СТАНДАРТ БАНКА РОССИИ

ОТКРЫТЫЕ ПРОГРАММНЫЕ ИНТЕРФЕЙСЫ

Профили Открытых программных интерфейсов для расширенного режима безопасности.

Технический стандарт

Дата введения: 2026-10-01
Версия: 1.0.0

Москва
2025

ОГЛАВЛЕНИЕ

1. История изменений

Версия	Дата	Автор	Комментарий

2. Предисловие

Настоящий стандарт разработан Ассоциацией развития финансовых технологий (Ассоциацией ФинТех) при участии Центрального банка Российской Федерации (Банка России).

ПРИНЯТ И ВВЕДЕН в действие приказом Банка России от 19 декабря 2025 года № ОД-2888 «О введении в действие стандарта Банка России СТО БР «Открытые программные интерфейсы. Профили Открытых программных интерфейсов для расширенного режима безопасности. Технический стандарт».

3. Введение

Настоящий стандарт содержит принципы и рекомендации по реализации протокола взаимодействия OpenID Connect (OIDC) при осуществлении взаимодействия через Открытые программные интерфейсы с использованием расширенного профиля безопасности API, обеспечивающего высокий уровень доверия к идентификации и аутентификации при передаче финансовой информации.

3.1. Область применения

Настоящий стандарт рекомендован к использованию организациями при обмене финансовыми сообщениями в среде Открытых программных интерфейсов.

Настоящий стандарт предназначен для:

- участников взаимодействия, осуществляющих обмен информацией о финансовых продуктах, счетах и других финансовых инструментах Пользователя, а также связанной с ними информацией;
- разработчиков информационного и программного обеспечения.

Положения настоящего стандарта носят рекомендательный характер и применяются совместно со следующими документами:

- Методические рекомендации МР.26.2.002-2024 ТК 26 «Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколах OpenID Connect» (далее - МР OIDC);

- Стандарт Банка России СТО БР ФАПИ.СЕК-1.6-2024 «Безопасность финансовых (банковских) операций. Прикладные программные интерфейсы обеспечения безопасности финансовых сервисов на основе протокола OpenID» (далее - ФАПИ.СЕК);
- Стандарт Банка России СТО БР БФБО-1.8-2024 «Обеспечение безопасности финансовых сервисов при проведении дистанционной идентификации и аутентификации. Состав мер защиты информации» (далее - БФБО 1.8);
- Стандарт Банка России СТО БР «Открытые программные интерфейсы. Общие положения»;
- Стандарт Банка России СТО БР «Открытые программные интерфейсы. Глоссарий»;
- Спецификация OpenAPI (Открытый стандарт OPENAPI Iniative. Подробнее на https://spec.openapis.org/);
- Другими документами комплекса стандартов Открытых API, размещенными на официальном сайте Банка России в информационно-телекоммуникационной сети «Интернет».

3.2. Термины и определения

В настоящем стандарте применяются термины и определения в соответствии с ФАПИ.СЕК, БФБО 1.8, СТО БР «Открытые программные интерфейсы. Глоссарий», а также следующие термины и определения:

- **API-шлюз** — программно-аппаратный или программный компонент, выступающий в роли единой точки входа для всех клиентских запросов к распределённым сервисам (микросервисам, внешним API и другим ресурсам).
- **Аутентификация Пользователя с умеренным уровнем доверия (УДА 2) —** аутентификация, при которой достигается умеренная уверенность в результате. Протокол обеспечивает многофакторную аутентификацию, но не является криптографическим и не обеспечивает взаимную аутентификацию. Все предъявленные аутентификаторы соответствуют цифровой идентичности Пользователя, подтверждён факт их обладания.
- **Аутентификация Пользователя с высоким уровнем доверия (УДА 3)** — аутентификация, при которой достигается значительная уверенность в результате. Протокол обеспечивает многофакторную аутентификацию с использованием криптографических средств, предусматривает взаимную аутентификацию и подтверждает факт обладания каждым предъявленным аутентификатором, привязанным к цифровой идентичности Пользователя.

4. Требования и ограничения

4.1. Использование токенов доступа

Для доступа к ресурсам среды Открытых программных интерфейсов вызов к каждой конечной точке API должен производиться с токеном доступа в авторотационном заголовке HTTP. Требуемая область действия токена доступа (scope) и тип предоставления доступа (grant_type) должны быть определены в каждой спецификации API (для каждого HTTP метода должна быть определена securityScheme в соответствии с OpenAPI Specification раздел Security Scheme Object).

4.1.1. Токен доступа

Настоящий стандарт накладывает следующие ограничения и требования, связанные с токеном доступа:

- Токен доступа должен быть реализован в виде ссылки согласно разделу 6.7.3 MP OIDC.
- Сервер авторизации должен связать токен доступа с информацией, определенной в разделе 6.7.3.1 MP OIDC.
- Сервер авторизации должен связать токен доступа с типом предоставления доступа (grant_type), для которого был сформирован запрос.
- Токен доступа должен соответствовать параметрам схемы безопасности (securityScheme), определенной для каждого метода API. Описание указания securityScheme представлено в OpenAPI Specification (раздел Security Scheme Object).

4.2. Использование криптографических средств

Для реализации механизмов аутентификации протокола OpenID Connect настоящий стандарт определяет необходимость формирования и проверки JWS и HMAC. При этом необходимо использовать сертифицированные библиотеки и криптографические средства, обеспечивающие требования, определенные в главе 9 MP OIDC.

4.3. Метаданные сервера авторизации

Сервер авторизации должен безопасным образом (с обеспечением контроля целостности и аутентификации источника) доставить клиентам свои метаданные, описывающие его адрес и параметры в соответствии с требованиями разделов 5.4.4.1 и 5.4.4.2 ФАПИ.СЕК.

4.4. Метаданные клиента

Клиент перед первым обращением к серверу авторизации безопасным образом (с обеспечением контроля целостности и аутентификации источника) предоставляет ему свои метаданные, описывающие его параметры в соответствии с требованиями разделов 5.4.4.3 ФАПИ.СЕК. Настоящий документ не регламентирует способы и протокол предоставления метаданных клиента серверу авторизации.

4.5. Методы аутентификации клиента

Настоящий стандарт требует использования метода аутентификации private_key_jwt (аутентификация на основе цифровой подписи). При этом в соответствии с разделом 7.2.1 пункт 7 ФАПИ.СЕК участники могут применять метод аутентификации tls_client_auth (аутентификация с использованием MTLS и PKI для связывания сертификата с клиентом) как дополнительный, но не альтернативный метод. Сервер авторизации должен заявлять о поддерживаемых методах аутентификации в своих метаданных.

4.6. Применение PKCE

Настоящий стандарт при реализации сценария гибридного потока с кодом авторизации не требует обязательности использования PKCE согласно раздела 7.2.1 ФАПИ.СЕК. Данное положение обеспечено следующими компенсирующими мерами:

- Запрос аутентификации должен включать подписанный объект запроса (параметр request).
- Положительный ответ на запрос аутентификации должен включать ID токен.
- При обмене кода авторизации на токен доступа клиент должен быть аутентифицирован методом private_key_jwt или tls_client_auth.

4.7. Требования к протоколу TLS

4.7.1. Безопасность взаимодействий

В среде Открытых программных интерфейсов все взаимодействия между клиентом и сервером авторизации, клиентом и сервером ресурсов, а также между сервером авторизации и сервером ресурсов должны защищаться с использованием TLS (HTTPS).

Приложения, соответствующие настоящему стандарту, должны выполнять следующие требования к использованию протокола TLS (в соответствии с разделом 10 OIDC):

- Реализация протокола TLS должна выполняться в соответствии с положениями Р 1323565.1.020-2020 или Р 1323565.1.030-2020. Требуется использование сертифицированных федеральным органом исполнительной власти в области обеспечения безопасности СКЗИ.
- Должна осуществляться проверка сертификата TLS сервера.
- Криптографические ключи, в том числе долговременные ключи, используемые в протоколе TLS, и ключи протокола OpenID Connect должны быть различными.
- Сертификат TLS сервера, используемый клиентом и сервером авторизации для установления соединения с агентом пользователя, должен содержать отличительное имя субъекта (subject distinguished name, DN) либо альтернативное имя субъекта (SAN).
- агент пользователя при перенаправлении на сервер должен использовать предсказуемый способ обработки значений отличительных имён при сравнении отличительного имени субъекта из сертификата TLS сервера с отличительным именем, указанным в перенаправлении. Например, правило distinguishedNameMatch из RFC 4517
- Сервер авторизации, сервер ресурсов не должны быть доступны без использования TLS. В случае обращения клиента без использования TLS, сервер авторизации, сервер ресурсов должны отказать в соединении.
- Должны использоваться только следующие криптонаборы:
  - TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_L (Р 1323565.1.030)
  - TLS_GOSTR341112_256_WITH_MAGMA_MGM_L (Р 1323565.1.030)
  - TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_S (Р 1323565.1.030)
  - TLS_GOSTR341112_256_WITH_MAGMA_MGM_S (Р 1323565.1.030)
  - TLS_GOSTR341112_256_WITH_KUZNYECCHIK_CTR_OMA (Р 1323565.1.020)

- TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC (P 1323565.1.020)

4.8. Кодирование байтовых строк

Настоящий стандарт требует, чтобы все байтовые строки, передаваемые в качестве параметров запросов и ответов в протоколе OIDC (например, значения id_token, nonce, state), должны быть закодированы в формате Base64url, если не указано иное.

4.8.1. Требования к кодированию

Процесс кодирования должен выполняться следующим образом:

- байтовая строка сначала преобразуется в формат Base64
- далее выполняется замена символов: + заменяется на -, а / — на _
- удаляются символы заполнения =, если они имеются в конце строки.

Пример:
Исходная байтовая строка: exampleData
Закодированная строка в формате Base64url: ZXhhbXBsZURhdGE

5. Протокол OpenID Connect с генерацией кода авторизации

5.1. Применяемые режимы

Согласно MP OIDC протокол OpenID Connect предоставляет три различных режима использования при генерации кода авторизации, в зависимости от уровня защиты и требований к передаче данных:

- **Режим 1:** Базовый протокол OpenID Connect с генерацией кода авторизации. В данном режиме используется стандартная процедура обмена кодом авторизации между клиентом и сервером авторизации для получения токенов доступа и ID токена. При этом сервер авторизации должен поддерживать response_type = "code" и response_mode = "query" или "fragment".
- **Режим 2:** Протокол OpenID Connect с генерацией кода авторизации и передачей ответа на запрос аутентификации, подписанного сервером авторизации в формате JWT (режим JARM (JWT Secured Authorization Response Mode). Подпись ответа позволяет обеспечить дополнительную целостность и подтверждение подлинности данных, отправленных сервером авторизации. При этом сервер авторизации должен поддерживать response_type = "code" и response_mode = "jwt".
- **Режим 3:** Протокол OpenID Connect с генерацией кода авторизации и передачей ответа на запрос аутентификации в формате ID токена. (Сценарий гибридного потока с кодом авторизации) В этом режиме ответ на запрос аутентификации включает ID токен, подписанный сервером авторизации, который подтверждает успешную аутентификацию Пользователя. При этом сервер авторизации должен поддерживать response_type = "code id_token" и response_mode = "fragment".

Расширенный профиль безопасности API требует обязательности применения гибридного потока с кодом авторизации для протокола OpenID Connect (Режим 3). При этом в соответствии с разделом 7.2.1 пункт 7 ФАПИ.СЕК участники могут применять Режим 2

(JARM). Сервер авторизации должен заявлять о поддерживаемых параметрах response_mode в своих метаданных.

5.2. Метод аутентификации private_key_jwt

Метод аутентификации private_key_jwt используется для того, чтобы клиент OAuth 2.0 аутентифицировался перед сервером авторизации, подписывая JWT (JSON Web Token) с помощью своего приватного ключа. Сервер авторизации затем проверяет этот JWT с использованием публичного ключа клиента, который он хранит или извлекает из URL JWKS клиента. Этот метод обеспечивает высокий уровень безопасности за счет асимметричной криптографии.

5.2.1. Шаги метода аутентификации private_key_jwt

5.2.1.1. Создание полезной нагрузки (payload) JWT. Клиент создает JWT с обязательными полями:

- **iss** (Issuer): идентификатор клиента (client_id).
- **sub** (Subject): содержит идентификатор клиента, как и в поле iss.
- **aud** (Audience): URL сервера авторизации, к которому направлен запрос (конечная точка токена при запросе токена доступа).
- **iat** (Issued At): время выпуска токена в формате UNIX-времени.
- **exp** (Expiration): время истечения токена.
- **jti** (JWT ID): уникальный идентификатор JWT (например, UUID4 или другой идентификатор с количеством символов 36-64). Используется для предотвращения повторных атак.

5.2.1.2 Подписание JWT. Клиент подписывает JWT с помощью своего приватного ключа для подписи авторизационных запросов. Применяемый ключ должен быть связан с публичным сертификатом, размещенным на JWKS и доступным серверу авторизации по идентификатору клиента (client_id) и идентификатору ключа (kid). Цифровая подпись должна быть вычислено по алгоритму ГОСТ Р 34.10-2012 согласно раздела 9.2.2 МР OIDC, за исключением случаев, когда контекст взаимодействия требует или допускает другие алгоритмы.

5.2.1.3 Создание заголовка (header) JWT. Клиент создает заголовок JWT, содержащий обязательные параметры вычисления цифровой подписи:

- **alg** идентификатор криптографического алгоритма цифровой подписи.
- **kid** Идентификатор ключа, который используется для защиты JWS.

Заголовок может содержать другие параметры, указанные в разделе 8.2.3 МП OIDC, если это требует контекст взаимодействия.

Пример подписанного JWT:

```
JOSE Header:
{
  "kid":"S1a01AAV",
```

"alg":"GOST341012"
}
JSON структура параметров JWS Payload:
{
"iss": "f917546e7a194df9bd02342632cd944f",
"sub": "f917546e7a194df9bd02342632cd944f",
"aud": "https://sb-as.openbankingrussia.ru/sandbox/as/aft/connect/token",
"exp": 1658763062,
"iat": 1658762462,
"jti": "90729a20-5eee-4035-9113-5c731e9e2c63"
}
Значение подписи JWS Signature:

"IuboWQdp6iMKaZMWYPQAhqvSY_h346YOLu7vciLPM6cBn5d0xGEZI89ptVz7wu
33IHxfJs6_ya13Q19cX72mPw"

5.2.1.4 Отправка запроса на сервер авторизации. Клиент отправляет запрос с параметрами:

- **client_assertion**: Подписанный JWT.
- **client_assertion_type**: Указывает тип аутентификации: urn:ietf:params:oauth:client-assertion-type:jwt-bearer.

5.2.1.5 Проверка client_assertion сервером авторизации. Когда сервер авторизации получает запрос с параметром client_assertion, он должен подтвердить подлинность JWT и аутентификацию клиента с помощью следующих проверок:

- **Проверка подписи JWT**: убедиться, что JWT подписан приватным ключом клиента. Сервер авторизации извлекает публичный сертификат клиента из метаданных через JWKS URL по идентификатору клиента (client_id) и идентификатору ключа (kid), полученного из заголовка JWT. Сервер авторизации проверяет актуальность сертификата клиента, в том числе на соответствие политикам области применения (сертификат содержит необходимые OID). Если сертификат не найден или некорректен, запрос отклоняется. Если ключ корректный, сервер авторизации проверяет, что подпись JWT совпадает с его содержимым. Если подпись неверна, запрос отклоняется.
- **Проверка параметра iss (Issuer)** - подтвердить, что токен выпущен корректным клиентом. Значение iss должно совпадать с client_id, зарегистрированным на сервере авторизации и прошедшим MTLS аутентификацию. Если идентификаторы не совпадают, или предъявленный сертификат не принадлежит данному клиенту, запрос отклоняется.
- **Проверка параметра sub (Subject)** - подтвердить, что субъектом токена является клиент. Поле sub должно совпадать с идентификатором клиента (client_id). Если они не совпадают, запрос отклоняется.
- **Проверка параметра aud (Audience)** - проверить, что токен предназначен для текущего сервера авторизации. Значение aud должно совпадать с URL

сервера авторизации (например, с конечной точкой получения токенов). Если значение некорректно, запрос отклоняется.

- **Проверка параметра iat (Issued At).** Значение параметра iat указывается в формате UNIX time (UTC). При обработке токена проверяющая сторона должна отклонять JWT, для которого значение iat находится необоснованно далеко от текущего времени. Конкретные пределы допустимого расхождения (максимальный возраст токена и допустимый рассинхрон часов) устанавливаются в соответствующем профиле/локальной политике безопасности и должны обеспечивать предотвращение атак повторной передачи.
- **Проверка параметра exp (Expiration)** - проверить, что срок действия токена не истек. Поле exp должно содержать время истечения токена в формате UNIX. Сервер авторизации проверяет, что текущее время меньше, чем значение exp. Если срок действия истек, запрос отклоняется.
- **Проверка параметра jti (JWT ID)** - предотвратить повторное использование одного и того же токена (Replay Attack). Поле jti должно содержать уникальный идентификатор токена. Сервер авторизации проверяет, что этот идентификатор не был использован ранее для запросов от данного клиента.

5.2.1.6 Обработка ошибок. Если любая из этих проверок не проходит, сервер авторизации возвращает ошибку (например, invalid_request или invalid_client), указывая на конкретную проблему.

Пример ответа с ошибкой:

```
{
  "error": "invalid_client",
  "error_description": "The provided client_assertion is invalid."
}
```

5.2.1.7 Выдача токена доступа. Если аутентификация успешна, сервер авторизации выдает клиенту токен доступа и, при необходимости, refresh token.

5.3. Требования к сценариям предоставления доступа

5.3.1. Предоставление доступа по учетным данным OIDC клиента (client credentials)

Тип доступа client_credentials используется для идентификации клиента в контексте, где отсутствует Пользователь и не требуется его согласие. Данный тип доступа поддерживается при аутентификации клиента на сервере авторизации поставщика услуг с использованием механизма аутентификации private_key_jwt (в соответствии с разделом 7.3 MP OIDC). При этом клиент, при запросе к конечной точке токена (POST /token), использует тип разрешения на доступ client_credentials в сочетании с assertions и jwt_bearer.

5.3.1.1. Параметры запроса

Запрос к конечной точке токена содержит метод POST, соответствующие заголовки и тело запроса.

Параметр	Описание	Обязательность
grant_type	Тип доступа. В данном случае указывается значение client_credentials.	Обязательный
scope	Область действия. Определяет права доступа, которые запрашивает клиент.	Обязательный
client_assertion_type	Тип утверждения клиента (urn:ietf:params:oauth:client-assertion-type:jwt-bearer).	Обязательный
client_assertion	Значение client_assertion, созданное и подписанное на предыдущем этапе.	Обязательный

5.3.1.2. Пример запроса

```
POST /sandbox/as/aft/connect/token HTTP/1.1
Host: sb-as.openbankingrussia.ru
Content-Type: application/x-www-form-urlencoded
Content-Length: 819
---
grant_type=client_credentials
&scope=accounts
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ...
```

5.3.1.3. Пример ответа

```
{
  "access_token": "eyJhbGciOiJQUzI1NiIsImtpZCI6IjdGMzZFMDAwMUFBRTFGOE...",
  "refresh_token": "dGhpcyBpcyBhIHJlZnJlc2ggdG9rZW4...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "scope": "accounts"
}
```

5.3.1.4. Пример декодированного (с помощью конечной точки интроспекции) токена доступа

```
{
  "nbf": 1660571646,
  "exp": 1660575246,
  "iss": "https://sb-as.openbankingrussia.ru/sandbox/as/aft",
  "aud": "https://sb-as.openbankingrussia.ru/sandbox/as/aft/resources",
  "client_id": "f917546e7a194df9bd02342632cd944f",
  "scope": "accounts",
  "jti": "4a07d081dc9b248c18f67d9acf9346bb841c1c9fb3e8faa69beb113eda6c3071",
```

```
  "client_jwks_uri": "https://sb-
jwks.openbankingrussia.ru/sandbox/jwks/f917546e7a194df9bd02342632cd944f",
  "cnf": {
    "x5t#St256": "C7qKLMKk_bJmQNCgT8MB50VV0d4HJBFgKANg25-Nj60"
  }
}
```


5.3.2. Предоставление доступа с использованием токена обновления (refresh_token)

Тип доступа refresh_token используется для получения нового токена доступа на сервере авторизации, когда текущий токен доступа становится недействительным или истекает срок его действия.

5.3.2.1. Связь offline_access и токена доступа

Когда в запросе аутентификации запрашивается scope=soffline_access, сервер авторизации выдает refresh token, что обеспечивает клиенту возможность получать новые токены доступа с помощью refresh token, не требуя от Пользователя заново проходить процесс аутентификации. offline_access не меняет сам токен доступа, но добавляет refresh token, который нужен для долгосрочного доступа к ресурсам, когда Пользователь не активен. Без offline_access клиент получает только токен доступа, срок действия которого ограничен. По истечении срока действия токена клиент запрашивает новый токен доступа через новую аутентификацию Пользователя.

5.3.2.2. Параметры запроса

Параметр	Описание	Обязательность	Пример значения
grant_type	Тип доступа. В данном случае указывается значение refresh_token.	Обязательный	refresh_token
refresh_token	Токен обновления, полученный в потоке с кодом авторизации и с использованием области доступа offline_access.	Обязательный	dGhpcyBpcyBhIHJlZnJlc2ggdG9rZW4...
client_assertion_type	Тип client_assertion, определяющий, что клиент использует JWT для аутентификации. Значение:	Обязательный	urn:ietf:params:oauth:client-assertion-type:jwt-bearer

Параметр	Описание	Обязательность	Пример значения
	urn:ietf:params:oauth:client-assertion-type:jwt-bearer.		
client_assertion	JWT, используемый для аутентификации клиента на сервере авторизации.	Обязательный	eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ...

5.3.2.3. Пример запроса

```
POST /sandbox/as/aft/connect/token HTTP/1.1
Host: sb-as.openbankingrussia.ru
Content-Type: application/x-www-form-urlencoded
---
grant_type=refresh_token
&refresh_token=dGhpcyBpcyBhIHJlZnJlc2ggdG9rZW4...
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ...
```

5.3.2.4. Пример ответа

```
{
  "access_token": "eyJhbGciOiJQUzI1NiIsImtpZCI6IjdGMzZFMDAwMUFBRTFGOE...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "scope": "accounts offline_access"
}
```

5.3.2.5. Пример декодированного (с помощью конечной точки интроспекции) нового токена доступа

```
{
  "nbf": 1660571646,
  "exp": 1660575246,
  "iss": "https://sb-as.openbankingrussia.ru/sandbox/as/aft",
  "aud": "https://sb-as.openbankingrussia.ru/sandbox/as/aft/resources",
  "client_id": "f917546e7a194df9bd02342632cd944f",
  "scope": "accounts offline_access",
  "sub": "1e3a7d4a-d213-416d-b4d3-ac8000f9d1d0",
  "jti": "6e8f4a47dc9b248c18f67d9acf9346bb841c1c9fb3e8faa69beb113eda6c3071",
  "client_jwks_uri": "https://sb-
jwks.openbankingrussia.ru/sandbox/jwks/f917546e7a194df9bd02342632cd944f",
  "cnf": {
```

```
   "x5t#St256": "C7qKLMKk_bJmQNCgT8MB50VV0d4HJBFgKANg25-Nj60"
  }
}
```


5.3.3. Особенности управления токенами обновления

- При получении нового refresh_token старый должен быть аннулирован.
- Токены обновления действуют дольше, чем токены доступа и могут быть ограничены по количеству обновлений.
- Сервер авторизации должен поддерживать автоматическую ротацию refresh_token.

5.3.4. Предоставление доступа с кодом авторизации (authorization_code)

Гибридный сценарий протокола OIDC с кодом авторизации (authorization_code) используется в соответствии с разделом 5.4.3 ФАПИ.СЕК и ограничениями, определенными в Главе 7 ФАПИ.СЕК. Запрос аутентификации должен выполняться с параметром типа запрашиваемого ответа (response_type) равного значению “code id_token” и использовать только параметры, включенные в подписанный объект запроса (параметр request). При этом клиент при запросе к конечной точке токена (POST /token) использует тип разрешения на доступ (grant_type) authorization_code в сочетании с assertions и jwt_bearer.

5.3.4.1. Запрос аутентификации

Параметры запроса

Параметр	Описание	Обязательность	Пример
scope	Область доступа. Определяет перечень свойств защищаемых данных Пользователя, к которым запрашивается доступ. Требования к применению: • должен содержать значение прикладной области доступа (например, accounts). • должен содержать значение openid, указывающее на то, что клиент запрашивает аутентификацию Пользователя с помощью OpenID	Обязательный	openid accounts offline_access

Параметр	Описание	Обязательность	Пример
	Connect и должен быть возвращен ID-токен. • для получения refresh_token должен содержать значение offline_access.		
response_type	Тип ответа и сценарий протокола авторизации; в данном сценарии используется значение "code id_token".	Обязательный	code id_token
response_mode	Значение, которое информирует сервер авторизации об используемом механизме, который возвращает параметры конечной точки авторизации; может принимать значение "fragment".	Опциональный	fragment
client_id	Идентификатор клиента, полученный при регистрации на сервере авторизации.	Обязательный	4abd59d5970247969965a4f317a8f817
redirect_uri	URI переадресации, на который будет отправлен ответ.	Обязательный	https://localhost.ru/cb
state	Строковое значение, используемое для синхронизации состояния между запросом и обратным вызовом; используется для защиты от атак межсайтовых запросов (CSRF); генерируется как случайная строка длиной не менее 20 байт.	Обязательный	98d6691382344e7fb03c853739d0a988
nonce	Случайное строковое значение, используемое для связывания запроса	Обязательный	642c0152a40a46bbb82bfda4e0799990

Параметр	Описание	Обязательность	Пример
	аутентификации с ID токеном и для защиты от атак повторного воспроизведения; генерируется как случайная строка длиной не менее 20 байт.		
request	Объект запроса; позволяет передавать параметры запроса аутентификации с цифровой подписью или кодом аутентификации клиента в форме JWT; обязательный для расширенного профиля безопасности.	Обязательный	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJyZXNwb2...
code_challenge	Запрос подтверждения кода по технологии PKCE	(Режима 2) - обязательный (Режима 3) - опциональный	eHprXzdhQ0tDd1NBOWlVT2FSSDA2S0tyNJhfTmm65o
code_challenge_method	метод вычисления code_challenge на основе code_verifier (параметры PKCE определены в разделе 6.2.6 MP OIDC).	Обязательный, если указан code_challenge	st256
login_hint	Подсказка серверу авторизации об конечном Пользователе для входа в систему. Например, может содержать ИНН, КПП юридического лица, от имени которого ожидается авторизация Пользователя.	Опциональный	{"taxId":"7728240000","taxType":"991230001"}

5.3.4.2. Верхнеуровневый список запрашиваемых параметров claims

Объект запроса (request) должен содержать параметр claims (подробнее OpenID Connect Core 1.0 раздел 5.5. Requesting Claims using the "claims" Request Parameter), который включает следующие объектные элементы:

1. **userinfo**. Запрашивает, чтобы указанные индивидуальные утверждения (Claims) были возвращены с конечной точки UserInfo. Если Userinfo присутствует, запрашиваются указанные утверждения, которые добавляются к любым утверждениям, запрашиваемым с использованием значений scope. Если параметр отсутствует, то запрашиваются только те утверждения, которые запрашиваются с использованием значений scope на UserInfo.

   Содержит следующие элементы:

   - openbanking_intent_id - идентификатор намерения (в контексте применения к согласию пользования в качестве значения идентификатора намерения указывается идентификатор ресурса согласия consentId), в привязке к которому запрашивается авторизация (направляется текущий запрос аутентификации).

   userinfo может включать другие элементы, требуемые прикладными стандартами. Это может быть отражено в спецификации сервера авторизации.

2. **id_token**. Запрашивает, чтобы указанные индивидуальные утверждения (Claims) были возвращены *в* ID токен. Если параметр id_token присутствует, запрашиваются указанные утверждения, которые добавляются к стандартным утверждениям ID токен. Если параметр отсутствует, запрашиваются только стандартные утверждения ID токен.

   Содержит следующие элементы:

   - openbanking_intent_id - идентификатор намерения.
   - acr_values (подробнее OpenID Connect Core 1.0 раздел 5.5.1.1. Requesting the "acr" Claim) - строковые значения, представляющие собой условные идентификаторы (из области имен), определяющие запрашиваемый метод аутентификации Пользователя. Порядок идентификаторов имеет значение: приоритет отдается значению, указанному первым. Значением может быть как единичный идентификатор, так и набор идентификаторов, расположенных в порядке приоритета применения контекста аутентификации. Перечень применяемых идентификаторов в рамках данной спецификации:
     - urn:rubanking:ca — идентификатор, определяющий контекст аутентификации, указывающий на применение аутентификации Пользователя с умеренным уровнем доверия (УДА 2);
     - urn:rubanking:sca — идентификатор, определяющий контекст аутентификации, указывающий на применение аутентификации Пользователя с высоким уровнем доверия (УДА 3).

id_token может включать другие элементы и идентификаторы, требуемые прикладными стандартами. Это может быть отражено в спецификации сервера авторизации.

5.3.4.3. Идентификатор ресурса согласия в запросе аутентификации

Идентификатор ресурса согласия (далее consent_id) служит уникальным идентификатором согласия Пользователя. Платформы, использующие Открытые программные интерфейсы, Поставщики услуг и Сторонние поставщики услуг могут использовать consent_id для получения и проверки статуса согласия Пользователя, гарантируя, что согласие получено и соблюдается надлежащим образом.

Минимизация данных и безопасность

Стандарты Открытых программных интерфейсов подчеркивают принципы минимизации данных и безопасности, обеспечивая сбор, обработку и передачу только необходимой информации во время операций. Параметр openbanking_intent_id поддерживает эти принципы, обеспечивая механизм связи запросов на авторизацию с конкретным намерением Пользователя дать долгосрочное согласие или провести операцию, минимизируя риск несанкционированного доступа к персональным данным.

Связь намерения с процессом аутентификации

Открытые программные интерфейсы требует строгой аутентификации Пользователя для определенных действий, таких как инициирование платежей или доступ к конфиденциальным финансовым данным. Протокол OIDC позволяет включать значение consent_id в параметр объекта запроса openbanking_intent_id. Передача объекта запроса через параметр request служит средством связи процессов аутентификации и согласия. Это гарантирует синхронизацию процесса получения согласия с потоком аутентификации, обеспечивая, что конкретное согласие Пользователя (которое уже определено) будет получено в контексте безопасного сеанса аутентификации.

Предотвращение несанкционированного доступа и контроль согласия

Связь намерения с процессом аутентификации помогает предотвратить несанкционированный доступ к данным Пользователя и гарантирует, что согласие будет получено контролируемым и проверяемым способом. Consent_id облегчает проверку, предоставляя точку отсчета для отслеживания истории действий и решений, связанных с согласием, что позволяет более точно и безопасно управлять процессом получения и использования согласия.

Пример объекта запроса аутентификации с со значение consent_id в качестве параметра openbanking_intent_id

Пример демонстрирует включение `openbanking_intent_id` в сведущие `claims`:"

- userinfo - связать информацию о Пользователе с идентификатором согласия (идентификатор согласия будет включен в ответ с конечной точки UserInfo).
- id_token - связать токен идентификации с идентификатором согласия (идентификатор согласия будет включен в ID токен).

```
{
  "kid":"S1a01AAV",
  "alg":"GOST341012"
}
{
 "response_type": "code id_token",
 "state": "98d6691382344e7fb03c853739d0a988",
 "scope": "openid accounts offline_access",
 "nonce": "642c0152a40a46bbb82bfda4e0799990",
 "exp": 1618760589,
 "max_age": 86400,
 "claims": {
 "userinfo": {
 "openbanking_intent_id": {
 "value": "0c9df54a-b926-4853-acc2-e318c9bd7c33",
 "essential": true
 }
 },
 "id_token": {
 "openbanking_intent_id": {
 "value": "0c9df54a-b926-4853-acc2-e318c9bd7c33",
 "essential": true
 },
 "acr": {
 "values": [
 "urn:rubanking:sca",
 "urn:rubanking:ca"
 ],
 "essential": true
 }
 },
 "participant": {
 "tax_id": {
 "value": "6148127514"
 },
 "tax_type": {
 "value": "583501001"
 }
 }
 },
 "aud": "https://sb-as.openbankingrussia.ru/sandbox/as/aft",
 "iss": "4abd59d5970247969965a4f317a8f817",
 "client_id": "4abd59d5970247969965a4f317a8f817",
 "redirect_uri": "https://localhost.ru/cb"
}
```

Применение параметра login_hint

Параметр login_hint используется для передачи серверу авторизации подсказки о том, какого Пользователя необходимо аутентифицировать. Он может содержать идентификатор Пользователя или другую информацию, которая помогает серверу авторизации определить, какого Пользователя аутентифицировать без необходимости запрашивать дополнительные данные от клиента. Значения параметра, представленные в виде JSON-объекта, необходимо передавать как URL-кодированную строку.

Пример преобразования JSON-объекта {"taxId":"7728240000","taxType":"991230001"} в URL-кодированную строку:

%7B%22taxId%22%3A%227728240000%22%2C%22taxType%22%3A%22991230001%22%7D

При проектировании сервера авторизации необходимо учитывать, что данный параметр является подсказкой, а для целевого указания необходимо включать информацию о Пользователе в заявленные свойства объекта запроса, используя запрашиваемые claims добавлять элемент participant.

Пример запроса:

https://sb-as.openbankingrussia.ru/sandbox/as/aft/connect/authorize?
client_id=4abd59d5970247969965a4f317a8f817
&response_type=code%20id_token
&state=98d6691382344e7fb03c853739d0a988
&redirect_uri=https://localhost.ru/cb
&nonce=642c0152a40a46bbb82bfda4e0799990
&scope=openid%20accounts%20offline_access
&request=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJyZXNwb2...

&login_hint=%7B%22taxId%22%3A%227728240000%22%2C%22taxType%22%3A%22991230001%22%7D

5.3.4.4. Ответ от сервера авторизации

Параметры ответа

Параметр	Описание	Обязательность	Пример
code	Код авторизации, полученный после успешной аутентификации.	Обязательный	10e5ded165a96d423aaa42a678cb9c09460963245
id_token	ID токен, содержащий информацию о	Обязательный	eyJhbGciOiJSUzI1NiIsImtpZCI6IkQwM0I4NkE4MEJBNjBCQjM0...

Параметр	Описание	Обязательность	Пример
	Пользователе и времени аутентификации.		
state	Состояние, переданное в запросе аутентификации, возвращается для проверки.	Обязательный	98d6691382344e7fb03c853739d0a988
session_state	Состояние сеанса на сервере авторизации.	Опциональный	G6rAVS56SipMpkdgSH-ZM3nJggTXo9MQ74sK8VE3n3o29c1bf0fa

Параметр session_state

session_state — опциональный параметр, возвращаемый в ответе на запрос авторизации и используемый для определения состояния сеанса Пользователя между клиентом и сервером авторизации. Он позволяет клиенту отслеживать изменения состояния сеанса, включая его завершение, разрыв соединения или повторную аутентификацию.

Механизм работы:

- После завершения процесса аутентификации сервер авторизации возвращает session_state вместе с кодом авторизации (`code`) на `redirect_uri` клиента.
- Клиент может использовать значение session_state для отслеживания состояния сеанса и предотвращения проблем безопасности, таких как фиксирование сеансов.
- session_state` поддерживает Cross-Origin OpFrame в браузере для управления состоянием и отслеживания изменений с помощью *check_session_iframe*.

Формат и структура

Значение session_state представляет собой зашифрованный идентификатор состояния, который обычно включает хеш от:

- Идентификатора клиента (client_id`)
- Идентификатора сессии
- redirect_uri

Этот идентификатор регулярно проверяется клиентом для определения, не изменился ли статус сеанса.

Пример успешного ответа

```
https://localhost.ru/cb#code=10e5ded165a96d423aaa42a678cb9c09460963245
&id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6IkQwM0I4NkE4MEJBNjBCQjM0...
```

&scope=openid%20accounts%20offline_access
&state=98d6691382344e7fb03c853739d0a988
&session_state=G6rAVS56SipMpkdgSH-ZM3nJggTXo9MQ74sK8VE3n3o29c1bf0fa

Пример ошибки

https://localhost.ru/cb#error=invalid_request
&error_description=Invalid+request+parameters
&state=98d6691382344e7fb03c853739d0a988
&session_state=G6rAVS56SipMpkdgSH-ZM3nJggTXo9MQ74sK8VE3n3o29c1bf0fa

5.3.4.5. Параметры токена идентификации

Параметр	Описание	Обязательность	Пример значения
iss	Эмитент токена. Значение URI сервера авторизации.	Обязательный	https://sb0.openbankingrussia.ru/sandbox0/as/aft
sub	Уникальный идентификатор субъекта, выданный сервером авторизации конечному Пользователю; регистрозависимая строка длиной не более 255 символов ASCII.	Обязательный	1e3a7d4a-d213-416d-b4d3-ac8000f9d1d0
aud	Идентификатор субъекта, представленного сервером авторизации, для которого выдается ID токен (должно быть равно значению "client_id" клиента).	Обязательный	a8cadb2f65944ce2b3b92ba21336ad53
exp	Период актуальности токена; определяется ПУ при условии, что выбранное значение не влияет на качество сервисов, предоставляемых через Открытые программные интерфейсы.	Обязательный	1607716325

Параметр	Описание	Обязательность	Пример значения
iat	Метка времени выпуска токена.	Обязательный	1607716025
auth_time	Метка времени события аутентификации Пользователя; обязательный при включении параметра "max_age".	Зависит от контекста	1607716014
nonce	Случайное значение, используемое в качестве условного идентификатора сессии обмена сообщениями.	Обязательный	642c0152a40a46bbb82bfda4e0799990
acr	Идентификатор типа контекста аутентификации.	Опциональный	urn:rubanking:sca
s_hash	Хэш-значение параметра; строковое значение, которое вычисляется сервером авторизации и проверяется клиентом как Base64url кодирование левой половины значения хэш-функции ГОСТ Р 34.11-2012 октетов ASCII представления значения параметра state, полученного в составе запроса аутентификации.	Обязательный	nVDApI-dUj2qei-oU9QeUw
at_hash	Хэш-значение токена доступа; вычисляется сервером авторизации и проверяется клиентом как Base64url кодирование левой половины значения хэш-функции ГОСТ Р	Опциональный	V1e8eU_GTK0-Z1_WF7n_JA

Параметр	Описание	Обязательность	Пример значения
	34.11-2012 октетов ASCII представления значения access token.		
c_hash	Хэш-значение кода авторизации; строковое значение, которое вычисляется сервером авторизации и проверяется клиентом как Base64url кодирование левой половины значения хэш-функции ГОСТ Р 34.11-2012 октетов ASCII представления значения параметра code.	Обязательный	OATPHzlrPpzO3PpMPLNknQ
nbf	Время, до которого ID токен не должен приниматься к обработке.	Опциональный	1607716025
openbanking_intent_id	Идентификатор ресурса, к которому запрашивается авторизация.	Опциональный	1726c4f8-af35-41ef-bd84-569fb4647e1a
amr	Массив строк в формате JSON, чувствительных к регистру.	Опциональный	["password"]
alg	Идентификатор криптографического алгоритма цифровой подписи.	Обязательный	GOST341012

5.3.4.6. Пример ID токена в ответе на запрос аутентификации

```
{
  "alg": "GOST341012",
  "type": "JWT"
}
```

```
{
 "nbf": 1607716025,
 "exp": 1607716325,
 "iss": "https://sb0.openbankingrussia.ru/sandbox0/as/aft",
 "aud": "a8cadb2f65944ce2b3b92ba21336ad53",
 "iat": 1607716025,
 "nonce": "642c0152a40a46bbb82bfda4e0799990",
 "c_hash": "OATPHzlrPpzO3PpMPLNknQ",
 "s_hash": "nVDApI-dUj2qei-oU9QeUw",
 "sub": "1e3a7d4a-d213-416d-b4d3-ac8000f9d1d0",
 "auth_time": 1607716014,
 "openbanking_intent_id": "1726c4f8-af35-41ef-bd84-569fb4647e1a",
 "amr": ["password"]
}
```


5.3.4.7. Проверка ID токена

Для обеспечения безопасности и подлинности ID токена необходимо выполнять следующие проверки в соответствии с разделом 6.7.2 MP OIDC по следующей последовательности шагов (ссылки указаны на разделы MP OIDC):

1. **Расшифровка ID токена:** Если ID токен зашифрован, клиент должен расшифровать его с использованием ключей и алгоритмов, указанных сервером авторизации, которые он применял для шифрования ID токена в формате JWE (см. раздел 8.2).
2. **Проверка подписи ID токена JWS**: убедиться, что ID токена подписан приватным ключом сервера авторизации. Клиент проверяет цифровую подпись структуры JWS ID токена с использованием алгоритма, указанного в параметре alg, и ключа, предоставленного сервером авторизации клиент извлекает публичный сертификат клиента из метаданных через JWKS URL сервера авторизации по идентификатору ключа (kid), полученного из заголовка JWT. клиент проверяет актуальность сертификата сервера авторизации, в том числе на соответствие политикам области применения (сертификат содержит необходимые OID). Если сертификат не найден или некорректен, ID токен считается не действительным. Если сертификат корректный, клиент проверяет, что подпись ID токена совпадает с его содержимым.
3. **Проверка идентификатора эмитента (iss):** Идентификатор эмитента (сервер авторизации), полученный клиентом от сервера авторизации (см. раздел 5.4), должен в точности совпадать со значением параметра iss в ID токене и соответствовать ранее сохраненному идентификатору сервера авторизации (см. раздел 6.2.3).
4. **Проверка идентификатора аудитории (aud):** Клиент убеждается, что значение параметра aud содержит значение client_id клиента. Если aud не содержит значение client_id, ID токен должен быть отклонен.
5. **Проверка параметра azp (Авторизованный получатель):** Если параметр aud включает несколько значений, клиент проверяет наличие параметра azp и убедиться, что его значение совпадает с client_id.
6. **Проверка хэш-значений:** Клиент проверяет значения параметров c_hash, s_hash, и at_hash (если они присутствуют), сравнив каждое из них с Base64url-кодированием левой половины значения хэш-функции ГОСТ октетов ASCII представления

сохраненного ранее значения параметра code, state или access_token в соответствии с формулами (1), (2) или (3).
    - (1) c_hash = BASE64URL(LMB16(HASH256(ASCII(code))));
    - (2) s_hash = BASE64URL(LMB16(HASH256(ASCII(state))));
    - (3) at_hash = BASE64URL(LMB16(HASH256(ASCII(access_token))));
7. **Проверка подписи JWS:** Клиент проверяет цифровую подпись структуры JWS (см. раздел 8.1) ID токена с использованием алгоритма, указанного в параметре alg, и ключа, предоставленного сервером авторизации.
8. **Проверка допустимости алгоритма:** Значение параметра alg должно быть одним из допустимых значений, указанных в разделе 9.1.
9. **Проверка времени истечения (exp):** Текущее время на момент проверки должно быть меньше времени, указанного в параметре exp, чтобы токен считался действительным.
10. **Проверка параметра nonce:** Значение параметра nonce в структуре ID токена должно совпадать со значением параметра nonce из запроса аутентификации, отправленного клиентом на сервер авторизации (см. раздел 6.2.3).
11. **Проверка времени аутентификации (auth_time):** Если параметр auth_time был запрошен (либо явным указанием, либо через параметр max_age в запросе аутентификации), клиент убеждается, что значение auth_time соответствует диапазону, определенному в max_age. В случае, если с момента последней аутентификации прошло больше времени, чем допустимо, клиент инициирует повторную аутентификацию.

Если клиент получил ID токен, который не прошел проверку, то клиент прекращает текущий процесс аутентификации и не использовать полученные данные и может инициировать повторную аутентификацию.

5.3.4.8. Получение токена доступа в обмен на код авторизации

После успешной аутентификации Пользователя на сервере авторизации поставщика услуг, проверки ответа аутентификации и получения кода авторизации клиент использует запрос к конечной точке токена (POST /token) с разрешением на доступ (grant_type) authorization_code в сочетании с assertions и jwt_bearer для получения токена доступа и ID токен в обмен на код авторизации. Данный тип доступа поддерживается при аутентификации клиента на сервере авторизации поставщика услуг с использованием механизма аутентификации private_key_jwt (в соответствии с разделом 7.3 MP OIDC).

Параметры запроса

Параметр	Описание	Обязательность	Пример значения
grant_type	Тип доступа. В данном случае указывается значение authorization_code.	Обязательный	authorization_code

Параметр	Описание	Обязательность	Пример значения
code	Код доступа, полученный в запросе аутентификации offline_access.	Обязательный	dGhpcyBpcyBhIHJlZnJlc2ggdG9rZW4...
client_assertion_type	Тип client_assertion, определяющий, что клиент использует JWT для аутентификации. Значение: urn:ietf:params:oauth:client-assertion-type:jwt-bearer.	Обязательный	urn:ietf:params:oauth:client-assertion-type:jwt-bearer
client_assertion	JWT, используемый для аутентификации клиента на сервере авторизации.	Обязательный	eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ...
code_verifier	Сохраненное при запросе аутентификации (раздел 6.2.3 ФАПИ.СЕК) подтверждение кода по технологии PKCE.	Опциональный	yJpc3MiOiJGdkldk61qWE1aas

Пример запроса с обязательными параметрами grant_type=authorization_code, code, redirect_uri и client_id.

```
POST /sandbox/as/aft/connect/token HTTP/1.1
Host: sb-as.test.openbankingrussia.ru
Content-Type: application/x-www-form-urlencoded
---
grant_type=authorization_code
&code=40ac728b26bf06a078538a65c1f18f89a9e8554899cb45a2ff2c918dc7742fe7
&redirect_uri=https://localhost.ru/cb
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer

&client_assertion=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI0YmEzYjk4YTRjNm...
```

Параметры ответа

Ссылка на описание параметров успешного ответа представлены в разделе "API - post /token".

Пример успешного ответа


```
{
  "id_token": "eyJhbGciOiJSUzI1NiIsI...",
  "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjU...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "13e29519e3a09bca92ca9c3f41a886ca"}
```


5.3.5. Информация о Пользователе

Клиент может получить информацию о Пользователе, либо извлекая ее из параметров ID токена, либо с помощью запроса к конечной точке UserInfo сервера авторизации.

Клиент может отправить запрос конечной точке UserInfo, используя токен доступа, полученный при аутентификации на сервере авторизации и содержащий область требуемую для данного метода область доступа (scope). Значения параметров в ответ возвращаются в виде JSON объекта, который содержит набор пар имя и значение параметра. Набор доступных клиенту параметров информации о Пользователе определяется в спецификации сервера авторизации.

Конечная точка UserInfo и id_token оба предоставляют информацию о Пользователе, но использование UserInfo имеет ряд преимуществ:

- **Обновляемая информация**: Конечная точка UserInfo позволяет получать актуальную информацию о Пользователе в режиме реального времени. В отличие от id_token, который содержит статическую информацию, полученную во время аутентификации, запрос к UserInfo возвращает данные, которые могут быть более актуальными на момент запроса.
- **Уменьшение размера токена**: id_token может стать большим, если в него включить много информации о Пользователе. Использование UserInfo позволяет минимизировать размер id_token, возвращая минимальный набор полей, необходимых для аутентификации, а дополнительную информацию можно получить через запрос к UserInfo.
- **Гибкость и безопасность**: Доступ к конечной точке UserInfo осуществляется с использованием токена доступа, что позволяет гибко управлять правами доступа к данным о Пользователе. Это также обеспечивает дополнительный уровень безопасности, поскольку информация о Пользователе не передается напрямую в id_token, а запрашивается отдельно через защищенный канал.
- **Снижение нагрузки на сервер авторизации**: Использование UserInfo может разгрузить сервер авторизации, так как основная информация о Пользователе передается через отдельный запрос, а не через JWT-токен в процессе авторизации. Это позволяет разделить функциональные обязанности между разными компонентами системы.

Таким образом, использование конечной точки UserInfo предпочтительно в сценариях, где требуется актуальная и безопасная передача информации о Пользователе, а также при необходимости минимизировать нагрузку на сервер авторизации.

5.3.6. Токен доступа, связанный с MTLS сертификатом клиента

5.3.6.1. Привязка токена доступа к MTLS сертификату клиента

Сервер авторизации должен поддерживать использования токенов доступа, связанных с клиентским сертификатом TLS (MTLS-bound access tokens), как предусмотрено в пункте 6.3.1.2 ФАПИ.СЕК. Связывание токена доступа с MTLS сертификатом клиента осуществляется путем добавления в токен хэш-кода сертификата клиента. Этот механизм гарантирует, что только клиент, владеющий соответствующим сертификатом, может использовать токен доступа для выполнения запросов к защищенным ресурсам.

Привязка к сертификату происходит следующим образом:

1. **Получение хэш-кода сертификата клиента:** сервер авторизации вычисляет хэш-код (например, с использованием алгоритма St256) на основе публичного ключа сертификата клиента, который использовался для установки MTLS-соединения.
2. **Добавление хэша в токен:** Хэш-код добавляется в поле cnf (confirmation) внутри токена доступа. Это поле выглядит следующим образом: Параметр x5t#St256 представляет собой значение хэш-кода сертификата клиента, закодированное в формате Base64 URL-safe.
3. **Выдача токена:** После добавления хэш-кода сертификата сервер авторизации выдает клиенту токен доступа, связанный с конкретным MTLS сертификатом.

5.3.6.2. Проверка токена доступа, связанного с MTLS сертификатом

Для проверки токена, связанного с MTLS сертификатом, сервер ресурсов должен убедиться, что клиент, предъявляющий токен, использует тот же сертификат, который был связан с этим токеном на этапе его выдачи. Проверка включает следующие шаги:

1. **Получение сертификата из MTLS-соединения:** Когда клиент делает запрос к защищенному ресурсу, сервер ресурсов извлекает публичный сертификат, который использовался клиентом при установлении MTLS-соединения.
2. **Извлечение хэш-кода из токена доступа:** сервер ресурсов извлекает значение хэш-кода сертификата из поля cnf токена доступа:
3. **Вычисление хэш-кода публичного сертификата клиента:** сервер ресурсов вычисляет хэш-код публичного ключа сертификата, полученного из MTLS-соединения, с использованием того же алгоритма (например, St256).
4. **Сравнение хэш-кодов:** сервер ресурсов сравнивает хэш-код из токена доступа с хэш-кодом, вычисленным из сертификата MTLS-соединения.
   Если значения совпадают, то клиент считается аутентифицированным, и запрос может быть обработан.
   Если значения не совпадают, сервер отклоняет запрос с кодом ошибки HTTP 401 и возвращает сообщение об ошибке с указанием причины: invalid_token.

Пример включения хэш-кода сертификата клиента в декодированный токен доступа:

```
{
  ... ,
  "cnf": {
```

```
    "x5t#St256": "значение хэша TLS сертификата клиента"
  }
}
```

5.3.7. Проверка запроса токена

При получении запроса на выдачу токена доступа сервер авторизации выполняет ряд проверок, в зависимости от запрашиваемого типа доступа (значение параметра grant_type).

5.3.7.1. Общие требования для всех типов доступа:

- Запрос токена отправлен через защищенное соединение (соединение установлено с помощью MTLS).
- Запрос токена должен быть аутентифицирован с помощью метода private_key_jwt.
- Запрашиваемый scope соответствует разрешенным областям для данного клиента.

5.3.7.2. authorization_code

Цель: Предоставить токен на основе авторизационного кода, полученного Пользователем после успешного прохождения процесса аутентификации.

- Проверить, что параметр grant_type имеет значение authorization_code.
- Проверить наличие и корректность параметра code, выданного сервером авторизации ранее.
- Проверить, что code не истек и не был использован ранее (для одноразовых кодов).
- Сравнить redirect_uri, переданный в запросе на получение токена, с тем, что использовался при получении кода.
- В случае использования PKCE проверить параметры code_verifier и code_challenge (при наличии).

5.3.7.3. client_credentials

Цель: Предоставить токен доступа клиенту без участия Пользователя, для доступа к защищенным ресурсам.

- Проверить, что параметр grant_type имеет значение client_credentials.
- Убедиться, что клиенту разрешено использовать данный grant_type.
- Проверить параметр scope, если он присутствует, и убедиться, что клиент имеет доступ к указанным областям (scopes).

5.3.7.4. refresh_token

Цель: Обновить токен доступа, используя ранее выданный refresh token.

- Проверить, что параметр grant_type имеет значение refresh_token.
- Проверить наличие и корректность параметра refresh_token.
- Проверить, что refresh_token действителен и не истек.
- Проверить, что клиент идентичен тому, которому был выдан refresh_token.

- Проверить, что клиент имеет право на обновление токенов с помощью refresh_token.
- Проверить параметр scope, чтобы убедиться, что клиент запрашивает допустимые области (scopes).

5.3.8. Проверка ответа токена

5.3.8.1. Указание области доступа (`scope`) в ответе на запрос токена

Сервер авторизации обязан возвращать параметр scope в ответе на запрос токена только если выданный токен доступа имеет меньше прав (более узкую область доступа), чем клиент запрашивал. Если предоставленная область доступа полностью соответствует области доступа запроса, scope может быть не возвращаться. В случае запроса токена с типом доступа refresh_token область доступа запроса определяется как область действия токена обновления.

5.3.8.2. Проверка ответа токена в зависимости от запрашиваемого типа доступа
5.3.8.3. client_credentials

Сервер авторизации возвращает токен доступа в ответ на успешную аутентификацию клиента. Проверка ответа включает:

1. **Проверка валидности токена:** Токен должен быть подписан сервером авторизации и действителен.
2. **Проверка области действия (scope):** Выданные области действия (scopes) должны соответствовать запрошенным или быть ограниченными в соответствии с политикой сервера авторизации.
3. **Проверка времени действия:** Поля iat и exp проверяются для подтверждения того, что токен еще действителен.

5.3.8.4. refresh_token

При использовании refresh token для обновления токена доступа, проверка ответа включает:

1. **Проверка refresh token:** Токен обновления должен быть действителен, не истек и принадлежит клиенту.
2. **Проверка области действия (scope):** Новые области действия должны соответствовать требованиям и политике сервера авторизации.
3. **Проверка времени действия нового токена:** Поля iat и exp проверяются для подтверждения валидности нового токена.

5.3.8.5. authorization_code

Проверка ответа токена при использовании кода авторизации, включая PKCE (Proof Key for Code Exchange), включает:

1. **Проверка кода авторизации:** Код авторизации проверяется на предмет валидности, клиента и Пользователя, для которых он был выдан, а также срок его действия.

2. **PKCE:** Сервер авторизации проверяет, что предоставленный code_verifier соответствует code_challenge, переданному на этапе запроса кода авторизации.
3. **Проверка области действия (scope):** Сервер авторизации проверяет, что области действия токена доступа соответствуют запрашиваемым клиентом.
4. **Проверка времени действия:** Поля iat и exp проверяются для подтверждения действительности токена доступа.

5.4. Доступ к защищенному ресурсу

Доступ клиента к защищенному ресурсу обеспечивает сервер ресурсов. Выполняя запрос, клиент передает серверу ресурсов полученный от сервера авторизации токен доступа в поле заголовка запроса Authorization с использованием схемы аутентификации Bearer.

Доступ клиента к серверу ресурсов должен осуществляться по защищенному каналу с использованием двухсторонней аутентификации по протоколу TLS.

5.4.1. Проверка токена доступа

Сервер ресурсов предоставляет доступ после проверки токена доступа. Проверка может быть проведена на API шлюзе, обеспечивающем доступ к серверу ресурсов, с использованием информации, полученной от сервера авторизации.

5.4.1.1. Проверка основных свойств токена доступа

Раздел 6.7.5.2 MP OIDC требует выполнения следующих условий:

- **Идентификатор сервера авторизации** (iss), связанный с токеном доступа, должен точно совпадать с идентификатором issuer в метаданных сервера авторизации.
- **Идентификатор сервера ресурсов** (aud) должен соответствовать адресу ресурса, обслуживаемого данным сервером ресурсов.
- **Запрашиваемый ресурс** должен присутствовать в области действия токена доступа (значение параметра scope).
- **Текущее время** должно быть больше времени, указанного в параметре времени выпуска токена доступа (значение параметра iat).
- **Текущее время** должно быть меньше времени, указанного в сроке действия токена доступа (значение параметра exp).
- **Отпечаток MTLS сертификата клиента**, с использованием которого он был аутентифицирован на сервере авторизации, должен соответствовать указанному идентификатору клиента. client_id.

5.4.2. Проверка на соответствие схеме безопасности

Токен доступа должен проверяться сервером ресурсов на соответствие схеме безопасности securityScheme, определённой в OpenAPI Specification для вызываемого метода application/json:

- **Тип доступа (grant_type)**, связанный с токеном доступа, должен соответствовать OAuth2 потоку, указанному в параметре flows в securityScheme.

- **Область доступа (scope)**, связанная с токеном доступа, должена быть определена парраметре scopes в securityScheme.

5.4.3. Проверка согласия

При доступе к ресурсам Пользователя в рамках среды Открытых программных интерфейсов сервер ресурсов должен проверить согласия Пользователя и соответствующих разрешений, если они указаны в согласии. Требования к проверкам согласия клиента определяются в прикладных стандартах Открытых программных интерфейсов.

6. Отзыв токена доступа

Когда Пользователь отзывает своё согласие на доступ к данным, все связанные с данным согласием токены доступа (access tokens) и токены обновления (refresh tokens) должны быть немедленно отозваны на стороне сервера авторизации с целью предотвращения несанкционированного использования данных.

Если необходимо принудительно аннулировать действующий токен, сервер авторизации может реализовать, а клиент использовать конечную точку отзыва токенов. Сервер авторизации должен заявлять о поддерживаемых методах в своих метаданных.

6.1. Параметры запроса отзыва токена

Параметр	Описание	Обязательность	Пример значения
token	Токен, который необходимо отозвать. Может быть access_token или refresh_token.	Обязательный	dGhpcyBpcyBhIHJlZnJlc2ggdG9rZW4...
token_type_hint	Тип токена. Значения: access_token или refresh_token.	Рекомендуемый	refresh_token
client_assertion_type	Тип client_assertion. Значение: urn:ietf:params:oauth:client-assertion-type:jwt-bearer.	Обязательный	urn:ietf:params:oauth:client-assertion-type:jwt-bearer
client_assertion	JWT для аутентификации клиента. Используется	Обязательный	eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ...

Параметр	Описание	Обязательность	Пример значения
	для проверки подлинности отзыва запроса.		

Пример запроса:

```
POST /sandbox/as/aft/connect/revocation HTTP/1.1
Host: sb-as.openbankingrussia.ru
Content-Type: application/x-www-form-urlencoded
---
token=dGhpcyBpcyBhIHJlZnJlc2ggdG9rZW4...
&token_type_hint=refresh_token
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ...
```

7. Спецификация сервера авторизации

7.1. Конечные точки

Спецификация API определяет следующие конечные точки:

Наименование API	Метод	Обязательность	Описание	Метод авторизации
Authorize	`get /authorize`	Да	Запрос аутентификации	
Token	`post /token`	Да	Запрос токена	
UserInfo	`get /userinfo`	Да	Запрос информации о Пользователе	Client Authorization code flow; scopes: obruprofile: Получение информации о Пользователе

7.1.1. Authorize

7.1.1.1. API - get /authorize

Запрос аутентификации (get /authorize).
Конечная точка, используемая клиентом для получения авторизации от владельца ресурса посредством перенаправления агента Пользователя.

7.1.1.2. Параметры query

Наименование	Обязательность	Тип	Описание	Паттерн
scope	Да	String	Область действия;	/^[\w\W]{1,80}$/

Наименование	Обязательность	Тип	Описание	Паттерн
			определяет свойства защищаемых данных Пользователя, к которым запрошен доступ; в случае использования протокола OpenID Connect параметр должен содержать строку “openid”	
response_type	Да	ResponseType	Тип ответа и сценарий протокола авторизации; в данном сценарии используется следующее значение: – “code id_token”: возвращает код авторизации и ID токен	/^[\w\W]{1,15}$/
redirect_uri	Да	URI; format: uri	URI перенаправления, на который сервер авторизации отправит ответ. Должен точно совпадать с зарегистрированным URI клиента.	
state	(Да)	String	Значение, используемое для синхронизации состояния между запросом и обратным вызовом; используется для защиты от атак	/^[\w\W]{27,512}$/

Наименование	Обязательность	Тип	Описание	Паттерн
			межсайтовых запросов (CSRF)	
client_id	Да	String	Идентификатор клиента, полученный при регистрации на сервере авторизации	
response_mode	Нет	ResponseMode	Значение, которое информирует сервер авторизации об используемом механизме, который возвращает параметры конечной точки авторизации.	/^[\w\W]{1,10}$/
nonce	Да	String	Случайное строковое значение, используемое для привязки сеанса клиента к ID токену и для защиты от атак повторного воспроизведения	/^[\w\W]{27,512}$/
request	Да	RequestParameters	Объект запроса	
login_hint	Нет	String	Подсказка серверу авторизации о Пользователе для входа в систему. (Например может содержать ИНН, КПП юридического лица, от имени которого ожидается	/^[\w\W]{1,2048}$/

Наименование	Обязательность	Тип	Описание	Паттерн
			авторизация пользователя).	

7.1.1.3. Продюсер

Данный метод API предоставляет следующий тип данных (media types): text/html.

Значение типа данных должно быть передано клиентом в параметре заголовка запроса - Accept: text/html и возвращено продюсером в ответе в виде заголовка - Content-Type: text/html.

7.1.1.4. Возвращаемые типы

HTTP код	Тип ответа
303	String
400	Error
500	Error

7.1.2. Token

7.1.2.1. API - post /token

Запрос токена (post /token).
Получение токена доступа дает возможность получить данные конкретного Пользователя (т.е. получить доступ к защищенному ресурсу). Токены доступа являются кратковременными и по мере достижения срока жизни должны заменяться путем обмена токена обновления на новый токен доступа.

7.1.2.2. Consumes

Данный метод API использует следующие типы мультимедиа через заголовок запроса Content-Type: application/x-www-form-urlencoded

7.1.2.3. Параметры заголовка запроса

Наименование	Обязательность	Тип	Описание	Паттерн
X-Request-ID	Нет	UUID; format: uuid	RFC4122 UID, используемый в качестве идентификатора запроса или корреляции. В случае, если Сервер авторизации поддерживает корреляцию запросов, то он может возвращать обратно значение данного идентификатора взаголовке ответа X-Request-ID	

7.1.2.4. Параметры form

Наименование	Обязательность	Тип	Описание	Паттерн
grant_type	Да	GrantType		
client_assertion_type	Да	ClientAssertionType		
client_assertion	Да	String	Утверждение, используемое для аутентификации клиента. Используется формат JWT	/^[\w\W]{32,8192}$/
client_id	Нет	String	Идентификатор сервиса клиента	/^[\w\W]{1,40}$/
code	Нет	String	Код авторизации	/^[\w\W]{27,512}$/
code_verifier	Нет	String	Сохраненное при запросе аутентификации (раздел 6.2.3 ФАПИ.СЕК) подтверждение кода по технологии PKCE	/^[\w\W]{32,128}$/
redirect_uri	Нет	URI; format: uri	URI переадресации, на который будет отправлен ответ; значение этого параметра должно совпадать со значением параметра "redirect_uri" запроса авторизации	/^[\w\W]{1,2048}$/
refresh_token	Нет	String	Токен обновления	/^[\w\W]{32,2048}$/
scope	Нет	String	Область применения	/^[\w\W]{1,40}$/

7.1.2.5. Тип ответа

TokenEndpointSuccessfulResponse

7.1.2.6. Продюсер

Данный метод API предоставляет следующие типы данных (media types): application/json

Значение типа данных должно быть передано клиентом в параметре заголовка запроса Accept: application/json и возвращено продюсером в ответе в виде заголовка Content-Type: application/json.

7.1.2.7. Возвращаемые типы

HTTP код	Тип ответа
200	TokenEndpointSuccessfulResponse
400	Error
429	
500	Error

8. UserInfo

8.1. API - get /userinfo

Запрос информации о Пользователе (get /userInfo).
Конечная точка возвращает подписанную JWS информацию о Пользователе, прошедшем аутентификацию и авторизовавшем предъявленный токен доступа.

8.1.1. Параметры заголовка запроса

Наименование	Обязательность	Тип	Описание	Паттерн
X-Request-ID	Нет	UUID; format: uuid	RFC4122 UID, используемый в качестве идентификатора запроса или корреляции. В случае, если Сервер авторизации поддерживает корреляцию запросов, то он может возвращать обратно значение данного идентификатора взаголовке ответа X-Request-ID	

8.1.2. Тип ответа

String

8.1.3. Продюсер

Данный метод API предоставляет следующие типы данных (media types):application/jwt

Значение типа данных должно быть передано клиентом в параметре заголовка запроса Accept: application/jwt и возвращены продюсером в ответе с указанием в параметре заголовка Content-Type: application/jwt.

8.1.4. Возвращаемые типы

HTTP код	Тип ответа
200	String
400	Error
401	
403	Error
405	Error

9. Модель данных

Спецификация API определяет типы и форматы данных, значения по умолчанию, перечисления и справочную информацию.

9.1. Общие типы данных

9.1.1. Error

Контейнер с детализацией ошибки

Наименование	Обязательность	Тип	Описание	Шаблон/Список
error	Да	string	Код ошибки	
error_description	Нет	string	Некорректный запрос к серверу	

9.1.2. Essential

Раздел для указания Essential Claim

Наименование	Обязательность	Тип	Описание	Шаблон/Список
essential	Да	boolean	Указывает, запрашивается ли Claim как Essential Claim.	

9.1.3. RequestIndividualClaim

Определяет структуру запрашиваемых Claims. Необходимо использовать один из элементов.

Наименование	Обязательность	Тип	Описание	Шаблон/Список
essential	Да	boolean	Указывает, запрашивается ли Claim как Essential Claim.	
value	Да	string	Указывает значение запрашиваемого Claim	
values	Нет	array[String]	Указывает список значений запрашиваемого Claim	

9.1.4. RequestParameters

Наименование	Обязательность	Тип	Описание	Шаблон/Список
exp	Да	integer	Срок действия; время, при наступлении и после наступления которого объект запроса считается недействительным; значением является число в формате JSON, представляющее количество секунд от 1970-01-01T0:0:0Z UTC до соответствующей даты/времени	
max_age	Нет	integer	Максимальный срок аутентификации, определяет допустимое время в секундах, прошедшее с момента последней активной аутентификации Пользователя сервером авторизации. Если истекшее время больше этого значения, сервер авторизации должен пытаться активно повторно аутентифицировать Пользователя. Если в запросе аутентификации присутствует параметр <max_age>, возвращаемый ID токен должен включать значение	

Наименование	Обязательность	Тип	Описание	Шаблон/Список
			параметра <auth_time>	
claims	Да	RequestParameters_claims		
aud	Да	string; format: uri	Идентификатор субъекта, для которого предоставляется объект запроса (содержит URL сервера авторизации)	
iss	Да	string	Идентификатор субъекта, выпустившего объект запроса (соответствует client_id клиента, отправляющего запрос)	/^[\w\W]{32,2048}$/
client_id	Да	string	Идентификатор клиента OAuth 2.0, полученный при регистрации на сервере авторизации	/^[\w\W]{4,8192}$/
redirect_uri	Да	string; format: uri	URI переадресации, на который будет отправлен ответ. Должен быть предварительно зарегистрирован на сервере авторизации	
response_type	Да	string	Тип ответа, указывающий, что клиент ожидает в ответ (code для кода авторизации)	
state	Да	string	Параметр для поддержания состояния между запросом и обратным вызовом	
scope	Да	string	Область действия	

Наименование	Обязательность	Тип	Описание	Шаблон/Список
nonce	Да	string	Случайное значение, используемое в качестве условного идентификатора сессии обмена сообщениями между Сторонним поставщиком и сервером авторизации	
code_challenge	Нет	AnyType		
code_challenge_method	Нет	AnyType		

9.1.5. RequestParameters_claims

Верхнеуровневый список запрашиваемых параметров claims

Наименование	Обязательность	Тип	Описание	Шаблон/Список
id_token	Нет	Map_Option_RequestIndividualClaim		
userinfo	Нет	Map_Option_RequestIndividualClaim		
participant	Нет	Map_Option_RequestIndividualClaim	Информация об участнике (юридическое или физическое лицо, кто запросил аутентификацию). Используемые значения параметра для запрашиваемых свойств: tax_id – Опциональный - ИНН участника. tax_type – Опциональный - КПП участника. Указывается для юридических лиц только при наличии значения параметра tax_id.	

9.1.6. TokenEndpointSuccessfulResponse

Наименование	Обязательность	Тип	Описание	Шаблон/Список
access_token	Да	string	Токен доступа	/^[\w\W]{32,4096}$/
id_token	Нет	string	Токен идентификации	/^[\w\W]{32,4096}$/

Наименование	Обязательность	Тип	Описание	Шаблон/Список
token_type	Да	TokenType	Тип токена доступа	
expires_in	Да	integer; format: int64	Срок жизни токена в секундах	
scope	Нет	string	Область применения	/^[\w\W]{1,40}$/
refresh_token	Нет	string	Токен обновления	/^[\w\W]{32,2048}$/

9.1.7. Value

Наименование	Обязательность	Тип	Описание	Шаблон/Список
value	Да	string	Указывает значение запрашиваемого Claim	

9.1.8. Values

Наименование	Обязательность	Тип	Описание	Шаблон/Список
values	Нет	array[String]	Указывает список значений запрашиваемого Claim	
essential	Нет	boolean	Указывает, запрашивается ли Claim как Essential Claim.	

9.2. Типы данных, представленных в виде кодов

9.2.1. ClientAssertionType

Тип утверждения клиента. Имеет значение urn:ietf:params:oauth:client-assertion-type:jwt-bearer

9.2.2. GrantType

Тип доступа. Сообщает конечной точке токена, что клиент использует тип предоставления кода авторизации

9.2.3. ResponseMode

Значение, которое информирует сервер авторизации об используемом механизме, который возвращает параметры конечной точки авторизации. Расширенный профиль безопасности требует fragment

9.2.4. ResponseType

Тип ответа и сценарий протокола авторизации; в данном сценарии используется следующее значение: – “code id_token” (возвращает код авторизации и ID токен).

9.2.5. TokenType

Тип токена (имеет значение Bearer - токен на предъявителя)