GS LABS
GS GROUP

Публичный - L0

Сервер передачи ключей Keys Transfer Server (KTS)

Руководство по установке

Индекс	KTS-IG
Конфиденциальность	Публичный - L0
Ревизия	1.0
Статус	Согласован

Содержание

1. Термины и сокращения

Термин	Определение
API (Application Programming Interface, Интерфейс программирования приложений)	Набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. Используется программистами для написания всевозможных приложений.
FTP (File Transfer Protocol)	Протокол передачи файлов – стандартный протокол, предназначенный для передачи файлов по TCP-сетям (например, Интернет). Протокол построен на архитектуре "клиент-сервер" и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером.
KGS	Система генерации ключей Keys Generation System (KGS). Продукт ООО "ПЦТ" для работы с ключами: генерация, экспорт, импорт, управление.
KTS сервер	Сервер, на котором развернуты bbx_server_go и база данных (компоненты "Сервера передачи ключей Keys Transfer Server (KTS)").
OTP-ключи (One Time Programmable)	Ключи, которые прошиваются в однократно программируемую область памяти в чипе.
Номер Партии (Part Number, PN)	Сущность, которая характеризует заказы чипов у производителя (чип-вендора). Как правило, в чипах используемое имя part number, выгравированное на корпусе. В KGS сущность кроме имени имеет внутренний идентификатор. В KGS имя задается пользователем и может не совпадать с маркировкой на чипе.
Тип Партии (Part Type, PT, PTTP)	Сущность, которая характеризует набор общих OTP-ключей в чипах одной партии: у всех чипов одной PT одинаковые общие ключи. В чипах используется только идентификатор Part Type ID, в KGS также используется пользовательское имя. Не следует путать Тип Партии с Номером Партии.

Сокращение	Расшифровка
БД	База данных
API	Application Programming Interface, Интерфейс программирования приложений
ID	Identifier
OTP	One-Time Programmable
TDE	Transparent Database Encryption

2. Введение

2.1. Назначение документа

Данное руководство описывает установку, настройку и последующее администрирование Сервера передачи ключей Keys Transfer Server (KTS) (далее по тексту - KTS или Система), включающего сервера, базы данных и клиентскую часть (последняя устанавливается на сервер персонализации).

> ⓘ Данный документ опубликован исключительно с целью изучения системных требований для установки продукта, а также ознакомления с последовательностью и деталями процесса установки.

2.2. Общее описание системы

Сервер передачи ключей Keys Transfer Server предназначен для доставки ключей на производственную линию и занесения их уникального набора в однократно-программируемую область чипа в процессе его персонализации.

Программное обеспечение предоставляет инфраструктуру для персонализации, с возможностью:

- импорта ключей из системы KGS для партии чипов, их загрузки и сохранения в системе;
- настройки структуры базы данных в зависимости от типа устройства, для которого предназначены ключи;
- персонализации различных типов чипов путем конфигурирования системы;
- хранения ключей в зашифрованном виде;
- фиксации текущего статуса персонализации и сохранения результатов персонализации для каждого чипа;
- формирования отчетности.

Язык программирования: C++, Go

Описания компонентов Системы и принципов их работы содержатся в "Общем описании" и "Техническом описании" (доступ предоставляется по запросу).

2.3. Требования к квалификации администратора

Для администрирования системы необходимо наличие навыков работы с консольной версией ОС Debian, а именно:

- создание разделов дисков, установка пакетов;
- создание и настройка сетевых подключений;
- установка и настройка PostgreSQL.

2.4. Системные требования

2.4.1. Требования к аппаратному обеспечению серверов

- Двух- или четырех- ядерный процессор с поддержкой виртуализации.
- 4-8 GB RAM (64-bit).

- Сетевой интерфейс.

Необходимое количество и объем жестких дисков: два жестких диска, по 1 TB каждый (или четыре, каждый по 500 GB), RAID-1.
В целях безопасности должен быть зашифрован раздел, на который будут установлены виртуальные машины (по одной на каждый экземпляр bbx_server_go) и базы данных. Шифрование производится стандартными средствами ОС Debian.

2.4.2. Требования к программному обеспечению

Для KTS Server:

- ОС Debian 11, 64bit
- СУБД PostgreSQL 13.***x***
- Интерфейс управления БД ODBC (рекомендуется последняя версия на момент установки).
- Библиотека OpenSSL (рекомендуется последняя версия на момент установки).

Для Programming Server (клиентская машина) возможно использование различных unix-ОС (Debian и др.).

- Требуется установка пакета OpenSSL. Рекомендуемая версия - **3.0.7**.

3. Компоненты, необходимые для установки системы

При установке, настройке и работе системы используются несколько раздельно поставляемых подсистем, каждая из которых отвечает за часть общего функционала.
Ниже приводится описание необходимых компонентов данных подсистем.

3.1. Подсистема bbx_server_go

Для работы bbx_server_go требуются:

- Исполняемый файл bbx_server_go, входит в комплект поставки.
- Файл *bbx_server_go.cfg.dft* (файл настроек), входит в комплект поставки.
- Файлы *cacert.pem*, *cakey.pem* и т.д., не входящие в комплект поставки, генерируются с помощью скрипта создания SSL-ключей.

Для создания SSL-ключей серверной и клиентской частей используется скрипт *SSL_keygen_script.sh*, входящий в комплект поставки. (Процесс создания SSL-ключей описывается в разделе Создание SSL ключей и сертификатов)

Компоненты подсистемы поставляются в виде установочного deb-пакета ***bbx_server_go-X.X.X***.deb***

3.2. Подсистема создания таблиц базы данных (BBX_CHIP_DB_SCH)

Сборка *bbx_chip_db_sch*, включающая:

- Папку *sql*, содержащую файлы скриптов создания схем баз данных, таблиц и пользователей
- Папку *common_db* со скриптами для установки подсистемы.
- Файлы *install.sh* и *install.bat* для установки подсистемы.

3.3. Подсистема API для работы с базой данных (BBX_CHIP_DB_API)

Сборка *bbx_chip_db_api*, включающая:

- Папку *common_db* со скриптами для установки подсистемы.
- Папку *scripts* с файлами скриптов для загрузки конфигурации, ключей, создания отчета о программировании.
- Папку *sql* с процедурами работы с БД.
- Папку *types* со структурами sql.
- Файлы *install.sh* и *install.bat* для установки подсистемы.

3.4. Скрипты для эксплуатации системы

Скрипты входят в состав KTS и лежат в отдельном репозитории (доступ к репозиторию предоставляется по запросу)

3.5. Драйверы HASP-ключа

- *aksusbd_**x.x**-1_i386.deb*

Данный пакет копируется в удобную папку на KTS Server.

3.6. Библиотеки, относящиеся к системе KGS

Пакеты библиотек KGS:

- *kmi_fw_hwrk-X.X.X-linux-x86_64.deb*
- *kmi_fw_tde-X.X.X-linux-x86_64.deb*

Файлы автоматически устанавливаются в папку */opt/kmi/lib* на KTS Server.

3.7. Подсистема клиентского сервера (BBX_CHIP_CLIENT)

Компоненты клиентской подсистемы (BBX_CHIP_CLIENT).

- bbx_chip_client-X.X.X_amd64.tar.gz - в случае если клиентская машина работает под управлением ОС семейства Linux 64bit.
  Содержит:
  - bbx_chip_cli.h
  - libbbx_chip_client.a
  - libbbx_chip_client.so

Файлы, необходимые для использования библиотеки:

- libbbx_chip_cli.a (статическая библиотека). Путь к библиотекам указывается при сборке Programming application (входит в комплект поставки).
- bbx_chip_cli.h – заголовочный файл с API для работы с клиентской библиотекой (входит в комплект поставки).
- cacert.pem, key.pem, client.pem – ssl ключи, создаются заранее (см. далее). Должны находиться в папке с исполняемым файлом.
- исполняемый файл с реализацией программы, использующей API библиотеки.

4. Подготовка серверных машин

4.1. Подготовка к настройке

4.1.1. Разбивка дисков на разделы

Для нормальной работы bbx_server_go требуется создать на диске следующие разделы:

- root 2Gb
- usr 19Gb
- var 14Gb
- tmp 1Gb
- home 17Gb
- swap 4-6Gb
- opt (все оставшееся место на диске) – этот раздел будет зашифрован и на нем будет разворачиваться виртуальная машина.

4.1.2. Установка операционной системы

На машину (KTS Server), предназначенную для bbx_server_go, предварительно стандартным образом устанавливается ОС Debian 11.

4.1.3. Установка пакетов

Для нормальной работы Системы требуется предварительно установить пакеты:

- python 3.***x*** (третьей версии)
- python3-psycopg2

  ⚠ Python3 обычно входит в состав образа Debian ("устанавливается из коробки").

- postgresql-contrib
- openssl
- lshw

4.2. Настройка сервера

Порядок действий:

1. Установить PostgreSQL.
2. Отредактировать конфигурационный файл PostgreSQL pg_hba.conf. Конфигурация параметров для этого файла предоставляется по запросу.
3. Перезапустить PostgreSQL:

```
sudo service postgresql restart
```

4. Установить ODBC:

```
sudo apt-get install odbc-postgresql
```

5. Отредактировать файл *odbcinst.ini*. Конфигурация параметров для этого файла предоставляется по запросу.
6. Отредактировать файл *odbc.ini*. Конфигурация параметров для этого файла предоставляется по запросу.
7. Установить openssl, lshw:

```
sudo apt-get install openssl lshw
```

5. Установка компонентов

5.1. Установка подсистемы bbx_server_go

Компоненты данной подсистемы поставляются в составе deb-пакета (файл *bbx_server_go-X.X.X-***.deb* , где X.X.X - номер текущей версии пакета).

5.1.1. Установка подсистемы

Для установки требуется скопировать установочный файл на KTS Server и запустить:

```
sudo dpkg -i bbx_server_go-X.X.X-***.deb
```

Пакет будет установлен в папку */opt/chipblackbox/*.

5.1.2. Удаление подсистемы

В случае наличия обновленной версии подсистемы, перед её установкой требуется удалить прежнюю версию.

Для удаления следует запустить команду удаления deb-пакета bbx_server_go:

```
sudo dpkg -r bbx-server-go
```

5.1.3. Автозапуск службы bbx_server_go

Перед запуском служба ожидает 10 секунд - это необходимая мера, которая позволяет убедиться, что при включении системы служба запустится после того, как запустится кластер базы данных (в случае, если сервер и база данных расположены на одной системе). Также задержка уменьшит рост объема лог-файла из-за неудачных подряд попыток запуска сервера (например, если удаленный сервер с БД пропал из сети, а сервер пытается каждый раз к нему подключиться).

Автозапуск гарантирован, даже если процесс был принудительно завершен командой типа:

```
sudo pkill bbx_server_go
```

Имейте это в виду, если вам нужно, чтобы сервер был выключен на необходимое вам время.

Чтобы остановить сервер до следующей перезагрузки, введите команду:

```
sudo systemctl stop bbx_server_go.service
```

Чтобы принудительно включить сервер обратно:

```
sudo systemctl start bbx_server_go.service
```

Чтобы отключить автозапуск:

```
sudo systemctl disable bbx_server_go.service
```

Чтобы включить автозапуск обратно:

```
sudo systemctl enable bbx_server_go.service
```

- Команду обязательно надо ввести вручную, чтобы автозапуск заработал. При установке .deb пакета автоматически автозапуск работать не будет.
- После выполнения этой команды нужно выполнить еще одну, т.е.:

  ```
  sudo systemctl enable bbx_server_go.service
  sudo systemctl daemon-reload
  ```

  Дополнительная команда обновляет конфигурацию службы bbx_server_go. При установке KTS "с нуля", эту команду можно пропустить.

Чтобы проверить статус автозапуска:

```
sudo systemctl is-enabled bbx_server_go.service
```

Если в результате вы получили "enabled", значит автозапуск функционирует.

5.2. Создание базы данных

Особенности:

- Скрипты выполняются от superuser
- Должны быть выданы права для postgres на папки с компонентами
- При любом расположении архива табличное пространство будет всегда расположено в */db /bbx_idx_tablespace*

5.2.1. Настройка локализации

Проверьте, что у вас активна локаль **ru_RU.UTF-8**. Например, в Debian это можно сделать так:

1. Выполните команду:

   ```
   sudo dpkg-reconfigure -plow locales
   ```

2. Убедитесь, что в списке локализаций отмечена **ru_RU.UTF-8**. Если это не так, выберите её в добавок к уже имеющимся и нажмите *Ok*.

3. В окне выбора локали по умолчанию выберите **en_US.utf8** и нажмите *Ok*.
4. Проверьте, что вывод имеет вид:

```
Generating locales (this might take a while)...
   en_US.UTF-8... done
   ru_RU.UTF-8... done
Generation complete.
```

5. Проверьте успешность выполнения команды. Проверка с помощью команды *locale -a*:

```
$ locale -a
C
C.UTF-8
en_US.utf8
POSIX
ru_RU.utf8
```

6. Перезагрузите систему, чтобы изменения вступили в силу:

```
sudo reboot
```

5.2.2. Создание таблиц базы данных

За создание таблиц базы данных отвечает подсистема *BBX_CHIP_DB_SCH*.

Порядок действий:

1. Создать папку */db/bbx_idx_tablespace* и предоставить права доступа к этой папке пользователю *postgres*:

```
sudo chown -R postgres /db/bbx_idx_tablespace
```

⚠ Директория с табличным пространством обязательно должна быть в корневой директории.

2. Распаковать архив bbx_chip_db_sch-X.X.X.zip, входящий в комплект поставки, в желаемую папку на разделе диска, где установлен PostgreSQL (рекомендуется */DB/BBX_CHIP_DB_SCH/*).

⚠ **Обратите внимание!** При любом расположении архива bbx_chip_db_***.zip табличное пространство будет всегда расположено в */db/bbx_idx_tablespace*

3. Предоставить права доступа к данной папке пользователю *postgres*:

```
sudo chown -R postgres db/bbx_chip_db_sch
```

⚠ Если при выполнении следующего шага (шаг 3) будет отказано в доступе, то необходимо выполнить команду:

```
sudo chmod -R +x db/bbx_chip_db_sch
```

4. Запустить (под sudo) скрипт check_install_sch.sh (скрипт лежит в подпапке common_db) со следующими параметрами:

```
bash check_install_sch.sh $1 $2 $3 $4 $5 $6 $7 $8 $9
```

где:

a. $1 - DB_HOST (host ip)
b. $2 - DB_PORT (host port)
c. $3 - DB_NAME (database name)
d. $4 - USER_NAME (admin user name)
e. $5 - USER_PASSWORD (admin user password)
f. $6 - PG_USER (postgres user name)
g. $7 - PG_PASSWORD (postgres user password)
h. $8 - DB_SCHEME (database scheme)
i. $9 - Additional params

5. Пример:

```
bash common_db/check_install_sch.sh 127.0.0.1 5432 bbx bbxadmin bbxadmin postgres postgres bbx
'TBS_IDX=bbx_idx_tablespace'
```

6. Проверить лог-файл install.log (в подпапке *common_db*) на отсутствие ошибок.

⚠ Если в лог-файле есть ошибки, то нужно переустановить схему базы данных:

a. Подключиться к СУБД (зайти в psql):

```
psql -U postgres
```

b. Удалить базу данных, выполнив команду:

```
DROP DATABASE bbx;
```

c. Удалить табличное пространство, выполнив команду:

```
DROP TABLESPACE bbx_idx_tablespace;
```

d. Устанавливать схему БД заново.

5.2.3. Установка API управления базой данных

За управление базой данных отвечает подсистема *BBX_CHIP_DB_API*.

Порядок действий:

1. Распаковать архив bbx_chip_db_api_X.X.X.zip, входящий в комплект поставки, в желаемую папку на разделе диска, где установлен PostgreSQL (рекомендуется */DB/BBX_CHIP_DB_SCH/*).
2. Предоставить права доступа к данной папке пользователю *postgres*:

```
sudo chown -R postgres db/bbx_chip_db_api
```

> ⚠ Если при выполнении следующего шага (шаг 3) будет отказано в доступе, то необходимо выполнить команду:
>
> `sudo chmod -R +x db/bbx_chip_db_api`

3. Запустить (под sudo) скрипт check_install_api.sh (скрипт лежит в подпапке common_db) со следующими параметрами:

```
bash check_install_api.sh $1 $2 $3 $4 $5 $6 $7 $8
```

где:

a. $1 - DB_HOST (host ip)
b. $2 - DB_PORT (host port)
c. $3 - DB_NAME (database name)
d. $4 - USER_NAME (admin user name)
e. $5 - USER_PASSWORD (admin user password)
f. $6 - PG_USER (postgres user name)
g. $7 - PG_PASSWORD (postgres user password)
h. $8 - DB_SCHEME (database scheme)

4. Пример:

```
bash common_db/check_install_api.sh 127.0.0.1 5432 bbx bbxadmin bbxadmin postgres postgres bbx
```

5. Проверить лог-файл install.log (в подпапке *common_db*) на отсутствие ошибок.

5.3. Установка драйверов HASP-ключа

Для повышения безопасности в комплект поставки Системы входит HASP-ключ и соответствующие драйверы.

> ⚠ **Внимание!**
>
> Без присутствующего и работающего HASP-ключа невозможны генерация ключа HWRK и запуск bbx_server_go.
>
> Однако, при установке библиотек KGS (см. ниже) можно задать параметр KMI_HASP_VERSION=no_hasp, позволяющий не пользоваться HASP.

Порядок действий:

1. Поместить прилагаемый HASP-ключ в порт USB сервера KTS.
2. Выполнить добавление 32-битной архитектуры и обновление списка пакетов 32-битной архитектуры (**здесь и далее в текущем разделе - под *root***):

```
cd /mnt/hgfs/s/chip_bb/tde
dpkg --add-architecture i386
apt-get update
```

3. Установить библиотеку *libc6* 32бит:

```
apt-get install libc6:i386
```

4. Скопировать пакет с драйверами HASP-ключа (*aksusbd_**x.x**-1_i386.deb*) на сервер KTS, если этого не было сделано ранее.
5. Выполнить установку пакета с драйверами:

```
dpkg -i aksusbd_x.x-1_i386.deb
```

6. Проверить статус установленных драйверов:

```
service aksusbd status
```

7. Выйти из root и вернуться к оболочке пользователя:

```
exit
```

⚠ vendor_code необходимо положить в папку */opt/chipblackbox/etc*.

ВНИМАНИЕ! vendor_code должен находиться не только в рабочей папке KTS, но и KGS (*/opt/kmi/etc*).

Также необходимо:

1. Проверить наличие файла kmi_cfg.xml в директории /opt/kmi/etc/
2. Если такого файла нет, то его нужно создать вручную. Содержимое файла следующее:

```
<?xml version=\"1.0\"?>
<Config>
    <KMI_vendor_code>/opt/kmi/etc/vendor_code</KMI_vendor_code>
</Config>
```

5.4. Подготовка библиотек KGS

ⓘ Версия библиотеки HASP указывается как значение переменной KMI_HASP_VERSION. Система KGS использует переменную KMI_HASP_VERSION для обработки того, какой вариант HASP должен быть установлен. Описание возможных значений KMI_HASP_VERSION приведено в отдельном документе (доступ строго ограничен).

Последовательность действий:

1. deb-пакеты с компонентами KGS, входящие в комплект поставки (см. выше), помещаются в папку на сервере KTS.
2. Установить новое значение KMI_HASP_VERSION = 'dev' + пакеты KGS:

```
sudo KMI_HASP_VERSION=dev dpkg -i kmi_fw_tde***.deb
sudo dpkg -i kmi_fw_hwrk***.deb
```

> ⚠ Экспорт переменной через *export KMI_HASP_VERSION='dev'* не работает.

3. Открыть файл /etc/ld.so.conf:

```
sudo nano /etc/ld.so.conf
```

4. Добавить в файл путь к папке с компонентами KGS:

```
/opt/kmi/lib
```

5. Запустить ldconfig:

```
sudo ldconfig
```

6. Настроить вызов исполняемых файлов из любой директории:

```
export PATH=$PATH:/opt/kmi/bin/
```

5.5. Генерация HWRK ключа сервера KTS

HWRK ключ имеет уникальную привязку к аппаратной части сервера KTS и генерируется с помощью утилиты kmi_fw_hwrk на сервере KTS.

> ⚠ **Внимание!**
>
> Для успешной генерации необходимо наличие установленного HASP-ключа (см. выше).

Для генерации требуется:

1. Перейти в папку, где в результате распаковки архива библиотек KGS (см. предыдущий раздел) был размещен исполняемый файл утилиты kmi_fw_hwrk (папка */opt/kmi/bin/*).
2. Запустить исполняемый файл:

```
art@deb:/opt/kmi$ sudo /bin/kmi_fw_hwrk
```

3. В случае, если для генерации/шифрования ключа с помощью утилиты, не хватает какого-либо пакета Debian, его следует установить. После этого следует снова запустить утилиту kmi_fw_hwrk.

4. В результате успешной работы утилиты в папке с исполняемым файлом появляются два файла: *kmi_file11.dat* и *kmi_file12.dat*.
5. Первый файл используется локально на сервере KTS, а второй файл необходимо перенести на FTP-сервер KGS.
6. На сервере KGS создается лестница ключей и конфигурация, предназначенные для данного сервера KTS (см. документацию к KGS). При этом бинарный файл с ключом BBMK (см.ниже) будет зашифрован с помощью ключа HWRK, переданного на сервер KGS.

5.6. Первоначальное наполнение базы данных

Перенос персонализационных данных (OTP-ключей), а также прочих данных с FTP-сервера KGS на сервер KTS и обратно осуществляется администратором KTS. Файлы при передаче дополнительно шифруются PGP-ключом администратора.

5.6.1. Загрузка лестницы ключей на KTS сервер

Настройка конкретного экземпляра системы KTS осуществляется путем загрузки в базу KTS лестницы ключей шифрования (персонализационные данные хранятся в БД KTS в зашифрованном виде).

1. Текстовый файл с лестницей ключей, а также бинарный файл с ключом BBMK (является частью лестницы ключей) переносятся с FTP-сервера KGS в рабочую папку БД KTS и расшифровываются PGP-ключом администратора. Доступ к формату файла предоставляется по запросу.
2. Пользователю *postgres* предоставляются права полного доступа к папке, содержащей скрипты для развертывания лестницы ключей.

> ⚠ **Примечание**
>
> Бинарный файл с ключом BBMK, а также файл с лестницей ключей должен помещаться в папку, доступную для bbx_server_go.
>
> Путь к файлу с ключом BBMK впоследствии должен быть указан в конфигурационном файле bbx_server_go.cfg (см. раздел Запуск bbx_server_go, параметр dbmk_file_name).

3. Запускается скрипт *blbx_load_keyladder.sh* (входит в комплект поставки). В качестве аргумента скрипта указывается полное имя файла (включая путь), содержащего лестницу ключей. **Параметр (имя файла, включая путь) задаётся "в кавычках"**.
   Пример:

```
cd /home/bb/bbx-scripts/scripts
sudo bash blbx_load_keyladder.sh -input_file "keyladder.dat"
```

Скрипт осуществляет загрузку лестницы ключей в целевые структуры БД.

> ⓘ Формат команды запуска скрипта с параметрами:
>
> ```
> bash blbx_load_keyladder.sh [-help] -input_file "<input file>" [-h <host>] [-p <port>] [-u
> <pg_user>] [-P <pg_user_password>] [-d <database>]
> ```
>
> Параметры запуска:

a. -help - вызов справки.
b. -input_file "<input file>" - путь к файлу с лестницей ключей. Доступ к формату файла предоставляется по запросу.
Параметр задаётся "в кавычках" (т.е. если задан путь к файлу, то в кавычках указывается весь путь, если задано только имя файла (лежит в той же папке, что и скрипт), то в кавычках указывается имя файла). Обязательный параметр. Файл выгружается из KMI.
c. -h <host> - имя хоста KTS DB. Значение по умолчанию (задается в файле скрипта) - "localhost".
d. -p <port> - номер порта KTS DB. Значение по умолчанию (задается в файле скрипта) - "5432".
e. -u <pg_user> - имя пользователя KTS DB. Значение по умолчанию (задается в файле скрипта) - "bbxadmin".
f. -P <pg_user_password> - пароль для доступа к KTS DB. Значение по умолчанию (задается в файле скрипта) - "bbxadmin".
g. -d <database> - имя базы KTS DB. Значение по умолчанию (задается в файле скрипта) - "bbx".

4. По окончании работы скрипта проверяется его лог файл (blbx_load_keyladder.log) на отсутствие ошибок. В случае успеха в логе должно быть сообщение вида *Script finished without errors*.

В дальнейшем, в процессе эксплуатации KTS, применяются дополнительные скрипты как для импорта (персонализационные данные) в БД KTS, так и для экспорта (отчет о программировании чипов).

Работа с этими скриптами описана в документе "Руководство администратора", в разделе "Перенос данных".

5.7. Установка клиента (BBX_CHIP_CLIENT)

1. Скопируйте компонент BBX_CHIP_CLIENT (библиотеки входят в состав релиза KTS), на клиентскую машину.
2. Распакуйте архив.

Для успешной работы клиента необходимо, чтобы все dll-библиотеки из архива были перемещены в папку с клиентской библиотекой на клиентской машине.

3. Сгенерируйте и положите необходимые ключи в папку с клиентской библиотекой на клиентской машине (Programming Server) (см. Создание SSL ключей и сертификатов).

5.8. Создание SSL ключей и сертификатов

ВНИМАНИЕ! Данная версия скрипта SSL_keygen_script.sh позволяется сгенерировать ключи и сертификат только **однократно**. При повторном запуске ключи и сертификат CA, а также серверный ключ и сертификат будут **перезаписаны**, соответственно, клиентский сертификат, сгенерированный ранее, станет навалидным. Будьте внимательны и не запускайте повторно скрипт!

Процедура генерации SSL-ключей и сертификатов:

1. Создание SSL ключей для сервера и клиента, а также сертификата для сервера производится с помощью запуска скрипта *SSL_keygen_script.sh* (sudo).
    a. Данный скрипт входит в комплект поставки.
    Обратите внимание! Скрипт следует запускать на той машине в защищенном окружении, которая используется для выпуска сертификатов. Иными словами, это не должен быть сервер KTS.
    b. Пример команды запуска:

```
sudo SSL_keygen_script.sh
```

2. В процессе выполнения скрипта:
    a. Создается 3 приватных ключа (*cakey.pem* - ключ центра сертификации (CA) для выпуска самоподписанных сертификатов, приватный ключ для сервера *server-key.pem* и приватный ключ для клиента *client-key.pem*) и 3 сертификата (*cacert.pem* - корневой сертификат CA, самоподписанные сертификаты *server-cert.pem* и *client-cert.pem*), а также дополнительные файлы, необходимые для выпуска сертификатов.
    b. При создании *cakey.pem* можно использовать произвольный пароль. Далее, при создании сертификатов выполняется их подпись данным ключом, для чего требуется повторить ввод этого пароля несколько раз (это указывается в информации в консоли).
    c. При генерации ключей *server-key.pem* и *client-key.pem* запрашивается ввод пароля (pass phrase). Необходимо ввести какой-либо пароль, например, *password*. Стоит отметить, что в финальной версии файлов *server-key.pem* и *client-key.pem*, которые сохраняются по завершению работы скрипта, пароль из файлов pem удаляется (см. также п.7 ниже).
    d. Из полей сертификатов нужно заполнять:
        i. для cacert: CN (Common Name) = <some_name>, где <some_name> - произвольное имя центра сертификации.
        ii. для server-cert: CN = <server_value>, где server_value - либо IP-адрес, либо доменное имя сервера (если назначено и используется при работе с ним).
        iii. для client-cert: CN = <client_value>, где client_value - либо IP-адрес, либо доменное имя клиента (если назначено и используется при работе с ним).
3. В папке с запускаемым скриптом будет создана папка *openssl*. Данная папка, в свою очередь, содержит файлы *cacert.pem*, *client-key.pem*, *client-cert.pem*, *openssl.cnf*, *openssl.cnf.bak*, *server-key.pem*, *server-cert.pem* , папки *private* и *newcerts*.

> ⚠ После запуска скрипта SSL_keygen_script.sh необходимо оставить 6 файлов: *cacert.pem*, *client-key.pem*, *client-cert.pem, server-key.pem, server-cert.pem, cakey.pem* (лежит в папке *private*).

4. Файл *cacert.pem* следует поместить в папку, в которой располагается исполняемый файл bbx_server_go. Возможно также указание полного пути для данных файлов в файле конфигурации bbx_server_go (см. далее).
ВНИМАНИЕ! Файл *cakey.pem* (данный ключ содержится в папке *private*) надо держать в секрете на той машине, которая используется для выпуска сертификатов. Ключ используется только для (пере)выпуска серверного или клиентского сертификатов, либо самоподписанного корневого сертификата центра сертификации. Перевыпуск нужен при окончании срока годности сертификата, выпуске нового - при появлении второго клиента.
5. В конфигурационном файле bbx_server_go следует задать путь и имена для файлов с ключами и сертификатами в параметрах SYSTEM_HTTPS_CA_CERTIFICATE, SYSTEM_HTTPS_SERVER_CERTIFICATE, SYSTEM_HTTPS_SERVER_KEY.
6. Файлы клиента должны иметь такие имена, которые сохранены в коде. Таким образом, следует переименовать следующие файлы, полученные в результате запуска скрипта генерации:

   a. *client-key.pem* должен быть переименован в *key.pem*
   b. *client-cert.pem* должен быть переименован в *client.pem*
7. **ВАЖНО:** при генерации *key.pem* ключ контента более **не** зашифрован паролем (клиентский ключ без пароля).
8. Поместить файлы *cacert.pem*, *key.pem*, *client.pem* в папку с клиентской библиотекой (BBX_CHIP_CLIENT) на клиентской машине (Programming Server).

6. Запуск bbx_server_go

> ⚠ **Внимание!**
>
> Запуск bbx_server_go невозможен без установленного HASP-ключа (см. выше).

Для запуска каждого из bbx_server_go следует указать все необходимые параметры в файле *bbx_server_go.cfg*.

Перейдите в папку с конфигурационным файлом bbx_server_go:

```
cd /opt/chipblackbox/etc
```

Изначально в системе присутствует шаблон файла конфигурации с расширением ***.cfg.dft***, содержащий дефолтные значения параметров.

Создайте копию этого файла с расширением *.cfg*, с которой в дальнейшем будет производиться работа:

```
sudo cp bbx_server_go.cfg.dft bbx_server_go.cfg
```

Таким образом, шаблон с начальными значениями параметров остается неизменным.

Откройте файл для редактирования:

```
sudo nano bbx_server_go.cfg
```

Задайте необходимое значение параметров.

Запустите bbx_server_go:

```
sudo service bbx_server_go start
```