Руководство по финансовым операциям и электронной подписи

Оглавление

Термины

Сервис

> Внешнее приложение, интегрированное с RooX UIDM и выполняющее запросы к нему.

Электронная подпись

> Информация, позволяющая однозначно определить лицо, подписавшее документ, а также буквенно-цифровая последовательность (хэш-сумма), которая однозначно соответствует определенному набору данных. Электронная подпись основана на хэше (hash) — алгоритме, позволяющем получить для определенного набора данных уникальную строку бит определенной длины.

Документ

> Строка бит произвольной конечной длины. Документ (пакет документов) состоит из тела документа (списка тел документов), а также метаданных документа (метаданных, общих для пакета документов в целом).
>
> Для целей настоящего документа термин `документ` означает также пакет документов (список тел документов + метаданные пакета документов).

Тело документа

> Строка бит, содержащая в себе существенную информацию об операции. Телом документа может быть как файл формата Adobe PDF или Microsoft Word (двоичные данные), так и текстовый файл(например, текстовый файл формата JSON, XML или YAML).

Метаданные документа

> Строка формата JSON (`"ключ"` : `значение`), соответствующая стандарту RFC 8259.

Одноразовый пароль (One-Time Password, OTP)

> Короткий код, генерируемый сервером и направляемый пользователю в ходе процедуры аутентификации. Основан на открытом стандарте RFC 4226 (HMAC-Based OTP Algorithm) от Open Authentication (OATH).
>
> По умолчанию RooX UIDM поддерживает направление OTP в SMS или электронной почтой. Возможна интеграция RooX UIDM с сервисом заказчика, обеспечивающим направление push-уведомлений на разрабатываемое заказчиком мобильное приложение.

Электронные подписи в RooX UIDM

При оказании банковских услуг или осуществлении внутрибанковского документооборота существует необходимость максимально сократить, а по возможности исключить любую форму бумажного документооборота. При этом у банка должна быть возможность неопровержимого доказательства наличия волеизъявления клиента на предоставление ему банковской услуги — например, исполнения платежного поручения — или подтверждения авторства электронного документа, созданного сотрудником банка. В этом случае собственноручную подпись можно заменить электронной подписью (при условии заключения соглашения между банком и клиентом или сотрудником банка).

Электронная подпись представляет собой последовательность конечной длины, полученную с использованием математических алгоритмов и связанную с электронным документом. Она обладает следующими свойствами:

- свойством авторства, которое позволяет идентифицировать автора электронного документа;
- свойством неотрекаемости, которое не позволяет автору отказаться от факта подписания электронного документа;
- свойством целостности, которое дает возможность определить факт внесения изменений в электронный документ после создания подписи.

RooX UIDM позволяет подписывать электронной подписью произвольные документы, в том числе финансовые. Такая электронная подпись в соответствии с действующим законодательством соответствует простой электронной подписи.

Если размер тела документа не превышает предельного значения, заданного в настройках RooX UIDM (по умолчанию этот размер задан в размере 2000 байт), то при выполнении операции (подписи) подписываемое тело документа сохраняется в базе данных RooX UIDM в исходной форме.

Если размер тела документа превышает заданное настройкой предельное значение размера, то при выполнении подписи в RooX UIDM сохраняется только хэш-сумма такого тела документа, рассчитанная при помощи хэш-функции, определенной в стандарте ГОСТ Р 34.11—2012 с длиной хэш-суммы 512 бит.

OTP-код генерируется на сервере RooX UIDM при помощи генератора случайных чисел. OTP-код имеет срок действия (время жизни). Каждое направленное SMS-сообщение нумеруется по счётчику; счётчик сбрасывается в полночь каждого дня.

При выполнении одной операции подписи может быть подписан комплект (пакет) документов.

В расчёте подписи документа используются:

- тело документа (при размере тела документа менее предельного значения размера) или его хэш-сумма (при размере тела документа более предельного);
- метаданные документа;
- номер телефона пользователя, записываемый в соответствии с рекомендацией E.164 Международного союза электросвязи, но без знака `+` (пример: `79001234567`);
- введённый пользователем OTP-код;
- порядковый номер SMS-сообщения запроса на подписание.

В качестве реквизитов подписавшего лица сохраняется телефонный номер подписавшего лица, номер попытки ввода OTP-кода, сам OTP-код, а также идентификатор клиента (пользователя) в RooX UIDM.

После направления документа на подписание, а также в результате успешного ввода OTP-кода возвращается уникальный идентификатор запроса на подписание документа (`extendedAttributes.signingRequestid` и `claims.sign_req_id`). В последнем случае этот идентификатор также помещается в выпускаемый токен одноразового доступа и помещается в БД аудита.

Включение возможности подписания документов на уровне политик

Включение необходимости подписания документа (которое выполняется при помощи токена одноразового

доступа) задаётся условием политики с заданным типом `PerOperationTokenCondition`.

Код политики, обеспечивающий требование подписания документа с использованием токена одноразового доступа

```
<Policy>
    <Conditions>
        <Condition name="perOperationToken" type="PerOperationTokenCondition">
            <AttributeValuePair>
                <Attribute name="required-if"/>
                <Value>true</Value>
            </AttributeValuePair>
            <AttributeValuePair>
                <Attribute name="require-signing"/>
                <Value>true</Value>
            </AttributeValuePair>
        </Condition>
    </Conditions>
</Policy>
```

Код политики, обеспечивающий требование подписания документа с использованием токена одноразового доступа при выполнении определенных условий

```
<Policy>
    <Conditions>
        <Condition name="perOperationToken" type="PerOperationTokenCondition">
            <AttributeValuePair>
                <Attribute name="required-if"/>
                <Value>( env['isFinal'] == 'Y' and env['fullForm'] == 'Y' )</Value>
            </AttributeValuePair>
            <AttributeValuePair>
                <Attribute name="require-signing"/>
                <Value>true</Value>
            </AttributeValuePair>
        </Condition>
    </Conditions>
</Policy>
```

Сценарии подписания документа электронной подписью

Сценарии подписания документов основаны на выпуске токена одноразового доступа.

Сценарий №1, в отличие от Сценария №2, требует повторного направления комплекта документов для выполнения подписи.

Предварительные требования

- пользователь аутентифицирован;
- в профиле пользователя указан номер телефона, позволяющий направить ему OTP-код.

Сценарий №1

Эта версия механизма подписи при первом направлении комплекта документов на подписание создаёт идентификатор запроса на подписание, который возвращается запрашивающей стороне, а также направляется ей же повторно после успешной подписи. Механизм также требует направления одного и того же документа дважды, и требует полной идентичности направляемых документов и порядка их следования по причине того, что контрольная

сумма (дайджест), используемая для проверки идентичности, является чувствительной к порядку следования бит в потоке данных.

Сценарий

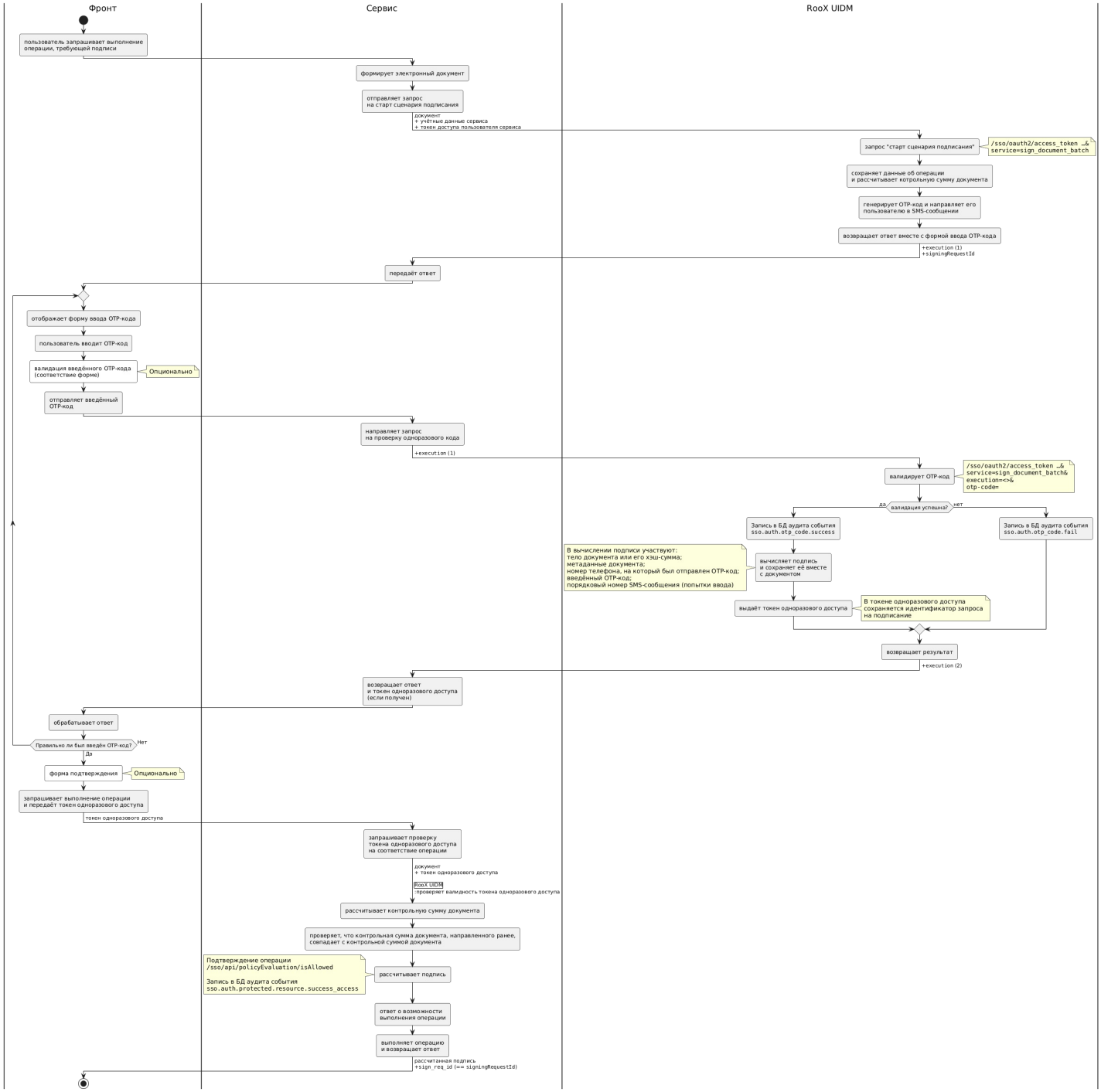
1. Пользователь сервиса инициирует операцию подписания документа или комплекта документов
2. Сервис формирует электронный документ, соответствующий запрашиваемой операции. Документ состоит из тела документа (основного содержимого) и метаданных документа.
3. Сервис отправляет в RooX UIDM запрос на старт процедуры подписания документа. Для этого сервис передаёт документ.

   В составе метаданных документа в том числе передаются:

   - токен доступа пользователя;
   - учётные данные сервиса.
4. RooX UIDM сохраняет документ полностью, если размер документа не превышает предельного размера, установленного соответствующей настройкой RooX UIDM.
5. RooX UIDM сохраняет метаданные документа.
6. RooX UIDM рассчитывает подпись документа и сохраняет её.
7. RooX UIDM назначает идентификатор для рассчитанной подписи и направляет его сервису.
8. RooX UIDM генерирует одноразовый пароль и направляет его пользователю.
9. Сервис отображает пользователю форму ввода одноразового пароля.
10. Пользователь вводит одноразовый пароль.

    Сервис может выполнить проверку корректности введенного одноразового пароля (соответствие одноразового пароля правилам: длины, допустимых символов и т. п.).
11. Сервис направляет RooX UIDM запрос на проверку введённого пользователем одноразового пароля.
12. В том случае, если одноразовый пароль верен, RooX UIDM направляет сервису токен одноразового доступа.

    При необходимости сервис запрашивает у пользователя подтверждение операции.
13. Сервис направляет RooX UIDM запрос на проведение операции и повторно передаёт RooX UIDM документ, а также выданный ранее токен одноразового доступа.
14. RooX UIDM проверяет, что токен одноразового доступа действителен и не был использован ранее.
15. RooX UIDM рассчитывает подпись документа, полученного на шаге **13.**
16. RooX UIDM проверяет, что подпись, рассчитанная на шаге **6**, совпадает с подписью, полученной на шаге **15.**
17. RooX UIDM записывает в БД аудита сообщение об успешном выполнении запрашиваемой операции.
18. RooX UIDM возвращает сервису сообщение об успешном выполнении запрашиваемой операции. В ответе содержится рассчитанная подпись, которая может быть передана пользователю (клиенту), нанесена на документы или сохранена сервисом для других целей.
19. RooX UIDM записывает в БД аудита сообщение об успешном доступе к защищаемому ресурсу.



Описанные далее запросы должны быть выполнены в последовательности Старт сценария подписания — Ввод OTP-кода — Подтверждение операции подписания документа, так как параметр `<execution>` из каждого ответа используется как параметр в последующих запросах.

ВАЖНО

Возвращаемое значение `<execution>` в каждом из запросов к RooX UIDM может обновляться. Необходимо использовать самое актуальное значение.

Старт сценария подписания

Для начала сценария подписания необходимо отправить запрос в RooX UIDM на адрес `/sso/oauth2/access_token`. В состав запроса входят подписываемые документы.

В ответе на этот запрос будет содержаться параметр `execution`, значение которого необходимо включить в следующий запрос к RooX UIDM.

В ответе также содержится описание формы ввода OTP-кода.

Запрос

```
POST /sso/oauth2/access_token

Host: <sso_host>
Accept: application/json
Content-Type: application/x-www-form-urlencoded

client_id=<client_id>&
client_secret=<client_secret>&
realm=<realm>&
grant_type=urn%3Aroox%3Aparams%3Aoauth%3Agrant-type%3Am2m&
service=sign_document_batch&
access_token=eyAiY3R5IjogIkpXVCIsICJ0eXAiO...&
operation=...
```

`<sso_host>`

> Базовый адрес сервера RooX UIDM, например `sso.rooxteam.com`. Имя сервера может содержать в себе порт: `sso.rooxteam.com:8080`.

`<client_id>`

> Идентификатор приложения-клиента. Возможные значения зависят от конфигурации.

`<client_secret>`

> Пароль приложения-клиента. Возможные значения зависят от конфигурации.

`<realm>`

> Группа пользователей RooX UIDM. Всегда используется значение `%2Fcustomer`, которое является uri-encoded значением `/customer`.

`<grant_type>`

> Способ авторизации пользователя:
>
> `urn:roox:params:oauth:grant-type:m2m`
>
> > для сценариев получения токена доступа, токена обновления доступа, токена автоматического входа.
>
> `client_credentials`
>
> > для сценария получения приложением-клиентом системного токена.

`<service>`

> Используемый сервис (цепочка аутентификации).
>
> В этом сценарии он всегда равен `sign_document_batch`.

`<access_token>`

> Текущий токен доступа (access token) пользователя.

`<operation>`

> Пакет для подписания (в формате JSON).

`<category>`

> Категория операции для выбора шаблона SMS-сообщения.

Тело запроса

```
{
  "actionName": "POST",
  "resourceName": "/payments/:id/sign",
  "realm": "/customer",
  "extraParams": {
    "meta1": "value1"
  },
  "signed_documents": [
    "id": "...document id...",
    "signed_document": "payload"
  ]
}
```

`actionName`

> Имя REST-метода API.

`resourceName`

> Логическое имя REST-ресурса API, согласуется при использовании функциональности.

`realm`

> Пользовательский реалм, например `/customer`.

`extraParams`

> Метаданные документа. Метаданные документа участвуют в расчете подписи.

`signed_documents`

> Массив подписываемых документов:
>
> `id`
>
> > Логический идентификатор или имя документа (для файлов можно передавать имя)

`signed_document`

> Тело документа. Для двоичных форматов тело документа передаётся закодированным по стандарту Base64.

Успешный ответ

В ответе содержится JSON объект, содержащий описание формы ввода OTP-кода, а также параметр `execution` со значением `<execution_value>`, который необходимо будет включить в следующий запрос к RooX UIDM.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

```
{
  "execution": "<execution_value>",
  "view": {
    "nextOtpCodePeriod": 9,
    "otpCodeAvailableAttempts": 6,
    "method": "SMS",
    "expireOtpCodeTime": 21599,
    "blockedFor": 0,
    "isBlocked": false,
    "otpCodeNumber": 4,
    "msisdn": "79989876549",
    "nextOtpPeriod": 9,
    "category": "otp-sign",
    "extendedAttributes": {
      "signingRequestId": "sso_____0efe12c4-d97f-4ffc-9a08-76998ccaec4e"
    },
    "geolocation": {
      "lat": {
        "valueDegrees": 49.849998
      },
      "lon": {
        "valueDegrees": 24.016666
      },
      "height": {
        "valueMeters": "NaN"
      }
    },
  },
  "form": {
    "name": "otpForm",
    "fields": {
      "otpCode": {
        "constraints": [
          {
            "name": "NotNull"
          },
          {
            "name": "Size",
            "attributes": {
              "min": 4,
              "max": 2147483647
            }
          },
          {
            "name": "Pattern",
            "attributes": {
              "flags": [],
              "regexp": "^[0-9]+$"
            }
          }
        ]
      }
    },
    "errors": []
  },
  "serverUrl": "<ignore>",
  "step": "enter_otp_form"
}
```

`execution`

> Значение этого параметра необходимо использовать для следующего запроса к RooX UIDM.

`step`

> Обозначение текущего шага сценария, в данном случае отображается форма ввода OTP-кода.

`form`

> Описание формы ввода OTP-кода.

`view`

> Информация о состоянии сценария, используется для отображения на UI:
>
> `method`
>
> > Как был отправлен текущий код (для данного сценария всегда `SMS`).
>
> `nextOtpCodePeriod`
>
> > Время в секундах до наступления возможности отправки нового OTP-кода.
>
> `otpCodeAvailableAttempts`
>
> > Количество оставшихся попыток ввода OTP-кода.
>
> `isBlocked`
>
> > Признак временной блокировки пользователя, ввод OTP-кода невозможен, следует заблокировать окно ввода.
>
> `blockedFor`
>
> > Если установлен параметр `view.isBlocked`, то поле содержит время до разблокировки пользователя в секундах.
>
> `msisdn`
>
> > Номер телефона, на который отправлен OTP-код.
>
> `otpCodeNumber`
>
> > Порядковый номер попытки ввода OTP-кода.
>
> `expireOtpCodeTime`
>
> > Время жизни OTP-кода в секундах. По истечении времени код будет считаться невалидным.
>
> `extendedAttributes.signingRequestId`
>
> > Уникальный идентификатор запроса на подписание документов.
> >
> > Этот идентификатор необходимо сохранить для запроса информации о состоявшейся операции подписания в дальнейшем (Проверка подписи).

Состав атрибута `form` указывает на то, что следует отобразить форму ввода OTP-кода и передать введенное значение OTP-кода в следующем запросе к API. Для атрибута `otpCode` установлено ограничение — значение не должно быть пустым, должно содержать ровно 4 символа, а его содержимое должно соответствовать регулярному выражению `^[0-9]+$` (в данном случае — содержать только цифры).

Следует сохранить `signingRequestId` в базе сервиса для ассоциации данных об операции на стороне сервиса и на стороне RooX UIDM.

Ввод OTP-кода

После ввода пользователем OTP-кода сервис отправляет OTP на проверку.

```
POST /sso/oauth2/access_token
Host: <sso_host>
Accept: application/json
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded

client_id=<client_id>&
client_secret=<client_secret>&
grant_type=urn:roox:params:oauth:grant-type:m2m&
realm=%2Fcustomer&
service=sign_document_batch&
execution=<execution_value>&
otpCode=<otpCode>&
_eventId=validate
```

`<sso_host>`

> Базовый адрес сервера RooX UIDM, например `sso.rooxteam.com`. Имя сервера может содержать в себе порт: `sso.rooxteam.com:8080`.

`<client_id>`

> Идентификатор приложения-клиента. Возможные значения зависят от конфигурации.

`<client_secret>`

> Пароль приложения-клиента. Возможные значения зависят от конфигурации.

`<realm>`

> Группа пользователей RooX UIDM. Всегда используется значение `%2Fcustomer`, которое является uri-encoded значением `/customer`.

`<grant_type>`

> Способ авторизации пользователя:
>
> `urn:roox:params:oauth:grant-type:m2m`
>
> > для сценариев получения токена доступа, токена обновления доступа, токена автоматического входа.
>
> `client_credentials`
>
> > для сценария получения приложением-клиентом системного токена.

`<service>`

> Используемый сервис (цепочка аутентификации).
>
> В этом сценарии он всегда равен `sign_document_batch`.

`<access_token>`

> Текущий токен доступа (access token) пользователя.

`<otpCode>`

> Введённый пользователем одноразовый пароль (OTP).

`<_eventId>`

> Идентификатор действия (перехода к следующему состоянию сценария). Определяется отдельно для каждого состояния.
>
> В данном запросе всегда равно `validate`.

`<execution>`

> Идентификатор предсессии аутентификации. В каждом конкретном запросе для продолжения сценария это значение должно быть равно значению поля `execution` из предыдущего ответа сервера на запрос к API.

В зависимости от правильности введенного OTP-кода ответ RooX UIDM будет указывать на способ продолжения сценария:

- `step=enter_otp_form` и непустой `form.errors` - код введен неверно или произошла другая ошибка, остаемся на текущем этапе сценария.

Успешный ответ

В ответе содержится токен одноразового доступа, который следует использовать для подтверждения действия пользователя.

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
Set-Cookie: execution=<execution_value>;Version=0;Path=/;Secure; SameSite=Lax; HttpOnly
```

```
{
  "access_token":
"eyAiY3R5IjogIkpXVCIsICJ0eXAiOiAiand0IiwgImVuYyI6ICJBMTI4Q0JDX0hTMjU2IiwgImFsZyI6ICJSU0FFU
19QS0NTMV9WMV81IiB9...",
  "claims": {
    "sign":
"g6kp//pNhWe9A7WVbvJFdnS1tKvhr9/yFyIkD1gpM9cGYJBHKjyVTVkksStegnedDG9lqDlh7hJ7LnIfs0/g+w=="
,
    "sign_req_id": "sso_____0efe12c4-d97f-4ffc-9a08-76998ccaec4e"
  },
  "expires_in": 1199
}
```

`<execution_value>`

> Значение, направленное на предыдущем шаге.

`claims.sign`

> Подпись.

`claims.sign_req_id`

> Дублируется идентификатор запроса на подписание пакета документов для случая, когда сервису удобно взять его из этого ответа.

`access_token`

> Токен одноразового доступа для выполнения запрашиваемой операции.

Наличие в ответе поля `access_token` говорит о том, что пакет сформирован, подписан и требует финального подтверждения операции пользователем для фиксации события в БД аудита безопасности.

На момент направления RooX UIDM ответа с токеном одноразового доступа событие в БД аудита еще не зафиксировано, так как предполагается, что приложение пользователя отобразит экран подтверждения действия.

Токен одноразового доступа можно использовать для подтверждения операции только один раз.

Подтверждение операции

Запрос

```
POST /sso/api/policyEvaluation/isAllowed

Host: <sso_host>
Accept: application/json
Content-Type: application/json
Authorization: Bearer {токен одноразового доступа, полученный на предыдущем шаге}
```

В заголовке `Authorization` передается полученный на шаге Ввод OTP-кода токен одноразового доступа.

Тело запроса

```
{
  "actionName": "POST",
  "resourceName": "/payments/:id/sign",
  "realm": "/customer",
  "extraParams": {
    ...
  },
  "signed_documents": [
    {
      "id": 0,
      "signed_document": "..."
    }]
}
```

В теле запроса должен быть передан объект с документами и метаданными, идентичный переданному на шаге Старт сценария подписания.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

```
{
  "decision": "Permit"
}
```

Наличие в ответе поля `"decision": "Permit"` однозначно указывает о том, что подписание выполнено успешно, информация зафиксирована в аудите безопасности.

Любые другие ответы должны интерпретироваться как отказ в операции. Если сервис не может обработать ответ, следует записать тело и заголовки ответа в лог и обратиться в техподдержку RooX UIDM.

```
HTTP/1.1 400 Bad Request
```

```
{
  "error": "invalid_grant",
  "error_description": "The provided access grant is invalid, expired, or revoked."
}
```

Не указан параметр `<_eventId>`

Формат ответа аналогичен ответу на шаге Старт сценария подписания.

Общие для всех шагов сообщения об ошибках

Таблица 1. Таблица возможных кодов ошибок RooX UIDM и их значений

Код ошибки	Описание причины возникновения
`email-exists`	Пользователь с заданным адресом электронной почты уже существует.
`email_verification_failed`	Невозможно подтвердить адрес электронной почты.
`error`	Пользователь не был аутентифицирован в результате неожиданного ответа от внешней системы.
`error_password_change`	Сценарий изменения пароля не был успешно завершен. Пароль не был изменен.
`error_sending_otp`	Невозможно отправить OTP пользователю.
`expired_password`	Время действия пароля, введенного пользователем, истекло. Пользователь не был аутентифицирован.

`external-api-error`	Выполнение сценария невозможно, поскольку получен неожиданный ответ от внешней системы или внешняя система не доступна.
`external-system-bad-response`	Выполнение сценария невозможно, поскольку получен неожиданный ответ от внешней системы или внешняя система не доступна.
`invalid_captcha`	Введенная пользователем CAPTCHA ошибочна.
`invalid_credentials`	Пользователь ввел неправильные данные учетной записи и не был аутентифицирован.
`invalid_otp`	Введенный OTP ошибочен.
`ip_blocked`	IP-адрес, с которого пользователь обращается в RooX UIDM, заблокирован.
`login-by-otp-disabled`	Пользователь попытался аутентифицироваться через сценарий с аутентификацией по OTP, но это невозможно, поскольку данный функционал отключен конфигурационным ключом.
`login-exists`	Невозможно зарегистрировать пользователя с заданным именем учётной записи, поскольку пользователь с заданным именем учётной записи уже существует.
`login_already_exists`	Невозможно изменить имя учётной записи пользователя на заданное, поскольку пользователь с заданным именем учётной записи уже существует.
`msisdn-is-not-b2b`	Введенный `msisdn` не принадлежит реалму `b2b`.
`msisdn-not-exists`	Пользователя с заданным `msisdn` не существует.
`need_captcha`	Для продолжения сценария пользователь должен ввести CAPTCHA.
`no-principal-found`	Пользователь не существует.
`no_email_found`	Пользователь не найден или у пользователя не существует адреса электронной почты.
`otp-error`	Сценарий ввода OTP завершился неуспешно.
`otp-expired`	Время действия введенного OTP истекло.
`otp_expired`	Время действия введенного OTP истекло.

`principal-not-exists`	Пользователь не существует.
`reset_required`	Пароль или не был задан либо был сброшен в результате каких-либо действий. Необходимо создать новый пароль. Отличается от `expired_password`, поскольку в том случае пароль существует и был провалидирован, но требуется его замена. В данном случае пароль не провалидирован, поскольку не задан.
`too_many_attempts`	Превышено число допустимых попыток изменения пароля.
`too_many_sms`	Невозможно отправить OTP пользователю, поскольку превышено число допустимых повторных запросов отправки OTP. Примечание: OTP может быть отправлен пользователю произвольным способом, а не только через SMS-сообщение.
`too_many_wrong_code`	Превышено число допустимых попыток ввода OTP.
`user-exists`	Невозможно зарегистрировать пользователя с заданными `msisdn`, адресом электронной почты или именем учётной записи, поскольку пользователь с такими параметрами уже существует.
`user-is-not-allowed`	Данному пользователю запрещено продолжать данный сценарий.
`user-not-found`	Пользователь не существует.
`user_blocked`	Учетная запись пользователя заблокирована.
`validate-otp-fail`	Введенный OTP ошибочен.

Примеры ответов об ошибке

Не указан или передан пустой параметр `execution`

Формат ответа с ошибкой:

```
HTTP/1.1 400 Bad Request
```

```
{
```

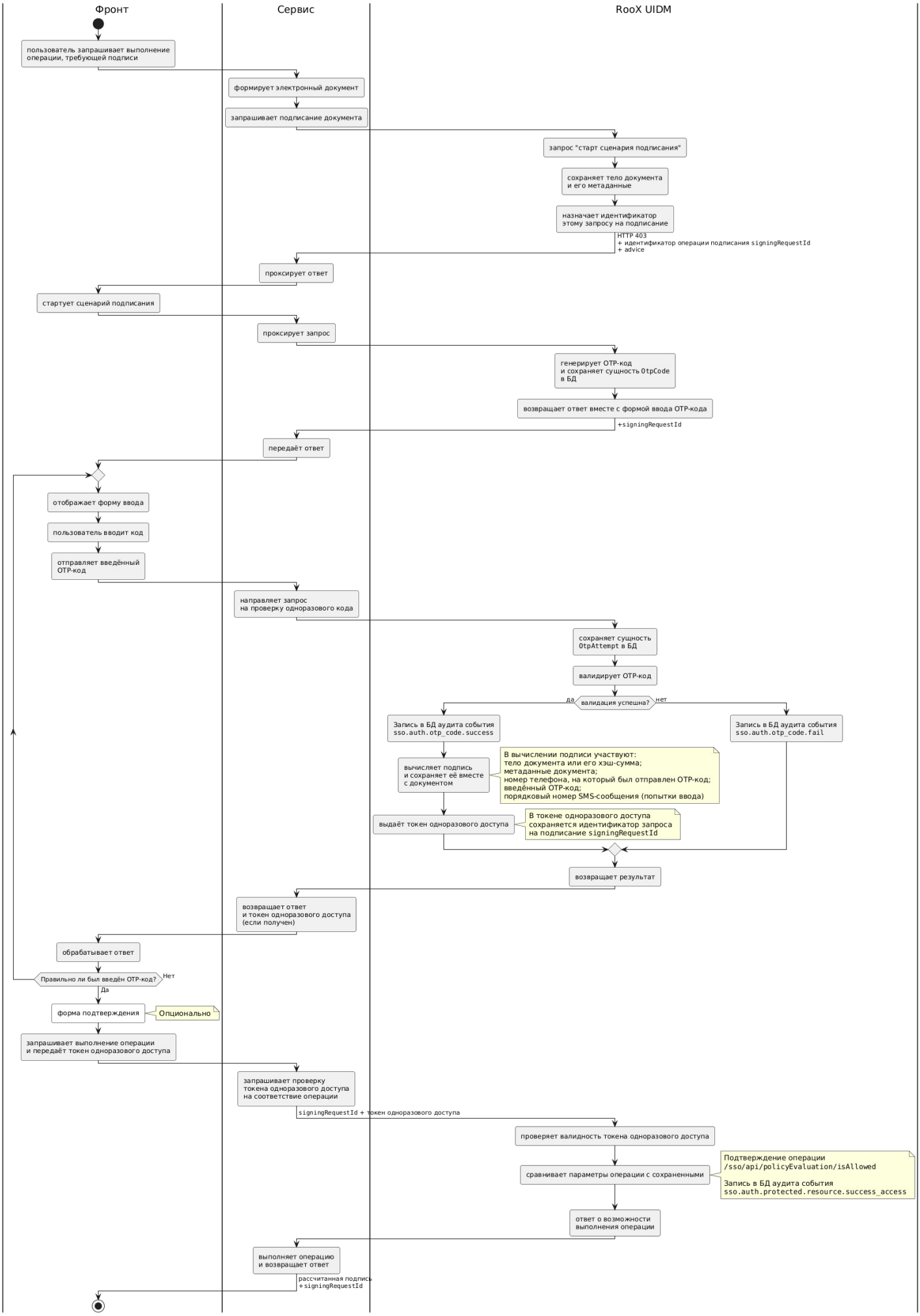
```
  "error": "invalid_grant",
  "error_description": "The provided access grant is invalid, expired, or revoked."
}
```

Не указан параметр `<_eventId>`

Формат ответа аналогичен ответу на шаге Старт сценария подписания.

Сценарий №2

Новая версия механизма подписи при первом направлении комплекта документов на подписание создаёт идентификатор запроса на подписание, который возвращается запрашивающей стороне, а также направляется повторно после успешной подписи. После получения токена одноразового доступа запрашивающей стороне для подписания комплекта документов достаточно предоставить только идентификатор запроса вместе с таким токеном одноразового доступа без необходимости направлять все подписываемые документы повторно.



Старт сценария подписания

Для начала сценария подписания одного или нескольких документов электронной подписью сервис направляет RooX UIDM подписываемые документы и передаёт в заголовке `Authorization` POST-запроса выданный ранее токен доступа.

Такой запрос запускает проверку политики доступа (см. выше) и одновременно создаёт запрос на подписание документов.

Запрос подписания документов электронной подписью

```
POST /sso/api/policyEvaluation/isAllowed
Accept: application/json
Authorization: Bearer b3adc19a-f54d-41ad-b9e9-e0742cb8ce9b
Content-Length: 31984
Content-Type: application/json
```

```
{
  "actionName": "POST",
  "serviceName": "iPlanetAMWebAgentService",
  "resourceName": "<имя операции, для которой выполняется подписание документа>",
  "realm": "/customer",
  "envParams": {
    "envParam_id1": "<значение параметра контекста № 1>",
    "envParam_id2": "<значение параметра контекста № 2>",
    "envParam_id3": "<значение параметра контекста № 3>",
    ... <дополнительные параметры>
  },
  "signed_documents": [
    {
      "id": 0,
      "signed_document": "<подписываемый документ № 0>"
    },
    {
      "id": 1,
      "signed_document": "<подписываемый документ № 1>"
    },
    {
      "id": 2,
      "signed_document": "<подписываемый документ № 2>"
    },
    ... <подписываемые документы>
  ]
}
```

Поле `serviceName` является для данного сценария константой. При использовании RooX UIDM SDK Java запрос сформируется автоматически на основе аннотации метода.

Поскольку операция подписания документов чаще всего требует повышения уровня полномочий пользователя (повышения уровня авторизации), RooX UIDM отклоняет запрос на подписание: сообщает решение (`decision`), а также действия, которые необходимо выполнить для повышения уровня авторизации пользователя — совет или советы (`advices`).

Ответ на запрос подписания документов электронной подписью

```
HTTP/1.1 403 Forbidden
```

```
Content-Type: application/json;charset=UTF-8
```

```
{
  "decision": "Deny",
  "advices": {
    "PerOperationTokenConditionAdvice": "PerOperationTokenRequired",
    "SigningRequiredAdvice": "sso_____f7758a1f-286c-4065-8a2c-5fcf99678177"
  }
}
```

Параметр `advices.SigningRequiredAdvice` — уникальный идентификатор созданного запроса на подписание пакета документов. Его может быть полезно сохранить для дальнейшего прохождения сценария.

Запрос и ввод OTP-кода

Для повышения уровня авторизации пользователя необходимо:

- направить запрос одноразового пароля (OTP-кода);
- получить одноразовый пароль;
- направить правильный одноразовый пароль RooX UIDM.

Запрос одноразового пароля (OTP-кода)

```
POST /sso/oauth2/access_token
Content-Type: application/x-www-form-urlencoded

client_id=<client_id>&
client_secret=<client_secret>&
access_token=b3adc19a-f54d-41ad-b9e9-e0742cb8ce9b&
realm=%2Fcustomer&
grant_type=<grant_type>&
service=sign_document_batch&
signingRequestId=sso_____f7758a1f-286c-4065-8a2c-5fcf99678177
```

`<client_id>`

> Идентификатор приложения-клиента. Возможные значения зависят от конфигурации.

`<client_secret>`

> Пароль приложения-клиента. Возможные значения зависят от конфигурации.

`<access_token>`

> Текущий токен доступа (access token) пользователя.
>
> Может быть получен в результате авторизации пользователя.

`<realm>`

> Группа пользователей RooX UIDM. Всегда используется значение `%2Fcustomer`, которое является uri-encoded значением `/customer`.

`<grant_type>`

> Способ авторизации пользователя:
>
> `urn:roox:params:oauth:grant-type:m2m`
>
> > для сценариев получения токена доступа, токена обновления доступа, токена автоматического входа.
>
> `client_credentials`
>
> > для сценария получения приложением-клиентом системного токена.
> >
> > В этом сценарии всегда `urn:roox:params:oauth:grant-type:m2m`.

`<service>`

> Используемый сервис (цепочка аутентификации).
>
> В этом сценарии `sign_document_batch`.

`signingRequestId`

> Идентификатор операции подписания документов.
>
> Равен параметру `advices.SigningRequiredAdvice` из ответа об отказе в совершении операции с текущим уровнем авторизации пользователя.

В ответ на запрос OTP-кода возвращаются в том числе: форма ввода OTP-кода, которую сервис должен отобразить пользователю, сведения об OTP-коде, а также `execution`.

Ответ с формой ввода OTP-кода, направленного пользователю заданным способом

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

```
{
  "execution": "<execution_value>",
  "view": {
    "nextOtpCodePeriod": 9,
    "otpCodeAvailableAttempts": 6,
    "method": "SMS",
    "expireOtpCodeTime": 119,
    "blockedFor": 0,
    "isBlocked": false,
    "otpCodeNumber": 20,
    "msisdn": "3210",
    "category": "otp-sign",
    "extendedAttributes": {
      "signingRequestId": "sso_____f7758a1f-286c-4065-8a2c-5fcf99678177"
    },
    "nextOtpPeriod": 9
  },
  "form": {
    "name": "otpForm",
    "fields": {
      "otpCode": {
        "constraints": [
          {
```

```
            "name": "NotNull"
          },
          {
            "name": "Size",
            "attributes": {
              "min": 4,
              "max": 2147483647
            }
          },
          {
            "name": "Pattern",
            "attributes": {
              "flags": [],
              "regexp": "^[0-9]+$"
            }
          }
        ]
      }
    },
    "errors": []
  },
  "serverUrl": "<server>/sso/auth/_otp-sms",
  "step": "enter_otp_form"
}
```


`execution`

> Идентификатор предсессии аутентификации.

`view.nextOtpCodePeriod`

> Время до наступления возможности направления нового одноразового пароля (в секундах). Значение идентично полю `view.nextOtpPeriod`.

`view.otpCodeAvailableAttempts`

> Количество попыток ввода одноразового пароля (OTP).

`view.method`

> Указывает на способ отправки OTP-кода (например, в других сценариях RooX UIDM код может отправляться на адрес электронной почты).

`view.expireOtpCodeTime`

> Срок действия одноразового пароля (в секундах).

`view.blockedFor`

> Время до разблокировки (в секундах).

`view.isBlocked`

> Признак блокировки пользователя.

`view.otpCodeNumber`

> Порядковый номер отправленного RooX UIDM OTP-кода. Глобальный счётчик отправленных сообщений сбрасывается ежедневно в 00:00.

`view.msisdn`

> Телефонный номер (MSISDN), на который было отправлено сообщение с одноразовым паролем (OTP).
>
> Телефонный номер маскируется перед передачей в соответствии с настройками `com.rooxteam.sso.masking.msisdn.search` и `com.rooxteam.sso.masking.msisdn.replace`. Если эти настройки заданы, то все вхождения подстроки из параметра `com.rooxteam.sso.masking.msisdn.search` будут заменены на значения, заданные в `com.rooxteam.sso.masking.msisdn.replace`.
>
> Если эти настройки не заданы, то будут переданы крайние правые цифры телефонного номера. Количество передаваемых цифр задано настройкой `com.rooxteam.uidm.masking.msisdn.characters.count`. Если эта настройка не задана, возвращаются четыре последних (правых) цифры телефонного номера.

`view.category`

> Указывает на тип сценария, в рамках которого выполняется проверка действительности OTP-кода.

`extendedAttributes.signingRequestId`

> Уникальный идентификатор запроса на подписание документов.

`view.nextOtpPeriod`

> Время до наступления возможности направления нового одноразового пароля (в секундах). Значение идентично полю `view.nextOtpCodePeriod`.

Введённый пользователем OTP-код направляется на проверку (валидацию).

Валидация OTP-кода

```
POST /sso/oauth2/access_token
Content-Type: application/x-www-form-urlencoded

_eventId=validate&
client_id=<client_id>&
client_secret=<client_secret>&
realm=%2Fcustomer&
execution=<execution_value>>&
grant_type=urn%3Aroox%3Aparams%3Aoauth%3Agrant-type%3Am2m&
service=sign_document_batch&
otpCode=6579
```

`<_eventId>`

> Идентификатор действия (перехода к следующему состоянию сценария). Определяется отдельно для каждого состояния.
>
> В данном запросе всегда равно `validate`.

`<execution>`

> Идентификатор предсессии аутентификации. В каждом конкретном запросе для продолжения сценария это значение должно быть равно значению поля `execution` из предыдущего ответа сервера на запрос к API.

`<service>`

> Используемый сервис (цепочка аутентификации).

> В этом сценарии `sign_document_batch`.

`otpCode`

> Атрибуты, относящиеся к одноразовому паролю (OTP).

В случае успешной валидации OTP-кода возвращается токен одноразового доступа (`access_token`). предназначенный для выполнения операции подписания документов.

Ответ с токеном одноразового доступа

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```


```
{
  "access_token": "759b0780-c70c-4057-80b1-0436bc47d553",
  "claims": {
    "executionId": "f08fbf9b-2dcf-425d-bac2-1f11de3cf187",
    "telephoneNumber": "79876543210",
    "sign":
"1hSaSf9yWjizcWNLioGD5b+OdA0Wg5YHkDh3ttHyn25Ba53L99UQs0pGa59M2o4rhFkmSVqs4/4PJLw2V0OCZw=="
,
    "sign_req_id": "sso_____f7758a1f-286c-4065-8a2c-5fcf99678177"
  },
  "token_type": "Bearer",
}
```


`<access_token>`

> Текущий токен доступа (access token) пользователя.
>
> В данном ответе в это поле помещается токен одноразового доступа, предназначенный для выполнения операции подписания документов.

`claims`

> Перечень клеймов выданного токена одноразового доступа.
>
> `claims.sign`
>
> > Подпись.
>
> `claims.sign_req_id`
>
> > Дублируется идентификатор запроса на подписание пакета документов для случая, когда сервису удобно взять его из этого ответа.

Выполнение подписания документов токеном одноразового доступа

При вызове этого метода RooX UIDM проверяет действительность (валидность) переданного токена доступа, а также то, что он был выдан именно для указанного набора документов.

```
POST /sso/api/policyEvaluation/isAllowed
```

```
Authorization: Bearer 759b0780-c70c-4057-80b1-0436bc47d553
Content-Type: application/json
```

```
{
  "actionName": "POST",
  "serviceName": "iPlanetAMWebAgentService",
  "resourceName": "<имя операции, для которой выполняется подписание документа>",
  "realm": "/customer",
  "envParams": {
    "envParam_1": "<значение параметра контекста № 1>",
    "envParam_2": "<значение параметра контекста № 2>",
    "envParam_3": "<значение параметра контекста № 3>",
    ... <дополнительные параметры>
  },
  "signed_documents": [
    {
      "id": 0,
      "signed_document": "<подписываемый документ № 0>"
    },
    {
      "id": 1,
      "signed_document": "<подписываемый документ № 1>"
    },
    {
      "id": 2,
      "signed_document": "<подписываемый документ № 2>"
    },
    ... <подписываемые документы>
  ]
}
```

После подписания документов токен одноразового доступа перестаёт быть действительным (инвалидируется).

Ответ об успешном подписании пакета документов

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

```
{
  "decision": "Permit"
}
```

Интроспекция использованного, с истекшим сроком действия или иным способом инвалидированного токена одноразового доступа невозможна; будет возвращен HTTP-код ошибки `401`.

Проверка подписи

Проверку подписи можно выполнить запросом на адрес вида `/sso/api/signingRequests/{signingRequestid}` с предоставлением токена доступа пользователя, где `{signingRequestId}` (идентификатор операции

подписания):

- Успешный ответ RooX UIDM на запрос о подписании, параметр успешного ответа `extendedAttributes.signingRequestId`
- Ответ RooX UIDM токеном одноразового доступа, параметр успешного ответа `sign_req_id`

Пример запроса проверки подписи с заданным идентификатором

```
POST /sso/api/signingRequests/sso_____3fcef82d-c6f8-46e2-a998-897c4edeeefb
Authorization: Bearer b3adc19a-f54d-41ad-b9e9-e0742cb8ce9b
```

Пример ответа RooX UIDM на запрос проверки подписи с заданным идентификатором


```
{
    "data" : {
        "id" : "sso_____3fcef82d-c6f8-46e2-a998-897c4edeeefb",
        "principalOwnerId" : "sso_____8e35eb06-1d36-40e9-a4af-2b91e92a64aa",
        "meta" : {
            "523a5029-e10a-4ac6-9aa5-9d4b2fecc008" : "8fcbfeb7-f629-4ca5-9f70-
9f70825a17db",
            "6f95c2f2-15fb-4944-ab14-1ed0b9ee94de" : "3655f176-99e9-4f70-9cc1-
b7369666e47b",
            "941b6eee-cdcb-4fe4-a1cd-cc42cf6d6630" : "ef521557-8fea-4afa-bace-
ea02b8aea50a",
            "bf5d77d8-790e-4d1d-adb9-4f6ceaf94d64" : "390704fd-172f-41a1-92c4-
ea6e6bb32282",
            "e2e58a60-a687-4efa-8803-8fcf8b13d5d1" : "480e5377-80e5-438c-a71b-
aa887536a248"
        },
        "creationTime" : 1633893913,
        "signatures" : [ {
            "id" : "sso_____ddc6cc5b-af7f-4759-ba5b-4b5e6849c765",
            "signingTime" : 1633893916,
            "hash" :
"FyTncQxafdzFoXHJ88GeNpKVr5u2YYULipY/PXNZ7N1pZilF0BU63Haq8Ivn+9vksby35Lr9scfzi5RDGn3pkQ=="
,
            "alg" : "OtpGost3411_2012_512",
            "principalSignerId" : "sso_____8e35eb06-1d36-40e9-a4af-2b91e92a64aa"
            "signingCredentials" : [ {
                "msisdn" : "79876543210"
            }, {
                "otpId" : "1"
            }, {
                "otpCode" : "4609"
            } ],
        } ]
    }
}
```


`data`

> Сведения о запросе (операции) подписания:
>
> `id`
>
> > Идентификатор запроса на подписание.

`principalOwnerId`

> Идентификатор владельца (principal ID — уникальный идентификатор учетной записи пользователя, запросившего подписание).

`meta`

> Дополнительные атрибуты (метаданные) запроса на вычисление политик (`policyEvaluation`), в том числе метаданные подписываемых документов — содержимое атрибута `extraParams` (см. тело запроса шага Старт сценария подписания).
>
> Эти атрибуты также попадают в контекст запроса на рассылку уведомлений (компонент webapi-noitifier) и таким образом могут использоваться в тексте направляемого уведомления (шаблона).

`creationTime`

> Время создания запроса на подписание.

`signatures`

> Список подписей в запросе.
>
> `id`
>
> > Идентификатор подписи.
> >
> > `signingTime`
> >
> > > Время подписания.
> > >
> > > `hash`
> > >
> > > > Вычисленное значение подписи.
> > > >
> > > > `alg`
> > > >
> > > > > Алгоритм расчёта подписи.
> > > > >
> > > > > `principalSignerId`
> > > > >
> > > > > > Идентификатор лица, подписавшего документ.
> > > > > >
> > > > > > `signingCredentials`
> > > > > >
> > > > > > > Реквизиты подписавшего лица:
> > > > > > >
> > > > > > > `msisdn`
> > > > > > >
> > > > > > > > Телефонный номер, на который был отправлен OTP-код для подписания.
> > > > > > > >
> > > > > > > > `otpId`
> > > > > > > >
> > > > > > > > > Порядковый номер кода доступа, который использовался для подписания.
> > > > > > > > >
> > > > > > > > > `otpCode`
> > > > > > > > >
> > > > > > > > > > OTP-код, введённый подписавшим лицом.

Подписание документов простой электронной подписью с использованием

RooX UIDM SDK Java

Для ускорения разработки своих приложений на Java или Kotlin, интегрируемых с RooX UIDM, разработчики могут использовать библиотеку RooX UIDM SDK Java, исходные коды которой расположены по адресу RooX UIDM SDK Java.

RooX UIDM SDK Java выполнен в формах «чистой» Java и с интеграцией в Spring.

Примеры использования RooX UIDM SDK Java размещены в репозитории RooX UIDM SDK Java Samples.

Фреймворк Spring, используемый в RooX UIDM SDK Java для взаимодействия с RooX UIDM API, использует механизм аннотаций Java, чтобы обеспечить роутинг запросов или инъекцию зависимостей.

Аннотация `@PreAuthorize` позволяет задать требования и условия совершения того или иного действия.

Третий параметр метода `isAllowed()`, передаваемого в аннотацию `@PreAuthorize` в примере ниже, управляет содержимым документа, отправляемого на подписание. В примере приведено направление на подпись всего тела запроса, однако существует возможность выделять и отправлять на подпись отдельные его части.

Пример метода контроллера для операции `payment` с аннотацией, которой устанавливается требование подписания документа электронной подписью

Для работы примера в RooX UIDM должна быть настроена политика с именем ресурса `/payment` и методом `POST`.

```
@RequestMapping(method = RequestMethod.POST, value = "/payment")
@PreAuthorize("isAuthenticated() && @uidmAuthz.isAllowed('/payment','POST',
{'body':#body})")
public Object payment(@RequestBody Object body, AuthenticationState authenticationState) {
    Map<String, Object> response = new HashMap<>();
    response.put("body", body);
    Optional.ofNullable(authenticationState)
            .map(AuthenticationState::getAttributes)
            .map(attrs -> attrs.get(EVALUATION_CLAIMS_ATTRIBUTE_NAME))
            .ifPresent(v -> response.put("claims", v));
    return response;
}
```

Чтобы получить от сервера `advices` (условия, которые необходимо выполнить для подписания документов электронной подписью, в том числе уникальный идентификатор запроса на подписание — `SigningRequiredAdvice`), необходимо добавить обработчик исключений:

Пример обработчика исключений для получения предусмотренных политикой условий подписи документов электронной подписью

```
@ExceptionHandler(AccessDeniedException.class)
public ResponseEntity<?> handleAccessDeniedException(AuthenticationState
authenticationState, AccessDeniedException e) {
    Map<String, Object> response = new HashMap<>();
    response.put("error", e.getMessage());
    Optional.ofNullable(authenticationState)
            .map(AuthenticationState::getAttributes)
            .map(attrs -> attrs.get(EVALUATION_ADVICES_ATTRIBUTE_NAME))
            .ifPresent(v -> response.put("advices", v));
    return ResponseEntity.status(HttpStatus.FORBIDDEN).body(response);
}
```

Начало сценария подписания документа со стороны SPA или мобильного приложения

```
POST http://localhost:8042/api/payment
Authorization: Bearer ba8b9609-8d24-4ec2-81bb-553f03b8816e
Content-Type: application/json
```

```
{
    "to": "40802810900001633906",
    "amount": "200.00",
    "currency": "RUB"
}
```

Запрос, направляемый в RooX UIDM для применения политики

```
POST /sso/api/policyEvaluation/isAllowed
Accept: application/json
Authorization: Bearer ba8b9609-8d24-4ec2-81bb-553f03b8816e
Content-Type: application/json
```

```
{
    "serviceName": "iPlanetAMWebAgentService",
    "actionName": "POST",
    "resourceName": "/payments/:id/sign",
    "envParams": {
        "body": {
            "to": "40802810900001633906",
            "amount": "200.00",
            "currency": "RUB"
        }
    },
    "extraParams": {
        "headers": {
            "host": [
                "..."
            ],
            "content-type": [
                "application/json"
            ],
            "connection": [
                "close"
            ],
            "accept-encoding": [
                "gzip, deflate"
            ]
        },
        "ip": "94.102.127.198",
        "httpMethod": "POST",
        "url": "/webapi-1.0/products/loans"
    },
    "realm": "/customer"
}
```

RooX UIDM отвечает отказом (HTTP-код `403`) и сообщает уникальный идентификатор этой операции подписания этого документа или пакета документов (`SigningRequiredAdvice`):

Отказ в совершении операции (HTTP-код `403`), вместе с уникальным идентификатором запроса на подписание (`SigningRequiredAdvice`)

```
{
    "decision": "Deny",
    "advices": {
        "PerOperationTokenConditionAdvice": "PerOperationTokenRequired",
        "SigningRequiredAdvice": "sso_____7fb91106-eff0-4503-bfe0-7f7895006e13"
    }
}
```

Приложение вызывает метод подписи запроса с использованием OTP-кода. В качестве `signingRequestId` (идентификатора запроса на подписание) используется полученный ранее `advices.SigningRequiredAdvice`.

Вызов метода подписания запроса OTP-кодом

```
POST /sign-operation/send
Host: api-domain.com
Authorization: Bearer ba8b9609-8d24-4ec2-81bb-553f03b8816e
Content-Type: application/x-www-form-urlencoded
signingRequestId=sso_____7fb91106-eff0-4503-bfe0-7f7895006e13
```

Методы `/sign-operation/*` входят в состав RooX UIDM SDK для запуска сценария `sign_document_batch` на подписание документов электронной подписью.

Для корректной работы методов необходимо, чтобы библиотека RooX UIDM SDK Java получила следующие настройки:

```
# URL RooX UIDM
com.rooxteam.aal.sso.endpoint=https://uidm-domain.com/sso
# `client_id` и `client_secret` приложения, от имени которого будут выполняться операции проверки политики и запроса токенов
com.rooxteam.aal.auth.client=smeportal_m2m
com.rooxteam.aal.auth.password=password

# Этот параметр указывает, что токен будет передаваться в параметр `access_token` для запросов запуска сценария `sign_document_batch`
com.rooxteam.aal.otp.token.current.name=access_token
```

Пример ответа WebAPI (HTTP-код `200`)

```
{
    "status": "OTP_REQUIRED",
    "otpFlowState": {
        "execution": "<execution_value>",
        "csrf": null,
        "serverUrl": "<server>/sso/auth/_otp-sms",
```

```
        "sessionId": "39B46EE50428A299222FFAE82F7F8B55"
    },
    "requiredFieldNames": [
        "otpCode"
    ],
    "availableAttempts": 6,
    "token": null,
    "blockedFor": 0,
    "nextOtpCodeOperationPeriod": 10,
    "otpCodeNumber": 1,
    "method": "SMS",
    "extendedAttributes": {
        "signingRequestId": "sso_____7fb91106-eff0-4503-bfe0-7f7895006e13"
    }
}
```


После получения OTP-кода для подписания документа направляется запрос на валидацию OTP-кода. В теле запроса направляется содержимое поля `otpFlowState` из предыдущего ответа.

Пример запроса к WebAPI для подписания документа с приложением OTP-кода

```
POST /sign-operation/validate?otpCode=9161
Host: api-domain.com
Authorization: Bearer e35b7baf-4179-4ce8-ab71-b5099a28faf7
Content-Type: application/json
```


```
{
    "execution": "<execution_value>",
    "csrf": null,
    "serverUrl": "<server>/sso/auth/_otp-sms",
    "sessionId": "39B46EE50428A299222FFAE82F7F8B55"
}
```


В случае успешной валидации OTP-кода возвращается токен одноразового доступа, который необходимо будет использовать для исполнения операции подписи документов электронной подписью.

Ответ об успешной валидации OTP-кода вместе с токеном одноразового доступа


```
{
    "status": "SUCCESS",
    "otpFlowState": null,
    "requiredFieldNames": null,
    "availableAttempts": null,
    "token": "c0e68bd4-eb3b-452d-88b5-3fd989bbdfcb",
    "blockedFor": null,
    "nextOtpCodeOperationPeriod": null,
    "otpCodeNumber": null,
    "method": null,
    "extendedAttributes": null
}
```

Токен одноразового доступа для подписания документов отправляется в заголовке `Authorization: Bearer` вместо «стандартного» токена доступа. Тело запроса (которое будет передано политике `PolicyEvaluation` для оценки) должно совпадать с направленным в первом запросе, иначе в доступе будет отказано.

В ответе `PolicyEvaluation` могут возвращаться специальные клеймы, которые конфигурируются параметрами. Эти значения в RooX UIDM SDK Java доступны через атрибут `evaluationClaims` объекта AuthenticationState.

Прочтите также

- Соответствие простой электронной подписи RooX UIDM требованиям законодательства
- Подписание документов простой электронной подписью