

УТВЕРЖДЕН  
643.72410666.00067-07 98 01-ЛУ

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ  
БАЗАМИ ДАННЫХ «JAТОВА»

Руководство по настройке. Часть 18.  
Соккрытие информации в файлах данных СУБД.  
Компонент «Jatoba crypto access storage»

643.72410666.00067-07 98 01-18

Листов 24

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

## АННОТАЦИЯ

В документе приведены сведения, необходимые для установки и эксплуатации компонента предназначенного для сокрытия информации в файлах данных СУБД «Jatoba crypto access storage» (далее по тексту – «компонент» или JCS).

Настоящее руководство предназначено для администраторов СУБД.



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра 4.x, для других версий все шаги выполняются аналогично, разница состоит в именах директорий.

Например, СУБД «Jatoba» версии 5.x по умолчанию устанавливается в директорию ОС Linux – «/usr/jatoba-5/bin».

Используемая версия компонента — 2.0.

Степени важности примечаний, применяемые в документе:



**Важная информация** – указания, требующие особого внимания



**Дополнительная информация** – указания, позволяющие упростить работу с изделием



**Важная информация**

Для сертифицированной версии СУБД «Jatoba» поддерживается работа только на ОС, указанных в формуляре на поставку!

## СОДЕРЖАНИЕ

1. Назначение компонента.....	4
1.1. Условия применения.....	4
2. Установка компонента.....	5
2.1. Установка компонента в ОС GNU/Linux.....	5
2.2. Установка расширения компонента .....	6
3. Функциональные возможности .....	8
3.1. Режим администратора .....	9
3.1.1. Создание ключа и вектора для преобразования данных по умолчанию без параметров.....	9
3.1.2. Создание ключа и вектора для преобразования данных текущей БД с выбором алгоритма AES. 11	
3.1.3. Создание ключа и вектора для преобразования данных с выбором БД и алгоритма AES .....	11
3.1.4. Создание ключа и вектора для преобразования данных с выбором БД и алгоритма TwoFish .....	14
3.1.5. Изменение пути сохранения файла ключей .....	15
3.1.6. Создание преобразованной таблицы .....	15
3.1.7. Чтение данных из преобразованной таблицы.....	16
3.2. Режим пользователя .....	16
3.2.1. Переключение в режим пользователя.....	16
3.2.2. Ввод ключа и вектора преобразования данных.....	17
3.2.3. Создание преобразованной таблицы в режиме пользователя.....	17
3.2.4. Чтение данных из преобразованной таблицы созданной в режиме пользователя .....	17
3.3. Просмотр директории хранения ключей .....	19
3.4. Просмотр ключа .....	19
3.5. Переключение в режим администратора.....	19
3.6. Получение номера текущей версии компонента.....	19
4. Удаление компонента .....	20
4.1. Удаление компонента при отсутствии зависимых от него объектов .....	20
4.2. Удаление компонента при наличии зависимых от него объектов.....	20
5. Сообщения об ошибках .....	21
5.1. Повторное создание ключа и вектора для БД .....	21
5.2. Создание преобразованной таблицы без предварительной генерации ключа и вектора преобразования данных.....	21
5.3. Чтение данных из преобразованной таблицы с отсутствующим файлом ключей .....	21
5.4. Чтение из преобразованной таблицы с поврежденным/измененным файлом «jcs.key».....	21
5.5. Создание ключа и вектора для шифрования с выбором БД без предустановленного расширения JCS .....	22
Перечень сокращений.....	23

## 1. НАЗНАЧЕНИЕ КОМПОНЕНТА

Компонент JCS предназначен для выполнения сокрытия данных в только в таблицах БД и предотвращения возможности ознакомления при их утрате.

Компонент JCS не выполняет функции безопасности, установленные методическим документом «Меры защиты информации в государственных информационных системах» (утв. ФСТЭК России 11.02.2014).

### 1.1. Условия применения

Компонент JCS может использоваться совместно с СУБД «Jatoba» версий 4.x и выше, под управлением ОС GNU/Linux.

В текущей реализации компонента не поддерживается:

- управление через компонент пользовательского веб-интерфейса для администраторов «Jatoba data safe»;
- преобразование данных секционированных таблиц;
- преобразование данных индексов и материализованных представлений СУБД.

Компонент JCS не рекомендуется использовать одновременно с компонентом сжатия данных на уровне страниц «ja\_Compression», поскольку в этом случае сжатие данных неэффективно.

Ограничений по совместимости с другими компонентами нет.



В случае утраты ключей преобразования данных восстановить данные таблиц будет невозможно.

## 2. УСТАНОВКА КОМПОНЕНТА

Компонент функционирует под управлением ОС семейства GNU Linux. Установка компонента должна производиться от имени пользователя, обладающего административными привилегиями в системе.

### 2.1. Установка компонента в ОС GNU/Linux

Компонент устанавливается в составе СУБД «Jatoba». Его возможно установить при первичной установке, либо доустановить.

Установку компонента возможно провести двумя способами:

- 1) установка из локального репозитория (CDROM) – производится из файлов, записанных на компакт-диск или скопированных с него;
- 2) установка непосредственно из deb/rpm-файлов – производится опционально, по усмотрению пользователя.

Компонент выполнен в виде отдельного deb или rpm-пакета. Установка компонента осуществляется средствами пакетного менеджера ОС.

Для разных типов пакетных менеджеров команда установки различается:

- для систем на основе пакетного менеджера APT (к таким системам относятся все ОС семейства Debian, использующие deb-пакеты):

```
apt-get install jatoba18-jcs
```

- для систем на основе пакетных менеджеров YUM/DNF (к таким системам относятся все ОС семейства Red Hat и вышедшие из нее, использующие rpm-пакеты):

```
yum install jatoba18-jcs
```

При установке компонента на ОС ALT Linux и openSUSE необходимо учитывать следующие особенности:

- ALT Linux использует пакетный менеджер APT, но распространяется в виде rpm-пакетов, команда установки выглядит аналогично Debian:

```
apt-get install jatoba18-jcs
```

Установка компонента в составе других версий СУБД «Jatoba» осуществляется аналогично в соответствии с номером версии СУБД, в составе которой он распространяется.

В наименовании компонента указывается:

- наименование СУБД;
- версия СУБД;
- знак разделителя;
- наименование компонента.

Например, jatoba4-jcs и т.п.

Для получения детальной информации по пакетному менеджеру рекомендуется обратиться к документации по ОС.

## 2.2. Установка расширения компонента

Предварительная настройка конфигурационного файла «postgresql.conf» для установки расширения компонента не требуется.

Расширение устанавливается при помощи SQL-команды:

```
CREATE EXTENSION jcs;
```

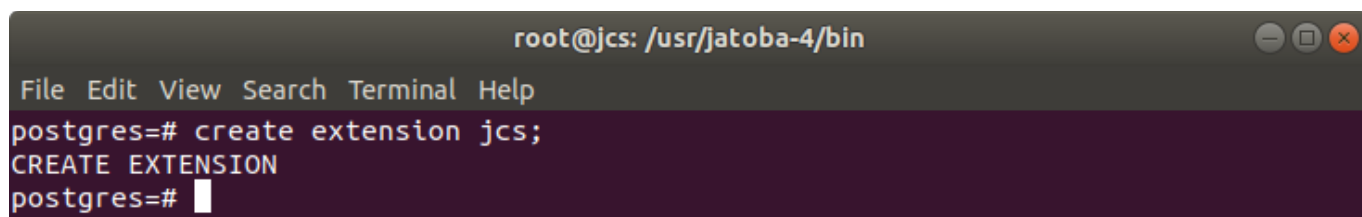


Рисунок 2.1 – Команда установки расширения в CLI под ОС Ubuntu

В результате выполнения SQL-команды будут созданы схема данных «jcs» и одноименное расширение (EXTENSION).

Проверка установки расширения и схемы данных, выполняется SQL-командами:

```
\dx  
\dn
```

```
root@jcs: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# create extension jcs;
CREATE EXTENSION
postgres=# \dx
               List of installed extensions
  Name  | Version | Schema  | Description
-----+-----+-----+-----
 jcs    | 2.0     | public  | encrypted data in storage
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)

postgres=# \dn
      List of schemas
  Name  | Owner
-----+-----
 jcs    | postgres
 public | postgres
(2 rows)

postgres=#
```

Рисунок 2.2 – Результат установки расширения и схемы данных в ОС GNU/Linux

### 3. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ

Компонент использует два алгоритма преобразования данных:

- AES (используется по умолчанию);
- TwoFish.

Функциональные возможности компонента реализуются в двух режимах функционирования:

- режим администратора (см. раздел 3.1);
- режим пользователя (см. раздел 3.2).

В режиме администратора возможно:

- создавать ключ и вектор преобразования данных;
- создавать преобразованные объекты, которые будут доступны другим пользователям СУБД и т.д.



Изменение ключа и вектора преобразования данных, алгоритма преобразования данных может привести к потере доступности данных.

Изменение параметров преобразования данных для объекта не поддерживается.

В режиме пользователя возможно создавать преобразованные объекты, доступ к которым возможен только при:

- переходе в пользовательский режим;
- указании ключа и вектора преобразования данных.

При этом для пользователя не обязательно получать доступ на использование расширения «jcs» и не требуется, чтобы были задан вектор и создан ключ преобразования данных для БД.



В качестве примера, создадим тестовую БД «test\_db» и проверим список БД:

```
CREATE DATABASE test_db;  
\1
```

Установим расширение «JSC».

```
CREATE EXTENSION jcs;
```

### 3.1. Режим администратора

При создании ключа, вектора преобразования данных и алгоритма для БД записываются в файл jcs.key. При изменении параметров таких как смена вектора преобразования данных и алгоритма преобразования данных, необходимо удалить соответствующую строку данных в файле jcs.key вручную. Автоматический режим удаления не поддерживается.



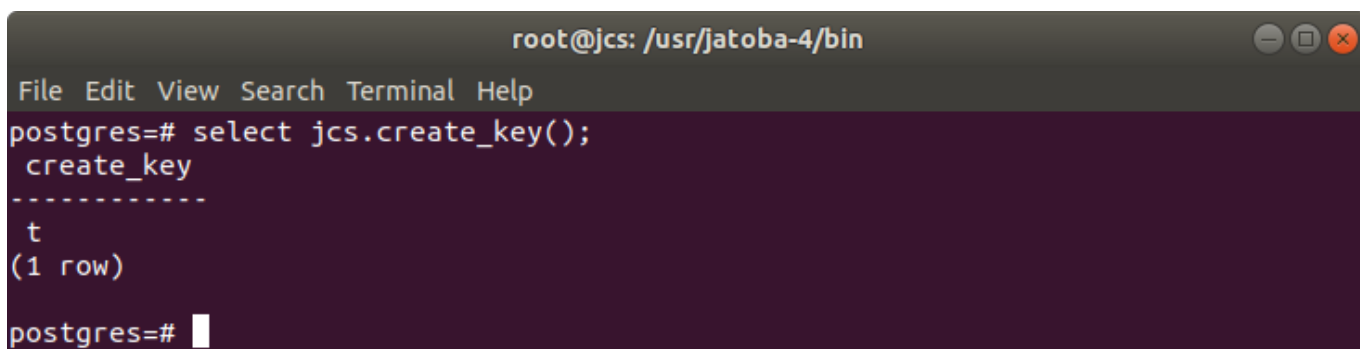
Требуется учитывать, что при изменении файла вручную, изменения в СУБД будут применяться в течении 1 минуты.

#### 3.1.1. Создание ключа и вектора для преобразования данных по умолчанию без параметров

В режиме администратора существует функциональная возможность создания ключа и вектора шифрования по умолчанию. При этом дополнительные параметры не указываются, а назначаются по умолчанию.

Для генерации ключа и вектора по умолчанию (без параметров) для текущей БД необходимо от имени и с правами пользователя, имеющего административные привилегии, выполнить SQL-команду:

```
SELECT jcs.create_key();
```



```
root@jcs: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# select jcs.create_key();
 create_key
-----
 t
(1 row)

postgres=#
```

Рисунок 3.1 – Команда создания jcs.key в ОС GNU/Linux

В результате будет сгенерирован ключ и вектор шифрования, записанные в файл jcs.key. Файл будет сохранен по пути:

```
/var/lib/jatoba/<версия>/jcs/jcs.key
```

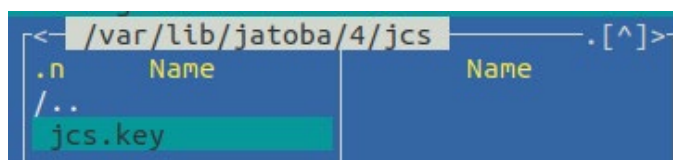


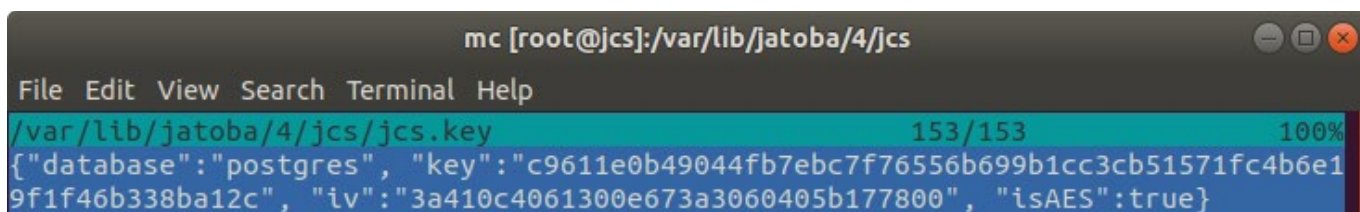
Рисунок 3.2 – Путь расположения файла jcs.key в ОС GNU/Linux

Файл jcs.key содержит следующие параметры:

- database – БД;
- key – ключ преобразования данных;
- iv – вектор преобразования данных;
- isAES – метод преобразования данных.

По умолчанию, устанавливается метод преобразования данных AES (isAES) и обозначается значением «true».

Пример содержания файла приведен на рисунке 3.3.



```
mc [root@jcs]:/var/lib/jatoba/4/jcs
File Edit View Search Terminal Help
/var/lib/jatoba/4/jcs/jcs.key 153/153 100%
{"database":"postgres", "key":"c9611e0b49044fb7ebc7f76556b699b1cc3cb51571fc4b6e1
9f1f46b338ba12c", "iv":"3a410c4061300e673a3060405b177800", "isAES":true}
```

Рисунок 3.3 – Содержание файла ключей в ОС GNU/Linux

### 3.1.2. Создание ключа и вектора для преобразования данных текущей БД с выбором алгоритма AES

Аналогично вышеописанной SQL-команде можно выполнить SQL-команду для создания ключа и вектора преобразования данных текущей БД с указанием алгоритма преобразования данных.

Для этого используется SQL-команда с параметром «true»:

```
SELECT jcs.create_key(true);
```

Файл «jcs.key» будет создан в каталоге по умолчанию.

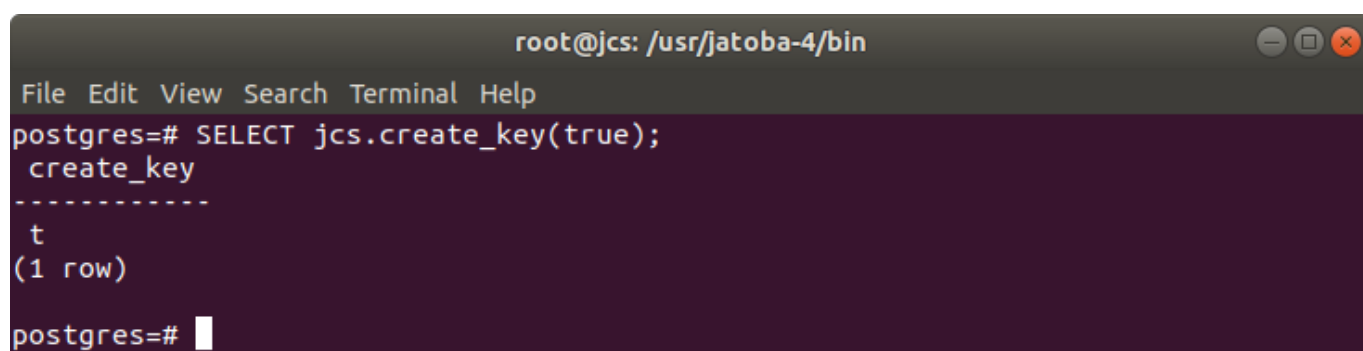


Рисунок 3.4 – Команда создания jcs.key с параметром AES в ОС GNU/Linux

### 3.1.3. Создание ключа и вектора для преобразования данных с выбором БД и алгоритма AES

Для формирования ключа и вектора преобразования данных, расширение JCS должно быть установлено для выбранной базы данных.

В качестве примера, создадим тестовую БД «test\_db» и проверим список БД:

```
CREATE DATABASE test_db;  
\1
```

```

root@jcs: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# create database test_db;
CREATE DATABASE
postgres=# \l

               List of databases
  Name      | Owner   | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----
postgres   | postgres | SQL_ASCII | C       | C     | 
template0  | postgres | SQL_ASCII | C       | C     | =c/postgres      +
           |          |          |          |          | postgres=CTc/postgres
template1  | postgres | SQL_ASCII | C       | C     | =c/postgres      +
           |          |          |          |          | postgres=CTc/postgres
test_db    | postgres | SQL_ASCII | C       | C     | 
(4 rows)

postgres=#

```

Рисунок 3.5 – Создание тестовой БД в ОС GNU/Linux

Чтобы установить расширение потребуется подключиться к требуемой БД:

```
\connect test_db
```

```

root@jcs: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# \connect test_db
You are now connected to database "test_db" as user "postgres".
test_db=#

```

Рисунок 3.6 – Команда подключения к тестовой БД в ОС GNU/Linux

Установим расширение «jcs» в тестовой БД «test\_db»:

```

CREATE EXTENSION jcs;
\dx

```

```

root@jcs: /usr/jatoba-4/bin
File Edit View Search Terminal Help
test_db=# create extension jcs;
CREATE EXTENSION
test_db=# \dx
               List of installed extensions
  Name  | Version | Schema  | Description
-----+-----+-----+-----
 jcs    | 2.0     | public  | encrypted data in storage
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)

test_db=#

```

Рисунок 3.7 – Установка расширения в тестовой БД в ОС GNU/Linux

Аналогично SQL-команде, описанной в п. 3.1.2, можно выполнить SQL-команду для создания ключа и вектора преобразования данных текущей БД с указанием алгоритма преобразования данных.

Ключ и вектор шифрования может быть создан с выбором БД и алгоритма преобразования данных.

Для этого используется SQL-команда с параметрами:

- true – алгоритм преобразования данных AES;
- 'database name' – имя базы данных.

Например:

```
SELECT jcs.create_key(true, 'test_db');
```

```

root@jcs: /usr/jatoba-4/bin
File Edit View Search Terminal Help
test_db=# select jcs.create_key(true, 'test_db');
 create_key
-----
 t
(1 row)

test_db=#

```

Рисунок 3.8 – Формирование файла «jcs.key» для выбранной БД в ОС GNU/Linux

Файл «jcs.key» будет создан в каталоге по умолчанию. В случае когда выполнялись действия по созданию ключа и вектора преобразования данных для другой БД, в файл «jcs.key» будет добавлена запись, представленная на рисунке 3.9.

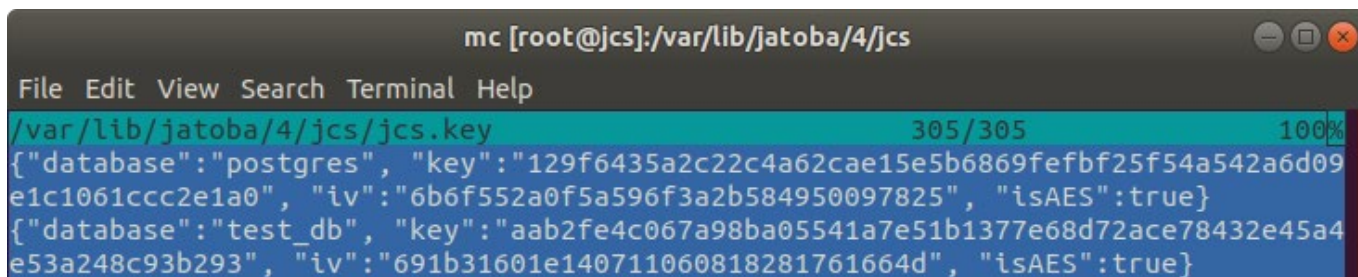


Рисунок 3.9 – Формат файла «jcs.key» в ОС GNU/Linux

### 3.1.4. Создание ключа и вектора для преобразования данных с выбором БД и алгоритма TwoFish

Применение алгоритма преобразования данных TwoFish с выбором БД, можно выполнить при помощи SQL-команды с параметрами:

- false – алгоритм преобразования данных TwoFish;
- 'user name' – имя пользователя.

Например:

```
SELECT jcs.create_key(false, 'postgres');
```

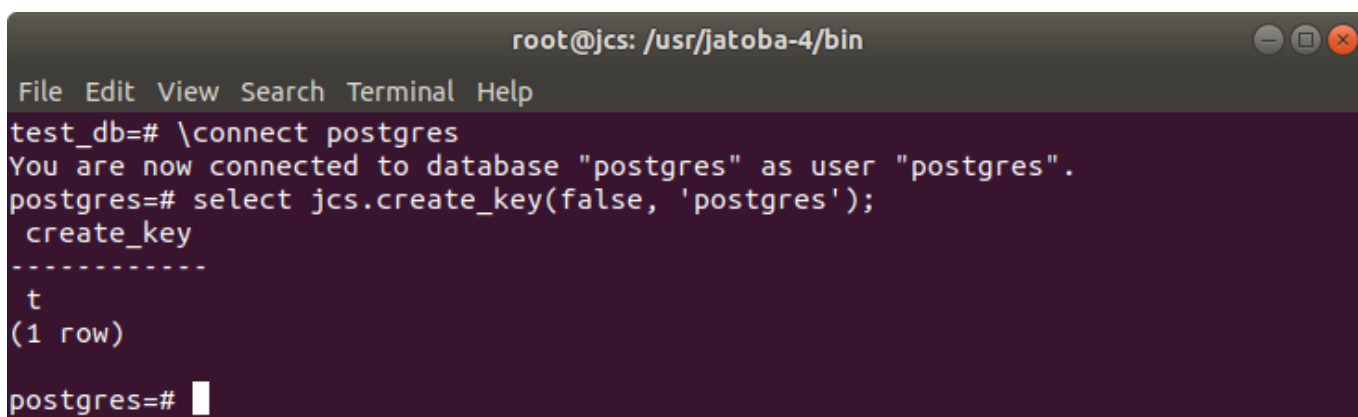


Рисунок 3.10 – Команда создания jcs.key с выбором БД и алгоритма TwoFish в ОС GNU/Linux

Файл «jcs.key» будет создан в каталоге по умолчанию. В нем будет указан метод преобразования данных AES (isAES) в значении «false», что соответствует алгоритму TwoFish.

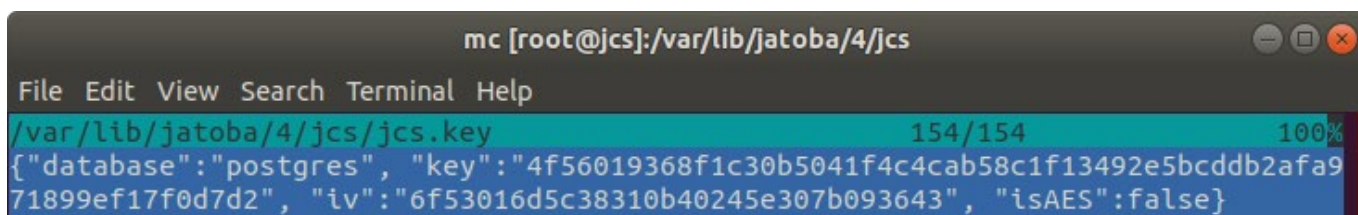


Рисунок 3.11 – Формат файла «jcs.key» с алгоритмом TwoFish в ОС GNU/Linux

### 3.1.5. Изменение пути сохранения файла ключей

При необходимости можно изменить путь хранения файла ключей, заданного по умолчанию. При этом директория должна быть защищена от возможного воздействия других пользователей ОС.

Например, необходимо сохранить файл ключей по пути:

```
'/var/lib/jatoba/<версия>/data/'
```

Изменение пути сохранения файла ключей выполняется SQL-командой:

```
SET jcs.work_dir = '/var/lib/jatoba/<версия>/data/';
```

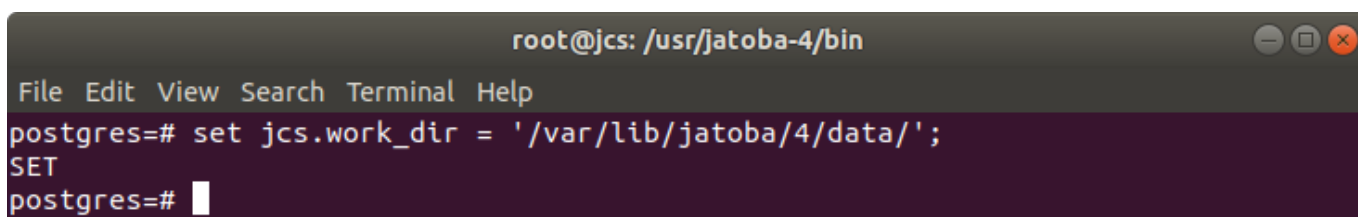


Рисунок 3.12 – Команда изменения пути хранения файла «jcs.key» в ОС GNU/Linux

После выполнения SQL-команды требуется проверить создание файла «jcs.key» в указанной директории.

### 3.1.6. Создание преобразованной таблицы

После создания ключа и вектора преобразования данных становится доступной возможность создания защищенной таблицы.

В качестве примера создадим таблицу с именем «table\_a» с использованием механизма преобразования данных SQL-командой:

```
CREATE TABLE table_a (i int, j int, n varchar(255)) using jcs;
```

Для последующей проверки доступности данных внесем две строки данных в таблицу:

```
insert into table_a values(1, 2, 'Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Nam ante. ');
insert into table_a values(2, 3, 'Morbi lobortis mattis
ultrices. Vestibulum elementum ut sem sed fringilla.');
```

### 3.1.7. Чтение данных из преобразованной таблицы

Вывести содержимое таблицы можно с помощью SQL-команды:

```
SELECT * from table_a;
```

При просмотре содержимого таблиц, все записи читаемы и полностью соответствуют тем, что добавлены после создания таблицы.

Для проверки доступности информации в таблице с преобразованными данными, создадим пользователя БД «user\_test» с атрибутом «Login»:

```
CREATE ROLE user_test NOSUPERUSER NOCREATEDB NOCREATEROLE
NOINHERIT LOGIN PASSWORD '12345678';
```

Предоставим все права на операции с таблицей «table\_a» в схеме данных «public»:

```
GRANT ALL ON TABLE public.table_a TO user_test;
```

Авторизуемся в БД «test\_db» от имени и с правами пользователя user\_test и сформируем аналогичный SQL-запрос на вывод содержимого таблицы «table\_a».

```
psql -h localhost -U user_test -d test_db
SELECT * from table_a;
```

## 3.2. Режим пользователя

В режиме пользователя, субъект может создавать собственные преобразованные таблицы. Доступ к таким таблицам возможен при вводе вектора и ключа преобразования данных.

### 3.2.1. Переключение в режим пользователя

Переключение в режим пользователя осуществляется SQL-командой:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------



```
SET jcs.user_mode = TRUE;
```

### 3.2.2. Ввод ключа и вектора преобразования данных

Ввод вектора преобразования данных осуществляется SQL-командой:

```
SET jcs.iv = '0178310C1A2968761662310B5F5F2100';
```

Ввод ключа преобразования данных осуществляется SQL-командой:

```
SET jcs.key =  
'0FCB80F6C6496DD21231764BD14369A66255E5EDD87F09E02B48A2729AC700  
05';
```

Ввод параметров выполняется в шестнадцатеричном формате, при этом они не сохраняются в файле jcs.key.

### 3.2.3. Создание преобразованной таблицы в режиме пользователя

После того, как был задан режим пользователя и введены ключ и вектор преобразования данных становится доступной возможность создания пользовательской таблицы с преобразованными данными.

```
CREATE TABLE table_c (i int, j int, n varchar(255)) using jcs;
```

Для последующей проверки доступности данных в таблице «table\_c», внесем две строки данных в таблицу:

```
insert into table_c values(1, 2, 'Lorem ipsum dolor sit amet,  
consectetur adipiscing elit. Nam ante.');
```

```
insert into table_c values(2, 3, 'Morbi lobortis mattis  
ultrices. Vestibulum elementum ut sem sed fringilla.');
```

### 3.2.4. Чтение данных из преобразованной таблицы созданной в режиме пользователя

В режиме пользователя и при введенном ключе и векторе преобразования данных, вывести содержимое таблицы можно при помощи SQL-команды:

```
SELECT * from table_c;
```

Если сеанс пользователя был прерван или закончен, при обращении к таблице будет выведено сообщение:

```
Error executing the cryptographic function
```

Для чтения данных из таблицы, после прерванного сеанса подключения СУБД, потребуется:

- перейти в режим пользователя:

```
SET jcs.user_mode = true;
```

- ввести ключ и вектор преобразования данных:

```
SET jcs.iv = '0178310C1A2968761662310B5F5F2100';  
SET jcs.key =  
'0FCB80F6C6496DD21231764BD14369A66255E5EDD87F09E02B48A2729AC700  
05';
```

- ввести SQL-команду чтения данных из таблицы:

```
SELECT * from table_c;
```

Рассмотрим пример обращения привилегированного пользователя.

Доступ к таблице «table\_c», созданной в пользовательском режиме, осуществляется от имени и с правами пользователя postgres:

```
psql -h localhost -U postgres -d test_db  
SELECT * from table_c;
```

В результате будет выведена ошибка:

```
Error executing the cryptographic function
```

Из чего следует, что другие пользователи, в независимости от имеющихся привилегий, не имеют доступа к преобразованной таблице.

### 3.3. Просмотр директории хранения ключей

Просмотр директории хранения ключей доступен как привилегированному пользователю, так и пользователю СУБД.

Просмотр директории хранения ключей выполняется SQL –командой:

```
SHOW jcs.work_dir;
```

В результате будет выведен путь к директории.

Аналогичный вывод информации будет для пользователя СУБД.

### 3.4. Просмотр ключа

Просмотр ключа выполняется SQL –командой:

```
SHOW jcs.key;
```

Выводится значение параметра «key» из файла «jcs.key», если SQL-запрос был выполнен от имени и с правами пользователя «postgres».

Если SQL-запрос выполнен от имени и с правами пользователя СУБД, значение параметра «key» будет скрыто.

### 3.5. Переключение в режим администратора

Переключение в режим администратора выполняется SQL – командой:

```
SET jcs.user_mode = FALSE;
```

### 3.6. Получение номера текущей версии компонента

Версия компонента выводится SQL-командой:

```
SELECT jcs.version();
```

## 4. УДАЛЕНИЕ КОМПОНЕНТА

### 4.1. Удаление компонента при отсутствии зависимых от него объектов

Удаление компонента осуществляется средствами пакетного менеджера ОС. При этом нужно использовать команду удаления, соответствующую пакетному менеджеру: remove, purge, erase и т.п.

Для удаления компонента потребуется авторизоваться в СУБД и выполнить команду:

```
DROP EXTENSION jcs;
```

В ОС GNU/Linux требуется выйти из psql и удалить пакет расширения, выполнив команду:

```
apt-get remove jatoba18-jcs
```

### 4.2. Удаление компонента при наличии зависимых от него объектов

Для удаления компонента вместе со всеми зависимыми от него объектами потребуется авторизоваться в СУБД и выполнить команду:

```
DROP EXTENSION jcs cascade;
```

## 5. СООБЩЕНИЯ ОБ ОШИБКАХ

### 5.1. Повторное создание ключа и вектора для БД

Создание ключа и вектора для преобразования данных с параметрами или по умолчанию осуществляется с помощью SQL-команды:

```
SELECT jcs.create_key();
```

В случае, если эти действия были ранее выполнены, то СУБД выведет ошибку:

```
Key is already defined for database
```

Это означает, что повторно сгенерировать ключ и вектор преобразования данных для текущей БД невозможно.

### 5.2. Создание преобразованной таблицы без предварительной генерации ключа и вектора преобразования данных

Создание преобразованной таблицы осуществляется с помощью SQL-команды:

```
CREATE TABLE table_a (g int) using jcs;
```

Если файл «jcs.key» отсутствует, то СУБД выведет сообщение об ошибке:

```
ERROR: Error reading encryption file
```

### 5.3. Чтение данных из преобразованной таблицы с отсутствующим файлом ключей

Чтение данных из преобразованной таблицы будет невозможно в случае удаления или перемещения файла «jcs.key» из директории хранения, СУБД выведет ошибку:

```
Error reading encryption file
```

Восстановить доступ к данным возможно, только при возвращении оригинального файла «jcs.key» в место хранения.

### 5.4. Чтение из преобразованной таблицы с поврежденным/измененным файлом «jcs.key»

Чтение данных из преобразованной таблицы будет невозможно в случае повреждения или изменения файла «jcs.key», СУБД выведет ошибку:

Error: Error executing the cryptographic function

Восстановить доступ к данным возможно только при возвращении оригинального файла «jcs.key» в место хранения.

#### **5.5. Создание ключа и вектора для шифрования с выбором БД без предустановленного расширения JCS**

В случае, если компонент JCS не установлен, то при попытке создания ключа и вектора для шифрования, СУБД выведет сообщение об ошибке:

SQL Error [3F000]: ERROR: schema "jcs" does not exist

## **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ**

SQL	–	Structured Query Language – язык структурированных запросов
CLI	–	Command-line interface – интерфейс командной строки
БД	–	База данных
ОС	–	Операционная система
СУБД	–	Система управления базами данных
ФСТЭК России	–	Федеральная служба по техническому и экспортному контролю России

## Лист регистрации изменений

№ изменения: \_\_\_\_\_

Подпись отв. лица: \_\_\_\_\_

Дата внесения изм: \_\_\_\_\_