opentext™

OpenText™ Documentum™ Content Management

Trusted Content Services Cryptography and FIPS Guide

Understand the cryptography and the FIPS status of Trusted Content Services-dependent applications.

EDCCS250400-TGD-EN-01

OpenText™ Documentum™ Content Management
Trusted Content Services Cryptography and FIPS Guide
EDCCS250400-TGD-EN-01
Rev.: 2025-Oct-17

This documentation has been created for OpenText™ Documentum™ Content Management CE 25.4.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

FIPS and OpenText Documentum Content Management (CM) Trusted Content Services

This guide lists information about the modules used, where they are used, and the Federal Information Processing Standard (FIPS) 140-2 certifications of the modules. This guide also describes where to obtain information about using FIPS 140-2 modules in the third-party application server that hosts Trusted Content Services-dependent applications.

This guide is intended for system and repository administrators, and for application server administrators. The system administrator installs and owns the OpenText Documentum Content Management (CM) installation. Repository administrators owns and are responsible for one or more repositories. Application server administrators owns and are responsible for the application server that hosts Trusted Content Services-dependent applications. Readers should be familiar with the general principles of client server architecture and networking. In addition, they should know and understand the Windows and Linux operating systems.

1.1 Introduction

The FIPS Publication 140-2 is a U.S. government computer security standard used to validate cryptographic modules. Starting with the 7.0 release, Trusted Content Services installs only FIPS 140-2 validated cryptographic modules. Any exceptions are listed in this document.

FIPS 140-2 certification is handled by third parties who certify the FIPS 140-2 libraries. FIPS 140-2 compliance is achieved by using these FIPS 140-2 certified libraries for our cryptographic functions.

1.2 FIPS compliant and enabled components

This section contains information on FIPS 140-2 compliant and enabled component definitions, converting components from FIPS 140-2 enabled to FIPS 140-2 compliant status, and custom client applications.

1.2.1 Definitions

OpenText Documentum CM components are either FIPS 140-2 compliant or FIPS 140-2 enabled, according to the following definitions:

- A FIPS 140-2 compliant application is one that uses only FIPS 140-2 approved cryptographic algorithms and only a FIPS 140-2 validated implementation of those algorithms. FIPS 140-2 compliant applications require no additional configuration and are enabled to use FIPS-140-2 algorithms on installation.
- A FIPS 140-2 enabled Java application is one that uses the Java Cryptographic Extension and/or Java Secure Sockets Extension APIs for all its cryptographic and Secure Sockets Layer (SSL) functions.

1.2.2 Supported algorithms

The supported algorithms are `AES-128`, `AES-192`, `AES-256`, and `3DES`. `3DES` is used only to decrypt keys for the releases prior to 7.0. For the 7.0 and 7.1 releases, `AES-128` is used. For the 7.2 release and later, `AES-128`, `AES-192`, and `AES-256` are used.

1.2.3 Conversion of components from FIPS-enabled to FIPS-compliant status

A FIPS-enabled application can be made FIPS-compliant by deploying and running the application in a Java Runtime Environment that has been configured to use FIPS validated Java Cryptographic Extension and Java Secure Sockets Extension providers.

To configure for the FIPS-compliant mode, you must first install a Java Cryptography Extension API provider. You can use any compliant provider you want.

Instructions to configure a Java Runtime Environment to use FIPS 140-2 validated Java Cryptographic Extension and Java Secure Sockets Extension vary depending on the vendor. Refer to the vendor documentation for specific instructions and guidelines.

If OpenText™ Documentum™ Content Management Foundation Java API is used for all cryptographic operations, then the application is FIPS 140-2 enabled and can be made FIPS 140-2 compliant by deploying and running it in a properly configured Java Runtime Environment.

Special considerations apply for applications deployed on all applications that are FIPS 140-2 compliant. For example, OpenText™ Documentum™ Content Management Server is FIPS 140-2 compliant, but Documentum Administrator, which must be deployed on the OpenText Documentum Content Management (CM) Server, is only FIPS 140-2 enabled. To ensure that both components are FIPS 140-2 compliant, Documentum Administrator must be deployed and running in a Java Runtime Environment that has been properly configured to use FIPS 140-2 compliant cryptographic algorithms.

1.2.4 FIPS status of components

The following table lists the components and their FIPS-compliant or FIPS-enabled status:

Table 1-1: OpenText Documentum CM components, custom client applications, and FIPS status

Component	FIPS 140-2 status
OpenText™ Documentum™ Content Management Server	Compliant
OpenText™ Documentum™ Content Management Trusted Content Services	Compliant
OpenText™ Documentum™ Content Management Foundation CMIS API	Compliant
OpenText™ Documentum™ Content Management Accelerated Content Services	Compliant
OpenText™ Documentum™ Content Management Branch Office Caching Services	Compliant
OpenText™ Documentum™ Content Management Messaging Service	Enabled
OpenText™ Documentum™ Content Management Foundation Java API	Compliant
OpenText™ Documentum™ Content Management Foundation SOAP API	Compliant
OpenText™ Documentum™ Content Management High-Volume Server	Compliant
OpenText™ Documentum™ Content Management XML Store	Enabled
OpenText™ Documentum™ Content Management Transformation Services	Enabled
OpenText™ Documentum™ Content Management Thumbnail Server	Enabled
OpenText™ Documentum™ Content Management Process Engine	Enabled
OpenText™ Documentum™ Content Management Documentum Administrator	Enabled
Connector for Network Appliance	Does not use FIPS
Content Services for Centera	See "ATMOS and Centera filestores" on page 10
Business Activity Monitor	Enabled
Process Integrator	Enabled

Component	FIPS 140-2 status
Documentum Imaging Services	Enabled
Content Intelligence Services	Enabled
Content Storage Services	Enabled
Custom client applications	See "Custom client applications" on page 8
Unified Content Facilities Client Java	Enabled
Unified Content Facilities Client .NET	Enabled
Unified Content Facilities Server	Enabled
xCelerated Composition Platform	Enabled
xCelerated Management System	Enabled
xPlore	Compliant, with exceptions See "Full-text indexing" on page 11

1.2.5 Custom client applications

Custom applications built using Web Development Kit, OpenText Documentum Content Management (CM) Foundation Java API, OpenText Documentum Content Management (CM) Foundation SOAP API, or any other technologies, could use encryption that does not meet FIPS 140-2 criteria. Therefore, the creator of the custom application is responsible for evaluating how any encryption is used by the application, and whether it meets FIPS 140-2 criteria. Even if all the OpenText Documentum CM-provided software meets the criteria, a custom application could still be an exception.

If you are delegating your cryptographic operations, including hash, to Foundation Java API, it is safe to say that you are FIPS 140-2 enabled. However, if you are using any cryptographic functions, including hash, in your portion of the code, you need to do an analysis for FIPS 140-2 readiness.

If custom APIs or applications are used in the system, then it is necessary to work with third-party vendors to replace the cryptographic libraries used with FIPS 140-2 enabled cryptographic algorithms before the application can be made FIPS 140-2 compliant. Before end-to-end FIPS 140-2 compliance can be achieved, an inventory of all components deployed with the Documentum CM Server must be made to ensure that there are no independent cryptographic operations that require third-party certification for FIPS 140-2 compliance.

Use the following flowchart to determine the FIPS 140-2 readiness of your custom application.

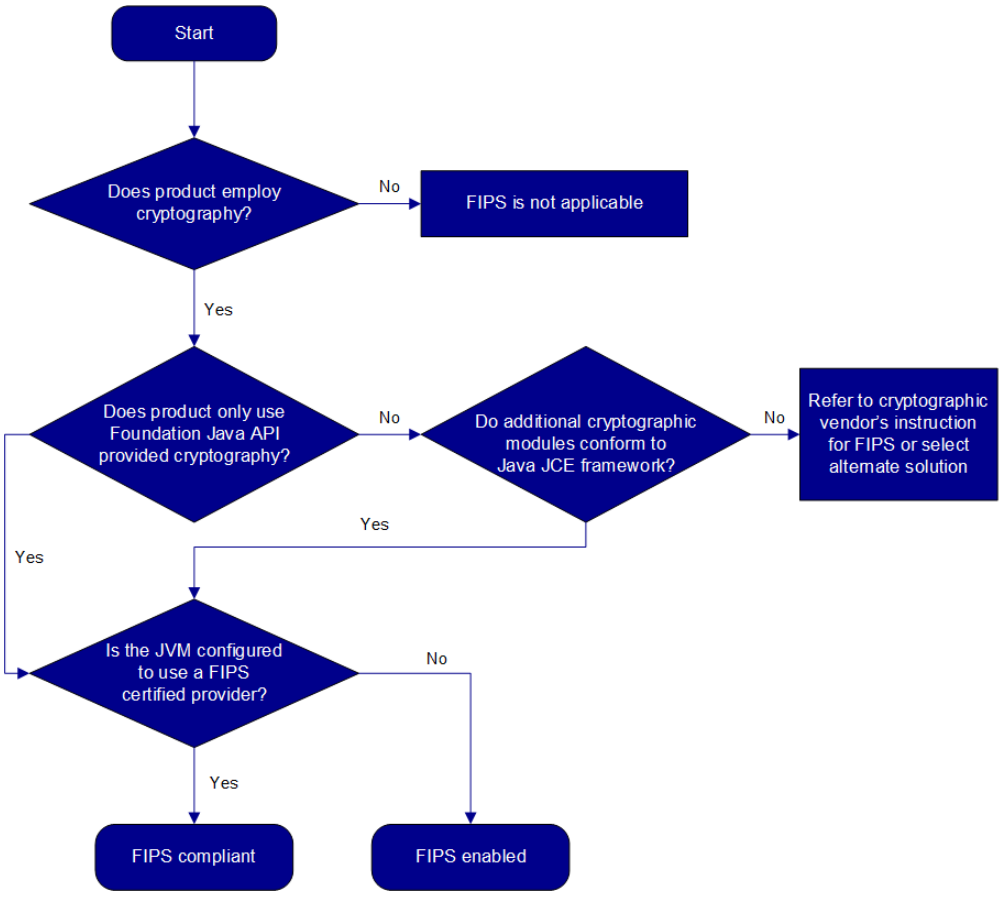

Figure 1-1: FIPS 140-2 classification procedure

Note that component environments and infrastructure can also affect FIPS 140-2 compliance. For example, web applications need to be supported by a FIPS 140-2 compliant application server for end-to-end compliant environments. Refer to the FIPS 140-2 documentation for all your infrastructure components.

1.3 FIPS 140-2 modules

The FIPS 140-2 standard does not specify what data must be encrypted or in what circumstances, so a system that uses FIPS 140-2 modules may encrypt only some data, or only for some cases. As a specific example, Documentum CM Server can be configured to communicate with a connection broker in native mode (unsecured, non-encrypted) or in secured mode (Transport Layer Security (TLS)/SSL encrypted). Only the TLS/SSL mode, and not the native mode, is relevant to FIPS 140-2.

A cryptographic hash algorithm is used in the OpenText Documentum CM system to securely manage and authenticate information by providing a unique hash key for a given unit of information. The OpenText Documentum CM system uses hash keys for signing and validating audit records and esignatures, generating

passphrase-based keys, and managing content deduplication, a process that relies on the content hash key to ensure that stored or embedded content is not duplicated in the repository.

Except for those modules listed in the exceptions section of this chapter, all encryption done by Trusted Content Services enabled Documentum CM Server uses FIPS 140-2 validated modules.

1.4 FIPS 140-2 exceptions

This section identify the cases where encryption modules are used by Trusted Content Services that do not satisfy FIPS 140-2 criteria.

1.4.1 Pre-7.0 Documentum CM Server and applications

Documentum CM Servers and applications prior to the 7.0 releases do not use validated FIPS 140-2 modules. A 7.0 and later version of Documentum CM Server can communicate with Documentum CM Servers, connection brokers, Foundation Java API instances and other applications prior to the 7.0 releases, and will do so using validated FIPS 140-2 cryptographic modules. Content encrypted with Documentum CM Servers prior to the 7.0 release will be decrypted by Documentum CM Server 7.0 using validated FIPS 140-2 cryptographic modules.

1.4.2 Operating environments

The operating environment (for example, application servers) should also be FIPS 140-2 enabled. For example, if you are deploying your application to a server that uses MD5, you will not be able to achieve FIPS 140-2 compliance through that particular application server flavor. You need to ensure that the third parties required by OpenText Documentum CM components are FIPS 140-2 compliant or enabled.

1.4.3 ATMOS and Centera filestores

ATMOS and Centera filestores use encryption that is not based on FIPS 140-2 validated cryptographic modules. Neither ATMOS nor Centera filestores can be used in a way that meets FIPS 140-2 criteria.

1.4.4 Relational databases

Repositories managed by Documentum CM Server use a relational database to store metadata. OpenText links to the vendor library for the supported databases. When OpenText delivers data to the database, OpenText does not control how it is handled. It may be handled in a way that does not meet FIPS 140-2 criteria, dependent on the configuration of the database. You will need to review your database vendor's instructions for any configuration restrictions or procedures.

1.4.5 Full-text indexing

The xPlore query plug-in uses a TLS/SSL implementation from OpenSSL that is not based on FIPS 140-2 validated cryptographic modules. Therefore, when xPlore is configured to use SSL for communication, it cannot be used in a way that meets the FIPS 140-2 criteria.

1.4.6 Electronic signatures

The software for encrypting electronic signatures uses the MD5 encryption and is not based on FIPS 140-2 validated modules. Therefore, electronic signatures cannot be used in a way that meets the FIPS 140-2 criteria.

Chapter 2

Cryptography used in OpenText Documentum CM

OpenText Documentum CM uses cryptography in several different contexts. For example, content files in a filestore can be encrypted, communications between Documentum CM Server and a connection broker can be protected with TLS/SSL connections, and access keys for different elements of Documentum CM Server can be protected cryptographically. Branch Office Caching Services caches content files and can be configured to encrypt them.

2.1 TLS/SSL cryptography

TLS/SSL version 1.2 and 1.3 communication can be used between several OpenText Documentum CM components and also between OpenText Documentum CM components and third-party products. This section describes which modules are used for each component when TLS/SSL communication is used.

2.1.1 Documentum CM Server TLS/SSL

Documentum CM Server uses OpenSSL version 3.5.2 libraries to connect to connection brokers and Foundation Java API instances when using TLS/SSL.

2.1.2 Connection broker TLS/SSL

Connection broker uses OpenSSL version 3.5.2 libraries to connect to Documentum CM Server and to Foundation Java API when using TLS/SSL.

2.1.3 Java Method Server TLS/SSL

Java Method Server uses the Java Sun security provider module for encryption when using TLS/SSL.

2.1.4 Foundation CMIS API TLS/SSL

Foundation CMIS API is installed into an application server, and relies on the JVM installed on that server when using TLS/SSL. To operate in FIPS 140-2 compliant mode you must ensure that the cryptography modules used by your application server JVM are FIPS 140-2 validated.

2.1.5 Foundation Java API TLS/SSL

There are two separate types of Foundation Java API installations. One type of Foundation Java API installation is the embedded Foundation Java API installed when Documentum CM Server is installed. The other type of Foundation Java API installation is a remote installation on a non-Documentum CM Server host. Both types of Foundation Java API installations rely on a Java virtual machine (JVM) to perform encryption.

2.1.5.1 Embedded Foundation Java API

An instance of Foundation Java API is installed during Documentum CM Server installation, on the same host as Documentum CM Server. This instance of Foundation Java API uses the Java Sun security provider module for encryption.

The instance of Foundation Java API uses a JVM that is provided by user while installing Documentum CM Server. An administrator can change the Java security policy files, and install non-FIPS 140-2 Java encryption libraries.

2.1.5.2 Remote Foundation Java API

An instance of Foundation Java API can be installed on a remote, non-Documentum CM Server host. This instance of Foundation Java API uses the JVM installed on that host. To operate in FIPS 140-2 compliant mode, you must ensure that the cryptography modules used by your JVM are FIPS 140-2 validated.

2.1.6 Foundation SOAP API TLS/SSL

Foundation SOAP API is installed into an application server, and relies on the JVM installed on that server when using TLS/SSL. To operate in the FIPS 140-2 compliant mode, you must ensure that the cryptography modules used by your JVM are FIPS 140-2 validated.

2.1.7 Documentum Administrator TLS/SSL

Documentum Administrator is installed into an application server, and relies on the JVM installed on that server when using TLS/SSL. To operate in FIPS 140-2 compliant mode, you must ensure that the cryptography modules used by your JVM are FIPS 140-2 validated.

2.1.8 Accelerated Content Services TLS/SSL

Accelerated Content Services server uses the Java Sun security provider module for encryption when using TLS/SSL. Accelerated Content Services is installed on the Tomcat application server and uses a JVM. An administrator can change the Java security policy files, and install non-FIPS 140-2 Java encryption libraries.

2.1.9 Branch Office Caching Services TLS/SSL

A Branch Office Caching Services server uses the Java Sun security provider module for encryption when using TLS/SSL. Branch Office Caching Services is installed on a Tomcat application server and uses a JVM. An administrator can change the Java security policy files and install non-FIPS 140-2 Java encryption libraries.

2.1.10 Messaging Service TLS/SSL

Messaging Service uses the Java Sun security provider module for encryption when using TLS/SSL. This instance of Messaging Service uses the JVM installed on that host. An administrator can change the Java security policy files, and install non-FIPS 140-2 Java encryption libraries.

2.1.11 Unified Content Facilities server TLS/SSL

Unified Content Facilities server uses the Java Sun security provider module for encryption when using TLS/SSL.

2.1.12 Unified Content Facilities client TLS/SSL

There are two types of Unified Content Facilities clients: Java and .NET

2.1.12.1 Java Unified Content Facilities client

A Java Unified Content Facilities client relies on the JVM installed on the host of the end user when using TLS/SSL. To operate in FIPS 140-2 compliant mode, you must ensure that the cryptography modules used by that JVM are FIPS 140-2 validated.

2.1.12.2 .NET Unified Content Facilities client

A .NET Unified Content Facilities client relies on the .NET runtime installed on the host of the end user when using TLS/SSL. To operate in the FIPS 140-2 compliant mode, you must ensure that the cryptography modules used by that .NET runtime are FIPS 140-2 validated. Install the required modules according to the instructions provided by Microsoft. *Microsoft* documentation contains more information.

2.2 Key generation

Several symmetric keys are used for encrypting and for signing data:

- Application Encryption Key (AEK)
- DocBase Key (DBK)
- Login Ticket Key (LTK)
- File Store Key (FSK)
- File Encryption Key (FEK)

The AEK is the root key for the system.

2.2.1 Documentum CM Server keys

Documentum CM Server uses OpenSSL version 3.5.2 libraries to generate and encrypt the keys.

2.2.2 Foundation Java API keys

Foundation Java API uses the AEK for encryption and decryption of data independently from a repository. A Foundation Java API instance can be either an embedded or remote Foundation Java API.

2.3 File encryption

Content files can be protected by encryption when in transit, using TLS/SSL. These files can also be protected at rest by using encryption either in a filestore, or in a file cache in a Branch Office Caching Services server.

2.3.1 Filestore encryption

Content files are stored in a filestore. Before the files are stored, Documentum CM Server encrypts them using the FEK. The FEK itself is encrypted using FSK. The encrypted FEK is stored in the content header of the encrypted file, and the file is then stored.

Documentum CM Server uses OpenSSL version 3.5.2 libraries to encrypt the files and keys.

2.3.2 Branch Office Caching Services encryption

A Branch Office Caching Services server uses the Java Sun security provider module for encryption for cached content files. Branch Office Caching Services is installed on a Tomcat application server and uses a JVM. An administrator can change the Java security policy files, and install non-FIPS 140-2 Java encryption libraries.