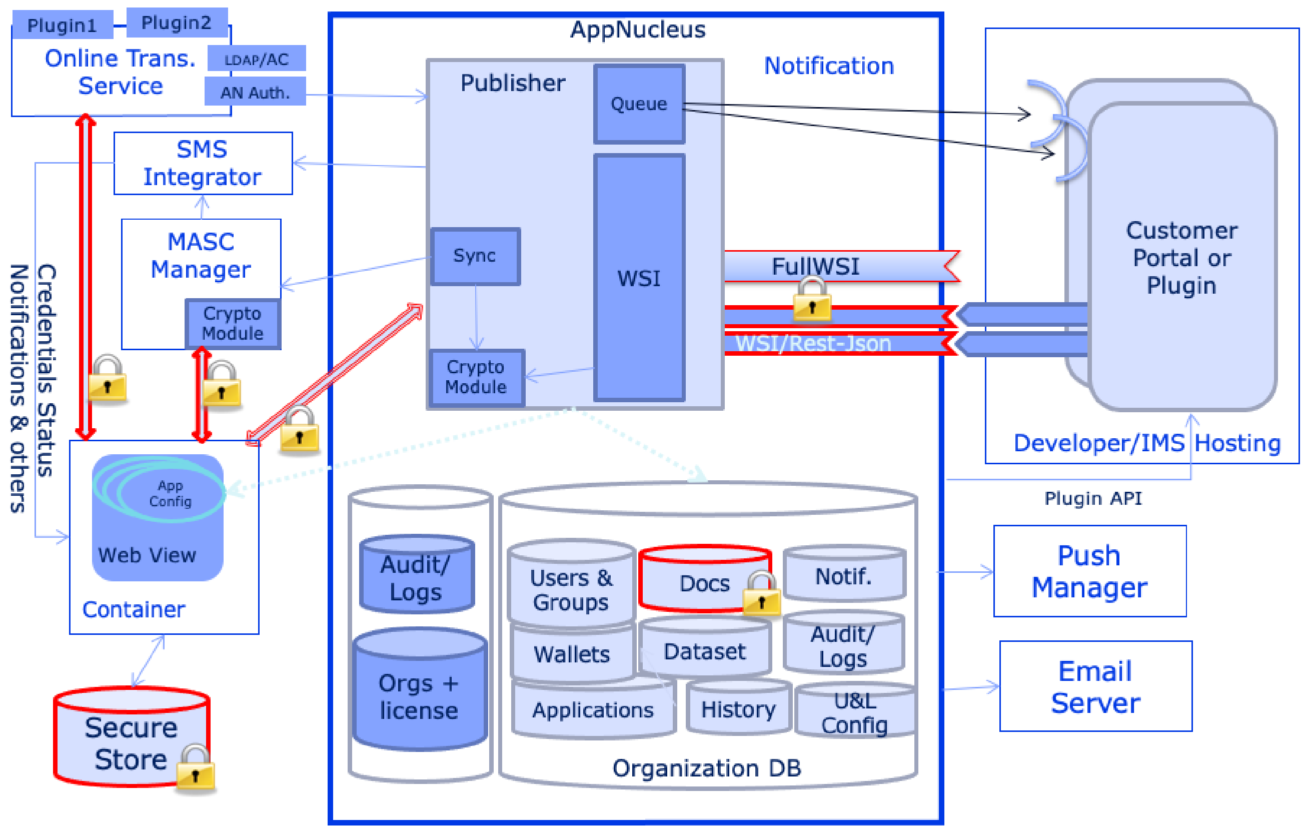


Figure 1. AppNucleus High-Level Architecture

The IQVIA Java Crypto Module (software version 1.0) is the FIPS validated module that provides cryptographic functionality for the AppNucleus® Platform. The IQVIA Java Crypto Module is validated at the following FIPS 140-2 Section levels:

Table 1 – Security Level per FIPS 140-2 Section

Section	Section Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1

Section	Section Title	Level
7	Cryptographic Key Management	1
8	EMI/EMC[5]	1
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

2.2 Module Specification

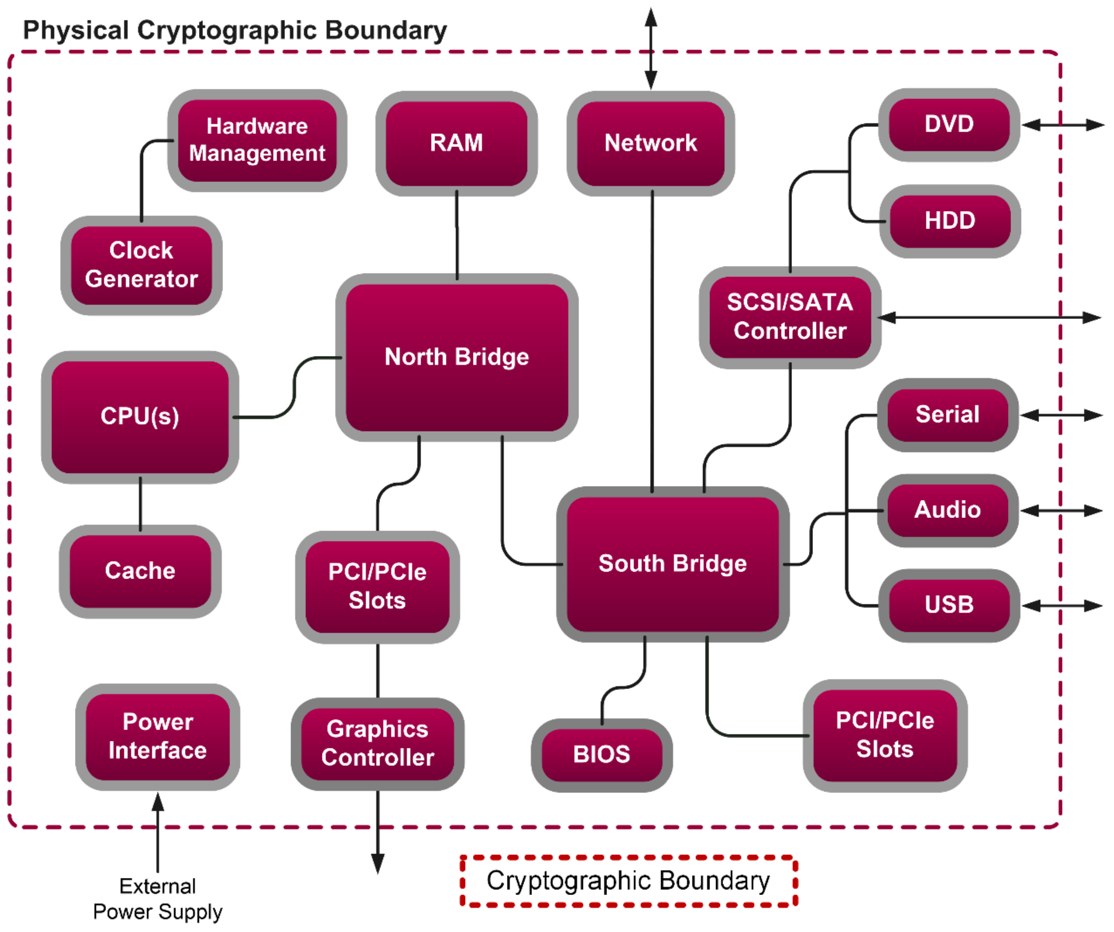
The IQVIA Java Crypto Module is a software module with a multi-chip standalone embodiment. The overall security level of the module is level 1. The following sections will define the physical and logical boundary of the module.

The cryptographic module is a single Java Archive (JAR) binary, named DvSecurityAPI.jar, that provides cryptographic and secure communication services for other applications developed by IQVIA. In this document, those applications will be referred to as the calling application. The module is a standalone cryptographic library that can be called to provide encryption, decryption, hash generation and verification, certificate generation and verification, random bit generation, and message authentication functions.

2.2.1 Physical Cryptographic Boundary

As a software cryptographic module, there are no physical security mechanisms implemented; the module must rely on the physical characteristics of the host platform. The physical cryptographic boundary of the IQVIA Java Crypto Module is defined by the hard enclosure around the host platform on which it executes. Figure 2 below shows the physical block diagram of the module which runs on Microsoft Windows Server 2016 with JDK v1.8 running on a ProLiant BL460c Gen8 with Intel Xeon E5-2640 CPUs.

[5] EMI/EMC – Electromagnetic Interference / Electromagnetic Compatibility



KEY:

BIOS – Basic Input/Output System	PCIe – PCI express
CPU – Central Processing Unit	HDD – Hard Disk Drive
SATA – Serial Advanced Technology Attachment	DVD – Digital Video Disc
SCSI – Small Computer System Interface	USB – Universal Serial Bus
PCI – Peripheral Component Interconnect	RAM – Random Access Memory

Figure 2. GPC Server Block Diagram

2.2.2 Logical Cryptographic Boundary

Figure 3, below, shows a logical block diagram of the module. The module's logical cryptographic boundary (also illustrated in Figure 3) encompasses all functionality provided by the module as described in this document. The module takes the form of a single shared library that can be called by calling applications, executing in the JVM 8, to provide cryptographic services.

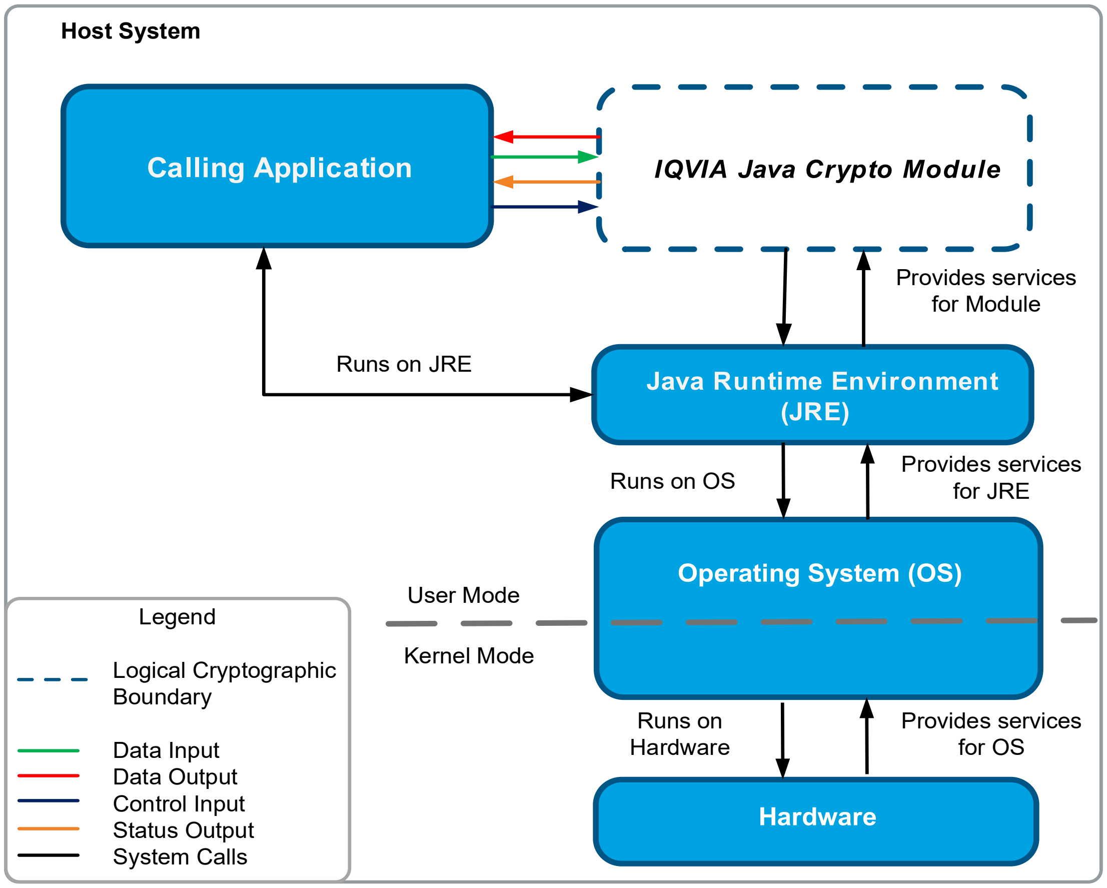


Figure 3. IQVIA Java Crypto Module Logical Block Diagram

2.3 Module Interfaces

The module's logical interfaces exist at a low level in the software as an Application Programming Interface (API). The module's logical and physical interfaces can be categorized into the following interfaces defined by FIPS 140-2:

- Data input
- Data output
- Control input
- Status output
- Power input

As a software module, the module has no physical characteristics. Thus, the module's manual controls, physical indicators, and physical and electrical characteristics are those of the host platform.

All of these physical interfaces of the host platform are separated into logical interfaces defined by FIPS 140-2, as described in the following table:

Table 2 – FIPS 140-2 Logical Interface Mappings

FIPS 140-2 Interface	Physical Interface	Module Interface (API)
Data Input	Network port, Serial port, USB[6] port, DVD[7], SCSI[8]/SATA[9] Controller	Function calls that accept, as their arguments, data to be used or processed by the module
Data Output	Network port, Serial port, USB port, SCSI/SATA Controller	Arguments for a function that specify where the result of the function is stored
Control Input	Network port, Serial port, USB port, Power button	Function calls utilized to initiate the module and the function calls used to control the operation of the module.
Status Output	Network port, Serial port, USB port, Graphics controller, Audio port	Return status for function calls
Power Input	Power Switch	Not Applicable

2.4 Roles and Services

There are two roles in the module (as required by FIPS 140-2) that operators may assume: A Crypto Officer role and a User role.Crypto Officer Role

The Crypto Officer role has the ability to initialize the module, determine module status, and zeroize all module keys and CSPs10. Descriptions of the approved services available to the Crypto Officer role are provided in
Table 3, below. Please note that the keys and CSPs listed in the table indicate the type of access required using the following notation:

- R – Read: The CSP is read.
- W – Write: The CSP is established, generated, modified, or zeroized.
- X – Execute: The CSP is used within an Approved or Allowed security function or authentication mechanism.

Table 3 – Crypto Officer Services

[6] USB – Universal Serial Bus
[7] DVD – Digital Versatile Disc
[8] SCSI – Small Computer Systems Interface
[9] SATA – Serial Advanced Technology Attachment
[10] CSP – Critical Security Parameter

Service	Description	CSP and Type of Access
Initialize module	Loads module and calls power-up self-tests	Integrity check HMAC[11] key – X
ModuleIntegrity.isModuleOperative()	Returns the current mode of the module	N/A
Zeroize key	Format hard drive of host GPC	AES key – W HMAC key – W RSA private key – W RSA public key – W

2.4.2 User Role

The User role has the ability to generate all CSPs supported by the module and perform encryption, decryption, and verification operations with the generated CSPs. Descriptions of the approved services available to the User role are provided in Table 4, below.

Table 4 – User Services

Service	Description	CSP and Type of Access
SHA256Engine.update(), SHA256Engine.complete()	Compute and return a message digest using the SHA-256 algorithm	None
SHA512Engine.update(), SHA512Engine.complete()	Compute and return a message digest using the SHA-512 algorithm	None
ModuleIntegrity.algorithmsTest(), ModuleIntegrity.integrityCheck()	Performs power-up self-tests	Integrity check HMAC key – RX AES key – X RSA private key – X RSA public key – X HMAC keys – X DRBG[12] seed/nonce/entropy input string/key/V – X

[11] HMAC – (Keyed-)Hash Message Authentication Code
[12] DRBG – Deterministic Random Bit Generator

Service	Description	CSP and Type of Access
CTRDRBGEngine.engineNextBytes ()	Generates 128-bit blocks of random bits, up to the requested bit size.	DRBG seed/nonce/entropy input string/key/V – X
HMacEngine.update(), HMacEngine.complete()	Compute and return a message authentication code using HMAC SHA-256, or HMAC SHA-512	HMAC key – RX
AESEngine.process()	Encrypt/decrypt plaintext/ciphertext using AES algorithm specification	AES key – RX
RSAEngine.rsaep()	Encrypt plaintext using the RSA public key of the recipient	RSA public key – RX
RSAEngine.rsadp()	Decrypt ciphertext using the RSA private key of the module	RSA private key – RX
RSAPSSEngine.sign()	Generate a signature for the supplied message using the specified key and algorithm	RSA private key – RX
RSAPSSEngine.verify()	Verify the signature on the supplied message using the specified key and algorithm	RSA public key – RX

2.4.3 Authentication

The module does not support any authentication mechanism. Operators of the module implicitly assume a role based on the service of the module being invoked. Since all services offered by the module can only be used by either the Crypto Officer or the User, the roles are mutually exclusive. Thus, when the operator invokes a Crypto Officer role service, he implicitly assumes the Crypto Officer role. When the operator invokes a User role service, he implicitly assumes the User role.

2.5 Physical Security

The IQVIA Java Crypto Module is a software module, which FIPS defines as a multi-chip standalone cryptographic module. As such, it does not include physical security mechanisms. Thus, the FIPS 140-2 requirements for physical security are not applicable.

2.6 Operational Environment

The IQVIA Java Crypto Module was tested and found compliant with the FIPS 140-2 requirements on the following operating system:

- Microsoft Windows Server 2016 with JDK v1.8 running on a ProLiant BL460c Gen8 with Intel Xeon E5-2640 CPUs.

All cryptographic keys and CSPs are under the control of the host OS, which protects the keys and CSPs against unauthorized disclosure, modification, and substitution. The module only allows access to keys and CSPs through its APIs. The module performs a Software Integrity Test using a FIPS-Approved message authentication code (HMAC SHA-256).

2.7 Cryptographic Key Management

The module implements the FIPS-Approved algorithms listed in Table 5 below.

Table 5 – FIPS-Approved Algorithm Implementations

Algorithm	Certificate Number	Standard
Symmetric Key		
AES: ECB[13] and CBC[14] modes for 128-, 192-, and 256-bit key sizes	#C2127	FIPS 197, SP 800-38A
Asymmetric Key		
RSA PKCS PSS[15] signature generation/verification: 1024-, 2048-bit (w/ SHA-256, SHA-512) 1024 bits is only approved for signature verification.	#C2127	FIPS 186-4
Secure Hashing Standard (SHS)		
SHA-256, SHA-512	#C2127	FIPS 180-4
Message Authentication		
HMAC SHA-256, HMAC SHA-512	#C2127	FIPS 198-1
Random Number Generator		

[13] ECB – Electronic Codebook
[14] CBC – Cipher-Block Chaining
[15] PKCS PSS – Public-Key Cryptography Standard Probabilistic Signature Scheme

Algorithm	Certificate Number	Standard
DRBG (Counter-based) Note that the module does not claim approved key generation	#C2127	SP 800-90A

Additionally, the module utilizes the following non-FIPS-approved but allowed algorithm implementations:

- RSA encrypt/decrypt
  - RSA (key wrapping) provides 112 bits of encryption strength.

The module supports the critical security parameters (CSPs) listed below in Table 6.

Table 6 – List of Cryptographic Keys, Cryptographic Key Components, and CSPs

CSP/Key	CSP/Key Type	Generation / Input	Output	Storage	Zeroization	Use
HMAC Integrity key	HMAC 512 - bit key	Never	Never output from the module	In module binary	N/A	Software integrity check
AES key	AES 128-, 192-, 256-bit key	Input electronically in plaintext	Never output from the module	Plaintext in volatile memory	Reboot	Encryption, decryption
HMAC key	HMAC 512 - , 1024 - bit key	Input electronically in plaintext	Never output from the module	Plaintext in volatile memory	Reboot	Message Authentication with SHS
RSA private key	RSA 2048-bit key	Input electronically in plaintext	Never output from the module	Plaintext in volatile memory	Reboot	Signature generation, decryption
RSA public key	RSA 1024-, 2048-bit key	Input electronically in plaintext	Never output from the module	Plaintext in volatile memory	Reboot	Signature verification, encryption

CSP/Key	CSP/Key Type	Generation / Input	Output	Storage	Zeroization	Use
DRBG seed	Up to 1000-bit entropy value	Internally generated from system entropy	Never output from the module	Plaintext in volatile memory	Reboot	Used to seed the DRBG
DRBG nonce	64 bits of random data	Internally generated from system entropy	Never output from the module	Plaintext in volatile memory	Reboot	Used to seed the DRBG
DRBG entropy input string	DRBG Entropy Input	Internally generated from system entropy	Never output from the module	Plaintext in volatile memory	Reboot	Used to seed the DRBG
DRBG key	128-bit secret value	Derived as per SP 800-90A	Never output from the module	Plaintext in volatile memory	Reboot	DRBG Internal State Value
DRBG V	128-bit secret value	Derived as per SP 800-90A	Never output from the module	Plaintext in volatile memory	Reboot	DRBG Internal State Value

2.8 Self-Tests

2.8.1 Power-Up Self-Tests

The IQVIA Java Crypto Module performs the following self-tests at power-up:

- Software integrity check (HMAC SHA-256)
- Known Answer Tests (KATs)
  - AES Encrypt KAT
  - AES Decrypt KAT
  - RSA PSS KAT for sign/verify
  - SHA-256 HMAC KAT
  - SHA-512 HMAC KAT
  - NIST SP 800-90A DRBG KAT

2.8.2 Conditional Self-Tests

The IQVIA Java Crypto Module performs the following conditional self-tests:

- NIST SP 800-90A DRBG Continuous test
- NIST SP 800-90A DRBG Health Tests
- NDRNG Continuous test

2.9 Mitigation of Other Attacks

This section is not applicable. The modules do not claim to mitigate any attacks beyond the FIPS 140-2 Level 1 requirements for this validation.

The IQVIA Java Crypto Module meets Level 1 requirements for FIPS 140-2. The sections below describe how to place and keep the module in FIPS-approved mode of operation.

Not following the guidelines in the below sections results in the module operating in a non-approved mode of operation.

2.10 Secure Management

The IQVIA Java Crypto Module is distributed only as part of IQVIA's system applications and is not distributed as a separate binary.

2.10.1 Initialization

When the module is installed and initialized the module runs in a FIPS mode of operation. This is achieved by calling a single initialization method. Upon initialization of the module, the module runs its power-up self-tests which includes software integrity test that checks the integrity of the module by using an HMAC SHA-256 digest. If the integrity check succeeds, then the module performs power-up self-tests. When the module passes the integrity test, the module is in its FIPS-Approved mode of operation. If any self-test fails, the module enters a critical error state, ceasing all cryptographic functionality, and throws an exception. The platform must be rebooted to leave the critical error state.
Power-up self-tests can also be performed on demand by invoking the ModuleIntegrity.algorithmsTest() methods or by cycling the power on the host platform.

The module is loaded by applications which make calls to the module as a cryptographic library.

The module is integrated into the application build as a dependency and installed along with the application. The module is loaded and configured during run time when the application makes calls to the module as a cryptographic library.

2.10.2 Management

Since the Crypto Officer cannot directly interact with the module, no specific management activities are required to ensure that the module runs securely; the module only executes in a FIPS-Approved mode of operation. If any irregular activity is noticed or the module is consistently reporting errors, then IQVIA Inc. Customer Support should be contacted.

2.10.3 Zeroization

The module does not persistently store any keys or CSPs. All ephemeral keys used by the module are zeroized upon reboot, or session termination.

2.11 User Guidance

In order to ensure compliance with NIST SP 800-90A, the Crypto-Officer must set a method, CTRDRBGEngine.setMaxTestTimeBetweenInstantiations() to 0 (seconds). This will require that the DRBG health tests be run on every instantiation. Only the module's cryptographic functionalities are available to the User. Users are responsible to use only the services that are listed in Table 4 above. Although the User does not have any ability to modify the configuration of the module, they should report to the Crypto Officer if any irregular activity is noticed.

3 Acronyms

Table 7 – Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
BIOS	Basic Input/Output System
CMVP	Cryptographic Module Validation Program
CCCS	Canadian Centre for Cyber Security
CSP	Critical Security Parameter
DRBG	Deterministic Random Bit Generator
FIPS	Federal Information Processing Standard
HMAC	(Keyed-) Hash Message Authentication Code
KAT	Known Answer Test
NIST	National Institute of Standards and Technology
OS	Operating System
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm