Банк России

СТАНДАРТ БАНКА РОССИИ

СТО БР ФАПИ.ПАОК-1.0-2021

БЕЗОПАСНОСТЬ ФИНАНСОВЫХ (БАНКОВСКИХ) ОПЕРАЦИЙ

ПРИКЛАДНЫЕ ПРОГРАММНЫЕ ИНТЕРФЕЙСЫ. ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ ФИНАНСОВЫХ СЕРВИСОВ ПРИ ИНИЦИАЦИИ OPENID CONNECT КЛИЕНТОМ ПОТОКА АУТЕНТИФИКАЦИИ ПО ОТДЕЛЬНОМУ КАНАЛУ

ТРЕБОВАНИЯ

МОСКВА
2021

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

1. РАЗРАБОТАН Ассоциацией развития финансовых технологий (Ассоциация ФинТех) и открытым акционерным обществом «Информационные технологии и коммуникационные системы» (ОАО «ИнфоТеКС») при участии Банка России.

2. ПРИНЯТ И ВВЕДЕН В ДЕЙСТВИЕ приказом Банка России от 23 июля 2021 года № ОД-1536.

3. ВВЕДЕН ВПЕРВЫЕ.

Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок – в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (https://www.rst.gov.ru/portal/gost).

1. ВВЕДЕНИЕ

Настоящий стандарт разработан на основе спецификаций OpenID Connect Client Initiated Backchannel Authentication Flow – Core и Client Initiated Backchannel Authentication Profile организации OpenID Foundation (OIDF), которая продвигает и поддерживает сообщество и технологии OpenID (OpenID Connect Core (OIDC), OpenID Connect Discovery (OIDD) и другие) и определяет порядок использования модели прикладных программных интерфейсов (application programming interface, API) со структурированными данными для повышения безопасности финансовых технологий в случае инициирования OpenID Connect клиентом потока аутентификации по отдельному каналу.

2. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящий стандарт позволяет проверяющей стороне, имеющей актуальный идентификатор конечного пользователя, использовать протокол OpenID Connect для инициирования потока аутентификации конечного пользователя без перенаправления через браузер и получать токены от сервера авторизации.

Настоящий стандарт рекомендован к использованию при создании и оценке соответствия программных средств, предназначенных для безопасного обмена финансовыми сообщениями в среде открытых банковских интерфейсов.

Настоящий стандарт предназначен для:

- участников получения информации о банковском счете (банки и их клиенты, а также сторонние поставщики);
- участников перевода денежных средств (банки и их клиенты, а также сторонние поставщики);
- разработчиков информационного и программного обеспечения, информационных систем.

Положения настоящего стандарта применяются на добровольной основе, если только в отношении конкретных положений обязательность их применения не установлена нормативными актами Банка России или условиями договоров.

Положения настоящего стандарта применяются совместно со следующими документами:

- стандарт Банка России СТО БР ФАПИ.СЕК-1.6 – 2020 «Безопасность финансовых (банковских) операций. Прикладные программные интерфейсы обеспечения безопасности финансовых сервисов на основе протокола OpenID» (далее – ФАПИ.СЕК);
- документ технического комитета ТК26 «Использование российских криптографических алгоритмов в протоколах OpenID Connect».

3. ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Авторизация (authorization): проверка, подтверждение и предоставление прав логического доступа при осуществлении субъектами доступа логического доступа.
[ГОСТ Р 57580.1–2017, пункт 3.15]

Аутентификация: действия по проверке подлинности субъекта доступа и/или объекта доступа, а также по проверке принадлежности субъекту доступа и/или объекту доступа предъявленного идентификатора доступа и аутентификационной информации.
Примечание. Аутентификация рассматривается применительно к конкретному субъекту доступа и/или конкретному объекту доступа.
[ГОСТ Р 58 833–2020, пункт 3.4]

Аутентификация по отдельному каналу: аутентификация конечного пользователя с использованием отдельного от основного потока OpenID Connect (дополнительного) канала взаимодействия с сервером авторизации.

Владелец ресурса (resource owner): субъект, способный предоставить доступ к защищенному ресурсу. [RFC6749]

Доступ: получение одной стороной информационного взаимодействия возможности использования ресурсов другой стороны.
[ГОСТ Р 58833–2020, пункт 3.17]

Примечания.
1. В качестве ресурсов стороны информационного взаимодействия, которые может использовать другая сторона информационного взаимодействия, рассматриваются информационные ресурсы, вычислительные ресурсы средств вычислительной техники и ресурсы автоматизированных (информационных) систем, а также средства вычислительной техники и автоматизированные (информационные) системы в целом.
2. Доступ к информации – возможность получения информации и ее использования.

Защищенный ресурс (protected resource): ресурс с ограниченным доступом. [RFC6749]

Идентификация (identification): действия по присвоению субъектам и объектам доступа идентификаторов и (или) по сравнению предъявляемого идентификатора с перечнем присвоенных идентификаторов.
[Р 50.1.053–2005, пункт 3.3.9]

Класс контекста аутентификации (Authentication Context Class): набор методов или процедур аутентификации, которые считаются эквивалентными друг другу в определенном контексте. [OpenID. Core]

Клиент (client): приложение, выполняющее запросы защищенных ресурсов от имени владельца ресурса и с его авторизацией.
[RFC6749]

Примечание. Термин «клиент» не подразумевает каких-либо конкретных характеристик реализации (например, выполняется приложение на сервере, настольном компьютере или других устройствах).

Конечная точка (endpoint): адрес ресурса, точка входа интерфейса. [RFC6749]

Примечание. В процессе авторизации технология OAuth 2.0 использует две конечные точки сервера авторизации (ресурсы HTTP): конечную точку авторизации и конечную точку токена – и конечную точку клиента.

Конечная точка авторизации (authorization endpoint): конечная точка, используемая клиентом для получения авторизации от владельца ресурса посредством перенаправления агента пользователя. [RFC6749]

Конечная точка аутентификации по отдельному каналу (Backchannel Authentication Endpoint): конечная точка, используемая клиентом для инициирования аутентификации конечного пользователя по отдельному каналу. [OpenID.CIBA.Core]

Конечная точка токена (token endpoint): конечная точка, используемая клиентом для обмена разрешения на доступ на токен доступа, обычно с аутентификацией клиента. [RFC6749]

Конечная точка уведомления клиента (Client Notification Endpoint): конечная точка, установленная клиентом во время процессов регистрации и обнаружения метаданных (Discovery), которую сервер авторизации вызывает после успешной или неудачной аутентификации конечного пользователя. [OpenID.CIBA.Core]

Конечный пользователь (end user): владелец ресурса, в случае если он является человеком. [RFC6749]

Контекст аутентификации (Authentication Contex): информация, которая может потребоваться проверяющей стороне, прежде чем она примет решение о предоставлении права на ответ об аутентификации. [OpenID. Core]

Конфиденциальный клиент (confidential client): клиент, который может обеспечить конфиденциальность своих учетных данных (например, клиент, реализованный на защищенном сервере с ограниченным доступом к учетным данным клиента) или может выполнить безопасную аутентификацию клиента с использованием других средств. [RFC6749]

Объект запроса (request object): токен JWT, который содержит набор параметров запроса в качестве своих заявленных свойств. [ФАПИ.СЕК]

Односторонняя аутентификация: аутентификация, обеспечивающая только лишь для одного из участников процесса аутентификации (объекта доступа) уверенность в том, что другой участник процесса аутентификации (субъект доступа) является тем, за кого себя выдает предъявленным идентификатором доступа.

[ГОСТ Р 58833 – 2020, пункт 3.36]

Параметр, заявленное свойство (claim): часть информации, заявленной о субъекте. Заявленное свойство представлено в виде пары имя/значение, состоящей из имени заявленного свойства (параметра) и значения заявленного свойства (параметра). [RFC7519]

Прикладной программный интерфейс (application program interface, API): интерфейс между прикладным программным средством и прикладной платформой, через который обеспечивается доступ ко всем необходимым службам (услугам).

[Р 50.1.041 – 2002, пункт 3.1.5]

Проверяющая сторона (Relying Party): клиентское приложение OAuth 2.0, требующее аутентификации конечного пользователя и предъявления заявленных свойств от сервера авторизации о событии аутентификации и конечном пользователе. [OpenID. Core]

Публичный клиент (public client): клиент, который не может обеспечить конфиденциальность своих учетных данных (например, клиент, выполняющийся на устройстве, используемом владельцем ресурса, таком как установленное нативное приложение или приложение на основе веб-браузера) и не может выполнить безопасную аутентификацию клиента с помощью других средств. [RFC6749]

Разрешение на доступ (authorization grant): свидетельство, представляющее авторизацию владельца ресурса (для доступа к его защищенным ресурсам), которое далее используется клиентом для получения токена доступа. [RFC6749]

Сервер авторизации (authorization server): сервер, выдающий клиенту токены доступа после успешной аутентификации владельца ресурса и получения авторизации. [RFC6749]

Сервер ресурсов (resource server): сервер, на котором размещены защищенные ресурсы, способный принимать и отвечать на запросы защищенных ресурсов с использованием токенов доступа. [RFC6749]

Токен доступа (access token): свидетельство, представляющее авторизацию, выданную клиенту сервером авторизации с одобрения владельца ресурса. Токен доступа содержит указание на конкретные области действия, к которым разрешен доступ, длительность доступа и другие параметры. [RFC6749]

Токен идентификации, ID токен (ID Token): JSON веб-токен, который содержит параметры события аутентификации. Может содержать также другие параметры. [RFC7519]

Токен на предъявителя (bearer token): токен доступа с тем свойством, что любая сторона, владеющая токеном (предъявитель), может использовать токен по назначению, не доказывая владение соответствующим криптографическим ключом. [RFC6750]

Токен обновления (refresh token): свидетельство, используемое для получения токенов доступа. Токен обновления выдается клиенту сервером авторизации и используется для получения нового токена доступа, когда текущий токен доступа становится недействительным или истекает его срок действия, или для получения дополнительных токенов доступа с идентичной или более узкой областью действия (токены доступа могут иметь более короткий срок службы и меньше разрешений на доступ, чем разрешено владельцем ресурса). [RFC6749]

Хэш-код (hash-code): строка бит, являющаяся выходным результатом хэш-функции.
[ГОСТ Р 34.10, пункт 3.1.13]

Хэш-функция (collision-resistant hash-function): функция, отображающая строки бит в строке бит фиксированной длины и удовлетворяющая следующим свойствам:

1) по данному значению функции сложно вычислить исходные данные, отображаемые в это значение;
2) для заданных исходных данных сложно вычислить другие исходные данные, отображаемые в то же значение функции;
3) сложно вычислить какую-либо пару исходных данных, отображаемых в одно и то же значение.

[ГОСТ Р 34.10, пункт 3.1.14]

[Электронная цифровая] подпись (signature): строка бит, полученная в результате процесса формирования подписи.
[ГОСТ Р 34.10, пункт 3.1.15]

Base64-кодирование: стандарт кодирования двоичных данных при помощи только 64 символов ASCII. Алфавит кодирования содержит алфавитно-цифровые латинские символы A-Z, a-z и 0 – 9 (62 знака) и два дополнительных символа, зависящих от системы реализации. [RFC4648]

Base64url-кодирование: Base64-кодирование строки октетов с использованием набора символов, допускающих использование результата кодирования в качестве имени файла или URL. [RFC4648]

JSON веб-токен (JWT): строка, представляющая набор параметров в формате объекта JSON, который закодирован в виде структуры JWS или JWE, сопровождая параметры цифровой подписью, кодом аутентификации и/или шифрованием. [RFC7519]

4. ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

API (Application Program Interface) – прикладной программный интерфейс.

ASCII – стандарт США 8-битного кодирования некоторого набора печатных и непечатных символов.

ASCII (STRING) – октеты ASCII представления последовательности ASCII символов STRING.

AD (Authentication Device) – устройство аутентификации, принадлежащее пользователю, на котором он аутентифицируется и авторизует запрос авторизации.

BASE64URL (OCTETS) – Base64url-кодирование строки октетов OCTETS.

CD (Consumption Device) – устройство, предоставляющее доступ конечному пользователю к услугам.

CIBA (Client Initiated Backchannel Authentication Flow) – инициируемый OpenID Connect клиентом поток аутентификации по отдельному каналу.

FAPI (Financial-grade API) – API обеспечения безопасности финансовых сервисов.

HTTP (Hypertext Transfer Protocol) – протокол передачи гипертекстовых сообщений.

HTTPS (Hypertext Transfer Protocol Secure) – протокол защищенной передачи гипертекстовых сообщений.

HMAC (Hash-based Message Authentication Code) – код аутентификации сообщения на основе хэш-функции.

JOSE (JavaScript Object Signing and Encryption) – технология подписи и шифрования объектов JavaScript.

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript (нотация объектов JavaScript).

JWE (JSON Web Encryption) – структура данных в формате JSON, представляющая зашифрованное и защищенное от модификации сообщение.

JWK (JSON Web Key) – структура данных в формате JSON, представляющая криптографический ключ.

JWS (JSON Web Signature) – структура данных в формате JSON, представляющая сообщение с цифровой подписью или кодом аутентификации сообщений.

JWT (JSON Web Token) – токен доступа, основанный на формате JSON.

MAC (Message Authentication Code) – код аутентификации сообщения.

MTLS (Mutual TLS) – процесс, в соответствии с которым при согласовании сеанса TLS выполняется взаимная аутентификация сервера TLS и клиента, а также подтверждение владения соответствующими ключами.

<name> – параметр (claim) некоторого субъекта с именем name.

OAuth – открытый протокол (схема) авторизации.

OIDC (OpenID Connect Core) – семейство протоколов, являющихся расширением протоколов OAuth 2.0, позволяющих расширить их функционал путем более точного описания процесса аутентификации владельца ресурса и возможности клиенту получить информацию о нем.

OIDF (OpenID Foundation) – некоммерческая организация, созданная для управления авторскими правами, товарными знаками, маркетинговыми компаниями и другой деятельностью, связанной с сообществом OpenID.

OpenID – открытый стандарт децентрализованной системы аутентификации.

REST (Representational State Transfer) – передача состояния представления объекта физического мира (либо информации об абстрактном объекте) в форме веб-ресурса на сервере. Архитектурный стиль взаимодействия компонентов распределенного приложения в сети.

RFC (Request for Comments) – предложения для обсуждения; серия нормативных документов, стандартизирующих протоколы Интернет.

TLS (Transport Layer Security) – протокол защиты транспортного уровня.

URI (Uniform Resource Identifier) – унифицированный идентификатор ресурса.

URL (Uniform Resource Locator) – унифицированный адрес ресурса (единый указатель ресурса).

URN (Uniform Resource Name) – унифицированное имя ресурса.

UTF-8 – стандарт кодирования символов, позволяющий компактно хранить и передавать символы Unicode, используя переменное количество байтов (от 1 до 4), и обеспечивающий полную обратную совместимость с кодировкой ASCII.

UTF8 (STRING) – октеты UTF-8 представления последовательности Unicode символов STRING.

5. ОБЩИЕ ПОЛОЖЕНИЯ

5.1. СТРУКТУРА СТАНДАРТА

Настоящий стандарт представляет дополнительные требования и рекомендации для обеспечения безопасного доступа к данным в финансовых сервисах реального времени с использованием модели обмена данными REST/JSON, защищенной технологией OAuth, включая профилирующий ее протокол OpenID Connect, при инициировании клиентом потока аутентификации по отдельному каналу.

Стандарт состоит из двух частей:

1. Реализация OID Connect при аутентификации по отдельному каналу (раздел 6).
2. Профиль безопасности OpenID API с использованием потока аутентификации по отдельному каналу для доступа к сервисам в режиме чтения и записи (раздел 7).

В разделе 5.2 приведены нормативные требования.

Первая часть стандарта основана на документе [OpenID.CIBA.Core], выпущенном OpenID Foundation. В ней представлены особенности реализации OIDC в зависимости от применяемых механизмов аутентификации клиента (подраздел 6.1), определяются дополнительные параметры и значения объектов обнаружения метаданных ([OpenID. Discovery]) и динамической регистрации клиентов ([OpenID. DCR]) (подраздел 6.2), параметры конечных точек аутентификации и уведомления клиента (подразделы 6.3, 6.4), рассмотрен процесс получения результата аутентификации (подраздел 6.5), а также сведения о применяемых кодах ошибок.

Вторая часть – источник [OpenID.CIBA.RW] – содержит дополнительные параметры профиля безопасности API для доступа к сервисам финансовых данных (описание профиля представлено в главе 7 ФАПИ.СЕК).

5.2. НОРМАТИВНЫЕ ТРЕБОВАНИЯ

При реализации криптографических механизмов настоящего стандарта должны использоваться средства криптографической защиты информации (СКЗИ), соответствующие требованиям федерального органа исполнительной власти в области обеспечения безопасности. Класс СКЗИ, используемого для реализации требований настоящего стандарта, определяется по результатам анализа системы, к компонентам которой применяются эти требования, в соответствии с нормативными актами Российской Федерации.

Вовлеченные стороны должны следовать требованиям обеспечения безопасности персональных данных, определенным законодательными и нормативными актами Российской Федерации, в частности:

- Федеральным законом от 27.07.2006 № 152-ФЗ «О персональных данных» [№ 152-ФЗ];
- постановлением Правительства Российской Федерации от 01.11.2012 № 1119 «Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных» [№ 1119];
- приказом Федеральной службы безопасности (ФСБ России) от 10.07.2014 № 378 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных с использованием средств криптографической защиты информации, необходимых для выполнения установленных Правительством Российской Федерации требований к защите персональных данных для каждого из уровней защищенности» [№ 378];
- приказом Федеральной службы по техническому и экспортному контролю (ФСТЭК России) от 18.02.2013 № 21 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных» [№ 21].

Прикладное программное обеспечение, реализующее требования настоящего стандарта, должно отвечать требованиям положений Банка России от 09.06.2012 № 382-П [№ 382-П], от 17.04.2019 № 683-П [№ 683-П], от 20.04.2021 № 757-П [№ 757-П] в части, предъявляемой к анализу уязвимостей прикладного финансового программного обеспечения как на этапе разработки, так и при его использовании в составе конечных целевых систем.

Разработка программного обеспечения, реализующего требования настоящего стандарта, должна осуществляться в соответствии с [ГОСТ Р 56 939]. В том числе должен проводиться регулярный поиск информации, связанной с уязвимостями программ, в общедоступных источниках, включая использование банка данных угроз безопасности информации ФСТЭК России.

Оценка соответствия и сертификация реализаций, отвечающих требованиям настоящего стандарта, производятся в соответствии с законодательством Российской Федерации.

6. РЕАЛИЗАЦИЯ OPENID CONNECT ПРИ АУТЕНТИФИКАЦИИ ПО ОТДЕЛЬНОМУ КАНАЛУ

Реализация протокола OpenID Connect в случае инициации клиентом потока аутентификации конечного пользователя по отдельному каналу использует дополнительную конечную точку аутентификации и асинхронный метод для уведомления о ее результатах посредством следующих дополнительных действий:

1. Клиент выполняет запрос к конечной точке аутентификации по отдельному каналу, чтобы запросить аутентификацию конечного пользователя по отдельному каналу.
2. Сервер авторизации отвечает уникальным идентификатором конечного пользователя в фоновом режиме.
3. Клиент получает ID токен, токен доступа и опционально токен обновления с помощью режимов Poll (пункт 6.1.1), Ping (пункт 6.1.2) или Push (пункт 6.1.3), определяемых при регистрации клиента (пункт 6.2.2).

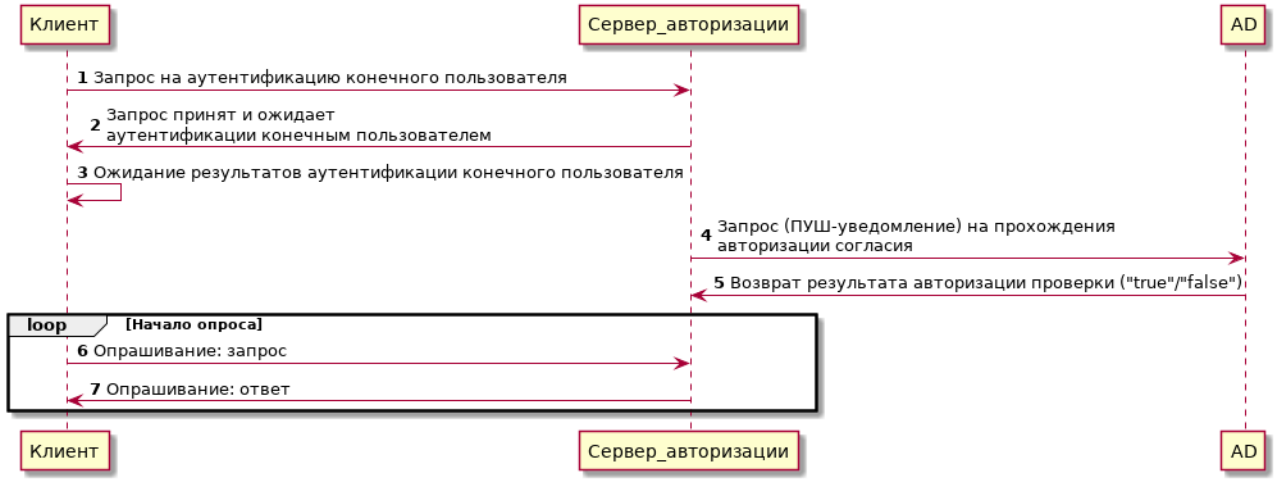
6.1. РЕЖИМЫ POLL, PING И PUSH

В настоящем разделе определяются режимы получения клиентом результата аутентификации.

6.1.1. Режим Poll

Режим Poll – механизм аутентификации по отдельному каналу, при котором клиент опрашивает конечную точку токена, чтобы получить ответ с токенами.

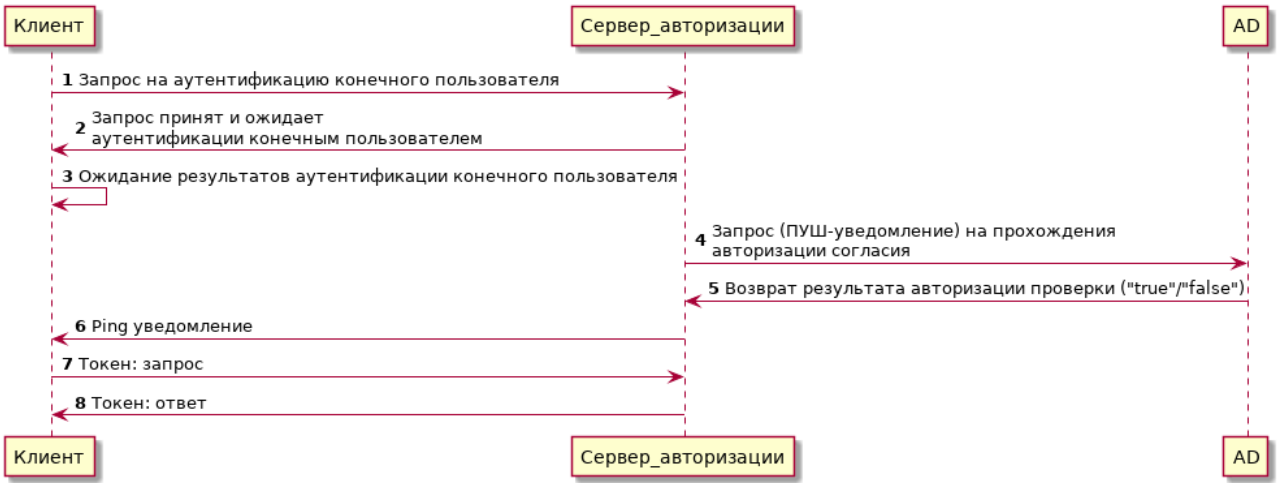
ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДЛЯ РЕЖИМА POLL *Рис. 6.1.1.1*



6.1.2. Режим Ping

Режим Ping – механизм аутентификации по отдельному каналу, при котором сервер авторизации отправляет на URI конечной точки уведомления клиента запрос с уникальным идентификатором, возвращенным из конечной точки аутентификации по отдельному каналу. После получения уведомления клиент делает запрос на конечную точку токена для получения токенов.

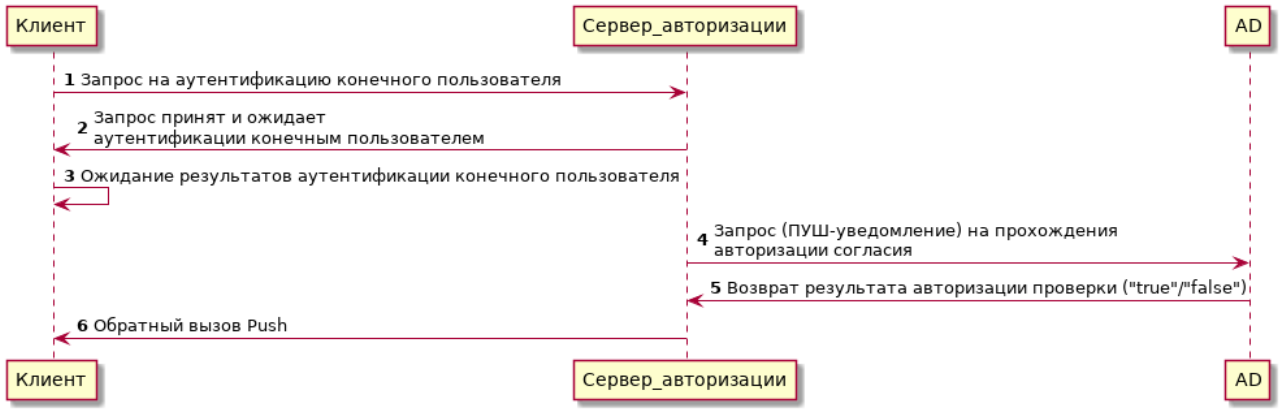
ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДЛЯ РЕЖИМА PING *Рис. 6.1.2.1*



6.1.3. Режим Push

Режим Push – механизм аутентификации по отдельному каналу, при котором сервер авторизации отправляет запрос с токенами на зарегистрированный клиентом URI обратного вызова для режима Push.

ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДЛЯ РЕЖИМА PUSH *Рис. 6.1.3.1*



6.2. РЕГИСТРАЦИЯ И ОБНАРУЖЕНИЕ МЕТАДАННЫХ

В настоящем подразделе определяются параметры и значения объектов процесса обнаружения метаданных ([OpenID. Discovery]) и динамической регистрации клиентов ([OpenID. DCR]) (пункт 5.4.4 ФАПИ.СЕК), относящиеся к определению поддерживаемых режимов аутентификации:

- метаданные сервера авторизации (подраздел 8.3 ФАПИ.СЕК);
- метаданные клиента (подраздел 8.2 ФАПИ.СЕК).

6.2.1. Метаданные сервера авторизации

6.2.1.1. Тип разрешения на доступ

Для обеспечения аутентификации конечного пользователя по отдельному каналу определяется дополнительный тип на доступ «urn: openid: params: grant-type: ciba», который используется в параметре <grant_types_supported> метаданных сервера авторизации, поддерживающего режимы Ping или Poll для обнаружения метаданных, а также объявляется клиентом в запросе авторизации к конечной точке токена для получения токена.

Примечание. Рекомендации по расширению типа доступа, определенные в подразделе 4.5 [RFC6749].

6.2.1.2. Параметры сервера авторизации

Для объявления поддерживаемых методов аутентификации клиента и их применимости, а также алгоритмов подписания JWT для конечной точки аутентификации по отдельному каналу определяются дополнительные параметры метаданных сервера авторизации: <token_endpoint_auth_methods_supported> и <token_endpoint_auth_signing_alg_values_supported>.

6.2.1.3. Дополнительные метаданные сервера авторизации

ДОПОЛНИТЕЛЬНЫЕ МЕТАДАННЫЕ СЕРВЕРА АВТОРИЗАЦИИ *Табл. 1*

Наименование параметра	Кратность	Определение	Примечание
backchannel_token_delivery_modes_supported	1..1	JSON-массив, содержащий значения, указывающие поддерживаемые режимы («poll», «ping» или «push»)	
backchannel_authentication_endpoint	1..1	URL-адрес конечной точки аутентификации по отдельному каналу сервера авторизации	
backchannel_authentication_request_signing_alg_values_supported	0..1	Массив JSON, содержащий список алгоритмов подписания JWS-запросов аутентификации, поддерживаемых сервером авторизации	В случае отсутствия параметра подписанные запросы аутентификации не поддерживаются сервером авторизации
backchannel_user_code_parameter_supported	0..1	Логическое значение, указывающее поддержку сервером авторизации параметра <user_code> (значение «true» означает поддержку)	В случае отсутствия параметра значением по умолчанию является «false»

6.2.2. Метаданные клиента

Клиенты, регистрирующиеся с использованием аутентификации по отдельному каналу, указывают режим доставки токена <backchannel_token_delivery_mode>.

При использовании режима Ping или Poll клиент указывает новый тип доступа «urn: openid: params: grant-type: ciba» в списке поддерживаемых типов авторизации (параметр <grant_types>).

При использовании режима Ping или Push клиент регистрирует конечную точку уведомления клиента значением параметра <backchannel_client_notification_endpoint>. Клиенты, использующие подписанные запросы аутентификации, регистрируют используемый алгоритм подписи значением параметра <backchannel_authentication_request_signing_alg>.

6.2.2.1. Дополнительные метаданные клиента

ДОПОЛНИТЕЛЬНЫЕ МЕТАДАННЫЕ КЛИЕНТА *Табл. 2*

Наименование параметра	Кратность	Определение	Примечание
backchannel_token_delivery_mode	1..1	Режим доставки токена. Принимает одно из следующих значений: «poll», «ping» или «push»	Должно быть указано, если параметр <grant_types> включает «urn: openid: params: grant-type: ciba»
backchannel_client_notification _endpoint	1..1	Конечная точка (URL-адрес HTTPS), на которую сервер авторизации отправит уведомление после успешной или неудачной аутентификации конечного пользователя	Должно быть указано, если параметр <grant_types> включает «urn: openid: params: grant-type: ciba» и параметр «backchannel_token_delivery_mode» определен как «ping» или «push»
backchannel_authentication_request _signing_alg	0..1	Алгоритм JWS, используемый как значение параметра <alg>, который клиент использует для подписания запроса аутентификации. Если не указан, клиент не отправляет подписанные запросы аутентификации	Должно быть указано, если параметр <grant_types> включает «urn: openid: params: grant-type: ciba»
backchannel_user_code_parameter _supported	0..1	Логическое значение, указывающее, поддерживает ли клиент параметр <user_code>	В случае отсутствия параметра значением по умолчанию является «false»

При выполнении прямых запросов клиентом к серверу авторизации значение параметра, определяющего метод аутентификации конечной точки токена <token_endpoint_auth_method>, должно использоваться для запросов как к конечной точке токена, так и к конечной точке аутентификации по отдельному каналу.

Пример запроса динамической регистрации клиента

```
POST /connect/register HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: server.example.com
Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.eyJ …
{
“application_type”: “web”,
“client_name”: “My Example”,
“logo_uri”: “https://client.example.ru/logo.png”,
“subject_type”: “pairwise”,
“token_endpoint_auth_method”: “private_key_jwt”,
“grant_types”: [“urn: openid: params: grant-type: ciba”],
“backchannel_token_delivery_mode”: “poll”,
“jwks_uri”: “https://client.example.org/my_public_keys.jwks”,
“contacts”: [“ve7aft@example.ru”, “andy@example.ru”]
}
```

6.3. КОНЕЧНАЯ ТОЧКА АУТЕНТИФИКАЦИИ ПО ОТДЕЛЬНОМУ КАНАЛУ

Настоящий стандарт определяет конечную точку аутентификации по отдельному каналу, которая используется для инициирования аутентификации конечного пользователя непосредственно у сервера авторизации путем отправки в нее запроса аутентификации от клиента. Передача сообщений между клиентом и сервером авторизации на конечной точке аутентификации по отдельному каналу должна производиться по аутентифицированному TLS-соединению в соответствии с требованиями к протоколу TLS, определенными в пункте 5.8.3 ФАПИ.СЕК. Взаимодействие AD с сервером авторизации и CD с клиентом AD в процессе аутентификации должно осуществляться с использованием протокола TLS.

6.3.1. Запрос аутентификации

Запрос аутентификации по отдельному каналу выполняется напрямую от клиента к серверу авторизации, без прохождения через браузер конечного пользователя. Клиент должен послать запрос аутентификации к серверу авторизации, создав запрос «HTTP POST», который предоставит всю необходимую информацию для аутентификации конечного пользователя без запроса его идентификационных данных.

Клиент аутентифицируется в конечной точке аутентификации по отдельному каналу зарегистрированным у сервера авторизации для его <client_id> методом, определенным в параметре <token_endpoint_auth_method> его метаданных. Методы аутентификации клиента регулируются требованиями обеспечения безопасности авторизации, применяемыми к серверу авторизации в зависимости от применяемого профиля безопасности OPENID API. При применении профиля безопасности для доступа к сервисам в режиме только для чтения допускается использование TLS с взаимной аутентификацией сторон взаимодействия протокола OAuth 2.0, <client_secret_jwt> или <private_key_WT> (подпункт 6.2.2.4 ФАПИ.СЕК). При применении профиля безопасности для доступа к сервисам в режиме чтения и записи допускается использование только TLS с взаимной аутентификацией сторон взаимодействия протокола OAuth 2.0 и <private_key_jwt> (подпункт 7.2.2.12 ФАПИ.СЕК).

При использовании TLS с взаимной аутентификацией сторон взаимодействия протокола OAuth 2.0 необходимо руководствоваться требованиями, определенными в пунктах 5.5.4 или 5.5.5 ФАПИ.СЕК.

При использовании механизмов аутентификации «client_secret_jwt» (пункт 5.5.2 ФАПИ.СЕК) и «private_key_jwt» (пункт 5.5.3 ФАПИ.СЕК) в параметре <client_assertion> в качестве заявленного значения аудитории (<aud>) следует использовать идентификатор эмитента сервера авторизации (параметр метаданных сервера авторизации <issuer> (подпункт 5.4.4.3 ФАПИ.СЕК)). Для обеспечения взаимодействия при запросе аутентификации сервер авторизации должен принимать данный идентификатор эмитента или URL конечной точки аутентификации по отдельному каналу в качестве значения, идентифицирующего его как целевую аудиторию (<aud>).

В состав запроса аутентификации клиент может включать следующие параметры:

- <scope>: (обязательный) область действия, определяет свойства защищаемых данных конечного пользователя, к которым запрошен доступ. В случае использования протокола OpenID Connect параметр <scope> должен содержать строку «openid»; (подпункт 5.4.2.2 ФАПИ.СЕК). Параметр <scope> может содержать и другие значения, которые определяются на этапе разработки сервера авторизации исходя из его прикладных целей и задач.

  Примечание. Дополнительные сведения об использовании параметра <scope> приведены в разделе 3.3 *[RFC6749]*.

- <client_notification_token>: (обязательный, если клиент зарегистрирован для использования режимов Ping или Push) предоставляемый клиентом токен на предъявителя, который будет использоваться сервером авторизации для аутентификации выполняемого клиентом запроса обратного вызова. Длина токена не должна превышать 1024 символов, и он должен соответствовать синтаксису токена на предъявителя.

  Примечание. Дополнительные сведения приведены в разделе 2.1 *[RFC6750]*.

- <acr_values>: (опциональный) запрошенные значения класса контекста аутентификации. Разделенная пробелами строка, определяющая идентификаторы классов контекста аутентификации, отображаемые в порядке предпочтения, которые сервер авторизации запрашивает для обработки запроса аутентификации ([OpenID. Core] подпункт 5.5.1.1). Средства аутентификации конечного пользователя имплементируются сервером авторизации, и требуемый класс контекста аутентификации возвращается в качестве заявленного свойства <acr> ID токена. При наличии параметра <acr_values> в запросе аутентификации полученный ID токен должен содержать значение заявленного свойства <acr>.

  Примечание. Используемые значения идентификаторов <acr> определяются участниками взаимодействия, использующими данное заявленное свойство и должны однозначно определять методы и факторы аутентификации [ГОСТ Р 58833–2020].

- <login_hint_token>: (опциональный) информация в виде токена, идентифицирующая конечного пользователя, для которого запрашивается аутентификация. Механизм формирования данного параметра определяется на этапе разработки сервера авторизации исходя из его прикладных целей и задач.
- <id_token_hint>: (опциональный) информация о пользователе в виде ID токена, ранее выданного клиенту сервером авторизации, в качестве рекомендации идентификации конечного пользователя.
- <login_hint>: (опциональный) информация о пользователе для сервера авторизации относительно конечного пользователя, для которого запрашивается аутентификация. Значение содержит идентификационные данные конечного пользователя, по которым сервер авторизации может однозначно его идентифицировать (адрес электронной почты, номер телефона, идентификатор субъекта), и может быть предварительно получено от конечного пользователя клиентом. Механизм формирования данного параметра определяется на этапе разработки сервера авторизации исходя из его прикладных целей и задач.
- <binding_message>: (опциональный) идентификатор или сообщение, предназначенные для отображения как на CD, так и на AD, чтобы связать их вместе для транзакции посредством визуальной информации для конечного пользователя. Это связывающее сообщение позволяет конечному пользователю убедиться, что действие, предпринятое на AD, связано с запросом, инициированным CD (например, код подтверждения транзакции). Поскольку различные устройства могут иметь ограниченные возможности отображения и сообщение предназначено для визуального просмотра конечным пользователем, значение <binding_message> должно включать не более 100 символов и использовать ограниченный набор символов в виде простого текста: алфавитно-цифровые символы «А – Я», «а – я», «A – Z», «a – z» и «0 – 9» и знаки «_», «!». Если предоставленное значение <binding_message> является неприемлемым, клиенту возвращается ошибка «invalid_binding_message».
- <user_code>: (опциональный) секретный код, известный только пользователю, но проверяемый сервером авторизации. Код используется для авторизации процесса отправки запроса аутентификации на AD пользователя. Этот параметр присутствует, если значение параметра метаданных клиента <backchannel_user_code_parameter> соответствует поддержке кода пользователя (имеет значение «true»).
- <requested_expiry>: (опциональный) значение времени жизни для запроса аутентификации – положительное целое число, позволяющее ограничивать по времени и корректного завершать сессии аутентификации.

  Примечание. Запрос аутентификации также может содержать дополнительные параметры, которые определяются на этапе разработки сервера авторизации исходя из его прикладных целей и задач.

Поскольку в потоке аутентификации по отдельному каналу сервер авторизации не взаимодействует с конечным пользователем через CD, клиент должен указывать один из параметров, содержащих информацию о пользователе: <login_hint_token>, <id_token_hint> или <login_hint>.

Запрос аутентификации выполняется с использованием метода HTTP POST в формате «application/x-www-form-urlencoded» и кодировкой символов UTF-8 в теле объекта HTTP-запроса.

В случае аутентификации применения механизмов аутентификации «client_secret_jwt» (пункт 5.5.2 ФАПИ.СЕК) и «private_key_jw» (пункт 5.5.3 ФАПИ.СЕК) используются дополнительные параметры <client_assertion> и <client_assertion_type>.

Пример запроса аутентификации

```
POST /bc-authorize HTTP/1.1

Host: server.example.ru

Content-Type: application/x-www-form-urlencoded

scope=openid%20email%20example-scope&

client_notification_token=8d67dc78-7faa-4d41-aabd-67707b374255&

binding_message=W4SCT&

login_hint_token=eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.     eyJzdWJfaWQiOnsic3Viam-
VjdF90eXBlIjoicGhvbmUiLCJwaG9uZSI6Iis3MTIzMDAwMDAwMSJ9fQ.A9kJnX3IUAFHsic0nUfXKVN-
lGDWLuo2L80y2GLMsmn82xJ8yu6wA5k3pIepOUxrBd74aD_CVFHgJ3qVibjRBmawThhOQMNrgoqTGfR-
NUWR8A9ogLogXHCSZEo7oCI1D4zxHhsDeSu0Mby61sLmr_uchscsEkSsfnKpmMpyaIn53wWdIu3OI5B-
VMALm7EMZGuweReM5NTKfCUNjXNmp5w&

client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Ajwt-bearer&

client_assertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.eyJpc3MiOiJzNkJoZFJrcXQ-
zIiwic3ViIjoiczZCaGRSa3F0MyIsImF1ZCI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUucnUiLCJqdGki-
OiJiZGMtWHNfc2YtM1lNbzRGU3pJSjJRIiwiaWF0IjoxNTkxMDc2NzE2LCJleHAiOjE1OTIwNzY3MTZ9.
FuBj8SjN3bufQae1HgaO6BoSp4ocrnl4dnTZLJE0KZpnafAA88Ra7eoUT2QP1LHtetVAlDcT3nRJmyyRNL-
c7oXTYtb5mqCbNYw4ThTm9hIgyV339hDO837vxX_GNzP6zY5nKA9nyqyRG3BbPldBkzanyQD1F0JtAVugnBZw
```

Декодированный login_hint_token

```
{
  “sub_id”: {
    “subject_type”: “phone”,
    “phone”: “+71230000001”
  }
}
```

Декодированный JWT параметр «client_assertion»

```
{
  “iss”: “s6BhdRkqt3”,
  “sub”: “s6BhdRkqt3”,
  “aud”: “https://server.example.ru”,
  “jti”: “bdc-Xs_sf-3YMo4FSzIJ2Q”,
  “iat”: 1591076716,
  “exp”: 1592076716
}
```

6.3.1.1. Подписанный запрос аутентификации

Подписанный запрос аутентификации создается посредством кодирования всех параметров запроса аутентификации в виде заявленных свойств подписанного JWT (подраздел 5.7 ФАПИ.СЕК), с представлением каждого наименования параметра запроса в качестве имени заявленного свойства (claim), а его значения – в виде строки JSON. Подписанный запрос аутентификации передается в качестве параметра «request» запроса «HTTP POST».

Примечание. Как исключение, параметры <requested_expiry> могут быть представлены также числом JSON.

При этом к JWT предъявляются следующие требования:

1) должен содержать все параметры запроса аутентификации;
2) быть защищенным асимметричной подписью (подпункт 5.8.1.2 ФАПИ.СЕК);
3) содержать следующие зарегистрированные заявленные свойства (claims):

- <aud>: аудитория, значение идентификатора эмитента сервера авторизации, который идентифицирует сервер авторизации как предполагаемую аудиторию;
- <iss>: источник, должен содержать идентификатор клиента;
- <exp>: время ограничения срока действия подписанного запроса аутентификации;
- <iat>: время создания подписанного запроса аутентификации;
- <nbf>: время, до которого подписанный запрос аутентификации недопустим;
- <jti>: уникальный идентификатор подписанного запроса аутентификации.

Пример подписанного запроса аутентификации

```
POST /bc-authorize HTTP/1.1

Host: server.example.ru

Content-Type: application/x-www-form-urlencoded

request=eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.      eyJpc3MiOiJzNkJoZFJrcXQzIiwiYX-
VkIjoiaHR0cHM6Ly9zZXJ2ZXIuZXhhbXBsZS5ydSIsImV4cCI6MTUzNzgyMDA4NiwiaWF0IjoxNTM3O-
DE5NDg2LCJuYmYiOjE1Mzc4MTg4ODYsImp0aSI6IjRMVENxQUNDMkVTQzVCV0NuTjNqNThFbkEiLCJzY-
29wZSI6Im9wZW5pZCBlbWFpbCBleGFtcGxlLXNjb3BlIiwiY2xpZW50X25vdGlmaWNhdGlvbl90b2tlbiI6I-
jhkNjdkYzc4LTdmYWEtNGQ0MS1hYWJkLTY3NzA3YjM3NDI1NSIsImJpbmRpbmdfbWVzc2FnZSI6Il-
c0U0NUIiwibG9naW5faGludF90b2tlbiI6ImV5SjBlWEFpT2lKS1YxUWlMQ0poYkdjaU9pSkhUMU5VTX-
pReE1DSjkuZXlKemRXSmZhV1FpT25zaWMzVmlhbVZqZEY5MGVYQmxJam9pY0dodmJtVWlMQ0p3YUc5dVpT-
STZJaXMzTVRJek1EQXdNREF3TVNKOWZRLkE5a0puWDNJVUFGSHNpYzBuVWZYS1ZObEdEV0x1bzJMOD-
B5MkdMTXNtbjgyeEo4eXU2d0E1azNwSWVwT1V4ckJkNzRhRF9DVkZIZ0ozcVZpYmpSQm1hd1RoaE9R-
TU5yZ29xVEdmUk5VV1I4QTlvZ0xvZ1hIQ1NaRW83b0NJMUQ0enhIaHNEZVN1ME1ieTYxc0xtcl91Y2hzY3N-
Fa1NzZm5LcG1NcHlhSW41M3dXZEl1M09JNUJWTUFMbTdFTVpHdXdlUmVNNU5US2ZDVU5qWE5tcDV3In0.
LH3heoqfD18vh9ljE28NWKmKQWkztaXY2GBMzDePM9yMz5L2vdvuTQ_5p9c6szleZBwYwEcxUv4P9uqIvF6zX-
8Mt_qXY1sGomG1x33EwbtKdqYqFjtAwE1FVHLOEA

&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A

&client-assertion-type%3Ajwt-bearer

&client_assertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.eyJpc3MiOiJzNkJoZFJrcXQ-
zIiwic3ViIjoiczZCaGRSa3F0MyIsImF1ZCI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUucnUiLCJqdGki-
OiJiZGMtWHNfc2YtM1lNbzRGU3pJSjJRIiwiaWF0IjoxNTkxMDc2NzE2LCJleHAiOjE1OTIwNzY3MTZ9.
FuBj8SjN3bufQae1HgaO6BoSp4ocrnl4dnTZLJE0KZpnafAA88Ra7eoUT2QP1LHtetVAlDcT3nRJmyyRNL-
c7oXTYtb5mqCbNYw4ThTm9hIgyV339hDO837vxX_GNzP6zY5nKA9nyqyRG3BbPldBkzanyQD1F0JtAVugnBZw
```

Декодированный JWT параметра «request»


```
{
“iss”: “s6BhdRkqt3”,
“aud”: “https://server.example.ru”,
“exp”: 1537820086,
“iat”: 1537819486,
“nbf”: 1537818886,
“jti”: “4LTCqACC2ESC5BWCnN3j58EnA”,
“scope”: “openid email example-scope”,
“client_notification_token”: “8d67dc78-7faa-4d41-aabd-67707b374255”,
“binding_message”: “W4SCT”,
“login_hint_token”:”eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.eyJzdWJfaWQiOnsic3Viam-
VjdF90eXBlIjoicGhvbmUiLCJwaG9uZSI6Iis3MTIzMDAwMDAwMSJ9fQ.A9kJnX3IUAFHsic0nUfXKVN-
lGDWLuo2L80y2GLMsmn82xJ8yu6wA5k3pIepOUxrBd74aD_CVFHgJ3qVibjRBmawThhOQMNrgoqTGfR-
NUWR8A9ogLogXHCSZEo7oCI1D4zxHhsDeSu0Mby61sLmr_uchscsEkSsfnKpmMpyaIn53wWdIu3OI5B-
VMALm7EMZGuweReM5NTKfCUNjXNmp5w”
}
```


6.3.1.2. Код пользователя

Поток аутентификации по отдельному каналу в качестве механизма предотвращения создания нежелательных запросов аутентификации на AD конечного пользователя поддерживает использование параметра <user_code> – код пользователя, секрет конечного пользователя, известный серверу авторизации.

Примечание. Запрещено использовать в качестве <user_code> пароль учетной записи конечного пользователя, зарегистрированной у сервера авторизации.

При поддержке параметра <user_code> сервером авторизации клиент запрашивает у конечного пользователя значение кода пользователя для снятия риска создания запросов на аутентификацию для конечного пользователя третьими лицами, знающими <login_hint> или другие идентификаторы конечного пользователя. В случае использования нестатичных <login_hint> сервер авторизации может принимать запросы без <user_code>.

Сервер авторизации объявляет о поддержке кода пользователя значением параметра метаданных <backchannel_user_code_parameter_supported>. Параметр регистрации клиента <backchannel_user_code_parameter> указывает, поддерживает ли клиент параметр <user code>.

Клиент запрашивает у конечного пользователя код пользователя для каждого потока аутентификации и не должен хранить его.

Механизм регистрации пользователем <user_code> на сервере авторизации выходит за рамки содержания настоящего стандарта. Сервер авторизации определяет механизм регистрации исходя из его прикладных целей и задач, а также должен предоставлять пользователю возможность изменения значения <user_code>.

6.3.2. Проверка запроса аутентификации

Сервер авторизации должен проверить полученный запрос аутентификации следующим образом:

1. Аутентифицировать клиента зарегистрированным для него методом аутентификации (пункт 5.5.1 ФАПИ.СЕК).
2. Если запрос аутентификации подписан, проверить переданный параметром «request» JWT на корректность его структуры и подписи (подраздел 5.7 ФАПИ.СЕК).
3. Проверить наличие всех обязательных параметров в запросе аутентификации.
4. Проверить значения параметров. Если запрос содержит более одного или ни одного указания на пользователя (<login_hint_token>, <id_token_hint> или <login_hint>), сервер авторизации возвращает ответ об ошибке «invalid_request».

5. Проверить предоставленную информацию о пользователе на корректность состава параметров и актуальность связанного с ним конечного пользователя, а в случае применения подписанного указания (<id_token_hint> или <login_hint_token>) проверить его подпись. Механизм проверки информации о пользователе и способы извещения клиента о требованиях к ней определяются на этапе разработки сервера авторизации исходя из его прикладных целей и задач.
6. Если информация о пользователе недействительна или если сервер авторизации не может определить конечного пользователя, то клиенту возвращается ответ об ошибке.

Сервер авторизации должен игнорировать нераспознанные параметры запроса. При непрохождении проверки запроса аутентификации сервер авторизации возвращает ответ об ошибке (подраздел 6.8).

6.3.3. Успешное подтверждение запроса аутентификации

Если запрос аутентификации подтвержден, сервер авторизации вернет клиенту ответ HTTP «200 OK», содержащий следующие параметры:

- <auth_req_id>: (обязательный) уникальный идентификатор запроса аутентификации от клиента. Должен быть случайным и содержать достаточную энтропию (минимум 160 бит). Необходимо использовать символы «A» – «Z», «a» – «z», «0» – «9», «.», «-» и «_» для поддержки unpadded base64url. Значение идентификатора не должно указывать клиенту на связь с другими данными или иметь смысловую нагрузку.
- <expires_in>: (обязательный) JSON с положительным целочисленным значением, указывающим время истечения действия <auth_req_id> в секундах с момента получения запроса аутентификации. Клиенту возвращается ответ об ошибке в случае обращения к конечной точке токена с истекшим идентификатором <auth_req_id>.
- <interval>: (опциональный) JSON с положительным целочисленным значением, указывающим минимальное количество времени в секундах, которое клиент ожидает между запросами на опрос к конечной точке токена. Указывается, если клиент зарегистрирован для использования режимов Poll или Ping. Если значение не указано, клиент использует значения по умолчанию «5».

Пример ответа на запрос аутентификации

```
HTTP/1.1 200 OK
    Content-Type: application/json
    Cache-Control: no-store
     {
      "auth_req_id": "1c266114-a1be-4252-8ad1-04986c5b9ac1",
      "expires_in": 120,
      "interval": 2
    }
```

6.3.4. Проверка положительного ответа на запрос аутентификации

При получении в ответе аутентификации значения HTTP «200 OK» клиент проверяет наличие требуемых параметров и сохраняет значение <auth_req_id> для проверки обратных вызовов, полученных в режимах Ping и Push, или для запроса токена в режимах Poll и Ping, а также сохраняет значение <expires_in> для удаления запросов аутентификации, по которым не получены ответы в обратных вызовах Ping или Push.

6.3.5. Получение сервером авторизации согласия/авторизации конечного пользователя

После того как сервер авторизации проверил запрос аутентификации, он идентифицирует конечного пользователя, выбирает способ аутентификации и авторизации запроса в соответствии с запрашиваемым методом и инициирует интерактивную сессию с AD конечного пользователя (подпункт 5.4.2.7 ФАПИ.СЕК). Как только конечный пользователь аутентифицирован, сервер авторизации должен получить результат авторизации запроса о согласии передавать информацию клиенту (подпункт 5.4.2.8 ФАПИ.СЕК).

6.4. КОНЕЧНАЯ ТОЧКА УВЕДОМЛЕНИЯ КЛИЕНТА

Конечная точка уведомления клиента – конечная точка, установленная клиентом во время процесса регистрации и обнаружения метаданных, которую сервер авторизации вызывает после успешной или неудачной аутентификации конечного пользователя. Передача сообщений между клиентом и сервером авторизации на конечной точке уведомления клиента должна производиться по аутентифицированному TLS-соединению в соответствии с требованиями к протоколу TLS, определенному в пункте 5.8.3 ФАПИ.СЕК.

Когда клиент настроен в режиме Ping, конечная точка получает уведомление от сервера авторизации о готовности извлечения результата аутентификации из конечной точки токена.

Когда клиент настроен в режиме Push, конечная точка уведомления клиента получает результат аутентификации (ID токен, токен доступа и (опционально) токен обновления или сообщение об ошибке авторизации).

Запросы к конечной точке уведомления клиента аутентифицируются с использованием токена на предъявителя, созданного клиентом и отправленного серверу авторизации в запросе аутентификации в качестве значения параметра <client_notification_token>.

Примечание. При формировании запроса к конечной точке уведомления клиента необходимо руководствоваться требованиями подраздела 7.3 ФАПИ.СЕК.

6.5. ПОЛУЧЕНИЕ РЕЗУЛЬТАТА АУТЕНТИФИКАЦИИ

Режим доставки токена для клиента (Poll, Ping или Push) определяется при регистрации клиента (параметр метаданных <backchannel_token_delivery_mode>). Клиент может зарегистрировать только один метод доставки токена, а сервер авторизации возвращает клиенту результат аутентификации только через зарегистрированный режим.

6.5.1. Запрос токена

Если клиент зарегистрирован для использования режимов Poll или Ping, то клиент получает результат аутентификации из конечной точки токена после прохождения аутентификации.

Передача сообщений между клиентом и сервером авторизации на конечной точке уведомления клиента должна производиться по аутентифицированному TLS-соединению в соответствии с требованиями к протоколу TLS, определенному в пункте 5.8.3 ФАПИ.СЕК.

Применимые методы аутентификации клиента на конечной точке токена определены в пункте 5.5.1 ФАПИ.СЕК.

Примечание. Подробная информация о параметрах запроса аутентификации клиента приведена в подразделе 5.5 ФАПИ.СЕК.

6.5.1.1. Особенности запроса токена в режиме Poll

Если клиент зарегистрирован для использования режима Poll, то он опрашивает конечную точку токена с интервалом, ограниченным сервером авторизации значением параметра ответа аутентификации <interval>. При этом сервер авторизации может осуществлять длительный опрос, когда сервер авторизации отвечает на запрос токена, только если результат аутентификации стал доступен или истекло время ожидания ответа. Для длительных опросов рекомендуется использовать интервал 30 секунд.

Примечание. Рекомендации по опросу приведены в подразделе 5.5 *[RFC6202]*.

Клиенты должны быть готовы ждать ответа не менее 30 секунд при использовании режима Poll. Сервер авторизации должен ответить в течение не более 30 секунд, даже при использовании длительного опроса. Интервал опроса измеряется с момента отправки запроса Poll. Для осуществления длительного опроса сервер авторизации может отвечать медленнее, чем предусмотрено интервалом. Клиент не должен отправлять два перекрывающихся запроса с одним и тем же значением <auth_req_id> и ждать получения ответа на предыдущий запрос, прежде чем отправлять следующий запрос. В случае если ответ получен, а интервал прошел, клиент может немедленно отправить следующий запрос. Для управления данной ситуацией сервер авторизации возвращает HTTP «503» с заголовком «Retry-After».

Примечание. Подробная информация приведена в пункте 7.1.3 *[RFC7231]*.

Примеры поведения клиента в зависимости от различных ответов сервера авторизации с учетом ожидания в течение пяти секунд:

- **Длинный опрос.** Клиент делает запрос токена, а сервер авторизации возвращает ошибку <authorization_pending> через 30 секунд. В этом случае клиент может сразу выполнить следующий запрос токена.
- **Стандартный опрос.** Клиент делает запрос токена, а сервер авторизации возвращает ошибку <authorization_pending> через две секунды. В этом случае клиент ждет три секунды, прежде чем сделать следующий запрос.

– **Сервер авторизации реагирует медленнее 30 секунд.** Клиент делает запрос токена, и сервер авторизации не возвращает ответ в течение 30 секунд. В этом случае клиент отменяет текущий запрос и выполняет новый.

6.5.1.2. Особенности запроса токена в режиме Ping

Если клиент зарегистрирован для использования режима Ping, то он получает на конечной точке уведомления клиента ответ с <auth_req_id>, проверенный по параметру <client_notification_token>, и вызывает конечную точку токена для получения результата аутентификации.

Примечание. Клиент, настроенный в режиме Ping, может также опросить конечную точку токена. Сервер авторизации должен обращаться с таким клиентом, как если бы он был зарегистрирован для использования режима Poll.

6.5.1.3. Параметры запроса токена

Клиент отправляет запрос HTTP POST на конечную точку токена в формате «application/x-www-form-urlencoded» со следующими параметрами:

– <grant_type>: (обязательный) – должен содержать значение «urn: openid: params: grant-type: ciba»;
– <auth_req_id>: (обязательный) – уникальный идентификатор запроса аутентификации от клиента. Сервер авторизации проверяет принадлежность <auth_req_id> клиенту в ответе на запрос аутентификации. В противном случае возвращается ошибка.

Пример запроса токена

```
POST /token HTTP/1.1
Host: server.example.ru
Content-Type: application/x-www-form-urlencoded
grant_type=urn%3Aopenid%3Aparams%3Agrant-type%3Aciba&
auth_req_id=1c266114-a1be-4252-8ad1-04986c5b9ac1&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-bearer&
client_assertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.eyJpc3MiOiJ
zNkJoZFJrcXQzIiwic3ViIjoiczZCaGRSa3F0MyIsImF1ZCI6Imh0dHBzOi8vc2Vy
dmVyLmV4YW1wbGUucnUvdG9rZW4iLCJqdGkiOiItX3AxNmo2SGNpWG8zMTdodlozM
TJjIiwiaWF0IjoxNTM3ODE5NDkxLCJleHAiOjE1OTEwNzc1OTN9.qQ-26mPcsCMyM
zsA5JV6Pv-O_I_-_43jxqsUMvdgoi4
```

Декодированный client_assertion

```
{
  «iss»: «s6BhdRkqt3»,
  «sub»: «s6BhdRkqt3»,
  «aud»: «https://server.example.ru/token”,
  «jti»: «-_p16j6HciXo317hvZ312c»,
  «iat»: 1537819491,
  «exp»: 1537819782
}
```

6.5.1.4. Успешный ответ токена

Примечание. При формировании успешного ответа необходимо руководствоваться общими требованиями, определенными в пункте 7.2.3 ФАПИ.СЕК.

После получения от клиента, проверки актуальности и авторизации запроса токена, а затем после аутентификации связанного с предоставленным <auth_req_id> конечного пользователя и авторизации им запроса сервер авторизации возвращает успешный ответ токена (подпункт 5.4.2.14 ФАПИ.СЕК).

После успешного обмена на токен доступа представленное значение <auth_req_id> становится неактуальным. Если конечный пользователь, связанный с предоставленным идентификатором <auth_req_id>, не был аутентифицирован, не авторизовал запрос или клиент использует неактуальный <auth_req_id>, сервер авторизации возвращает ответ об ошибке (раздел 6.6).

Пример успешного ответа токена

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
{
“access_token”: “G5kXH2wHvUra0sHlDy1iTkDJgsgUO1bN”,
“token_type”: “Bearer”,
“refresh_token”: “4bwc0ESC_IAhflf-ACC_vjD_ltc11ne-8gFPfA2Kx16”,
“expires_in”: 120,
“id_token”: “eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.eyJpc3MiOiJodH
RwczovL3NlcnZlci5leGFtcGxlLnJ1Iiwic3ViIjoiMjQ4Mjg5NzYxMDAxIiwiYX
VkIjoiczZCaGRSa3F0MyIsImVtYWlsIjoiamFuZWRvZUBleGFtcGxlLnJ1IiwiZX
hwIjoxNTkxMDc3NjUxLCJpYXQiOjE1Mzc4MTk1MDMsImp0aSI6IjUzN2YwMGQ1LT
g1NGItNGJkYy05MzBlLTM5NmRkZDc2MDQwZCJ9.WOFhMAJ4OQ6Byu0-9oNmB0jDz
PZoEqS8SdfSVhZxnMI”
}
```

Декодированный id_token

```
{
  “iss”: “https://server.example.ru”,
  “sub”: “248289761001”,
  “aud”: “s6BhdRkqt3”,
  “email”: “janedoe@example.ru”,
  “exp”: 1537819803,
  “iat”: 1537819503
}
```

6.5.2. Обратный вызов в режиме Ping

Если клиент зарегистрирован в режиме Ping, сервер авторизации после успешной или неудачной аутентификации конечного пользователя отправляет запрос в конечную точку уведомления клиента. В этом режиме сервер авторизации указывает в заголовке авторизации полученное значение <client_notification_token> в качестве токена на предъявителя и <auth_req_id> в теле запроса в формате application/json.

Пример обратного вызова Ping к конечной точке уведомления клиента

```
POST /cb HTTP/1.1
Host: client.example.ru
Authorization: Bearer 8d67dc78-7faa-4d41-aabd-67707b374255
Content-Type: application/json

{
 "auth_req_id": "1c266114-a1be-4252-8ad1-04986c5b9ac1"
}
```

6.5.2.1. Проверка обратного вызова в режиме Ping

Клиент проверяет, что полученный <client_notification_token>, используемый для аутентификации запроса как токен на предъявителя, действителен и связан с <auth_req_id>. Если токен на предъявителя недействителен, возвращается сообщение об ошибке «HTTP 401 Unauthorized response». Для действительных запросов к конечной точке уведомления клиента возвращается ответ HTTP «204 No Content». Сервер авторизации также должен принимать ответы HTTP «200 OK», а тело в ответе игнорировать. Клиент не должен возвращать код HTTP «3xx», а сервер авторизации – следовать за перенаправлениями.

Для действительных запросов клиент использует полученный <auth_req_id> для создания запроса на токен доступа к конечной точке токена с использованием типа гранта «urn: openid: params: grant-type: ciba».

Примечание. Обработка кодов ошибок HTTP в диапазонах 4xx и 5xx определена подразделами 5.5 и 5.6 спецификации протокола HTTP *[RFC7231]*.

6.5.3. Обратный вызов Push

6.5.3.1. Успешный ответ токена

Примечание. При формировании успешного ответа токена в режиме Push необходимо руководствоваться общими требованиями, определенными в подпункте 5.4.2.14 ФАПИ.СЕК.

Если клиент зарегистрирован в режиме Push, а конечный пользователь прошел аутентификацию и авторизовал запрос, сервер авторизации передает для конечной точки уведомления клиента сообщение, включая в полезную нагрузку в формате «application/json»: ID токен, токен доступа, (опционально) токен обновления и параметр <auth_req_id>.

Для привязки ID токена, токена доступа и <auth_req_id> сервер авторизации включает значение хэш-функции токена доступа и <auth_req_id> в токене идентификации, используя заявленные свойства <at_hash> и <urn: openid: params: jwt: claim: auth_req_id> соответственно. В случае отправки токена обновления значение его хэш-функции присутствует в токене идентификации в качестве заявленного свойства <urn: openid: params: jwt: claim: rt_hash>.

Значение параметра <at_hash> определяется в соответствии с требованиями подпункта 5.4.3.4 ФАПИ.СЕК, а значение параметра <urn: openid: params: jwt: claim: auth_req_id> аналогично <at_hash> с учетом использования <auth_req_id>. При этом применяется алгоритм хэширования, указанный в параметре <alg> JOSE заголовка ID токена. Этот же метод используется для вычисления хэш-значения токена обновления (параметр <urn: openid: params: jwt: claim: rt_hash> ID токена).

Примечание. Данное требование применимо только для режима Push.

Пример обратного вызова Push

```
POST /cb HTTP/1.1
Host: client.example.com
Authorization: Bearer 8d67dc78-7faa-4d41-aabd-67707b374255
Content-Type: application/json

{
“auth_req_id”: “1c266114-a1be-4252-8ad1-04986c5b9ac1”,
“access_token”: “G5kXH2wHvUra0sHlDy1iTkDJgsgUO1bN”,
“token_type”: “Bearer”,
“refresh_token”: “4bwc0ESC_IAhflf-ACC_vjD_ltc11ne-8gFPfA2Kx16”,
“expires_in”: 120,
“id_token”: “eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.eyJpc3MiOiJodHRwczovL3Nl
cnZlci5leGFtcGxlLnJ1Iiwic3ViIjoiMjQ4Mjg5NzYxMDAxIiwiYXVkIjoiczZCaGRSa3F0My
IsImVtYWlsIjoiamFuZWRvZUBleGFtcGxlLnJ1IiwiZXhwIjoxNTkxMDc3ODY2LCJpYXQiOjE1
Mzc4MTk1MDMsImF0X2hhc2giOiJXdDBrVkZYTWFjcXZuSGV5VTAwMDF3IiwidXJuOm9wZW5pZD
pwYXJhbXM6and0OmNsYWltOnJ0X2hhc2giOiJzSGFoQ3VTcFhDUmc1bWtERHZ2cjR3IiwidXJu
Om9wZW5pZDpwYXJhbXM6and0OmNsYWltOmF1dGhfcmVxX2lkIjoiMWMyNjYxMTQtYTFiZS00Mj
UyLThhZDEtMDQ5ODZjNWI5YWMxIiwianRpIjoiMWRlMTI5NDYtZDYyYy00ZjRkLTg1MzItNzc0
ZDkyYjFkYjUzIn0.WE3SpNtWJYQb6NDLO9YqNmq9svF2v6YBbjieblpZ6qY”
}
```

6.5.3.2. Проверка ответа токена

Клиент проверяет, что полученный <client_notification_token>, используемый для аутентификации запроса как токен на предъявителя, действителен и связан со значением параметра <auth_req_id>. Если токен на предъявителя недействителен, возвращается сообщение об ошибке «HTTP 401 Unauthorized response». Клиент проверяет ID токен (параметр <id_token>).

Примечание. Подробное описание проверки ID токена приведено в подпункте 5.4.2.17 ФАПИ.СЕК.

Клиент проверяет соответствие значения заявленного свойства <urn: openid: params: jwt: claim: auth_req_id> в <id_token> значению параметра <auth_req_id> в запросе аутентификации.

Клиент проверяет полученный токен доступа с помощью значения <at_hash> из ID токена путем сравнения значения base64url-кодирования левой половины хэш-кода полученного значения <access_token> с полученным значением параметра <at_hash> и аналогичным образом токен обновления (при наличии параметра <refresh_token>), используя <urn: openid: params: jwt: claim: rt_hash> из ID токена.

Примечание. Проверка производится в соответствии с рекомендациями подпункта 5.4.3.5 ФАПИ.СЕК.

Для действительных запросов к конечной точке уведомления клиента возвращается HTTP «204 No Content». Сервер авторизации также должен принимать ответы HTTP «200 OK», а тело в ответе игнорировать. Клиент не должен возвращать код HTTP «3xx», а сервер авторизации – следовать за перенаправлениями.

Примечание. Обработка кодов ошибок HTTP в диапазонах 4xx и 5xx определена подразделами 5.5 и 5.6 спецификации протокола HTTP *[RFC7231]*.

Пример декодированного ID токена

```
{
  “iss”: “https://server.example.ru”,
  “sub”: “248289761001”,
  “aud”: “s6BhdRkqt3”,
  “email”: “janedoe@example.ru”,
  “exp”: 1537819803,
  “iat”: 1537819503,
  “at_hash”: “Wt0kVFXMacqvnHeyU0001w”,
  “urn:openid:params:jwt:claim:rt_hash”: “sHahCuSpXCRg5mkDDvvr4w”,
  “urn:openid:params:jwt:claim:auth_req_id”:”1c266114-a1be-4252-8ad1-04986c5b9ac1”
}
```

6.6. ОТВЕТ ОБ ОШИБКЕ ТОКЕНА

Если запрос токена является недействительным или неавторизованным, сервер авторизации создает ответ об ошибке «400 Bad Request» в дополнение к кодам ошибок, определенным в подразделе 5.2 The OAuth 2.0 Authorization Framework (*[RFC6749]*), с применением следующих дополнительных кодов ошибок:

- «authorization_pending»: запрос авторизации ожидает рассмотрения, поскольку конечный пользователь еще не прошел аутентификацию;
- «slow_down»: (аналогично «authorization_pending») запрос авторизации находится на рассмотрении, и опрос должен продолжаться, но интервал должен быть увеличен как минимум на пять секунд для данного и всех последующих запросов;
- «expired_token»: срок действия <auth_req_id> истек. Клиент должен сделать новый запрос аутентификации;
- «access_denied»: пользователь отклонил запрос авторизации.

Примечание. Перечень стандартных кодов ошибок запроса авторизации определен в разделе 5.2 *[RFC6749]*.

При определении кода ошибок необходимо учитывать следующие требования:

1. В случае если <auth_req_id> недействителен или был выдан другому клиенту, сервер авторизации возвращает код ошибки «invalid_grant».
2. В случае если клиент опрашивает быстрее, чем в рамках установленного интервала, сервер авторизации возвращает код ошибки «invalid_request».
3. В случае если клиент получает код ошибки «invalid_request», он не должен делать дальнейшие запросы для того же значения <auth_req_id>.
4. В случае если клиент зарегистрирован для использования режима Push, но вызывает конечную точку токена с типом гранта «urn: openid: params: grant-type: ciba», сервер авторизации возвращает код ошибки «unauthorized_client».
5. Когда клиент получает код ответа «4xx» с полезной нагрузкой JSON, он должен проверить полезную нагрузку, чтобы определить код ошибки.

6.7. ПОЛЕЗНАЯ НАГРУЗКА ОШИБКИ PUSH

При использовании режима Push сообщения об ошибках в ответах от конечной точки уведомления клиента отправляются с использованием полезной нагрузки в формате «application/json» и следующих параметров:

- <error_description>: (опциональный) читаемый человеком текст ASCII [USASCII], предоставляющий дополнительную информацию, используемый для помощи разработчику клиента в понимании возникшей ошибки. Значения параметра <error_description> не должны включать символы вне набора «% x20-21/% x23-5B/% x5D-7E;».
- <error>: (обязательный) ASCII код ошибки.
- <auth_req_id>: (опциональный) идентификатор запроса аутентификации.

При этом в качестве кода ошибки <error> используются следующие значения:

– «access_denied»: конечный пользователь отклонил запрос авторизации.
– «expired_token»: срок токена доступа истек.

Примечание. Поставщик OpenID может не отправлять эту ошибку, но клиент должен поддерживать ее обработку.

– «transaction_failed»: сервер авторизации столкнулся с неопределенной ошибкой, которая не позволила ему успешно завершить транзакцию. Этот код ошибки может использоваться для информирования клиента о том, что транзакция была неудачной по причинам, отличным от тех, которые явно определены кодами ошибок «access_denied» и «expired_token».

6.8. ОТВЕТ ОБ ОШИБКЕ АУТЕНТИФИКАЦИИ

Ответ об ошибке аутентификации возвращается непосредственно из конечной точки аутентификации в ответ на отправленный клиентом запрос аутентификации с использованием полезной нагрузки в формате «application/json» и нижеприведенных параметров:

– <error>: (обязательный) ASCII код ошибки.
– <error_description>: (опциональный) читаемый человеком текст ASCII [USASCII], предоставляющий дополнительную информацию, используемый для помощи разработчику клиента в понимании возникшей ошибки. Значения параметра <error_description> не должны включать символы вне набора «% x20-21/% x23-5B/% x5D-7E».
– <error_uri>: (опциональный) URI, идентифицирующий удобочитаемую веб-страницу с информацией об ошибке, которая используется для предоставления разработчику клиента дополнительной информации об ошибке. Значения параметра <error_uri> должны соответствовать синтаксису URI-ссылки и не должны включать символы вне набора «% x21/% x23-5B/% x5D-7E».

6.8.1. Коды ошибок аутентификации, связанные с ошибками HTTP

6.8.1.1. HTTP 400 Bad Request:

– «invalid_request»: отсутствует обязательный параметр, содержит недопустимое значение параметра, содержит параметр более одного раза, содержит более одного параметра с информацией о пользователе или имеет иные неправильные значения;
– «invalid_scope»: запрошенная область действия (scope) недействительна, не определена или имеет неправильный формат;
– «expired_login_hint_token»: указанный в запросе аутентификации <login_hint_token> недействителен, поскольку срок его действия истек;
– «unknown_user_id»: сервер авторизации не может определить, какого конечного пользователя клиент аутентифицирует с помощью указанной в запросе информации о пользователе (<login_hint_token>, <id_token_hint> или <login_hint>);
– «unauthorized_client»: клиент не авторизован для использования потока аутентификации;
– «missing_user_code»: код пользователя требуется, но отсутствует в запросе;
– «invalid_user_code»: неверный код пользователя;
– «invalid_binding_message»: связанное сообщение ошибочно или неприемлемо для использования в контексте данного запроса.

6.8.1.2. HTTP 401 Unauthorized:

– «invalid_client»: сбой аутентификации клиента (например, неверные учетные данные клиента, неизвестный клиент, аутентификация клиента не включена или неподдерживаемый метод аутентификации).

6.8.1.3. HTTP 403 Forbidden:

– «access_denied»: владелец ресурса или сервер авторизации отклонили запрос.

Примечание. Если ответ об ошибке аутентификации получен до взаимодействия с пользователем, то такая ошибка может быть получена только в том случае, если владелец ресурса или сервер авторизации принял решение отклонить определенный тип запроса или запросов от определенного типа клиентов.

Пример ответа об ошибке аутентификации

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
{
 «error»: «unauthorized_client»,
 «error_description»: «The client ‘client.example.ru’ is not allowed to use CIBA.»
}
```

7. ПРОФИЛЬ БЕЗОПАСНОСТИ OPENID API С ИСПОЛЬЗОВАНИЕМ ПОТОКА АУТЕНТИФИКАЦИИ ПО ОТДЕЛЬНОМУ КАНАЛУ ДЛЯ ДОСТУПА К СЕРВИСАМ В РЕЖИМЕ ЧТЕНИЯ И ЗАПИСИ

7.1. ОБЩИЕ ТРЕБОВАНИЯ

Профиль безопасности OpenID API с использованием потока аутентификации по отдельному каналу применяется как к API только для чтения, так и к API для чтения и записи.

При реализации потока аутентификации по отдельному каналу необходимо руководствоваться общими требованиями к профилям безопасности OpenID API (разделы 6 – 8 ФАПИ.СЕК), а также применять дополнительные требования, изложенные в данном разделе.

7.2. СЕРВЕР АВТОРИЗАЦИИ

Сервер авторизации должен поддерживать положения, указанные в пункте 7.2.2 ФАПИ.СЕК.

Кроме того, для всех операций сервер авторизации:

1. должен поддерживать только конфиденциальных клиентов для инициированных клиентом потоков аутентификации по отдельному каналу;
2. должен обеспечить наличие однозначного определения требуемой для авторизации информации в запросе авторизации или требовать наличия <binding_message> в запросе аутентификации;
3. должен не поддерживать режим Push;
4. должен поддерживать режим Pool;
5. может поддерживать режим Ping;
6. должен требовать подписания запросов к конечной точке аутентификации по отдельному каналу (подпункт 6.3.1.1);
7. должен требовать уровень аутентификации пользователя, соответствующий требованиям операций, которые клиент будет уполномочен выполнять от имени пользователя;
8. если он поддерживает класс контекста аутентификации, указанный через заявленное свойство <acr> в запросе клиента, должен вернуть указанное значение <acr> в запрашиваемый ID токен;
9. должен требовать, чтобы подписанный запрос аутентификации содержал утверждения заявленных свойств <nbf> и <exp>, которые ограничивают срок действия запроса не более чем 60 минутами;
10. может требовать от клиентов предоставить заявленное свойство <request_context> (подраздел 7.4);
11. должен не использовать <login_hint> или <login_hint_token> для передачи идентификаторов намерений или любых других метаданных авторизации.

Примечания.

Согласно данной спецификации <login_hint>, <login_hint_token> и <id_token_hint> используются только для определения конечного пользователя.

Профиль поддерживает только режимы Ping и Pool, поэтому получить токены доступа и при необходимости токены обновления можно только из конечной точки токена. В связи с этим применяются те же требования безопасности, которые определены ФАПИ.СЕК для конечной точки токена.

7.3. КОНФИДЕНЦИАЛЬНЫЙ КЛИЕНТ

7.3.1. Основные положения

Конфиденциальный клиент должен поддерживать положения, указанные в пункте 7.2.3 ФАПИ.СЕК.

Кроме того, конфиденциальный клиент:

1. Должен отправлять в конечную точку аутентификации отдельного канала только подписанные запросы аутентификации.
2. Должен обеспечить наличие информации для определения требуемого уровня авторизации в запросе авторизации или должен включать в запрос аутентификации <binding_message>.
3. Должен убедиться, что сервер авторизации аутентифицировал пользователя, используя соответствующие предполагаемой цели клиента методы авторизации. Способы извещения клиента о требуемых методах авторизации определяются на этапе разработки сервера авторизации исходя из его прикладных целей и задач.

7.4. РАСШИРЕНИЯ ДЛЯ ЗАПРОСА АУТЕНТИФИКАЦИИ

Профиль определяет следующее расширения запроса аутентификации (пункт 6.3.1):

– <request_context>: (опциональный) JSON-объект, содержащий данные для информирования, с целью анализа попыток мошенничества и угроз (серийный номер устройства, версия операционной системы, данные геолокации). Механизм формирования данного параметра определяется на этапе разработки сервера авторизации исходя из его прикладных целей и задач.

7.5. ДОСТУП К ЗАЩИЩЕННЫМ РЕСУРСАМ

В профиле применяются положения касательно доступа клиента к защищенным ресурсам, представленные в разделах 6 и 7 ФАПИ.СЕК, с учетом того, что выданные в потоке аутентификации по отдельному каналу токены используются так же, как и токены, выпущенные через потоки кода авторизации.

7.5.1. Положения клиента

Клиент, реализующий требования данного стандарта, должен поддерживать положения, указанные в пункте 6.3.3 ФАПИ.СЕК.

Кроме того, в ситуациях, когда клиент не управляет CD, клиент:

- не должен передавать <x-fapi-customer-ip-address> или <x-fapi-auth-date> в заголовке HTTP;
- должен передавать метаданные о контексте сессии CD с конечным пользователем (тип устройства, данные геолокации).

7.5.2. Механизмы защиты

В профиле применяются механизмы защиты, указанные в подразделе 5.8 ФАПИ.СЕК, а также следующие дополнительные меры:

1. Сервер авторизации должен гарантировать, что установленная во время динамической регистрации клиента (подпункт 5.4.4.5 ФАПИ.СЕК) параметром <backchannel_client_notification_endpoint> конечная точка уведомления клиента находится в окружении администрирования клиента.
2. <login_hint_token> должен быть подписан цифровой подписью, что обеспечивает подлинность данных и снижает возможность атаки с помощью внедрения данных. Подпись позволяет серверу авторизации аутентифицировать и авторизовать отправителя информации о пользователе и предотвратить сбор пользовательских идентификаторов злоумышленниками, выступающими в роли клиента.

7.6. ПОДТВЕРЖДЕНИЕ ПРОЦЕССА АУТЕНТИФИКАЦИИ

7.6.1. Инициация сессий аутентификации без участия конечного пользователя

В потоке аутентификации по отдельному каналу сервер авторизации должен иметь информацию о согласии конечного пользователя с процессом аутентификации. Для этого клиент должен передать идентификатор пользователя серверу авторизации. Если этот идентификатор в запросе аутентификации в качестве <login_hint> является статичным (например, номер телефона), то злоумышленник, выступающий в роли клиента, зная статичный идентификатор пользователя <login_hint>, может указывать его в запросах серверу авторизации и посылать такие запросы на AD пользователей. В качестве меры защиты необходимо дополнять запрос на аутентификацию пользователя некоторым идентификатором, служащим подтверждением согласия пользователя на аутентификацию. Такой идентификатор может быть получен одним из следующих способов:

1. **<login_hint> с заявленным свойством <nonce>.** Заявленное свойство <nonce> генерируется на устройстве AD, после чего попадает на сервер авторизации, с которым взаимодействует AD (например, фронт сервера авторизации), и затем этот <nonce> передается клиенту для включения в <login_hint>.
2. **Одноразовый идентификатор конечного пользователя, перенесенный с AD на CD.** Сервер авторизации может генерировать одноразовый идентификатор, который передается AD на CD в начале потока. При этом конечный пользователь аутентифицируется в AD и запрашивает идентификатор для потока в виде QR-кода, сканирует его на CD, после чего он передается с CD на клиента, где декодируется и используется для значения <login_hint_token> и проверяется сервером авторизации при инициировании потока аутентификации по отдельному каналу.
3. **Одноразовый идентификатор конечного пользователя, перенесенный с CD на AD.** Клиент может генерировать одноразовый идентификатор, который передается CD на AD в начале потока. При этом конечный пользователь аутентифицируется в AD и сканирует на AD сгенерированный CD идентификатор для потока в виде QR-кода, после чего он передается с AD на сервер авторизации. Декодированный идентификатор передается клиентом на сервер авторизации в качестве значения <login_hint_token> и проверяется сервером авторизации.

К одноразовым идентификаторам конечного пользователя должны применятся требования, аналогичные заявленному свойству <nonce>.

7.6.2. Подтверждение пользователем значения <binding_message>

В зависимости от информации о пользователе, используемой для его идентификации и процессов аутентификации пользователя клиентом, мошенник может осуществить запуск атаки подмены потока аутентификации, запустив собственный поток аутентификации на AD одновременно с подлинным потоком, причем оба потока будут использовать актуальный идентификатор пользователя. Если область запраши-

ваемого доступа аналогична, то, чтобы убедиться, что пользователь авторизует правильную транзакцию, необходимо, чтобы он сравнил значение <binding_message> на AD и CD или использовал альтернативные механизмы проверки <binding_message> (например, передавая его на устройство аутентификации через QR-код) либо временные идентификаторы пользователя, сгенерированные на AD (разделы 1, 2, пункт 7.6.1).

БИБЛИОГРАФИЯ

Название	Описание
[ГОСТ Р 34.10 – 2012]	Национальный стандарт Российской Федерации ГОСТ Р 34.10 – 2012 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи»
[Р 50.1.041 – 2002]	Рекомендации по стандартизации Р 50.1.041 – 2002 «Информационные технологии. Руководство по проектированию профилей среды открытой системы (СОС) организации-пользователя»
[ГОСТ Р 58833 – 2020]	Национальный стандарт Российской Федерации ГОСТ Р 58833 – 2020 «Защита информации. Идентификация и аутентификация. Общие положения»
[ГОСТ Р 57580.1 – 2017]	Национальный стандарт Российской Федерации ГОСТ Р 57580.1 – 2017 «Безопасность финансовых (банковских) операций. Защита информации финансовых организаций. Базовый состав организационных и технических мер»
[ГОСТ Р 56939]	Национальный стандарт Российской Федерации ГОСТ Р 56939 – 2016 «Защита информации. Разработка безопасного программного обеспечения. Общие требования»
ФАПИ.СЕК	Стандарт Банка России СТО БР ФАПИ.СЕК-1.6 – 2020 «Безопасность финансовых (банковских) операций. Прикладные программные интерфейсы обеспечения безопасности финансовых сервисов на основе протокола OpenID»
[ietf-oauth-mtls]	«*OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens*», RFC8705, февраль 2020
[OpenID. Core]	«*OpenID Connect Core 1.0*», ноябрь 2014
[OpenID. Discovery]	«*OpenID Connect Discovery 1.0*а», ноябрь 2014
[OpenID. DCR]	«*OpenID Connect Dynamic Client Registration 1.0*», ноябрь 2014
[OpenID.CIBA.Core]	*OpenID Connect Client Initiated Backchannel Authentication Flow – Core 1.0 draft-03*, январь 2020
[OpenID.CIBA.RW]	*Financial-grade API: Client Initiated Backchannel Authentication Profile*, август 2019
[RFC6750]	«*The OAuth 2.0 Authorization Framework: Bearer Token Usage*», RFC 6750, октябрь 2012
[RFC7519]	«*JSON Web Token (JWT)*», RFC 7519, май 2015
[IANA. JWT]	*IANA*, «*JSON Web Token (JWT)*», март 2020
[IANA.OAuth.Parameters]	*IANA*, «*OAuth Parameters*», март 2020
[RFC6202]	«*Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP*», RFC 6202, апрель 2011
[RFC6749]	«*The OAuth 2.0 Authorization Framework*», RFC 6749, октябрь 2012
[RFC7231]	«*Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*», RFC 7231, июнь 2014
[RFC7521]	«*Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants*»", RFC 7521, май 2015
[RFC7591]	«*OAuth 2.0 Dynamic Client Registration Protocol*», RFC 7591, июль 2015
[RFC8414]	«*OAuth 2.0 Authorization Server Metadata*», RFC 8414, июнь 2018
[RFC8628]	«*OAuth 2.0 Device Authorization Grant*», RFC 8628, август 2019
[RFC8705]	«*OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens*», RFC8705, февраль 2020
[RFC4648]	«*The Base16, Base32, and Base64 Data Encodings*», RFC4648, октябрь 2006
[№152-ФЗ]	Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных»
[№1119]	Постановление Правительства Российской Федерации от 01.11.2012 № 1119 «Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных»
[№378]	Приказ ФСБ России от 10.07.2014 № 378 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных с использованием средств криптографической защиты информации, необходимых для выполнения установленных Правительством Российской Федерации требований к защите персональных данных для каждого из уровней защищенности»
[№21]	Приказ ФСТЭК России от 18.02.2013 № 21 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных»
[№ 382-П]	Положение Банка России от 09.06.2012 № 382-П «О требованиях к обеспечению защиты информации при осуществлении переводов денежных средств и о порядке осуществления Банком России контроля за соблюдением требований к обеспечению защиты информации при осуществлении переводов денежных средств»
[№ 683-П]	Положение Банка России от 17.04.2019 № 683-П «Об установлении обязательных для кредитных организаций требований к обеспечению защиты информации при осуществлении банковской деятельности в целях противодействия осуществлению переводов денежных средств без согласия клиента»
[№ 757-П]	Положение Банка России от 20.04.2021 № 757-П «Об установлении обязательных для некредитных финансовых организаций требований к обеспечению защиты информации при осуществлении деятельности в сфере финансовых рынков в целях противодействия осуществлению незаконных финансовых операций»