Right Line ACS. Руководство по администрированию для АО БАНК «СНГБ»

Оглавление

Введение

О технологии 3D Secure

Технология 3D Secure была создана для увеличения доверия владельцев карт к онлайн-транзакциям, таким как платежи или покупки. Технология стандартизует взаимодействие между всеми участниками платежа, позволяя закрепить ответственность за эмитентом или эквайером. Для защиты от мошеннических операций 3D Secure добавляет ещё один шаг аутентификации при обработке онлайн-транзакций. Этот добавочный шаг позволяет торговым точкам и банкам дополнительно убедиться, что операцию совершает именно держатель карты. Таким образом, 3-D Secure (3DS) предназначена для защиты мерчантов, эмитентов и держателей карт от мошенничества с платежами без присутствия карты (транзакции card not present или CNP), путем проверки подлинности держателей карт, и возможностью переноса ответственности за мошенничество эквайером на эмитента, так как именно эмитент аутентифицирует держателя карты. В противном случае, без применения технологии ответственность за операцию несёт эквайер, и в случае мошеннической транзакции именно эквайер должен вернуть деньги владельцу карты.

3DS v1

Первая версия 3DS изначально была разработана компанией Visa в начале двухтысячных и была названа Verified by Visa. Во время использования технологии 3D Secure 1 система отображает всплывающее окно или встроенный фрейм, требуя от пользователя ввода пароля (постоянного или разового), чтобы банк мог аутентифицировать пользователя.

Общая схема взаимодействия 3DS v1

[Общая схема взаимодействия 3DS v1] | *imgs/3DSv1_common.png*

1. Покупатель вводит параметры карты на сайте торговца.
2. Если сайт торговца поддерживает 3D Secure, он запускает плагин торговца (MPI). MPI взаимодействует с Payment System Directory Server (DS), расположенным в домене взаимодействия. Для проверки регистрации карты в 3-D Secure, в DS отправляется запрос на проверку регистрации (`VEReq`), который содержит номер карты (PAN).
3. DS определяет Access Control Server (ACS) издателя карты на основе PAN и связывается с ним, чтобы проверить зарегистрирована ли карта в 3D Secure.
4. ACS передаёт DS ответ, в котором указывается, зарегистрирована ли карта в платёжной системе.
5. DS отвечает MPI сообщением Verifying Enrolment Response (`VERes`), передавая MPI зарегистрирована ли карта в 3D Secure. Если карта вовлечена, то сообщение `VERes` включает URL компоненты ACS издателя карты.
6. MPI торговца перенаправляет страницу браузера по адресу, добавляя к POST-request подписанный запрос Payer Authentication Request (`PAReq`), который включает PAN и другие данные по транзакции.
7. Владелец карты проходит аутентификацию через ACS. В зависимости от поддерживаемого способа аутентификации это может быть одноразовый пароль, постоянный пароль, вход в Интернет банк, и так далее. Сообщение персональной гарантии (personal assurance message (PAM)), выбранное владельцем карты во время

регистрации, может отображаться на этой странице, если эмитент поддерживает такую возможность.

8. Когда владелец карты завершает заполнение, ACS возвращает ответ MPI, добавляя в него сообщение Payer Authentication Response (`PARes`). Когда владелец карты заполнил форму аутентификации, как указано выше, ACS перенаправляет введённые данные обратно в MPI, добавляя сообщение ответа аутентификации плательщика (`PARes`).
9. Браузер передаёт ответ `PARes` в MPI. `PARes` содержит статус транзакции, который отображает действительно ли покупатель успешно аутентифицировался через 3D Secure. В зависимости от правил платёжной системы и статуса транзакции, указанном в сообщении `PARes`, торговец может инициировать запрос авторизации платежа.

Недостатки 3DS v1

Несмотря на основное преимущество 3D Secure – перенос ответственности с торговца на эмитента, многие торговцы не используют 3D Secure, так как выгода от переноса рисков не компенсирует потери из-за недостатков 3DS v1:

- Рабочий процесс может отталкивать покупателя наличием лишних действий, что приводит к проблеме заброшенных тележок.
- Страница ACS может быть несовместима с мобильными устройствами, что также приводит к отказу от покупок.
- Отсутствие бесшовной интеграции с современными платежными средствами. Например, кошельками.
- Ограниченный набор возможных методов аутентификации, некоторые из которых устарели и небезопасны.

3DS v2

В октябре 2016 года была опубликована новая версия технологии — 3-D Secure 2.0. Разработкой новой версии занимается организация EMVCo, работу которой контролируют 6 участников: American Express, Discover, JCB, MasterCard, UnionPay и Visa. 3DS v2 обладает рядом преимуществ по сравнению с 3DS v1:

- Возможность встраивания в приложения для мобильных устройств, автомобилей, телевизоров, холодильников и других бытовых устройств.
- Аутентификация без вовлечения держателя карты, за счет оценки и интеллектуального анализа рисков на стороне эмитента.
- Результат аутентификации передается торговцу через DS и 3DS Server по отдельному каналу, что делает процесс более безопасным.
- Возможность использования современных и более надежных способов аутентификации: отпечаток пальца, распознавание лица или голоса, двух-факторная аутентификация и другие.
- Возможность аутентификации без участия торговца и возможность отложенной аутентификации.
- Привязка карты и другие неплатежные операции без проведения платежа.

Вторая версия спецификации активно развивается и на текущий момент актуальной является версия 2.2.0 от 13 декабря 2018. Поддержка первой версии должна прекратиться в начале 2020 года.

Общая схема взаимодействия 3DS v2

При использовании 3-D Secure v2 обработка возможна как без взаимодействия с клиентом, так и с взаимодействием. Первый способ называется Frictionless Flow и не требует дальнейшего взаимодействия с держателем карты для успешной аутентификации. Если компонент ACS определяет, что требуется дальнейшее прямое взаимодействие с держателем карты для завершения аутентификации, то применяется сценарий Challenge Flow. Например, проверка (Challenge) может оказаться необходимой, если операция будет считаться высокорискованной или требующей более надежного уровня аутентификации из-за требований законодательств определенных стран.

Frictionless Flow

[3DSv2 frictionless] | *imgs/3DSv2_frictionless.png*

Старт: Держатель карты инициирует операцию на клиентском устройстве.

1. Компонентами среды 3DS Requestor Environment собирается необходимая информация и предоставляется 3DS Server'у для включения в сообщение `AReq`. Как предоставлена эта информация, и от какого компонента она поступила, зависит от следующего
    - Device Channel (канал) – App-based (на основе загружаемого приложения предприятия торговли/услуг) или Browser-based (на основе браузера в клиентском устройстве).
    - Message Category (категория сообщения) – Payment (платежное) или Non-Payment (неплатежное).
    - Реализации компонента 3DS Requestor.
2. Используя данные, полученные от 3DS Requestor Environment, 3DS Server создает и отправляет сообщение `AReq` компоненту DS, который затем направляет сообщение в ACS.
3. В ответ на сообщение `AReq` компонент ACS возвращает сообщение `ARes` компоненту DS, который затем передает его обратно 3DS Server'у. Перед возвращением ответа `ARes` компонент ACS оценивает данные, представленные в сообщении `AReq`. В сценарии Frictionless Flow компонент ACS определяет, что дополнительного взаимодействия с держателем карты для завершения аутентификации не требуется.
4. 3DS Server передаёт результат сообщения `ARes` компоненту 3DS Requestor Environment. 3DS Requestor определяет, каким образом реализуется взаимодействие между этими компонентами.
5. ТСП направляет авторизационный запрос банку эквайреру.
6. Эквайрер может обработать авторизацию, запросив об этом эмитента и вернув после этого результат такой обработки в ТСП.

Challenge Flow

[3DSv2 challenge] | *imgs/3DSv2_challenge.png*

Старт: Держатель карты инициирует операцию на клиентском устройстве.

1. Компонентами среды 3DS Requestor Environment собирается необходимая информация и предоставляется компоненту 3DS Server для включения в сообщение AReq. Как предоставлена эта информация, и от какого компонента она поступила, зависит от следующего:
    - Device Channel (канал) – App-based (на основе загружаемого приложения предприятия торговли/услуг) или Browser-based (на основе браузера в клиентском устройстве).
    - Message Category (категория сообщения) – Payment (платежное) или Non-Payment (неплатежное).
    - Реализации компонента 3DS Requestor.
2. Используя информацию, предоставленную держателем карты и данные, полученные в 3DS Requestor Environment, компонент 3DS Server создает и отправляет сообщение AReq компоненту DS, который затем направляет сообщение в соответствующий компонент ACS.
3. В ответ на сообщение AReq компонент ACS возвращает сообщение ARes, компоненту DS, который затем передает его обратно 3DS Server'у. Перед возвращением ответа ARes компонент ACS оценивает данные, представленные в сообщении AReq. В сценарии Challenge Flow компонент ACS определяет, что для завершения аутентификации требуется дополнительное взаимодействие с держателем карты.
4. 3DS Server передаёт результат сообщения ARes компоненту 3DS Requestor Environment. 3DS Requestor определяет, каким образом реализуется взаимодействие между этими компонентами.
5. 3DS Client инициирует сообщение CReq используя информацию, которая была получена в сообщении ARes. Способ, при помощи которого это происходит, зависит от модели:
    - Модель, ориентированная на приложение (app-based). Сообщение CReq формируется компонентом 3DS SDK и перенаправляется методом post по адресу ACS URL, полученному в сообщении ARes.
    - Модель, ориентированная на браузер (browser-based). Сообщение CReq формируется компонентом 3DS Server и методом post через браузер держателя карты перенаправляется компонентом 3DS Requestor по адресу ACS URL, полученному в сообщении ARes.
6. ACS принимает сообщение CReq и взаимодействует с компонентом 3DS Client, чтобы поддержать интерфейс с держателем карты. Способ, при помощи которого это происходит, зависит от модели:
    - Модель, ориентированная на приложение (app-based). Компонент ACS использует пары сообщений CReq и CRes для выполнения Проверки (Challenge). В ответ на полученное сообщение CReq компонент ACS формирует сообщение CRes, которое запрашивает держателя карты ввести данные для аутентификации, и направляется в компонент 3DS SDK.
    - Модель, ориентированная на браузер (browser-based). Компонент ACS отправляет пользовательский интерфейс для аутентификации держателя карты в браузер его клиентского устройства. Держатель карты явно (заполняет поля) или неявно (out-of-

band метод, например, идентификацией через мобильное приложение) вовлекается в процесс аутентификации. При этом также явно или неявно передаются данные, которые должны быть проверены компонентом ACS. В ответ на полученное сообщение `CReq` компонент ACS формирует сообщение `CRes` и отправляет его 3DS Server'у, чтобы сообщить результат аутентификации. Примечание. Для модели app-based шаги 5 и 6, описанные выше, могут повторяться до тех пор, пока в компоненте ACS не появится определенное решение в отношении результатов аутентификации.

7. ACS отправляет сообщение `RReq` компоненту DS, которое включает AV (Authentication Value), а компонент DS, в свою очередь, затем отправляет это сообщение соответствующему компоненту 3DS Server, используя адрес 3DS Server URL, полученный в сообщении AReq.
8. Компонент 3DS Server получает сообщение `RReq` и отвечает на него сообщением `RRes` компоненту DS, который затем направляет это сообщение в соответствующий компонент ACS.
9. ТСП направляет авторизационный запрос банку эквайреру.
10. Эквайрер может обработать авторизацию, запросив об этом эмитента и вернув после этого результат такой обработки в ТСП.

Архитектура ACS

ACS осуществляет аутентификацию держателя карты, реализуя взаимодействие в рамках спецификаций 3-D Secure (1.0.2 и 2.1.0). Для второй версии поддерживаются способы аутентификации Frictionless и Challenge, для первой версии поддерживается только Challenge. Способ аутентификации Frictionless может быть определен на основе белого списка мерчантов, более детально в настройках скоринга. Если условия для проведения Frictionless не выполнены, аутентификации будет проведена через Challenge. В качестве Challenge данная версия поддерживает аутентификацию только через OTP (one time password). При совершении платежа, для аутентификации, держателю карты предлагается ввести одноразовый код, который отсылается ему на привязанное устройство (push или смс). Вводя такой код, держатель карты идентифицирует себя и подтверждает платеж. В независимости от способа аутентификации, по ее результатам формируется специальная криптограмма (AV), в которой содержатся данные о прохождении аутентификации.

На схеме ниже отображены все варианты взаимодействия с ACS и его окружение. Все входящие запросы направляются в ACS через балансировщик, который балансирует трафик между всеми нодами ACS. Балансировка может осуществляться любым способом, т.к. все запросы stateless и не требуют передачи какого-либо состояния. Также, на балансировщике терминируется внешний SSL (DS и клиентский), а взаимодействие с нодами ACS шифруется с помощью внутреннего SSL. Исходящие запросы отправляются напрямую в DS и шифруются SSL для DS.

[acs schema sngb] | *imgs/acs-schema-sngb.png*

Приложение реализовано в виде единого исполняемого jar файла, включающего в себя все необходимые библиотеки. Для запуска приложения необходима среда исполнения Java и СУБД Oracle, с необходимой структурой таблиц и конфигурационный файл.

Также, в линейке решений Right Line есть и другие компоненты 3-D Secure окружения, такие как E-comm Gateway, 3DS Server, 3DS SDK. На схеме ниже отображены все компоненты 3-D Secure и компоненты Right Line обозначены утолщённой рамкой.

[3DS RL] | *imgs/3DS_RL.png*

Примечание: модули из домена платежных систем (на схеме выделены жёлтым цветом) будут отсутствовать в рамках обработки On-Us операций.

Состав дистрибутива

Дистрибутив включает в себя:

- Пакет для установки приложение ACS, поставляется в виде rpm пакета (`acs-sngb-<version>.x86_64.rpm`)
- Руководство по администрированию (данный документ)

Установка и настройка

Минимальные системные требования

Для развертывания системы потребуется как минимум одна виртуальная (или физическая) машина. Минимально необходимая конфигурация машины: 2 core x 2GHz CPU, 4Gb RAM, 60 Gb свободного места. В качестве операционной системы может использоваться 64-битная ОС Linux RedHat/Centos с предустановленным ПО Java 11 (OpenJDK JRE или OracleJRE). Для построения отказоустойчивого балансируемого кластера системы потребуется как минимум две виртуальные (или физические) машины и балансировщик. В качестве СУБД используется Oracle версии не ниже 12c.

Установка

Утановка rpm пакета

Система поставляется в виде rpm пакета `acs-sngb-<version>.x86_64.rpm`, установка осуществляется согласно инструкции менеджера пакетов RPM.

Команда для установки:

```
sudo rpm -ivh acs-sngb-<version>.x86_64.rpm
```

В результате установки в системе создастся сервис с именем `acs-sngb`.

Схема БД

Для развертывания системы необходима предустановленная СУБД Oracle версии не ниже

12c. Ниже приводится скрипт для создания пользователя и схемы БД:

```
create user <user> identified by <password>;
grant all privileges to <user>;
grant select on v_$session to <user>;
grant select_catalog_role to <user>;
grant select any dictionary to <user>;
```

где <user> - имя пользователя БД, а <password> - пароль пользователя БД.

Дистрибутив поставляется вместе инструментом liquibase, который обеспечивает создание и обновление схемы БД. При установке rpm в директории `/opt/acs-sngb/config/liquibase` размещается все необходимое для работы liquibase и начальной инициализации схемы БД. Также, при обновлении rpm будут доставляться инкрементальные изменения схеме БД. Перед запуском liquibase необходимо указать реквизиты доступа к БД в файле настроек `/opt/acs-sngb/config/liquibase/liquibase.properties`. Запуск осуществляется из директории `/opt/acs-sngb/config/liquibase` с помощью скрипта `update.sh`. Выполнение скрипта обеспечит начальную инициализацию схемы или обновит схему до последней версии.

Настройка

После установки необходимо сконфигурировать приложение и подготовить его к первому запуску. В директории `/opt/acs-sngb/config` должен располагаться конфигурационный файл `application.yml`. В данном файле крайне важно сохранять формат отступов. Если формат не будет сохранен, это может привести к аварийной остановке или некорректной работе приложения. Пример файла с описанием параметров:

```
server: # настройки сервера
  port: 8080 # порт для основной группы эндпоинтов, по умолчанию 8080
  ssl: # настройка сертификатов
    key-store-type: PKCS12 # тип контейнера допустимые параметры PKCS12 и JKS
    key-store: ./config/test.p12 # расположение контейнера с ключевой парой
    key-store-password: acs # пароль от ключевого контейнера
    key-alias: test.local # алиас (идентификатор) ключевой пары
    key-password: acs # пароль от ключевой пары
spring: # глобальные настройки Spring
  datasource: # настройки источника данных (базы)
    username: test # логин для подключения к базе
    password: test # пароль для подключения к базе
    hikari: # настройки пула соединений Hikari, все возможные ключи
      # конфигурации описаны тут
      # https://github.com/brettwooldridge/HikariCP#configuration-knobs-baby
      # тут указываются наименование ключей и значения
management:
  server:
    port: 8081 # порт для служебной группы эндпоинтов, по умолчанию публикуется
      # на порту группы основных эндпоинтов
logging: # настройки подсистемы логирования
```

```
  config: config/logback.groovy # расположение файла конфигурации логгера
acs: # настройки acs
  crypto: # настройка криптографии
    pathToKeystore: ./config/keystore.jks # расположение ключевого контейнера.
      # Содержит ключи, необходимые для взаимодействия с компонентами 3DS.
    pathToTrustKeystore: ./config/trust_keystore.jks # расположение контейнера
      # с доверенными сертификатами. В нем должны храниться исключительно
      # корневые сертификаты от платежных систем.
    acsKeyStorePassword: acs # пароль от ключевого контейнера
    acsTrustedKeyStorePassword: acs # пароль от хранилища доверенных сертификатов
    visaCAVVKeyIndicator: "01" # CAVVKeyIndicator для 3DS v1.0
    mcBinKeyIdentifier: "1" #  BinKeyIdentifier для 3DS v1.0
    acsEmpSharedKeys: # Ключ, который должен быть предоставлен для
      # проверки/вычисления authenticationValue в ACS и платежном шлюзе
      MASTER_CARD: B039878C1F96D212F509B2DC4CC8CD1B # значение для MasterCard
      VISA: B039878C1F96D212F509B2DC4CC8CD1B # значение для Visa
      MIR: B039878C1F96D212F509B2DC4CC8CD1B # значение для Mir
    keys: # настроки для ключей внутри ключевого контейнера
      v1: # для 3DS v1.0
        MASTER_CARD: # для MasterCard
          keyAlias: v1_mc_acs # алиас (идентификатор) ключевой пары
          keyPassword: acs # пароль от ключевой пары
        VISA: # для Visa
          keyAlias: v1_visa_acs # алиас (идентификатор) ключевой пары
          keyPassword: acs # пароль от ключевой пары
        MIR: # для Mir
          keyAlias: v1_mir_acs # алиас (идентификатор) ключевой пары
          keyPassword: acs # пароль от ключевой пары
      v2: # для 3DS v2.0
        MASTER_CARD: # для MasterCard
          dsKey: # ключ для взаимодействия с DS. Клиентский ключ. Используется
            # для отправки RReq.
            keyAlias: v2_mc_acs_client # алиас (идентификатор) ключевой пары
            keyPassword: acs # пароль от ключевой пары
          scKey: # Ключ для шифрования Singed content в котором содержится
            # адрес для прохождения челенжа и исходник для генерации cek.
            # Используется только для application flow
            keyAlias: v2_mc_acs_sdk_sign # алиас (идентификатор) ключевой пары
            keyPassword: acs # пароль от ключевой пары
        VISA: # для Visa
          dsKey: # ключ для взаимодействия с DS. Клиентский ключ. Используется
            # для отправки RReq.
            keyAlias: v2_visa_acs_client # алиас (идентификатор) ключевой пары
            keyPassword: acs # пароль от ключевой пары
          scKey: # Ключ для шифрования Singed content в котором содержится
            # адрес для прохождения челенжа и исходник для генерации cek.
            # Используется только для application flow
            keyAlias: v2_visa_acs_sdk_sign # алиас (идентификатор) ключевой пары
            keyPassword: acs # пароль от ключевой пары
        MIR: # для Mir
          dsKey: # ключ для взаимодействия с DS. Клиентский ключ. Используется
```

```
            # для отправки RReq.
            keyAlias: v2_mir_acs_client # алиас (идентификатор) ключевой пары
            keyPassword: acs # пароль от ключевой пары
          scKey: # Ключ для шифрования Singed content в котором содержится
            # адрес для прохождения челенжа и исходник для генерации cek.
            # Используется только для application flow
            keyAlias: v2_mir_acs_sdk_sign # алиас (идентификатор) ключевой пары
            keyPassword: acs # пароль от ключевой пары
  http: # настройки пула подключений к DS. Для подключения к каждому DS используется
отдельный пулл.
    httpClientPoolSize: 10 # размер пула
    httpClientKeepAlive: 120 # таймаут в мс отправки keepalive сообщения.
      # Не рекомендуется менять.
    connectionRequestTimeout: 9000 # таймаут в мс получения соединения из пула.
      # Крайне не рекомендуется менять, поскольку значение связано со
      # спецификацией.
    connectTimeout: 9000 # таймаут в мс на установку соединения. Крайне не
      # рекомендуется менять, поскольку значение связано со спецификацией.
    readTimeout: 5000 # таймаут на чтение из сокета. Крайне не рекомендуется
      # менять, поскольку значение связано со спецификацией.
  settings: # общие настройки
    acsReferenceNumber: 3DS_LOA_ACS_PPFU_020100_00009 # серийный номер ACS
      # может меняться только для целей прохождения тестирования с платежными
      # системами. В проде должно отсутствовать.
    challengePrefixUrl: https://test.rtln.ru # префикс URL адреса на который
      # будет отправлен CReq. Может содержать адрес порта
    acsMaximumChallenges: 3 # Максимальное количество попыток прохождения
      # челенжа (ввода верного OTP)
    cancelOrFailSubmitAuthenticationLabel: Continue # Надпись для кнопки в
      # приложении при отмене или ошибке прохождения челенжа. Используется
      # только в application flow.
    timeoutRReqSenderThreadPoolSize: 20 # Размер пула для отправки сообщений
      # ошибки, если не удалось отправить RReq
    nodeStateCacheExpirationTimeout: 300000 # Таймаут в миллисекундах для
      # перечитывания файла статуса. Не рекомендуется ставить значение ниже 60000.
    operatorIds: # идентификатор ACS в различных платежных системах
      MASTER_CARD: acsOperatorUL # для MasterCard
      VISA: 2201380209 # для Visa
      MIR: 2201380209 # для Mir
rtln: # настройка библиотек производства Right Line
  hsm: # Настройки библиотек для работы с HSM
    payshield9000: # Настройки библиотек для работы с HSM Pay Shield 9000
      primaryHsmGroup: # настройки основной группы подключений HSM. Должен
        # содержать хотя бы один элемент перечисления.
        - # элемент перечисления. Может быть несколько в одной группе.
          host: 10.10.0.10 # Адрес HSM
          port: 15000 # Порт подключения
          socketTimeout: 1000 # таймаут подключения
          numberOfSockets: 5 # количество открываемых сокетов. Сокеты держатся
            # все время работы приложения и восстанавливаются автоматически,
            # если произошел сбой и возможно восстановление.
```

```
          cvk: "00000000000000000000000000000000000000000000" # Ключ в виде
            # HEX представления массива байт. Используется для вычисления
            # CAV. Может быть как keyblock так и variant
      fallbackHsmGroup: # настройки резервной группы подключений HSM. Должен
        # содержать хотя бы один элемент перечисления.
        - # элемент перечисления. Может быть несколько в одной группе.
          host: 10.10.0.11 # Адрес HSM
          port: 15000 # Порт подключения
          socketTimeout: 1000 # таймаут подключения
          numberOfSockets: 5 # количество открываемых сокетов. Сокеты держатся
            # все время работы приложения и восстанавливаются автоматически,
            # если произошел сбой и возможно восстановление.
          cvk: "00000000000000000000000000000000000000000000" # Ключ в виде
            # HEX представления массива байт. Используется для вычисления
            # CAV. Может быть как keyblock так и variant
      sendCommandExecutorThreadPoolSize: 5 # размер пула для одновременной
        # отправки команд в HSM
      sendRetryNumber: 2 # количество повторных попыток отправки в случае если
        # попали на отказавший HSM. Рекомендуется ставить пропорционально
        # количеству элементов в основной и резервной группе HSM
      restoreBrokenSocketsPeriod: 60000 # период запуска задачи восстановления
        # подключений к HSM. Крайне не рекомендуется ставить меньше 60000
      echoSendPeriod: 60000 # период запуска задачи опроса HSM, чтобы не
        # происходило отключение сокетов. Крайне не рекомендуется ставить меньше 60000
sngb: # настройки специфичные для банка SNGB
  properties: # общие настройки
    defaultMinFrictionlessLimit: 0 # нижняя граница лимита по умолчанию в
      # рамках которого транзакции будут проходить как frictionless
    defaultMaxFrictionlessLimit: 0 # верхняя граница лимита по умолчанию в
      # рамках которого транзакции будут проходить как frictionless
    defaultCurrencyCode: 643 # код валюты лимита по умолчанию
    preferredInterface: NATIVE_UI # предпочитаемый интерфейс для взаимодействия
      # с мобильным приложением. Не рекомендуется менять.
    smsSendUrl: https://localhost:8080/send_sms # адрес сервиса отправки смс
    enableRangeCacheUpdateJob: true # флаг включения задачи обновления
      # локального кэша с интервалами карт банка
    updateRangeCachePeriod: 1800000 # таймаут в миллисекундах задачи на
      # обновление локального кэша с интервалами карт банка.
    smsServiceCertKeystorePath: ./config/sngb_keystore.jks # путь до хранилища
      # доверенных сертификатов SNGB
    smsServiceCertKeystorePass: test # пароль от доверенного хранилища
    maxNumberOtpSmsPerChallenge: 3 # Максимальное количество повторов отправки
      # смс с otp
    formCheckPhrases: # Фразы для формы челенжа
      MASTER_CARD: Пожалуйста, введите одноразовый пароль для MASTER CARD # для
MasterCard
      VISA: Пожалуйста, введите одноразовый пароль для VISA # для Visa
      MIR: Пожалуйста, введите одноразовый пароль для MirAccept # для Mir
    formResendOtpTimeoutInMillis: 30000 # количество миллисекунд, на которое
      # будет блокироваться бэк для повторной отправки otp
  app-messages: # Сообщения для UI мобильного приложения при прохождении
```

```
    # челенжа. Подробнее можно прочитать в спеке 3DS v2.0 EMV ® 3-D Secure Protocol
and Core Functions Specification v2.1.0 доступной на сайте EMVCo
https://www.emvco.com/
    challengeInfoHeader: Подтверждение платежа # заголовок
    challengeInfoLabel: Код из СМС # лэйбел информации
    # шаблон фразы описывающей информацию о платеже. Не рекомендуется менять.
    challengeInfoText: "%s %s\nМагазин\t%s\nНомер карты\t%s\nКод отправлен на
номер\t%s\n"
    challengeInfoTextIndicator: N # индикатор использования дополнительной
      # подсветки при ошибке
    expandInfoLabel: Дополнительная информация # лэйбэл дополнительной информации
    expandInfoText: Некая дополнительная информация # дополнительная информация
    resendInformationLabel: Выслать код повторно # лэйбел повторной отправки OTP
    submitAuthenticationLabel: Отправить # сообщение для отправки результата
    whyInfoLabel: Не приходит СМС # лэйбел дополнительной информации
    # Дополнительная информация
    whyInfoText: "Это могло произойти, если:\n\nвы изменили номер телефона и не
сообщили его банку."
    issuerImage: # расположения логотипа банка эмитента
      MEDIUM: https://localhost:8080/images/sngb/logo # в среднем качестве
      HIGH: https://localhost:8080/images/sngb/logo # в высоком качестве
      EXTRA_HIGH: https://localhost:8080/images/sngb/logo # в очень высоком качестве
    psImage: # расположения логотипа платежной системы
      MASTER_CARD: # для MasterCard
        MEDIUM: https://localhost:8080/images/mc/logo # в среднем качестве
        HIGH: https://localhost:8080/images/mc/logo # в высоком качестве
        EXTRA_HIGH: https://localhost:8080/images/mc/logo # в очень высоком качестве
      VISA: # для Visa
        MEDIUM: https://localhost:8080/images/visa/logo # в среднем качестве
        HIGH: https://localhost:8080/images/visa/logo # в высоком качестве
        EXTRA_HIGH: https://localhost:8080/images/visa/logo # в очень высоком качестве
      MIR: # для Mir
        MEDIUM: https://localhost:8080/images/mir/logo # в среднем качестве
        HIGH: https://localhost:8080/images/mir/logo # в высоком качестве
        EXTRA_HIGH: https://localhost:8080/images/mir/logo # в очень высоком качестве
```

ВАЖНО

Размер пула потоков для отправки `RReq` должен соотноситься с максимальным количеством доступных соединений с БД. Поэтому рекомендуется, чтобы значение параметра `acs.settings.timeoutRReqSenderThreadPoolSize` не превышало значения параметра `spring.datasource.hikari.maximumPoolSize`.

Пароли и ключи должны быть указаны в конфигурационном файле в зашифрованном виде. Для шифрования параметров необходимо:

1. Получить от вендора по защищенному каналу мастер пароль
2. Зашифровать с помощью утилиты http://www.jasypt.org/cli.html значение на предоставленный пароль
3. Заменить необходимый параметр на значение ENC(<зашифрованное-значение>)

Обновление

Новые версии поставляется в виде rpm пакета `acs-sngb-<version>.x86_64.rpm`, вместе с актуальной документацией и списком изменений (release-notes.pdf), который, при необходимости, будет содержать требования по миграции на новую версию.

Необходимо выполнить обновление rpm согласно инструкции менеджера пакетов RPM.

Команда для обновления rpm

```
sudo rpm -Uvh acs-sngb-<version>.x86_64.rpm
```

Далее, необходимо обновить структуру БД с помощью Liquibase. Перед запуском Liquibase необходимо указать реквизиты доступа к БД в файле настроек `/opt/acs-sngb/config/liquibase/liquibase.properties` и затем, из директории `/opt/acs-sngb/config/liquibase`, запустить скрипт `update.sh`. В случае кластера, обновление БД необходимо проводить только один раз (с одной ноды).

Если в release-notes.pdf присутствуют требования по миграции на новую версию, то их также необходимо выполнить.

После всех манипуляций необходимо перезапустить сервис.

Перезапуск

```
sudo systemctl restart acs-sngb
```

Управление

Производится через утилиту управления systemd - systemctl

Запуск

```
sudo systemctl start acs-sngb
```

Остановка

```
sudo systemctl stop acs-sngb
```

Проверка статуса

```
sudo systemctl status acs-sngb
```

Включение автоматического запуска при старте сервера

```
sudo systemctl enable /opt/acs-sngb/acs-sngb.service
```

Настройка сетевой инфраструктуры

Приложение публикует несколько эндпоинтов, которые в рамках приложения делятся на две группы: служебные и основные. К группе основных относятся эндпоинты для держателей карт, для DS платежных систем и для внутренних систем Банка. Каждая группа эндпоинтов, через конфигурационный файл, может быть опубликована на отдельном порту:

```
server:
  port: 8080 # порт для основной группы эндпоинтов, по умолчанию 8080
management:
  server:
    port: 8081 # порт для служебной группы эндпоинтов, по умолчанию публикуется на
порту группы основных эндпоинтов
```

Эндпоинты основной группы не могут быть разнесены по разным портам. При необходимости это может быть реализовано за счет настройки компонентов сетевой инфраструктуры. Ниже приведен возможный вариант настройки на базе Nginx.

Балансировка нагрузки и проверка статуса

```
# Ноды кластера ACS
upstream backend {
    server acs1.private.sngb.ru;
    server acs2.private.sngb.ru;
}

# Правила для проверки активности ноды кластера
match http_ok {
    expect ~ "200 OK"; # пока получаем 200 OK, нода считается активной
}
```

Настройки внутренних эндпоинтов

```
server {
        ssl on;
        listen 1443 ssl;
        server_name acs.sngb.ru;

        ssl_protocols TLSv1.2;
        ssl_certificate /etc/nginx/ssl/sngb-lb.cer;
        ssl_certificate_key /etc/nginx/ssl/sngb-lb.key;

        charset UTF-8;

        location ~ (sngb/*|/acs/(v2.1.0|v1.0.2)/(mir|mc|visa)/check_av) {
                proxy_http_version 1.1;
                proxy_pass https://backend:8080;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                health_check match=http port=8081 uri=/actuator/health;
        }
}
```

Настройки эндпоинтов для DS платежных систем

```
# MIR endpoints
server {
        ssl on;
        listen 2443 ssl;
        server_name acs.sngb.ru;

        ssl_protocols TLSv1.2;
        ssl_certificate /etc/nginx/ssl/sngb-mir.cer;
        ssl_certificate_key /etc/nginx/ssl/sngb-mir.key;

        ssl_verify_client optional;
        ssl_verify_depth 5;
        ssl_client_certificate /etc/nginx/ssl/ca-mir.cer;

        charset UTF-8;

        location ~ /acs/(v2.1.0|v1.0.2)/mir/(authentication|challenge/start) {
                proxy_http_version 1.1;
                proxy_pass https://backend:8080;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                health_check match=http port=8081 uri=/actuator/health;
        }
}
```

```
# Visa endpoints
server {
        ssl on;
        listen 4443 ssl;
        server_name acs.sngb.ru;

        ssl_protocols TLSv1.2;
        ssl_certificate /etc/nginx/ssl/sngb-visa.cer;
        ssl_certificate_key /etc/nginx/ssl/sngb-visa.key;

        ssl_verify_client optional;
        ssl_verify_depth 5;
        ssl_client_certificate /etc/nginx/ssl/ca-visa.cer;

        charset UTF-8;

        location ~ /acs/(v2.1.0|v1.0.2)/visa/authentication {
                proxy_http_version 1.1;
                proxy_pass https://backend:8080;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                health_check match=http port=8081 uri=/actuator/health;
        }
}

# MasterCard endpoints
server {
        ssl on;
        listen 5443 ssl;
        server_name acs.sngb.ru;

        ssl_protocols TLSv1.2;
        ssl_certificate /etc/nginx/ssl/sngb-mc.cer;
        ssl_certificate_key /etc/nginx/ssl/sngb-mc.key;

        ssl_verify_client optional;
        ssl_verify_depth 5;
        ssl_client_certificate /etc/nginx/ssl/ca-mc.cer;

        charset UTF-8;

        location ~ /acs/(v2.1.0|v1.0.2)/mc/authentication {
                proxy_http_version 1.1;
                proxy_pass https://backend:8080;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                health_check match=http port=8081 uri=/actuator/health;
        }
```

```
}
```

Настройки эндпоинтов для держателей карт

```
server {
        ssl on;
        listen 443 ssl;
        server_name 3ds.sngb.ru;

        ssl_protocols TLSv1.2;
        ssl_certificate /etc/nginx/ssl/3ds.sngb.cer;
        ssl_certificate_key /etc/nginx/ssl/3ds.sngb.key;

        charset UTF-8;

        location ~
/acs/(v2.1.0|v1.0.2)/(mir|mc|visa)/((challenge|validation)/(start|finish|resend_otp))
{
                proxy_http_version 1.1;
                proxy_pass https://backend:8080;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                health_check match=http port=8081 uri=/actuator/health;
        }
}
```

Логирование

В приложении используется подсистема логирования logback. Для настройки параметров логирования используется отдельный конфигурационный файл, расположение которого указывается в общем конфигурационном файле с ключом `logging.config`. Через конфигурационный файл logback можно настроить 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. В примере конфигурационного файла используется Groovy DSL.

Алгоритм формирования AV

Общая часть для всех платежных систем. Исходная уникальная байтовая последовательность для транзакции (далее `atnSourceBytes`) рассчитывается следующим образом:

- Берется номер карты и добивается справа буквами `F` до размера 20 символов. К нему прибавляется `dsTransId` в верхнем регистре. После чего производится очистка от пробельных символов и символов `-`.

```
final var atnSourceData =(StringUtils.rightPad(pan, 20, "F") + dsTransId.toUpperCase(
)).replaceAll("\\s+", "").replaceAll("-", "");
```

- После чего производится вычисление `HMAC` по алгоритму `HmacSHA256`. В качестве секретного ключа используется `EmpSharedKey` из конфигурации для соответствующей платежной системы.

```
final var sha256HMAC = Mac.getInstance("HmacSHA256");
final var secretkey = new SecretKeySpec(Hex.decodeHex(cryptoProps.getAcsEmpSharedKeys
().get(ps).toCharArray()), "HmacSHA256");
sha256HMAC.init(secretkey);
final var atnSourceBytes = sha256HMAC.doFinal(Hex.decodeHex(atnSourceData.toCharArray
()));
```

Дальнейшие вычисления зависят от платежной системы.

Реализация для МИР и Visa

- Вычисляется `atn`.

```
var atnSource = StringUtils.leftPad(Hex.encodeHexString(atnSourceBytes).toUpperCase(),
16, "0");
final var charArrayResult = atnSource.toCharArray();
final var digitsPart = new StringBuilder();
final var lettersPart = new StringBuilder();
for (char c : charArrayResult) {
    final var number = Integer.valueOf(Character.toString(c), 16);
    if (number > 9) {
        lettersPart.append(number - 10);
    } else {
        digitsPart.append(number);
    }
}
atnSource = digitsPart.toString() + lettersPart.toString();
final var atn = atnSource.substring(0, 16);
```

- производится вычисление `cvp2` на HSM куда в качестве входных данных подается `pan`, `atn.substring(12, 16)` и комбинация из статуса транзакции `AUTHENTICATION_SUCCESSFUL ? "0" : "7"` + метода аутентификации(всегда `02` - SMS_OTP).
- производится полная сборка `AV`

```
final var rb = new StringBuilder();
rb.append(transactionStatus == AUTHENTICATION_SUCCESSFUL ? "00" : "07");
rb.append(authMethod == SMS_OTP ? "02" : "06"); // SMS_OTP or KBA(frictionless)
rb.append("01");
rb.append(cvp2);
rb.append(atn, 12, 16);
rb.append(atn);
rb.append(paymentSystem == MIR ? "2222222222" : "0000000000"); // Mir or Visa
final var av = Base64.encodeBase64String(Hex.decodeHex(rb.toString().toCharArray()));
```

Реализация для MasterCard

- производится полная сборка `AV`

```
final var arrayToConvert = Arrays.copyOfRange(atnSourceBytes, 0, 4);
final var encodedHexadecimal = Hex.encodeHexString(arrayToConvert).toUpperCase();
final var iavValue = "C604" + encodedHexadecimal + "00000000000000000000000000000000";
final var av = Base64.encodeBase64String(Hex.decodeHex(iavValue.toCharArray()));
```

Скоринг и определение способа аутентификации

Скоринг проводится в 3 этапа:

- Первый этап - проверка статуса карты
  1. Номер карты(pan) проверяется среди банковских диапазонов в представлении `V_CARD_RANGE`. Если не находится в ответ отдается `ARes` с кодом транзакции `U` и причиной `13`.
  2. Проверяется статус карты в соответствии с представлением `V_CARD_EC`. Ищется карту по пану
  3. Если не нашли возвращаем кодом транзакции `N` и причиной `06`
  4. В случае нахождения записи о карте:
     - Если `valid == N` берем `reason`, в котором ожидаем увидеть причину иначе отдаем `12` с кодом транзакции `R`
     - Если `auth_3ds == N` возвращаем код транзакции `A` и причину `13`
     - Если phone пустой возвращаем код транзакции `N` и причину `12`
     - Иначе переходим к следующему этапу проверки
- Второй этап - вхождение мерчанта в черный список для данной карты
  - Проверяется наличие записи в таблице BLACK_LIST для пары pan(aReq.acctNumber) и имя мерчанта(aReq.merchantName)

    - Если запись найдена, то возвращается код транзакции "U" с причиной "12"

Примечание : управлять черным списком можно через методы контроллера с общим префиксом /sngb/blacklist.

- Третий этап - определение сценария авторизации транзакции
    1. Если транзакция не платежная (aReq.purchaseAmount == null), то проверяется наличие мерчанта в таблице TRUSTED_MERCHANT по имени мерчанта из AReq(merchantName),
        - если мерчант не задан в таблице TRUSTED_MERCHANT, то производится взаимодействие по challenge сценарию.
        - если мерчант задан в таблице TRUSTED_MERCHANT, то
        - если NPA_FRICTIONLESS = N (или null), то производится взаимодействие по challenge сценарию.
        - если NPA_FRICTIONLESS = Y, то производится взаимодействие по frictionless сценарию
    2. Если мерчант находится в таблице `TRUSTED_MERCHANT` по имени мерчанта из `AReq(merchantName)`, то проверяется соответствие суммы транзакции и валюты на попадание в диапазон frictionless для данного мерчанта:
        - если диапазон для мерчанта не задан, то значения не проверяются.
        - если транзакция попадает в диапазон, то производится взаимодействие по frictionless сценарию
        - если транзакция не попадает в диапазон, то производится взаимодействие по challenge сценарию
    3. Если мерчант не находится в таблице `TRUSTED_MERCHANT` по имени мерчанта из `AReq(merchantName)`, проверяются диапазоны по умолчанию, которые задаются в файле конфигурации.
    4. Если сумма транзакции находится в заданном диапазоне, то производится взаимодействие по frictionless сценарию, иначе производится взаимодействие по challenge сценарию

Описание структкуры таблиц приложения

Таблица THREE_DS_MESSAGE

Таблица содержит сообщения 3DS 2.0 : `AReq`, `ARes`, `CReq`, `CRes`, `RReq`, `RRes`.

Поля:

- **ID** Уникальный идентификатор. Генерируется на основе последовательности `three_ds_message_seq`
- **ACCOUNT_NUMBER** Номер карты
- **ACS_INTERFACE** Тип взаимодействия с SDK при челенже. Допустимые значения(в

скобках цифровые значения, которые приходят в сообщении `ARes`):

  - NATIVE_UI (01)
  - HTML_UI (02)

- **ACS_TRANS_ID** Уникальный идентификатор сессии со стороны ACS.
- **AUTHENTICATION_TYPE** Тип аутентификации, который будет применен для прохождения челенжа. Допустимые значения (в скобках цифровые значения, которые отправляются в сообщении `ARes`):
  - STATIC (01)
  - DYNAMIC (02)
  - OOB (03)
- **BROWSER_LANGUAGE** Язык браузера держателя карты, который используется для 3DS аутентификации
- **CARDHOLDER_NAME** Имя держателя карты
- **CREATE_DATE** Время создания записи. Используется локальное время сервера ACS
- **DS_TRANS_ID** Уникальный идентификатор сессии со стороны DS.
- **DS_URL** URL на который будет отправлен `RReq` в случае челенжа
- **EXPIRY_DATE** Срок действия карты в формате `YYMM`
- **MERCHANT_COUNTRY_CODE** 2-буквенное обозначение страны мерчанта. В `AReq` предоставляется в виде кода `ISO 3166-1`
- **MERCHANT_NAME** Имя мерчанта.
- **MESSAGE_CATEGORY** Индикатор платежной/не платежной транзакции. Допустимые значения (в скобках цифровые значения, которые отправляются в `AReq` сообщении):
  - CAT_01_PA (01)
  - CAT_02_NPA (02)
- **MESSAGE_EXTENSION** Список расширений сообщения, сериализованных в `JSON`.
- **MESSAGE_TYPE** Тип сообщения. Допустимые значения:
  - AREQ
  - ARES
  - CREQ
  - CRES
  - RREQ
  - RREQ
  - RRES
- **NOTIFICATION_URL** FQDN URL на который будет отправлен `Final CRes` или `Error` сообщение
- **PAYMENT_SYSTEM** Платежная система, которая определена для данного сообщения на основе pan. Допустимые значения:

    - MASTER_CARD
    - VISA
    - MIR
- **PURCHASE_AMOUNT** Сумма покупки в минимальных единицах валюты (Например, для рубля это будет количество копеек)
- **PURCHASE_CURRENCY** Валюта покупки в виде 3-буквенного обозначения. В `AReq` предоставляется в виде трехзначного кода `ISO 4217`
- **PURCHASE_DATE** Дата совершения покупки
- **PURCHASE_EXPONENT** Степень 10, показывающая количество разрядов, отводимых под дробную часть валюты
- **SDK_TRANS_ID** Уникальный идентификатор сессии со стороны SDK.
- **THREE_DS_SERVER_TRANS_ID** Уникальный идентификатор транзакции со стороны 3DS Server по которому можно выбрать все сообщения цепочки (AReq, ARes, CReq, CRes, RReq, RRes).
- **THREE_DS_REQUESTOR_URL** FQDN URL 3DS Requestor'a (банка, IPSP или ТСП)
- **UI_TYPE** Вид челенжа SDK. Допустимые значения(в скобках цифровые значения, которые отправляются в сообщении `ARes`):
    - TEXT (01) - Челенж, который предполагает ввод некоторого текста (Например, цифр OTP)
    - SINGLE_SELECT (02) - Челенж, который предполагает выбор одного варианта из нескольких (radio button)
    - MULTI_SELECT (03) - Челенж, который предполагает выбор нескольких вариантов (check boxes)
    - OOB (04) - Челенж, который предполагает взаимодействие со внешней системой (например push уведомление в приложение по сторонним каналам)
    - HTML_OTHER (05) - используется только для HTML_UI. Челенж, который подразумевает взаимодействие, зашитое во встраиваемой странице.
- **WHOLE_OBJECT_AS_JSON** - Сериализованное в `JSON` сообщение со всеми полями, которое было получено или отправлено.
- **AUTHENTICATION_METHOD** - Тип аутентификации, который был использован. Допустимые значения (в скобках значение, использующееся в сообщениях) :
    - STATIC_PASSCODE(01)
    - SMS_OTP(02)
    - KEY_FOB_OR_EMV_CARD_READER_OTP(03)
    - APP_OTP(04)
    - OTP_OTHER(05)
    - KBA(06)
    - OOB_BIOMETRICS(07)

    - OOB_LOGIN(08)
    - OOB_OTHER(09)
    - OTHER(10)
- **AUTHENTICATION_VALUE** Кодовая последовательность подтверждения прохождения аутентификации
- **ECI** Electronic Commerce Indicator. Специфичен для каждой платежной системы.
- **TRANS_STATUS** Статус транзакции. Допустимые значения (в скобках значение, использующееся в сообщениях):
    - AUTHENTICATION_SUCCESSFUL(Y)
    - NOT_AUTHENTICATED(N)
    - AUTHENTICATION_COULD_NOT_BE_PERFORMED(U)
    - ATTEMPTS_PROCESSING_PERFORMED(A)
    - CHALLENGE_REQUIRED©
    - ACCOUNT_VERIFICATION_REJECTED®
- **DEVICE_CHANNEL** Индикатор канала сообщения (типа источника) Допустимые значения (в скобках значение, использующееся в сообщениях):
    - CHAN_01_APP (01) - Приложение. Источник SDK
    - CHAN_02_BRW (02) - Браузер. Источник 3DS Requestor
    - CHAN_03_3RI (03) - 3DS Server
- **VERSION** Версия протокола, по которому была осуществлена данная транзакция. Допустимые значения(в скобках значение, которое использовалось в сообщениях 3DS):
    - TWO_ONE_ZERO (2.1.0)

Таблица USED_ID

Таблица содержит результат обработки транзакции для 3DS 2.0.

Поля:

- **ID** Уникальный идентификатор. Генерируется на основе последовательности `used_id_seq`
- **ACS_TRANS_ID** Уникальный идентификатор транзакции со стороны ACS.
- **CREATE_DATE** Время создания записи. Используется локальное время сервера ACS
- **DS_TRANS_ID** Уникальный идентификатор транзакции со стороны DS.
- **ERROR_CODE** Код ошибки, с которым завершилась транзакция. Допустимые значения (в скобках даны значения, которые отправляются в `RReq` в Error сообщении):
    - MESSAGE_RECEIVED_INVALID (101)
    - MESSAGE_VERSION_NUMBER_NOT_SUPPORTED(102)
    - SENT_MESSAGES_LIMIT_EXCEEDED(103)

    - REQUIRED_DATA_ELEMENT_MISSING(201)
    - CRITICAL_MESSAGE_EXTENSION_NOT_RECOGNISED(202)
    - FORMAT_OF_DATA_ELEMENT_IS_INVALID(203)
    - DUPLICATE_DATA_ELEMENT(204)
    - TRANSACTION_ID_NOT_RECOGNISED(301)
    - DATA_DECRYPTION_FAILURE(302)
    - ACCESS_DENIED_OR_INVALID_ENDPOINT(303)
    - ISO_CODE_INVALID(304)
    - TRANSACTION_DATA_NOT_VALID(305)
    - MCC_NOT_VALID_FOR_PAYMENT_SYSTEM(306)
    - SERIAL_NUMBER_NOT_VALID(307)
    - TRANSACTION_TIMED_OUT(402)
    - TRANSIENT_SYSTEM_FAILURE(403)
    - PERMANENT_SYSTEM_FAILURE(404)
    - SYSTEM_CONNECTION_FAILURE(405)
- **ERROR_DESCRIPTION** Информативное описание ошибки со ссылкой на спецификацию EMV ® 3-D Secure Protocol and Core Functions Specification v2.1.0 доступной на сайте EMVCo https://www.emvco.com/
- **ERROR_DETAIL** Детали ошибки. Как правило, содержит список полей сообщения, в которых зафиксирована ошибка
- **ITERATION_COUNTER** Количество итераций прохождения челенжа
- **SDK_TRANS_ID** Уникальный идентификатор транзакции со стороны SDK.
- **THREE_DS_SERVER_TRANS_ID** Уникальный идентификатор транзакции со стороны 3DS Server.
- **VERSION** Версия протокола, по которому была осуществлена данная транзакция. Допустимые значения(в скобках значение, которое использовалось в сообщениях 3DS):
    - TWO_ONE_ZERO (2.1.0)

Таблица V1_THREE_DS_MESSAGE

Таблица содержит сообщения 3DS 1.0 : `VEReq`, `VERes`, `PAReq`, `PARes`.

Поля:

- **ID** Уникальный идентификатор. Генерируется на основе последовательности `v1_three_ds_message_seq`
- **ACCT_ID** Уникальный идентификатор транзакции по которому можно выбрать все сообщения цепочки (VEReq, VERes, PAReq, PARes).
- **ACQ_BIN** Идентификационный код банка-эквайера

- **CREATE_DATE** Время создания записи. Используется локальное время сервера ACS
- **CURRENCY** Валюта покупки в виде 3-буквенного обозначения. В оригинале предоставляется в виде трехзначного кода ISO 4217
- **ENROLLED** Индикатор возможности проведения 3DS операций по данному pan. Возможные значения(в скодбках значение, используемое в сообщениях)
  - AUTHENTICATION_AVAILABLE(Y)
  - CARDHOLDER_NOT_PARTICIPATING(N)
  - UNABLE_TO_AUTHENTICATE(U)
- **EXPONENT** Степень 10, показывающая количество разрядов, отводимых под дробную часть валюты
- **MD** "Merchant Data" поле, переданное вместе с PaReq
- **MERCHANT_ID** Идентификатор мерчанта
- **MESSAGE_ID** Идентификатор сообщения. Одинаковое для пар VeReq/VeRes и PaReq/PaRes.
- **MESSAGE_TYPE** Тип сообщения. Допустимые значения:
  - VEReq
  - VERes
  - PAReq
  - PARes
- **PAN** Номер карты
- **PURCH_AMOUNT** Сумма покупки в минимальных единицах валюты (Например, для рубля это будет количество копеек)
- **PURCHASE_DATE** Дата совершения покупки
- **TERM_URL** URL, на который должен быть отправлен PaRes.
- **WHOLE_OBJECT_AS_XML** Сериализованное в XML сообщение со всеми полями, которое было получено или отправлено.
- **XID** Уникальный идентификатор транзакции в момент прохождения челенжа
- **MERCHANT_NAME** Имя мерчанта
- **CARD_EXPIRY_DATE** Срок действия карты
- **ECI** Electronic Commerce Indicator. Специфичен для каждой платежной системы.
- **STATUS** Статус транзакции. Допустимые значения (в скобках значение, использующееся в сообщениях):
  - AUTHENTICATION_SUCCESSFUL(Y)
  - AUTHENTICATION_FAILED(N)
  - AUTHENTICATION_COULD_NOT_BE_PERFORMED(U)
  - ATTEMPTS_PROCESSING_PERFORMED(A)
- **CAVV** Кодовая последовательность подтверждения прохождения аутентификации

- **CAVV_ALGORITHM** Код алгоритма, который был использован для формирования `CAVV`. Допустимые значения:
  - 0 - `HMAC` (as per SETTM TransStain)
  - 1 - `CVV`
  - 2 - `CVV` with `ATN`

Таблица V1_USED_ID

Таблица содержит результат обработки транзакции для 3DS 1.0.

Поля:

- **ID** Уникальный идентификатор. Генерируется на основе последовательности `v1_used_id_seq`
- **ACCT_ID** Уникальный идентификатор транзакции.
- **CLOSED** Признак завершенности сессии `0` - не завершена, `1` - завершена
- **CREATE_DATE** Время создания записи. Используется локальное время сервера ACS
- **USE_COUNT** Количество попыток прохождения челенжа

Таблица TRUSTED_MERCHANT

Таблица содержит список доверенных мерчантов с определнными для них лимитами для Frictionless Flow

Поля:

- **ID** Уникальный идентификатор записи.
- **MAX_FRICTIONLESS_LIMIT** Верхний край интервала, внутри которого возможен Frictionless Flow
- **MIN_FRICTIONLESS_LIMIT** Нижний край интервала, внутри которого возможен Frictionless Flow
- **NAME** Имя мерчанта, по которму производится сопоставление при скоринге.
- **CURRENCY_CODE** Код валюты в виде трехзначного кода `ISO 4217` по которому производится проверка в интервале. Если валюты не совпадают, то интервал не учитывается.
- **NPA_FRICTIONLESS** Селектор поведения при NPA транзакции. Допустимые значения (Y|N). При Y транзакция уходит по Frictionless Flow.

Таблица SESSION_DATA

Данные специфичные для прохождения челенжа

Поля:

- **THREE_DS_SERVER_TRANS_ID** Уникальный идентификатор транзакции.
- **OTP** Одноразовый пароль, который был сгенерирован для прохождения челенжа
- **PHONE_NUMBER** Телефонный номер держателя карты, на который отправляется OTP
- **CARDHOLDER_NAME** Имя держателя карты
- **CREATED** Дата начала прохождения челенжа
- **LAST_UPDATED** Дата последней итерации челенжа

Таблица CHALLENGE_DATA

Общие данные для прохождения челенжа

Поля:

- **ACS_TRANS_ID** Уникальный идентификатор транзакции.
- **CEK** Base64 encoded content encryption key
- **COUNTER** Счетчик итераций прохождения челенжа
- **CREATE_DATE** Время начала челенжа
- **CHALLENGE_TIMEOUT** Время, когда истекает положенный таймаут для текущей стадии челенжа
- **PROCESSED** Флаг обработанной сессии. Допустимые значения `Y`/`N`

Представление V_CARD_RANGE

Данные о диапазонах карточных номеров, принадлежащих банку

Поля:

- **ID** Уникальный идентификатор записи.
- **PAYMENT_SYSTEM** Идентификатор платежной системы, к которой принадлежит данный диапазон
- **BIN** BIN, который зарегистрирован в платежной системе для выпуска карты.
- **BIN_DESCRIPTION** Название BIN-а
- **PAN_INDEX_RANGE_LOW** Начало диапазона
- **PAN_INDEX_RANGE_HIGH** Конец диапазона

Представление V_CARD_EC

Данные о статусе карты

Поля:

- **NCRD** Номер карты. Уникальный идентификатор записи.

- **VALID** Признак блокировки карты. Допустимые значения `Y`/`N`.
- **AUTH_3DS** Признак доступности операций без присутствия карты. Допустимые значения `Y`/`N`.
- **REASON** Числовой код причины блокировки карты. В случае `null` используется значение 12 (Недопустимая операция для держателя карты). Допустимые значения:
  - 04 Превышена частота аутентификаций по карте
  - 05 Карта просрочена
  - 09 Ошибка безопасности
  - 10 Укарденая карта
  - 11 Подозрение в мошенничестве
  - 12 Недопустимая операция для держателя карты
- **PHONE** Номер держателя карты для отправки авторизационных сообщений
- **MESSAGE** Поле, предназначенное для имени держателя карты

Представление BLACK_LIST

Данные черных списков для пары pan-merchantName.

Поля:

- **ID** Уникальный идентификатор записи.
- **PAN** Номер карты.
- **MERCHANT_NAME** Имя мерчанта
- **CREATED** Дата создания записи

Таблица T_LOCK

Таблица для синхронизации задач обновления данных между нодами ACS. Используется для создания блокировок, которые не позволят выполнять одну операцию обновления одновременно двум и более нодам.

Поля:

- **ID** - Уникальный идентификатор записи
- **OPERATION** - Тип операции, для которой выполняется синхронизация
- **CREATE_DATE** - Время создания блокировки.

На данный момент T_LOCK используется для синхронизации одной операции - обработки записей CHALLENGE_DATA с истекшим таймаутом. Значения полей для операции: **ID** = 1, **OPERATION** = `EXPIRED_CHALLENGES_PROCESSING`. По завершению обработки запись о блокировке удаляется. Блокировка, не снятая по причине какой-либо исключительной ситуации, теряет свое действие по истечении 30 секунд.

Описание эндпоинтов

Служебные

Эндпоинты для мониторинга/управлением состоянием ноды.

- Все стандартные эндпоинты, поддерживающиеся Spring Boot actuator, включая health check. Подробнее можно прочитать здесь.
- Эндпоинт изменения состояния ноды. Доступен, как один из эндпоинтов актуатора. Пример использования: На URL `<URL ACS>/actuator/state/set` с типом `"application/json"` посылается `POST` запрос с телом сообщения:

```
{
"state" : "ON"
}
```

Допустимые значения параметра state: `ON` и `OFF` для включения и выключения соответственно.

Эндпоинты взаимодействия с внутренними системами банка

Эндпоинт получения статуса транзакции

Адрес: `<URL ACS>/acs/{acs_version}/{paymentSystem}/check_av`

Где:

- `{acs_version}` - версия ACS. Например: `v2.1.0`
- `{paymentSystem}` - краткое значение для платежной системы. Допустимые значения:
  - `mc` для MASTER_CARD
  - `visa` для VISA
  - `mir` для MIR

Type: `POST`, Content-type: `application/json`

Полный пример: `<URL ACS>/acs/v2.1.0/mir/check_av`

Пример запроса:

```
{
    "pan" : "1234567890123456",
    "dsTransID" : "3a9d2540-8c10-5746-8000-0000000f67e9",
    "av" : "AAIBBmI1hiBpV5Z4FDWGIiIiIiI="
}
```

Пример ответа:

```
{
    "transactionStatus": "Y"
}
```

Эндпоинт получения blacklist по pan

Адрес: `<URL ACS>/sngb/blacklist/get/{pan}`

Где:

- `{pan}` - номер карты

Type: `GET`

Полный пример: `<URL ACS>/sngb/blacklist/get/1234567890123456`

Пример ответа:

```
[
  {
    "id": "1",
    "merchantName" : "some merchant 1",
    "created" : "201912122112"
  },
  {
    "id": "2",
    "merchantName" : "some merchant 2",
    "created" : "201912122113"
  }
]
```

Эндпоинт удаления из blacklist по id

Адрес: `<URL ACS>/sngb/blacklist/delete/{id}`

Где:

- `{id}` - уникальный идентификатор записи в таблице `BLACK_LIST`

Type: `DELETE`

Полный пример: `<URL ACS>/sngb/blacklist/delete/1`

В ответ вернется пустой ответ с кодом `200`.

Эндпоинт добавления в blacklist

Адрес: `<URL ACS>/sngb/blacklist/put`

Type: `POST`, Content-type: `application/json`

Полный пример: `<URL ACS>/sngb/blacklist/put`

Пример запроса:

```
{
    "pan" : "1234567890123456",
    "merchantName" : "some merchant 1"
}
```

Пример ответа:

```
{
  "id": "1",
  "merchantName" : "some merchant 1",
  "created" : "201912122112"
}
```

Эндпоинт получения trusted_merchant по merchantName

Адрес: `<URL ACS>/sngb/merchant/get/{merchantName}`

Где:

- `{pan}` - имя мерчанта. Поддерживаются пробелы.

Type: `GET`

Полный пример: `<URL ACS>/sngb/merchant/get/some merchant 1`

Пример ответа:

```
{
    "id": 2,
    "minFrictionlessLimit": 0,
    "maxFrictionlessLimit": 100,
    "currencyCode": 643
}
```

В случае отсутствующего мерчанта с таким именем придет пустой ответ.

Эндпоинт удаления trusted_merchant по id

Адрес: `<URL ACS>/sngb/merchant/delete/{id}`

Где:

- `{id}` - уникальный идентификатор записи в таблице `TRUSTED_MERCHANT`

Type: `DELETE`

Полный пример: `<URL ACS>/sngb/merchant/delete/1`

В ответ вернется пустой ответ с кодом `200`.

Эндпоинт добавления trusted_merchant

Адрес: `<URL ACS>/sngb/merchant/put`

Type: `POST`, Content-type: `application/json`

Полный пример: `<URL ACS>/sngb/merchant/put`

Пример запроса:

```
{
    "merchantName" : "some merchant 1",
    "minFrictionlessLimit" : "0",
    "maxFrictionlessLimit" : "100",
    "currencyCode" : "643"
}
```

Пример ответа:

```
{
    "result": "INSERTED",
    "id": 2
}
```

В ответе возможны 3 результата: INSERTED, UPDATED, ERROR в зависимости от результата операции. В случае ERROR придет поле errorDescription с описанием ошибки.

Для DS

Эндпоинты для взаимодействия с DS.

Эндпоинт аутентификации. AReq/ARes взаимодействие с DS.

Адрес: `<URL ACS>/acs/{acs_version}/{paymentSystem}/authentication`

Где:

- `{acs_version}` - версия ACS. Например: `v2.1.0`
- `{paymentSystem}` - краткое значение для платежной системы. Допустимые значения:
  - `mc` для MASTER_CARD
  - `visa` для VISA
  - `mir` для MIR

Type: `POST`, Content-type: `application/json`

Полный пример: `<URL ACS>/acs/v2.1.0/mir/authentication`

Пример `AReq`:


```
{
  "threeDSRequestorAuthenticationInd": "01",
  "threeDSRequestorAuthenticationInfo": {
    "threeDSReqAuthData": "00",
    "threeDSReqAuthMethod": "04",
    "threeDSReqAuthTimestamp": "201909110843"
  },
  "threeDSRequestorChallengeInd": "01",
  "threeDSRequestorID": "467",
  "threeDSRequestorName": "EMVCo 3DS Test Requestor",
  "threeDSRequestorURL": "https://some.requestor.url.com/417853f2-8ae0-4bac-8ed5-
06eda3f3cb2b",
  "threeDSServerRefNumber": "3DS_LOA_SER_PPFU_020100_00008",
  "threeDSServerOperatorID": "threeDSServerOperatorUL",
  "threeDSServerTransID": "ee35809f-d708-4b3c-b026-d14cbeda0f6a",
  "threeDSServerURL": "https://test.rtln.ru/3dss/v2.1.0/mc/result",
  "acctType": "02",
  "acquirerBIN": "555555",
  "acquirerMerchantID": "555555",
  "addrMatch": "Y",
  "cardExpiryDate": "2212",
  "acctInfo": {
    "chAccAgeInd": "05",
    "chAccChange": "20170101",
    "chAccChangeInd": "04",
    "chAccDate": "20170101",
    "chAccPwChange": "20170101",
    "chAccPwChangeInd": "05",
    "nbPurchaseAccount": "01",
    "provisionAttemptsDay": "000",
    "txnActivityDay": "1",
```

```
    "txnActivityYear": "01",
    "paymentAccAge": "20170101",
    "paymentAccInd": "05",
    "shipAddressUsage": "20170101",
    "shipAddressUsageInd": "04",
    "shipNameIndicator": "01",
    "suspiciousAccActivity": "01"
  },
  "acctNumber": "5204240438720050123",
  "acctID": "EMVCo 3DS Test Account 000000001",
  "billAddrCity": "City Name",
  "billAddrCountry": "840",
  "billAddrLine1": "Address Line 1",
  "billAddrLine2": "Address Line 2",
  "billAddrLine3": "Address Line 3",
  "billAddrPostCode": "Postal Code",
  "billAddrState": "AZ",
  "email": "example@example.com",
  "homePhone": {
    "cc": "123",
    "subscriber": "123456789"
  },
  "mobilePhone": {
    "cc": "123",
    "subscriber": "123456789"
  },
  "cardholderName": "Frictionless One",
  "shipAddrCity": "City Name",
  "shipAddrCountry": "840",
  "shipAddrLine1": "Address Line 1",
  "shipAddrLine2": "Address Line 2",
  "shipAddrLine3": "Address Line 3",
  "shipAddrPostCode": "Postal Code",
  "shipAddrState": "AZ",
  "workPhone": {
    "cc": "123",
    "subscriber": "123456789"
  },
  "deviceChannel": "01",
  "deviceRenderOptions": {
    "sdkInterface": "03",
    "sdkUiType": [
      "01",
      "02",
      "03",
      "04",
      "05"
    ]
  },
  "mcc": "7922",
  "merchantCountryCode": "840",
```

```
  "merchantName": "Ticket Service",
  "merchantRiskIndicator": {
    "deliveryEmailAddress": "example@example.com",
    "deliveryTimeframe": "02",
    "giftCardAmount": "01",
    "giftCardCount": "01",
    "giftCardCurr": "840",
    "preOrderDate": "20300101",
    "preOrderPurchaseInd": "01",
    "reorderItemsInd": "01",
    "shipIndicator": "01"
  },
  "messageCategory": "01",
  "messageType": "AReq",
  "messageVersion": "2.1.0",
  "purchaseAmount": "01",
  "purchaseCurrency": "840",
  "purchaseExponent": "2",
  "purchaseDate": "20190911084303",
  "sdkAppID": "c15373da-7ca3-491a-be08-98d1fe9a6903",
  "sdkEncData": "eyJhbGciOiJSU0EtT0FFUC0yNTYiLCJlbmMiOiJBM.....",
  "sdkEphemPubKey": {
    "kty": "EC",
    "crv": "P-256",
    "x": "JzGkF3l7XlgrRz5MON_gp87YL_wCdURiUIq9rf6urGc",
    "y": "Nr8RieLOoW2WRHb_dEk1ftthXFWLwXifES3YfAg2HoY"
  },
  "sdkMaxTimeout": "05",
  "sdkReferenceNumber": "3DS_LOA_SDK_PPFU_020100_00007",
  "sdkTransID": "908804fe-7980-4a44-9af7-88c5c918f54f",
  "transType": "01"
}
```


Пример `ARes`:


```
{
  "threeDSServerTransID": "ee35809f-d708-4b3c-b026-d14cbeda0f6a",
  "acsOperatorID": "someOperatorID",
  "acsReferenceNumber": "3DS_LOA_ACS_PPFU_020100_00009",
  "acsTransID": "747e79b6-fb48-4997-844e-7633343e2013",
  "authenticationValue": "kJMZRiDunhPsBwAU0hu9rrx0eWN6",
  "dsReferenceNumber": "3DS_LOA_DIS_PPFU_020100_00010",
  "dsTransID": "ce8a8245-f5f4-4711-b481-eb21eed7fd8f",
  "eci": "02",
  "messageType": "ARes",
  "messageVersion": "2.1.0",
  "sdkTransID": "908804fe-7980-4a44-9af7-88c5c918f54f",
  "transStatus": "Y"
}
```

Эндпоинт проверки участия карты в 3DS v1.0. VeReq/VeRes взаимодействие.

Адрес: `<URL ACS>/acs/{acs_version}/{paymentSystem}/authentication`

Где:

- `{acs_version}` - версия ACS. Например: `v1.0.2`
- `{paymentSystem}` - краткое значение для платежной системы. Допустимые значения:
  - `mc` для MASTER_CARD
  - `visa` для VISA
  - `mir` для MIR

Type: `POST`, Content-type: `application/xml`

Полный пример: `<URL ACS>/acs/v1.0.2/mir/authentication`

`VeReq`:

```
<ThreeDSecure>
    <Message id="1703086a-992f-478c-a78a-25a9fe9f4d3d">
        <VEReq>
            <version>1.0.2</version>
            <pan>5213240000005235</pan>
            <Merchant>
                <acqBIN>546901</acqBIN>
                <merID>781000006125</merID>
            </Merchant>
            <Browser>
                <deviceCategory>0</deviceCategory>
            </Browser>
        </VEReq>
    </Message>
</ThreeDSecure>
```

`VeRes`:

```
<ThreeDSecure>
    <Message id="1703086a-992f-478c-a78a-25a9fe9f4d3d">
        <VERes>
            <version>1.0.2</version>
            <CH>
                <enrolled>Y</enrolled>
                <acctID>JEd8g4TbUoWeVvjTR1Nrv1StJRxH</acctID>
            </CH>
            <url>https://test.rtln.ru/acs/validation/start</url>
            <protocol>ThreeDSecure</protocol>
        </VERes>
    </Message>
</ThreeDSecure>
```

Для взаимодействия с держателем карты

Эндпоинт начала прохождения челенжа. В случае мобильного приложения полный обмен CReq/CRes в шифрованном виде.

Адрес: `<URL ACS>/acs/{acs_version}/{paymentSystem}/challenge/start`

Где:

- `{acs_version}` - версия ACS. Например: `v2.1.0`
- `{paymentSystem}` - краткое значение для платежной системы. Допустимые значения:
  - `mc` для MASTER_CARD
  - `visa` для VISA
  - `mir` для MIR

Полный пример: `<URL ACS>/acs/v2.1.0/mc/challenge/start`

- Взаимодействие через браузер

Type: `POST`, Content-type: `application/x-www-form-urlencoded`

Кодированный `CReq`

```
POST /acs/v2.1.0/mc/challenge/start / HTTP/1.1
Host: acs.bank.com
Content-Type: application/x-www-form-urlencoded
creq=eyJ0aHJlZURTU2VydmVyVHJhbnNJRCI6IjJmZmNjYTRiLTRjZDYtNDBkYy1iYjI5LTg1YmJhZ
WMyMTFhNiIsImFjc1RyYW5zSUQiOiJmODE4ZTBmMi01OTUzLTQwYjgtOGM5ZC01MzMxNzA1YTUwODA
iLCJtZXNzYWdlVHlwZSI6IkNSZXEiLCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4wIiwiY2hhbGxlbmdlV
2luZG93U2l6ZSI6IjAyIn0%3D&threeDSSessionData=VGhpcyBpcyBteSBzZXNzaW9uIGRhdGEgM
TIzNDU2Nzg5MA%3D
```

В ответ отдается страница для ввода OTP или ошибка, в случае, если расшифрованный CReq содержит ошибки.

- Взаимодействие через мобильное приложение

Взаимодействие происходит посредством обмена JWE сообщениями. Подробнее о JWE тут

Type: POST, Content-type: application/jose

Шифрованный CReq

```
POST /acs/v2.1.0/mc/challenge/start / HTTP/1.1
Host: acs.bank.com
Content-Type: application/jose
eyJhbGciOiJkaXIiLCJraWQiOiJBQ1NUcmFuc2FjdGlvbklEIiwiZW5jIjoiQTEyOENC
Qy1IUzI1NiJ9
.
.
MduEJ9zyW-6myF0OP2Zf6g
.
Ek1a4hfYBjURaVUrp1BDJqbKVIGuBKPnE0Orb7Ltt9yZHrLI4XgwZ901R9Z70cV5k-
KA1-I2EJAbaVmamVYISgJdPv3sacBBb0Iwy7xgat7sRk0SX8dT-
7pdlppNgfmJtzghzSyRoDVYiASELFqp58txyAVBsYupS7-
dwW4xhJx6r6gionbxwhTQFkJ-pi1L0KRMp6pKJ91XSel9SJ-
ncM0OKSsl6puURzU15e2g4Pl-Fg5xyzymGyDIcTDdHrifWJLh3kaDx7g3Gx8R3j-
5cTtK0I95KKCg8vcoKbCrlDoBAzIfRFbV3xugnDWnXG47
.
-Q84vHampMJBuvBa-nilKg
```

В ответ формируется аналогичное сообщение, содержащее CRes.

```
eyJhbGciOiJkaXIiLCJraWQiOiJBQ1NUcmFuc2FjdGlvbklEIiwiZW5jIjoiQTEyOEdD
TSJ9
.
.
_____________8B
.
85r0V6__KhxNoaVr_ZHLR6T1Y1AH8aZYmnkCsaDaweRaphUD4IzJlKg4BVge6kwoJLrf
Mp_eqHQgEZ7S_VxI4p8G_IqsfVotx79FSRQYBdQKvQpZPyEmSVi_tjgbvz20BC03R7GP
6UQU81UHEdQOiaoB8AY58fDl6yKSG59If_35EKy5-rN9wWcUQqzrtAV914TIRGIm4EJ-
zit30Ma1dIAOhY6q_NrKRTSpRzIzNGv4CdvfZjcvqKD5hLEil0E4E9k7fZ5sjOvie53f
O-Qm8xneglQy_MHv4hq-
gZW2_G6R8BesmrZ48xzM7vrTiWLmGWcQR_Tthe94ZL2gqhL32liwH8Zp1xiuHrOjCDl_
EqakvX5MGDaWVzoKjr10Pun8TzFK9X3O_ukHn87pGUCz1Q9mktRVMhr_JCgtL7AuadDN
pwLTiGycGGBk8X-zHaYN3--
gGOx66IAJg0J1THedVkeGF5GIvNqo0pmF3XMDz3H7YrX7GsOABf19imeOIigm0CXOEFS
JzYsSjwp_k9AepmNfg3JZt0bk9YPfQ7LiVq1rs2OQ_6iK_Mdh4dsx2AdydrOkhcwQ5u7
rBkGm3TtaHNJ8fj0V2MD7MFE-QcvTz5ht0-
sDCnLNeuglnjXlFXhnw4caim9M5iJD9TW3rMVc2n1X3w4fBUuOlR2ccmyWxTTTMLCIQn
dXedQdG37QVbx0MlWejzVwVn-
3np7z8B8SwUIdnqZlcDgPC3OKszXpeqJrH3f2uwqTcksgJANMrGmLpvtisvvv6VYKV9d
H9I2ffU1eU_2Cja9xpSluD8O5uvjnbSXG-Va8-QbLGoyb9EBUtKNFhATj-
m45V_A7Pl3QCGllnRXi6Hhk7TiKcdR5Cm6eUKy_akCBpI4csO-
6RlCLBV9iW1h6qhBabky74km0yDidzBCKhd0V8_fXTRbITzs9sedZYYImRRNOsT4uxRd
xIODBY7t0ZPk87SN-XZtXKuLbdpvxoHNRmlDqHo7JXsrvL22IK1s-
q6WrGAsjjIzmb7uVEqZsVGfapInWQYc3HzyBvhSsuo7m7hS8C_KJLUmfU5j3bgOGBO_M
-
JrFK4QQgjNWXf8zWjbW6RckbDl3EF3gnvck9EWTYnd9sVz8YbaCKvJ8fhIS4kjv8qmus
ItKuKVxqfDqcf5r8YdPQ6g6Wkt-
UozwOUNCmHVx87vu3BL4zpcb2Q3oKXWhtYLyrD2K5OBBmKY5eBd7VcFeSqO0zrMRu6wG
LL-Dz_HRkSMbf1fsJyVJRvpUCzvfzksl6GXCaxB-HSlVg-
du466WYfUR9ulm2mshyZvPmTNYmMbE_q0eQ93m9bRWazlXFliTY2YNXCa5uH7v63cHIR
vH3uwPYJEkhwzx2popm4w8Dzo11w66llwPXh-
HMqFq9HXBgqHBM47KaqBaTzVm6YYXhTuJoVY7ON2hR9YKxF6Siw-a-
TARtWayfpLZuN0mxO3mRMAaRHgI9lyoKYmLTHiFqb8-
Yk279owO555JlwvZRoos96WmD8R8RACo61_MXrohn3Qfc0_A_zPcp0H0KttnYneOINnK
L45UMGWpRoJd_iUfR6EId1SdJd8b9XFIbeJT4DcBzPuEbTKgGSrXnu4xJlMcXTbUufNk
NKKHKc5jvvBpp1w63dp-4fo667x1gB75HiKn-tgNRox43lNAQlRen4rLb3licM-
DBdgHuZtPWWu0c3UNIG_ujMn8V3srIMfjbi6LRyNBgMxIEhnBtASYPucX1w
.
yj8Vq9UZKoY1LL19g1sznQ
```

Эндпоинт проверки челенжа 3DS v2.0.

Адрес: `<URL ACS>/acs/{acs_version}/{paymentSystem}/challenge/finish`

Где:

- `{acs_version}` - версия ACS. Например: `v2.1.0`
- `{paymentSystem}` - краткое значение для платежной системы. Допустимые значения:

- `mc` для MASTER_CARD
- `visa` для VISA
- `mir` для MIR

Type: `POST`, Content-type: `application/json`

Полный пример: `<URL ACS>/acs/v2.1.0/mir/challenge/finish`

Пример запроса

```
{"threeDSServerTransID":"267f84dd-62c5-4a4c-bac4-fca976acc274", "result":"123456"}
```

Ответ

```
{
  "final" : "true",
  "notificationURL" : "https://test.rtln.ru/3dss/v2.1.0/mc/result",
  "encodedCRes" : "JrFK4QQgjNWXf8zWjbW6RckbDl3EF3gnvck9EWTYnd9sVz8YbaC.....",
  "threeDSServerTransID" : "267f84dd-62c5-4a4c-bac4-fca976acc274"
}
```

Эндпоинт получения формы челенжа 3DS v1.0. PaReq/форма ввода OTP

Адрес: `<URL ACS>/acs/{acs_version}/{paymentSystem}/validation/start`

Где:

- `{acs_version}` - версия ACS. Например: `v1.0.2`
- `{paymentSystem}` - краткое значение для платежной системы. Допустимые значения:
  - `mc` для MASTER_CARD
  - `visa` для VISA
  - `mir` для MIR

Type: `POST`, Content-type: `application/x-www-form-urlencoded`

Полный пример: `<URL ACS>/acs/v1.0.2/mir/validation/start`

`PaReq`

```
<ThreeDSecure>
    <Message id="PRQ1627147201">
        <PAReq>
            <version>1.0.2</version>
            <Merchant>
                <acqBIN>220220</acqBIN>
                <merID>400000022766</merID>
                <name>mirkorma</name>
                <country>643</country>
                <url>https://www.some.url.ru</url>
            </Merchant>
            <Purchase>
                <xid>OTIwNTE1NTE1NjE1NDUyMjM0NDI=</xid>
                <date>20190626 10:33:43</date>
                <amount>RUB 6042.00</amount>
                <purchAmount>604200</purchAmount>
                <currency>643</currency>
                <exponent>2</exponent>
            </Purchase>
            <CH>
                <acctID>WcY0wLC9sA2XadoeAkzMixBIyZuk</acctID>
                <expiry>2403</expiry>
            </CH>
        </PAReq>
    </Message>
</ThreeDSecure>
```

В ответ отправляется форма прохождения челенжа для ввода OTP.

Эндпоинт проверки челенжа 3DS v1.0. PaReq/форма ввода OTP

Адрес: `<URL ACS>/acs/{acs_version}/{paymentSystem}/validation/finish`

Где:

- `{acs_version}` - версия ACS. Например: `v1.0.2`
- `{paymentSystem}` - краткое значение для платежной системы. Допустимые значения:
  - `mc` для MASTER_CARD
  - `visa` для VISA
  - `mir` для MIR

Type: `POST`, Content-type: `application/json`

Полный пример: `<URL ACS>/acs/v1.0.2/mir/validation/finish`

Запрос:

```
{
  "acctID": "WcY0wLC9sA2XadoeAkzMixBIyZuk",
  "otp" : "123456",
  "cancelled" : "false"
}
```

Ответ

```
{
  "retryAvailable": "false",
  "successfully" : "true",
  "termURL" : "https://www.some.url.ru",
  "md" : "LHerlkvmJEFJoegjergjoreorgKEF",
  "encodedPaRes" : "<some compressed and encoded PaRes response>"
}
```

Эндпоинт повторной отправки OTP 3DS v2.0

Адрес: `<URL ACS>/acs/{acs_version}/{paymentSystem}/challenge/resend_otp`

Где:

- `{acs_version}` - версия ACS. Например: `v2.1.0`
- `{paymentSystem}` - краткое значение для платежной системы. Допустимые значения:
  - `mc` для MASTER_CARD
  - `visa` для VISA
  - `mir` для MIR

Type: `POST`, Content-type: `application/json`

Полный пример: `<URL ACS>/acs/v2.1.0/mir/challenge/resend_otp`

Пример запроса

```
{"threeDSServerTransID":"267f84dd-62c5-4a4c-bac4-fca976acc274"}
```

Пример ответа

```
{"successful": "true","threeDSServerTransID":"267f84dd-62c5-4a4c-bac4-fca976acc274"}
```

Эндпоинт повторной отправки OTP 3DS v1.0

Адрес: `<URL ACS>/acs/{acs_version}/{paymentSystem}/validation/resend_otp`

Где:

- `{acs_version}` - версия ACS. Например: `v1.0.2`
- `{paymentSystem}` - краткое значение для платежной системы. Допустимые значения:
  - `mc` для MASTER_CARD
  - `visa` для VISA
  - `mir` для MIR

Type: `POST`, Content-type: `application/json`

Полный пример: `<URL ACS>/acs/v1.0.2/mir/challenge/resend_otp`

Пример запроса

```
{"acctID":"267f84dd-62c5-4a4c-bac4-fca976acc274"}
```

Пример ответа

```
{"successful": "true","acctID":"267f84dd-62c5-4a4c-bac4-fca976acc274"}
```