ЗАО «Сигнал-КОМ»

УТВЕРЖДЁН
ШКНР.00054-01 32 09-ЛУ

# ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС УДОСТОВЕРЯЮЩЕГО ЦЕНТРА «NOTARY-PRO 2.8»

# TSP SERVER СЕРВЕР ШТАМПОВ ВРЕМЕНИ

Версия 2.3

Руководство системного программиста

ШКНР.00054-01 32 09
Листов 21

2019

# АННОТАЦИЯ

Приложение Signal-COM TSP Server предназначено для выпуска электронных штампов времени в соответствии с RFC 3161 [1].

Приложение Signal-COM TSP Server является компонентом программно-аппаратного комплекса удостоверяющего центра (ПАК УЦ) «Notary-PRO 2.8».

Настоящий документ содержит руководство системного программиста (администратора) приложения Signal-COM TSP Server.

# СОДЕРЖАНИЕ

# 1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

## 1.1. Назначение

Приложение Signal-COM TSP Server предназначено для выпуска электронных штампов времени в соответствии с RFC 3161 [1]. Приложение поддерживает как российские, так и зарубежные криптографические алгоритмы.

## 1.2. Список сокращений

В настоящем руководстве используются следующие сокращения:

- ДСЧ – датчик случайных чисел;
- ПАК – программно-аппаратный комплекс;
- СКЗИ – средство криптографической защиты информации;
- УЦ – удостоверяющий центр;
- ЭП – электронная подпись;
- CRL – Certificate Revocation List;
- GPS – Global Positioning System;
- HTTP – Hypertext Transfer Protocol;
- ITU-T – International Telecommunication Union - Telecommunication sector;
- OID – Object Identifier;
- NTP – NetworkTtime Protocol;
- RFC – Request for Comments;
- TSA – Time Stamping Authority;
- TSP – Time-Stamp Protocol.

## 1.3. Термины и определения

В настоящем руководстве используются следующие термины:

- ключ электронной подписи – уникальная последовательность символов, предназначенная для создания электронной подписи;
- удостоверяющий центр – юридическое лицо, осуществляющие функции по созданию и выдаче сертификатов ключей проверки электронных подписей;
- сертификат ключа проверки электронной подписи – документ в электронном виде или документ на бумажном носителе, выданные Удостоверяющим центром либо доверенным лицом Удостоверяющего центра и подтверждающие принадлежность ключа проверки электронной подписи владельцу сертификата ключа проверки электронной подписи;
- средство электронной подписи – шифровальные (криптографические) средства, используемые для реализации хотя бы одной из следующих функций - создание электронной подписи, проверка электронной подписи, создание ключа электронной подписи и ключа проверки электронной подписи;
- электронная подпись – информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию.

## 1.4. Технические характеристики

Signal-COM TSP Server использует в качестве транспортного протокол HTTP, в соответствии с разделом 3.4 RFC 3161 [1].

Сервер способен поддерживать любые алгоритмы хэширования, в том числе:

- ГОСТ Р 34.11-2012 [13];
- 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 [14];

- RSA [7].

Максимальная точность штампов времени – 1 мс.

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

Signal-COM TSP Server использует сертификаты ключей проверки электронной подписи в формате ITU-T X.509 [3], RFC 3280 [4], RFC 4491 [19] и рекомендаций ТК26 [**Ошибка! Источник ссылки не найден.**, 21]. Сертификат ключа проверки электронной подписи TSA должен удовлетворять требованиям, изложенным в разделе 2.3 RFC 3161 [1].

Приложение Signal-COM TSP Server обращается к реализации криптографических примитивов через интерфейс JCA [19]. В частности, российские криптографические алгоритмы реализованы в провайдере Signal-COM JCP [18].

При разработке приложения использовано средство криптографической защиты информации (СКЗИ) «Signal-COM JCP 3.1» [16], имеющее сертификат соответствия ФСБ России.

# 2. СТРУКТУРА ПРОГРАММЫ

Приложение Signal-COM TSP Server состоит из следующих компонентов:

- tsp-server.war – архив, содержащий приложение и необходимые библиотеки;
- tsp-conf.xml – файл конфигурации.

# 3. НАСТРОЙКА ПРОГРАММЫ

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

## 3.1. Подготовка окружения

Для развёртывания приложения Signal-COM TSP 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.

## 3.2. Быстрый старт

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

Для быстрого старта приложения выполните следующие действия:

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

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

Тестовые наборы ключей и тестовый идентификатор политики штампов времени (1.3.6.1.4.1.5849.3.4.1) не должны использоваться при штатной эксплуатации приложения Signal-COM TSP Server.

## 3.3. Источник точного времени

При формировании штампов времени приложением используется системное время, обязанность обеспечения точности системных часов возлагается на администратора.

Системное время, в зависимости от требований к точности и безопасности, может синхронизироваться от:

- автономного источника времени (использующего атомные часы либо GPS/ГЛОНАСС-приёмник);
- сетевого источника времени (по протоколу NTP/SNTP).

## 3.4. Создание ключей ЭП и ключей проверки ЭП

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

- приложение Admin-PKI [20];
- приложение keytool из состава JRE/JDK.

Для создания ключей электронной подписи и ключей проверки электронной подписи ГОСТ Р 34.10-2012 должны использоваться средства криптографической защиты информации, сертифицированные ФСБ России.

## 3.5. Датчик случайных чисел

При создании электронной подписи ГОСТ Р 34.10-2012 приложение Signal-COM TSP Server должно использовать датчик случайных чисел «GOST28147PRNG», реализованный в СКЗИ «Signal-COM JCP 3.1» [18].

Инициализация датчика случайных чисел «GOST28147PRNG» производится от вектора состояния, хранящегося в ключевом контейнере СКЗИ «Signal-COM JCP 3.1» [16]. Приложение Signal-COM TSP Server не содержит средств генерации ключевых контейнеров – эта операция должна производиться с использованием любых других средств, использующих СКЗИ (например, Admin-PKI [20]).

В составе дистрибутива (в каталоге native) поставляются платформозависимые модули, необходимые для работы датчика случайных чисел СКЗИ «Signal-COM JCP 3.1». Модули представляют собой динамические библиотеки:

- rdtsc.dll - для Windows;
- librdtsc.so - для Linux и Solaris.

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

## 3.6. Конфигурирование

Конфигурирование приложения Signal-COM TSP Server осуществляется путём редактирования двух файлов: дескриптора развёртывания (см. п. 3.7) и файла конфигурации (см. п. 3.10).

## 3.7. Дескриптор развёртывания

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

В дескрипторе 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/tsp-conf.xml</param-value>
</context-param>
…
```

Журналирование для приложения Signal-COM TSP Server настраивается путём редактирования параметров фильтра serverLogger. Журналирование может осуществляться одним из следующих способов:

- с использованием стандартного программного интерфейса java.util.logging (см. п. 3.8);
- с использованием базы данных (см. п. 3.9).

В дескрипторе развёртывания текущей версии приложения не должен использоваться элемент <distributable/>.

## 3.8. Журналирование с использованием log4j через slf4j

При использовании log4j через slf4j в файле web.xml должен быть настроен фильтр с именем serverLogger, реализованный в классе ru.signalcom.tsp.server.TSPDummyLogger.

Список необходимых библиотек (указаны минимальные номера версий):

- slf4j-api-1.7.21.jar
- log4j-api-2.6.2.jar
- log4j-core-2.6.2.jar
- log4j-slf4j-impl-2.6.2.jar
- log4j-web-2.6.2.jar

В файл $CATALINA_HOME/conf/catalina.properties необходимо добавить свойство:

```
log4j.configurationFile=${sys:catalina.base}/conf/log4j2.xml
```

Настройка параметров журналирования осуществляется путём редактирования файла $CATALINA_HOME/conf/log4j2.xml.

Для сохранения параметров запроса и ответа сервера уровень журналирования для класса ru.signalcom.tsp.server.TSPDummyLogger должен быть установлен в значение DEBUG, например:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN" monitorInterval="60">
  <Appenders>
    <RollingFile name="RollingFileTSPServer"
fileName="${sys:catalina.base}/logs/tsp-server.log"
                 filePattern="${sys:catalina.base}/logs/archives/$${date:yyyy-MM}/tsp-
server-%d{yyyy-MM-dd}-%i.log.gz">
      <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %-5level
%logger{36} - %msg%n" />
      <Policies>
        <TimeBasedTriggeringPolicy />
        <SizeBasedTriggeringPolicy size="1000 MB"/>
      </Policies>
      <DefaultRolloverStrategy max="100000"/>
    </RollingFile>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %-5level
%logger{36} - %msg%n" />
    </Console>
  </Appenders>
  <Loggers>
    <Logger name="ru.signalcom.tsp.server" level="DEBUG" additivity="false">
      <AppenderRef ref="RollingFileTSPServer"/>
    </Logger>
    <Root level="ERROR">
      <AppenderRef ref="Console"/>
    </Root>
  </Loggers>
</Configuration>
```

При использовании log4j через slf4j серийные номера штампов времени, создаваемых сервером, могут формироваться одним из следующих способов:

- неявно, т. е. автоматически (см.п. 3.8.1);
- с сохранением серийного номера в файле (см. п. 3.8.2).

### 3.8.1. Автоматическое формирование серийного номера

Для автоматического формирования серийных номеров штампов времени достаточно не указывать никаких параметров в настройках фильтра serverLogger:

```
…
<filter>
  <filter-name>serverLogger</filter-name>
  <filter-class>ru.signalcom.tsp.server.TSPDummyLogger</filter-class>
</filter>
…
```

### 3.8.2. Файл серийного номера

Серийный номер штампа времени сохраняется в файле, задаваемом параметром фильтра serverLogger с именем serialNumberFileName, например:

```
…
<filter>
```

```
    <filter-name>serverLogger</filter-name>
    <filter-class>ru.signalcom.tsp.server.TSPDummyLogger</filter-class>
    <init-param>
      <param-name>serialNumberFileName</param-name>
      <param-value>${catalina.home}/conf/tsp-serial</param-value>
    </init-param>
  </filter>
  …
```

Файл с серийным номером представляет собой текстовый файл, в котором в десятичном представлении хранится значение последовательного номера, включённого в последний выпущенный сервером штамп времени. Максимальное значение серийного номера – $2^{64}$-1. Значение серийного номера в файле обновляется приложением автоматически; редактирование значения серийного номера вручную должно осуществляться администратором лишь в крайних случаях и только при незагруженном приложении.

## 3.9. Журналирование с использованием базы данных

Для ведения журнала в базе данных необходима установка соответствующего JDBC-драйвера (например, JDBC-драйвер для MySQL доступен для скачивания по адресу http://dev.mysql.com/downloads/connector/j/).

В файле web.xml должен быть настроен фильтр с именем serverLogger, реализованный в классе ru.signalcom.tsp.server.TSPDatabaseLogger, например:

```
  …
  <filter>
    <filter-name>serverLogger</filter-name>
    <filter-class>ru.signalcom.tsp.server.TSPDatabaseLogger</filter-class>
    <init-param>
      <param-name>dataSourceName</param-name>
      <param-value>jdbc/TSP</param-value>
    </init-param>
  </filter>
  …
```

Параметр фильтра с именем dataSourceName должен указывать на источник данных, настраиваемый в файле $CATALINA_HOME/conf/context.xml, например:

```
  <Context>
    …
    <Resource name="jdbc/TSP"
          auth="Container"
          type="javax.sql.DataSource"
          maxActive="100"
          maxIdle="30"
          maxWait="10000"
          username="root"
          password="******"
          driverClassName="com.mysql.jdbc.Driver"
          url="jdbc:mysql://localhost:3306/tsp_server?autoReconnect=true" />
    …
  </Context>
```

Для создания таблицы в базе данных необходимо использовать SQL-запрос, приведённый в файле db/create.sql дистрибутива.

В качестве серийного номера выпускаемых сервером штампов времени используется значение автоинкрементного поля таблицы tsp. Максимальное значение серийного номера – $2^{64}$-1.

## 3.10. Файл конфигурации

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

В следующих разделах содержится подробное описание синтаксиса файла конфигурации.

### 3.10.1. Элемент <tsp-server>

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

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

Список атрибутов элемента <tsp-server>:

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

Элемент <tsp-server> должен содержать следующие элементы:

- <hashalg> – см. п. 3.10.2;
- <signer> – см. п. 3.10.5;
- <profile> – см. п. 3.10.10.

Элемент <tsp-server> может содержать следующие элементы:

- <random> – см. п. 3.10.3;
- <keystore> – см. п. 3.10.4.

### 3.10.2. Элемент <hashalg>

Элемент <hashalg> описывает алгоритм хэширования.

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

Список атрибутов элемента <hashalg>:

- name – обязательный атрибут, задающий символьное имя алгоритма хэширования;
- oid – необязательный атрибут, содержащий OID алгоритма хэширования в десятично-точечном представлении;
- mdlen – необязательный атрибут, указывающий длину хэш-значения в байтах.

Элемент <tsp-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 атрибуты oid и mdlen могут не указываться:

- «GOSTR3411-2012-512» – ГОСТ Р 34.11-2012 с длиной хэш-значения 512 бит [13];
- «GOSTR3411-2012-256» – ГОСТ Р 34.11-2012 с длиной хэш-значения 256 бит [13];
- «GOST3411-CP» – ГОСТ Р 34.11-94 с узлами замены id-GostR3411-94-CryptoProParamSet [22];
- «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"/>
<hashalg name="MD5" oid="1.2.840.113549.2.5" mdlen="16"/>
```

### 3.10.3. 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 СКЗИ «Signal-COM JCP 3.1»;
- 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 <tsp-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"/>
```

### 3.10.4. 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 <tsp-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="*****"/>
```

### 3.10.5. Элемент <signer>

Элемент <signer> описывает контекст подписи TSA.

```
DTD:
<!ELEMENT signer ((keyentry, signature?) | (priv, cert, signature?))>
```

```
<!ATTLIST signer
    name CDATA #REQUIRED
    type CDATA #REQUIRED
    init 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.

Атрибут type может принимать следующие значения:

- keystore – если ключ электронной подписи и сертификат ключа проверки электронной подписи располагаются в хранилище (это рекомендуемый способ); подробнее см. разделы 3.10.4, 3.10.6;
- file – если ключ электронной подписи и сертификат ключа проверки электронной подписи хранятся в отдельных файлах; подробнее см. разделы 3.10.7, 3.10.8.

Атрибут init в текущей версии приложения может принимать единственное значение:

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

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

Элемент <signer> должен содержать либо элемент <keyentry> (см. п. 3.10.6), либо пару элементов: <priv> (см. п. 3.10.7) и <cert> (см. п. 3.10.8). Элемент <signer> может также содержать элемент <signature> (см. п. 3.10.9)

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

Примеры:

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

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

### 3.10.6. Элемент <keyentry>

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

```
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. 3.10.4);
- alias – задаёт имя записи в хранилище, указывающее на ключ электронной 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. 3.10.3); обязательный атрибут для ключей электронной подписи ГОСТ Р 34.10-2012.

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="tsa-ecgost" random="ecgost"/>
</signer>

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

### 3.10.7. Элемент <priv>

Элемент <priv> служит для описания ключа подписи TSA. Должен использоваться совместно с элементом <cert> (см. п. 3.10.8), описывающим сертификат TSA.

```
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 электронной подписи;
- type – необязательный атрибут; наименование представления ключа электронной подписи, по умолчанию – «PKCS#8»;
- provider – необязательный атрибут; имя JCA-провайдера.
- password – содержит пароль к ключу электронной подписи; необязательный атрибут;
- random – ссылка на элемент <random>; значение должно соответствовать значению атрибута name элемента <random> (см. п. 3.10.3); обязательный атрибут для ключей электронной подписи ГОСТ Р 34.10-2012.

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

Примеры:

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

### 3.10.8. Элемент <cert>

Элемент <cert> служит для описания сертификата ключа проверки электронной подписи TSA. Должен использоваться совместно с элементом <priv> (см. п. 3.10.7), описывающим ключ электронной подписи TSA.

```
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 ключа проверки электронной подписи;

- type – необязательный атрибут; наименование представления сертификата ключа проверки электронной подписи, 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/tsa.p8" random="ecgost" password="*****"/>
   <cert type="X.509" provider="SC" file="/some/dir/tsa.cer"/>
</signer>
```

### 3.10.9. Элемент <signature>

Элемент <signature> служит для описания параметров электронной подписи и должен использоваться как вложенный элемент элемента <signer> (см. п. 3.10.5).

```
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. 3.10.2); служит для задания алгоритма хэширования при создании электронной подписи TSA под штампом времени; 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="tsa-rsa"/>
   <signature hashalg="SHA-256"/>
</signer>
```

### 3.10.10. Элемент <profile>

Элемент <profile> служит для описания контекста штампов времени.

```
DTD:
<!ELEMENT profile (accuracy?, imprint*)>
<!ATTLIST profile
      policy CDATA #REQUIRED
      signer CDATA #REQUIRED
      includetsa CDATA #IMPLIED
      default CDATA #IMPLIED>
```

Список атрибутов элемента <profile>:

- policy – обязательный атрибут; содержит идентификатор политики штампов времени (OID) в десятично-точечном представлении;
- signer – обязательный атрибут; ссылка на элемент <signer>; значение должно соответствовать значению атрибута name элемента <signer> (см. п. 3.10.5); управляет контекстом электронной подписи штампов времени;
- includetsa – управляет включением имени TSA в штамп времени; может принимать значение true или false; необязательный атрибут – при отсутствии атрибута используется значение false;
- default – указывает, является ли контекст используемым по умолчанию; может принимать значение true или false; контекст по умолчанию используется при отсутствии явного указания идентификатора политики в запросе на штамп времени; необязательный атрибут – при отсутствии атрибута используется значение false; только один контекст в конфигурации может быть помечен как контекст по умолчанию.

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

Элемент <profile> может содержать элемент <accuracy> (см. п. 3.10.11) и элементы <imprint> (см. п. 3.10.12).

Примеры:

```
<profile default="true" policy="1.3.6.1.4.1.5849.3.4.1" signer="rsa"/>

<profile policy="1.3.6.1.4.1.5849.3.4.3" signer="ecgost" includetsa="true">
   <accuracy seconds="1" millis="0"/>
   <imprint hashalg="GOST3411-CP"/>
</profile>
```

### 3.10.11. Элемент <accuracy>

Элемент <accuracy> служит для описания точности штампов времени.

```
DTD:
<!ELEMENT accuracy EMPTY>
<!ATTLIST accuracy
      seconds CDATA #IMPLIED
      millis CDATA #IMPLIED
      micros CDATA #IMPLIED>
```

Список атрибутов элемента <accuracy>:

- seconds – необязательный атрибут; содержит число секунд в представлении точности штампа времени; по умолчанию используется значение 0;
- millis – необязательный атрибут; содержит число миллисекунд в представлении точности штампа времени; атрибут должен иметь значение в диапазоне от 0 до 999; по умолчанию используется значение 0;
- micros – необязательный атрибут; содержит число микросекунд в представлении точности штампа времени; атрибут должен иметь значение в диапазоне от 0 до 999; в текущей версии приложения всегда используется значение 0.

Элемент <profile> может содержать только один элемент <accuracy>. При отсутствии элемента <accuracy> точность времени в штампе времени не указывается.

Элемент <accuracy> не содержит вложенных элементов.

Примеры:

```
<profile policy="1.3.6.1.4.1.5849.3.4.3" signer="ecgost" includetsa="true">
   <accuracy seconds="1" millis="0"/>
</profile>
```

### 3.10.12. Элемент <imprint>

Элемент <imprint> служит для описания алгоритмов хэширования, поддерживаемых текущим контекстом штампов времени. Должен использоваться как вложенный элемент элемента <profile> (см. п. 3.10.10).

```
DTD:
<!ELEMENT imprint EMPTY>
<!ATTLIST imprint
      hashalg CDATA #REQUIRED
      signer CDATA #IMPLIED>
```

Список атрибутов элемента <imprint>:

- hashalg – обязательный атрибут, задающий ссылку на элемент <hashalg>; значение должно соответствовать значению атрибута name элемента <hashalg> (см. п. 3.10.2); служит для задания алгоритма хэширования при создании электронной подписи TSA под штампом времени; в текущей версии приложения может использоваться только с ключами RSA;
- signer – необязательный атрибут; ссылка на элемент <signer>; значение должно соответствовать значению атрибута name элемента <signer> (см. п. 3.10.5); используется при необходимости перекрыть значение атрибута signer, задаваемое в элементе <profile> (см. п. 3.10.10).

Элемент <signer> может содержать несколько элементов <imprint>. Если элемент <signer> не содержит ни одного элемента <imprint>, то в контексте поддерживаются все алгоритмы хэширования, задаваемые элементами <hashalg> (см. п. 3.10.2) текущей конфигурации.

Элемент <imprint> не содержит вложенных элементов.

Примеры:

```
<profile policy="1.3.6.1.4.1.5849.3.4.1" signer="ecgost">
  <imprint hashalg="GOST3411-CP"/>
  <imprint hashalg="SHA-256" signer="rsa"/>
</profile>
```

## 3.11. Управление приложением

Приложение Signal-COM TSP 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 Signal-COM TSP 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=/tsp-server&war=file:///tmp/tsp-server.war
http://localhost:8080/manager/text/list
http://localhost:8080/manager/text/stop?path=/tsp-server
http://localhost:8080/manager/text/start?path=/tsp-server
http://localhost:8080/manager/text/undeploy?path=/tsp-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.

## 3.12. Требования по безопасности

Эксплуатация приложения Signal-COM TSP Server допустима только при полном соблюдении мер защиты, изложенных в Правилах пользования СКЗИ «Signal-COM JCP 3.1» (см. [17]).

Ключи электронной подписи и ключевые контейнеры, используемые приложением Signal-COM TSP Server, должны быть защищены в соответствии с регламентом хранения ключевых носителей СКЗИ «Signal-COM JCP 3.1» [16].

Защите от несанкционированного чтения и изменения подлежат также все файлы приложения Signal-COM TSP 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).

# 4. ПРОВЕРКА ПРОГРАММЫ

Проверка работоспособности приложения Signal-COM TSP Server осуществляется с использованием средств администрирования сервера приложений, а также путём анализа записей в журнале событий (см. п. 5).

# 5. СООБЩЕНИЯ СИСТЕМНОМУ ПРОГРАММИСТУ

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

Для вывода отладочной информации приложение использует библиотеку slf4j (подробнее см. п. 3.8)

# ЛИТЕРАТУРА

1. C. Adams, P. Cain, D. Pinkas, R. Zuccherato, Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), RFC 3161, August 2001.
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 34.11-2012. Информационная технология. Криптографическая защита информации. Функция хэширования.
14. ГОСТ Р 34.10-2012. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи.
15. СКЗИ «Signal-COM JCP 3.1». Формуляр. ШКНР.00049-01 30 01. ЗАО Сигнал-КОМ, 2019.
16. СКЗИ «Signal-COM JCP 3.1». Подсистема управления ключевой информацией. Общее описание. ШКНР.00049-01 31 01. ЗАО Сигнал-КОМ, 2019.
17. СКЗИ «Signal-COM JCP 3.1». Правила пользования. ШКНР.00049-01 90 01. ЗАО Сигнал-КОМ, 2019.
18. СКЗИ «Signal-COM JCP 3.1». Инструкция по встраиванию. ШКНР.00049-01 33 01. ЗАО Сигнал-КОМ, 2019.
19. Java Cryptography Architecture API Specification & Reference, https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html
20. Admin-PKI. Версия 5.3. Руководство пользователя. ЗАО Сигнал-КОМ, 2019.
21. Р 1323565.1.023-2018 «Информационная технология. Криптографическая защита информации. Использование алгоритмов ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012 в сертификате, списке аннулированных сертификатов (CRL) и запросе на сертификат PKCS #10 инфраструктуры открытых ключей X.509». Рекомендация по стандартизации. Технический комитет по стандартизации «Криптографическая защита информации» (ТК 26), 2018.
22. 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.