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.

> **ВАЖНО!** Для KTS необходимо использовать специальную версию внешней HASP-библиотеки (KMI_HASP_VERSION = 'dev'). Данная информация используется при установке библиотек KGS (см. Подготовка библиотек KGS).

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
- libboost версии 1.74.0 и связанные пакеты:

  - [ ] libboost-system1.74.0
  - [ ] libboost-system1.74-dev
  - [ ] libc6
  - [ ] libboost-thread1.74.0
  - [ ] libboost-date-time1.74.0
  - [ ] libboost-regex1.74.0
  - [ ] libboost-filesystem1.74.0
  - [ ] libboost-program-options1.74.0

Пакет libboost 1.74.0 и связанные с ним пакеты допустимо устанавливать после установки подсистем базы данных.

4.2. Настройка сервера

Порядок действий:

1. Установить PostgreSQL.
2. Отредактировать конфигурационный файл PostgreSQL pg_hba.conf в соответствии с приведенным рисунком:

```
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres    127.0.0.1/32                md5
#host    replication     postgres    ::1/128                     md5
```

3. Перезапустить PostgreSQL:

```
sudo service postgresql restart
```

4. Установить ODBC:

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

5. При этом редактировать файл *odbcinst.ini* в соответствии с приведенным рисунком:

```
[PostgreSQL ANSI]
Description=PostgreSQL ODBC driver (ANSI version)
Driver=psqlodbca.so
Setup=libodbcpsqlS.so
Debug=0
CommLog=1
UsageCount=1

[PostgreSQL Unicode]
Description=PostgreSQL ODBC driver (Unicode version)
Driver=psqlodbcw.so
Setup=libodbcpsqlS.so
Debug=0
CommLog=1
UsageCount=1
```

6. Редактировать файл *odbc.ini* в соответствии с рисунком:

```
[bb]
Description             = PostgreSQL ANSI
Driver          = PostgreSQL ANSI
Trace           = No
TraceFile               =
Database                = bbx
Servername              = localhost
Username                = bbxadmin
Password                = bbxpass
Port            = 5432
Protocol                = 6.4
ReadOnly                = No
RowVersioning           = No
ShowSystemTables                = No
ShowOidColumn           = No
FakeOidIndex            = No
ConnSettings            =
```

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
```

5.2. 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. Проверьте, что вывод имеет вид:

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

5.2.2. 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 дополнительных пакетов Debian

> Приведенные ниже процедуры актуальны для Debian 11.

Порядок действий:

1. Остановить *Postgres*.
2. (При необходимости) подтвердить перезапуск сервисов для установки и конфигурирования пакета libc (запрашивается автоматически, **необходимо выбрать <Yes>**).
3. Установить связанные пакеты *libboost 1.74*:

libboost-system1.74.0:

```
sudo apt-get install libboost-system1.74.0
```

libboost-system1.74-dev:

```
sudo apt-get install libboost-system1.74-dev
```

libboost-thread1.74.0:

```
sudo apt-get install libboost-thread1.74.0
```

libboost-date-time1.74.0:

```
sudo apt-get install libboost-date-time1.74.0
```

libboost-regex1.74.0:

```
sudo apt-get install libboost-regex1.74.0
```

libboost-filesystem1.74.0:

```
sudo apt-get install libboost-filesystem1.74.0
```

libboost-program-options1.74.0:

```
sudo apt-get install libboost-program-options1.74.0
```

4. Запустить *Postgres*.

5.4. 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:

```
sudo 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:

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

6. Проверить статус установленных драйверов:

```
sudo service aksusbd status
```

⚠ 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*).

5.5. 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).

⚠ В KTS по умолчанию используется значение KMI_HASP_VERSION = 'bb'.

Тем не менее, **для KTS необходимо использовать специальную версию внешней HASP-библиотеки (KMI_HASP_VERSION = 'dev')**, которая позволяет использовать удаленный терминал. Поэтому перед установкой библиотек KGS сначала требуется установить новое значение KMI_HASP_VERSION = 'dev' и лишь затем устанавливать kmi_fw_***.

Если реальная версия библиотеки HASP ('dev') и значение переменной KMI_HASP_VERSION ('bb') не совпадают, то после установки компонентов KGS сервер KTS не может быть запущен из-за ошибки инициализации.

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/
```

5.6. 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. Подобная ситуация (с пакетом lshw) показана на рисунке:

```
[sudo] password for malyshev1:
Generating RSA keys...
RSA keys success generated
Start calculating binding key...
sh: 1: lshw: not found
[BindingKeyException]: Serial numbers not enough
malyshev1@VM-Ubuntu1204Malyshev1:~/lib$ sudo apt-get install lshw
Чтение списков пакетов… Готово
Построение дерева зависимостей
Чтение информации о состоянии… Готово
Следующие пакеты устанавливались автоматически и больше не требуются:
  libgssglue1 libmpc2
Для их удаления используйте «apt-get autoremove».
НОВЫЕ пакеты, которые будут установлены:
  lshw
обновлено 0, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 126
9 пакетов не обновлено.
Необходимо скачать 234 kB архивов.
После данной операции, объём занятого дискового пространства возрастёт на 814 kB
.
Получено:1 http://mirror.yandex.ru/debian/ jessie/main lshw amd64 02.17-1.1 [234
 kB]
Получено 234 kB за 0c (3 204 kB/c)
Выбор ранее не выбранного пакета lshw.
(Чтение базы данных … на данный момент установлено 153244 файла и каталога.)
Распаковывается пакет lshw (из файла …/lshw_02.17-1.1_amd64.deb) …
Обрабатываются триггеры для man-db …
Настраивается пакет lshw (02.17-1.1) …
```

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 содержит приватную часть ключа HWRK и используется локально на сервере KTS, а второй файл, содержащий публичную часть ключа HWRK, 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.

> При подготовке файла с конфигурацией для KTS в системе KGS должен быть выбран *format version = 2*.

5.7. 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.

5.7.1. Загрузка лестницы ключей и конфигурации на KTS сервер

Настройка конкретного экземпляра системы KTS осуществляется путем загрузки в базу KTS конфигурации (набора настроек для персонализации определенной партии чипов) и лестницы ключей шифрования (персонализационные данные хранятся в БД KTS в зашифрованном виде).

1. Текстовый файл с лестницей ключей и файл конфигурации, 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 конфигурации и/или лестницы ключей.

   > **Примечание**
   >
   > Бинарный файл с ключом BBMK, а также файлы с лестницей ключей или конфигурацией должны помещаться в папку, доступную для bbx_server_go.

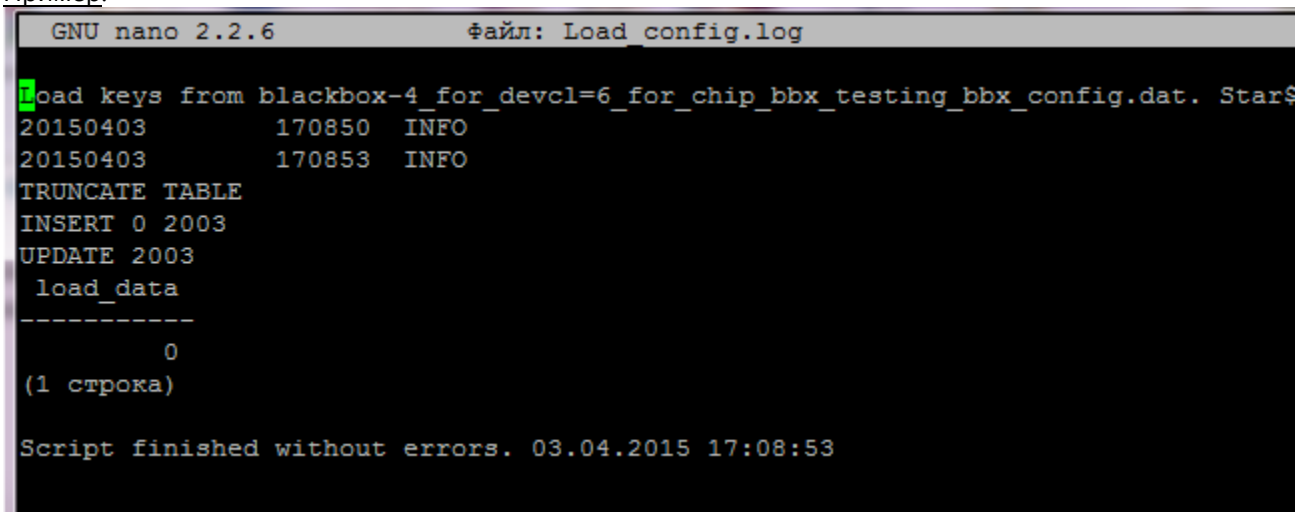
3. Запускается скрипт *blbx_load_config.sh* (входит в комплект поставки). В качестве аргумента скрипта указывается полное имя файла (включая путь), содержащего конфигурацию или лестницу ключей. **Параметр (имя файла, включая путь) задаётся "в кавычках"**.
   Пример:

   ```
   cd /home/bb/bbx-scripts/scripts
   sudo bash blbx_load_config.sh -input_file "blackbox-4_for_devc1\=6_for_chip_bbx_testing_bbx_config.dat"
   sudo bash blbx_load_config.sh -input_file "keyladder.dat"
   ```

   Скрипт осуществляет загрузку лестницы ключей или конфигурации (наличие данных определяется автоматически) в целевые структуры БД.

   > Парамеры запуска скриптов подробно описаны в документе "Руководство администратора".

4. Лог загрузки лестницы ключей/конфигурации содержится в файле blbx_load_config.log.
   Пример:

5.7.2. Загрузка OTP-ключей на KTS сервер

1. Файл с данными переносится с FTP-сервера KGS в рабочую папку БД KTS и расшифровывается PGP-ключом администратора.

2. Пользователю *postgres* предоставляются права полного доступа к папке, содержащей скрипты для развертывания базы данных ключей.
3. Запускается скрипт *blbx_load_keys.sh* (входит в комплект поставки). В качестве аргументов скрипта указываются полное имя файла (включая путь), содержащего OTP-ключи, + Part Number name + полное имя файла (включая путь), содержащего хеш лестницы ключей. **Параметры - имена файлов, включая путь - задаются "в кавычках"**.
   Пример:

```
cd /home/bb/bbx-scripts/scripts
bash blbx_load_keys.sh -key_file "/home/user1/data_kmi/dvcl_pn1_prod
/Neotion_BBX_0000007E_16_30_exported_keys" -part_num dvcl_pn1_prod -tde_hash "/home/user1/data_kmi
/dvcl_pn1_test/7_key_ladder_hash_16_30"
```

   Скрипт осуществляет загрузку OTP-ключей в целевые структуры БД.

   ⓘ Парамеры запуска скриптов подробно описаны в документе "Руководство администратора".

4. Лог загрузки OTP-ключей содержится в файле blbx_load_keys.log.
   Пример:

```
  GNU nano 2.2.6              Файл: Load_keys.log

Load keys from exported.keys_1-10000. Start 03.04.2015 17:25:56

Load keys

Load 10000 recs. 03.04.2015 17:26:14
TRUNCATE TABLE
INSERT 0 9999
INSERT 0 2
UPDATE 10001
Run load_data(). 03.04.2015 17:26:16
 load_data
-----------
         0
(1 строка)

Script finished without errors. 03.04.2015 17:26:24
```

5.8. Установка клиента (BBX_CHIP_CLIENT)

1. Скопируйте компонент BBX_CHIP_CLIENT (библиотеки входят в состав релиза KTS), на клиентскую машину.
2. Распакуйте архив.
3. Сгенерируйте и положите необходимые ключи в папку с клиентской библиотекой на клиентской машине (Programming Server) (см. Создание SSL ключей и сертификатов).

5.9. Создание SSL ключей и сертификатов

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). Данный скрипт входит в комплект поставки.
2. В процессе выполнения скрипта у пользователя запрашивается кодовая фраза (pass phrase). В качестве ответа следует вводить слово *password* (малые буквы, латиница). Данное значение сохранено в коде. Также нужно заполнить значение для Common name.
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*).

   Прежде чем запускать или обновлять bbx_server_go, нужно в конфиг файле актуализировать пути до файлов с сертификатами (т.е. указать верные пути к файлам). Помещать файлы в конкретные папки (см. шаги ниже) - в случае сервера не обязательно.

4. Полученные файлы *cakey.pem* (данный ключ содержится в папке *private*) и *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).
5. В случае, если при создании файлов с ключами им были заданы другие имена, в конфигурационном файле bbx_server_go следует заменить имена по умолчанию на заданные.
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*.

Рабочая директория KTS - */opt/chipblackbox*. При этом в рабочей директории (*/opt/chipblackbox*) должны находиться ключи (названия по умолчанию - kmifile11.dat, kmifile12.dat, bbmk.dat) и сертификаты (.pem). Конфиг находится в подпапке *etc* (*/opt/chipblackbox/etc*). Исполняемый файл bbx_server_go находится в подпапке *bin* (*/opt/chipblackbox/bin*). vendor_code должен находиться в папке */opt/chipblackbox/etc*.

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

При запуске с vendor_code нужно:

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

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

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. Содержание файла конфигурации приведено в таблице:

Параметр	Описание
LOGGER_LEVEL	Степень логирования событий. Возможные значения: • trace, • debug, • info (значение по умолчанию), • warning, • error, • fatal. Для тестирования рекомендуется "debug" или "trace". Подробное описание уровней логирования приведено в разделе ниже.
LOGGER_OUTPUT_FILE	Параметр для вывода логов в файл. В параметре указывается полный путь до файла, куда пишутся логи. Если параметр пустой или отсутствует, то логи пишутся в stdout, если параметр задан, то логи пишутся в файл.
SYSTEM_TRACER_ENABLED	Настройки взаимодействия с системой отслеживания запросов. Флаг включения трассировки. Возможные значения: • false - трассировка выключена. • true - трассировка включена.
SYSTEM_TRACER_AGENT_HOST_PORT	Адрес и номер порта agent, т.е. сервера, на который идет отправка параметров трассировки.
SYSTEM_HTTP_ADDRESS	Адрес для запуска сервера в контейнере. Формат записи: <*IP-адрес*>:<*номер порта*>
SYSTEM_HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных).
SYSTEM_HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных).

SYSTEM_HTTP_IS_USED	Флаг включения взаимодействия по http. Возможные значения: • false - Взаимодействие выключено. • true - Сервис взаимодействует по http.
SYSTEM_HTTPS_ENABLED	Флаг включения взаимодействия по https, т.е. используется или нет протокол **защищенной** сети (= включение работы с SSL сертификатами). Возможные значения: • true - секция включена; • false - секция выключена. Если параметр отсутствует, то считается, что = false.
SYSTEM_HTTPS_VERIFY_MODE	Режим работы с сертификатами. Возможные значения: • 0 - NoClientCert ClientAuthType = iota NoClientCert означает, что сертификат клиента не будет запрашиваться во время взаимодействия. Если какие-либо сертификаты будут отправлены, то они не будут подвергаться проверке. • 1 - RequestClientCert RequestClientCert означает, что сертификат клиента должен быть запрошен во время взаимодействия, однако не требует, чтобы клиент отправлял какие-либо сертификаты. • 2 - RequireAnyClientCert RequireAnyClientCert означает, что сертификат клиента должен быть запрошен во время взаимодействия, и что по крайней мере один сертификат должен быть отправлен клиентом, но этот сертификат не обязательно должен быть действительным (валидным). • 3 - VerifyClientCertIfGiven VerifyClientCertIfGiven означает, что сертификат клиента должен быть запрошен во время взаимодействия, но не требует, чтобы клиент отправлял сертификат. Если клиент отправляет сертификат, он должен быть действительным. • 4 - RequireAndVerifyClientCert RequireAndVerifyClientCert означает, что сертификат клиента должен быть запрошен во время взаимодействия, и что клиент должен отправить по крайней мере один действительный сертификат. Если параметр отсутствует, то считается, что = 0.

SYSTEM_HTTPS_CA_CERTIFICATE	Путь к файлу с CA сертификатом. Обязательное поле.
SYSTEM_HTTPS_SERVER_CERTIFICATE	Путь к файлу с server certificate. Обязательное поле.
SYSTEM_HTTPS_SERVER_KEY	Путь к файлу с server key. Обязательное поле.
SYSTEM_DB_MASTER_HOST	IP-адрес мастер реплики (инстанс кластера KTS DB)
SYSTEM_DB_MASTER_PORT	Номер порта мастер реплики (инстанс кластера KTS DB)
SYSTEM_DB_MASTER_USER	Логин пользователя, под которым осуществляется подключение к мастер реплике (инстансу кластера KTS DB)
SYSTEM_DB_MASTER_PASSWORD	Пароль для подключения к мастер реплике (инстансу кластера KTS DB)
SYSTEM_DB_MASTER_DB_NAME	Имя мастер реплики (инстанс кластера KTS DB) в PostgreSQL
SYSTEM_DB_MASTER_TIMEOUT	Время ожидания ответа (в сек.) от мастер реплики (инстанс кластера KTS DB)
SYSTEM_DB_MASTER_MAX_CONNS	Максимальное количество соединений, создаваемых сервером с мастер репликой (инстансом кластера KTS DB)
SYSTEM_DB_ASYNC_REPLICA_HOST	IP-адрес асинхронной реплики (инстанс кластера KTS DB)
SYSTEM_DB_ASYNC_REPLICA_PORT	Номер порта асинхронной реплики (инстанс кластера KTS DB)
SYSTEM_DB_ASYNC_REPLICA_USER	Логин пользователя, под которым осуществляется подключение к асинхронной реплике (инстансу кластера KTS DB)
SYSTEM_DB_ASYNC_REPLICA_PASSWORD	Пароль для подключения к асинхронной реплике (инстансу кластера KTS DB)
SYSTEM_DB_ASYNC_REPLICA_DB_NAME	Имя асинхронной реплики (инстанс кластера KTS DB) в PostgreSQL.
SYSTEM_DB_ASYNC_REPLICA_TIMEOUT	Время ожидания ответа (в сек.) от асинхронной реплики (инстанс кластера KTS DB)
SYSTEM_DB_ASYNC_REPLICA_MAX_CONNS	Максимальное количество соединений, создаваемых сервером с асинхронной репликой (инстансом кластера KTS DB)
SYSTEM_PROMETHEUS_HTTP_ENABLED	Флаг включения прослушки Prometheus. Возможные значения: • true - секция включена; • false - секция выключена.

SYSTEM_PROMETHEUS_HTTP_ADDRESS	Адрес для запуска сервера в контейнере. Формат записи: *<IP-адрес>:<номер порта>*
SYSTEM_PROMETHEUS_HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных).
SYSTEM_PROMETHEUS_HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных).
SYSTEM_PPROF_ENABLED	Флаг включения взаимодействия с pprof. Возможные значения: • false - pprof выключен. Значение по умолчанию. • true - pprof включен. **Примечание**. Подробное описание приведено в разделе ниже.
SYSTEM_PPROF_ACCESS_KEY	Значение ключа доступа, отправляемого в запросе к pprof. Запрос используется для профилирования сервисов (исследования CPU).
SYSTEM_TDE_DSN	DSN базы данных (KTS DB), которую использует tde (механизм шифрования данных по лестнице ключей)
SYSTEM_TDE_HWRK_FILE_NAME	Полный путь и название private HWRK-ключа, сгенерированного на сервере KTS **Примечание**. Ключ является частью лестницы ключей, применяемой при шифровании данных в KTS DB
SYSTEM_TDE_DBMK_FILE_NAME	Полный путь и название BBMK-ключа, шифрованного public HWRK-ключом. Файл с ключом экспортируется из системы KGS **Примечание**. Ключ является частью лестницы ключей, применяемой при шифровании данных в KTS DB
SYSTEM_TDE_USE_MLOCK	Флаг. Если установлен в true, то блокируется возможность сброса на диск данных, хранящихся в оперативной памяти в модуле по работе с tde

Пример файла:

bbx_server_go.cfg

```
{
  "logger": {
      "level": "debug",
      "output_file": ""
  },
  "system" : {
    "tracer" : {
        "enabled" : false,
        "agent_host_port": "agent:6831"
    },
    "http": {
      "address": "0.0.0.0:8080",
      "write_timeout": 45,
      "read_timeout": 15,
      "is_used" : true
    },
    "https": {
      "enabled": true,
      "verify_mode": 2,
      "ca_certificate": "cert/cacert.pem",
      "server_certificate": "cert/server-cert.pem",
      "server_key": "cert/server-key.pem"
    },
    "db" : {
        "master": {
           "host": "192.168.14.47",
           "port": 5432,
           "user": "bbxadmin",
           "password": "bbxadmin",
           "db_name": "bbx_server",
           "timeout" : 50,
           "max_conns" : 10
        },
        "async_replica": {
           "host": "192.168.14.47",
           "port": 5432,
           "user": "bbxadmin",
           "password": "bbxadmin",
           "db_name": "bbx_server",
           "timeout" : 50,
           "max_conns" : 10
         }
    },
    "prometheus_http" :
    {
       "address" : "0.0.0.0:9102",
       "write_timeout" : 5,
       "read_timeout" : 5
    },
    "pprof": {
        "enable" : true,
        "access_key" : "fc64a74a-51ca-4cb9-afea-5d5c633bc4d0"
    },
    "tde": {
        "dsn": "BBX_STAND",
        "hwrk_file_name": "/opt/chipblackbox/kmi_file11.dat",
        "dbmk_file_name": "/opt/chipblackbox/bbmk.dat",
        "use_mlock": true
    }
  }
}
```

Запустите bbx_server_go:

```
sudo service bbx_server_go start
```

6.1. Взаимодействие с pprof

Для анализа узких мест в реализации серверов и точечной настройки Golang серверов используется сторонний компонент pprof.

Для работы с ним в конфигурационном файле каждого Golang сервера, а также в default.yaml/production.yaml имеется секция pprof.

По умолчанию pprof выключен. Чтобы включить взаимодействие с pprof, необходимо:

- в production.yaml выставить pprof.enable равным true и заменить access_key на собственное значение:

```
...
pprof:
  enabled: true
  access_key: fc64a74a-51ca-4cb9-afea-5d5c633bc4d0
...
```

> ⚠ Значение pprof в production.yaml имеет более высокий приоритет, чем в конфигурационных файлах компонентов, поэтому менять значения в конфигурационном файле каждого отдельного сервиса не требуется.

- развернуть либо перезапустить все службы KTS.

Если pprof включен, то он позволяет отправлять запрос вида:

Пример запроса

```
curl --request GET '192.168.11.86:30170/debug/pprof/profile?seconds=15' --header 'X-Api-Key: fc64a74a-51ca-4cb9-afea-5d5c633bc4d0' --output fasentry.bin
```

Наличие параметра *--header 'X-Api-Key: значение_access_key_из_конфига'* в запросе обязательно.

Запрос используется для профилирования сервисов (исследования CPU).