

Программный продукт  
«СмартВиста Электронная коммерция»

**Руководство по эксплуатации**

**ООО "БПС ПРОГРАММНЫЕ ПРОДУКТЫ" (с) 2022**

## Важное примечание

Обладателем исключительного права на ПО "СмартВиста Электронная коммерция" является ООО "БПС ПРОГРАММНЫЕ ПРОДУКТЫ" на основании ст. 1229 IV части ГК РФ.

Вся информация, содержащаяся в настоящей документации, а также описанное в нем программное обеспечение являются собственностью компании ООО "БПС ПРОГРАММНЫЕ ПРОДУКТЫ", никакая часть этого документа не может быть воспроизведена или передана в любой форме или с помощью электронных, механических средств, записи или любым другим способом без предварительного письменного разрешения ООО "БПС ПРОГРАММНЫЕ ПРОДУКТЫ".

## 1 Введение

Программное обеспечение «СмартВиста Электронная коммерция» (далее в тексте – ПО) предназначено для проведения платежей за электронные покупки и аутентификации по технологии 3-ДС в соответствии с правилами международных и локальных платежных систем (Visa, Mastercard, НСПК «МИР»). ПО применяется эквайерами электронной коммерции и эмитентами платёжных карт, выполняет проверку вовлеченности банка-эмитента и карты в технологию 3-ДС, аутентификацию и авторизацию пользователя, авторизацию оплаты товаров и услуг.

Программа реализует следующие функциональные возможности:

- передача сообщений о платеже от торгово-сервисного предприятия (ТСП) в систему авторизации транзакций;
- инициация операций покупки, возврата или возмещения средств, предавторизации, завершения покупки, перевода средств с карты на карту;
- интерфейс со справочными серверами платежных систем;
- формирование отчетов;
- аудит операций и действий пользователя;
- эквайерские функции технологии 3-ДС;
- эмитентские функции технологии 3-ДС;
- аутентификация держателя карты при веб-покупке;
- регистрация карт по технологии 3-ДС и управление их состоянием;
- пользовательский интерфейс для просмотра истории операций и управления системой.

Структурно программа состоит из следующих модулей:

- эмитентский модуль (сервер контроля доступа, сервер ACS)
- эквайерский модуль (сервер MPI и подсистема управления торгово-сервисными предприятиями)

## 2 Эмитентский модуль

### 2.1 Инфраструктура сети ПО

Пример инфраструктуры сети ПО представлен на рисунке 1. Сервер ПО взаимодействует со следующими серверами:

- Серверы каталогов (Directory Servers) платежных систем:
  - Сервер каталогов VISA
  - Сервер каталогов MasterCard
  - Сервер каталогов НСПК
- Серверы истории аутентификаций (AHS):
  - AHS Visa
  - AHS MasterCard
- Сервер базы данных ACS
- Сервер ПО СмартВиста Фронт Энд Рус (SVFE)
- Криптомодуль (HSM)
- Смс-шлюз

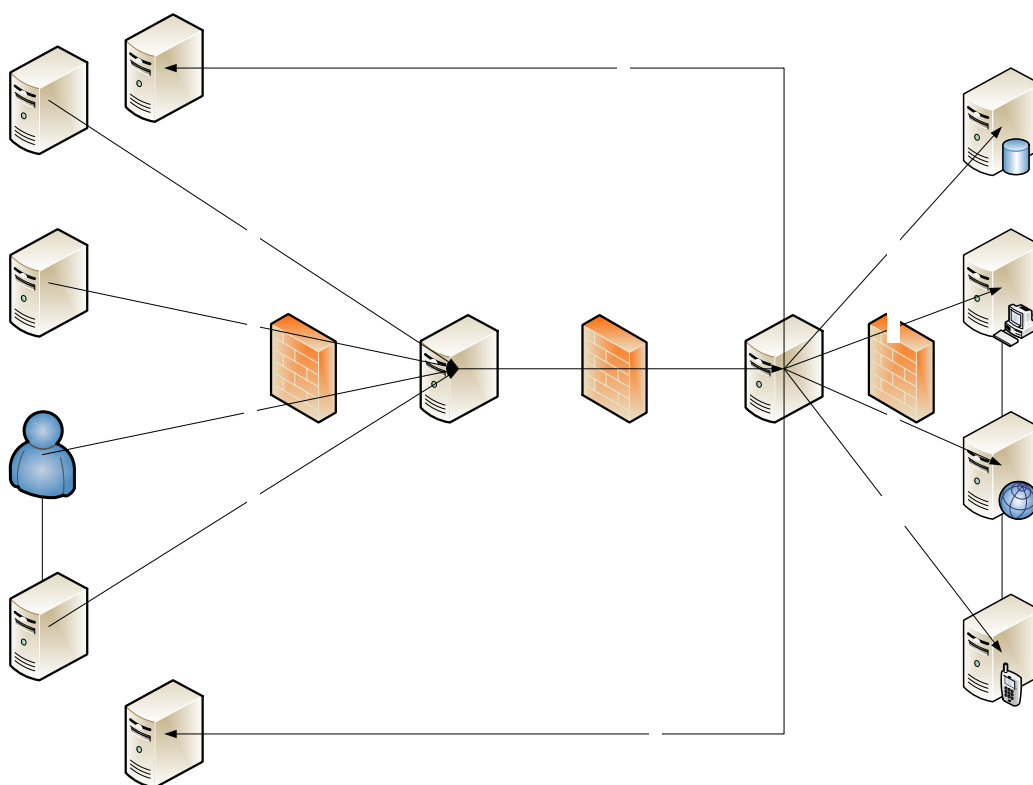


Рисунок 1. Инфраструктура сети ПО

В примерах, приведенных в последующих разделах, bank.com – внешний адрес.

### 2.1.1 Сервер каталогов VISA для ACS-соединения

Сервер каталогов VISA подключается к web прокси-серверу по адресу <https://bank.com:9643/acs/vereq>, который обычно называется «URL-адрес регистрации проверок ACS». Используется двусторонняя аутентификация по протоколу SSL. Сертификат Банка должен быть подписан VISA. Затем сертификат и соответствующий закрытый ключ устанавливаются на прокси-сервер для виртуального узла: 9643. Корневой и промежуточный CA-сертификаты VISA должны быть установлены как доверенные сертификаты на тот же виртуальный узел.

Затем запрос пересылается на сервер приложения ACS

`http(s)://<application server host>:<application server port>/acs/vereq`

SSL-сертификат, подписанный VISA, обычно называется «Сертификат ACS сервера».

#### 2.1.1.1 Особенности использования портов:

- Использование порта 9643 объясняется следующими причинами. VISA устанавливает ограничения на номер порта, используемого в рабочих средах. Может использоваться как 443, так и 96xx/97xx. Банк, как правило, использует один внешний IP-адрес, поэтому, чтобы обеспечить требуемое разрешение для SSL сертификата, необходимы разные порты. Порт 443 является стандартным и должен использоваться для соединения 4 (от пользователя к ACS). Таким образом, свободными остаются редко используемые в обычной практике порты.
- Устройство VISA PIT для проверки соединения от сервера каталогов к ACS может использовать только порт 443. Для успешного проведения проверки необходимо либо использовать другой порт для соединения 4 (например, 4443 вместо 443), либо в конфигурационном файле прокси-сервера указать виртуальный узел пользователя после виртуального узла VISA DS. Однако в рабочей среде следует использовать порт 9643 для VISA DS и 443 – для пользователей.

### 2.1.2 Сервер каталогов MasterCard для ACS-соединения

Данное соединение устанавливается аналогично VISA DS к ACS. Сервер каталогов MasterCard подключается к прокси-серверу веб-сайта по адресу <https://bank.com:5443/acs/vereq>, который обычно называется «URL-адрес регистрации проверок ACS». Используется двусторонняя аутентификация по протоколу SSL. Сертификат Банка должен быть подписан MasterCard. Затем сертификат и соответствующий закрытый ключ должны быть установлены на прокси-сервер для виртуального узла \*:5443. Корневой и промежуточный CA-сертификаты MasterCard должны быть установлены как доверенные сертификаты на тот же виртуальный узел.

Затем запрос пересылается на

`http(s)://<application server host>:<application server port>/acs/vereq`

SSL-сертификат, подписанный MasterCard, обычно называется « Сертификат ACS сервера ».

### 2.1.3 Сервер каталогов НСПК для ACS-соединения

Соединение устанавливается аналогично подключению VISA DS или MasterCard DS к ACS. При этом используемый порт должен отличаться от портов, предназначенных для любых других входящих SSL-соединений.

### 2.1.4 Соединение от пользователя к ACS

Данный тип соединения используется при отображении страницы аутентификации пользователю. Пример URL-адреса страницы аутентификации: <https://bank.com/acs/pareq>.

Для данного типа соединения используется односторонняя SSL-аутентификация. Чтобы сертификат Банка считался доверенным для веб-браузера, он должен быть подписан Thawte, Verisign или другим подобным центром сертификации. В запросе на подписание сертификата должно быть указано соответствующее имя сервера (Common Name).

Затем запрос пересылается:

`http(s)://<application server host>:<application server port>/acs/pareq`

При выводе страницы используются также следующие пути:

`/acs/pages/*`

`/acs/js*`

`/acs/style*`

`/acs/logoimage*`

`/acs/sessionExpired.jsf`

`/acs/unknownError.jsf`

`/acs/notFound.html`

Таким образом, согласно правилам, настроенным для прокси-сервера, должен быть разрешен доступ к указанным ресурсам.

Не рекомендуется задавать для прокси-правил лишь значения `/acs/*`.

### 2.1.5 Соединение от ACS к серверу AHS VISA

ACS подключается напрямую к AHS-серверу VISA для передачи данных журнала аутентификаций. Адрес AHS-сервера VISA предоставляется VISA и затем указывается в `acs.properties` (см. Приложение 1: [Параметры приложения](#)).

Используется двусторонняя SSL-аутентификация. SSL-сертификат, используемый для данного типа соединения, называется «Сертификатом ACS клиента». Сертификат должен быть подписан VISA. Сертификат клиента ACS и соответствующий закрытый ключ должны быть указаны в свойствах ACS (см. Приложение 1), так же как корневой и промежуточный CA-сертификаты.

### 2.1.6 Соединение от ACS к серверу AHS MasterCard

Соединение данного типа устанавливается аналогично соединению от ACS к AHS-серверу VISA. Используемый для соединения SSL-сертификат должен быть подписан MasterCard.

### 2.1.7 Соединение от ACS к СУБД

Данное соединение используется для подключения к базе данных ACS. В этой базе данных хранятся диапазоны карт, вовлеченных в технологию 3-ДС. Соединение настраивается на стороне сервера приложения в конфигурации источника данных. Для JNDI-имени источника данных следует указать значение `jdbc/acs/db`.

Для более подробной информации, см. документацию к используемому серверу приложений.

### 2.1.8 Соединение от ACS к криптомодулю (HSM)

Криptomодуль используется ПО для подписи PRes-сообщений. Соединение от ACS к криптомодулю настраивается в зависимости от типа используемого устройства. ACS работает со следующими типами криптомодулей (HSM): Thales payShield 9000 и Gemalto SafeNet Luna EFT 2. См. документацию для соответствующего типа криптомодуля.

### 2.1.9 Соединение от ACS к СмартВиста Фронт Энд Рус

ACS вызывает систему СмартВиста Фронт Энд Рус (далее в тексте – также SVFE) через интерфейс ISO8583 для:

- проверки вовлеченности карты в технологию 3-ДС
- генерации CAVV.

Параметры соединения настраиваются в `acs.properties`.

### 2.1.10 Соединение от ACS к SMS шлюзу

ACS использует соединение SMS Gateway для подключения к SMS-шлюзу, через который отправляются сообщения с разовым паролем. Настройка соединения варьируется в зависимости от используемого интерфейса для отправки SMS-сообщений (см. соответствующую документацию). Схема взаимодействия ACS с SMS-шлюзом изображена на рисунке. SMS Шлюз – внешний сервис клиента, используемый для sms взаимодействия

с картодержателями. Данный компонент не входит в состав предоставляемого программного обеспечения "СмартВиста Электронная коммерция".

## 3 Эквайерский модуль

### Основные функции

Эквайерский модуль, входящий в состав продукта «СмартВиста Электронная коммерция» поддерживает следующий набор функций:

1. Настройка и управление торгово-сервисными предприятиями, осуществляющими платежи электронной коммерции (далее по тексту – ТСП);
2. Настройка платежных web-страниц для отдельных ТСП. Под платежной страницей понимается специальная защищенная web-страница, на которой вводятся данные карты для проведения платежа электронной коммерции. Данная настройка включает кастомизацию HTML, JS, CSS в соответствии со стилем, темой и шрифтами, используемыми на основном web-сайте ТСП (Интернет-магазин);
3. Взаимодействие с Visa/MasterCard Directory Server;
4. Эквайринговая обработка в рамках технологии 3-ДС (как для карт локального банка, так и по картам сторонних эмитентов);
5. Обработка Us-On-Us транзакций, используя специальные диапазоны BINов, например для карт лояльности;
6. Управление токенами;
7. Поддержка мультивалютности для ТСП;
8. Поддержка функционала уведомлений ТСП о результатах обработки транзакций, следующими способами:
  - 8.1. Рассылка email уведомлений
  - 8.2. Организация обратных звонков через основной web-сайт ТСП.

Система СмартВиста Фронт Энд Рус выполняет обработку платежей электронной коммерции в рамках стандартной функциональности в соответствии с требованиями МПС Visa/MasterCard.

### 3.1. Управление токенами

Эквайерский модуль поддерживает функции создания, хранения, изменения и использования токенов и связок токенов, объединенных по уникальному значению ID пользователя, получаемых со стороны web-сайта ТСП. При этом обеспечивается выполнение следующих функций:

1. Возможность распространения одного токена среди нескольких ТСП (в соответствии с договоренностью);
2. Возможность создания, активации/деактивации связок токенов пользователем;
3. Связка может быть создана одновременно с созданием платежа (методом клика на соответствующий элемент checkbox на стороне web-сайта ТСП).

### 3.2 Подсистема управления ТСП

В состав эквайерского модуля включена подсистема управления ТСП, обеспечивающая обслуживание торгово-сервисных предприятий. Доступ к порталу ТСП предоставляется для как для сотрудников банка (администраторы), так и для сотрудников ТСП, поддерживающих операции электронной коммерции. Подсистема управления ТСП выполняет следующие функции:

1. Просмотр истории транзакций по каждому ТСП, поиск и установку фильтров, формирование детализированной отчетности по транзакциям;
2. Управление пользователями системы (администраторы, сотрудники банка и ТСП, пользователи API), в том числе:
  - 2.1. Настройка прав доступа и управление ролями пользователей;
  - 2.2. Управление паролями пользователей;
3. Возможность формирования отчетов по операциям электронной коммерции для банка и ТСП в текстовом формате и в формате excel;
4. Возможность создания и обработки ручных операций (отмены, компенсации, завершения авторизаций);
5. Мониторинг транзакций в режиме online;
6. Настройка справочников карточных BIN-ов, IP-адресов и справочников стран;
7. Поддержка мультиязычного интерфейса;
8. Системный аудит.



### 3.3 Поддержка платежных Web-страниц

Платежная web-страница используется для взаимодействия с держателем карты в сети Интернет в процессе обработки операций электронной коммерции. Основные функции платежной web-страницы:

1. Обеспечение безопасной обработки платежа;
2. Обеспечение ввода информации по карте;
3. Отображение результатов обработки платежа.

Предоставляется стандартный шаблон платежной web-страницы, а также поддерживается загрузка в эквайерский модуль брендированных платежных страниц ТСП. Платежные web-страницы ТСП разрабатываются в соответствии с требованиями безопасности, сформулированными со стороны МПС Visa/MasterCard, а также PCI SSC.

### 3.4 Сервер Merchant Plug-in (MPI)

В состав подсистемы включен сервер Merchant Plug-in (MPI) который используется для эквайринговой обработки платежей электронной коммерции в рамках технологии 3-ДС (Verified by Visa, MasterCard Secure Code).

Основные функции Merchant Plug-in (MPI) – это поддержка бизнес-логики обработки платежей, а также протоколов взаимодействия с системами Visa/MasterCard в рамках технологии 3-ДС (Verified by Visa, MasterCard Secure Code).

Merchant Plug-in (MPI) обеспечивает интерфейс API с системой SmartVista Фронт Энд Рус для обработки платежей, а также проверку криптографических величин (CAVV, AAV), взаимодействуя с Directory Servers Visa/MasterCard.

## Установка приложения

### Создание домашней папки MPI

Для работы MPI необходимо создать домашнюю папку, в которой будут храниться его настройки и логи.

Например, /home/appserver/mri\_home. Далее будем называть эту папку <mri\_home>.

В папке <mri\_home> необходимо создать две папки conf и logs.

В папке conf должны находиться два файла:

mri.properties (файл с параметрами приложения)

logback.xml (файл с настройками логирования)

## Указание пути к домашней папке

Путь к домашней папке задается в виде системной переменной Java. Для этого нужно добавить в файле <domain>/bin/setDomainEnv.sh строки:

```
JAVA_OPTIONS="$ {JAVA_OPTIONS} -DMPJ_HOME=<mpi_home>"  
export JAVA_OPTIONS
```

Чтобы данное изменение вступило в силу, необходимо перезапустить сервер приложений.

## Создание пользователя БД

Для работы MPI необходимо создать отдельного пользователя БД, чтобы для него создать объекты БД, которые MPI использует для работы. Далее будем называть это "базой MPI".

Объекты создаются скриптом scheme.sql, который передается вместе с MPI.

## Создание источника данных

На сервере приложений, на котором будет разворачиваться MPI, необходимо создать источник данных, который будет смотреть на базу MPI.

Единственной обязательной характеристикой создаваемого источника данных является то, что JNDI-имя должно быть следующим: **jdbc/MPIDataSource**

Для этого необходимо нажать ссылку **Источники данных**, выбрать вариант **Новый источник** и на открывшейся странице заполнить следующие поля:

Name  
JNDI Name  
Database Type  
Database Driver

После этого нажать кнопку **Продолжить**, далее оставить во всех полях значения по умолчанию и еще раз нажать **Продолжить**.

На открывшейся странице необходимо заполнить следующие поля:

- Database Name                      имя базы данных
- Host Name                            сетевой адрес сервера (IP или сетевое имя)
- Port                                    порт
- Database User Name                имя учетной записи в БД
- (Confirm) Password                пароль учетной записи в БД

Нажать еще раз кнопку **Продолжить** и далее на открывшейся странице нажать кнопку **Проверка связи**. Если появится надпись зеленого цвета **Соединение успешно установлено**, -- подключение настроено корректно. Нажать кнопку **Продолжить**.

Если появится надпись красного цвета **Соединение отсутствует**, то настройка выполнена некорректно, и необходимо вернуться на предыдущий шаг, чтобы исправить ошибку.

Выделите флажком сервер, на котором будет установлено подключение (например, **AdminServer**). Завершите настройку источника данных кнопкой **Завершить**.

## Регистрация ключей и сертификатов

Для работы MPI нужно иметь два набора ключей и сертификатов, используемых для установления SSL-соединений с Directory-серверами VISA и MasterCard.

Данные ключи и сертификаты должны храниться в JKS-хранилище, путь к которому и пароль задаются параметрами `sslKeyStoreFile` и `sslKeyStorePassword`

### а. Создание хранилища:

Для создания пустого хранилища нужно выполнить две команды. Нужно запомнить пароль создаваемого хранилища, он понадобится в дальнейшем:

```
keytool -genkey -alias foo -keystore bpсmpi.jks
```

```
keytool -delete -alias foo -keystore bpсmpi.jks
```

### б. Импорт доверенных корневых и промежуточных сертификатов МПС

Для импорта доверенных корневых и промежуточных сертификатов МПС (обычно по два сертификата для каждой МПС) нужно выполнить команду:

```
keytool -import -trustcacerts -alias "<разумное название доверенного сертификата>" -file <файл импортируемого сертификата>.crt -keystore bpсmpi.jks
```

### с. Для каждой МПС сгенерировать пару клиентских ключей, для каждого ключа сгенерировать запрос на подпись сертификата, импортировать подписанный сертификат

Для генерации пары ключей нужно выполнить команду:

```
keytool -genkey -alias <название ключа в хранилище> -keyalg RSA -keystore bpсmpi.jks -keysize 2048
```

*(при генерации ключей для прохождения VISA PIT тестов нужно указывать длину ключа 1024)*

Для генерации запроса на подпись сертификата нужно использовать команду:

```
keytool -certreq -alias <название ключа в хранилище> -keystore keystore.jks -file <название файла с запросом подписи сертификата>.csr
```

Далее `csr` файл нужно передать МПС, чтобы там его подписали. МПС должна вернуть подписанный сертификат. Пусть файл называется `signed.crt`. Теперь его нужно импортировать в хранилище. *Важно иметь в виду, что `alias` при импорте должен совпадать с `alias`-ом при генерации пары ключей.* Команда:

```
keytool -import -keystore bpсmpi.jks -file signed.crt -alias <название ключа в хранилище>
```

## Установка приложения

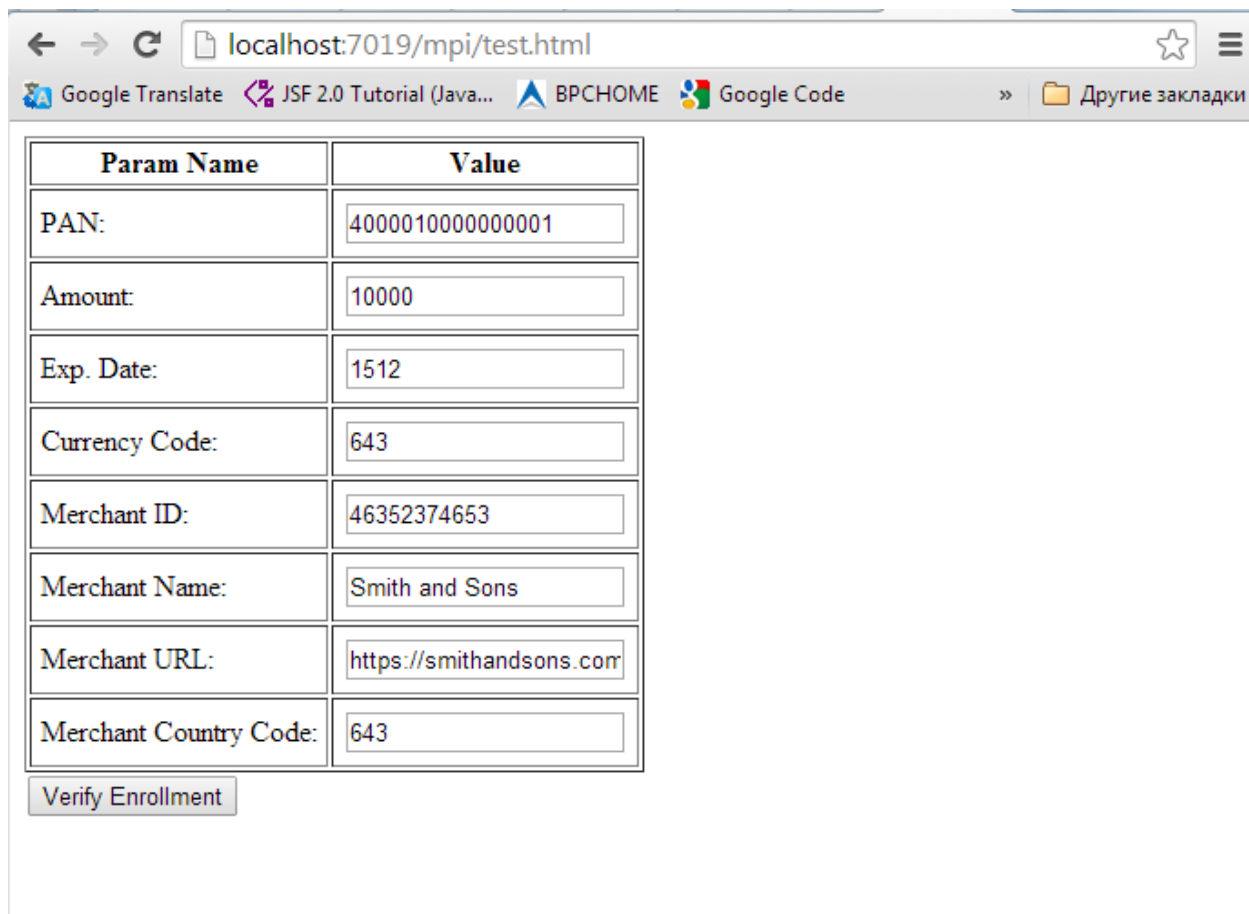
После того как все настройки выполнены можно приступить к установке приложения на сервер.

- a. Сохранить файл предыдущей сборки MPI.
- b. Войти в консоль сервера приложений.
- c. Войти в пункт меню **Операции установки**
- d. Отметить старое приложение MPI и нажать **Остановить**.
- e. Приложение должно перейти в состояние **Готово**. После этого нужно его снова отметить галочкой и нажать **Удалить**.
- f. Теперь старая сборка удалена. Для установки новой сборки нажать кнопку **Установить**.
- g. На открывшейся форме нажать ссылку **Загрузить файл(ы)**.
- h. Выбрать файл в поле **Архив для установки**.
- i. Выбрать файл с новой сборкой и нажать ссылку **Открыть**
- j. Нажать **Продолжить**.
- k. После загрузки файла на сервер снова нажать **Продолжить**.
- l. Выбрать пункт **Установить как приложение** и нажать **Продолжить**.
- m. На открывшейся форме нажать **Продолжить** еще раз.
- n. На открывшейся форме выбрать пункт **Я просмотрю конфигурацию позднее** и нажать **Завершить**.
- o. Приложение должно перейти в статус **Активно**.

## Проверка работоспособности

Для проверки работоспособности необходимо сделать следующее:

- a. Убедиться, что приложение находится в состоянии **Активно**.
- b. Открыть в браузере адрес `http://<appserver_ip>:<appserver_port>/mpi/test.html` . Должна открыться следующая форма:



Param Name	Value
PAN:	<input type="text" value="4000010000000001"/>
Amount:	<input type="text" value="10000"/>
Exp. Date:	<input type="text" value="1512"/>
Currency Code:	<input type="text" value="643"/>
Merchant ID:	<input type="text" value="46352374653"/>
Merchant Name:	<input type="text" value="Smith and Sons"/>
Merchant URL:	<input type="text" value="https://smithandsons.com"/>
Merchant Country Code:	<input type="text" value="643"/>

С помощью этой формы можно сделать запрос проверки вовлеченности карты к MPI. Для этого можно указать номер заведомо вовлеченной карты, а также при необходимости изменить прочие параметры запроса, затем нажать кнопку «Verify enrollment».

- c. Если всё работает корректно, то должна открыться страница с результатом проверки вовлеченности карты:



На данной странице отображается ответ MPI в формате JSON, который MPI отправил бы при вызове его API со стороны ПО ТСП.

## 4. Форматы запросов и ответов при взаимодействии ТСП с MPI

### 4.1.1. Запрос проверки вовлеченности

**Формат:** HTTP POST

**URL:** [https://<mpi\\_ip>:<mpi\\_port>/mpi/verifyEnrollment.do](https://<mpi_ip>:<mpi_port>/mpi/verifyEnrollment.do)

**Параметры запроса** (все параметры являются обязательными, если не указано обратное):

pan – номер карты

amount – сумма покупки, в копейках

expiry – срок действия карты в формате YYMM

currency – числовой код валюты операции

merchantId – идентификатор мерчанта, заданный эквайером

merchantName – название мерчанта

merchantUrl – интернет-адрес мерчанта

merchantCountryCode – числовой код страны мерчанта

merchantPassword – пароль мерчанта (*необязательный параметр*)

paymentSystemOverride – код платежной системы, для которой выполняется проверка (необязательный параметр). Данный параметр «пересиливает» платежную систему, вычисляемую на основании BIN карты. Допустимые значения: VISA, MASTERCARD, MIR.

**Формат ответа:** JSON

Поля ответа:

isEnrolled – флаг вовлеченности карты. Y – карта вовлечена, N – карта не вовлечена, U – неизвестно (скорее всего была ошибка при обработке запроса на стороне ACS). Поле может отсутствовать в случае ошибки.

error – ошибка при обработке запроса. Поле может отсутствовать в случае отсутствия ошибок.

acctId – значение acctId, возвращенное от ACS эмитента. Поле присутствует только в случае вовлеченности карты.



acsURL – значение AcsURL, возвращенное ACS эмитента. Поле присутствует только в случае вовлеченности карты.

pareq – запрос PAREq, сформированный MPI. (XML-документ, сжатый по алгоритму deflate, закодированный в Base64). Поле присутствует только в случае вовлеченности карты. Данное значение через HTML-форму должно передаться на ACS в параметре PaReq (шаги 8-9).

md – значение Merchant Data, сформированное MPI. Поле присутствует только в случае вовлеченности карты. Данное значение через HTML-форму должно передаться на ACS в параметре MD, затем оно должно вернуться на MPI.

xid – идентификатор транзакции, сформированный MPI. Поле присутствует только в случае вовлеченности карты. Строка Base64.

#### Примеры ответов:

*Ошибка при проверке вовлеченности карты на стороне ACS: {"isEnrolled":"U"}*

*Карта не вовлечена: {"isEnrolled":"N"}*

*Карта вовлечена:*

```
{ "xid" : "MDAwMDAwMDE0MDQ5MTk1OTkyMzg=", "isEnrolled" : "Y", "md" : "bdad51c8-5094-4b62-9193-42ae5d62e4c8", "pareq" : "eJxlUu9vgjAQ/VcI30ep/FJz1uhwmVnYnLJ976BRoi3agmP769cCzpmRNLn3+u6u9w6YNvxgnZ1URSkmNnZc22IiK/NCbCf2W/pwN7SnBNkdZCzesKyWjEDClKJbZhX5xOZq62CbwGq2ZicCfSWiCzkDQBeoU2S2o6IiQLPTfPlMfC+KIq3oIXAmlzHxQy8YeJEfBh6gJgJBOSMbXlQ7i4rc2pRCaWpJyMpaVPKLhL6WXwDU8kB2VXVUY4SUydNpSmc5WckR/SjrCpDRALq+alWbSOmaTZGTJJ59dmfhJvFrkKR7/JLuv5Lv7QSQUUBOK0YGLvbdyB1ZeDQehGNvBKjlgXLzGIJd/ekZOwRH02R2c/WXAu2u1OZf5rkgYM2xFEwrtF+/MeRMZf98aU1A13HuH43lWwXm9fHI98JQrgP89D4f0nzWLPj8XHMamEW0ItOs0C7iAHfdDABkyqB+x6jfv45u/osfc9DCRQ==", "acctId" : "44194366rR51KVB8adAxEmBvuma5", "acsURL" : "https://192.168.56.2/acs/pareq" }
```

### 4.1.2. Запрос проверки результатов аутентификации клиента

**Формат:** HTTP POST

**URL:** [https://<mpi\\_ip>:<mpi\\_port>/mpi/checkPARes.do](https://<mpi_ip>:<mpi_port>/mpi/checkPARes.do)

**Параметры запроса (все параметры являются обязательными):**

PaRes – значение параметра PaRes HTTP-запроса, пришедшее от ACS. (XML-документ, сжатый по алгоритму deflate, закодированный в Base64)

MD – значение параметра MD HTTP-запроса, пришедшее от ACS. Должно совпадать со значением поля md полученного от MPI в ответе на запрос проверки вовлеченности карты.

**Формат ответа: JSON**

Поля ответа:

status – статус (результат) аутентификации держателя карты: Y – успешно (также должно присутствовать поле CAVV), N – неуспешно (пользователь отменил операцию или ввел неправильные данные), U – не удалось провести аутентификацию (в связи с техническими или иными проблемами), A – проведена попытка аутентификации (также должно присутствовать поле CAVV).

xid – идентификатор транзакции, сформированный MPI. Строка Base64. Значение совпадает со значением в ответе на запрос проверки вовлеченности.

error – сообщение об ошибке в случае ошибки, может отсутствовать или принимать пустое значение.

checked – true/false – результат проверки транзакции.

eci – значение ECI (Electronic Commerce Indicator), полученное в PAREs от ACS.

cavv – значение CAVV, полученное от ACS.

cavvAlgorithm – алгоритм расчета CAVV, значение полученное от ACS.

**Примеры ответов:**

*Успешная аутентификация клиента:*

```
{ "status": "Y", "xid": "MDAwMDAwMDE0MDQ5Nzg0OTI1MjM=", "checked": true, "eci": "05", "cavv": "AAAAAAAAAAAAAAAAAOFAAAAAAAA=", "cavvAlgorithm": "2" }
```

*Пользователь отменил аутентификацию:*

```
{ "status": "N", "error": "", "checked": false }
```

*Ошибка при проведении аутентификации:*

```
{ "status": "U", "checked": false }
```

*Ошибка при проверке ЭЦП PAREs:*

```
{ "error": "Error in processing Signature PaRes", "checked": false }
```

*Неверное значение параметра MD:*

```
{ "error": "No transaction with MD=f4079b7b-8128-4a3c-ab75-fcacf26d46f7q found", "checked": false }
```

*Ошибка при парсинге PAREs:*

```
{ "error": "Failed to parse PaRes", "checked": false }
```

## Приложение 1: Конфигурация взаимодействия с криптомодулем

Для корректного взаимодействия с HSM в файле конфигурации HSM должны быть заданы значения следующих параметров:

Parameter	Description
<b>hsm.host</b>	Имя сервера или IP-адрес устройства.
<b>hsm.port</b>	Порт устройства.
<b>hsm.timeout</b>	Время ожидания ответа от криптомодуля (в секундах). Значение по умолчанию – 0, т.е. ограничение на время выполнения отсутствует.
<b>hsm.msg.header.length</b>	Длина заголовка сообщения. Значение по умолчанию: 4.

Пример файла:

```
hsm.host=192.168.103.127
hsm.port=6000
hsm.timeout=10
hsm.msg.header.length=4
```

Путь к файлу конфигурации с настройками взаимодействия с криптомодулем задается в системном параметре.

## Приложение 2: Запросы сертификатов для ACS

### VISA

В данном разделе описаны запросы сертификатов для платежной системы Visa.

#### Запрос сертификата для ACS SSL сервера для VISA

Запрашиваемый сертификат используется для входящих SSL-соединений от сервера VISA Directory к серверу ACS (серверу управления доступом).

Команда запроса сертификата:

```
openssl req -new -newkey rsa:2048 -nodes -keyout <new private key file name>.pem -out <certificate request file name>.pem
```

Пример:

```

[root@ACSWEB1 ds_incoming]# openssl req -new -newkey rsa:2048 -nodes -keyout
bank_acs_server_visa_prod_private_key.pem -out
bank_acs_server_visa_prod_cert_request.pem
Generating a 2048 bit RSA private key
.....
.....+++
.....+++
writing new private key to 'bank_acs_server_visa_prod_private_key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:RU
State or Province Name (full name) []:Moscow
Locality Name (eg, city) [Default City]:Moscow
Organization Name (eg, company) [Default Company Ltd]:BANK LLC
Organizational Unit Name (eg, section) []:Processing
Common Name (eg, your name or your server's hostname) []:3ds.bank.com
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@ACSWEB1 ds_incoming]#

```

Когда запрос подписан, закрытый ключ <new private key file name>.pem необходимо задать с помощью параметра `SSLCertificateKeyFile` в конфигурации прокси-сервера для соответствующего виртуального узла. Подписанный сертификат задается параметром `SSLCertificateFile`, а цепочка сертификатов – параметром `SSLCertificateChainFile`. Пример настройки ключа и сертификатов для Apache сервера:

```

SSLCertificateFile /etc/pki/acs/visa/pit_new_dns/ACSserver_certificate.pem

    SSLCertificateKeyFile
/etc/pki/acs/visa/pit_new_dns/bank_acs_server_visa_pit_private_key.pem

    SSLCertificateChainFile
/etc/pki/acs/visa/pit_new_dns/ACSserver_certificate_chain.pem

```

В данном примере файлы `ACSserver_certificate.pem` и `ACSserver_certificate_chain.pem` получены от системы VISA в ответ на запрос подписи сертификата.

## Запрос SSL клиентского сертификата ACS для VISA

Данный сертификат используется в исходящем SSL-соединения от сервера ACS к AHS-серверу (Authentication History Server) VISA.

Команда запроса сертификата:

```
openssl req -new -newkey rsa:2048 -nodes -keyout <new private key file name>.pem -out <certificate request file name>.pem
```

Пример:

```
[root@SVACS1 2014_09_02]# openssl req -new -newkey rsa:2048 -nodes -keyout
bank_ahs_client_visa_private_key.pem -out
bank_ahs_client_visa_cert_request.pem
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'bank_ahs_client_visa_private_key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:RU
State or Province Name (full name) []:Moscow
Locality Name (eg, city) [Default City]:Moscow
Organization Name (eg, company) [Default Company Ltd]:BANK LLC
Organizational Unit Name (eg, section) []:Processing
Common Name (eg, your name or your server's hostname) []:Verified By Visa
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Когда запрос подписан, <new private key file name>.pem и подписанный сертификат необходимо конвертировать в файл хранилища ключей PKCS12 с помощью команды

```
openssl pkcs12 -export -in <signed certificate file> -inkey <private key file>
-out <keystore name>.p12 -name "..."
```

Пример:

```
openssl pkcs12 -export -in ACSclient_certificate.pem -inkey
bank_ahs_client_visa_private_key.pem -out visa_ahs_identity.p12 -name
"visa_ahs_identity"
```

Также необходимо записать пароль, используемый для хранилища ключей и созданный с помощью команды `openssl pkcs12 ...`( например, "ahs\_identity\_password").

Кроме этого, в хранилище ключей доверенных сертификатов JKS-типа следует поместить два файла с сертификатами CA (certification authority), полученные от VISA (например, Visa\_eCommerce\_root.cer и Visa\_eCommerce.cer),

Для этого необходимо::

1. Создать пустое хранилище ключей:
  - `keytool -genkey -alias foo -keystore truststore.jks`
  - `keytool -delete -alias foo -keystore truststore.jks`

На данном шаге необходимо записать пароль от хранилища ключей. Например, "ahs\_trust\_password".

2. Добавить открытый сертификат в JKS-хранилище:

```
keytool -import -trustcacerts -alias "sensible-name-for-ca" -file CAcert.crt -keystore <keystore file>.jks
```

Пример:

(сначала идет корневой CA-сертификат, затем – CA-сертификат):

```
keytool -import -trustcacerts -alias "visa_ecommerce_root" -file Visa_eCommerce_root.cer -keystore truststore.jks
keytool -import -trustcacerts -alias "visa_ecommerce" -file Visa_eCommerce.cer -keystore truststore.jks
```

3. Указать созданные файлы в acs.properties:

- **ahs.identityStorePath=<path\_to\_visa\_ahs\_identity\_keystore>/visa\_ahs\_identity.p12**
- **ahs.identityStorePass=ahs\_identity\_password**
- **ahs.trustStorePath=<path\_to\_visa\_ahs\_trust\_keystore>/truststore.jks**
- **ahs.trustStorePass=ahs\_trust\_password**

## MasterCard

Команды и процедуры для MasterCard аналогичны командам и процедурам VISA.

Примеры команд приведены ниже.

## Запрос сертификата для сервера MC ACS

```
[root@ACSWEB1 ds_incoming]# openssl req -new -newkey rsa:2048 -nodes -keyout bank_acs_server_mc_prod_private_key.pem -out bank_acs_server_mc_prod_cert_request.pem
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'bank_acs_server_mc_prod_private_key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:RU
State or Province Name (full name) []:Moscow
Locality Name (eg, city) [Default City]:Moscow
Organization Name (eg, company) [Default Company Ltd]:BANK LLC
Organizational Unit Name (eg, section) []:Card IPG
Common Name (eg, your name or your server's hostname) []:securecode.bank.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
```

```
A challenge password []:  
An optional company name []:
```

Затем полученные файлы необходимо указать в конфигурации прокси-сервера для виртуального узла MasterCard так же, как и для VISA.

## Примеры вызовов

### Запрос сертификата подписи участника ACS для VISA

```
java -jar generator.jar -h 10.1.53.25 -p 5555 -t THALES -a SHA256withRSA -l  
2048 -n  
"C=US,ST=California,L=Sacramento,O=SomeBank,OU=Processing,CN=acs_signing_visa_  
prod"
```

При успешном выполнении программы выводится информация вида:

```
18:15:58.277 [main] DEBUG r.b.c.t.k.BpcThalesKeyPairGeneratorRsa -  
  
BpcThalesKeyPairGeneratorRsa#GenerateKeyPair Called  
18:15:58.277 [main] INFO r.b.c.t.comms.BpcThalesConnection - Trying to  
connect to Thales HSM with host: 10.7.1.81 and port: 8000  
18:15:58.277 [main] INFO r.b.c.t.comms.BpcThalesConnection - Command sent to  
hsm  
18:16:19.132 [main] INFO r.b.c.t.comms.BpcThalesConnection - Receiving reply  
with length bytes : 1 -17  
18:16:19.132 [main] INFO r.b.c.t.comms.BpcThalesConnection - Receiving reply  
with length : 495  
18:16:19.132 [main] INFO r.b.c.t.comms.BpcThalesConnection - Received reply:  
  
5A 31 64 33 45 4A 30 30 30 81  
88 02 81 80 AD 79 C4 62 14 18  
E9 C9 BA 2B 32 70 14 48 8C 7A  
F4 11 5C D5 74 89 C5 56 8E BF  
E4 1A 86 0B 58 5D ED 51 71 38  
62 AD 69 CB F5 79 CD D5 A3 7F  
...  
5C EF C5 24 00 1E 26 8C AA 8D  
FE 7C EF 18 D8 80 12 AD 74 C9  
59 43 EA DC 00  
  
18:16:19.132 [main] INFO r.b.c.t.k.BpcThalesKeyPairGeneratorRsa - Parsing  
Public Key from HSM output  
18:16:19.163 [main] INFO r.b.c.t.k.BpcThalesKeyPairGeneratorRsa - Finished  
parsing Public Key from HSM output  
18:16:19.179 [main] INFO r.b.c.t.k.BpcThalesKeyPairGeneratorRsa - Reading  
Private Key from HSM output  
18:16:19.179 [main] INFO r.b.c.t.k.BpcThalesKeyPairGeneratorRsa - Private  
Key length = 344  
  
BpcThalesSha1WithRsa#InitSign Called
```

```
18:16:19.194 [main] INFO r.b.c.t.comms.BpcThalesConnection - Trying to
connect to Thales HSM with host: 10.7.1.81 and port: 8000
18:16:19.194 [main] INFO r.b.c.t.comms.BpcThalesConnection - Command sent to
hsm
18:16:19.411 [main] INFO r.b.c.t.comms.BpcThalesConnection - Receiving reply
with length bytes : 0 -116
18:16:19.411 [main] INFO r.b.c.t.comms.BpcThalesConnection - Receiving reply
with length : 140
18:16:19.411 [main] INFO r.b.c.t.comms.BpcThalesConnection - Received reply:

6F 30 38 38 45 58 30 30 30 31
32 38 8E 89 1F 82 2C 1B 43 F8
B2 07 31 1A 42 DA 5B FE F3 44
77 5A 04 20 52 58 53 E8 AC 22
...
```

Файлы `private.pkf` и `csr.pem` генерируются в выходном каталоге:

- `private.pkf` — двоичный файл, который содержит секретный ключ, зашифрованный HSM. Этот ключ используется для подписи PARes-сообщений. (Далее его нужно определить в `acs.properties`.)
- `csr.pem` — содержит запрос подписи сертификата (Certificate Signing Request).

Чтобы получить цепочку сертификатов:

1. Копируйте текст из файла `csr.pem`.
2. Вставьте текст в специальную форму в эмуляторе Visa (<https://dropit.3dsecure.net/PIT/>).
3. Нажмите **Отправить (Submit)**.

**Примечание:** Данный пример описывает работу с эмулятором от VISA. В реальной рабочей среде отдельные шаги для получения цепочки сертификатов могут отличаться, но в целом последовательность такая, как приведена.



**Certificate Type:**

- MPI SSL Client Certificate (for authentication to DS)
- ACS SSL Client Certificate (for authentication to AHS)
- ACS SSL Server Certificate
- ACS Signing Certificate

**Email Address:**

**Cert Request (DER):**

- OR -

**Cert Request (PEM):**

```

-----BEGIN CERTIFICATE REQUEST-----
MIIBhzCB8QIBADBKMqwCQYDVQQGEwJVVUzEzMBcGA1UEChMQU3VvIE1pY3Jvc31z
dG9tZCERMA8GA1UECXMISmF2YVNVZnQxDALBgNVBAMTBER1a2UwgZ8wDQYJKoZI
hvcNAQEBBQADgY0AMIGJAoGBAKhWMLrvc1YCF5KuS0nt2SurYe7oVbpgO/n009ts
kk7DqtCACEIQcUZWFtBTY99pTyB3+oofy4RzEzVpxu6Q1kazGt1VsmcyWUId6vf
2XOR2Hv98GycdA2LRTJNxb9cUDuH+gESc1yGA/N5om/uHyL26WuwySmgCz6co8A
lyz1AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAfnTqMp5q/9gYKjyglBwWdQ16+SfC
cB51fwgz9dg/AF8XBfhju8pJsmmbSse4+5FUwd4E28ojO+ZV76UoWUqaAOc+CS
E3XmZ4us2c5KfvMaKdedyLOR2rsi8eT6IxcAERSvYfnleOQoWUNNkKtnedgVbK
Wk8EKgy1XOea02w=
-----END CERTIFICATE REQUEST-----

```

Сгенерированный и подписанный сертификат с цепочкой подлинности отобразится в форме. Сертификат отображается как текст в PEM-формате.

Your PEM encoded PKCS#7 chain is:

```

-----BEGIN PKCS7-----
MIIGIYJKoZIhvcNAQcCoIIghCCBoICAQExADALBgkqhkiG9w0BBwGgggZqMIIChzCCAyigAwIBAg
IULLsSuN65wJ0AJ3MkEqaVJe4ayrYwDQYJKoZIhvcNAQEFBQADgYEAfnTqMp5q/9gYKjyglBwW
dQ16+SfCcB51fwgz9dg/AF8XBfhju8pJsmmbSse4+5FUwd4E28ojO+ZV76UoWUqaAOc+CS
E3XmZ4us2c5KfvMaKdedyLOR2rsi8eT6IxcAERSvYfnleOQoWUNNkKtnedgVbK
Wk8EKgy1XOea02w=
-----END PKCS7-----

```

**Certificate Type:**

- MPI SSL Client Certificate (for authentication to DS)
- ACS SSL Client Certificate (for authentication to AHS)
- ACS SSL Server Certificate
- ACS Signing Certificate

4. Скопируйте сгенерированную цепочку (выделенная область на рисунке выше) и сохраните ее в какой-либо файл, например, в cert\_chain\_visa.pem.

## Запрос подписи сертификата MC ACS

Запросы ключа, сертификата и подписи сертификата ACS генерируются таким же образом, как и *Запрос сертификата подписи участника ACS для VISA*. Например:

```
java -jar generator.jar SAFENET 10.110.6.31 8892
./generated_keys_mc_prod_2048_2014_09_15/ "C=RU,ST=Moscow,L=Moscow,O=BANK
LLC,OU=Card IPG,CN=BANK_ACS_SIGNING_MC_PROD" 2048
```

Полученные файлы необходимо указать в `acs.properties` в соответствующих параметрах MasterCard. Например:

```
cryptography.certchain.path.mc=/home/acs_home/keys/signing/mc/12325.p7c
safenet.pk.path.mc=/home/acs_home/keys/signing/mc/private.pkf
```

## Ссылки на источники

- *VISA 3-D Secure Protocol Specification Core Function*
- *VISA 3-D Secure Functional Requirements Access Control Server*
- *MasterCard SecureCode Issue Implementation Guide*
- *Стандарт платежной системы «Мир». MirАсcept 2.0. Функциональные требования к 3DS Server*