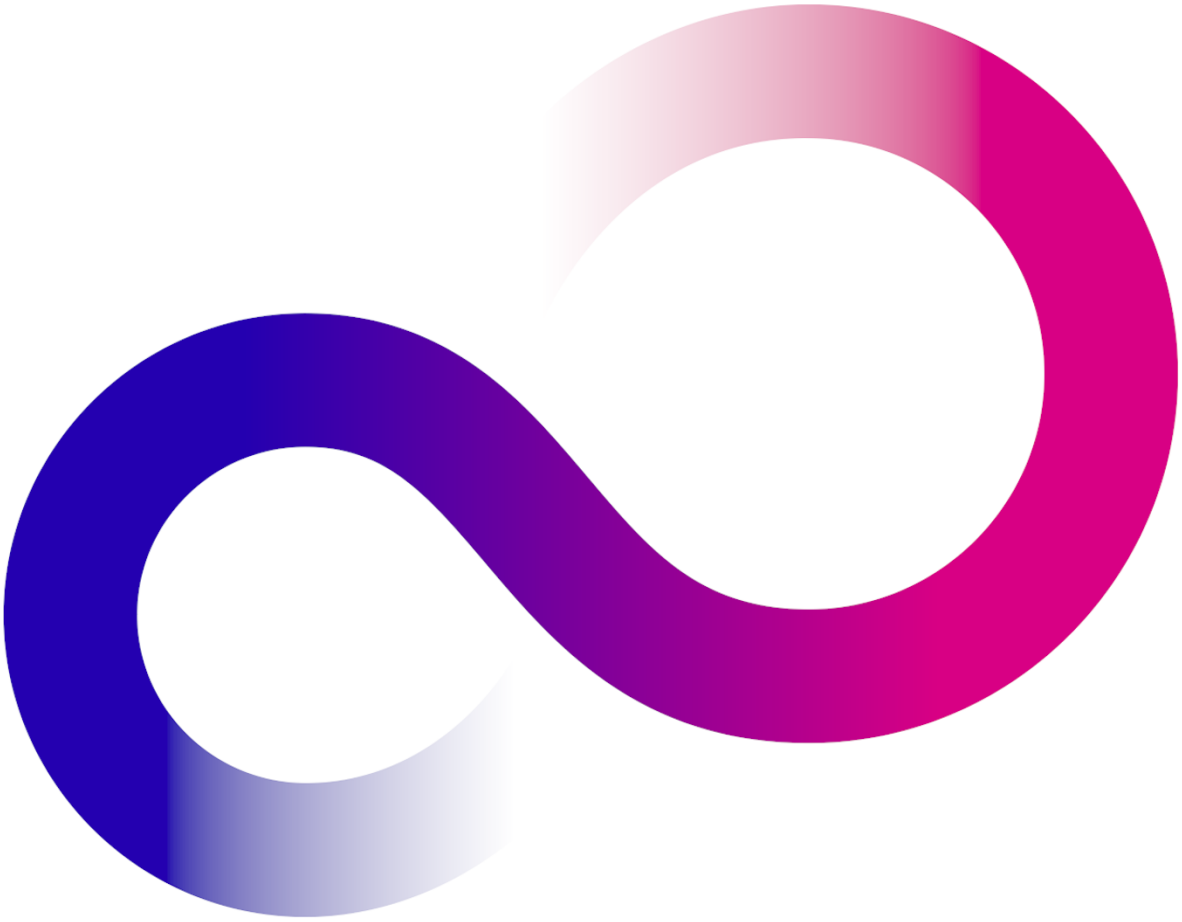
FUJITSU



Fujitsu DSEngine (CryptoModule) für SecDocs 3.2

Version DSEngine 2.0.1.1, Stand: 28.06.2022

Inhaltsverzeichnis

1. Einleitung . . . 2
2. Softwarevoraussetzungen . . . 3
3. Lieferumfang . . . 4
4. Installation . . . 5
    4.1. Installierte Verzeichnisse/Dateien . . . 5
5. Deinstallation . . . 8
6. Konfiguration . . . 9
    6.1. Konfigurationsdatei dsengine-custom.properties . . . 9
    6.2. Konfigurationsdatei log4j2.xml . . . 10
    6.3. Konfigurationsdatei serverSettings . . . 10
7. CryptoModule Anwendung starten/stoppen . . . 11
    7.1. CryptoModule starten . . . 11
    7.2. CryptoModule beenden . . . 11
    7.3. Direkter Aufruf des Skripts cryptoModule . . . 11
8. Skripte der Ablaufumgebung (Verzeichnis bin) . . . 12
    8.1. cmBackupSettings . . . 12
    8.2. cmClearCache . . . 12
    8.3. cmGetPolicyData . . . 13
    8.4. cmGetStatusTL . . . 13
    8.5. cmJConsole . . . 14
    8.6. cmStatus . . . 14
    8.7. cmVersion . . . 15
    8.8. cmCreateServerKeyStore . . . 17
    8.9. cmCreateService / cmCreateServiceFile / cmDeleteService . . . 17
    8.10. cmGC.cron . . . 18
    8.11. cmRemoveLogs . . . 18
    8.12. cmHandleOutOfMemoryError . . . 18
    8.13. cryptoModule . . . 18
9. Anhang . . . 24
    9.1. HTTPS Konfiguration . . . 24
    9.2. Ändern der Default Policy . . . 25
    9.3. Logging . . . 25
    9.4. JRE (Java Runtime Environment) . . . 30
    9.5. Tuning . . . 32
    9.6. Fehlerbehebung . . . 35

Juni 2022

DSEngine 2.0.1 und Runtime Umgebung 1 für SecDocs 3.2

SecDocs Team (secdocs@ts.fujitsu.com)

Chapter 1. Einleitung

Das Fujitsu CryptoModule ist als *Fujitsu Digital Signature Engine (DSEngine)* Teil der *Fujitsu Digital Signature Platform*. Für SecDocs wird der DSEngine in einem vorkonfiguriertem Apache Tomcat 9 Server und zusätzlichen Skripts ausgeliefert.

In den folgenden Kapiteln wird in der Regel kurz der Begriff *CryptoModule* statt *SecDocs Ablaufumgebung für das Fujitsu Digital Signature Engine* verwendet.

Chapter 2. Softwarevoraussetzungen

Die Software ist für das Betriebssystem SuSE SLES15 SP3 64bit (AMD64/x64) oder einer höheren SLES15 SPx Version freigegeben.

Chapter 3. Lieferumfang

Das Fujitsu CryptoModule ist eine Java Enterprise Webanwendung, die in Java programmiert ist und auf einem Tomcat Server abläuft. Die Software wird ablauffähig ausgeliefert und besteht aus den folgenden Komponenten:

- Fujitsu DSEngine 2.0.1
- Apache Tomcat 9.0.64
- Adoptium OpenJDK8 Update 332
- cacerts_mozilla.jks (Stand: 26.04.2022)
  Datei cacert-2022-04-26.pem
  Quelle: CA certificates extracted from Mozilla
- Fujitsu CryptoModule (DSEngine) 2.0.1 Runtime Umgebung 1 für SecDocs V3.2
  (Skripte und Tomcat Server Konfiguration)

Chapter 4. Installation

Die CryptoModule Software wird vom Benutzer root mit Hilfe des RPM Verfahrens installiert:

```
# rpm -ivh secdocscm-2.0.1.1-1.x86_64.rpm
```

Bei der Installation wird (soweit nicht schon auf dem Rechner vorhanden) der Linux Benutzer secdocs und die zugehörige Linux Gruppe secdocs angelegt. Der neu angelegte Benutzer hat das Homeverzeichnis */home/secdocs*.

Mit dem folgenden Kommando kann man sich die Dateien anzeigen lassen, die als Konfigurationsdateien markiert sind.

```
$ rpm -qc -p secdocscm-2.0.1.1-1.x86_64.rpm
/home/secdocs/CryptoModule/bin/setCryptoModuleEnv.sh
/home/secdocs/CryptoModule/server/conf/catalina.policy
/home/secdocs/CryptoModule/server/conf/catalina.properties
/home/secdocs/CryptoModule/server/conf/context.xml
/home/secdocs/CryptoModule/server/conf/dsengine/dsengine-custom.properties
/home/secdocs/CryptoModule/server/conf/dsengine/log4j2.xml
/home/secdocs/CryptoModule/server/conf/dsengine/serverSettings
/home/secdocs/CryptoModule/server/conf/jaspic-providers.xml
/home/secdocs/CryptoModule/server/conf/logging.properties
/home/secdocs/CryptoModule/server/conf/server.xml
/home/secdocs/CryptoModule/server/conf/tomcat-users.xml
/home/secdocs/CryptoModule/server/conf/web.xml
```

Wurden diese Dateien vor der Deinstallation verändert, werden sie nicht gelöscht sondern bleiben mit der zusätzlichen Endung *.rpmsave* erhalten.

4.1. Installierte Verzeichnisse/Dateien

Nach dem Ausführen der obigen Schritte ist die CryptoModule Installation abgeschlossen und man hat im Homeverzeichnis der Kennung secdocs die folgenden Dateiverzeichnisse:

- CryptoModule
- CryptoModule/bin
  Die Skripte der Fujitsu Ablaufumgebung für das CryptoModule
- CryptoModule/data
  Dieses Verzeichnis enthält die Dateien
  cacerts_mozilla.jks
  und die CryptoModule Policy Dateien
  policy.xsd (XML Schema für die CryptoModule Policy Dateien)
  policy.xml (Standard Policy für die Signaturprüfung)
- CryptoModule/docs/licenses
  Dieses Verzeichnis enthält die Lizenztexte der im CryptoModule verwendeten Open Source Komponenten. In der Datei ThirdPartyLicenseReadme.txt findet man eine Angabe aller

verwendeten Komponenten

- CryptoModule/OpenJDK
  OpenJDK Ablaufumgebung
- CryptoModule/OpenJDK.tar.gz
  Ein TAR-Archiv der OpenJDK Ablaufumgebung
- CryptoModule/OpenJDK.tar.gz.sha256
  Der SHA-256 Hashwert der Datei *CryptoModule/OpenJDK.tar.gz*.
- CryptoModule/server
- CryptoModule/tomcat
  Entpackte Apache Tomcat 9 Software
- CryptoModule/tomcat.tar.gz
  Ein TAR-Archiv der Tomcat 9 Software
- CryptoModule/tomcat.tar.gz.sha256
  Der SHA-256 Hashwert der Datei *CryptoModule/tomcat.tar.gz*.

Hinweis: Wenn nicht anders angegeben, ist bei allen weiteren Verzeichnisangaben das Installationsverzeichnis */home/secdocs/CryptoModule* das aktuelle Verzeichnis.

Beispiel: ist das Verzeichnis *server/conf* genannt, ist das Verzeichnis */home/secdocs/CryptoModule/server/conf* gemeint.

4.1.1. Backup der CryptoModule Webanwendung

Im Verzeichnis *CryptoModule/server* befinden sich die Dateien

- ROOT.tar.gz
- ROOT.tar.gz.sha256

Die Datei *ROOT.tar.gz* ist ein TAR-Archiv des Verzeichnisses *webapps/ROOT*. Die Datei ROOT.tar.gz.sha256 enthält den SHA-256 Hashwert (der aktuelle Hashwert steht in der Freigabemitteilung) der TAR-Datei. Falls es Zweifel an der Integrität der Daten im Verzeichnis *webapps/ROOT* gibt, kann man dieses Verzeichnis löschen und die TAR-Datei im Verzeichnis *webapps* entpacken:

```
cd CryptoModule/server/webapps
rm -fr ROOT
tar xvf ../ROOT.tar.gz
```

Hinweis: man kann das alte ROOT Verzeichnis auch aufbewahren. Es darf dann jedoch nicht (egal unter welchem Namen) im Verzeichnis *webapps* liegen.

Zusätzlich sind die JAR-Dateien der CryptoModule Implementierung versiegelt. Eine Manipulation dieser

Dateien wird beim Laden der Klassen von der JVM (Java Virtual Machine) erkannt .

Chapter 5. Deinstallation

Die CryptoModule Software kann vom Benutzer root mit Hilfe des RPM Verfahrens deinstalliert werden:

```
# rpm -e secdocscm-2.0.1.1-1
```

Chapter 6. Konfiguration

In den folgenden 3 Dateien kann die Standardkonfiguration geändert werden:

- server/conf/dsengine/dsengine-custom.properties
  Konfiguration des CryptoModule.
- server/conf/dsengine/log4j2.xml
  Loggingausgaben des CryptoModules in der Datei *server/logs/cryptoModule.log*.
- server/conf/dsengine/serverSettings
  Konfiguration der Tomcat basierten Ablaufumgebung.

6.1. Konfigurationsdatei dsengine-custom.properties

Die Datei enthält alle Konfigurationsparameter des Fujitsu CryptoModules mit ihrem Defaultwert in der Form

Name=Wert

Zeilen, die mit einem # beginnen, werden nicht bewertet.

Da die meisten Parameter mit ihrem Defaultwert angegeben sind, beginnen die meisten Zeilen mit einem #.

Für den Produktivbetrieb ist der folgenden Block in dieser Datei von Interesse.

6.1.1. Proxy Settings

Normalerweise laufen Anwendungen in einer Produktivumgebung hinter einer Firewall. Damit das CryptoModule auf die benötigten externen Daten via HTTP und/oder HTTPS zugreifen kann, muss der zu verwendende Proxy gesetzt werden. Ob für die Proxynutzung ein User (plus Password) benötigt wird, hängt vom verwendeten Proxy ab.

```
# proxy.http.enabled=false
# proxy.http.host=
# proxy.http.port=0
# proxy.http.user=
# proxy.http.password=
# proxy.http.exclude=
#
# proxy.https.enabled=false
# proxy.https.host=
# proxy.https.port=0
# proxy.https.user=
# proxy.https.password=
# proxy.https.exclude=
```

6.2. Konfigurationsdatei log4j2.xml

Diese Datei enthält die Log4J2 Konfiguration für die Loggingausgaben des CryptoModules in der Datei *server/logs/cryptoModule .log*.

6.3. Konfigurationsdatei serverSettings

Die Datei enthält alle Konfigurationsparameter der Tomcat Server Instanz mit ihrem Defaultwert in der Form

Name=Wert

Zeilen, die mit einem # beginnen, werden nicht bewertet werden. Die kommentierten Werte entsprechen den Defaultwerten.

Für den Produktivbetrieb sind die folgenden Parameter in dieser Datei von Interesse:

- CM_PORT_BASE=18000
  18000 ist die Basisnummer für die verwendeten Portnummern:
  Shutdown Port: CM_PORT_BASE + 5 : 18005
  HTTP Port: CM_PORT_BASE + 80 : 18080
  HTTPS Port. CM_PORT_BASE + 443 : 18443
- CM_PORT_OFFSET=0
  Sind die Standard Ports (18005, 18080, 18443) bereits belegt, kann man hier einfach einen Offset angeben, d.h. auf alle Portnummern wird einfach der Offsetwert addiert.

Mit den Standardwerten ist die Webseite des CryptoModules über die folgende(n) URL(s) zu erreichen:

- http://localhost:18080/
- https://localhost:18443/
  (nur wenn der HTTPS Connector aktiviert wurde, s. Kapitel HTTPS Konfiguration)

Eine Nutzung des CryptoModules von einem anderen Rechner aus ist aus Gründen der Sicherheit nicht konfiguriert.

In besonderen Fällen (z.B: hohe Last oder große Datenmengen) kann es nötig sein, die folgenden Parameter zu verändern:

```
CM_JAVA_START_MEM="4g"
CM_JAVA_MAX_MEM="4g"
CM_JAVA_MEM_THREAD_STACK="-Xss512k"
CM_JAVA_GC="-XX:+UseParallelGC -XX:+UseParallelOldGC"
```

Weitere Informationen zu diesen Parametern findet man im Kapitel “Tuning” (Tomcat JVM Heap Size und Tomcat JVM Garbage Collection).

Chapter 7. CryptoModule Anwendung starten/stoppen

Nach der RPM Installation ist der systemd Service cryptoModule.service eingerichtet. D.h. bei einem Neustart der Maschine wird die Fujitsu CryptoModule Anwendung automatisch gestartet. Mit Hilfe dieses Service kann der Administrator (Kennung root) die CryptoModule Anwendung auch einfach manuell starten und beenden.

7.1. CryptoModule starten

```
# systemctl start cryptoModule.service
```

7.2. CryptoModule beenden

```
# systemctl stop cryptoModule.service
```

Der CryptoModule Administrator (Kennung secdocs) kann auch direkt die folgenden Aufrufe ausführen:

7.3. Direkter Aufruf des Skripts cryptoModule

```
$ bin/cryptoModule start
$ bin/cryptoModule stop
```

Hinweis: Die CryptoModule Instanz sollte entweder über den zugehörigen systemd Service (systemctl ... cryptoModule.service) oder aber über das Skript *cryptoModule* gestartet und gestoppt werden.

Chapter 8. Skripte der Ablaufumgebung (Verzeichnis bin)

- Alle Skripte in diesem Verzeichnis rufen das Konfigurationsskript *bin/setCryptoModuleEnv.sh* auf. Dadurch ist sichergestellt, dass alle Skripte dieselbe Konfiguration verwenden.
- Alle änderbaren Informationen kann man in der Datei *server/conf/dsengine/serverSettings* setzen, die in dieser Datei aufgerufen wird.
- Die Skripte bieten die Option -h (auch --help) an, um einen Hinweis auf die Aufrufsyntax zu bekommen.

8.1. cmBackupSettings

Das Skript sichert alle im RPM Paket als Konfigurationsdateien markierte Dateien:

Die zentrale Konfigurationsdatei für die Ablaufumgebung

```
bin/setCryptoModuleEnv.sh
```

Die Settingsdateien für das CryptoModule und die Ablaufumgebung

```
server/conf/dsengine/dsengine-custom.properties
server/conf/dsengine/log4j2.xml
server/conf/dsengine/serverSettings
server/conf/dsengine/serverKeyStore.p12 (falls vorhanden)
```

Die Konfigurationsdateien der Tomcat Instanz

```
server/conf/catalina.policy
server/conf/catalina.properties
server/conf/context.xml
server/conf/jaspic-providers.xml
server/conf/logging.properties
server/conf/server.xml
server/conf/tomcat-users.xml
server/conf/web.xml
```

Die Dateien werden in das Verzeichnis *data/backup/YYYYMMDDhhmmss* kopiert.

Hinweis 1: Das Verzeichnis *data/backup* wird erst beim 1. Backup angelegt und (falls vorhanden) bei der Deinstallation der Software nicht gelöscht.

Hinweis 2: nach jeder Änderung der Konfiguration des CryptoModules empfiehlt es sich, mit dem Skript *bin/cmBackupSettings* ein Backup der aktuellen Konfigurationdateien zu erstellen.

8.2. cmClearCache

Löscht die Daten in den Tomcat internen Verzeichnissen *server/conf/Catalina*, *server/temp* und

server/work. Zusätzlich wird, falls vorhanden, die Datei *cryptoModule.pid* gelöscht.

Dieses Skript läuft nur ab, wenn die Tomcat Instanz **nicht** läuft.

8.3. cmGetPolicyData

In einer laufenden Anwendung kann man sich mit diesem Skript die Default Policy XML Datei und das zugehörige Schema herunterladen.

$ cmGetPolicyData -h

```
Usage: cmGetPolicyData [<options>]

Get policy data from the CryptoModule (DSEngine)

Options:
-h|--help : show this help and exit
-V|--version : show version information and exit
-H|--host : set host (default: 127.0.0.1)
-p|--port : set port (default: 18080)
-s|--secure : use HTTPS instead of HTTP
--xml : get default policy (default)
--xsd : get policy schema
```

Dieses Skript läuft nur ab, wenn die Tomcat Instanz läuft.

Die Default Policy Dateien findet man auch im Verzeichnis *data*:

- *policy.xml*
  Default Policy für die Signaturprüfung
- *policy.xsd*
  Policy Schema Datei

8.4. cmGetStatusTL

In einer laufenden Anwendung kann man sich mit diesem Skript den (Lade-) Status der TLs (Trust Lists) anzeigen lassen.

```
$ cmGetStatusTL -h

Usage: cmGetStatusTL [<options>] [<country> [<LOTL>]]
       Get TL status data from the CryptoModule (DSEngine)

    Options:
    -h|--help    : show this help and exit
    -V|--version : show version information and exit
    -H|--host    : set host (default: 127.0.0.1)
    -p|--port    : set port (default: 18080)
    -s|--secure  : use HTTPS instead of HTTP
    --lotls      : status of configured LOTL(s)
    --tls        : short state description of all TLs
    --json       : output in JSON format (default)
    --xml        : output in XML format

    country : territory of TL
              (default: DE)
    LOTL    : territory of LOTL
              (default: EU)
```

Ist eine der Optionen *--lotls* oder *--tls* angegeben, wird die Angabe von country und LOTL ignoriert.

Werden keine Parameter (country/LOTL) angegeben, wird der Status der deutschen Trust List ausgegeben. Wird als country ein Ländercode angegeben (z.B. ES für Spanien), wird die Trust List des entsprechenden Landes ausgegeben. Normalerweise ist im CryptoModule nur eine LOTL (für den Bereich EU) konfiguriert. Sind mehrere LOTLs konfiguriert, kann man zusätzlich zum Ländercode auch den Bezeichner für die zu nutzende LOTL angeben.

Beispiel:

```
$ cmGetStatusTL | python3 -m json.tool
{
    "url": "https://tl.bundesnetzagentur.de/TL-DE.XML",
    "lastSuccessSynchronizationTime": "Thu Feb 10 15:13:40 CET 2022",
    "downloadCacheState": "SYNCHRONIZED",
    "downloadIsError": false,
    "downloadLastStateTransitionTime": "Thu Feb 10 14:13:26 CET 2022",
    "territory": "DE",
    "parsingCacheState": "SYNCHRONIZED",
    "parsingIsError": false,
    "parsingLastStateTransitionTime": "Thu Feb 10 14:13:26 CET 2022",
    "issueDate": "Thu Dec 16 10:30:09 CET 2021",
    "nextUpdateDate": "Thu Jun 16 09:30:09 CEST 2022",
    "validationCacheState": "SYNCHRONIZED",
    "validationIsError": false,
    "validationLastStateTransitionTime": "Thu Feb 10 14:13:26 CET 2022",
    "indication": "TOTAL_PASSED",
    "signingTime": "Thu Dec 16 09:29:48 CET 2021",
    "signingCertificateSubject": "countryName=DE,organizationName=Federal Network Agency,commonName=German Trusted List Signer 5",
    "signingCertificateIssuer": "countryName=DE,organizationName=Federal Network Agency,commonName=German Trusted List Signer 5",
    "signingCertificateSerial": "23",
    "signingCertificateNotBefore": "Tue Apr 27 14:18:25 CEST 2021",
    "signingCertificateNotAfter": "Thu Apr 27 14:18:25 CEST 2023"
}
```

8.5. cmJConsole

Dieses Skript ruft die OpenJDK JConsole (*CryptoModule/OpenJDK/bin/jconsole*) mit dem JTop Plugin auf.

Weitere Details zur Nutzung des Tools JConsole findet man im Kapitel Using JConsole im Java SE Monitoring and Management Guide.

8.6. cmStatus

Mit Hilfe dieses Skripts kann man den Status des CryptoModules überprüfen.

```
$ cmStatus -h

Usage: cmStatus [<options>]
       Show run state of the CryptoModule (DSEngine)

    Options:
    -h|--help    : show this help and exit
    -V|--version : show version information and exit
    -H|--host    : set host (default: 127.0.0.1)
    -p|--port    : set port (default: 18080)
    -s|--secure  : use HTTPS instead of HTTP
    -S|--silent  : silent mode (different exit codes)
    --tsl-check  : show date/time of last TSL refresh
```

Bei einem allgemeinen Fehler (falsche Parameter) gibt das Skript eine Fehlermeldung aus und beendet sich mit dem Exit Code 1.

Beim Ablauf können die folgenden Meldungen und Exit Codes auftreten:

- cmStatus: CryptoModule (DSEngine) process not running
  Exit Code: 2
  Der JVM Linux Prozess läuft nicht (kann nur bei einer lokalen Abfrage auftreten).
- cmStatus: CryptoModule (DSEngine) is starting
  Exit Code: 3
  Es gibt den JVM Linux Prozess, aber der Tomcat Server ist noch nicht initialisiert und antwortet deshalb noch nicht auf Anfragen.
- cmStatus: CryptoModule (DSEngine) is not available
  Exit Code: 4
  Die CryptoModule Web Anwendung ist noch nicht initialisiert.
- cmStatus: running server is not a CryptoModule (DSEngine) instance
  Exit Code: 5
  Es läuft zwar eine Web Anwendung, aber es handelt sich dabei nicht um das CryptoModule.
- cmStatus: error in querying CryptoModule (DSEngine)
  Exit Code: 6
  Beim Abfragen der Versionsinformation des CryptoModules ist ein Fehler aufgetreten. D.h.: wahrscheinlich handelt es sich bei der laufenden Web Anwendung nicht um das OpenLimit CryptoModule.
- cmStatus: CryptoModule (DSEngine) is running but TSL is not yet loaded
  Exit Code: 7
  Die CryptoModule Web Anwendung ist ablaufbereit aber die TSL wurde noch nicht vollständig geladen.
  Dieser Exit Code wird nur erzeugt, wenn die Option *--tsl-check* beim Aufruf angegeben wurde.
- cmStatus: CryptoModule (DSEngine) is running
  Exit Code: 0
  Die CryptoModule Web Anwendung kann von der SecDocs Anwendung genutzt werden.

8.7. cmVersion

Mit diesem Skript kann für ein laufendes CryptoModule die Versionsinformation ausgegeben werden.

```
$ cmVersion -h
Usage: cmVersion [<options>]
       Get version information from a running CryptoModule (DSEngine)

       Options:
         -h|--help    : show this help and exit
         -V|--version : show version information and exit
         -H|--host    : set host (default: 127.0.0.1)
         -p|--port    : set port (default: 18080)
         -s|--secure  : use HTTPS instead of HTTP
         --soap       : SOAP request/response (instead of REST)
```

Dieses Skript läuft nur ab, wenn die Tomcat Instanz läuft.

Beispiel:

```
$ cmVersion
{"name":"Digital Signature Engine","version":"2.0.1","majorVersion":"2","minorVersion":"0","dssVersion":"5.7","oem":"fj-oem-2.0.1
(5.7)","vendor":"Fujitsu","buildBranch":"RELEASE-2.0.1","buildHash":"54f029c","buildDirty":"","buildTime":"2022-05-
10T14:08:18","buildTags":"","buildVersion":"2.0.1"}
```

Ist Python 3 (python3) installiert, kann man die Ausgabe einfach formatieren:

```
$ cmVersion | python3 -m json.tool
{
    "name": "Digital Signature Engine",
    "version": "2.0.1",
    "majorVersion": "2",
    "minorVersion": "0",
    "dssVersion": "5.7",
    "oem": "fj-oem-2.0.1 (5.7)",
    "vendor": "Fujitsu",
    "buildBranch": "RELEASE-2.0.1",
    "buildHash": "54f029c",
    "buildDirty": "",
    "buildTime": "2022-05-10T14:08:18",
    "buildTags": "",
    "buildVersion": "2.0.1"
}
```

Beispiel SOAP Response:

```
$ cmVersion --soap
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><ns2:getVersionResponse
xmlns:ns2="http://verification.esig.dsengine.ts.fujitsu.com/" xmlns:ns3="http://verification.esig.dsengine.ts.fujitsu.com"
xmlns:ns4="http://ers.esig.dsengine.ts.fujitsu.com" xmlns:ns5="http://esig.dsengine.ts.fujitsu.com/validation/detailed-report"
xmlns:ns6="http://esig.dsengine.ts.fujitsu.com/validation/diagnostic" xmlns:ns7="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns8="http://uri.etsi.org/19102/v1.2.1#" xmlns:ns9="http://uri.etsi.org/01903/v1.3.2#"
xmlns:ns10="http://uri.etsi.org/02231/v2#" xmlns:ns11="http://esig.dsengine.ts.fujitsu.com/validation/simple-report"
xmlns:ns12="http://esig.dsengine.ts.fujitsu.com/validation/simple-certificate-report"><response><name>Digital Signature
Engine</name><version>2.0.1</version><majorVersion>2</majorVersion><minorVersion>0</minorVersion><dssVersion>5.7</dssVersion><oem>fj
-oem-2.0.1 (5.7)</oem><vendor>Fujitsu</vendor><buildBranch>RELEASE-
2.0.1</buildBranch><buildHash>54f029c</buildHash><buildDirty></buildDirty><buildTime>2022-05-
10T14:08:18</buildTime><buildTags></buildTags><buildVersion>2.0.1</buildVersion></response></ns2:getVersionResponse></soap:Body></so
ap:Envelope>
```

Ist xmllint auf dem Rechner installiert, kann man die Ausgabe folgendermaßen formatieren:

```
$ cmVersion --soap | xmllint -format -
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getVersionResponse xmlns:ns2="http://verification.esig.dsengine.ts.fujitsu.com/"
xmlns:ns3="http://verification.esig.dsengine.ts.fujitsu.com" xmlns:ns4="http://ers.esig.dsengine.ts.fujitsu.com"
xmlns:ns5="http://esig.dsengine.ts.fujitsu.com/validation/detailed-report"
xmlns:ns6="http://esig.dsengine.ts.fujitsu.com/validation/diagnostic" xmlns:ns7="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns8="http://uri.etsi.org/19102/v1.2.1#" xmlns:ns9="http://uri.etsi.org/01903/v1.3.2#"
xmlns:ns10="http://uri.etsi.org/02231/v2#" xmlns:ns11="http://esig.dsengine.ts.fujitsu.com/validation/simple-report"
xmlns:ns12="http://esig.dsengine.ts.fujitsu.com/validation/simple-certificate-report">
      <response>
        <name>Digital Signature Engine</name>
        <version>2.0.1</version>
        <majorVersion>2</majorVersion>
        <minorVersion>0</minorVersion>
        <dssVersion>5.7</dssVersion>
        <oem>fj-oem-2.0.1 (5.7)</oem>
        <vendor>Fujitsu</vendor>
        <buildBranch>RELEASE-2.0.1</buildBranch>
        <buildHash>54f029c</buildHash>
        <buildDirty/>
        <buildTime>2022-05-10T14:08:18</buildTime>
        <buildTags/>
        <buildVersion>2.0.1</buildVersion>
      </response>
    </ns2:getVersionResponse>
  </soap:Body>
</soap:Envelope>
```

8.8. cmCreateServerKeyStore

Erzeugt ein self-signed Server Zertifikat für den HTTPS Connector. Weitere Details sind im Kapitel HTTPS Konfiguration beschrieben.

8.9. cmCreateService / cmCreateServiceFile / cmDeleteService

Die 3 Skripte haben die folgende Bedeutung

- cmCreateService
  Es wird eine systemd Service für das CryptoModule erzeugt. Die Beschreibung hierzu wird in der Datei */etc/systemd/system/cryptoModule.service* abgelegt. Für das Erzeugen der Datei wird das Skript cmCreateServiceFile genutzt.
  Der Service muss nach dem Anlegen noch gestartet werden.
  Dieses Skript kann nur vom Administrator (Benutzer root) ausgeführt werden.
- cmCreateServiceFile
  Dieses Skript gibt den Inhalt der systemd Service Datei auf stdout aus.
- cmDeleteService
  Der angelegte systemd Service wird gelöscht.
  Der Service wird, falls er noch läuft, vor dem Löschen gestoppt.
  Dieses Skript kann nur vom Administrator (Benutzer root) ausgeführt werden.

Hinweis: Während der RPM Installation wird das Skript *cmCreateService* ausgeführt, d.h. der systemd Service steht nach der Installation bereits zur Verfügung, ist aber noch nicht gestartet. Das Skript *cmDeleteService* wird während der Deinstallation ausgeführt.

8.10. cmGC.cron

Optionale Cron Job Vorlage, mit der eine Java Garbage Collection auf der laufenden Tomcat Instanz ausgelöst werden kann.

Beispiel:

0 * * * * /home/secdocs/CryptoModule/bin/cmGC.cron

Mit diesem Eintrag (Benutzer secdocs: crontab -e) wird jede Stunde eine Java Garbage Collection ausgelöst. Der letzte Ablauf des Skripts wird in der Datei *bin/cmGC.cron.log* protokolliert.

Auf der Webseite Crontab Guru findet man eine gut Beschreibung der Cron Syntax. Außerdem kann man dort seine Werte eingeben und sich anzeigen lassen, wann der entsprechende Job ausgeführt wird.

Dieses Skript läuft nur ab, wenn die Tomcat Instanz läuft.

8.11. cmRemoveLogs

Löscht alle Loggingdateien im Verzeichnis *server/logs*. Zusätzlich wird, falls vorhanden, die Datei *cryptoModule.pid* gelöscht.

Dieses Skript läuft nur ab, wenn die Tomcat Instanz **nicht** läuft.

8.12. cmHandleOutOfMemoryError

Dieses Skript ist ein internes Skript, dass vom Skript *cryptoModule* im Falle eines Java OutOfMemoryErrors aufgerufen wird. Es wird in diesem Fall versucht, eine jstack Ausgabe in der Datei *jstackOutOfMemoryError.log* zu erzeugen. Danach wird der Tomcat Java Prozess beendet.

8.13. cryptoModule

Lifecyle Skript (Start/Stop/...) für die CryptoModule Instanz.

Hinweis1: Die CryptoModule Instanz sollte entweder über dieses Skript gestartet und gestoppt werden oder aber über den zugehörigen systemd Service (systemctl ... cryptoModule.service).

Hinweis2: das Skript verwendet als Editor den vi. Soll ein anderer Editor benutzt werden, genügt es, im Environment (z.B. ~/.profile) die Environmentvariable *VISUAL* entsprechend zu setzen. Z.B.:

```
export VISUAL=/usr/bin/emacs
```

8.13.1. cryptoModule start

Startet die CryptoModule Instanz.

Nach einem erfolgreichen Start der Tomcat Instance wird die Datei */home/secdocs/CryptoModule/cryptoModule.pid* erzeugt, die die PID des zugehörigen Java JVM Prozesses enthält.

8.13.2. cryptoModule stop

Beendet die CryptoModule Instanz.

Die Datei */home/secdocs/CryptoModule/cryptoModule.pid* wird gelöscht.

8.13.3. cryptoModule restart

Löst einen Restart der CryptoModule Instanz aus (also: stop/start)

8.13.4. cryptoModule status

Zeigt den Status der Tomcat Instanz an

```
$ cryptoModule status
SUCCESS: cryptoModule: CryptoModule (DSEngine) is running
SUCCESS: cryptoModule: pid: 11342 , HTTP port: 18080

oder (wenn der HTTPS Conenctor konfiguriert ist):

SUCCESS: cryptoModule: pid: 11342 , HTTP port: 18080 , HTTPS port: 18443
```

oder

```
$ cryptoModule status
SUCCESS: cryptoModule: CryptoModule (DSEngine) is not running
```

8.13.5. cryptoModule edit

Öffnet die wichtigsten Konfigurationsdateien im Editor (Standard: vi):

- cryptoModule edit config
  Öffnet die Datei *server/conf/dsengine/dsengine-custom.properties*
- cryptoModule edit context
  Öffnet die Datei *server/conf/context.xml*
- cryptoModule edit log
  Öffnet die Datei *server/conf/dsengine/log4j2.xml*

(Loggingkonfiguration des CryptoModules)

- cryptoModule edit log tc (oder tomcat)
  Öffnet die Datei *server/conf/logging.properties*
  (Loggingkonfiguration des CryptoModules)
- cryptoModule edit server
  Öffnet die Datei *server/conf/dsengine/serverSettings*
- cryptoModule edit server tc (oder tomcat)
  Öffnet die Datei *server/conf/server.xml*
- cryptoModule edit web
  Öffnet die Datei *server/conf/web.xml*
- cryptoModule edit
  Öffnet die Datei *server/conf/dsengine/serverSettings*

8.13.6. cryptoModule log

Die Logdateien der Tomcat Instanz werden im Verzeichnis *server/logs* abgelegt. Einige dieser Dateien kann man direkt mit dem Skript *cryptoModule* öffnen (Standardeditor: vi):

- cryptoModule log boot
  Öffnet die Datei *server/logs/boot.log*
- cryptoModule log out
  Öffnet die Datei *server/logs/catalina.out*
- cryptoModule log server
  Öffnet die derzeit aktuelle Tomcat Server Logdatei *server/logs/catalina.YYYY-MM-DD.log*
- cryptoModule log tail
  Führt das Kommando "tail -f" auf der Datei
  server/logs/cryptoModule.log aus.
- cryptoModule log
  Öffnet die Datei *server/logs/cryptoModule.log*

Für den Anwender ist in der Regel nur die Datei *cryptoModule.log* von Interesse.

8.13.7. cryptoModule pid

Zeigt die PID des laufenden CryptoModules (= Tomcat JVM).

```
$ cryptoModule pid
16982
```

8.13.8. cryptoModule config

Gibt die aktuelle Konfiguration des CryptoModule aus.

```
$ cryptoModule config
cryptoModule: CryptoModule (DSEngine) configuration

CryptoModule (DSEngine) runtime environment settings
----------------------------------------------------
CM_ENV_VERSION           : 1
CM_ENV_VERSION_DATE      : 23-MAY-2022
CM_VERSION               : 2.0.1
CM_HOME                  : /home/secdocs/CryptoModule
CM_USER                  : secdocs
CM_GROUP                 : secdocs
CM_LANG                  : en_US.UTF-8
CM_SERVER_NAME           : CryptoModule
CM_JAVA_HOME             : /home/secdocs/CryptoModule/OpenJDK
CM_JRE_HOME              : /home/secdocs/CryptoModule/OpenJDK/jre
CM_CATALINA_HOME         : /home/secdocs/CryptoModule/tomcat
CM_CATALINA_BASE         : /home/secdocs/CryptoModule/server
CM_CATALINA_TMPDIR       : /home/secdocs/CryptoModule/server/temp
CM_PATH                  : /home/secdocs/CryptoModule/bin:/home/secdocs/CryptoModule/OpenJDK/bin:/bin:/usr/bin:/sbin:/usr/sbin
CM_LD_LIBRARY_PATH       :
CM_BIND_ADDRESS_HTTP     : 127.0.0.1
CM_BIND_ADDRESS_HTTPS    : 127.0.0.1
CM_PORT_BASE             : 18000
CM_PORT_OFFSET           : 0
CM_PORT_SHUTDOWN         : 18005
CM_PORT_HTTP             : 18080
CM_PORT_HTTPS            : 18443
CM_JAVA_START_MEM        : 4g
CM_JAVA_MAX_MEM          : 4g
CM_JAVA_MEM_THREAD_STACK : -Xss512k
CM_JAVA_GC               : -XX:+UseParallelGC -XX:+UseParallelOldGC -XX:+DisableExplicitGC
CM_JAVA_EXTRA_OPTS       :
CM_JAVA_HTTP_PROXY       :
CM_JAVA_HTTPS_PROXY      :
CM_JAVA_SOCKS_PROXY      :
============================================================
Overwritten CryptoModule (DSEngine) properties
----------------------------------------------
ocsp.cache.expirationtime=120
crl.cache.expirationtime=120
datasource.driver.class=org.hsqldb.jdbcDriver
datasource.url.type=file
datasource.url.file=jdbc:hsqldb:file:${catalina.base}/temp/cache/cachedb;hsqldb.lock_file=false
datasource.username=cmdba
etsi.validation.report.enabled=true
```

8.13.9. cryptoModule env

Gibt eine Liste aller gesetzten Environmentvariablen aus.

8.13.10. cryptoModule jcmd <Kommando>

Führt das Java SDK Kommando jcmd auf der laufenden Tomcat Instanz aus. Alle verfügbaren Kommandos erhält man mit dem Aufruf

```
cryptoModule jcmd help
```

8.13.11. cryptoModule jinfo

Führt das Java SDK Kommando jinfo auf der laufenden Tomcat Instanz aus.

8.13.12. cryptoModule jstack

Führt das Java SDK Kommando jstack auf der laufenden Tomcat Instanz aus.

8.13.13. cryptoModule clearCache

Ruft das Skript cmClearCache auf.

8.13.14. cryptoModule removeLogs

Ruft das Skript cmRemoveLogs auf.

8.13.15. cryptoModule uptime

Gibt für die laufende Tomcat Instanz die Startzeit, die Laufzeit in Sekunden und die Laufzeit im Format Tage:Stunden:Minuten:Sekunden.Millisekunden aus:

```
$ cryptoModule uptime
Tue Oct 26 09:22:18 2021
1744.640 seconds
000:00:29:04.640 (days:hours:minutes:seconds:milliseconds)
```

8.13.16. cryptoModule version

Gibt die verwendeten Software Versionen aus:

```
$ cryptoModule version
CM_ENV_VERSION      : 1
CM_ENV_VERSION_DATE : 23-MAY-2022
CM_VERSION          : 2.0.1

Tomcat
Using CATALINA_BASE:   /home/secdocs/CryptoModule/server
Using CATALINA_HOME:   /home/secdocs/CryptoModule/tomcat
Using CATALINA_TMPDIR: /home/secdocs/CryptoModule/server/temp
Using JRE_HOME:        /home/secdocs/CryptoModule/OpenJDK/jre
Using CLASSPATH:       /home/secdocs/CryptoModule/tomcat/bin/bootstrap.jar:/home/secdocs/CryptoModule/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Server version: Apache Tomcat/9.0.64
Server built:   Jun 2 2022 19:08:46 UTC
Server number:  9.0.64.0
OS Name:        Linux
OS Version:     5.3.18-150300.59.68-default
Architecture:   amd64
JVM Version:    1.8.0_332-b09
JVM Vendor:     Temurin

OpenJDK
openjdk version "1.8.0_332"
OpenJDK Runtime Environment (Temurin)(build 1.8.0_332-b09)
OpenJDK 64-Bit Server VM (Temurin)(build 25.332-b09, mixed mode)
```

8.13.17. cryptoModule fdCount

Zeigt die Anzahl der offenen Filedeskriptoren der laufenden CryptoModule Instanz an. Läuft das

CryptoModule nicht, wird 0 ausgegeben.

8.13.18. cryptoModule GC

Startet eine Java Garbage Collection für die laufende CryptoModule Instanz.

Chapter 9. Anhang

9.1. HTTPS Konfiguration

Nach der Installation ist derzeit der HTTPS Connector in der Datei *server/conf/server.xml* auskommentiert:

```
<!--
<Connector address="${bindAddress.https}" port="${port.https}"
...
</Connector>
-->
```

Werden die Kommentare entfernt, z.B.:

```
<!-- -->
<Connector address="${bindAddress.https}" port="${port.https}"
...
</Connector>
<!-- -->
```

dann können auch HTTPS Requests an die Anwendung gesendet werden.

Hinweis: Der HTTPS Connector erlaubt nur TLS 1.2 Verbindungen.

Standardmäßig wird das Server Zertifikat aus der Datei *server/conf/dsengine/serverKeyStore.p12* gelesen. Ein self-signed Zertifikat kann man mit Hilfe des Skripts *bin/cmCreateServerKeyStore* erzeugen.

Hat man die Daten in einer eigenen PKCS12 Datei, so kann man einfach in der Datei *server/conf/server.xml* in der HTTPS Connector Konfiguration

```
<Connector address="${bindAddress.https}" port="${port.https}"
...
</Connector>
```

die Zeilen

```
keystoreFile="${catalina.base}/conf/dsengine/serverKeyStore.p12"
keystorePass="changeit"
```

anpassen. Sollte es sich bei der Datei um ein JKS-Datei handeln (JKS: Java Key Store), so ist zusätzlich die Zeile

```
keystoreType="PKCS12"
```

in

```
keystoreType="JKS"
```

abzuändern.

9.1.1. Erzeugen eines self-signed Zertifikats für den HTTPS Connector

Das Skript *bin/cmCreateServerKeyStore* erzeugt im Verzeichnis *serverKeyStore* die Dateien:

- cryptoModule.cer
  Server Zertifikat im DER Format
- cryptoModule.csr
  Certificate Sign Request Datei
- serverKeyStore.p12
  Keystore mit dem Public/Private Keypair
- cryptoModule.rfc.cer
  Server Zertifikat im PEM Format

Existiert das Verzeichnis *serverKeyStore* noch nicht, wird es angelegt.

Wird das Skript *bin/cmCreateServerKeyStore* mit der Option *-i* (oder *--install*) aufgerufen, so wird abschließend die Datei *serverKeyStore.p12* in das Verzeichnis *server/conf/dsengine* kopiert.

9.2. Ändern der Default Policy

In einer laufenden Anwendung kann man sich mit dem Skript *cmGetPolicyData* die Default Policy XML Datei herunterladen. Diese Datei findet man auch unter *data/policy.xml*.

Die Default Policy lässt sich durch eine eigene, angepasste Version ersetzten. Die angepasste Policy Datei kann man in der Konfigurationsdatei *server/conf/dsengine/dsengine-custom.properties* eintragen:

```
validation.policy.path=/full-path-to-file/own_default_policy.xml
```

Achtung: eine Anpassung der Default Policy sollte nur in Rücksprache mit dem Fujitsu Service durchgeführt werden.

9.3. Logging

Im Verzeichnis *server/logs* liegen Dateien der folgenden Form

- access.YYYY-MM-DD.log
  Tomcat Access Logdatei (vergleichbar zur Access Logdatei eines Webservers).
  Standardmäßig wird diese Datei **nicht** angelegt.

- catalina.YYYY-MM-DD.log
  Tomcat Logdatei.
- catalina.out
  In dieser Datei werden alle Ausgaben gesammelt, die ohne Logger direkt auf stdout geschrieben werden.
- boot.log
  ogging des Bootvorgangs des Tomcat Servers.
- cryptoModule.log
  Loggingdatei des CryptoModules.
  In der Regel ist dies die einzige Datei, die für den Benutzer von Interesse ist.

9.3.1. Tomcat Access Logging (access.YYYY-MM-DD.log)

Am Ende der Datei *server/conf/server.xml* gibt es den Eintrag

```
<!--
<Valve className="org.apache.catalina.valves.AccessLogValve"
directory="logs"
prefix="access."
suffix=".log"
buffered="true"
fileDateFormat="yyyy-MM-dd"
maxDays="14"
rotatable="true"
pattern="%h %t %H %m %U %s %D %b" />
-->
```

Wird dieser Eintrag entkommentiert, dann werden in der Access Logdatei Zeilen der folgenden Form abgelegt:

```
192.168.3.45 [12/Feb/2019:16:33:59 +0100] HTTP/1.1 GET / 200 2807 3235
```

Bedeutung der einzelnen Einträge:

- 192.168.3.45
  IP Adresse des Aufrufers
- [12/Feb/2019:16:33:59 +0100]
  Zeitstempel des Aufrufs
- HTTP/1.1
  Genutzte Protokollvariante
- GET
  Verwendete HTTP Methode
- /
  Verwendete URL
- 200

HTTP Status nach Ausführung des Requests

- 2807
  Requestdauer in Millisekunden
- 3235
  Anzahl der in der Response gesendeten Bytes.

Für Diagnosezwecke kann man das Pattern um die Option %I erweitern

```
pattern="%h %t %H %m %U %s %D %b - %I"
```

Es wird dann der Thread, in dem der Request ausgeführt wird, mitprotokolliert. Die Zeilen sehen dann so aus:

```
192.168.3.45 [12/Feb/2019:16:33:59 +0100] HTTP/1.1 GET / 200 2807 3235 -
http-nio-0.0.0.0-18080-exec-1
```

In dieser Konfiguration wird täglich beim ersten Schreiben eines Logsatzes die Loggingdatei geändert. Dateien, die älter als 14 Tage sind (maxDays), werden gelöscht. Details zur Konfiguration findet man im Kapitel Access Logging der Tomcat 9 Beschreibung.

9.3.2. Tomcat Logging (catalina.YYYY-MM-DD.log)

Alle Tomcat Server relevanten Logsätze werden in dieser Datei abgelegt. Die Konfiguration für die Loggingdatei und das Logging selbst, findet man in der Datei *server/conf/logging.properties*:

```
handlers = 1catalina.org.apache.juli.AsyncFileHandler
.handlers = 1catalina.org.apache.juli.AsyncFileHandler
1catalina.org.apache.juli.AsyncFileHandler.level = FINE
1catalina.org.apache.juli.AsyncFileHandler.directory =
${catalina.base}/logs
1catalina.org.apache.juli.AsyncFileHandler.prefix = catalina.
1catalina.org.apache.juli.AsyncFileHandler.rotatable = true
1catalina.org.apache.juli.AsyncFileHandler.maxDays = 7
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].level =
INFO
org.apache.cxf.level = WARNING
org.springframework.level = WARNING
```

In dieser Konfiguration wird täglich beim ersten Schreiben eines Logsatzes die Loggingdatei geändert. Dateien, die älter als 7 Tage sind (maxDays), werden gelöscht. Eine Beschreibung der möglichen Parameter findet man in der JavaDoc zur Klasse AsyncFileHandler, die wiederum im Wesentlichen die Parameter der Klasse FileHandler verwendet.

9.3.3. Tomcat Loggingdateien catalina.out und boot.log

Diese beiden Tomcat Loggingdateien werden nur in Spezialfällen benutzt:

- catalina.out

Hier werden alle Ausgaben protokolliert, die in der Anwendung direkt auf stdout (also ohne eine Logger) geschrieben werden.

- boot.log
  Hier wird nur das Starten des Tomcat Servers protokolliert. Sobald der Server angelaufen ist, werden alle weiteren Ausgaben in der Tomcat Loggingdatei protokolliert.

Die Datei *boot.log* hat normalerweise den folgenden Inhalt

```
Using CATALINA_BASE: /home/secdocs/CryptoModule/server
Using CATALINA_HOME: /home/secdocs/CryptoModule/tomcat
Using CATALINA_TMPDIR: /home/secdocs/CryptoModule/server/temp
Using JRE_HOME: /home/secdocs/CryptoModule/OpenJDK/jre
Using CLASSPATH:
/home/secdocs/CryptoModule/tomcat/bin/bootstrap.jar:/home/secdocs/CryptoModule/tomcat/bin/tomcat-juli.jar

Tomcat started.
```

Beiden Dateien werden **nicht** gewechselt. D.h.: mit der Zeit wachsen diese Dateien. Da in der Regel nur wenig in diese Dateien geschrieben wird, sollte das im Produktivbetrieb kein Problem sein. Will man auf Nummer sicher gehen, kann man für diese Dateien eine Logrotate Konfiguration anlegen.

Logrotate

Auf der Seite logrotate(8) findet man die folgende Beschreibung zum Tool Logrotate:

logrotate is designed to ease administration of systems that generate large numbers of log files. It allows automatic rotation, compression, removal, and mailing of log files. Each log file may be handled daily, weekly, monthly, or when it grows too large.

Man kann z.B. die folgenden Logrotate Konfigurationsdateien verwenden:

Datei */etc/logrotate.d/olcm-tomcat-catalina.conf*:

```
/home/secdocs/CryptoModule/server/logs/catalina.out {
rotate 5
size 20M
nocompress
notifempty
missingok
copytruncate
su secdocs secdocs
}
```

Datei */etc/logrotate.d/olcm-tomcat-boot.conf*:

```
/home/secdocs/CryptoModule/server/logs/boot.log {
rotate 5
size 20M
nocompress
notifempty
missingok
copytruncate
su secdocs secdocs
}
```

Testen der Log Rotation (Benutzer root)

```
# logrotate --force /etc/logrotate.d/olcm-tomcat-catalina.conf

# logrotate --force /etc/logrotate.d/olcm-tomcat-boot.conf
```

Weitere Details zur Nutzung von Logrotate findet man z.B. hier: „https://linux.die.net/man/8/logrotate[logrotate(8) - Linux man page]".

9.3.4. CryptoModule Logging (cryptoModule.log)

Alle Loggingsätze des CryptoModules werden in dieser Datei abgelegt. Das CryptoModule verwendet das Apache Log4j2 Framework. Die zugehörige Konfigurationsdatei ist unter *server/conf/dsengine/log4j2.xml* abgelegt.

Die Log4J2 Konfiguration ist für den Produktivbetrieb des CryptoModules vorkonfiguriert.

Für Diagnosezwecke kann man in dieser Datei, nach Rücksprache mit dem Service, die Konfiguration der Logger für das CryptoModule verändern. Die Datei wird alle **60 Sekunden** auf eine **Veränderung** geprüft und falls eine Änderung gefunden wurde, neu geladen.

Die Wartezeit kann man über das Attribut *monitorInterval* verändern:

```
<Configuration monitorInterval="60" ...
```

Die Zeitangabe erfolgt in Sekunden. Setzt man den Wert auf *0*, wird die Datei nie neu geladen.

Die Eigenschaften der Loggingdatei cryptoModule.log sind in der Datei in den folgenden Elementen festgelegt:

```
<RollingFile ... >
...
<Policies>
<OnStartupTriggeringPolicy />
<SizeBasedTriggeringPolicy *size="100 MB*" />
</Policies>
<DefaultRolloverStrategy **max="10"**>
<Delete basePath="${logPathBackup}">
<IfAccumulatedFileCount *exceeds="10"* />
</Delete>
</DefaultRolloverStrategy>
</RollingFile>
```

D.h.: die Datei wir maximal **100 MB** groß. Ist diese Größe überschritten, wird die aktuelle Datei geschlossen und in das Verzeichnis *server/logs/backup* verschoben. Von diesen alten Dateien werden maximal **10** aufbewahrt.

Loggen der SOAP Nachrichten
In der Datei *log4j2.xml* gibt es den Eintrag

```
<asynclogger name="org.apache.cxf.services"
             level="warn" additivity="false">
    <appender-ref ref="file" />
</asynclogger>
```

Setzt man hier das Log Level von **warn** auf **all**, dann werden die SOAP Nachrichten in die Datei *cryptoModule.log* geschrieben.

```
<asynclogger name="org.apache.cxf.services"
             level="all" additivity="false">
    <appender-ref ref="file" />
</asynclogger>
```

9.4. JRE (Java Runtime Environment)

9.4.1. JRE Konfigurationsdateien

Im Verzeichnis *CryptoModule/OpenJDK/jre/lib/security* befinden sich die Konfigurationsdateien für das verwendete JRE (Java Runtime Environment). Zu jeder dieser Dateien existiert eine readonly Sicherheitskopie mit der Endung *.ORIG*:

- blacklisted.certs.ORIG
- cacerts.ORIG
- java.policy.ORIG
- java.security.ORIG

9.4.2. JRE Trustbase

Für die HTTPS Verbindungen zu den einzelnen Ressourcen wird als Basis die JRE (Java Runtime Environment) Trustbase in der Datei *OpenJDK/jre/lib/security/cacerts* verwendet. Es kann dabei vorkommen, dass benötigte Zertifikate in dieser Trustbase (noch) nicht enthalten sind.

In dieser Situation bieten sich 2 Lösungen an

1. Die fehlenden Zertifikate zur Standard JRE Trustbase Datei cacerts hinzufügen.
2. Verwendung einer alternative Trustbase
   Hinweis: die Qualität/Vertrauenswürdigkeit kann nicht durch Fujitsu garantiert werden.

Damit die Originaldatei des JRE im Zweifelsfall nicht verloren geht, ist sie als Sicherungskopie unter dem Namen *CryptoModule/OpenJDK/jre/lib/security/cacerts.ORIG* abgelegt.

Hinzufügen von Zertifikaten zur JRE Trustbase

```
$ cd CryptoModule/OpenJDK/jre/lib/security
$ keytool -importcert \
          -keystore ./cacerts -storepass changeit \
          -alias aliasName -file certFile \
          -noprompt -trustcacerts -v
```

- aliasName
  In einem JKS Keystore muss jedes Zertifikat einen Namen (Alias) haben. Diesen Namen kann man mit der Option -alias benennen. Der Defaultname ist *mykey*. D.h.: man kann ohne die Option -alias maximal 1 Zertifikat importieren.
- certFile
  Dateiname der Datei, die das zu importierende Zertifikat enthält.

Löschen von Zertifikaten in der JRE Trustbase

Es kann auch sein, dass Zertifikate, die in der JRE Trustbase enthalten sind, nicht mehr verwendet werden sollen/dürfen. So ein Zertifikat kann man mit dem folgenden Kommando aus der JRE Trustbase entfernen:

```
$ cd CryptoModule/OpenJDK/jre/lib/security
$ keytool -delete -keystore ./cacerts -storepass changeit \
-alias aliasName
```

Falls der Alias Name (*aliasName*) des gewünschten Zertifikats in der JRE Trustbase nicht gefunden wird, kann man sich alle Einträge und ihre Alias Namen mit dem folgenden Kommando ansehen:

```
$ cd CryptoModule/OpenJDK/jre/lib/security
$ keytool -list -keystore ./cacerts -storepass changeit -v
```

Verwendung des Mozilla CA Certificate Stores

Die Mozilla Wiki enthält die Seite Mozilla Included CA Certificate. Dort findet man den Verweis auf die Datei certdata.txt.

Eine lesbare Beschreibung der enthaltenen Zertifikate findet man hier: Description of Included CA Certificates. Eine zugehörige CSV Datei, die auch die Zertifikate im PEM-Format enthält, ist abgelegt unter CSV with PEM of raw certificate data. Den Inhalt kann man sich auch einfach auf der Seite

https://curl.haxx.se/docs/caextract.html

als PEM Datei (https://curl.haxx.se/ca/cacert.pem) herunterladen.

Die in dieser Datei enthaltenen Zertifikate kann man in eine JKS Keystore Datei (Passwort: *changeit*) importieren. Diese Datei kann man dann unter dem Namen *OpenJDK/jre/lib/security/cacerts* ablegen. Eine Version dieser Daten ist unter *CryptoModule/data/cacerts_mozilla.jks* abgelegt und kann einfach auf

den Namen *CryptoModule/OpenJDK/jre/lib/security/cacerts* kopiert werden.

Damit das Erstellungsdatum dieser Datei einfach zu erkennen ist, fügen wir ein eigenes, selbst signiertes Zertifikat hinzu, dass eine Gültigkeit von 1 Tag hat. Dieses Zertifikat kann man sich folgendermaßen ansehen:

```
$ cd CryptoModule/data
$ keytool -list -keystore ./cacerts_mozilla.jks \
          -storepass changeit -alias secdocs_mozilla_ca -v | head -10
Alias name: secdocs_mozilla_ca
Creation date: Feb 1, 2022
Entry type: trustedCertEntry

Owner: CN=MozillaCA for SecDocs, O=Fujitsu, OU=SecDocs, L=Munich, ST=Bavaria, C=DE
Issuer: CN=MozillaCA for SecDocs, O=Fujitsu, OU=SecDocs, L=Munich, ST=Bavaria, C=DE
Serial number: 1a619090e3894cc3f0a86661f124a0e6a84da1e6
Valid from: Tue Feb 01 09:19:58 CET 2022 until: Wed Feb 02 09:19:58 CET 2022
Certificate fingerprints:
         SHA1: D9:1A:87:9D:E5:2C:F5:2D:04:71:1B:77:0D:F0:62:1A:3A:3A:73:2A
         SHA256: 32:77:65:3F:0F:32:63:4B:9F:96:51:D7:E4:C9:9D:5C:EE:9F:C5:76:D7:5F:60:FC:2A:C8:46:09:C7:C7:80:EF
Signature algorithm name: SHA512withRSA
Subject Public Key Algorithm: 4096-bit RSA key
Version: 3
```

Interessant an diesem Eintrag sind nur der Alias Name ***secdocs_mozilla_ca*** und das Kennzeichen "Valid from“. Der Eintrag "***Valid from: Tue Feb 01 09:19:58 CET 2022***" gibt das Erstellungsdatum dieser Datei wieder.

9.5. Tuning

9.5.1. Tomcat Shutdown Password

Am Anfang der Datei *server/conf/server.xml* findet man die Zeile

```
<Server address="127.0.0.1" port="${port.shutdown}"
shutdown="*SHUTDOWN*">
```

D.h.: wenn jemand vom lokalen Rechner (localhost) eine Socket Verbindung (Standardport: 18005) zur Tomcat Server Instanz aufbaut und den String "SHUTDOWN" sendet, wird die Tomcat Server Instanz heruntergefahren. ***Diesen String sollte man aus Sicherheitsgründen abändern.***

9.5.2. Tomcat JVM Heap Size

Die Default JVM Heap Größe ist auf 4096 MB (= 4 GB) gesetzt. Dieser Wert kann bei Bedarf (z.B. OutOfMemoryError) verändert werden. Der Wert kann in der Datei *server/conf/dsengine/serverSettings* geändert werden:

```
CM_JAVA_MEM=-Xms4096m
```

Beim Starten der Tomcat JVM wird dieser Wert als Min/Max Heap Size übergeben:

```
-Xms${CM_JAVA_START_MEM} -Xmx${CM_JAVA_MAX_MEM}
```

9.5.3. Tomcat JVM Heap Dump bei einem OutOfMemoryError

Im Fall eines OutOfMemoryError in der JVM wird der Tomcat Server beendet. Für Diagnosezwecke kann zusätzlich vor dem Beenden ein Heap Dump der JVM erzeugt werden.

Dazu muss in der Datei *server/conf/dsengine/serverSettings* der Eintrag

```
CM_JAVA_DUMP_HEAP_ON_MEMERR="-XX:+HeapDumpOnOutOfMemoryError"
```

hinzugefügt werden.

Der Java Heap Dump wird in der Datei *server/java_pid**pid**.hprof* abgelegt. Dabei ist **pid** die aktuelle PID des JVM Prozesses.

9.5.4. Tomcat JVM Garbage Collection

Je nach Einsatzszenario kann es sein, dass man die Java Garbage Collection anpassen muss, da es sonst periodisch zu längeren Antwortzeiten kommen kann. Wir benutzen den Parallel Garbage Collector. Die benötigte Option wird in der Datei *server/conf/dsengine/serverSettings* gesetzt:

```
CM_JAVA_GC="-XX:+UseParallelGC -XX:+UseParallelOldGC"
```

Alternativ kann auch der G1 GC (Garbage-First Garbage Collector) benutzt werden:

```
CM_JAVA_GC="-XX:+UseG1GC"
```

Eine Liste aller zusätzlich verfügbaren Optionen und ihrer Standardwerte findet man im Dokument Garbage-First Garbage Collector Tuning.

Das Verhalten der Garbage Collection läßt sich folgendermaßen in einer Datei (hier: gc.log) ausgeben:

```
CM_JAVA_GC="-XX:+UseParallelGC -XX:+UseParallelOldGC
-Xloggc:${CM_HOME}/gc.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps"
```

Im obigen Beispiel wird die Datei gc.log im Installationsverzeichnis (Default: */home/secdocs/CryptoModule*) abgelegt.

Die Datei lässt sich z.B. mit dem Tool GCViewer visualisieren. Dieses Tool kann man auf der Seite GCViewer Releases herunterladen.

9.5.5. SecDocs (MSOS) ⇄ CryptoModule Kommunikation mit großen Datenmengen

In der Property Datei *server/conf/dsengine/dsengine-custom.properties* kann man die maximal erlaubten Datengröße auf die folgenden Maximalwerte erhöhen:

```
cxf.stax.maxTextLength=2147483647
cxf.attachmentThreshold=2147483647
cxf.stax.attachmentMaxSize=2147483647
cxf.mtom.enabled=true
cxf.debug=false
cxf.logging.limit=n oder cxf.logging.limit=0
```

Anmerkung: *cxf.logging.limit=0* bedeutet, dass alles Daten im Logging ausgegeben werden. Will man diesen Wert begrenzen, kann man hier eine Zahl angegeben, die der maximalen Anzahl Bytes, die ausgegeben werden sollen, entspricht.

Hinweis: Das CryptoModule nutzt das Apache CXF Services Framework. Die Defaultwerte für verschiedene Begrenzungen findet man auf der Seite Securing CXF Services.

9.5.6. Manuelle Java Garbage Collection auslösen

Abhängig von der Belastung eines Java Systems kann es nötig sein, manuell eine Garbage Collection auszulösen. Eine Garbage Collection kann durch das Skript *bin/cmGC.cron* ausgelöst werden. Dieses Skript kann auch als Cron Job für den Benutzer *secdocs* eingetragen werden. Siehe für weitere Details cmGC.cron

9.5.7. ETSI (Validierungs-) Report deaktivieren

Ab der Version 1.1.9 des Fujitsu CryptoModuls wird bei der Validierung von Signaturdaten zusätzlich im Validierungsreport ein ETSI Report miterzeugt.

Das hat 2 Konsequenzen:

1. Die Validierungsreports werden größer
2. Die Performance ist etwas geringer.

Benötigt man die Daten des ETSI Reports nicht, kann man diese Funktionalität in der Datei *server/conf/dsengine/dsengine-custom.properties* abschalten.

Die letze Zeile in der Datei ist dazu von

```
etsi.validation.report.enabled=true
```

auf

```
etsi.validation.report.enabled=false
```

abzuändern.

9.6. Fehlerbehebung

Die CryptoModule Anwendung kann während der Nutzung und im Laufe der Zeit verschiedene Fehler loggen. Nicht all diese geloggten Fehler sind direkt als eine Fehlfunktion der Anwendung zu verstehen bzw. anzunehmen. Denn solche Fehler können auch auf der Seite von und/oder während der Kommunikation mit einem involvierten dritten Service-Provider, der die CryptoModule Anwendung nutzt, auftreten (z.B. Netzwerkprobleme, Protokollfehler etc.).

9.6.1. EU Trusted Lists (TL)

Die EU-Mitgliederstaaten sind verpflichtet vertrauenswürdige Listen qualifizierter Vertrauensdiensteanbieter und der von ihnen bereitgestellten qualifizierten Vertrauensdienste zu erstellen, zu führen und zu veröffentlichen. Zusätzlich tragen sie die Verantwortung bzgl. Wartung und Aufrechterhaltung dieser Liste bzw. dieser Informationen.

Die Informationen werden von der CryptoModule Applikation während der Ausführung verschiedener Funktionen in Anspruch genommen. Die Kommunikation mit diesen externen Providern kann ab und zu, aus verschiedenen Gründen, zu verschiedenen Fehlern führen (z.B. Netzwerkprobleme).

9.6.2. Typische Fehlerbilder

Um solche Fehler richtig zu diagnostizieren und sie zu verstehen bzw. zu beheben, wurde folgende Tabelle/FAQ erstellt:

Symptom	**Analyse**
Zertifikate konnten nicht geladen werden	1. Ist der TL Service über Internet Browser (z.B. Chrome, Edge, Firefox) erreichbar? 2. Ist eine aktuelle JRE cacerts Datei installiert? 3. Ist das TL Service Zertifikat zeitlich gültig (nicht abgelaufen)? 4. ASN1-Struktur der betreffenden Zertifikate mittels einer anderen Applikation (ASN1-Validator) prüfen

Symptom	Analyse
Kommunikation mit einem TL Land fehlgeschlagen	Typischerweise kann ein HTTPS Server Zertifikat, das beim Laden einer Trust List verifiziert wird, nicht verifiziert werden.
Anzahl der geladenen TLs stimmt nicht überein (z.B. Nb of loaded trusted lists : 31/32)	Mögliche Ursache: ein Zertifikat kann nicht validiert werden. Für weitere Details muss man sich die Fehlermeldungen in CryptoModule Logdatei ansehen.
WARN There is no DataLoader defined to load Certificates from AIA extension	Diese Warnung kann im Kontext LoTL/TL (also insbesondere beim Start der Anwendung und beim Refresh) ignoriert werden.