API подписи документов простой электронной подписью

Оглавление

API предназначено для подписания электронных документов или операций ПЭП.

Описание функциональности доступно по этому адресу.

Данное API используется только из серверных приложений и защищено:

1. Учетными данными системы, выполняющей запрос (например, WebAPI).
2. Текущим токеном доступа пользователя, выполняющего подписание.
3. На уровне WAL должны быть запрещены запросы из недоверенных сетей.

Сценарий подписания пакета документов с подтверждением одноразовым паролем

UIDM трактует понятие «документа» в широком смысле; это некоторый набор байтов и метаинформации. С пользовательской точки зрения это может быть так сканированный PDF-документ, так и электронный документ, то есть существенная информация, характеризующая операцию. В UIDM можно передавать и бинарные документы (PDF, TIFF) и структурированные (JSON, XML).

Термином «веб-приложение» называется серверный компонент, предоставляющий API конечному пользователю. Это может быть комбинация из SPA + WebAPI, мобильное приложение + WebAPI или полностью серверное приложение. Важно, что у приложения должна быть серверная часть.

Предусловия

- Пользователь аутентифицирован
- В профиле пользователя указан номер телефона для получения одноразового пароля (OTP)

Сценарий

1. Пользователь передает документы на подписание в WebAPI (допустимо промежуточное хранение документов где-то на стороне веб-приложения).
   a. Веб-приложение отправляет запрос на старт сценария подписания. Передается полный набор документов и метаданных.
   b. Сервер сохраняет данные об операции: документы (учитывая заметку о максимальном размере документа), метаданные, рассчитывает подпись и сохраняет в свою СУБД.
   c. Сервер генерирует код, отправляет его в SMS-сообщении пользователю и отвечает описанием формы ввода одноразового кода. Вместе с ответом отдается уникальный идентификатор подписания, чтобы веб-приложение могло ссылаться на него в дальнейшем.
   d. Веб-приложение отображает форму ввода одноразового кода.
2. Пользователь вводит одноразовый код, полученный в SMS.
   a. Приложение валидирует ввод.
   b. Приложение отправляет запрос на проверку одноразового кода.
   c. Сервер проверяет код.
      i. Если код введен неверно, сервер отвечает ошибкой и, возможно, предоставляет еще попытку ввода.
      ii. Если код введен верно, сервер отвечает токеном под операцию.
3. Веб-приложение может отобразить дополнительный шаг — подтверждение операции или сразу же перейти к следующему шагу.
   a. Приложение отправляет запрос на проведение операции, передавая токен под операцию и заново документы + метаданные.
   b. Сервер проверяет, что токен под операцию валиден и не был использован ранее.
   c. Сервер проверяет, что подпись, рассчитанная на шаге один совпадает с подписью, рассчитанной заново для переданных в этом запросе документов.
   d. Сервер записывает событие в БД Аудита `sso.auth.protected.resource.success_access`.
   e. Сервер возвращает успешный ответ. В ответе содержится рассчитанная подпись, которую можно вернуть клиенту или отобразить на документах.

Постусловия

- Пакет документов подписан, рассчитан хеш подписи
- В БД Аудита запротоколировано событие `sso.auth.protected.resource.success_access`
- При использовании одноразового пароля (OTP) как механизма подтверждения операции будут запротоколированы события `sso.auth.otp_code.success` или `sso.auth.otp_code.fail`

Замечания по работе сценария

Все запросы должны быть выполнены в приведенной последовательности, так как параметр `execution` из каждого ответа используется как параметр в последующих запросах.

ВАЖНО

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

Старт сценария

Для начала сценария подписания необходимо отправить запрос в UIDM на `/sso/oauth2/access_token`.

Следует передать подписываемые документы

В ответе будет содержаться параметр ‹execution›, который необходимо включить в следующий запрос к UIDM.

Так же в ответе содержится описание формы ввода идентификатора пользователя, который восстанавливает пароль.

Формат запроса

```
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=...
```

Параметры

- ‹sso_host› — базовый адрес сервера UIDM, например sso.rooxteam.com
- ‹client_id› — идентификатор клиента, например selfcare
- ‹client_secret› — пароль клиента, например selfcare_password
- ‹realm› — url-encoded пользовательский realm, например %2Fcustomer
- ‹grant_type› — способ интеграции, всегда используется значение urn:roox:params:oauth:grant-type:m2m
- ‹service› — используемый сервис. В этом сценарии 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

Структура пакета документов для подписания простой электронной подписью

```
{
  "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› — название реалма с пользователями
- ‹extraParams› — метаинформации о пакете документов в формате ключ-значение. Участвует в расчете подписи.
- ‹signed_documents› — массив подписываемых документов, далее описываются поля в элементе
- ‹id› — логический идентификатор или имя документа (для файлов можно передавать имя)
- ‹signed_document› — содержимое документа. Для двоичных форматов передавать содержимое в Base64.

Формат успешного ответа

В ответе содержится JSON объект, содержащий описание формы ввода одноразового пароля (OTP), а так же ‹execution›, который необходимо включить в следующий запрос к 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"
}
```


Поля ответа

- ‹execution_value› — значение параметра ‹execution› для следующего запроса к UIDM
- step — обозначение текущего шага сценария, в данном случае отображается форма ввода одноразового пароля (OTP)
- form — описание формы
- view — информация о состоянии сценария, используется для отображения на UI
- view.method* — как был отправлен текущий код, для данного сценария всегда: SMS
- view.nextOtpCodePeriod — время в секундах до наступления возможности отправки нового одноразового пароля (OTP)
- view.otpCodeAvailableAttempts — количество оставшихся попыток ввода одноразового пароля (OTP)
- view.isBlocked — признак временной блокировки пользователя, ввод одноразового пароля невозможен, следует заблокировать окно ввода
- view.blockedFor — если установлен isBlocked, то поле содержит время до разблокировки в секундах
- view.msisdn — номер телефона, на который отправлен одноразовый пароль
- view.otpCodeNumber — порядковый номер сообщения, сбрасывается ежедневно в 0:00
- view.expireOtpCodeTime — время жизни одноразового пароля (OTP) в секундах. По истечении времени код будет считаться невалидным
- view.extendedAttributes.signingRequestId — уникальный идентификатор подписания, нужен для запроса информации по нему в дальнейшем.

Состав формы говорит, что следует отобразить форму ввода одноразового пароля (OTP) и передать введенное значение в следующем запросе к API. Имя поля `otpCode`, ограничение — значение не должно быть пустым, содержать ровно 4 символа и соответствовать регулярному выражению `^[0-9]+$` (в данном случае — содержать только цифры).

Следует сохранить `signingRequestId` в базе веб-приложения для ассоциации данных об операции на стороне веб-приложения и данных на стороне UIDM.

Ввод одноразового пароля (OTP)

После ввода пользователем одноразового пароля (OTP) веб-приложение отправляет одноразовый пароль 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>&
otpCode=<otpCode>&
_eventId=validate
```

Параметры запроса

- ‹sso_host› — базовый адрес сервера UIDM, например sso.rooxteam.com
- ‹client_id› — идентификатор клиента, например selfcare
- ‹client_secret› — пароль клиента, например selfcare_password
- ‹realm› — url-encoded пользовательский realm, например %2Fcustomer
- ‹grant_type› — способ интеграции, всегда используется значение urn:roox:params:oauth:grant-type:m2m
- ‹service› — используемый сервис. В этом сценарии всегда равен sign_document_batch.
- ‹access_token› — текущий токен доступа (access token) пользователя
- otpCode — введенный одноразовый пароль (OTP)
- _eventId — имя перехода к следующему состоянию сценария, в данном запросе всегда равно `validate`
- execution — значение равно значению поля ‹execution› полученному из ответа на предыдущий запрос к API

В зависимости от правильности введенного одноразового пароля ответ сервера 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

Формат успешного ответа

В ответе содержится токен под операцию, 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
}
```


Поля ответа

- ‹execution_value› — значение параметра ‹execution› для следующего запроса к UIDM
- step — обозначение текущего шага сценария, в данном случае отображается форма ввода одноразового пароля
- claims.sign — подпись
- claims.sign_req_id — дублируется идентификатор пакета документов на случай, если приложению удобно взять его из этого ответа

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

В настоящее время событие еще не зафиксировано, так как предполагается, что веб-приложение отобразит экран-подтверждение действия.

Токен можно использовать ровно один раз для подтверждения операции.

Подтверждение операции

Формат запроса

```
POST /sso/api/policyEvaluation/isAllowed
Host: <sso_host>
Accept: application/json
Content-Type: application/json
Authorization: Bearer {токен под операцию, полученный на предыдущем шаге}
```

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

В заголовке Authorization передается полученный на предыдущем шаге токен под операцию

В теле передается точно такой же объект с документами и метаданными, который передавался на самом первом шаге.

```
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 веб-приложение не может обработать ответ, следует записать тело и заголовки ответа в лог и обратиться в службу технической поддержки 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 первом шаге: Старт сценария

Общие для всех шагов сообщения об ошибках

Код ошибки	Описание причины возникновения
invalid_credentials	Пользователь ввел неправильные данные учетной записи и не был аутентифицирован.
expired_password	Время действия пароля, введенного пользователем, истекло. Пользователь не был аутентифицирован.
user_blocked	Учетная запись пользователя заблокирована.
ip_blocked	IP адресс, с которого пользователь обращается в WebSSO, заблокирован.
invalid_captcha	Введенная пользователем CAPTCHA неверная.
need_captcha	Для продолжения сценария пользователь должен ввести CAPTCHA.

error	Пользователь не был аутентифицирован в результате неожиданного ответа от внешней системы.
login-by-otp-disabled	Пользователь попытался аутентифицироваться через сценарий с аутентификацией по OTP, но это невозможно, поскольку данный функционал отключен конфигурационным ключем.
error_sending_otp	Невозможно отправить OTP пользователю.
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.
invalid_otp	Введенный OTP неверный.
validate-otp-fail	Введенный OTP неверный.
otp_expired	Время действия введенного OTP истекло.
otp-expired	Время действия введенного OTP истекло.
otp-error	Сценарий ввода OTP завершился неуспешно.
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.
msisdn-is-not-b2b	Введенный msisdn не принадлежит реалму b2b.
too_many_attempts	Превышено число допустимых попыток изменения пароля.
user-is-not-allowed	Данному пользователю запрещено продолжать данный сценарий.
error_password_change	Сценарий изменения пароля не был успешно завершен. Пароль не был изменен.
no_email_found	Пользователь не найден или у пользователя не существует email.
msisdn-not-exists	Пользователя с заданным msisdn не существует.
principal-not-exists	Пользователь не существует.

no-principal-found	Пользователь не существует.
user-not-found	Пользователь не существует.
login-exists	Невозможно зарегестрировать пользователя с заданным login, поскольку пользователь с заданным login уже существует.
login_already_exists	Невозможно изменить login пользователя на заданный, поскольку пользователь с заданным login уже существует.
email-exists	Пользователь с заданным email уже существует.
user-exists	Невозможно зарегестрировать пользователя с заданными msisdn или email или login, поскольку пользователь с такими параметрами уже существует.
email_verification_failed	Невозможно подтвердить email.
reset_required	Пароль или не был задан либо был сброшен в результате каких-то действий, необходимо создать новый пароль. Отличается от expired_password, поскольку в том случае пароль существует и был провалидирован, но требуется смена. В данном случае пароль не провалидирован, поскольку не задан.

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 первом шаге: Старт сценария

Требования к безопасности

Требования к веб-приложению

1. Убедиться, что прямой доступ к загруженным документам без авторизации невозможен.

2. Принять меры для безопасного хранения `client_secret`, используемого для вызовов API. Секрет не должен возвращаться в ответе пользователю и не должен передаваться в любые третьи приложения.
3. Не протоколировать немаскированные токены доступа, токен под операцию и `client_secret`.
4. Не протоколировать введенный одноразовый пароль. Допустимо протоколирование или запись в СУБД одноразового пароля, который уже был использован для подписания документов.
5. Обеспечить надежное хранение исходных документов. Документы нужны для перерасчета подписи в случае судебных споров. Документы небольшого размера (примерно до 1.5 кб хранятся в СУБД UIDM).
6. Принять меры для безопасного обмена данными между пользователем и веб-приложением. Обеспечить шифрование канала связи (TLS1.2), защиту от XSS, кроссайтовых запросов.
7. Рекомендуется воспользоваться классификатором OWASP для получения более полного списка защит.

Требования к инфраструктуре

Доступ к адресу `https://{uidm_host}/sso/oauth2/access_token` с параметром `service=sign_document_batch` закрыт для доступа из Интернет и открыт явным образом только для WebAPI — то есть серверных компонентов, которые выполняют операции, требующие подписания.

Требования к процессам обслуживания

1. Заводить разные системные учетные записи веб-приложений для приложений разного назначения.
2. Использовать стойкие секреты, сгенерированные СГЧ.
3. Периодически не менее одного раза в год и/или при компрометации ротировать секреты учетных записей веб-приложений.

Рассчитанный хеш подписи

Хеш подписи (представляет собой строку вида `g6kp//pNhWe9A7WVbvJFdnS1tKvhr9/yFyIkD1gpM9cGYJBHKjyVTVkksStegnedDG9lqDlh7hJ7LnIfs0/g+w==`) **НЕ** является секретным.

По нему невозможно восстановить или найти документ или узнать его содержимое. Разрешено использование хеша в необходимых случаях, например отобразить клиенту, сохранить рядом с документами и прочее.