ЗАО «Сигнал-КОМ»

Версия 1.1

Руководство администратора

(17.12.2015)

Москва

2015

Приложение OCSP Server предназначено для проверки статуса X.509 сертификатов в соответствии с RFC 2560. Приложение поддерживает как российские, так и зарубежные криптографические алгоритмы.

Содержание

1. Общие сведения

1.1. Назначение

Приложение OCSP Server предназначено для проверки статуса X.509 сертификатов в соответствии с RFC 2560 [1]. Приложение поддерживает российские и зарубежные криптографические алгоритмы.

1.2. Технические характеристики

Сервер поддерживает следующие алгоритмы хэширования:

- ГОСТ Р 34.11-2012 [14];
- ГОСТ Р 34.11-94 [12];
- SHA-1 [8];
- SHA-256 [9];
- SHA-384 [9];
- SHA-512 [9];

Сервер поддерживает следующие алгоритмы электронной подписи:

- ГОСТ Р 34.10-2012 [15];
- ГОСТ Р 34.10-2001 [13];
- RSA [7].

Приложение OCSP Server выполнено по технологии сервлетов и может исполняться на любом сервере приложений (или контейнере сервлетов), реализующем спецификацию Servlet API 2.5+.

OCSP Server использует сертификаты и списки отозванных сертификатов (СОС) в формате ITU-T X.509 [3], RFC 3280 [4], RFC 4491 [19] и рекомендаций ТК26 [25]. Сертификат OCSP должен удовлетворять требованиям, изложенным в разделе 4.2.2.2 RFC 6960 [1].

Приложение OCSP Server обращается к реализации криптографических примитивов через интерфейс JCA [22]. В частности, российские криптографические алгоритмы реализованы в провайдере Signal-COM JCP [23].

При разработке приложения использовано средство криптографической защиты информации (СКЗИ) «Крипто-КОМ» [16], имеющее сертификаты соответствия ФСБ РФ.

1.3. Особенности реализации

OCSP Server использует в качестве транспорта для получения запросов и передачи ответов протокол HTTP в соответствии с разделом Appendix A RFC 6960.

Ответы сервера могут быть заверены как непосредственно ключом электронной подписи издателя сертификатов (т.е. ключом Удостоверяющего центра), так и ключом, предназначенным специально для подписи OCSP-ответов – в соответствии с разделом 2.6 RFC 6960.

Источником информации о статусах сертификатов для OCSP Server являются списки отозванных сертификатов, выпускаемые УЦ и размещаемые либо в файловой системе либо на сервере LDAP [?]. Текущая версия сервера не поддерживает работу с deltaCRL.

Сервер периодически (в соответствии с настройками) обновляет информацию о статусах сертификатов.

Все объекты, используемые сервером (сертификаты, списки отозванных сертификатов), проходят обязательную проверку целостности и действительности.

Сервер может обрабатывать запросы о статусе сертификатов для произвольного количества УЦ (издателей сертификатов).

Клиент может запрашивать информацию о статусе как одного, так и сразу нескольких сертификатов; при этом все сертификаты в запросе должны принадлежать одному издателю (УЦ).

OCSP Server не поддерживает работу с OCSP-запросами, содержащими электронную подпись (подпись игнорируется).

Из списка расширений, приведённых в RFC 6960, приложение OCSP Server поддерживает только работу с Nonce (см. раздел 4.4.1) – остальные расширения игнорируются.

2. Установка приложения

Установка и конфигурирование приложения OCSP Server рассмотрены на примере использования контейнера сервлетов Apache Tomcat 7 (http://tomcat.apache.org).

2.1. Подготовка среды

Для развёртывания приложения OCSP Server необходимо предварительно установить:

- JRE/JDK 1.6.0+ (см. http://java.com или http://www.oracle.com/technetwork/java/javase/downloads/index.html);
- Apache Tomcat 7.0+ (см. http://tomcat.apache.org/download-70.cgi).

Подробные инструкции по установке и настройке Apache Tomcat 7 см. http://tomcat.apache.org/tomcat-7.0-doc/setup.html.

2.2. Быстрый старт

В составе дистрибутива имеется тестовый пример конфигурации (файл conf/ocsp-conf.xml) и набор тестовых ключей (каталог conf/ocsp).

Для быстрого старта приложения выполните следующие действия:

- убедитесь, что сервер Apache Tomcat загружен;
- выполните разархивацию дистрибутива OCSP Server в произвольном каталоге (jar xvf ocsp-server-x.x.x.jar);
- скопируйте файлы тестовой конфигурации conf/OCSP-conf.xml и conf/OCSP-conf.dtd в каталог $CATALINA_HOME/conf;
- скопируйте каталог с тестовым набором ключей conf/ocsp в каталог $CATALINA_HOME/conf;
- скопируйте файл ocsp-server.war в каталог $CATALINA_HOME/webapps.

Приложение будет автоматически развёрнуто и загружено на выполнение в контексте /ocsp-server (т.е. доступно по URL http://yourserver:8080/ocsp-server).

Тестовые наборы ключей не должны использоваться при штатной эксплуатации приложения OCSP Server.

3. Генерация ключей

Приложение OCSP Server не содержит средств генерации и обслуживания криптографических ключей и сертификатов X.509. Для этих целей необходимо использовать другие средства, например:

- приложение keytool из состава JRE/JDK;
- приложение Admin-PKI [24].

Для генерации ключей электронной подписи ГОСТ Р 34.10 должны использоваться средства, сертифицированные ФСБ РФ.

4. Генератор случайных чисел

При формировании электронной подписи ГОСТ Р 34.10 приложение OCSP Server должно использовать генератор случайных чисел «GOST28147PRNG», реализованный в JCA-провайдере Signal-COM JCP [23].

Инициализация генератора случайных чисел «GOST28147PRNG» производится от вектора состояния, хранящегося в ключевом контейнере формата СКЗИ «Крипто-КОМ» [17]. Приложение OCSP Server не содержит средств генерации ключевых контейнеров, следовательно, операция должна производиться с помощью любых других средств, использующих СКЗИ «Крипто-КОМ» (например, Admin-PKI [24]).

В составе дистрибутива (в каталоге native) поставляются платформозависимые модули, необходимые для работы генератора случайных чисел СКЗИ «Крипто-КОМ». Модули представляют собой динамические библиотеки:

- rdtsc.dll - для Windows;
- librdtsc.so - для Linux и Solaris;
- librdtsc.jnilib - для Mac OS X.

Соответствующий модуль должен быть размещен в доступном каталоге, описанном в переменной окружения PATH (Windows), либо LD_LIBRARY_PATH (Linux, Solaris), либо DYLD_LIBRARY_PATH (Mac OS X).

5. Конфигурирование

Конфигурирование приложения OCSP Server осуществляется редактированием двух настроечных файлов формата XML: описателя процесса развёртывания (п. 5.1) и файла конфигурации (см. п. 5.2).

5.1. Описатель процесса развёртывания

Описатель развёртывания (файл web.xml) располагается в каталоге WEB-INF приложения.

В описателе развёртывания настраиваются:

- путь к файлу конфигурации;
- способ журналирования.

Путь к файлу конфигурации задаётся в параметре приложения с именем configFileName, например:

```
…
<context-param>
    <param-name>configFileName</param-name>
    <param-value>${catalina.home}/conf/ocsp-conf.xml</param-value>
</context-param>
…
```

Запись в журнал событий приложения OCSP Server настраивается путём редактирования параметров фильтра serverLogger.

5.2. Файл конфигурации

Файл конфигурации (**ocsp-conf.xml**) представляет собой XML-файл, соответствующий файлу определения (DTD) **ocsp-conf.dtd**. Файл **ocsp-conf.dtd** должен располагаться в том же каталоге, что и файл конфигурации приложения **ocsp-conf.xml**.

В нижеследующих разделах содержится подробное описание синтаксиса файла конфигурации.

5.2.1. Элемент <ocsp-server>

Файл конфигурации должен содержать корневой элемент <ocsp-server>.

```
DTD:
<!ELEMENT ocsp-server (hashalg+,random*, keystore*, signer+, checkpool+, profile+,
certs+, respscenario?)>
<!ATTLIST ocsp-server
   version CDATA #REQUIRED>
```

Список атрибутов элемента <ocsp-server>:

- version – обязательный атрибут; для текущей версии конфигурации должен иметь значение «1.0».

Элемент <ocsp-server> должен содержать следующие элементы:

- <hashalg> – см. п. 5.2.2;
- <signer> – см. п. 5.2.5;
- <checkpool> - см. п. 5.2.10
- <certs> - см. п. 5.2.11
- <profile> – см. п.5.2.13.

Элемент <ocsp-server> может содержать следующие элементы:

- <random> – см. п. 5.2.3;
- <keystore> – см. п. 5.2.4.
- <respscenario> - см. 5.2.12

5.2.2. Элемент <hashalg>

Элемент <hashalg> описывает алгоритм хэширования.

```
DTD:
<!ELEMENT hashalg EMPTY>
<!ATTLIST hashalg
   name CDATA #REQUIRED
   oid CDATA #IMPLIE>
```

Список атрибутов элемента <hashalg>:

- name – обязательный атрибут, задающий символьное имя алгоритма хэширования;
- oid – необязательный атрибут, содержащий OID алгоритма хэширования в десятично-точечном представлении.

Элемент <ocsp-server> должен содержать хотя бы один элемент <hashalg>.

Имя алгоритма хэширования, задаваемое атрибутом name, должно быть уникальным в контексте конфигурации.

Для перечисленных ниже имён алгоритмов хэширования атрибут oid может не указываться:

- «GOSTR3411-2012-512» – ГОСТ Р 34.11-2012 с длиной хэш-значения 512 бит [14];
- «GOSTR3411-2012-256» – ГОСТ Р 34.11-2012 с длиной хэш-значения 256 бит [14];
- «GOST3411-CP» – ГОСТ Р 34.11-94 с узлами замены id-GostR3411-94-CryptoProParamSet [21];
- «GOST3411» – ГОСТ Р 34.11-94 с узлами замены СКЗИ «Крипто-КОМ» (см. раздел 3.1 [19]);
- «SHA-1» – SHA-1 [8];
- «SHA-256» – SHA-256 [9];
- «SHA-384» – SHA-384 [9];
- «SHA-512» – SHA-512 [9].

Примеры:

```
<hashalg name="GOSTR3411-2012-256"/>
<hashalg name="GOST3411-CP"/>
<hashalg name="SHA-256"/>
```

5.2.3. Элемент <random>

Элемент <random> описывает датчик случайных чисел.

```
DTD:
<!ELEMENT random EMPTY>
<!ATTLIST random
    name CDATA #REQUIRED
    type CDATA #IMPLIED
    provider CDATA #IMPLIED
    dir CDATA #IMPLIED
    password CDATA #IMPLIED>
```

Список атрибутов элемента <random>:

- name – обязательный атрибут, задающий символьное имя ДСЧ;
- type – необязательный атрибут; имя типа ДСЧ, реализованного в JCA-провайдере;
- provider – необязательный атрибут; имя JCA-провайдера;
- dir – необязательный атрибут; указывает путь к каталогу PSE; используется с типом ДСЧ «GOST28147PRNG», реализованным в провайдере «SC»;
- password – необязательный атрибут; содержит пароль к PSE; используется с типом ДСЧ «GOST28147PRNG», реализованным в провайдере «SC».

Элемент <ocsp-server> может не содержать ни одного элемента <random>.

Имя, задаваемое атрибутом name, должно быть уникальным в контексте конфигурации.

Примеры:

```
<random name="ecgost" type="GOST28147PRNG" dir="/path/to/pse"/>
<random name="sha1" type="SHA1PRNG"/>
```

5.2.4. Элемент <keystore>

Элемент <keystore> описывает хранилище ключей.

```
DTD:
<!ELEMENT keystore EMPTY>
<!ATTLIST keystore
    name CDATA #REQUIRED
    file CDATA #REQUIRED
    password CDATA #IMPLIED
    type CDATA #IMPLIED
    provider CDATA #IMPLIED>
```

Список атрибутов элемента <keystore>:

- name – обязательный атрибут, задающий символьное имя хранилища;
- file – обязательный атрибут; указывает путь к хранилищу;
- password – содержит пароль к хранилищу; необязательный атрибут, по умолчанию используется пустая строка;
- type – необязательный атрибут; наименование типа хранилища, по умолчанию – «JKS»;
- provider – необязательный атрибут; имя JCA-провайдера.

Элемент <ocsp-server> может не содержать ни одного элемента <keystore>.

Имя, задаваемое атрибутом name, должно быть уникальным в контексте конфигурации.

Примеры:

```
<keystore name="store" file="/path/to/store.jks" type="JKS" password="*****"/>
```

5.2.5. Элемент <signer>

Элемент <signer> описывает контекст подписи.

```
DTD:
```

```
<!ELEMENT signer ((keyentry, signature?) | (priv, cert, signature?))>
<!ATTLIST signer
   name CDATA #REQUIRED
   type CDATA #REQUIRED
   init CDATA #IMPLIED
   default CDATA #IMPLIED>
```

Список атрибутов элемента <signer>:

- name – обязательный атрибут, задающий символьное имя контекста подписи;
- type – необязательный атрибут; наименование типа контекста подписи;
- init – необязательный атрибут, описывающий способ инициализации контекста подписи.
- default – необязателный атрибут, необходим для подписи сообщения о проверке статуса при установленном флаге alienaserror=false (см. 5.2.12), при отсутствии атрибута – первый по счету задекларированный в конфигурационном файле элемент данных ‘signer’.

Атрибут type может принимать следующие значения:

- keystore – если ключ подписи и сертификат располагаются в хранилище (это рекомендуемый способ); подробнее см. разделы 5.2.4, 5.2.6;
- file – если ключ подписи и сертификат хранятся в отдельных файлах; подробнее см. разделы 5.2.7, 5.2.8.

Атрибут init в текущей версии приложения может принимать единственное значение:

- auto – означает автоматическую загрузку ключей подписи при старте приложения; это значение используется по умолчанию.

Элемент <ocsp-server> должен содержать хотя бы один элемент <signer>.

Элемент <signer> должен содержать либо элемент <keyentry> (см. п. 5.2.6), либо пару элементов: <priv> (см. п. 5.2.7) и <cert> (см. п. 5.2.8). Элемент <signer> может также содержать элемент <signature> (см. п. 5.2.9)

Имя, задаваемое атрибутом name, должно быть уникальным в контексте конфигурации.

Примеры:

```
<signer name="ecgost" type="file">
  <priv type="PKCS#8" file="/some/dir/ocsp.p8" random="ecgost" password="*****"/>
  <cert type="X.509" provider="SC" file="/some/dir/ocsp.cer"/>
</signer>

<signer name="rsa" type="keystore">
  <keyentry keystore="store" alias="ocsp-rsa"/>
</signer>
```

5.2.6. Элемент <keyentry>

Элемент <keyentry> служит для описания ключа подписи и сертификата.

```
DTD:
<!ELEMENT keyentry EMPTY>
<!ATTLIST keyentry
   keystore CDATA #REQUIRED
   alias CDATA #REQUIRED
   password CDATA #IMPLIED
   random CDATA #IMPLIED>
```

Список атрибутов элемента <keyentry>:

- keystore – обязательный атрибут, задающий ссылку на элемент <keystore>; значение должно соответствовать значению атрибута name элемента <keystore> (см. п. 5.2.4);
- alias – задаёт имя записи в хранилище, указывающее на ключ подписи; обязательный атрибут;

- password – задаёт пароль к ключу; необязательный атрибут; если не задан, то используется тот же пароль, что и при доступе к хранилищу;
- random – ссылка на элемент <random>; значение должно соответствовать значению атрибута name элемента <random> (см. п. 5.2.3); обязательный атрибут для ключей ГОСТ Р 34.10.

Элемент <signer> может содержать только один элемент <keyentry>.

Элемент <keyentry> не содержит вложенных элементов.

Примеры:

```
<keystore name="store" file="/path/to/store.jks" type="JKS" password="*****"/>

<signer name="ecgost" type="file">
  <keyentry keystore="store" alias="ocsp-ecgost" random="ecgost"/>
</signer>

< signer name="rsa" type="keystore">
  <keyentry keystore="store" alias="ocsp-rsa"/>
</signer>
```

5.2.7. Элемент <priv>

Элемент <priv> служит для описания ключа подписи. Должен использоваться совместно с элементом <cert> (см. п. 5.2.8), описывающим сертификат.

```
DTD:
<!ELEMENT priv EMPTY>
<!ATTLIST priv
    file CDATA #REQUIRED
    type CDATA #IMPLIED
    provider CDATA #IMPLIED
    password CDATA #IMPLIED
    random CDATA #IMPLIED>
```

Список атрибутов элемента <priv>:

- file – обязательный атрибут; указывает путь к файлу ключа подписи;
- type – необязательный атрибут; наименование представления ключа, по умолчанию – «PKCS#8»;
- provider – необязательный атрибут; имя JCA-провайдера.
- password – содержит пароль к ключу; необязательный атрибут;
- random – ссылка на элемент <random>; значение должно соответствовать значению атрибута name элемента <random> (см. п. 5.2.3); обязательный атрибут для ключей ГОСТ Р 34.10.

Элемент <signer> может содержать только один элемент <priv>.

Примеры:

```
<signer name="ecgost" type="file">
  <priv type="PKCS#8" file="/some/dir/ocsp.p8" random="ecgost" password="*****"/>
  <cert type="X.509" provider="SC" file="/some/dir/ocsp.cer"/>
</signer>
```

5.2.8. Элемент <cert>

Элемент <cert> служит для описания сертификата. Должен использоваться совместно с элементом <priv> (см. п. 5.2.7), описывающим ключ подписи.

```
DTD:
<!ELEMENT cert EMPTY>
<!ATTLIST cert
    file CDATA #REQUIRED
    type CDATA #IMPLIED
    provider CDATA #IMPLIED>
```

Список атрибутов элемента <cert>:

- file – обязательный атрибут; указывает путь к файлу сертификата;
- type – необязательный атрибут; наименование представления сертификата, по умолчанию – «X.509»;
- provider – необязательный атрибут; имя JCA-провайдера.

Элемент <signer> может содержать только один элемент <cert>.

Элемент <cert> не содержит вложенных элементов.

Примеры:

```
<signer name="ecgost" type="file">
  <priv type="PKCS#8" file="/some/dir/ocsp.p8" random="ecgost" password="*****"/>
  <cert type="X.509" provider="SC" file="/some/dir/ocsp.cer"/>
</signer>
```

5.2.9. Элемент <signature>

Элемент <signature> служит для описания параметров подписи и должен использоваться как вложенный элемент элемента <signer> (см. п. 5.2.5).

```
DTD:
<!ELEMENT signature EMPTY>
<!ATTLIST signature
    hashalg CDATA #REQUIRED>
```

Список атрибутов элемента <signature>:

- hashalg – обязательный атрибут, задающий ссылку на элемент <hashalg>; значение должно соответствовать значению атрибута name элемента <hashalg> (см. п. 5.2.2); служит для задания алгоритма хэширования при формировании подписи под ответом сервера; в текущей версии приложения может использоваться только с ключами RSA.

Элемент <signer> может содержать только один элемент <signature>.

Элемент <signature> не содержит вложенных элементов.

Примеры:

```
<signer name="rsa" type="keystore">
  <keyentry keystore="store" alias="ocsp-rsa"/>
  <signature hashalg="SHA-256"/>
</signer>
```

5.2.10. Элемент <checkpool>

Элемент <checkpool> служит для описания множеств Списков Отозванных Сертификатов (СОС).

```
DTD:
<!ELEMENT checkpool EMPTY>
<!ATTLIST checkpool
name CDATA #REQUIRED
uri CDATA #REQUIRED
scantimesec CDATA # IMPLIED>
```

Список атрибутов элемента <checkpool>:

- name – обязательный атрибут, необходимый для установления связи между заданным профилем и множеством СОС.
- uri – обязательный элемент, описывающий местоположение внешнего хранилища СОС для загрузки сервером OCSP. Присваиваемое значение элемента ‘uri’- комплексное, которое включает несколько частей:
  - схема (scheme)– обязательная часть, содержит знак двоеточия (:), завершается двумя косыми чертами (//). Схема определяет механизм интерпретации последующих данных. После знака двоеточия может быть указан дополнительный параметр (порт)[1].

[1] В текущей версии сервера могут использоваться только две схемы: ‘ldap’ и ‘file’

- владелец (authority)– указывает принадлежность данных[1].
- путь (path) – указывает месторасположение загружаемых объектов, содержит иерархические цепочки, которые зависят от используемой схемы.

Дополнительнения uri:

- ? запрос (request) - используется для передачи дополнительной информации;
- #фрагмент (fragment) – может использоваться как дополнительный указатель на часть данных.

Типовые значения:

uri="ldap://www.e-notary.ru/o=ca" – в случае использования сервера LDAP в качестве хранилища, развернутого по адресу ‘www.e-notary.ru’ с начальным виртуальным каталогом, заданным фильтром ‘o=ca’ (см. подробнее 5.2.10.2);

uri="file:///../conf/ocsp/crls" – в случае использования файловых хранилищ, определяет локальный каталог ‘/conf/ocsp/crls’. Специальная последовательность ‘..’ в начале указывает, что упомянутый каталог должен быть расположен на уровень выше по отношению к рабочему каталогу развертывания сервера OCSP (см. подробнее 5.2.10.1).

- scantimesec – дополнительный (необязательный) атрибут, указывающий период сканирования (в секундах) внешнего хранилища СОСПри отсутствии данного параметра дата и время следующего обновления вычисляется через дату и время окончания срока действия актуального списка отозванных сертификатов (СОС) внутри каждого профиля.

5.2.10.1. Правила работы со схемой ‘file:’

Схема ‘file:’ в конфигурации OCSP используется для указания места хранения различных объектов (сертификаты, списки отозванных сертификатов) в виде файлов. Приведенные в текущем разделе правила формирования имен следуют из требований, изложенных в документе [26].

Корректные значения для указания локальных файлов:

- file:///path/to/file - ‘традиционный’ файловый uri для локального файла(ов) с неуказанным владельцем;
- file:/path/to/file - минимальное представление локального файла с абсолютным путем, который начинается с разделителя ‘/’ (слеш) и с отсутствующим полем владельца;
- file:c:/path/to/file – минимальное представление локального абсолютного пути, который начинается с буквы локального диска (‘c:’) в среде Windows.

Корректные значения для указания нелокальных (сетевых) файлов и каталогов:

- file://host.example.com/path/to/file - представление не локального файла с расширенным полем владельца;
- file:////host.example.com/path/to/file - ‘традиционное’ представление нелокального файла с ‘пустым’ владельцем и полным (преобразованным) машинонезависимым путем (UNC);
- file://///host.example.com/path/to/file - представление нелокального файла с экстра разделителем (слэш) между пустым полем владельца и полным (преобразованным) машинонезависимым путем (UNC).

5.2.10.2. Правила работы со схемой ‘ldap:’

Схема ‘ldap:’ в конфигурации OCSP используется для указания места внешнего хранения данных (сертификаты, списки отозванных сертификатов) в виде бинарных объектов. Приведенные в текущем разделе правила формирования имен следуют из требований, изложенных в документе [27].

[1] Может отсутствовать, например, в случае локального хранилища файлов для схемы ‘file’

Представление данных в схеме ldap описывается по правилам URI (см там же):

URI = [схема ":"] иерархическая-часть ["?" запрос] ["#" фрагмент]

Корректные значения для указания нелокальных данных в схеме ‘ldap:’:

- ldap://www.e-notary.ru/o=ca?reversedn – описание сервера LDAP в качестве источника данных, использующего интернет-адрес ‘www.e-notary.ru’ и параметр запроса ‘reversedn’- атрибут, необходимый для указания обратного порядка следования атрибутов уникального имени;
- ldap://[2001:db8::7]/c=GB?objectClass?one - представление не локального хранилища LDAP с использованием дополнительных запросов;
- ldap://127.0.0.1:8088/o=ca?reversedn - сервер LDAP использует локальный IP-адрес и нестандартный порт, а также параметр запроса ‘reversedn’ - атрибут, указывающий обратный порядок следования атрибутов уникального имени при организации поиска.

5.2.11. Элемент <certs>

Элемент <certs> описывает множества (пулы) сертификатов, необходимых для построения сертификационного пути в процедуре проверки валидности сертификатов OCSP всех задекларированных профилей конфигурации (см. 5.2.13).

```
DTD:
<!ELEMENT certs EMPTY>
<!ATTLIST certs
    trusted CDATA #REQUIRED
    uri CDATA #IMPLIED
    keystore CDATA #IMPLIED
    provider CDATA #IMPLIED>
```

Список атрибутов элемента <certs>:

- trusted – обязательный атрибут, признак доверенных сертификатов;
- uri – обязательный атрибут (альтернатива данным keystore, см. следующий пункт), описывающий местоположение внешнего хранилища сертификатов для загрузки сервером OCSP;
- keystore – ссылка (альтернатива данным uri), указывающая множество сертификатов (цепочку) из хранилища ключей, декларируемого элементом <keystore> (см. 5.2.4);
- provider - необязательный атрибут; имя JCA-провайдера.

Примеры:

```
    <certs uri="file:///../conf/ocsp/certs/intermediate" trusted="false" provider="SC"/>
    <certs uri="file:///../conf/ocsp/certs/root" trusted="true" provider="SC"/>
    <certs uri="LDAP://www.e-notary.ru/o=ca?reversedn " trusted="false"
provider="SC"/>
    <certs uri="file:///../conf/ocsp/certs/intermediate/cert-ocsp-srv.cer" trusted="false"
provider="SC"/>
    <certs keystore="store" trusted="false" provider="SC"/>
    <certs keystore="store1" trusted="true" provider="SC"/>
```

5.2.12. Элемент <respscenario>

В ходе эксплуатации возможна ошибочная ситуация, когда OCSP-запрос не может быть обработан по причине отсутствия информации об издателе проверяемого сертификата на сервере OCSP. В этом случае сервер, в зависимости от настроек, может вернуть код ошибки, либо вернуть ответ, содержащий ЭЦП.

Элемент <respscenario> используется для выбора сценария генерации ответа OCSP сервером в ситуации проверки статуса сертификата, который не подчинен текущей иерархии, задаваемой множеством элементов, отмеченных <signer>[1].

[1] Издатель сертификата не известен, сертификат можно охарактеризовать как “чужой”.

```
DTD:
<!ELEMENT respscenario EMPTY>
<!ATTLIST respscenario
   alienaserror CDATA #REQUIRED>
```

Список атрибутов элемента <respscenario>:

alienaserror - обязательный атрибут, принимает значения ‘true’ или ‘false’. При установленном значении ‘true’ OCSP сервер возвращает ошибку для случая сертификата, который не принадлежит иерархии имен, определяемой набором профилей. При установленном значении ‘false’, в том же случае, сервер генерирует полный ответ с ЭЦП на запрос о статусе сертификата. Ответ содержит статус сертификата ‘Не известен’. При этом, при формировании ЭЦП используются параметры подписи (элемент данных ‘signer’) по умолчанию (см.5.2.5).

5.2.13. Элемент <profile>

Элемент <profile> служит описанием контекста для проверки статуса сертификатов X.509.

```
DTD:
<!ELEMENT profile EMPTY>
<!ATTLIST profile
   name CDATA #REQUIRED
   signer CDATA #REQUIRED
   checkpool CDATA #REQUIRED>
```

Список атрибутов элемента <profile>:

- name - обязательный атрибут, указывает имя профиля обработки запросов OCSP ;
- signer – обязательный атрибут; ссылка на элемент <signer>; значение должно соответствовать значению атрибута name элемента <signer> (см. п. 5.2.5); управляет контекстом подписи;
- checkpool – обязательный атрибут, устанавливает ссылку на элемент <checkpool> (см. 5.2.10); значение должно соответствовать значению атрибута name элемента <checkpool>.

Элемент <ocsp-server> должен содержать хотя бы один элемент <profile>.

Примеры:

```
<profile name =”my” signer="ecgost" checkpool=”LDAP1” >
 </profile>
```

6. Управление приложением

Приложение OCSP Server может быть развёрнуто любым из стандартных методов (см. http://tomcat.apache.org/tomcat-7.0-doc/deployer-howto.html):

- помещением war-файла в каталог appBase (по умолчанию $CATALINA_HOME/webapps); этот способ работает только при значении атрибута autoDeploy=true в элементе <Host> (см. файл конфигурации $CATALINA_HOME/conf/server.xml);
- помещением разархивированного приложения (т.е. каталогов META-INF и WEB-INF) в один из подкаталогов appBase; этот способ также работает только при значении атрибута autoDeploy=true;
- с помощью Tomcat Manager (см. http://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html);
- с использованием Tomcat Client Deployer.

Запуск/останов и удаление приложения OCSP Server производятся также стандартными методами (см. http://tomcat.apache.org/tomcat-7.0-doc/deployer-howto.html).

Примеры использования Tomcat Manager:

```
http://localhost:8080/manager/text/deploy?path=/ocsp-server&war=file:///tmp/ocsp-server.war
http://localhost:8080/manager/text/list
http://localhost:8080/manager/text/stop?path=/ocsp-server
http://localhost:8080/manager/text/start?path=/ocsp-server
http://localhost:8080/manager/text/undeploy?path=/ocsp-server
```

Для использования графической административной HTML-оболочки Apache Tomcat 7 необходимо настроить в файле $CATALINA_HOME/conf/tomcat-users.xml доступ для роли manager-gui, например:

```
…
<role rolename="manager-gui"/>
<user username="admin-gui" password="*****" roles="manager-gui"/>
…
```

В результате при загруженном приложении manager пользователю с идентификатором admin-gui будет доступно приложение по адресу http://localhost:8080/manager/html/, позволяющее осуществлять административные функции.

7. Требования безопасности

Эксплуатация приложения OCSP Server допустима только при полном соблюдении мер защиты, изложенных в Правилах пользования СКЗИ «Крипто-КОМ» (см. [18]).

Ключи электронной подписи и ключевые контейнеры, используемые приложением OCSP Server, должны быть защищены в соответствии с регламентом хранения ключевых носителей СКЗИ «Крипто-КОМ» [17].

Защите от несанкционированного чтения и изменения подлежат также все файлы приложения OCSP Server, включая файл конфигурации.

Меры обеспечения безопасности сервера приложений описаны в соответствующем разделе документации (для Apache Tomcat 7 – http://tomcat.apache.org/tomcat-7.0-doc/security-howto.html).

8. Отладка конфигурации

Для отладки файла конфигурации может понадобиться выводить информацию о ходе загрузке приложения и выполнении основных операций.

Для вывода отладочной информации приложение использует программный интерфейс java.util.logging. Настройка параметров отладки осуществляется стандартным способом, путём редактирования файла $CATALINA_HOME/conf/logging.properties.

Для отладки различных аспектов работы приложения администратор может пользоваться именами классов:

- ru.signalcom.ocsp.server.OCSPConfig;
- ru.signalcom.ocsp.server.OCSPConfigLoader;
- ru.signalcom.ocsp.server.OCSPHandler.

Пример:

```
…
2localhost.org.apache.juli.FileHandler.level = FINEST
2localhost.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
2localhost.org.apache.juli.FileHandler.prefix = localhost.
…
java.util.logging.ConsoleHandler.level = FINE
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
…
ru.signalcom.ocsp.server.OCSPConfig.level = INFO
ru.signalcom.ocsp.server.OCSPConfig.handlers = java.util.logging.ConsoleHandler, 2localhost.org.apache.juli.FileHandler
```

```
        ru.signalcom.ocsp.server.OCSPConfigLoader.level = FINER
        ru.signalcom.ocsp.server.OCSPConfigLoader.handlers =
java.util.logging.ConsoleHandler, 2localhost.org.apache.juli.FileHandler

        ru.signalcom.ocsp.server.OCSPHandler.level = FINEST
        ru.signalcom.ocsp.server.OCSPHandler.handlers =
2localhost.org.apache.juli.FileHandler
```

Литература

1. M, Myers, at al. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, RFC 2560, June 1999.
2. R. Housley, Cryptographic Message Syntax (CMS), RFC 3852, July 1996.
3. ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997.
4. Housley, R., Polk, W., Ford, W. and D. Solo, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 3280, April 2002.
5. S. Kille, A String Representation of Distinguished Names, RFC 1779, March 1995.
6. M.Wahl, S.Kille, T.Howes, Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names, RFC 2253, December 1997.
7. RSA Laboratories. PKCS #1: RSA Cryptography Standard. Version 2.0, October 1, 1998.
8. NIST FIPS PUB 180-1, Secure Hash Standard, May 1993.
9. NIST FIPS PUB 180-4, Secure Hash Standard, March 2012.
10. R. Rivest, The MD5 Message-Digest Algorithm, RFC 1321, April 1992.
11. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритмы криптографического преобразования.
12. ГОСТ Р 34.11-94. Информационная технология. Криптографическая защита информации. Функция хэширования.
13. ГОСТ Р 34.10-2001. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи.
14. ГОСТ Р 34.11-2012. Информационная технология. Криптографическая защита информации. Функция хэширования.
15. ГОСТ Р 34.10-2012. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи.
16. СКЗИ «Крипто-КОМ 3.3». Формуляр. ШКНР.00035-07 30 03. ЗАО Сигнал-КОМ, 2014.
17. СКЗИ «Крипто-КОМ 3.3». Подсистема управления ключевой информацией. Общее описание. ШКНР.00035-07 31 01. ЗАО Сигнал-КОМ, 2014.
18. СКЗИ «Крипто-КОМ 3.3». Правила пользования. ШКНР.00035-07 90 06. ЗАО Сигнал-КОМ, 2014.
19. СКЗИ «Крипто-КОМ 3.3». Параметры криптографических алгоритмов. ШКНР.00035-07 90 03. ЗАО Сигнал-КОМ, 2014.
20. S. Leontiev, D. Shefanovski, Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC 4491, May 2006.
21. Additional cryptographic algorithms for use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 algorithms. V.Popov, I.Kurepkin, S.Leontiev, RFC 4357, January 2006.
22. Java Cryptography Architecture API Specification & Reference. August 4, 2002, http://docs.oracle.com/javase/1.4.2/docs/guide/security/CryptoSpec.html.
23. Signal-COM Java Cryptographic Provider. Версия 3.0. Руководство программиста. ЗАО Сигнал-КОМ, 2014.

24. Admin-PKI. Версия 5.0. Руководство пользователя. ЗАО Сигнал-КОМ, 2014.
25. Методические рекомендации. Использование алгоритмов ГОСТ Р 34.10, ГОСТ Р 34.11 в профиле сертификата и списке отзыва сертификатов (CRL) инфраструктуры открытых ключей X.509. Технический комитет по стандартизации «Криптографическая защита информации» (ТК 26).
26. The file URI Scheme, draft-ietf-appsawg-file-scheme-04. M.Kerwin, Internet-Draft, November 2015.
27. Uniform Resource Identifier (URI): Generic Syntax. T. Berners-Lee W3C/MIT, R. Fielding Day Software, L. Masinter Adobe Systems, RFC 3986, January 2005.