Right Line СБП-Коннект. Руководство по администрированию для АО «ДОМ.РФ»

Общее описание

СБП-Коннект предназначен для подключения кредитных организаций к Системе быстрых платежей. В данном руководстве рассматривается версия **C2C (Consumer-to-consumer)**.

СБП-Коннект состоит из нескольких модулей, которые взаимодействуют между собой. Для постоянного хранения данных используется реляционная база данных.

Каждый из модулей является отдельным приложением, реализованным в виде единого исполняемого **jar файла**, включающего в себя все необходимые библиотеки для работы.

Основные модули СБП-Коннект

Модуль client-app

Модуль реализует все основные сценарии переводов C2C. Также, модуль осуществляет взаимодействие с НСПК и интеграцию с расчетными и учетными системами Банка. Для интеграции, у client-app имеется свой API.

Архитектура client-app

```
sbp-connect-client-app/
 ├──── certs - каталог с контейнерами p12
 │   ├──── keystore.p12 - контейнер с ключами и сертификатами
 │   └──── truststore.p12 - контейнер с доверенными сертификатами
 ├──── config - каталог с конфигурационными файлами
 │   └──── application.yml.example - пример конфигурационного файла с описанием блоков
настроек
 ├──── ignite - создаётся сервисом автоматически
 │   ├──── README.txt
 │   └──── work
 │       ├──── db
 │       ├──── diagnostic
 │       └──── marshaller
 │           ├──── 104259563.classname0
 │           ├──── 104290315.classname0
 │           └──── -929047093.classname0
 ├──── logs -  логи
 ├──── sbp-connect-client-app.conf - конфиг с указанием кодировки JAVA_OPTS
 └──── sbp-connect-client-app.jar - основной jar файл микросервиса
```

Модуль crypto-app

Модуль реализует интеграцию с СКЗИ (СКАД «Сигнатура») и предоставляет API для использования крипто-функций. Развертывание модуля осуществляется на машинах с СКЗИ. Установка данного модуля на сервера ПРОД, обычно осуществляется силами сотрудников банка.

Архитектура crypto-app

```
Services/
 ├──── application.yml - конфигурационный файл с описанием блоков настроек
 ├──── bat - скрипты по установке/удалению , запуску/остановке сервиса
 │    ├──── installService.bat - установка
 │    ├──── queryService.bat
 │    ├──── setenv.bat
 │    ├──── startService.bat - запуск
 │    ├──── stopService.bat - остановка
 │    ├──── uninstallService.bat - удаление
 │    ├──── wrapper.bat
 │    └──── wrapperW.bat
 ├──── log  - логи запуска микросервиса
 ├──── logs - логи зашифрованных сообщений и взаимодействия с СКЗИ
 ├──── conf
 │    ├──── logback.groovy - файл настройки логирования
 │    └──── wrapper.conf - конфигурационный файл для installService.bat установки в
services.msc
 ├──── lib - библиотеки
 ├──── sbp-crypto-1.3-SNAPSHOT.jar - основной jar файл микросервиса
 ├──── wrapperApp.jar - jar файлы отвечают за запуск и установку сервиса
 └──── wrapper.jar - jar файлы отвечают за запуск и установку сервиса
```

Модуль dashboard-app

Модуль представляет собой веб-приложение, которое работает во всех современных браузерах. Приложение предназначено для сотрудников Банка и предоставляет им данные по операциям, сводную статистику, результаты сверки и прочее.

Архитектура dashboard-app

```
sbp-connect-dashboard/
 ├──── config
 │    ├──── application.yml.example  - пример конфигурационного файла
 ├──── logs - директория, куда сохраняются логи.
 │    ├──── 2020-12-08.root.log
 │    └──── 2020-12-16.root.log
 ├──── sbp-connect-dashboard-1.6.0-RELEASE.conf - конфигурационный файл с кодировкой
 ├──── sbp-connect-dashboard-1.6.0-RELEASE.jar - основной пакет backend
 └──── src
 └──── resources
 └──── unzipped-reports - директория для сохранения отчётов
 └──── 100000000032
 ├──── 2020-11-15
 └──── 2020-11-16
```

Состав дистрибутива СБП-Коннект

Дистрибутив СБП-Коннект включает в себя:

- модуль client-app, rpm пакет - `sbp-connect-domrf-client-app{-номер_версии}~RELEASE.x86_64.rpm`
- модуль crypt-app, архив - `sbp-crypto-{номер_версии}.zip`
- модуль веб-интерфейса dashboard-app, rpm пакет - `sbp-connect-domrf-dashboard{номер_версии}~RELEASE.x86_64.rpm`
- архив с UI для dashboard-app -`sbp-connect-admin-ui-domrf-{номер_версии}.zip`
- папку с файлами `liquibase` для настройки базы данных (БД)

Установка и настройка модуля Crypto-app

1. Требования к программно-техническому обеспечению стенда

Для функционирования модуля crypto-app необходимо как минимум: 2 core x 2GHz CPU, 4Gb RAM, 60 Gb SSD/HDD. В качестве операционной системы может использоваться 64-битная ОС Windows 10 или Windows Server.

Требования к программному обеспечению

- наличие установленного окружения Java Runtime Environment 11 версия или выше.
- учётная запись с правами Windows (AD): Создание и запуск служб либо с правами локального администратора
- установленный программный комплекс СКАД "Сигнатура", где должен быть настроен профиль со справочниками сертификатов (сертификаты банка и СБП)
- прикладной программный интерфейс СКАД «Сигнатура» для платформы Java

1.1 Настройка переменных среды окружения Windows

Общие сведения Переменная `PATH` — это системная переменная, которую операционная система использует для того, чтобы найти нужные исполняемые объекты в командной строке или окне терминала.

Системную переменную PATH можно задать с помощью системной утилиты в панели управления Windows по следующей инструкции:

1. В строке "Поиск" выполните поиск: Система (Панель управления) (либо на рабочем столе правой кнопкой нажмите на значок Компьютер ⇒ Свойства)
2. Откройте ссылку "Дополнительные параметры системы".

3. Выберите "Переменные среды".
4. В разделе "Системные переменные" нажмите "Создать" и укажите имя переменной: JAVA_HOME, значение переменной: полный путь к рабочей директории OpenJDK, например (C:\java)
5. В разделе "Системные переменные" выберите переменную среды `PATH`. Нажмите "Изменить".
6. В окне "Изменение системной переменной" нажмите "Создать" и укажите значение: %JAVA_HOME%\bin
7. Нажмите "ОК". Закройте остальные открытые окна, нажимая "ОК".
8. Откройте окно командной строки и выполните команду `java --version`.

2. Установка модуля Crypto-app

Архив `sbp-crypto-{номер_версии}.zip` необходимо распаковать в папку в "C:\Services\SbpCrypto"

3. Настройка модуля crypt-app

ВНИМАНИЕ!	При настройке будет рассматриваться рабочая директория сервиса. Пути к файлам будут даваться относительно данной директории, т.е. вместо "C:\SbpCrypto\sbp-crypto-{номер_версии}\some_dir\some_file", будет указываться `some_dir\some_file`

3.1 Настройка аутентификации для сервиса(службы)

Откройте файл `conf\wrapper.conf` и укажите параметры учётной записи Windows (AD), от имени которой будет устанавливаться сервис:

```
wrapper.app.account = [аккаунт]
wrapper.app.password = [пароль]
```

ВНИМАНИЕ!	У данной учётной записи должен быть доступ к профилю СКАД Сигнатура (т.е. к справочникам сертификатов)

Также в файле `conf\wrapper.conf` должно быть прописано правильное имя файла модуля crypto-app в виде:

wrapper.java.app.jar = .\\\\sbp-crypto-1.4.0-SNAPSHOT.jar

Далее, нужно запустить скрипт bat\installService.bat, который создаст службу с именем(обычно SbpCryptoService) указанным во wrapper.conf С помощью апплета Панели Управления Администрирование\Службы необходимо удостоверится, что в службах Windows появился пункт SbpCryptoService

Пример настройки wrapper.conf

```
wrapper.working.dir=${wrapper_home}

wrapper.tmp.path = ${jna_tmpdir}

wrapper.app.account = [аккаунт]
wrapper.app.password = [пароль]

wrapper.console.loglevel=INFO

wrapper.logfile=${wrapper_home}/log/wrapper.log
wrapper.logfile.maxsize=10m
wrapper.logfile.maxfiles=10

wrapper.console.title=SbpCryptoService
wrapper.ntservice.name=SbpCryptoService
wrapper.ntservice.displayname=SbpCryptoService
wrapper.ntservice.description=SbpCryptoService

wrapper.daemon.run_level_dir=${if (new File('/etc/rc0.d').exists()) return
'/etc/rcX.d' else return '/etc/init.d/rcX.d'}
wrapper.tray = true
wrapper.tray.port = 15002
wrapper.on_exit.0=SHUTDOWN
wrapper.on_exit.default=RESTART
wrapper.on_signal.9=SHUTDOWN
wrapper.on_signal.default=RESTART
wrapper.filter.trigger.0=Exception
wrapper.filter.script.0=${wrapper_home}/scripts/trayMessage.gv
wrapper.filter.script.0.args=Exception
wrapper.java.app.jar = .\\\\sbp-crypto-X.X.X-SNAPSHOT.jar
wrapper.java.command = ${JAVA_HOME}\\\\bin\\\\java.exe
wrapper.java.additional.1=-server
wrapper.java.additional.2=-Dfile.encoding=UTF-8
wrapper.java.additional.3=-Djava.net.preferIPv4Stack=true
wrapper.java.additional.4=-Dlogback.configurationFile=conf\\\\logback.groovy
```

3.2 Настройка сервиса crypto-app

Параметры настройки crypto-app прописываются в файле `application.yml`

ВНИМАНИЕ!	В данном файле используется YAML-синтаксис, т.е. необходимо сохранение имеющихся отступов. При нарушении форматирования возможна некорректная работа/невозможность запуска сервиса.

Значения в конфигурационном файле (в т.ч. пароли) могут быть указаны как в открытом, так и в закрытом(зашифрованном) виде: Сформировать зашифрованное значение можно с помощью инструмента Jasypt CLI Tools. Ключ для шифрования (salt) будет предоставлен

отдельно. Более подробную информацию о Jasypt можно найти на http://www.jasypt.org/cli.html

Если справочники СКАД "Сигнатура" располагаются не в реестре, а в виде файлов (registry: false в application.yml), то необходимо указать расположение этих файлов. Для этого в рабочей директории сервиса (в той же, где располагается .jar-файл сервиса), необходимо создать текстовый конфигурационный файл pki1.conf со следующим содержимым:

```
default: test # Имя профиля СКАД "Сигнатура" по умолчанию
local: test # Имя локального профиля СКАД "Сигнатура"
pse: pse://signed/C:\SKAD\profiles\local.pse # Путь к Персональному Справочнику
Сертификатов
localstore: file://C:\SKAD\profiles\local.gdbm # Путь к Локальному Справочнику
Сертификатов
```

3.3 Настройка сертификатов.

Т.к. модуль `crypto-app` связывается с модулем connect-app по протоколу https - необходима настройка TLS-сертификатов. В модуле crypto-app для этой задачи используются два криптоконтейнера стандарта PKCS12:

- контейнер, содержащий закрытый ключ и сертификат сервера, на котором размещается модуль crypto-app (в примере конфигурационного файла обозначен как keyStore.p12)
- контейнер, содержащий доверенный сертификат, т.е. сертификат сервера, от которого разрешено принимать соединения (в примере конфигурационного файла обозначен как TrustStore.p12)

Все сертификаты, упомянутые в данном разделе, генерируются с помощью криптографической инфраструктуры банка (являются самоподписнными).

3.4 Протоколирование работы модуля crypto-app

Модуль `crypto-app` использует библиотеку логгирования logback. Для настройки параметров используется отдельный конфигурационный файл `conf\logback.groovy`. В нем можно указать уровень логгирования (INFO, DEBUG, TRACE), расположение и формат логов, параметры ротации и архивирования и т.д.

С синтаксисом logback.groovy можно ознакомится на http://logback.qos.ch/manual/groovy.html

Информация о шифруемых сообщениях записывается в файл `logs\%d{yyyy-MM-dd}.crypto`

Пример конфигурационного файла application.yml для sbp-crypto:

```
users: # параметры пользователей крипто-сервиса
test: # логин пользователя для базовой аутентификации на сервисе
type: SIGNATURA # тип шифрования
authPassword: ENC(t8jfrDvmGFkwWDF1TDIy/Q==) # пароль пользователя для базовой
аутентификации на сервисе, в данном случае функцией ENC() указан его хэш
profile: test # имя профиля со справочниками сертификатов для шифрования, уточнить его
можно в настройках СКАД "Сигнатура"
registry: false # расположение справочников СКАД "Сигнатура", true - реестр Windows,
false - из локальных файлов справочников
ecryptKeyId: 1297CHCMRP01 # id ключа для шифрования и проверки подписи - уточнить его
можно в настройках СКАД "Сигнатура"
serialNumber: 40:50:13:C0:DF:5A:0D:92:5C:5D:AF:85:5D:EE:5F:C0 # серийный номер ключа
для шифрование и проверки подписи, уточнить его можно в настройках СКАД "Сигнатура"
ssl:
password: ENC(gIOUHfvfkUWRgfsw8+jHr0NTEAHp) # переменная служит для записи пароля от
криптохранилищ сервиса
logging:
config: ./conf/logback.groovy # путь к файлу настроек логирования
server:  # настройки сервера
port: 443 # порт для основной группы эндпоинтов
ssl: # настройка сертификатов
key-store-type: PKCS12   # тип контейнера, допустимые параметры PKCS12 и JKS
key-store: ./conf/keyStore.p12 # расположение криптоконтейнера с ключевой парой
(закрытый ключ+сертификат)
key-store-password: ${ssl.password} # пароль от контейнера (в данном случае,
подставляется с помощью переменной)
trust-store-type: PKCS12 # тип контейнера с доверенными сертификатами параметры PKCS12
и JKS
trust-store: ./conf/TrustStore.p12  # путь до контейнера с доверенными сертификатами
trust-store-password: ${ssl.password}  # пароль от контейнера с доверенными
сертификатами
client-auth: need # необходима ли проверка сертификата удаленного клиента при ssl-
handshake
```

Установка и настройка СБП-Коннект

1.1 Минимальные системные требования для установки

Для построения отказоустойчивой системы, потребуется как минимум два виртуальных (или физических) хоста - один для установки модулей, второй для размещения базы данных и балансировщик сетевой нагрузки.

Для функционирования модулей `client-app` и `dashboard-app` необходимо как минимум: 2 core x 2GHz CPU, 4Gb RAM, 60 Gb SSD/HDD. В качестве операционной системы может использоваться 64-битная ОС Linux (желательно CentOS 7) с предустановленным ПО - Java 11

(OpenJDK JRE или OracleJRE) и Nginx 1.16.

В качестве СУБД может использоваться PostgreSQL/MySQL/Microsoft SQL/Oracle Database.

Балансировщик может быть как аппаратным, так и программным решением.

1.2 Установка модулей client-app и dashboard-app

Установка rpm-пакета `sbp-connect-domrf-client-app-{номер_версии}~RELEASE.x86_64.rpm` осуществляется в папку /opt/sbp-connect-domrf-client-app/ автоматически, с помощью менеджера пакетов rpm.

Команда для установки в командной строке Linux не из под root пользователя:

```
sudo rpm -ivh --force #sbp-connect-domrf-client-app-{номер_версии}~RELEASE.x86_64.rpm
```

В процессе установки, в папке /etc/init.d/ создаётся символическая ссылка на установленный (исполняемый) jar-файл.

Установка rpm-пакета `sbp-connect-domrf-dashboard-{номер_версии}~RELEASE.x86_64.rpm` осуществляется в папку /opt/sbp-connect-domrf-dashboard/

Команда для установки в командной строке Linux не из под root пользователя:

```
sudo rpm -ivh --force sbp-connect-domrf-dashboard-{номер_версии}~RELEASE.x86_64.rpm`
```

В процессе установки, в папке /etc/init.d/ создаётся символическая ссылка на установленный (исполняемый) jar-файл.

1.3 Инициализация базы данных

Дистрибутив поставляется вместе с инструментом `liquibase`, который обеспечивает создание и обновление схемы базы данных(БД). При установке приложения sbp-connect-client-app, в папке /opt/sbp-connect-domrf-client-app/liquibase размещается все необходимое для работы liquibase и начальной инициализации схемы БД, а так же пример конфигурационного файла application.yml.example.

Предварительно, необходимо создать БД, с которой он будет работать `sbp-connect-client-app`. Также рекомендуется создать/назначить пользователя-владельца данной схемы.

Перед запуском Liquibase в файле liquibase.properties необходимо указать настройки подключения к БД.

Для PostgreSQL:

```
url = jdbc:postgresql://ip:port/dbName
#defaultSchemaName=
```

Для развертывания схемы в базу данных необходимо запустить скрипт run.sh:

```
./run.sh update
```

При запуске, скрипт потребует ввода имени пользователя - владельца рабочей схемы БД (либо иного пользователя БД, имеющего право записи в используемую схему) и его пароля.

Пример вывода скрипта установки:

```
user@server liquibase]$ ./run.sh update
username: user
password: password
execute: update
Starting Liquibase at WEEKDAY, dd mm yyyy hh:mm:ss YEKT (version 3.6.3 built at yyyy
hh:mm:ss)
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.vmplugin.v7.Java7$1
(file:/opt/sbp-connect-sngb-c2c/liquibase/lib/groovy-2.5.8.jar) to constructor
java.lang.invoke.MethodHandles$Lookup(java.lang.Class,int)
WARNING: Please consider reporting this to the maintainers of
org.codehaus.groovy.vmplugin.v7.Java7$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective
access operations
WARNING: All illegal access operations will be denied in a future release
Liquibase: Update has been successful.
```

1.4 Список таблиц базы данных

Имя таблицы	**Описание**
bank_info_tab	Профиль банка
c2b_refund_tab	Возвраты c2b
c2b_transfer_tab	Платежи c2b
c2c_mismatch_tab	Старая таблица (больше не используется)
c2c_transfer_tab	Платежи c2c
databasechangelog	Технологическая таблица liquibase. Отображает внесенные изменения
atabasechangeloglock	Технологическая таблица liquibase
file_journal_tab	Реестр обмена ed форм (таблица устарела)
nspk_response_code	Таблица мапинга nspk кодов на коды из АБС
receiver_transfer_data_tab	Старая таблица (больше не используется)
reconciliation_source_data_tab	Сверки. Поля из источника
reconciliation_source_tab	Сверки. Источник. ссылается на таблицу reconciliation_source_data_tab
rtln_process_tab	Таблица для процессов в процесс менеджере
rtln_stage_tab	Таблица для стадий в процесс менеджере

Имя таблицы	Описание
rtln_step_tab	Таблица шагов в процесс менеджере
sbp_connect_message_tab	Сообщения
sbp_connect_user_tab	Пользователи личного кабинета СБП-Коннект
status_tab	Старая таблица (больше не используется)
transaction_reconciliation_tab	Сверки. основная таблица сверок

1.5 Настройка базы данных

Для корректной работы службы `sbp-connect-client-app`, в БД изначально должна быть информация о параметрах банка. Первой всегда заполняется таблица `BANK_INFO_TAB`.

В таблицу BANK_INFO_TAB необходимо добавить данные банка:

```
member_id - идентификатор Банка (выдается НСПК)
endpoint_id - Endpoint Банка (выдается НСПК)
opkc_member_id 000000000000 - идентификатор ОПКЦ
bic xxxxxxxxx - БИК
```

```
INSERT INTO BANK_INFO_TAB (sbp_member_id, sbp_endpoint, opkc_member_id, bic) VALUES
(sbpMemberIdValue, sbpEndpointValue, opkcMemberValue, bic);
```

В таблицу SBP_CONNECT_USER_TAB необходимо добавить данные пользователя для аутентификации в модуле dashboard-app и API модуля client-app id - идентификатор пользователя (в данной таблице)

```
name - имя пользователя
password - пароль
bank_info_id - идентификатор Банка (выдается НСПК)
roles - роли пользователя, согласно ролевой модели(см. Таблицу 1)
```

```
INSERT INTO SBP_CONNECT_USER_TAB (id, name, password, bank_info_id,roles) VALUES
(nextval('sbp_connect_user_seq'), basicAuthUserName, basicAuthUserPass,
sbpMemberIdValue,'ROLE_APP,ROLE_READER_TRANSFERS,ROLE_WRITER_TRANSFERS,ROLE_READER_MES
SAGES,ROLE_WRITER_RECONCILIATION,ROLE_READER_RECONCILIATION');
```

ВНИМАНИЕ!

basicAuthUserPass для поля password, в таблице SBP_CONNECT_USER_TAB, должно быть сформировано при помощи хэш-функции BCrypt. Для этого можно воспользоваться любым из онлайн-сервисов: https://www.browserling.com/tools/bcrypt https://bcrypt-generator.com/

```
При использовании ОС Linux, для хэширования пароля оффлайн можно воспользоваться
утилитой htpasswd из дистрибутива веб-сервера Apache:
htpasswd -bnBC 10 '' 'пароль' | sed 's/$2y/$2a/'
```

Таблица 1. Ролевые модели

Роль	**Предоставляемые права**
ROLE_APP	роль для работы сервиса с БД
ROLE_READER_TRANSFERS	просмотр переводов
ROLE_WRITER_TRANSFERS	ручной запуск переводов в АБС
ROLE_READER_MESSAGES	просмотр сообщений в рамках операции
ROLE_READER_RECONCILIATION	просмотр сверок
ROLE_WRITER_RECONCILIATION	ручной запуск сверки

Также необходимо занести в таблицу nspk_response_code значения ошибок НСПК.

Значения ошибок НСПК

```
INSERT INTO "public"."nspk_response_code" VALUES ('DEFAULT', 'Неизвестная ошибка',
NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05001', 'Недостаточно данных об
Отправителе или Получателе', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('B05002', 'Невозможно зачислить
сумму перевода на счет Получателя', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05008', 'Нет такого значения ЭБД
{24} Идентификатор Банка Получателя (ИД БП)', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05014', 'Нет такого значения ЭБД
{47} Тип Идентификатора Получателя (ТИД ПО)', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05021', 'PAM Отправителя и
Получателя не совпали (только для Me2Me)', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05037', 'Ограничения
законодательства', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05038', 'Ограничения
законодательства', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('B05005', 'Запрещено кредитование
счета Получателя', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('B05006', 'Найден больше чем один
Получатель', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('B05007', 'Не найден Получатель',
NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('B05008', 'Ограничения
законодательства на зачисление (например, сумма превысила допустимую для данного
платежного средства или уровень идентификации недостаточен)', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('B05009', 'Получатель не дал
согласие на получение средств через СБП', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('B05010', 'Получатель отказался от
получения средств через СБП', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('B05011', 'Счет Получателя
заблокирован или закрыт', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('B05013', 'Счет Получателя не
найден', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05999', 'Технологические работы в
Банке получателя. Попробуйте повторить перевод через некоторое время.', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I07002', 'Превышено время ожидания
ответа. Пожалуйста, повторите перевод позже.', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I01091', 'Превышено время ожидания
ответа. Пожалуйста, повторите перевод позже.', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I04010', 'Превышено время ожидания
ответа. Пожалуйста, повторите перевод позже.', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05043', 'Свяжитесь с Получателем
средств и уточните реквизиты для зачисления денежных средств.', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05019', 'OTP введен неверно.
Операция завершена', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I05020', 'OTP введен неверно.
Повторите ввод.', NULL, NULL);
INSERT INTO "public"."nspk_response_code" VALUES ('I07005', 'Превышенно допустимое
количество попыток по установке Банка по умолчанию в день', NULL, NULL);
```

1.6 Настройка модуля client-app

После установки модуля client-app, его необходимо настроить и подготовить к первому запуску.

В папке /opt/sbp-connect-domrf-client-app/config/ располагается пример конфигурационного файла application.yml.example. На его основе нужно создать конфигурационный файл по следующей инструкции:

1. Копируем либо переименовываем application.yml.example в application.yml
2. Открываем файловым редактором application.yml, читаем описание блоков конфига
3. Вносим изменения и URI в соответствии с настройками подключения к вашим БД и микросервисам.

ВНИМАНИЕ!	В данном файле крайне важно сохранять формат отступов. Если формат не будет сохранен, это может привести к аварийной остановке или некорректной работе модуля.

1.6.1 Настройка модуля dashboard-app

Пример конфигурационного файла application.yml.example для dashboard-app находится в папке /opt/sbp-connect-domrf-dashboard/config/. На его основе, необходимо создать конфигурационный файл application.yml и отредактировать необходимые параметры.

ВНИМАНИЕ!	В данном файле крайне важно сохранять формат отступов. Если формат не будет сохранен, это может привести к аварийной остановке или некорректной работе модуля.

1.6.2 Установка UI для dashboard-app

Содержимое архива с UI Нужно разархивировать в папку /var/www/dashboard

1.6.3 Настройка nginx для dashboard-app

По умолчанию dashboard-app слушает порт 8080, но при желании его можно изменить, добавив в конец конфигурационного файла application.yml следующие строки:

```
server: # настройки сервера
port: 1443 # порт для основной группы эндпоинтов
```

1.6.4 Пример конфигурационного файла для nginx

Пример конфигурационного файла nginx

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
```

```
pid        /var/run/nginx.pid;
events {
worker_connections  1024;
}
http {
sendfile on;
tcp_nopush on;
tcp_nodelay on;
reset_timedout_connection on;
client_body_timeout 10;
keepalive_timeout 65;
include mime.types;
log_format  '[$time_local] - $remote_addr - $host - $addr - $remote_user'
'"$request" $status $body_bytes_sent '
'"$http_referer" "$http_user_agent"';
limit_req_zone $binary_remote_addr zone=stoptheflood:10m rate=1r/s;
server {
listen 80;
server_name sbp-app01.roscap.com 10.12.213.43;
return 301 https://$server_name:8443;
}
server  {
listen 8443 default_server ssl http2;
root /var/www/dashboard/;
error_page 404 =200 /index.html;
index index.html;
charset utf8;
server_name sbp-app01.roscap.com:8443;
access_log /var/log/nginx/dashboard_access.log;
error_log /var/log/nginx/dashboard_error.log;
ssl_certificate        dash_certs/server/dashboard.pem;
ssl_certificate_key    dash_certs/server/dashboard_pass.key;
ssl_trusted_certificate dash_certs/root/dash_ca.pem;
ssl_verify_client off;
ssl_password_file dash_certs/passwd;
# JS & CSS files
location ~* \.(?:css|js)$ {
try_files $uri =404;
expires 1y;
access_log off;
add_header Cache-Control "public";
}
# Any route containing a file extension
location ~ ^.+\..+$ {
try_files $uri =404;
}
#Main location
location / {
if ($request_uri = /login) {
set $test ml_;
}
```

```
if ($request_method = POST) {
set $test "${test}mr";
}
if ($test = ml_mr) {
proxy_pass http://localhost:8080;
}
}
#Proxifying to backend
location /sbp/ {
proxy_pass http://localhost:8080;
}
}
}
```

1.7 Протоколирование работы модуля client-app

Модули sbp-connect-client-app и dashboard-app использует библиотеку логгирования `logback` . Для настройки параметров используется отдельный конфигурационный файл `config/logback.groovy`. В нем можно указать уровень логирования (INFO, DEBUG, TRACE), расположение и формат логов, параметры ротации и архивирования и т.д. В качестве примера, в директории `config` располагается `logback.groovy.example`, который показывает некоторые возможности конфигурации. С синтаксисом logback.groovy можно ознакомится на http://logback.qos.ch/manual/groovy.html

В application.yml имеется блок конфига:

```
logging:
config: ./config/logback.groovy # путь к файлу настроек логирования
```

Файлы логов расположены в /opt/sbp-connect-domrf-client-app/logs/

- логи об обмене сообщениями с НСПК - {yyyy-MM-dd}.rest.log
- логи взаимодействия с крипто-сервисом - {yyyy-MM-dd}.crypto.log
- все остальные логи(в том числе и от dashboard-app)- {yyyy-MM-dd}.root.log

Обновление модулей СБП-Коннект

Дистрибутив с обновлением обычно поставляется в виде zip-архива `sbp-connect-domrf-{версия_релиза}-RELEASE.zip`

Дистрибутив с обновлениями обычно включает в себя:

- модуль client-app - rpm пакет `sbp-connect-domrf-client-app-{версия_релиза}~RELEASE.x86_64.rpm`
- модуль веб-интерфейса dashboard-app - rpm пакет `sbp-connect-domrf-dashboard-{версия_релиза}~RELEASE.x86_64.deb`
- модуль C2B-переводов - `sbp-connect-domrf-merchant-app-{версия_релиза}~RELEASE.x86_64.rpm`

- папка с файлами `liquibase` для настройки базы данных (БД)
- папка `documentation` с файлами документации

Доставьте архив с обновлением программы на требуемый узел.

Перед обновлением нужного модуля необходимо остановить службу sbp-connect-client-app или sbp-connect-dashboard

2.1 Остановка службы

Для остановки службы, используется следующая команда, не из под root пользователя:

```
sudo service sbp-connect-domrf-client-app stop
```

Для проверки состояния службы, используется следующая команда, не из под root пользователя:

```
service sbp-connect-domrf-client-app status
```

2.2 Обновление модулей

Архив с дистрибутивом необходимо распаковать во временную папку на удаленном комьютере или скопировать на удаленный компьютер уже в распакованном состоянии.

На удаленном компьютере сделайте распакованную папку текущей:

```
cd <имя_папки_распакованного_архива>
```

Для обновления модуля используйте команду:

```
sudo rpm -Uvh --force <имя_пакета>.rpm
```

Установка rpm-пакета осуществляется в директорию /opt/sbp-connect-domrf-client-app/

Также, при установке в директории /etc/init.d/ создаётся символическая ссылка на сервис (исполняемый jar-файл).

В директории /opt/sbp-connect-domrf-client-app/ присутствует файл sbp-connect-domrf-client-app-{версия_релиза}.conf, который по имени должен совпадать с именем исполняемого jar-файла, находящегося в этой же папке.

2.3 Обновления базы данных.

Перед обновлением схемы базы данных(БД), нужно сделать полный бэкап базы. Для развёртывания компонентов схемы применяется библиотека Liquibase, всё необходимое находится в директории liquibase, включая пример конфигурационного файла liquibase.properties.example.

Перед запуском Liquibase в файле liquibase.properties необходимо указать настройки подключения к БД.

Для Oracle SQL

```
url = jdbc:oracle:thin:@ip:port/dbName
#defaultSchemaName=
```

В параметре defaultSchemaName указывается имя рабочей схемы, например sbp. Если данная строка закомментирована с помощью символа #, то по умолчанию скрипт установки выполнит развертывание в схему public.

Сделайте текущей директорию liquibase.

Сделайте файл скрипта исполняемым с помощью команды:

```
chmod +x run.sh
```

Запустите скрипт командой:

```
./run.sh update
```

При запуске, скрипт потребует ввода имени пользователя - владельца рабочей схемы БД (либо иного пользователя БД, имеющего право записи в используемую схему) и его пароля.

Пример вывода скрипта установки

```
user@server liquibase]$ ./run.sh update
username: user
password: password
execute: update
Starting Liquibase at WEEKDAY, dd mm yyyy hh:mm:ss YEKT (version 3.6.3 built at yyyy
hh:mm:ss)
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.vmplugin.v7.Java7$1 (file:
/opt/sbp-connect-sngb-c2c/liquibase/lib/groovy-2.5.8.jar) to constructor java.lang
.invoke.MethodHandles$Lookup(java.lang.Class,int)
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy
.vmplugin.v7.Java7$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective
access operations
WARNING: All illegal access operations will be denied in a future release
Liquibase: Update has been successful.
```

Управление модулями СБП-Коннект

1. Управление модулем crypto-app

Запуск службы crypto-app

Выполните все необходимые настройки по настройке модуля, а затем запустите службу с помощью апплета Панели Управления Службы Панель Управления\Администрирование\Службы Нажмите правой кнопкой мыши по пункту SbpCryptoService в списке служб, в контекстном меню выберите Запуск

Также можно запустить сервис вручную с помощью .bat скрипта:

```
bat\startService.bat
```

Остановка службы crypto-app

Остановить службу crypto-app можно также при помощи апплета Панели Управления Службы Панель Управления\Администрирование\Службы Нажмите правой кнопкой мыши по пункту SbpCryptoService в списке служб, в контекстном меню выберите Остановка

Кроме этого, для остановки сервиса можно воспользоваться скриптом:

```
bat\stopService.bat
```

Просмотр статуса модуля

Состояние модуля crypto-app можно увидеть с помощью апплета Службы Панели Управления, в столбце Состояние

Для проверки состояния crypto-app с помощью командной строки, выполните:

```
sc query "SbpCryptoService"
```

Автозапуск модуля при загрузки Windows

Кроме апплета "Службы", автозапуск модуля можно настроить с помощью командной строки

```
sc config "SbpCryptoService" start= auto
```

Если возникли проблемы при запуске, информацию можно посмотреть в директории tmp\err_{id}

2. Управление модулем client-app

За работу модуля client-app отвечает служба sbp-connect-client-app.

Для запуска/остановки sbp-connect-client-app и проверки, используются следующие команды, не из под root пользователя:

Запуск службы client-app

```
sudo service service sbp-connect-domrf-client-app start
```

Остановка службы client-app

```
sudo service sbp-connect-domrf-client-app stop
```

Проверка состояния client-app

```
sudo service sbp-connect-domrf-client-app status
```

Включение автозапуска службы при загрузке операционной системы

```
sudo chkconfig service sbp-connect-domrf-client-app on
```

3. Управление модулем dashboard-app

За работу модуля dashboard-app отвечает служба sbp-connect-dashboard.

Для запуска/остановки sbp-connect-dashboard и проверки используются следующие команды, не из под root пользователя:

Запуск службы dashboard-app

```
sudo systemctl start sbp-connect-domrf-dashboard
```

Остановка службы dashboard-app

```
sudo systemctl stop sbp-connect-domrf-dashboard
```

Проверка состояния client-app

```
sudo systemctl status sbp-connect-domrf-dashboard
```

Right Line контакты

Телефон: +7 (499) 517-96-95

Email: support@rtln.ru

Адрес: 117105, г. Москва, ул. Варшавское шоссе, д. 26, офис 209