Банк России

СТАНДАРТ БАНКА РОССИИ

СТО БР ФАПИ.ПАОК-1.0-2024

Безопасность финансовых (банковских) операций

Прикладные программные интерфейсы. Обеспечение безопасности финансовых сервисов при инициации OpenID Connect клиентом потока аутентификации по отдельному каналу

Требования

Москва
2024

ПРЕДИСЛОВИЕ

1. Разработан Ассоциацией развития финансовых технологий (Ассоциация ФинТех) и акционерным обществом «Информационные технологии и коммуникационные системы» (АО «ИнфоТеКС») при участии Центрального банка Российской Федерации (Банка России).

2. Принят и введен в действие приказом Банка России от 07.10.2024 № ОД-1616.

3. Взамен СТО БР ФАПИ.ПАОК-1.0-2021.

Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29.06.2015 № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту, пересмотре (замене) или отмене стандарта размещается на сайте Банка России в сети Интернет (http://www.cbr.ru/).

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящий стандарт разработан для применения кредитными организациями, некредитными финансовыми организациями, субъектами национальной платежной системы, лицами, оказывающими профессиональные услуги на финансовом рынке (далее – организации финансового рынка), прикладных программных интерфейсов (application programming interface, API) и основан на спецификациях технологии OpenID Connect Core (OIDC), которые определяют порядок использования модели API со структурированными данными и модели токена для повышения безопасности финансовых технологий в случае инициирования OpenID Connect клиентом потока аутентификации по отдельному каналу.

Настоящий стандарт устанавливает требования к информационной безопасности при реализации взаимодействия с использованием API, обеспечивающие необходимый уровень защищенности информации при передаче персональных данных, банковской тайны и иной защищаемой информации в целях реализации требований Банка России на технологическом участке взаимодействия в среде Открытых банковских интерфейсов для осуществления требуемого уровня доверия к идентификации, аутентификации и авторизации сторонних поставщиков финансовых услуг.

1. 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 требованиям настоящего стандарта программных средств, предназначенных для безопасного обмена финансовыми сообщениями в среде Открытых банковских интерфейсов в соответствии со стандартами Банка России:

– «Открытые банковские интерфейсы. Общие положения» [1];

– «Открытые банковские интерфейсы. Получение публичной информации о кредитной организации и ее продуктах» [2];

– «Открытые банковские интерфейсы. Инициирование перевода денежных средств клиента третьей стороной в валюте Российской Федерации» [3];

– «Открытые банковские интерфейсы. Получение информации о счете клиента третьей стороной» [4].

Положения настоящего стандарта носят рекомендательный характер, если только обязательность применения отдельных из них не установлена нормативными правовыми актами, в том числе нормативными актами Банка России. Настоящий стандарт может быть использован для включения ссылок на него и/или прямого включения содержащихся в нем положений во внутренние документы организаций финансового рынка, а также в договоры, заключенные между организациями.

Положения настоящего стандарта применяются совместно со следующими документами:

– 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 Connect. Требования» [5];

– методические рекомендации МР. 26.2.002-2024 Технического комитета ТК 26 «Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколах OpenID Connect» [6].

2. НОРМАТИВНЫЕ ССЫЛКИ

В настоящем стандарте использованы ссылки на следующие документы:

– ГОСТ Р 34.10-2012 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи» [30];

– ГОСТ Р 56939-2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования»;

– ГОСТ Р 57580.1-2017 «Безопасность финансовых (банковских) операций. Защита информации финансовых организаций. Базовый состав организационных и технических мер»;

– ГОСТ Р 58833-2020 «Защита информации. Идентификация и аутентификация. Общие положения» [29];

– Р 50.1.041-2002 «Информационные технологии. Руководство по проектированию профилей среды открытой системы (СОС) организации-пользователя»;

– Р 50.1.053-2005 «Информационные технологии. Основные термины и определения в области технической защиты информации».

Примечание. При пользовании настоящим стандартом целесообразно проверить действие ссылочных документов в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год. Если заменен ссылочный документ, на который дана недатированная ссылка, рекомендуется использовать действующую версию этого документа с учетом всех внесенных в данную версию изменений. Если заменен ссылочный документ, на который дана датированная ссылка, рекомендуется использовать версию этого документа с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный документ, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, это положение рекомендуется применять без учета данного изменения. Если ссылочный документ отменен без замены, положение, в котором дана ссылка на него, применяется в части, не затрагивающей эту ссылку.

3. ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

3.1.

Авторизация (authorization): проверка, подтверждение и предоставление прав логического доступа при осуществлении субъектами доступа логического доступа.

[ГОСТ Р 57580.1-2017, пункт 3.15]

3.2.

Аутентификация (authentication): 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 58833-2020, пункт 3.4]

3.3.

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

3.4.

Владелец ресурса (resource owner): субъект, способный предоставить доступ к защищенному ресурсу. [12]

3.5.

Доступ (access): получение одной стороной информационного взаимодействия возможности использования ресурсов другой стороны информационного взаимодействия.

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.

[ГОСТ Р 58833-2020, пункт 3.17]

3.6.

Защищенный ресурс (protected resource): ресурс с ограниченным доступом. [12]

3.7.

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, подпункт 3.3.9]

3.8.

Класс контекста аутентификации (authentication context class): набор методов или процедур аутентификации, которые считаются эквивалентными друг другу в определенном контексте. [7]

3.9.

Клиент (client): приложение, целью которого является получение доступа к защищенным ресурсам пользователя от его имени после выполнения процедуры авторизации. [12]

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

3.10.

Конечная точка (endpoint): адрес (как правило, URL) сетевого ресурса, через который осуществляется доступ к определенному сервису. [12]

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

3.11.

Конечная точка авторизации (authorization endpoint): конечная точка, используемая клиентом для получения авторизации от владельца ресурса посредством перенаправления агента пользователя. [12]

3.12.

Конечная точка аутентификации по отдельному каналу (backchannel authentication endpoint): конечная точка, используемая клиентом для инициирования аутентификации конечного пользователя по отдельному каналу. [8]

3.13.

Конечная точка токена (token endpoint): конечная точка, используемая клиентом для обмена разрешения на доступ на токен доступа, обычно с аутентификацией клиента. [12]

3.14.

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. [8]

3.15.

Конечный пользователь (end user): владелец ресурса в случае, если он является человеком. [12]

3.16.

Контекст аутентификации (authentication contex): информация, которая может потребоваться проверяющей стороне, прежде чем она примет решение о предоставлении права на ответ об аутентификации. [7]

3.17.

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 выполнить безопасную аутентификацию клиента с использованием других средств. [12]

3.18.

Объект запроса (request object): токен JWT, который содержит набор параметров запроса аутентификации в качестве своих заявленных свойств. [5]

3.19.

Односторонняя аутентификация (one-way authentication): 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]

3.20.

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). [31]

3.21.

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, подпункт 3.1.5]

3.22.

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. [7]

3.23.

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. [12]

3.24.

Разрешение на доступ (authorization grant): свидетельство, подтверждающее авторизацию владельца ресурса (для доступа к его защищенным ресурсам), которое далее используется клиентом для получения токена доступа. [12]

3.25.

Сервер авторизации (authorization server): сервер, выдающий клиенту токены доступа после успешной аутентификации владельца ресурса и прохождения процедуры авторизации. [12]

3.26.

Сервер ресурсов (resource server): сервер, на котором размещены защищенные ресурсы, способный принимать и отвечать на запросы к защищенным ресурсам с использованием токенов доступа. [12]

3.27.

Токен доступа (access token): свидетельство, подтверждающее факт авторизации доступа к определенному ресурсу, выданное определенному клиенту сервером авторизации с одобрения владельца ресурса. Токен доступа указывает на конкретные области данных, к которым разрешен доступ, длительность доступа и другие параметры. [12]

3.28.

ID токен, токен идентификации (ID Token): JSON веб-токен, который содержит параметры аутентификации конечного пользователя сервером авторизации. Может содержать также другие параметры. [31]

3.29.

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. [13]

3.30.

Токен обновления (refresh token): символьная строка, используемая 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). [12]

3.31.

Хэш-код (hash-code): строка бит, являющаяся выходным результатом хэш-функции.

[ГОСТ Р 34.10-2012, пункт 3.1.13]

3.32.

Хэш-функция (collision-resistant hash-function): функция, отображающая строки бит в строки 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 34.10-2012, подпункт 3.1.14]

3.33.

[Электронная цифровая] подпись (signature): строка бит, полученная в результате процесса формирования подписи.

[ГОСТ Р 34.10-2012, подпункт 3.1.15]

3.34.

Base64-кодирование (base64-encode): 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 2 дополнительных символа, зависящих от системы реализации. [32]

3.35.

Base64url-кодирование (base64url-encode): Base64-кодирование строки байт с использованием набора символов, допускающих использование результата кодирования в качестве имени файла или URL. [32]

3.36.

JSON веб-токен (JWT): строка, представляющая набор параметров в формате объекта JSON, который закодирован в виде структуры JWS или JWE. При этом параметры объекта JSON сопровождаются цифровой подписью, кодом аутентификации и/или шифрованием. [31]

4. ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

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

API (Application Program Interface) – прикладной программный интерфейс

ASCII (American Standard Code for Information Interchange) – стандарт США 8-битного кодирования некоторого набора печатных и непечатных символов

ASCII (STRING) – байт ASCII представления последовательности ASCII символов STRING

BASE64URL (OCTETS) – Base64url-кодирование строки байт

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

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

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

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

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

5. ОБЩИЕ ПОЛОЖЕНИЯ

5.1. Структура стандарта

В настоящем стандарте представлены дополнительные требования и рекомендации для обеспечения безопасного доступа к данным в финансовых сервисах реального времени с использованием модели обмена данными REST/JSON, защищенной технологией авторизации OAuth 2.0, включая профилирующий ее протокол OpenID Connect при инициировании клиентом потока аутентификации по отдельному каналу.

Стандарт состоит из следующих частей:

– реализация OIDC при аутентификации по отдельному каналу;

– профиль безопасности OpenID API с использованием потока аутентификации по отдельному каналу.

В подразделе 5.2 приведены нормативные требования.

Первая часть настоящего стандарта (раздел 6) основана на документе [8], 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 [9] и динамической регистрации клиентов (подраздел 6.2 [10]), параметры конечных точек аутентификации и уведомления клиента (подразделы 6.3 и 6.5), описываются процесс получения результата аутентификации (подраздел 6.6), а также сведения о применяемых кодах ошибок.

Вторая часть (раздел 7) основана на документе [11] и представляет дополнительные параметры профиля безопасности API для доступа к сервисам финансовых данных (описание профиля представлено в разделах 6 и 7 [5]).

5.2. Нормативные требования

5.2.1. При реализации указанных в настоящем стандарте криптографических алгоритмов (механизмов) на территории Российской Федерации 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 (модели угроз информационной безопасности), к компонентам которой применяются эти требования, в соответствии с нормативными правовыми актами Российской Федерации.

5.2.2. Вовлеченные стороны должны следовать требованиям обеспечения безопасности персональных данных, определенным нормативными правовыми актами Российской Федерации, в частности:

– Федеральным законом от 27.07.2006 № 152-ФЗ «О персональных данных» [16];

– постановлением Правительства Российской Федерации от 01.11.2012 № 1119 «Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных» [17];

– 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» [18];

– 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» [19].

5.2.3. Прикладное программное обеспечение, реализующее требования настоящего стандарта, должно отвечать требованиям положений Банка России от 17.08.2023 № 821-П [20], от 17.04.2019 № 683-П [21], от 20.04.2021 № 757-П [22], от 25.07.2022 № 802-П [23], от 17.10.2022 № 808-П [33] в части прохождения сертификации в системе сертификации федерального органа исполнительной власти, уполномоченного в области противодействия техническим разведкам и технической защиты информации, или оценки соответствия по требованиям к оценочному уровню доверия (ОУД) не ниже чем ОУД 4, в соответствии с пунктом 7.6 национального стандарта Российской Федерации ГОСТ Р ИСО/МЭК 15 408-3-2013 [24].

Разработка программного обеспечения, реализующего требования настоящего стандарта, должна вестись с учетом требований к разработке безопасного программного обеспечения, установленных следующими документами:

– ГОСТ Р 56939-2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования» или раздел 7.4 методического документа «Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций» [25];

– методический документ ФСТЭК России «Руководство по организации процесса управления уязвимостями в органе (организации)» [26];

– методический документ ФСТЭК России «Методика тестирования обновлений безопасности программных, программно-аппаратных средств» [27];

– методический документ ФСТЭК России «Методика оценки уровня критичности уязвимостей программных, программно-аппаратных средств» [28].

Должен проводиться в том числе регулярный поиск информации, связанной с уязвимостями программ, в общедоступных источниках, включая использование банка данных угроз безопасности информации ФСТЭК России.

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 (раздел 3 [8]):

1. Клиент должен выполнить запрос HTTP POST к конечной точке аутентификации по отдельному каналу, чтобы запросить аутентификацию конечного пользователя по отдельному каналу.

2. Сервер передает уникальный идентификатор (назначенный к конкретной аутентификации пользователя) клиенту, чтобы клиент мог идентифицировать операцию аутентификации пользователя.

3. Клиент получает ID токен, токен доступа и при необходимости токен обновления с помощью режимов Poll (пункт 6.1.1), Ping (пункт 6.1.2) или Push (пункт 6.1.3). Режим должен быть определен при регистрации клиента (пункт 6.2.2).

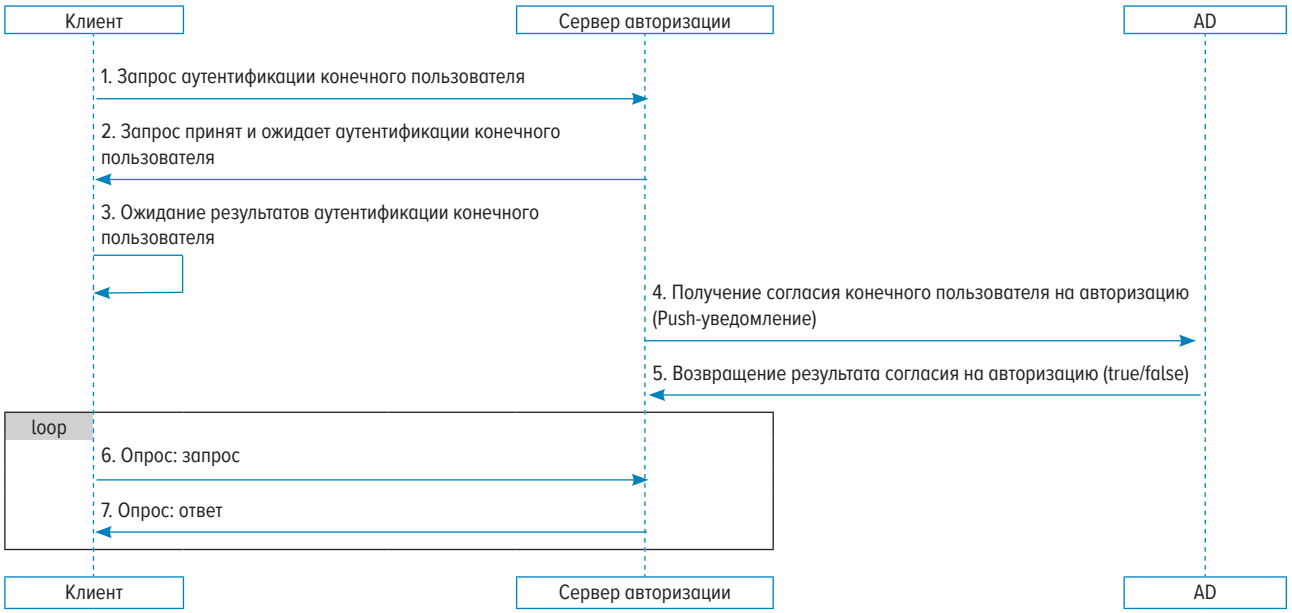
6.1. Режимы Poll, Ping и Push

В настоящем подразделе определяются режимы получения клиентом результата аутентификации (разделы 3 и 5 [8]).

6.1.1. Режим Poll

Режим Poll – механизм аутентификации по отдельному каналу, при котором клиент для получения токенов опрашивает конечную точку токена.

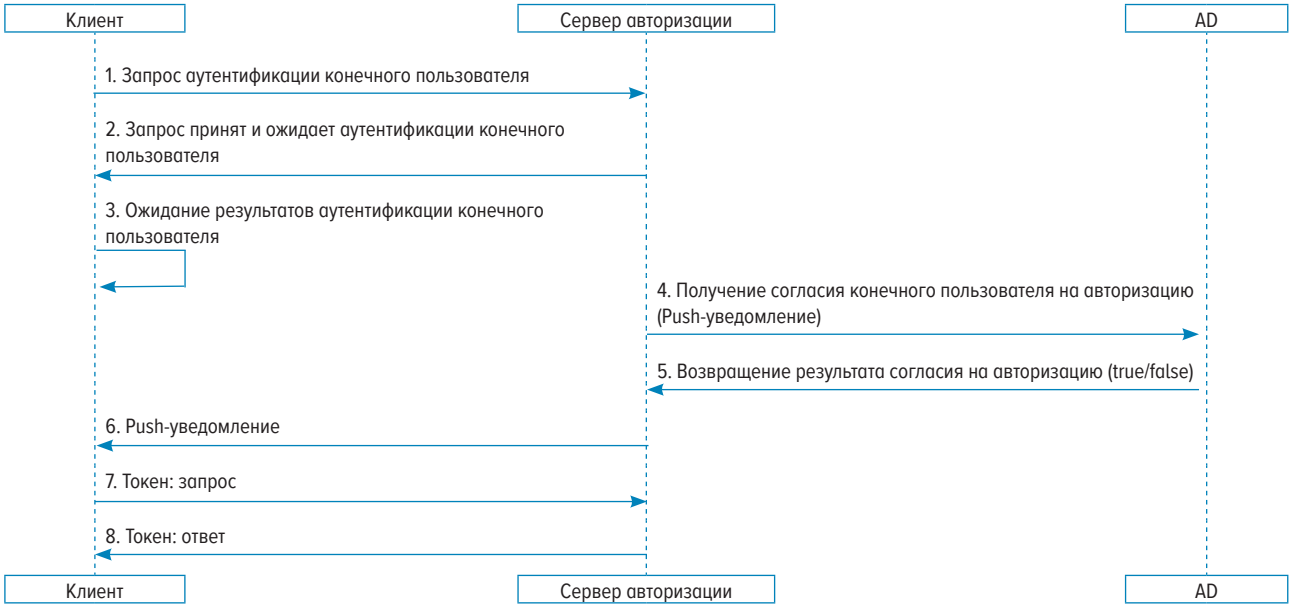
СЦЕНАРИЙ АУТЕНТИФИКАЦИИ ПО ОТДЕЛЬНОМУ КАНАЛУ В РЕЖИМЕ POLL *Рис. 1*



6.1.2. Режим Ping

Режим Ping – механизм аутентификации по отдельному каналу, при котором сервер авторизации возвращает на URI конечной точки уведомления (предварительно зарегистрированной клиентом) уникальный идентификатор, полученный на конечной точке аутентификации по отдельному каналу. После получения уведомления клиент делает запрос на конечную точку токена для получения токенов.

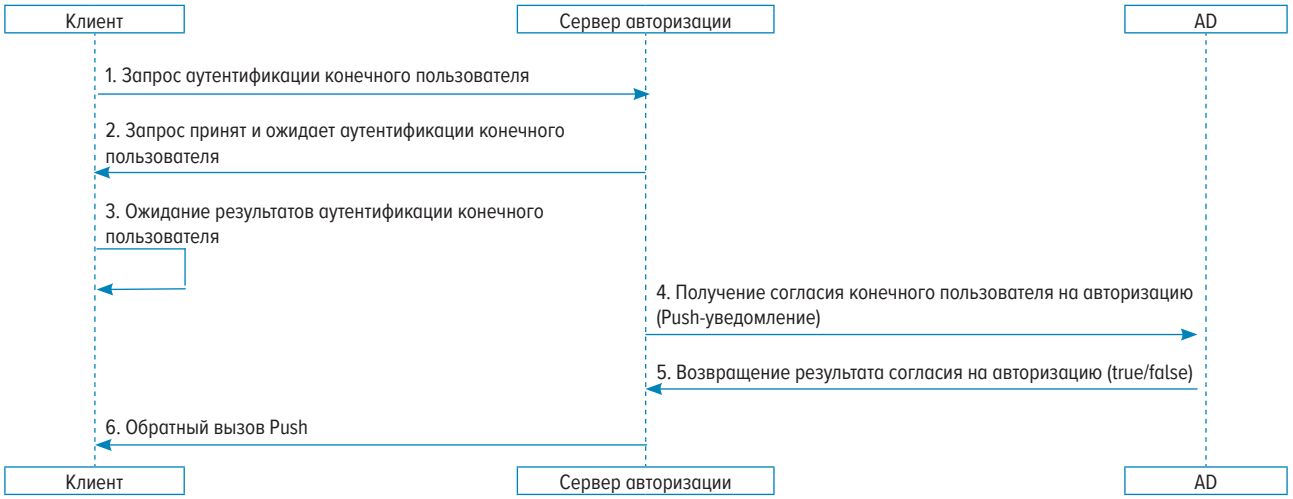
СЦЕНАРИЙ АУТЕНТИФИКАЦИИ ПО ОТДЕЛЬНОМУ КАНАЛУ В РЕЖИМЕ PING *Рис. 2*



6.1.3. Режим Push

Режим Push – механизм аутентификации по отдельному каналу, при котором сервер авторизации отправляет токены на зарегистрированный клиентом URI.

СЦЕНАРИЙ АУТЕНТИФИКАЦИИ ПО ОТДЕЛЬНОМУ КАНАЛУ В РЕЖИМЕ PUSH *Рис. 3*



6.2. Регистрация и обнаружение метаданных

В настоящем подразделе определяются параметры и значения объектов процесса обнаружения метаданных [9] и динамической регистрации клиентов [10] (подраздел 5.3 [6]), относящиеся к определению поддерживаемых режимов аутентификации:

– метаданные сервера авторизации (подраздел 5.4 [6]),

– метаданные клиента (подраздел 5.5 [6]).

6.2.1. Метаданные сервера авторизации

Для целей этого документа используются следующие метаданные сервера авторизации (раздел 4 [8]):

– `<grant_types_supported>`: (опциональный) перечень поддерживаемых сервером авторизации типов разрешений на доступ (grants); для обеспечения аутентификации конечного пользователя по отдельному каналу определяется дополнительный тип разрешения на доступ `"urn: openid: params: grant-type: ciba"`. Данный тип разрешения на доступ должен использоваться в метаданных сервера авторизации, поддерживающего режимы Ping или Poll;

Примечание. Рекомендации по расширению типов разрешений на доступ определены в подразделе 4.5 [12].

– `<token_endpoint_auth_methods_supported>`: (опциональный) перечень поддерживаемых сервером авторизации методов аутентификации клиента;

– `<token_endpoint_auth_signing_alg_values_supported>`: (опциональный) перечень алгоритмов подписи JWS конечной точки аутентификации по обратному каналу сервера;

– `<backchannel_token_delivery_modes_supported>`: (обязательный) JSON-массив с перечнем поддерживаемых режимов; возможные варианты: `"poll", "ping", "push"`;

– `<backchannel_authentication_endpoint>`: (обязательный) URL-адрес конечной точки аутентификации по отдельному каналу сервера авторизации;

– `<backchannel_authentication_request_signing_alg_values_supported>`: (опциональный) JSON-массив с перечнем идентификаторов алгоритмов подписи структуры JWS (параметр <alg>), которые поддерживает сервер авторизации для обработки подписанных запросов аутентификации (подпункт 6.3.1.1). Сервера авторизации, соответствующие настоящему стандарту, должны поддерживать значения, приведенные в подразделе 9.1 [6]. Если параметр отсутствует, подписанные запросы аутентификации не поддерживаются сервером авторизации;`<backchannel_user_code_parameter_supported>`: (опциональный) логическое значение, указывающее, поддерживает ли сервер авторизации параметр `<user_code>` (значение `"true"` означает поддержку). Если параметр отсутствует, значением по умолчанию является `"false"`.

6.2.2. Метаданные клиента

Для целей данного документа используются следующие метаданные клиента (раздел 4 [8]):

– `<grant_types>`: (опциональный) JSON-массив, содержащий перечень поддерживаемых клиентом типов разрешений на доступ (grants); для обеспечения аутентификации конечного пользователя по отдельному каналу должен принимать значение `"urn: openid: params: grant-type: ciba"` для режимов Ping или Poll;

– <backchannel_token_delivery_mode>: (обязательный) режим доставки токена; может принимать одно из следующих значений: "poll", "ping", "push";

– <backchannel_client_notification_endpoint>: (обязательный) HTTPS URL-адрес конечной точки уведомления клиента для режимов Ping или Push;

– <backchannel_authentication_request_signing_alg>: (опциональный) идентификатор алгоритма подписи структуры JWS (параметр <alg>), который клиент использует в подписанных запросах аутентификации (подпункт 6.3.1.1). Если параметр отсутствует, клиент не отправляет подписанные запросы аутентификации;

– <backchannel_user_code_parameter>: (опциональный) логическое значение, указывающее, поддерживает ли клиент параметр <user_code> (значение "true" означает поддержку). Если параметр отсутствует, значением по умолчанию является "false".

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

Ниже приведен 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 (раздел 7 [8]). Передача сообщений между клиентом и сервером авторизации на конечной точке аутентификации по отдельному каналу должна производиться по TLS-соединению в соответствии с требованиями к протоколу, определенными в пункте 10.1.4 [6].

6.3.1. Запрос аутентификации

Запрос аутентификации по отдельному каналу выполняется напрямую от клиента к серверу авторизации, без прохождения через агента 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 идентификатора (подраздел 7.1 [8]).

Клиент должен пройти аутентификацию на конечной точке аутентификации по отдельному каналу, используя метод аутентификации, зарегистрированный для его <client_id> (метод, определенный в параметре <token_endpoint_auth_method> метаданных клиента). Методы аутентификации клиента регулируются требованиями обеспечения безопасности авторизации, применяемыми к серверу авторизации в зависимости от профиля безопасности OpenID API. При применении базового профиля безопасности допускается использование TLS с взаимной аутентификацией сторон взаимодействия протокола OAuth 2.0: "tls_client_auth", "client_secret_jwt" или "private_key_jwt" (раздел 7 [6]). При применении расширенного профиля безопасности допускается использовать только "tls_client_auth" и "private_key_jwt" (раздел 7 [6]).

При использовании "tls_client_auth" необходимо руководствоваться требованиями, определенными в подразделе 7.4 [6].

При использовании механизмов аутентификации "client_secret_jwt" (подраздел 7.2 [6]) и "private_key_jwt" (подраздел 7.3 [6]) в параметре <client_assertion> в качестве параметра <aud> следует использовать идентификатор эмитента сервера авторизации (параметр метаданных сервера авторизации <issuer> (подраздел 5.4 [6]). 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 запроса; определяет перечень свойств защищаемых данных конечного пользователя, к которым запрошен доступ; параметр <scope> должен содержать значение "openid" (пункт 6.2.1 [6]). 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 подразделе 3.3 [12].

– <client_notification_token>: (обязательный, если клиент использует режим Ping или Push) предоставляемый клиентом токен на предъявителя, который будет использоваться сервером авторизации для аутентификации запроса обратного вызова к клиенту. Длина токена не должна превышать 1024 символа, он должен соответствовать синтаксису токена на предъявителя и содержать достаточную энтропию (не менее 160 бит);

Примечание. Дополнительные сведения приведены в подразделе 2.1 [13].

– <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. Средства аутентификации конечного пользователя имплементируются сервером авторизации, и требуемый класс контекста аутентификации возвращается в качестве параметра <acr> ID токена. При наличии параметра <acr_values> в запросе аутентификации полученный ID токен должен содержать значение параметра <acr>. Значения параметра <acr> определяются участниками взаимодействия,

использующими данный параметр, и должны однозначно определять методы и факторы аутентификации согласно ГОСТ Р 58833-2020 [29];

– <login_hint_token>: (опциональный) токен, содержащий информацию о конечном пользователе, для которого проводится аутентификация. Механизм формирования данного параметра определяется на этапе разработки сервера авторизации, исходя из его прикладных целей и задач;

– <id_token_hint>: (опциональный) информация о конечном пользователе в формате ID токена, выданного клиенту сервером авторизации; возвращается клиентом серверу авторизации для идентификации конечного пользователя;

– <login_hint>: (опциональный) информация о конечном пользователе для сервера авторизации, для которого запрашивается аутентификация. Значение может содержать идентификационные данные конечного пользователя, по которым сервер авторизации может однозначно его идентифицировать (адрес электронной почты, номер телефона, номер учетной записи, идентификатор субъекта и так далее), 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, предназначенное для отображения как на CD, так и на AD, чтобы связать их вместе для транзакции посредством визуальной информации для конечного пользователя. Это сообщение позволяет конечному пользователю убедиться, что действие, предпринятое на AD, связано с запросом, инициированным на CD (например, код подтверждения транзакции). Поскольку различные устройства могут иметь ограниченные возможности отображения, значение <binding_message> сообщения, предназначенного для визуального просмотра конечным пользователем, должно содержать не более 100 символов и использовать алфавитно-цифровые символы A–Я, a–я, A–Z, a–z и 0–9 и символы «_», «!». Если предоставленное значение <binding_message> является недопустимым, клиенту возвращается ошибка "invalid_binding_message";

– <user_code>: (опциональный) секретный код, который известен только пользователю, но может быть проверен сервером авторизации. Код используется для авторизации процесса отправки запроса аутентификации на AD пользователя. Указанный параметр должен присутствовать, если значение параметра метаданных клиента <backchannel_user_code_parameter> имеет значение "true";

– <requested_expiry>: (опциональный) положительное целое число, определяющее время жизни запроса аутентификации; позволяет ограничивать по времени и корректно 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 (и только один) параметр, содержащий информацию о конечном 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 применения механизмов аутентификации "client_secret_jwt" (подраздел 7.2 [6]) и "private_key_jwt" (подраздел 7.3 [6]) используются дополнительные параметры <client_assertion> и <client_assertion_type>.

Ниже приведен 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. eyJzdWJfaWQiOnsic3ViamVj
dF90eXBlIjoicGhvbmUiLCJwaG9uZSI6Iis3MTIzMDAwMDAw MSJ9fQ.A9kJnX3IUAFHsic0nUfXKVNlGDWLuo
2L80y2GLMsmn82xJ8yu6wA5k3pIepOUxrBd 74aD_CVFHgJ3qVibjRBmawThhOQMNrgoqTGfRNUWR8A9ogLogX
HCSZEo7oCI1D4zxHhsDeSu 0Mby61sLmr_uchscsEkSsfnKpmMpyaIn53wWdIu3OI5BVMALm7EMZGuweReM5NT
KfCUNjXNm p5w&
   client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Ajwt-bearer&
   client_assertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.eyJpc3MiOiJzNkJoZFJrcXQzI
iwic3ViIjoiczZCaGRSa3F0MyIsImF1ZCI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUucnUiLCJqdGkiOiJiZGM
tWHNfc2YtM1lNbzRGU3pJSjJRIiwiaWF0IjoxNTkxMDc2NzE2LCJleHAiOjE1OTIwNzY3MTZ9.FuBj8SjN3buf
Qae1HgaO6BoSp4ocrnl4dnTZLJE0KZpnafAA88Ra7eoUT2QP1LhtetVAlDcT3nRJmyyRNLc7oXTYtb5mqCbNYw
4ThTm9hIgyV339hDO837vxX_GNzP6zY5nKA9nyqyRG3BbPldBkzanyQD1F0JtAVugnBZw
```

Декодированный <login_hint_token>:

```
   {
     "sub_id":
     {
       "subject_type": "phone",
       "phone": "+71230000001"
     }
   }
```

Декодированный JWT из параметра <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 параметров подписанного JWT(раздел 8 [6]) с представлением каждого наименования параметра запроса в качестве имени параметра (claim), а его значения в виде строки JSON. Подписанный запрос аутентификации передается в качестве параметра <request> запроса HTTP POST (пункт 7.1.1 [8]).

Примечание. Как исключение, параметр <requested_expiry> может быть представлен также числом JSON.

При этом JWT должен:

- содержать все параметры запроса аутентификации;
- быть подписан цифровой подписью (подраздел 8.1 [6]);
- содержать следующие зарегистрированные параметры (claims):

  а) <aud>: должен содержать значение идентификатора эмитента сервера авторизации, который идентифицирует сервер авторизации как предполагаемую аудиторию,

  б) <iss>: источник, должен содержать идентификатор клиента,

  в) <exp>: время окончания срока действия подписанного запроса аутентификации,

  г) <iat>: время создания подписанного запроса аутентификации,

  д) <nbf>: время, до которого подписанный запрос аутентификации не должен приниматься к обработке,

  е) <jti>: уникальный идентификатор подписанного запроса аутентификации.

Зашифрованные JWT не поддерживаются в подписанных запросах аутентификации.

Ниже приведен пример подписанного запроса аутентификации:

```
   POST /bc-authorize HTTP/1.1
   Host: server.example.ru
   Content-Type: application/x-www-form-urlencoded

   request=eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9. eyJpc3MiOiJzNkJoZFJrcXQzIiwiYXVkI
joiaHR0cHM6Ly9zZXJ2ZXIuZXhhbXBsZS5ydSIs ImV4cCI6MTUzNzgyMDA4NiwiaWF0IjoxNTM3ODE5NDg2LC
JuYmYiOjE1Mzc4MTg4ODYsImp0 aSI6IjRMVENxQUNDMkVTQzVCV0NuTjNqNThFbkEiLCJzY29wZSI6Im9wZW5
pZCBlbWFpbCBl eGFtcGxlLXNjb3BlIiwiY2xpZW50X25vdGlmaWNhdGlvbl90b2tlbiI6IjhkNjdkYzc4LTdm
YWEtNGQ0MS1hYWJkLTY3NzA3YjM3NDI1NSIsImJpbmRpbmdfbWVzc2FnZSI6Ilc0U0NUIiwi bG9naW5faGlud
F90b2tlbiI6ImV5SjBlWEFpT2lKS1YxUWlMQ0poYkdjaU9pSkhUMU5VTXpR eE1DSjkuZXlKemRXSmZhV1FpT2
5zaWMzVmlhbVZqZEY5MGVYQmxJam9pY0dodmJtVWlMQ0p3 Yuc5dVpTSTZJaXMzTVRJek1EQXdNREF3TVNKOWZ
RLkE5a0puWDNJVUFGSHNpYzBuVWZYS1ZO bEdEV0x1bzJMODB5MkdMTXNtbjgyeEo4eXU2d0E1azNwSWVwT1V4
ckJkNzRhRF9DVkZIZ0oz cVZpYmpSQm1hd1RoaE9RTU5yZ29xVEdmUk5VV1I4QTlvZ0xvZ1hIQ1NaRW83b0NJM
UQ0enhI aHNEZVN1ME1ieTYxc0xtcl91Y2hzY3Nfa1NzZm5LcG1NcHlhSW41M3dXZEl1M09JNUJWTUFM bTdFT
VpHdXdlUmVNNU5US2ZDVU5qWE5tcDV3In0.LH3heoqfD18vh9ljE28NWKmKQWkztaXY 2GBMzDePM9yMz5L2vd
vuTQ_5p9c6szleZBwYwEcxUv4P9uqIvF6zX8Mt_qXY1sGomG1x33Ew btKdqYqFjtAwE1FVHLOEA
   &client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
   &client-assertion-type%3Ajwt-bearer
   &client_assertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJHT1NUMzQxMCJ9.eyJpc3MiOiJzNkJoZFJrcXQz
Iiwic3ViIjoiczZCaGRSa3F0MyIsImF1ZCI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUucnUiLCJqdGkiOiJiZG
MtWHNfc2YtM1lNbzRGU3pJSjJRIiwiaWF0IjoxNTkxMDc2NzE2LCJleHAiOjE1OTIwNzY3MTZ9.FuBj8SjN3bu
fQae1HgaO6BoSp4ocrnl4dnTZLJE0KzpnafAA88Ra7eoUT2QP1LhtetVAlDcT3nRJmyyRNLc7oXTYtb5mqCbNY
w4ThTm9hIgyV339hDO837vxX_GNzP6zY5nKA9nyqyRG3BbPldBkzanyQD1F0JtAVugnBZw
```

Декодированный JWT из 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.A9kJnX3IUAFHsic0nUfXKVNlGDWLu
o2L80y2GLMsmn82xJ8yu6wA5k3pIepOUxrBd74aD_CVFHgJ3qVibjRBmawThhOQMNrgoqTGfRNUWR8A9ogLogX
HCSZEo7oCI1D4zxHhsDeSu0Mby61sLmr_uchscsEkSsfnKpmMpyaIn53wWdIu3OI5BVMALm7EMZGuweReM5NTK
fCUNjXNmp5w"
   }
```

6.3.1.2. Код пользователя

Для предотвращения появления нежелательных запросов на AD конечного пользователя может быть использован параметр <user_code> (пункт 7.1.2 [8]). Код пользователя – это секрет конечного пользователя, который может быть проверен сервером авторизации.

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

Если сервер авторизации поддерживает параметр <user_code>, клиенты должны запрашивать у конечного пользователя значение кода пользователя для каждого потока аутентификации. Это позволяет предотвратить создание запросов на аутентификацию третьими лицами, знающими <login_hint> или другие идентификаторы конечного пользователя. Сервер авторизации может принимать запросы аутентификации без <user_code>, если клиент не использует статичные <login_hint>.

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

Клиент не должен сохранять код пользователя, а должен запрашивать его у пользователя при каждом использовании потока аутентификации по отдельному каналу.

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

6.3.2. Проверка запроса аутентификации

Сервер авторизации должен проверить полученный запрос аутентификации следующим образом (подраздел 7.2 [8]):

– аутентифицировать клиента с использованием метода, зарегистрированного для его `<client_id>` (подраздел 7.1 [6]). Рекомендуется, чтобы клиенты не отправляли общие секреты в запросе аутентификации, а использовали криптографию с открытым ключом;

– если запрос аутентификации подписан, проверить цифровую подпись параметра `<request>` как JWT, а также проверить структуру JWT на корректность (раздел 8 [6]);

– проверить наличие всех обязательных параметров в запросе аутентификации;

– проверить значения параметров. Если запрос содержит более одной или не содержит информации о пользователе (`<login_hint_token>`, `<id_token_hint>` или `<login_hint>`), сервер авторизации должен вернуть сообщение об ошибке `"invalid_request"`;

– проверить предоставленную информацию о пользователе на корректность состава параметров и актуальность связанного с ним конечного пользователя, а в случае применения подписанной информации о пользователе (`<id_token_hint>` или `<login_hint_token>`) проверить его цифровую подпись. Механизм проверки информации о пользователе и способы извещения клиента о требованиях к ним определяются на этапе разработки сервера авторизации исходя из его прикладных целей и задач.

Если информация о пользователе недействительна или если сервер авторизации не может определить конечного пользователя, клиенту возвращается сообщение об ошибке.

Сервер авторизации должен игнорировать нераспознанные параметры запроса. Если проверка запроса аутентификации не пройдена, сервер авторизации должен вернуть сообщение об ошибке (подраздел 6.9).

6.3.3. Успешное подтверждение запроса аутентификации

Получив и успешно проверив запрос аутентификации (пункт 6.3.2), сервер авторизации возвращает клиенту HTTP-ответ с кодом состояния 200 (OK), содержащий следующие параметры (подраздел 7.3 [8]):

– `<auth_req_id>`: (обязательный) уникальный идентификатор запроса аутентификации от клиента. Должен быть случайным и содержать достаточную энтропию (минимум 160 бит). Должен состоять из символов 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 срок действия идентификатора `<auth_req_id>` в секундах с момента получения запроса аутентификации. Клиенту возвращается сообщение 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 секунд, которое клиент ожидает между запросами к конечной точке токена. Указывается, если клиент зарегистрирован для использования режимов Poll или Ping. Если значение не указано, клиент использует значение по умолчанию «5».

Ниже приведен 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 на запрос аутентификации значения HTTP-ответа с кодом состояния 200 (OK) клиент должен проверить наличие обязательных параметров и сохранить значение идентификатора `<auth_req_id>` для проверки обратных вызовов в режимах Ping и Push или для запроса токена в режимах Poll и Ping. Также клиент должен сохранить значение `<expires_in>` для удаления запросов аутентификации, по которым не получены ответы в обратных вызовах Ping или Push (подраздел 7.4 [8]).

6.4. Получение сервером авторизации согласия / авторизации конечного пользователя

После успешной проверки запроса аутентификации сервер авторизации идентифицирует конечного пользователя и выбирает способ аутентификации пользователя и авторизации запроса в соответствии с параметром `<acr_values>` (раздел 8 [8]). Сервер авторизации инициирует интерактивную сессию с AD конечного пользователя (пункт 6.3.1 [6]). После аутентификации конечного пользователя сервер авторизации должен получить его разрешение об авторизации доступа к запрошенному ресурсу, прежде чем предоставлять информацию клиенту (пункт 6.3.2 [6]).

6.5. Конечная точка уведомления клиента

Конечная точка уведомления клиента – конечная точка, установленная клиентом во время процесса регистрации и обнаружения метаданных (пункт 6.2), на которую сервер авторизации отправляет сообщения после успешной или неудачной аутентификации конечного пользователя (раздел 9 [8]). Передача сообщений между сервером авторизации и конечной точкой уведомления клиента должна производиться по аутентифицированному TLS-соединению в соответствии с требованиями к протоколу TLS (пункт 10.1.4 [6]).

Когда клиент использует режим Ping, конечная точка уведомления получает от сервера авторизации сообщение о возможности получения результата аутентификации из конечной точки токена.

Когда клиент использует режим Push, конечная точка уведомления получает от сервера авторизации результат аутентификации (ID токен, токен доступа и, если необходимо, токен обновления или сообщение об ошибке авторизации).

Запросы к конечной точке уведомления клиента должны быть аутентифицированы 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 раздела 7 [5].

6.6. Получение результата аутентификации

Режим доставки токена клиенту (Poll, Ping или Push) определяется при регистрации клиента (параметр <backchannel_token_delivery_mode>). Клиент может зарегистрировать только один метод доставки токена, а сервер авторизации должен вернуть клиенту результат аутентификации только через зарегистрированный режим (раздел 10 [8]).

6.6.1. Запрос токена

Если клиент зарегистрирован для использования режимов Poll или Ping, то клиент получает результат аутентификации из конечной точки токена (подраздел 10.1 [8]). Клиент должен быть аутентифицирован (раздел 7 [6]).

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 (пункт 10.1.4 [6]).

Применимые механизмы аутентификации клиента на конечной точке токена определены в подразделе 7.1 [6].

Примечание. Подробнее о параметрах запроса аутентификации клиента изложено в подразделе 7.1 [6].

6.6.1.1. 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 <interval>. При этом сервер авторизации может осуществлять длительный опрос. Длительный опрос – это опрос, при котором сервер авторизации отвечает на запрос токена только в том случае, если результат аутентификации стал доступен или если истекло время ожидания ответа. Для длительных опросов рекомендуется использовать интервал 30 секунд.

Примечание. Рекомендации по опросу указаны в подразделе 5.5 [14].

При использовании режима Poll клиентам рекомендуется ждать ответа не менее 30 секунд. Даже при использовании длительного опроса серверу авторизации рекомендуется ответить в течении 30 секунд.

Интервал опроса измеряется с момента отправки клиентом запроса Poll. При длительном опросе время ответа сервера авторизации может превышать интервал <interval>. Клиент не должен отправлять два перекрывающихся запроса с одинаковым идентификатором <auth_req_id>, а должен ждать получения ответа на предыдущий запрос, прежде чем отправлять следующий запрос. В случае, когда ответ получен, а интервал прошел, клиент может немедленно отправить следующий запрос. Для управления данной ситуацией сервер авторизации возвращает HTTP-ответ с кодом состояния 503 с заголовком "Retry-After".

Примечание. Подробная информация приведена в пункте 7.1.3 [15].

Примеры поведения клиента в зависимости от ответов сервера авторизации (интервал опроса по умолчанию равен 5 секундам):

– **длительный опрос:** клиент передает запрос токена, а сервер авторизации через 30 секунд возвращает ошибку <authorization_pending>. В этом случае клиент может сразу отправить следующий запрос токена;

– **стандартный опрос:** клиент передает запрос токена, а сервер авторизации через 2 секунды возвращает ошибку <authorization_pending>. В этом случае клиент должен подождать 3 секунды, прежде чем передать следующий запрос;

– **сервер авторизации отвечает дольше 30 секунд:** клиент передает запрос токена, а сервер авторизации не возвращает ответ в течение 30 секунд. В этом случае клиент может отменить текущий запрос и передать новый.

6.6.1.2. Особенности запроса токена в режиме Ping

Если клиент зарегистрирован для использования режима Ping, то получив на конечную точку уведомления обратный вызов с идентификатором <auth_req_id>, и проверив параметр <client_notification_token>, клиент должен сделать запрос на конечную точку токена для получения результата аутентификации.

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

6.6.1.3. Параметры запроса токена

Клиент передает 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, сделанного клиентом. Сервер авторизации должен проверить, был ли <auth_req_id> выдан клиенту в ответе на запрос аутентификации. В противном случае должен вернуть ошибку.

Ниже приведен 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
}
```

6.6.1.4. Ответ на запрос токена

После получения от клиента, проверки актуальности и авторизации запроса токена, а также после аутентификации связанного с предоставленным <auth_req_id> конечного пользователя и авторизации им запроса сервер авторизации возвращает ответ на запрос токена, как определено в пункте 6.6.1 [6].

После отправки ответа на запрос токена 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 должен вернуть ответ об ошибке (подраздел 6.7).

Ниже приведен пример ответа на запрос 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
}
```

6.6.2. Обратный вызов в режиме Ping

Если клиент зарегистрирован для использования режима Ping, сервер авторизации, после успешной или неудачной аутентификации конечного пользователя, отправляет запрос HTTP POST в конечную точку уведомления клиента (подраздел 10.2 [8]). Сервер авторизации не должен отправлять обратный вызов Ping для неактуального `<auth_req_id>`, так как клиент знает срок жизни идентификатора из параметра `<expires_in>`.

В этом режиме сервер авторизации передает в заголовке HTTP "Authorization" в качестве токена на предъявителя значение параметра `<client_notification_token>`, а в теле запроса только `<auth_req_id>`. Используется формат `"application/json"`.

Ниже приведен 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"
}
```

6.6.2.1. Проверка обратного вызова в режиме Ping

Клиент должен проверить, что полученный `<client_notification_token>`, используемый для аутентификации запроса как токен на предъявителя, актуален и связан с `<auth_req_id>`. Если токен на предъявителя недействителен, рекомендуется вернуть сообщение об ошибке HTTP с кодом состояния 401 с заголовком "Unauthorized response".

Если запрос действителен, конечной точке уведомления клиента рекомендуется отправить ответ HTTP 204 (No Content). Сервер авторизации также может принимать HTTP-ответы с кодом состояния 200 (OK), а тело сообщения в ответе игнорировать.
Клиент не должен возвращать HTTP – ответы с кодом 3xx, а сервер авторизации – следовать за перенаправлениями.

Если обратный вызов действителен, клиент может использовать полученный идентификатор <auth_req_id> для передачи запроса на токен к конечной точке токена с использованием типа разрешения на доступ "urn: openid: params: grant-type: ciba". Обработка кодов ошибок HTTP в диапазонах 4xx и 5xx должна определяться подразделами 5.5 и 5.6 [15].

6.6.3. Обратный вызов в режиме Push

6.6.3.1. Ответ на запрос токена в режиме Push

При формировании ответа на запрос токена в режиме Push необходимо руководствоваться общими требованиями, определенными в пункте 6.6.1 [6].

Если клиент зарегистрирован для использования режима Push и конечный пользователь прошел аутентификацию и авторизовал запрос, сервер авторизации передает на конечную точку уведомления клиента сообщение, включающее параметр `<auth_req_id>`, ID токен, токен доступа и при необходимости токен обновления (пункт 10.3.1 [8]). Используется формат `"application/json"`.

Для привязки ID токена, токена доступа и <auth_req_id> сервер авторизации должен включить значение хэш-кода токена доступа и <auth_req_id> в ID токен, используя параметры <at_hash> и <urn: openid: params: jwt: claim: auth_req_id> соответственно. В случае, если токен обновления передается клиенту, значение его хэш-кода должно быть также добавлено в ID токен с использованием параметра <urn: openid: params: jwt: claim: rt_hash>.

Значение параметра <at_hash> определяется в соответствии с требованиями пункта 6.7.1 [6], а значение параметра <urn: openid: params: jwt: claim: auth_req_id> аналогично <at_hash> с учетом использования <auth_req_id>. При этом используется алгоритм хэширования, указанный в параметре <alg> JOSE заголовка ID токена. Этот же метод применяется для вычисления хэш-кода токена обновления (параметр <urn: openid: params: jwt: claim: rt_hash> ID токена).

Примечание. Данное требование используется только для режима Push.

Ниже приведен пример обратного вызова в режиме 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.eyJpc3MiOiJodHRwczovL3NlcnZlc
i5leGFtcGxlLnJ1Iiwic3ViIjoiMjQ4Mjg5NzYxMDAxIiwiYXVkIjoiczZCaGRSa3F0MyIsImVtYWlsIjoiamF
uZWRvZUBleGFtcGxlLnJ1IiwiZXhwIjoxNTkxMDc3ODY2LCJpYXQiOjE1Mzc4MTk1MDMsImF0X2hhc2giOiJXd
DBrVkZYTWFjcXZuSGV5VTAwMDF3IiwidXJuOm9wZW5pZDpwYXJhbXM6and0OmNsYWltOnJ0X2hhc2giOiJzSGF
oQ3VtcFhDUmc1bWtERHZ2cjR3IiwidXJuOm9wZW5pZDpwYXJhbXM6and0OmNsYWltOmF1dGhfcmVxX2lkIjoiM
WMyNjYxMTQtYTFiZS00MjUyLThhZDEtMDQ5ODZjNWI5YWMxIiwianRpIjoiMWRlMTI5NDYtZDYyYy00ZjRkLTg
1MzItNzc0ZdkyYjFkYjUzIn0.WE3SpNtWJYQb6NDLO9YqNmq9svF2v6YbbjieblpZ6qY"
   }
```

6.6.3.2. Проверка обратного вызова в режиме Push

Клиент должен проверить, что полученный <client_notification_token>, используемый для аутентификации запроса как токен на предъявителя, действителен и связан со значением идентификатора <auth_req_id>, полученным в обратном вызове Push. Если токен на предъявителя недействителен, рекомендуется вернуть сообщение об ошибке HTTP с кодом состояния 401 с заголовком "Unauthorized response".

Клиент должен проверить ID токен (пункт 6.7.2 [6]).

Клиент должен убедиться, что значение параметра <urn: openid: params: jwt: claim: auth_req_id> из ID токена соответствует значению <auth_req_id> из запроса аутентификации.

Клиент должен проверить 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 пункта 6.7.2 [6].

Если запрос действителен, конечной точке уведомления клиента рекомендуется отправить ответ HTTP с кодом состояния 204 с заголовком "No Content". Сервер авторизации также может принимать HTTP-ответы с кодом состояния 200 (OK), а тело в ответе игнорировать. Клиент не должен возвращать код HTTP 3xx, а сервер авторизации – следовать за перенаправлениями. Обработка кодов ошибок HTTP с кодом состояния в диапазонах 4xx и 5xx должна определяться подразделами 5.5 и 5.6 [15].

Ниже приведен 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"
}
```

6.7. Ответ об ошибке на запрос токена

Если запрос токена является недействительным или неавторизованным, сервер авторизации создает ответ об ошибке (раздел 11 [8]). В теле HTTP-ответа используется тип содержимого "application/json" с HTTP-ответом с кодом состояния 400 с заголовком "Bad Request". В дополнение к кодам ошибок, определенным в пункте 6.4.2 [6], применяются следующие дополнительные коды 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 5 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.

При определении ошибки необходимо учитывать следующие требования:

– в случае, когда <auth_req_id> недействителен или был выдан другому клиенту, сервер авторизации должен вернуть ошибку "invalid_grant";

– в случае, когда клиент опрашивает чаще, чем установленный интервал, сервер авторизации может вернуть ошибку "invalid_request";

– в случае, когда клиент получает ошибку "invalid_request", он не должен делать дальнейшие запросы с тем же значением <auth_req_id>;

– в случае, когда клиент зарегистрирован для использования режима Push, но вызывает конечную точку токена с типом разрешения на доступ "urn: openid: params: grant-type: ciba", сервер авторизации должен вернуть сообщение об ошибке, что запрошенная авторизация клиента не соответствует режиму Push;

– когда клиент получает код ответа 4xx с полезной нагрузкой JSON, он должен проверить полезную нагрузку, чтобы определить ошибку.

6.8. Дополнительные сведения об ошибках в режиме Push

Если клиенты зарегистрированы для использования режима Push, они могут получать дополнительные сведения об ошибках от конечной точки уведомления клиента. Ошибки передаются в формате "application/json" со следующими параметрами (раздел 12 [8]):

– <error_description> (опциональный): читаемый человеком текст ASCII [USASCII], предоставляющий дополнительную информацию разработчику клиента для понимания 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> (обязательный): идентификатор запроса аутентификации.

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

– "access_denied": конечный пользователь отклонил запрос авторизации;

– "expired_token": срок действия идентификатора <auth_req_id> истек;

– "transaction_failed": сервер авторизации столкнулся с неопределенной ошибкой, которая не позволила ему успешно завершить операцию. Этот код ошибки может использоваться для информирования клиента о том, что операция завершилась неудачей по причинам, отличным от тех, которые определены ошибками "access_denied" и "expired_token".

6.9. Ответ об ошибке аутентификации

Ответ об ошибке аутентификации возвращается непосредственно из конечной точки аутентификации в ответ на отправленный клиентом запрос аутентификации (раздел 13 [8]). Ответ об ошибке аутентификации передается в теле запроса HTTP в формате "application/json" со следующими параметрами:

– <error> (обязательный): ASCII код ошибки;

– <error_description> (опциональный): читаемый человеком текст ASCII [USASCII], предоставляющий дополнительную информацию разработчику клиента для понимания 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 дополнительной информацией разработчику клиента для понимания возникшей ошибки. Значения параметра <error_uri> не должны включать символы вне набора: %x20-21 / %x23-5B / %x5D-7E.

Список ошибок аутентификации, связанных с ошибками HTTP:

– HTTP-ответ с кодом состояния 400 с заголовком "Bad Request":

  1) "invalid_request": в запросе отсутствует обязательный параметр, он содержит недопустимое значение параметра, содержит параметр более одного раза, содержит более одного параметра с информацией о пользователе или имеет иные неправильные значения,

  2) "invalid_scope": запрошенная область действия (scope) недействительна, не определена или имеет неправильный формат,

  3) "expired_login_hint_token": указанный в запросе аутентификации <login_hint_token> недействителен, поскольку срок его действия истек,

  4) "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>),

  5) "unauthorized_client": клиент не авторизован для использования потока аутентификации,

  6) "missing_user_code": код пользователя требуется, но отсутствует в запросе,

  7) "invalid_user_code": неверный код пользователя,

  8) "invalid_binding_message": связанное сообщение ошибочно или неприемлемо для использования в контексте данного запроса;

– HTTP-ответ с кодом состояния 401 с заголовком "Unauthorized":

  1) "invalid_client": сбой аутентификации клиента (например, неверные учетные данные клиента, неизвестный клиент, аутентификация клиента не включена или неподдерживаемый метод аутентификации);

– HTTP-ответ с кодом состояния 403 с заголовком "Forbidden":

  1) "access_denied": владелец ресурса или сервер авторизации отклонили запрос.

Примечание. Если ответ об ошибке аутентификации получен до взаимодействия с пользователем, то ошибка будет получена только в том случае, если владелец ресурса или сервер авторизации принял решение об отклонении определенного типа запроса или запросов от определенного типа клиента.

Ниже приведен 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

7.1. Положения о безопасности аутентификации по обратному каналу, инициированной клиентом

Профиль безопасности OpenID API с использованием потока аутентификации по отдельному каналу применяется как к базовому, так и к расширенному профилю безопасности.

При реализации потока аутентификации по отдельному каналу необходимо руководствоваться общими требованиями к профилям безопасности OpenID API, определенными в разделах 6 и 7 [5], а также применять дополнительные требования, изложенные в данном разделе.

7.1.1. Сервер авторизации

Сервер авторизации должен (пункт 5.2.2 [11]) поддерживать положения, указанные в подразделе 7.2 [5].

Кроме того, сервер авторизации для всех операций:

– должен поддерживать только конфиденциальных клиентов для инициированных клиентом потоков аутентификации по отдельному каналу;

– должен обеспечивать однозначное определение требуемой для авторизации информации в запросе авторизации или требовать наличия `<binding_message>` в запросе аутентификации;

– не должен поддерживать режим Push;

– должен поддерживать режим Pool;

– может поддерживать режим Ping;

– должен поддерживать подписанные запросы аутентификации к конечной точке аутентификации по отдельному каналу (подпункт 6.3.1.1);

– должен требовать уровня аутентификации пользователя, соответствующего требованиям операций, которые клиент будет уполномочен выполнять от имени пользователя;

– должен, если он поддерживает класс контекста аутентификации, указанный через параметр `<acr>` в запросе клиента, возвращать указанное значение `<acr>` в запрашиваемом ID токене;

– должен требовать, чтобы подписанный запрос аутентификации содержал параметры `<nbf>` и `<exp>`, которые ограничивают время жизни запроса 60 минутами;

– может требовать от клиентов предоставить параметр `<request_context>`, как это определено в подразделе 7.2;

– не должен использовать `<login_hint>` или `<login_hint_token>` для передачи «идентификаторов намерений» или любых других метаданных авторизации.

Примечания:

1. Для целей настоящего стандарта <login_hint>, <login_hint_token> и <id_token_hint> используются только для определения конечного пользователя.

2. 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, что и определенные [5] для конечной точки токена.

7.1.2. Конфиденциальный клиент

Конфиденциальный клиент должен (пункт 5.2.3 [11]) поддерживать положения, указанные в подразделе 7.3 [5].

Кроме того, конфиденциальный клиент:

– должен отправлять в конечную точку аутентификации отдельного канала только подписанные запросы аутентификации (подпункт 6.3.1.1);

– должен обеспечивать в запросе аутентификации наличие информации для определения требуемого уровня аутентификации или включать в запрос аутентификации параметр <binding_message>;

– должен быть уверен, что сервер авторизации аутентифицировал пользователя с использованием методов авторизации, соответствующих цели клиента. Способы извещения клиента о требуемых методах авторизации определяются на этапе разработки сервера авторизации исходя из его прикладных целей и задач.

7.2. Расширения для запроса аутентификации

Профиль определяет (подраздел 5.3 [11]) следующее расширение 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.

7.3. Доступ к защищенным ресурсам

В профиле применяются положения касательно доступа клиента к защищенным ресурсам, представленные в подразделах 6.4 и 7.4 [5] с учетом того, что после выпуска токенов в потоке аутентификации по отдельному каналу их можно использовать так же, как и токены, выпущенные через потоки кода авторизации (подраздел 6.1 [11]).

7.3.1. Положения клиента

В ситуациях, когда клиент не контролирует CD (подраздел 6.2 [11]):

– клиент не должен передавать <x-fapi-customer-ip-address> или <x-fapi-auth-date> в заголовке HTTP;

– клиенту рекомендуется передавать метаданные о CD (например, тип устройства и геолокацию).

7.3.2. Механизмы защиты

В профиле применяются механизмы защиты, указанные в подразделе 5.8 [5], а также следующие дополнительные меры:

– сервер авторизации должен быть уверен, что установленная во время динамической регистрации клиента (подпункт 5.4.4.4 [5]) параметром `<backchannel_client_notification_endpoint>` конечная точка уведомления клиента находится в окружении администрирования клиента;

– `<login_hint_token>` должен быть подписан цифровой подписью эмитента, что гарантирует подлинность данных и снижает угрозу атаки с помощью внедрения данных. Подпись позволяет серверу авторизации аутентифицировать и авторизовать отправителя информации о пользователе и предотвращать сбор пользовательских идентификаторов злоумышленниками, выступающими в роли клиента.

7.4. Регистрация и обнаружение метаданных

Настоящий стандарт предусматривает дополнительные параметры метаданных сервера авторизации и метаданных динамической регистрации клиентов (раздел 7 [11]).

Метаданные сервера авторизации:

– `<backchannel_endpoint_login_hint_token_types_supported>`: (опциональный) JSON-массив, в котором перечислены поддерживаемые сервером типы `<login_hint_token>`.

Метаданные динамической регистрации клиента:

– `<backchannel_endpoint_login_hint_token_types>`: (опциональный) JSON-массив, который клиент может использовать для регистрации типа `<login_hint_token>.`

7.5. Вопросы безопасности

7.5.1. Инициация сессий аутентификации без ведома или участия конечного пользователя

В потоке аутентификации по отдельному каналу сервер авторизации должен иметь информацию о согласии конечного пользователя с процессом аутентификации (подраздел 8.2 [11]).
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, чтобы `<login_hint>` был непредсказуемым. Параметр генерируется на устройстве AD, после чего попадает на сервер авторизации, с которым взаимодействует AD (например, фронт сервера авторизации), и затем этот параметр передается клиенту для включения в `<login_hint>`.
В качестве альтернативного варианта идентификации конечного пользователя может быть использован `<id_token_hint>`.

Если клиент будет сохранять <id_token>, полученный от сервера авторизации, для дальнейшего использования в качестве <id_token_hint>, необходимо, чтобы механизм аутентификации клиента соответствовал используемому каналу.

Кроме того, для защиты от этой угрозы может быть использован параметр <user_code> (подпункт 6.3.1.2).

7.5.2. Подтверждение пользователем значения <binding_message>

В зависимости от информации о пользователе, используемой для его идентификации и процессов аутентификации пользователя клиентом, мошенник может осуществить атаку подмены потока аутентификации, запустив собственный поток аутентификации на AD одновременно с подлинным потоком, причем оба потока будут использовать актуальный идентификатор пользователя. Если область запрашиваемого 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 использовать временные идентификаторы пользователя, сгенерированные на AD (пункт 7.5.1).

7.5.3. Другие вопросы безопасности

Для обеспечения информационной безопасности при аутентификации и получения доступа следует руководствоваться разделом 6 [5].

БИБЛИОГРАФИЯ

[1] Стандарт Банка России «Открытые банковские интерфейсы. Общие положения». 23.10.2020. https://www.cbr.ru/StaticHtml/File/59420/standart_1. pdf (дата обращения: 22.01.2024).

[2] Стандарт Банка России «Открытые банковские интерфейсы. Получение публичной информации о кредитной организации и ее продуктах». 08.07.2021. http://www.cbr.ru/statichtml/file/59420/standart_08072021. pdf (дата обращения: 22.01.2024).

[3] Стандарт Банка России «Открытые банковские интерфейсы. Инициирование перевода денежных средств клиента третьей стороной в валюте Российской Федерации». 23.10.2020. https://www.cbr.ru/StaticHtml/File/59420/standart_3. pdf (дата обращения: 22.01.2024).

[4] Стандарт Банка России «Открытые банковские интерфейсы. Получение информации о счете клиента третьей стороной». 23.10.2020. https://www.cbr.ru/StaticHtml/File/59 420/standart_2. pdf (дата обращения: 22.01.2024).

[5] 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».

[6] Методические рекомендации МР. 26.2.002-2024 ТК 26 «Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколах OpenID Connect. 2024».

[7] Sakimura N., Bradley J., Jones M., de Medeiros B., Mortimore C. OpenID Connect Core 1.0 incorporating errata set 1. November 8, 2014. https://openid.net/specs/openid-connect-core-1_0. html (дата обращения: 22.01.2024).

[8] Fernandez G., Walter F., Nennker A., Tonge D., Campbell B. OpenID Connect Client-Initiated Backchannel Authentication Flow – Core 1.0. September 1, 2021. https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0. html (дата обращения: 22.01.2024).

[9] Sakimura N., Bradley J., Jones M., Jay E. OpenID Connect Discovery 1.0 incorporating errata set 1. November 8, 2014. https://openid.net/specs/openid-connect-discovery-1_0. html (дата обращения: 22.01.2024).

[10] Sakimura N., Bradley J., Jones M. OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 1. November 8, 2014. https://openid.net/specs/openid-connect-registration-1_0. html (дата обращения: 22.01.2024).

[11] Tonge D., Heenan J., Lodderstedt T., Campbell B. Financial-grade API: Client Initiated Backchannel Authentication Profile. January 11, 2023. openid/fapi/Financial_API_WD_CIBA. md – Bitbucket (дата обращения 22.01.2024).

[12] Hardt D. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012. https://tools.ietf.org/html/rfc6749 (дата обращения: 22.01.2024).

[13] Jones M., Hardt D. The OAuth 2.0 Authorization Framework: Bearer Token Usage. RFC 6750, October 2012. https://tools.ietf.org/html/rfc6750 (дата обращения: 22.01.2024).

[14] Loreto S., Saint-Andre P., Salsano S., Wilkins G. Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP, RFC 6202. April 2011. https://tools.ietf.org/html/rfc6202 (дата обращения: 22.01.2024).

[15] Fielding R., Reschke J. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231. June 2014. https://datatracker.ietf.org/doc/html/rfc7231 (дата обращения: 22.01.2024).

[16] Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных».

[17] Постановление Правительства Российской Федерации от 01.11.2012 № 1119 «Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных».

[18] 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».

[19] 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».

[20] Положение Банка России от 17.08.2023 № 821-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».

[21] 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».

[22] 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».

[23] Положение Банка России от 25.07.2022 № 802-П «О требованиях к защите информации в платежной системе Банка России».

[24] ГОСТ Р ИСО/МЭК 15 408-3-2013 «Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности».

[25] Методический документ Банка России «Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций».

[26] Методический документ ФСТЭК «Руководство по организации процесса управления уязвимостями в органе (организации)» (утв. ФСТЭК России 17.05.2023). https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-17-maya-2023-g (дата обращения: 22.01.2024).

[27] Методический документ ФСТЭК «Методика тестирования обновлений безопасности программных, программно-аппаратных средств» (утв. ФСТЭК России 28.10.2022). https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-28-oktyabrya-2022-g (дата обращения: 22.01.2024).

[28] Методический документ ФСТЭК «Методика оценки уровня критичности уязвимостей программных, программно-аппаратных средств» (утв. ФСТЭК России 28.10.2022).

[29] ГОСТ Р 58833-2020 «Защита информации. Идентификация и аутентификация. Общие положения».

[30] ГОСТ Р 34.10-2012 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи».

[31] Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. May 2015. https://datatracker.ietf.org/doc/html/rfc7519 (дата обращения: 22.01.2024).

[32] Josefsson S. The Base16, Base32, and Base64 Data Encodings. RFC 4648. October 2006. https://datatracker.ietf.org/doc/html/rfc4648 (дата обращения: 22.01.2024).

[33] Положение Банка России от 17.10.2022 № 808-П «О требованиях к обеспечению защиты информации при осуществлении деятельности в сфере оказания профессиональных услуг на финансовом рынке в целях противодействия осуществлению незаконных финансовых операций, обязательных для лиц, оказывающих профессиональные услуги на финансовом рынке, к обеспечению бюро кредитных историй защиты информации, указанной в статье 4 Федерального закона «О кредитных историях», при ее обработке, хранении и передаче сертифицированными средствами защиты, а также к сохранности информации, полученной в процессе деятельности кредитного рейтингового агентства».