EBU

OPERATING EUROVISION AND EURORADIO

TECH 3366

THE CROSS PLATFORM AUTHENTICATION PROTOCOL

VERSION 1.0

SOURCE: CROSS PLATFORM AUTHENTICATION GROUP

Geneva
September 2014

TECH 3366

THE CROSS PLATFORM AUTHENTICATION PROTOCOL

VERSION 1.0

Внимание!

Данный перевод **НЕ** претендует на аутентичность и может содержать отдельные неточности.
Оригинал документа на сайте http://tech.ebu.ch

МЕЖПЛАТФОРМНЫЙ ПРОТОКОЛ АУТЕНТИФИКАЦИИ

ВЕРСИЯ 1.0

ИСТОЧНИК: CROSS PLATFORM AUTHENTICATION GROUP

Женева
Сентябрь 2014

Версии документа

Предполагается, что данный документ превратится в серию, в которой будут постепенно добавляться аспекты протокола. Документ определяет ассоциацию удостоверенных устройств, а также взаимодействие между провайдером услуг и провайдером авторизации.

Следующие итерации включают связь между серверами и браузерные потоки типа Native applications и Connected TVs.

Обратите внимание, что каждая их них описывает версию 1.0 протокола CPA, как описано в версии протокола.

Авторы

Редактор: Sean O'Halpin (BBC).

Авторы: Chris Needham (BBC), Michael Barroco (EBU).

Официальное уведомление

EBU выражает признательность следующим лицам за работу над подготовкой данной спецификации.

Floris Daelemans (VRT), Sébastien Noir (RTS), Rafał Wiosna (TVP), Robin Cooksey (Frontier Silicon), Byrion Smith (Global Radio), Andy Buckingham (togglebit).

Содержание

Межплатформный протокол аутентификации

Комитет EBU	*Первый выпуск*	*Переработка*	*Переиздание*
TC	2014		

Ключевые слова: Аутентификация, OAuth, Провайдер услуг, Провайдер авторизации, IP, Smart TV, HTTPS, токен.

1. Введение

Настоящий документ определяет версию 1.0 протокола Cross Platform Authentication *(CPA)*.

Основная цель CPA – определить, как безопасно связать IP-подключенное медиа устройство (например, гибридное радио, приставку или смарт-ТВ) с интернет-аккаунтом пользователя ряда веб-услуг, передаваемых на это устройство.

Этот протокол специально предназначен для устройств с ограниченными возможностями ввода и отображения, не охваченных существующими стандартами, и для компаний, имеющих общего оконечного провайдера авторизации для управления идентификаторами, но с отдельной реализацией услуг.

Протокол CPA определяет четкое разделение ответственности между провайдером услуг и провайдером аутентификации. Это позволяет применять протокол в различных бизнес-конфигурациях, типичных для индустрии вещания. Например, протокол CPA можно использовать в следующих сценариях:

- провайдером услуг и провайдером авторизации управляет одна и та же компания
- зонтичная компания управляет провайдером авторизации (централизованное управление идентификаторами), но держит провайдеров услуг отдельно друг от друга
- провайдеры услуг из разных компаний используют одного центрального провайдера авторизации, которым управляет отдельная компания (например, национальная федеративная служба идентификации)

2. Область применения

Протокол, описанный в данном документе, предназначен для устройств с ограниченными возможностями ввода, например, гибридных радиоприемников, IP-подключенных приставок и смарт-ТВ, которые могут соединяться с веб-услугами по HTTPS.

Протокол определяет API между клиентским устройством и провайдером авторизации.

2.1 Ограниченный ввод и отображение

Устройство с ограниченным вводом имеет как минимум возможность принятия или отмены предлагаемого действия.

Ограниченное отображение – возможность отображать как минимум две строки 16 буквенно-числовых символов из ISO-646 Invariant Code Set (примерно 7-bit ASCII).

3. Определения и аббревиатуры

Следующие термины и аббревиатуры используются в данном документе со следующими значениями.

3.1 Определения

Application
Приложение
Программа, работающая в устройстве, которое может использовать HTTPS по IP для соединения с публично адресуемыми услугами и управления состояниями клиента.

Association
Ассоциация
Присвоение идентификатора клиенту, который может использоваться провайдером услуг для сохранения данных, связанных с этим клиентом.

Неудостоверенная ассоциация – форма ассоциации, где идентификатор клиента не связан с идентификатором пользователя.

Удостоверенная ассоциация – форма ассоциации, где идентификатор клиента связан с идентификатором пользователя.

Authorization Provider *Провайдер авторизации*	Веб-служба, публично адресуемая через HTTPS, которая управляет идентификаторами клиентов, связью идентификаторов клиентов с удостоверенными идентификаторами пользователей, выдачей токенов клиентам и связью этих токенов с соответствующими идентификаторами клиентов. Провайдер услуг использует токен, включенный в клиентские запросы, для запроса пользовательской информации у провайдера авторизации.
Bearer Token *Токен на предъявителя*	Вид *токена,* который дает доступ к защищенному ресурсу через предъявителя, который попросту им владеет. Другими словами, если у вас есть действительный токен на предъявителя, то у вас есть доступ к ресурсу без дальнейших вопросов.
Client *Клиент*	Один экземпляр отношения с провайдером авторизации. Каждый клиент имеет уникальный идентификатор, предоставленный провайдером авторизации, обозначенный его `client_id`. *Клиент* не представляет непосредственно пользователя. Он представляет использование провайдера услуг приложением в устройстве. Например, лучше считать идентификатор клиента представлением «Моего кухонного радио при слушании 1-го канала», чем «Меня».
Client Mode *Клиентский режим*	Означает опциональный режим авторизации, в котором взаимодействие между клиентом, провайдером услуг и провайдером авторизации идентифицируется только по client_id и не требуется, чтобы клиент был связан с удостоверенным пользовательским аккаунтом. Этот режим позволяет хранение данных у провайдера услуг, идентифицированного только client_id для поддержки «стандартного» опыта. Ср. с *пользовательским режимом*.
Device *Устройство*	IP-подключенное устройство, например, гибридное радио, приставка или смарт-ТВ, которое может содержать приложения и хранить данные для управления клиентами.
Identity Provider *Провайдер идентификаторов*	Служба, которая удостоверяет идентичность пользователя.
Registration *Регистрация*	Процесс, в котором клиент получает идентификатор клиента у провайдера авторизации.
Service Provider *Провайдер услуг*	Онлайн-служба, которая требует авторизации для доступа к защищенным ресурсам.
Token *Токен*	Уникальная непрозрачная строка, используемая для представления авторизации на пользование услугой.
User *Пользователь*	Лицо или агент, пользующийся услугами, предоставленными провайдером услуг через клиента.
User Mode *Пользовательский режим*	Означает опциональный режим авторизации, в котором взаимодействие между клиентом, провайдером услуг и провайдером авторизации идентифицируется по связи с пользовательским аккаунтом.

3.2 Аббревиатуры

API	Application Programming Interface Интерфейс прикладного программирования
CPA	Cross Platform Authentication Межплатформная аутентификация (тема данной спецификации)
HTTP	Hypertext Transfer Protocol Протокол передачи гипертекста
HTTPS	HTTP Secure
OAuth	Open Authorization Открытая авторизация
URI	Uniform Resource Identifier Унифицированный идентификатор ресурса

4. Обзор

Спецификация потока устройства CPA определяет протокол между клиентским устройством и провайдером авторизации, у которого клиентское устройство получает токен авторизации, который может использовать для доступа к защищенным ресурсам и по которому провайдер услуг может идентифицировать клиентское устройство и соответствующий пользовательский аккаунт.

Хотя CPA может использоваться более широко, этот документ сосредоточен на IP-подключенных медиа устройствах с ограниченным вводом и отображением. Поскольку такое устройство обычно не может запускать HTTP User Agent со всеми возможностями, например, браузер, пользователь должен иметь доступ к другому компьютеру, где можно запустить такую программу для завершения авторизации.

В типичном потоке устройства CPA, описанном ниже, шаги, связанные с приложением, имеют префикс *(App),* а связанные с CPA - *(CPA)*:

- *(App)*: Приложение в устройстве запрашивает защищенный ресурс у провайдера услуг.
- *(App)*: Провайдер услуг сверяется с провайдером авторизации, авторизовано ли устройство.
- *(App)*: Провайдер авторизации показывает, что устройство не авторизовано.
- *(App)*: Ответ провайдера услуг устройству включает оконечную точку, где устройство может инициировать поток устройства, и какие режимы ассоциации поддерживает провайдер услуг.
- *(CPA)*: Устройство регистрируется у провайдера авторизации и получает взамен `client_id` и `client_secret`.
- *(CPA)*: Устройство делает запрос провайдеру авторизации на ассоциацию устройства.
- *(CPA)*: Провайдер авторизации возвращает URL подтверждения и одноразовый `user_code`, который устройство показывает пользователю. Он также возвращает соответствующий `device_code`, который не отображается.
- *(CPA)*: Устройство опрашивает провайдера авторизации, используя для идентификации `device_code`.
- *(App)*: Тем временем пользователь в браузере на другом компьютере открывает приложенный URL подтверждения, чтобы ввести код. В этом процессе пользователю потребуется удостоверить свою идентичность, обычно путем регистрации через провайдера идентификаторов.
- *(CPA)*: После того, как пользователь подтвердил ассоциацию, введя свой код, провайдер авторизации отвечает устройству, возвращая токен на предъявителя.
- *(App)*: Приложение в устройстве может дать повторный запрос на защищенный ресурс, на этот раз с полученным токеном на предъявителя.
- В вышеописанном потоке устройства CPA определяет только протокол между клиентским устройством и провайдером авторизации. Этот протокол определяется тремя оконечными точками, данными провайдером авторизации:
- /register для получения идентификатора клиента, состоящего из `client_id` и `client_secret`
- /associate для инициации ассоциации между клиентом и пользовательским аккаунтом
- /token для получения токена

Все прочие взаимодействия не входят в рамки данной спецификации. Однако она включает иллюстративные примеры, как можно интегрировать CPA в поток, связанный с приложением. Эти точки интеграции включают:

- как клиент обнаруживает оконечную точку авторизации
- как провайдер услуг и провайдер авторизации делятся информацией
- как пользователь подтверждает ассоциацию между своим устройством и пользовательским аккаунтом

Эти примеры призваны помочь реализаторам в понимании семантики спецификации CPA, но сами не являются частью спецификации.

4.1 Режимы ассоциации

CPA определяет два режима ассоциации: *удостоверенная* ассоциация ('user mode'), которая связывает устройство с пользовательским аккаунтом, и *неудостоверенная* ассоциация ('client mode'), которая дает идентификатор клиента устройству без связи его с пользовательским аккаунтом.

В обоих случаях устройство получает постоянный клиентский ID, чтобы клиент мог идентифицироваться в запросах провайдеру услуг, который использует того же провайдера авторизации.

Провайдеры авторизации могут предлагать следующие формы ассоциации:

- только клиентский режим
- только пользовательский режим
  клиентский и пользовательский режим

4.1.1 Удостоверенная ассоциация ('user mode')

Если провайдер авторизации предлагает пользовательский режим, то клиентский ID может быть связан с пользовательским аккаунтом, чтобы запросы устройства можно было связать с этим пользователем. Такая связь устройства и пользователя называется «удостоверенной ассоциацией» или *user mode* (см. § 7.3, *User mode*).

После установки удостоверенной ассоциации провайдер услуг может обеспечивать унифицированную персонализированную услугу через ряд устройств.

4.1.2 Неудостоверенная ассоциация ('client mode')

CPA также определяет способ создания *неудостоверенной* ассоциации между устройством и провайдером авторизации, известной как *client mode* (см. § 7.4, *Client mode*), и способ перехода от этой неудостоверенной ассоциации к удостоверенной.

Цель неудостоверенной ассоциации – поддержка «стандартного» опыта, которая требует постоянства со стороны услуги, но не требует ассоциации с интернет-аккаунтом пользователя.

Примечание: Под «стандартным» опытом мы подразумеваем использование устройства для доступа к веб-услугам без требования от пользователя удостоверенного интернет-аккаунта и потому без требования конфигурации.

Используя `client_id` как идентификатор, провайдер услуг может идентифицировать клиента от сеанса к сеансу и таким образом обеспечить персонализацию, ограниченную определенным устройством.

5. Основные понятия

5.1 Токен на предъявителя

Поток устройства CPA построен на концепции *токена на предъявителя*, обладание которым дает доступ к защищенным услугам.

Базовая функция протокола CPA – безопасное помещение этого токена на предъявителя в устройство.

Токен – уникальная строка, исключающая возможность подделки. Токен – общий секрет клиента, провайдера услуг и провайдера авторизации.

Токен на предъявителя – это токен, простое обладание которым дает соответствующий авторитет его предъявителю.

В CPA токен также служит ключом для поиска ассоциации между клиентом и пользовательским аккаунтом. Это не постоянный идентификатор, т.к. может кончиться и замениться новым токеном.

5.2 Домен

Токены действительны только в пределах имени хоста провайдера услуг, для которого они выданы. Это не дает провайдерам услуг выдавать токены для доменов, за которые они не отвечают.

Токены действительны для всех оконечных точек провайдера услуг в данном домене. Например, токен, выданный для домена ***api.example.com,*** действителен для оконечных точек ***https://api.example.com/tag*** и ***https://api.example.com/epg***.

Домен определяется клиентом при запросе токена. Провайдер авторизации решает, разрешить ли клиенту иметь токен для запрашиваемого домена.

6. Роли

Роли, определяемые CPA – это *клиент*, *провайдер услуг* и *провайдер авторизации*.

6.1 Клиент

Клиент представляет пользователя услуги в устройстве.

Когда клиент регистрируется у провайдера авторизации, он получает уникальный `client_id` и `client_secret`.

`client_id` известен клиенту, провайдеру услуг и провайдеру авторизации. Этот идентификатор клиента общий для всех провайдеров услуг, использующих одного провайдера авторизации.

`client_id` должен быть уникальным внутри провайдера авторизации, и данный `client_id` действителен только для этого провайдера авторизации. Разные провайдеры авторизации не имеют общих идентификаторов клиентов.

`client_secret` известен только клиенту и провайдеру авторизации. Все запросы клиента провайдеру авторизации должны включать `client_id` и `client_secret`, и этот `client_secret` должен использоваться только в запросах от клиента провайдеру авторизации.

6.2 Провайдер услуг

Провайдер услуг обеспечивает одну или более веб-услуг в одном домене.

Каждый токен на предъявителя действителен только для того домена провайдера услуг, для которого выдан.

Точные детали того, как провайдер услуг показывает клиенту, какого провайдера авторизации использовать и какие защищенные ресурсы доступны с токеном на предъявителя CPA, зависят от приложения и не определены в настоящем документе.

Один из способов может заключаться в том, что провайдер услуг возвращает заголовок `WWW-Authenticate` в ответ на запрос защищенного ресурса, как описано в § 7.7.1, *Вызов аутентификации*. Этот заголовок сообщит клиенту URL провайдера авторизации и доступные режимы ассоциации.

6.3 Провайдер авторизации

Провайдер авторизации обеспечивает идентификаторы клиента одному или более клиентов и поддерживает связь между идентификатором клиента и идентификатором пользователя.

Провайдер авторизации может отвечать на запросы авторизации одного или более провайдеров услуг.

Провайдеры услуг могут группироваться внутри провайдера авторизации для коллективного пользования доменом авторизации. Это позволяет автоматическую выдачу токенов провайдерам услуг в одной группе после проверки клиента для одного провайдера услуг внутри этой группы.

Предполагается, что между провайдером услуг и провайдером авторизации используется безопасное доверительное соединение и что провайдер авторизации может проверить идентичность провайдера услуг.

7. Поток устройства

7.1 Версия протокола

В этом документе описана версия 1.0 протокола CPA. Метод, который должны использовать провайдеры услуг для индикации версии, зависит от приложения, например, см. § 7.7.1, *Вызов аутентификации*.

7.2 Основные принципы

7.2.1 HTTPS

Все сообщение между клиентом и провайдером услуг, между клиентом и провайдером авторизации и между провайдером услуг и провайдером авторизации ДОЛЖНО производиться по HTTPS согласно [2]. Клиенты ДОЛЖНЫ проверять сертификаты, используемые провайдером услуг и провайдером авторизации.

7.2.2 JSON

Все запросы, сделанные клиентом провайдеру авторизации, используют JSON в теле запроса для указания параметров. Все ответы провайдера авторизации используют JSON в теле ответа для возвращения данных клиенту.

Ответы об ошибках (коды состояний 4xx) содержат тело в форме:

```
{
     "error" : "..."
}
```

где поле error ОБЯЗАТЕЛЬНО, а ... указывает определенную ошибку, которая произошла.

7.2.3 Интеграция приложений с протоколом CPA

Как клиент обнаруживает URI провайдера авторизации и как он указывает токен на предъявителя провайдеру услуг, зависит от приложения и не входит в рамки данной спецификации. Однако есть некоторые соглашения, возникшие на практике, которые мы рекомендуем.

Например, протокол RadioTAG использует следующие заголовки:

- Вызов `WWW-Authenticate` (см. § 7.7.1, *Вызов аутентификации*)
- Заголовок токена на предъявителя в запросе (см. § 7.7.2, *Доступ к защищенному ресурсу*)

В самом протоколе CPA клиент и провайдер авторизации всегда используют для общения полезную нагрузку JSON.

7.2.4 Пример удостоверенной ассоциации с потоком устройства CPA

Ниже приведен пример удостоверенной ассоциации с потоком устройства CPA с точки зрения пользователя.

- Алиса включает радиоприемник, и он начинает воспроизводить ее любимый канал. Приложение в приемнике обнаруживает, что для этого канала есть услуга, позволяющая Алисе добавить воспроизводимую дорожку в избранное, нажав кнопку 'tag'. Алиса нажимает кнопку 'tag'.
- Приемник вызывает провайдера услуг и получает ответ, содержащий вызов авторизации с указанием провайдера авторизации. Приемник *регистрируется* у провайдером авторизации (`/register`) и получает идентификатор клиента и секрет. Теперь приемник может дать Алисе возможность *ассоциации* ее радиоприемника и интернет-аккаунта с вещателем. Алиса решает это сделать.
- Приемник вызывает провайдера авторизации для инициации ассоциации (`/associate`). Провайдер авторизации возвращает URI и код для входа. Приемник показывает это Алисе, которая затем посещает адреса, указанные URI, в своем браузере, входит в систему (это не входит в рамки данной спецификации) и вводит код.
- Тем временем приемник запрашивает оконечную точку `/token`, чтобы определить, завершила ли Алиса этот процесс. Когда Алиса успешно введет код, запрос `/token` вернет новый токен на предъявителя, авторизующий доступ к запрашиваемой услуге.
- Теперь приемник может использовать этот токен на предъявителя в будущих вызовах провайдера услуг. Провайдер услуг спросит провайдера авторизации, авторизован ли токен. Провайдер авторизации скажет, что да, и вернет идентификатор аккаунта Алисы и `client_id` радио клиента.

7.3 User mode

В пользовательском режиме взаимодействие между клиентом и провайдером услуг связано с пользовательским аккаунтом. Процесс получения клиентом токена для доступа к защищенным ресурсам у провайдера услуг показан на **Рис. 1**.

1. Сначала клиент запрашивает ресурс у провайдера услуг, либо без токена на предъявителя, либо с недействительным токеном. Провайдер услуг проверяет действительность токена, используя оконечную точку `/authorized` провайдера авторизации. Если токен отсутствует или недействителен, провайдер авторизации возвращает провайдеру услуг `404 Not Found`. Затем провайдер услуг возвращает клиенту `401 Unauthorized`, включив в ответ вызов авторизации через HTTP заголовок `WWW-Authenticate`, который указывает URI провайдера аутентификации и режимы, поддерживаемые провайдером услуг, как описано в § 7.7.1, *Вызов аутентификации*.
2. Если клиент еще не зарегистрирован у этого провайдера авторизации, клиент передает POST в оконечную точку `/register` провайдера авторизации. Провайдер авторизации регистрирует клиента и возвращает `client_id` и `client_secret` со статусом HTTP `201 Created`. Клиент хранит эту информацию связанной с именем домена провайдера авторизации для возможности использовать `client_id` с другим провайдером услуг, использующим того же провайдера авторизации.

3. Клиент инициирует процесс ассоциации с пользовательским аккаунтом, передавая свои `client_id`, `client_secret` и домен провайдера услуг в оконечную точку провайдера авторизации `/associate`. Провайдер авторизации возвращает `user_code`, отображаемый устройством пользователю, а также `device_code` и `verification_uri`.
4. Используя `device_code`, клиент начинает запрашивать оконечную точку `/token` провайдера авторизации, чтобы проверить, подтверждена ли ассоциация пользователем. Провайдер авторизации возвращает ответ `202 Accepted`, что означает. что пользователь пока не подтвердил ассоциацию с помощью `user_code`.

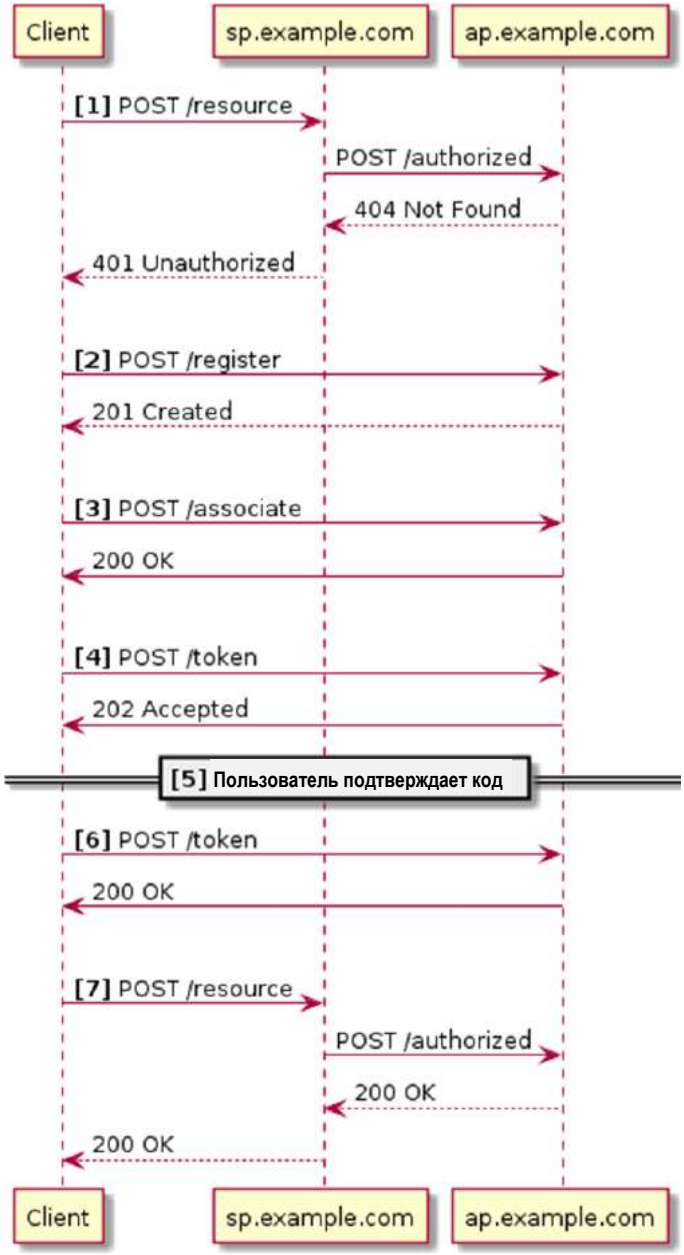

Рис. 1: Обзор потока устройства в пользовательском режиме CPA

5. Пользователь идет на `verification_uri`, входит в систему безопасным методом (который не входит в рамки данной спецификации) и вводит `user_code`. Теперь провайдер авторизации может создать связь между `client_id` и идентификатором пользователя.
6. Клиент запрашивает оконечную точку `/token` провайдера авторизации и получает ответ `200 OK`, содержащий `access_token` и `user_name`.
7. Теперь клиент может запрашивать защищенный ресурс у провайдера услуг, используя `access_token`. Провайдер услуг проверяет, действителен ли `access_token`, с помощью запроса POST в оконечную точку `/authorized` провайдера авторизации. Если он действителен, провайдер авторизации возвращает провайдеру услуг `200 OK`, включая `client_id` и `user_id`. Провайдер услуг возвращает клиенту защищенный ресурс, возможно, персонализированный с помощью `user_id`.

7.4 Client mode

В клиентском режиме взаимодействие между клиентом и провайдером услуг не связано с пользовательским аккаунтом. Провайдер авторизации выдает клиенту уникальный идентификатор, позволяющий провайдеру услуг хранить информацию и персонализировать действия, связанные с этим устройством.

Процесс получения клиентом токена для доступа к защищенным ресурсам провайдера услуг показан на **Рис. 2**.

1. Сначала клиент запрашивает ресурс у провайдера услуг, либо без токена на предъявителя, либо с недействительным токеном. Провайдер услуг проверяет действительность токена, используя оконечную точку `/authorized` провайдера авторизации. Если токен отсутствует или недействителен, провайдер авторизации возвращает провайдеру услуг `404 Not Found`. Затем провайдер услуг возвращает клиенту `401 Unauthorized`, включив в ответ вызов авторизации через HTTP заголовок `WWW-Authenticate`, который указывает URI провайдера аутентификации и режимы, поддерживаемые провайдером услуг, как описано в § 7.7.1, *Вызов аутентификации*.
2. Если клиент еще не зарегистрирован у этого провайдера авторизации, клиент передает POST в оконечную точку `/register` провайдера авторизации. Провайдер авторизации регистрирует клиента и возвращает `client_id` и `client_secret` со статусом HTTP `201 Created`. Клиент хранит эту информацию связанной с именем домена провайдера авторизации для возможности использовать `client_id` с другим провайдером услуг, использующим того же провайдера авторизации.
3. Клиент передает POST в оконечную точку `/token` провайдера авторизации, включив в ответ свои `client_id` и `client_secret`, а также домен провайдера услуг. Провайдер авторизации возвращает в ответ `200 OK`, содержащий `access_token` для запрашиваемого провайдера услуг.
4. Теперь клиент может запрашивать защищенный ресурс у провайдера услуг, используя `access_token`. Провайдер услуг проверяет, действителен ли `access_token`, с помощью запроса POST в оконечную точку `/authorized` провайдера авторизации. Если он действителен, провайдер авторизации возвращает провайдеру услуг `200 OK`, включая `client_id`. Провайдер услуг возвращает клиенту защищенный ресурс, возможно, персонализированный с помощью `client_id`.

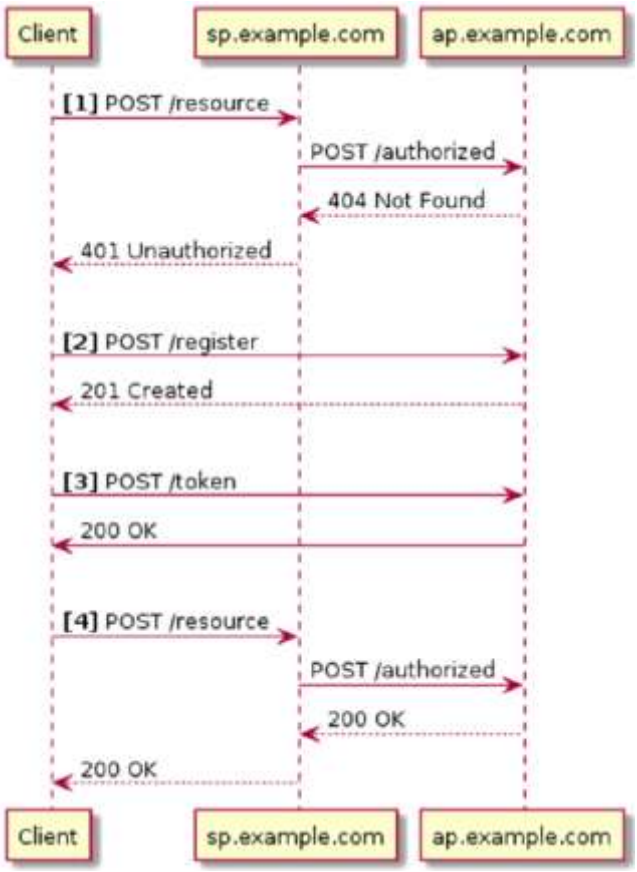

Рис. 2: Обзор потока устройства в клиентском режиме CPA

7.5 Автоматическая выдача токенов

Провайдер авторизации может автоматически выдавать токен для домена провайдера услуг для возможности единой регистрации у множества провайдеров услуг. Этот механизм использует существующую ассоциацию пользователя у провайдера авторизации и внутренние бизнес-правила, отражающие отношения между провайдерами услуг. Это позволит пользователю регистрироваться лишь однажды, чтобы взаимодействовать с разными провайдерами услуг, потенциально – из множества организаций.

`client_id` у провайдера авторизации может использоваться для определения, имеет ли уже устройство связанный с ним пользовательский аккаунт. После идентификации пользовательского аккаунта таким образом любое взаимодействие с дальнейшими провайдерами услуг могут быть автоматически ассоциированы с этим аккаунтом или другим аккаунтом, принадлежащим тому же пользователю, в зависимости от потребностей провайдеров услуг:

1. Провайдер авторизации может требовать от пользователя войти в систему и ввести код сопряжения.
2. Провайдер авторизации может требовать от пользователя входа в систему и подтверждения без ввода кода сопряжения.
3. Провайдер авторизации может автоматически давать токен доступа, не требуя от пользователя никаких действий.

Эти три варианта описаны в следующих подразделах.

7.5.1 Пользователь входит в систему и вводит код сопряжения

Провайдер авторизации может требовать от пользователя войти в систему и ввести код сопряжения, прежде чем дать доступ к новому провайдеру услуг. Процесс получения клиентом токена доступа для данного провайдера услуг показан на **Рис. 3**. В этом случае процесс аналогичен описанному в § 7.3, *User mode*.

1. Клиент запрашивает защищенный ресурс у провайдера услуг, не имея действительного токена для домена провайдера услуг. Провайдер услуг возвращает ответ `401 Unauthorized`, включая заголовок `WWW-Authenticate`, указывающий провайдера авторизации и то, что поддерживается пользовательский режим.
2. Клиент уже зарегистрирован у этого провайдера авторизации и имеет `client_id` и `client_secret`. Клиент передает POST в оконечную точку `/associate` провайдера авторизации с `client_id`, `client_secret` и доменом провайдера услуг в качестве параметров. Провайдер авторизации возвращает клиенту `device_code` с `user_code`, отображаемым устройством пользователю, и `verification_uri`.
3. Используя `device_code`, клиент запрашивает оконечную точку `/token` провайдера авторизации, чтобы проверить, подтверждена ли ассоциация пользователем. Провайдер авторизации возвращает ответ `202 Accepted`, что означает. что пользователь пока не подтвердил ассоциацию с помощью `user_code`.
4. Пользователь идет на `verification_uri`, входит в систему и вводит `user_code`. Теперь провайдер авторизации может создать связь между `client_id` и идентификатором пользователя.
5. Клиент запрашивает оконечную точку `/token` провайдера авторизации и получает ответ `200 OK`, содержащий `access_token` и `user_name`.
6. Теперь клиент может запрашивать защищенный ресурс у провайдера услуг, используя `access_token`. Провайдер услуг проверяет, действителен ли `access_token`, с помощью запроса POST в оконечную точку `/authorized` провайдера авторизации. Если он действителен, провайдер авторизации возвращает провайдеру услуг `200 OK`, включая `client_id` и `user_id`. Провайдер услуг возвращает клиенту защищенный ресурс, возможно, персонализированный с помощью `user_id`.

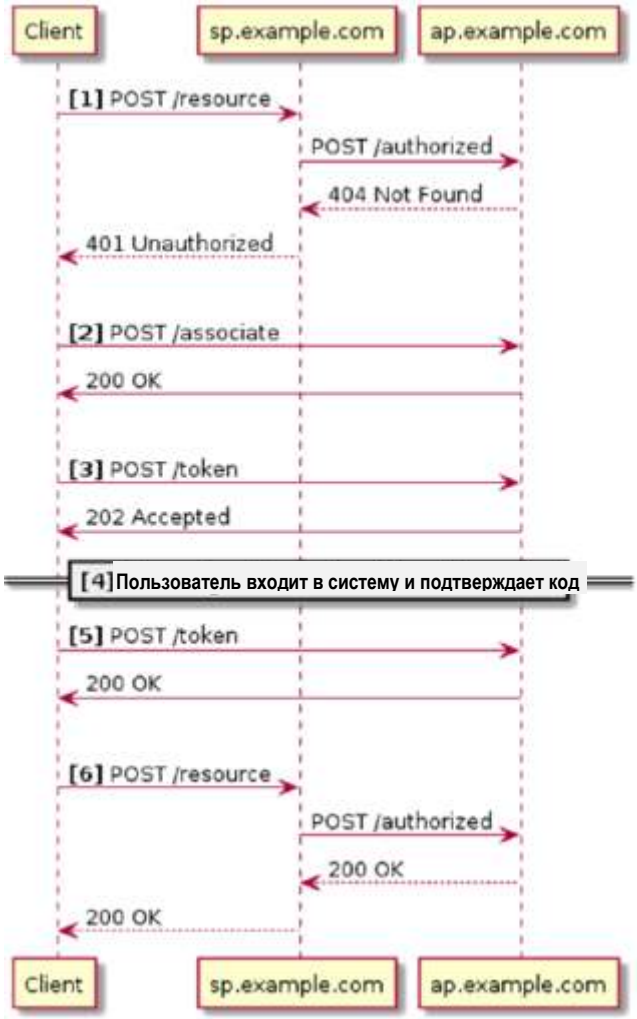

Рис. 3: Выдача токена для существующего клиента с новым провайдером услуг с вводом кода сопряжения

7.5.2 Пользователь входит в систему и дает подтверждение

Провайдер авторизации может требовать от пользователя войти в систему, не вводя код сопряжения, прежде чем дать доступ к новому провайдеру услуг. Процесс получения клиентом токена доступа для данного провайдера услуг показан на **Рис. 4**.

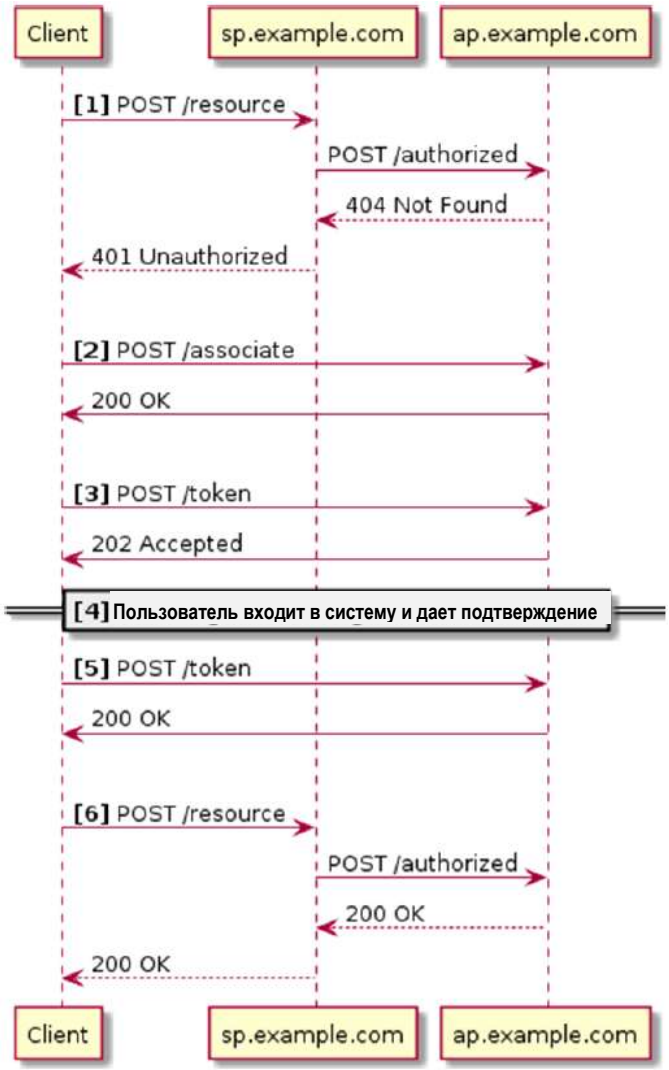

Рис. 4: Выдача токена для существующего клиента с новым провайдером услуг с пользовательским подтверждением

1. Клиент запрашивает защищенный ресурс у провайдера услуг, не имея действительного токена для домена провайдера услуг. Провайдер услуг возвращает ответ `401 Unauthorized`, включая заголовок `WWW-Authenticate`, указывающий провайдера авторизации и то, что поддерживается пользовательский режим.
2. Клиент уже зарегистрирован у этого провайдера авторизации и имеет `client_id` и `client_secret`. Клиент передает POST в оконечную точку `/associate` провайдера авторизации с `client_id`, `client_secret` и доменом провайдера услуг в качестве параметров. Провайдер авторизации возвращает клиенту `device_code`. Обратите внимание, что `user_code` не возвращается.
3. Пока пользователь входит в систему и еще не дал подтверждения, клиент повторяет POST в оконечную точку `/token` провайдера авторизации, передавая `client_id`, `client_secret` и `device_code`. Провайдер авторизации возвращает ответ `202 Accepted`, показывающий, что пользователь пока не дал подтверждения.
4. Пользователь входит в систему провайдера авторизации и дает подтверждение для предоставления доступа своему устройству к новому провайдеру услуг.
5. Клиент передает POST в оконечную точку `/token` провайдера услуг с `client_id`, `client_secret` и `device_code`. Провайдер авторизации возвращает ответ `200 OK`, содержащий `access_token` для запрашиваемого провайдера услуг.
6. Теперь клиент может запрашивать защищенный ресурс у провайдера услуг, используя `access_token`. Провайдер услуг проверяет, действителен ли `access_token`, с помощью запроса POST в оконечную точку `/authorized` провайдера авторизации. Если он действителен, провайдер авторизации возвращает провайдеру услуг `200 OK`, включая `client_id` и `user_id`. Провайдер услуг возвращает клиенту защищенный ресурс, возможно, персонализированный с помощью `user_id`.

7.5.3 Токен выдается автоматически

Провайдер авторизации может автоматически давать токен доступа для нового провайдера услуг, не требуя от пользователя никаких действий.

Процесс получения клиентом токена доступа для данного провайдера услуг показан на **Рис. 5**.

1. Клиент запрашивает защищенный ресурс у провайдера услуг, не имея действительного токена для домена провайдера услуг. Провайдер услуг возвращает ответ `401 Unauthorized`, включая заголовок `WWW-Authenticate`, указывающий провайдера авторизации и то, что поддерживается пользовательский режим.
2. The client has already registered with this authorization provider, so has an existing `client_id` and `client_secret`. The client POSTs to the authorization provider's `/associate` endpoint, passing the `client_id`, `client_secret`, and service provider's domain as parameters. The authorization provider returns a `device_code` to the client.
3. Клиент уже зарегистрирован у этого провайдера авторизации и имеет `client_id` и `client_secret`. Клиент передает POST в оконечную точку `/associate` провайдера авторизации с `client_id`, `client_secret` и доменом провайдера услуг в качестве параметров. Провайдер авторизации возвращает клиенту `device_code`.
4. Клиент передает POST в оконечную точку `/token` провайдера услуг с `client_id`, `client_secret` и `device_code`. Провайдер авторизации возвращает ответ `200 OK`, содержащий `access_token` для запрашиваемого провайдера услуг.
5. Теперь клиент может запрашивать защищенный ресурс у провайдера услуг, используя `access_token`. Провайдер услуг проверяет, действителен ли `access_token`, с помощью запроса POST в оконечную точку `/authorized` провайдера авторизации. Если он действителен, провайдер авторизации возвращает провайдеру услуг `200 OK`, включая `client_id` и `user_id`. Провайдер услуг возвращает клиенту защищенный ресурс, возможно, персонализированный с помощью `user_id`.

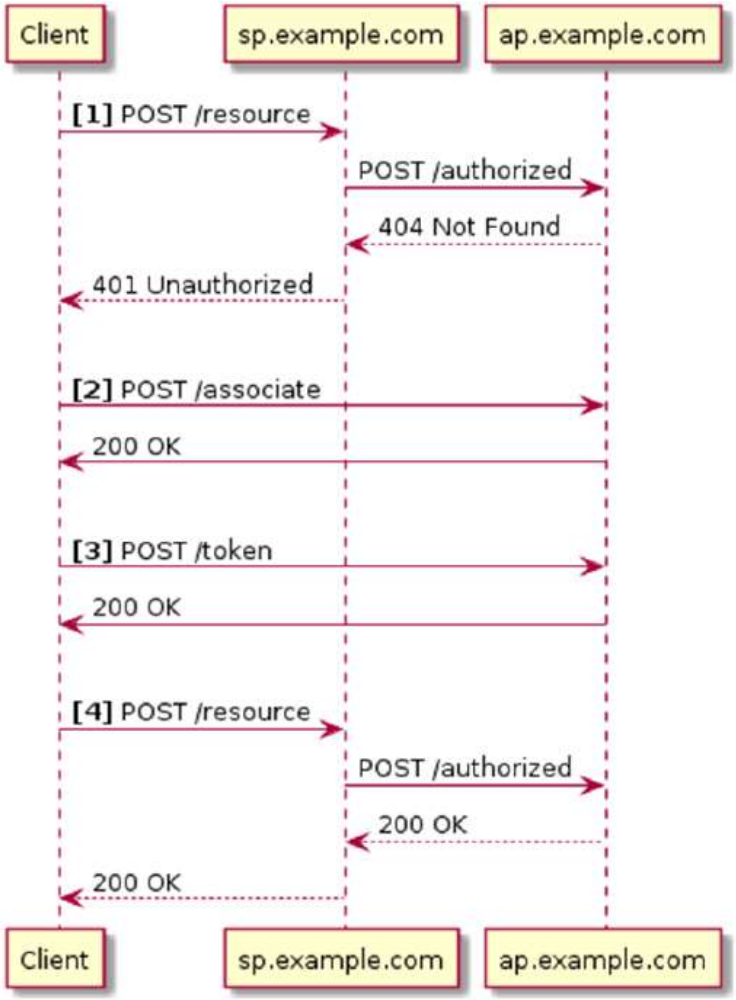

Рис. 5: Автоматическая выдача токена для существующего клиента

7.5.4 Обновление истекшего токена доступа

Провайдеры авторизации МОГУТ выдавать токены доступа, заканчивающиеся через интервал времени. Клиент, использующий истекший токен для доступа к защищенному ресурсу, следует шагам, показанным на **Рис. 6,** для получения нового токена доступа.

1. Клиент запрашивает защищенный ресурс у провайдера услуг, предоставляя имеющийся токен доступа. Путем запроса провайдера авторизации, как описано в § 9, *API провайдера услуг / провайдера авторизации*, провайдер услуг определяет, что токен истек. Тогда провайдер услуг возвращает клиенту ответ HTTP `401 Unauthorized`, включая заголовок `WWW-Authenticate`, как описано в § 7.7.1, *Вызов аутентификации*.
2. Клиент делает запрос в оконечную точку `/token` провайдера авторизации и передает свои `client_id` и `client_secret` и домен провайдера услуг. После подтверждения, что `client_id` и `client_secret` действительны для этого домена, провайдер авторизации выдает этому клиенту новый токен, который заменяет любые токены, ранее выданные этому клиенту.
3. Клиент повторяет запрос на защищенный ресурс, используя новый токен доступа, и теперь достигает цели.

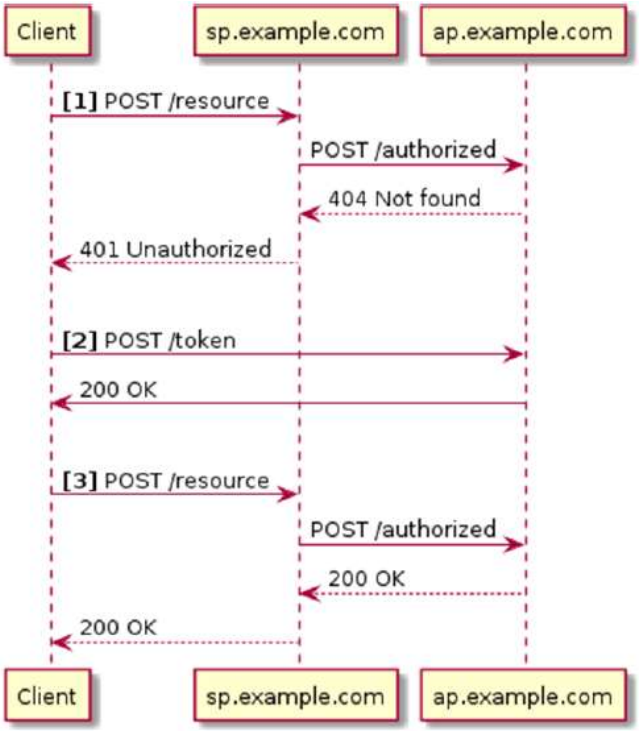

Рис. 6: Обновление истекшего токена доступа

7.6 Удаление ассоциации между клиентом и провайдером авторизации

Протокол CPA дает возможность удаления ассоциации между клиентом и провайдером авторизации либо со стороны клиента, либо со стороны провайдера авторизации. Методы для этого не входят в спецификацию CPA – для удаления ассоциации нет определенной оконечной точки API. Ниже дано краткое описание управления удалением ассоциаций клиентом и провайдером авторизации.

7.6.1 Удаление ассоциации со стороны клиента

Для удаления ассоциации между клиентом и провайдером авторизации со стороны клиента клиент должен удалить `client_id` и `client_secret` для провайдера авторизации и токены, связанные с этим провайдером.

Без токена клиент получит вызов аутентификации при следующем обращении к провайдеру услуг. Если клиент сохранил the `client_id` и `client_secret`, он сможет получить другой токен для ассоциации, следуя пунктам, изложенным в Автоматической выдаче токенов.

Без `client_id` клиент будет выглядеть совершенно незнакомым клиентом, и провайдер услуг инициирует процесс ассоциации, как для нового клиента (что сейчас и происходит).

7.6.2 Удаление ассоциации со стороны провайдера авторизации

Для удаления ассоциации со стороны провайдера авторизации он должен удалить все токены, связанные с `client_id`, и все преобразования между `client_id` и `user_id`.

7.7 Как интегрировать приложения с протоколом CPA

В этом разделе описано, как провайдеры услуг могут информировать клиентов о наличии опции аутентификации с протоколом CPA и как следует использовать токен на предъявителя, полученный через CPA, для доступа к защищенным ресурсам. Этот раздел не нормативный. Однако мы настоятельно рекомендуем по возможности следовать этим условиям для максимизации взаимодействия.

7.7.1 Вызов аутентификации

Если клиент делает запрос на защищенный ресурс без действительного токена на предъявителя провайдеру услуг, то провайдер услуг может либо отклонить запрос (с ответом 4xx), либо вернуть неавторизованную версию ресурса (с ответом 2xx). В любом случае ответ должен включать следующий заголовок `WWW-Authenticate` (см. [5]*, раздел 3)*:

```
WWW-Authenticate: CPA version="1.0"
                  name="Example Authorization Provider"
                  uri="https://ap.example.com/cpa"
                  modes="client,user"
```

Заголовок включает следующие значения:

version
ОБЯЗАТЕЛЬНО. Версия протокола CPA, используемая сервером. Это может быть список, версий разделенный запятыми, если сервер поддерживает множество версий протокола. См. § 7.1, *Версии протокола*.

name
ОБЯЗАТЕЛЬНО. Отображаемое имя провайдера авторизации. Клиенты должны хранить свои `client_id` и `client_secret` по домену, а не по имени.

uri
ОБЯЗАТЕЛЬНО. URI провайдера авторизации. Клиент прикрепляет к этому URI путь к различным оконечным точкам, определенным в данной спецификации, например, `register`, `associate` и `token`.

Примечание: Провайдер авторизации, изменивший значение домена, будет считаться клиентами новым провайдером авторизации, требуя новых client ID и client secret.

modes
ОБЯЗАТЕЛЬНО. Если клиент не предоставил на запрос действительный токен на предъявителя, это значение должно быть списком режимов авторизации (`client` и/или `user`), разделенным запятыми и поддерживаемым провайдером авторизации.

Если клиент the client предоставил на запрос действительный *client mode*, а провайдер авторизации также поддерживает *user mode*, это значение ДОЛЖНО быть "`user`", указывая клиенту, что есть опция подъема до *user mode*.

Затем клиент использует протокол CPA для запроса токена у провайдера авторизации.

7.7.2 Доступ к защищенному ресурсу

Когда клиент получил токен на предъявителя для провайдера услуг, он должен использовать этот токен во всех будущих запросах защищенных ресурсов у провайдера услуг.

Тип токена «на предъявителя» используется путем включения токена в заголовок Authorization в запросе, как описано в [5], раздел 2.1.

```
GET /resource HTTP/1.1
Host: sp.example.com
Authorization: Bearer 28b8caec68ae4a8c89dffaa37d131295
```

8. API клиента / провайдера авторизации

Этот раздел определяет API, предлагаемый клиенту провайдером авторизации. Он состоит из трех оконечных точек:

- `/register` – регистрация клиента
- `/associate` – ассоциация клиента с пользовательским аккаунтом
- `/token` – получение токена на предъявителя

8.1 /register – регистрация клиента

Чтобы зарегистрироваться у провайдера авторизации, клиент делает запрос в оконечную точку регистрации провайдера авторизации, /register. В ответ провайдер авторизации присваивает уникальный идентификатор клиента и соответствующий client secret.

8.1.1 Запрос /register

Клиент делает запрос HTTP POST в оконечную точку регистрации провайдера авторизации и включает следующие параметры в объект JSON в теле запроса:

client_name
ОБЯЗАТЕЛЬНО. Строка, содержащая удобочитаемое имя клиента.

software_id
ОБЯЗАТЕЛЬНО. Строка, содержащая идентификатор программы клиента. Это значение НЕ ДОЛЖНО меняться при смене версии программы. Провайдер авторизации ДОЛЖЕН считать это поле утверждаемым клиентом и НЕ ДОЛЖЕН принимать никаких решений о значении этого поля.

software_version
ОБЯЗАТЕЛЬНО. Идентификатор версии программы, которая содержит клиент. Это значение СЛЕДУЕТ менять при каждом обновлении программы клиента. Провайдер авторизации ДОЛЖЕН считать это поле утверждаемым клиентом и НЕ ДОЛЖЕН принимать никаких решений о значении этого поля.

Пример запроса

```
POST /register HTTP/1.1
Host: ap.example.com
Content-Type: application/json

{
    "client_name": "Test client",
    "software_id": "cpa-test-client",
    "software_version": "1.0.0"
}
```

8.1.2 Ответ /register

Если регистрация успешна, провайдер авторизации генерирует новый идентификатор клиента. Провайдер авторизации отвечает HTTP со статусом 201 и включает в объект JSON в теле ответа следующую информацию:

client_id
ОБЯЗАТЕЛЬНО. Строка, содержащая уникальный идентификатор, выданный клиенту в процессе регистрации. Этот идентификатор клиента ДОЛЖЕН быть уникальным в сервере и НЕ ДОЛЖЕН использоваться любым другим клиентом.

client_secret
ОБЯЗАТЕЛЬНО. Строка, содержащая client secret, который ДОЛЖЕН быть уникальным для каждого client_id. Это значение используется клиентами во всех последующих запросах провайдеру авторизации. Клиенты НЕ ДОЛЖНЫ включать client secret в запросы провайдерам услуг.

Пример ответа

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "client_id": "1234",
    "client_secret": "sdalfqealskdfnk13984r2n23klndvs"
}
```

Если происходит ошибка, провайдер авторизации отвечает HTTP со статусом 400 и включает в объект JSON в теле ответа следующую информацию

error
ОБЯЗАТЕЛЬНО. Одно из следующих значений ошибки:

`invalid_request`

Один или более требуемых значений параметров отсутствует или недействителен.

Пример ответа об ошибке

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "invalid_request"
}
```

8.2 /associate – ассоциация клиента с пользовательским аккаунтом

Для ассоциации клиента с пользовательским аккаунтом клиент сначала делает запрос в оконечную точку провайдера авторизации `/associate`. В ответ провайдер авторизации присваивает проверочный код пользователя и возвращает его клиенту вместе с URI, который пользователь должен посетить для своей аутентификации и ввести код пользователя для связи своего клиента со своим аккаунтом.

8.2.1 Запрос /associate

Клиент делает запрос HTTP POST в оконечную точку регистрации провайдера авторизации и включает следующие параметры в объект JSON в теле запроса:

client_id
ОБЯЗАТЕЛЬНО. Идентификатор клиента, выданный клиенту при регистрации у провайдера авторизации. См. § 8.1, *Регистрация клиента*.

client_secret
ОБЯЗАТЕЛЬНО. Секрет клиента, выданный клиенту при регистрации у провайдера авторизации. См. § 8.1, *Регистрация клиента*.

domain
ОБЯЗАТЕЛЬНО. Имя домена провайдера услуг (которое может включать номер порта).

ПРИМЕЧАНИЕ: Токен доступа действителен только для домена, указанного в этом запросе. Клиенты не должны посылать токены доступа в другие домены.

Пример запроса

```
POST /associate HTTP/1.1
Host: ap.example.com
Content-Type: application/json

{
    "client_id": "1234",
    "client_secret": "sdalfqealskdfnk13984r2n23klndvs",
    "domain": "sp.example.com"
}
```

8.2.2 Ответ /associate

Есть всего три возможных успешных результатов, как говорилось в § 7.5, *Автоматическая выдача токена*:

1. Клиент пока не ассоциирован с идентификатором пользователя.
2. Клиент уже ассоциирован с идентификатором пользователя, но провайдер авторизации требует пользовательского подтверждения, прежде чем дать доступ к новому провайдеру услуг.
3. Клиент уже ассоциирован с идентификатором пользователя и провайдер авторизации может автоматически дать доступ к новому провайдеру услуг, не требуя пользовательского подтверждения.

В следующих подразделах описан ответ на запрос `/associate` в каждом из трех случаев.

8.2.2.1 Связь клиента с пользовательским аккаунтом

Если регистрация успешна, провайдер авторизации генерирует новый код устройства для клиента. Провайдер авторизации отвечает HTTP со статусом 200 и включает в объект JSON в теле ответа следующую информацию:

device_code
ОБЯЗАТЕЛЬНО. Временный проверочный код устройства в формате UUID согласно [3].

user_code
ОБЯЗАТЕЛЬНО. Временный проверочный код пользователя. Это строка из 8 символов. Значение ДОЛЖНО использовать только буквенно-числовые символы из ISO-646 Invariant Code Set. Оно ДОЛЖНО быть уникальным у провайдера авторизации в течение ожидания обработки запроса на ассоциацию.

verification_uri
ОБЯЗАТЕЛЬНО. Проверочный URI конечного пользователя у провайдера авторизации. Этот URI должен отображаться клиентом пользователю. URI ДОЛЖЕН быть коротким, чтобы отображаться клиентским устройством (см. § 2.1, *Ограниченный ввод и отображение*), и потому, что пользователю придется вручную вводить его в свой пользовательский агент.

interval
ОБЯЗАТЕЛЬНО. Целое значение в числовом формате JSON, которое показывает минимальное время ожидания клиента между запросами в оконечную точку `/token`, в секундах.

expires_in
ОБЯЗАТЕЛЬНО. Целое значение в числовом формате JSON, которое показывает максимальное количество времени, когда `device_code` и `user_code` остаются действительными, в секундах, с момента выдачи ответа.

Во избежание кэширования кода устройства провайдер авторизации ДОЛЖЕН включать в ответ следующие заголовки HTTP: `Cache-Control: no-store` и `Pragma: no-cache`.

Пример ответа

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "device_code": "197bf88c-749a-42e2-93f0-e206bac2252f",
    "user_code": "AbfZDgJr",
    "verification_uri": "https://ap.example.com/verify",
    "interval": 5,
    "expires_in": 1800
}
```

Клиентское устройство обычно использует в этом ответе информацию для показа сообщения типа:

> To associate your device, please visit https://ap.example.com/verify and, when prompted, enter the code AbfZDgJr

или:

> Enter "AbfZDgJr"
> @ bit.ly/cpa123

Управление процессом верификации не входит в рамки данной спецификации. Однако ожидается, что когда пользователь зайдет на URI, указанный в `verification_uri`, ему придется войти в систему в процессе ввода `user_code` (и таким образом удостоверить свою идентичность). Затем `user_code` и идентификатор пользователя are передаются провайдеру авторизации, чтобы он мог сопоставить `user_code` с ожидающим запросом на ассоциацию и преобразовать `client_id` в идентификатор пользователя.

8.2.2.2 Требование подтверждения перед предоставлением доступа к новой услуге

Если клиент уже ассоциирован с пользовательским аккаунтом, и этому аккаунту дан доступ к провайдеру услуг, входящему в группу провайдеров услуг у провайдера авторизации, то когда этот клиент хочет получить токен для доступа к другому провайдеру услуг в той же группе, провайдер авторизации может потребовать от пользователя подтвердить доступ, прежде чем выдать токен. Это может

использоваться, например, для того, чтобы попросить пользователя принять условия провайдера услуг. В этом случае провайдер авторизации отвечает HTTP со статусом 200 и включает в объект JSON в теле ответа следующую информацию:

device_code
ОБЯЗАТЕЛЬНО. Временный проверочный код устройства в формате UUID согласно [3]. ДОЛЖЕН быть уникальным у провайдера авторизации в течение ожидания обработки запроса на ассоциацию.

verification_uri
ОБЯЗАТЕЛЬНО. Проверочный URI конечного пользователя у провайдера авторизации. Этот URI должен отображаться клиентом пользователю.

URI должен быть коротким и легко запоминаемым, т.к. конечным пользователям придется вручную вводить его в свой пользовательский агент.

interval
ОБЯЗАТЕЛЬНО. Целое значение в числовом формате JSON, которое показывает минимальное время ожидания клиента между запросами в оконечную точку `/token`, в секундах.

expires_in
ОБЯЗАТЕЛЬНО. Целое значение в числовом формате JSON, которое показывает максимальное количество времени, когда `device_code` остается действительными, в секундах, с момента выдачи ответа.

Во избежание кэширования кода устройства провайдер авторизации ДОЛЖЕН включать в ответ следующие заголовки HTTP: `Cache-Control: no-store` и `Pragma: no-cache`.

Пример ответа

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "device_code": "197bf88c-749a-42e2-93f0-e206bac2252f",
    "verification_uri": "https://ap.example.com/verify",
    "interval": 5,
    "expires_in": 1800
}
```

8.2.2.3 Доступ автоматически предоставляется новой услуге

Если клиент уже ассоциирован с пользовательским аккаунтом, то для единой регистрации пользователя провайдер авторизации может автоматически давать доступ к новому провайдеру услуг, не требуя пользовательского подтверждения. В этом случае провайдер авторизации отвечает HTTP со статусом 200 и включает в объект JSON в теле ответа следующую информацию:

device_code
ОБЯЗАТЕЛЬНО. Временный проверочный код устройства в формате UUID согласно [3]. ДОЛЖЕН быть уникальным у провайдера авторизации в течение ожидания обработки запроса на ассоциацию.

expires_in
ОБЯЗАТЕЛЬНО. Целое значение в числовом формате JSON, которое показывает количество времени, когда `device_code` остается действительными, в секундах, с момента приема.

Во избежание кэширования кода устройства провайдер авторизации ДОЛЖЕН включать в ответ следующие заголовки HTTP: `Cache-Control: no-store` и `Pragma: no-cache`.

Пример ответа

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "device_code": "197bf88c-749a-42e2-93f0-e206bac2252f",
    "expires_in": 1800
}
```

Если происходит ошибка, провайдер авторизации отвечает HTTP со статусом 400 и включает в объект JSON в теле ответа следующую информацию

error
ОБЯЗАТЕЛЬНО. Одно из следующих значений ошибки:

`invalid_request`

Один или более требуемых значений параметров отсутствует или недействителен.

`invalid_client`

`client_id` и/или `client_secret` недействительно.

Пример ответа об ошибке

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "invalid_client"
}
```

8.3 /token – получение токена на предъявителя

Для получения токена доступа клиент делает запрос в оконечную точку токена провайдера авторизации, `/token`. Параметры и из содержание зависят от режима клиента – *client mode* или *user mode*.

В *client mode,* поскольку провайдер авторизации не требует никаких дальнейших действий со стороны пользователя, провайдер авторизации может автоматически выдать токен.

В *user mode* клиент повторно запрашивает провайдера авторизации, пока пользователь не введет корректный проверочный код пользователя и либо даст, либо не даст доступ к клиенту, либо пока не истечет срок пользовательского кода.

Клиент делает следующий запрос с произвольным, но разумным интервалом, который НЕ ДОЛЖЕН превышать минимальный интервал, указанный провайдером авторизации в ответе `/associate`.

8.3.1 Запрос /token

8.3.1.1 Client mode

Клиент делает запрос HTTP POST в оконечную точку токена провайдера авторизации и включает следующие параметры в объект JSON в теле запроса:

grant_type
ОБЯЗАТЕЛЬНО. Это значение должно быть строкой "`http://tech.ebu.ch/cpa/1.0/client_credentials`".

client_id
ОБЯЗАТЕЛЬНО. Идентификатор клиента, выданный клиенту при регистрации у провайдера авторизации. См. § 8.1, *Регистрация клиента*.

client_secret
ОБЯЗАТЕЛЬНО. Секрет клиента, выданный клиенту при регистрации у провайдера авторизации. См. § 8.1, *Регистрация клиента*.

domain
ОБЯЗАТЕЛЬНО. Имя домена провайдера услуг (которое может включать номер порта).

Пример запроса

```
POST /token HTTP/1.1
Host: ap.example.com
Content-Type: application/json

{
    "grant_type": "http://tech.ebu.ch/cpa/1.0/client_credentials",
    "client_id": "1234",
    "client_secret": "sdalfqealskdfnk13984r2n23klndvs",
    "domain": "sp.example.com"
}
```

8.3.1.2 User mode

Клиент делает запрос HTTP POST в оконечную точку токена провайдера авторизации с произвольным, но разумным интервалом, который НЕ ДОЛЖЕН превышать минимальный интервал, указанный провайдером авторизации в ответе `/associate`. Ответ включает следующие параметры в объект JSON в теле запроса:

grant_type
ОБЯЗАТЕЛЬНО. Это значение должно быть строкой "`http://tech.ebu.ch/cpa/1.0/device_code`".

device_code
ОБЯЗАТЕЛЬНО. Временный проверочный код устройства, возвращенный в ответ на `/associate`. Клиент должен сбросить `device_code` после успешного ответа на этот вызов. Провайдер авторизации ДОЛЖЕН объявить недействительным `device_code` после его обмена на токен доступа.

client_id
ОБЯЗАТЕЛЬНО. Идентификатор клиента, выданный клиенту при регистрации у провайдера авторизации. См. § 8.1, *Регистрация клиента*.

client_secret
ОБЯЗАТЕЛЬНО. Секрет клиента, выданный клиенту при регистрации у провайдера авторизации. См. § 8.1, *Регистрация клиента*.

domain
ОБЯЗАТЕЛЬНО. Имя домена провайдера услуг (которое может включать номер порта).

Пример запроса

```
POST /token HTTP/1.1
Host: ap.example.com
Content-Type: application/json

{
    "grant_type": "http://tech.ebu.ch/cpa/1.0/device_code",
    "device_code": "197bf88c-749a-42e2-93f0-e206bac2252f",
    "client_id": "1234",
    "client_secret": "sdalfqealskdfnk13984r2n23klndvs",
    "domain": "sp.example.com"
}
```

8.3.1.3 Обновление истекшего токена доступа

Для замены истекшего токена доступа клиент делает запрос HTTP POST в оконечную точку токена провайдера авторизации и включает следующие параметры в объект JSON в теле запроса:

grant_type
ОБЯЗАТЕЛЬНО. Это значение должно быть строкой
"`http://tech.ebu.ch/cpa/1.0/client_credentials`".

client_id
ОБЯЗАТЕЛЬНО. Идентификатор клиента, выданный клиенту при регистрации у провайдера авторизации. См. § 8.1, *Регистрация клиента*.

client_secret
ОБЯЗАТЕЛЬНО. Секрет клиента, выданный клиенту при регистрации у провайдера авторизации. См. § 8.1, *Регистрация клиента*.

domain
ОБЯЗАТЕЛЬНО. Имя домена провайдера услуг (которое может включать номер порта).

Пример запроса

```
POST /token HTTP/1.1
Host: ap.example.com
Content-Type: application/json

{
    "grant_type": "http://tech.ebu.ch/cpa/1.0/client_credentials",
    "client_id": "1234",
    "client_secret": "sdalfqealskdfnk13984r2n23klndvs",
    "domain": "sp.example.com"
}
```

8.3.2 Ответ /token

В случае положительного результата провайдер авторизации генерирует новый токен доступа для клиента и объявляет недействительным любой существующий токен этого клиента для данного домена провайдера услуг. Провайдер авторизации отвечает HTTP со статусом 200 и включает следующую информацию в объект JSON в теле ответа:

user_name
Если клиент зарегистрирован в client mode, это поле не включается, т.к. клиент не ассоциирован с пользовательским аккаунтом. В user mode это поле ДОЛЖНО быть включено и содержит имя пользователя, если оно есть. Если имени нет, провайдер авторизации должен вернуть пустую строку. Это значение предназначено для отображения в клиентском устройстве.

access_token
ОБЯЗАТЕЛЬНО. Токен доступа.

token_type
ОБЯЗАТЕЛЬНО. Должно быть строкой "`bearer`".

domain_name
ОБЯЗАТЕЛЬНО. Строка, содержащая имя провайдера услуг, подходящее для отображения в клиентском устройстве.

expires_in
РЕКОМЕНДУЕТСЯ. Если есть, то целое число в числовом формате JSON, которое показывает срок действия токена доступа, в секундах, с момента выдачи ответа. Если нет, токен доступа не истек.

Во избежание кэширования токена доступа провайдер авторизации ДОЛЖЕН включать в ответ следующие заголовки HTTP: `Cache-Control`: `no-store` и `Pragma`: `no-cache`.

Пример ответа

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "user_name": "Alice",
    "access_token": "28b8caec68ae4a8c89dffaa37d131295",
    "token_type": "bearer",
    "domain_name": "Channel 1"
}
```

Если пользователь еще не ввел корректно проверочный код пользователя и/или не получил доступ, и проверочный код пользователя не истек, провайдер авторизации отвечает HTTP со статусом 202 и включает следующую информацию в объект JSON в теле ответа:

reason
ОБЯЗАТЕЛЬНО. Статус ассоциации. Это значение должно быть строкой `authorization_pending`", которая показывает, что запрос на авторизацию все еще ожидает обработки, т.к. конечный пользователь еще не посетил провайдера авторизации и не ввел свой проверочный код.

Пример ответа

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "reason": "authorization_pending"
}
```

Если происходит ошибка, провайдер авторизации отвечает HTTP со статусом 400 и включает в объект JSON в теле ответа следующую информацию

error
ОБЯЗАТЕЛЬНО. Одно из следующих значений ошибки:

`invalid_request`

Один или более требуемых значений параметров отсутствует или недействителен.

`invalid_client`

`client_id` и/или `client_secret` недействительно.

`slow_down`

Клиент превысил лимит ожидания обработки, определенный параметром интервала в ответе `/associate`. Провайдер авторизации МОЖЕТ включить в ответ поле `retry_in` для сообщения количества времени, которое клиент должен ждать до повтора запроса.

expired
Время, данное пользователю на проверку окончания срока действия ассоциации.

cancelled
Провайдер авторизации хочет отменить процесс связи. Это бывает, например, когда пользователь отказался ассоциировать клиента со своим пользовательским аккаунтом, например, не приняв условия, представленные провайдером авторизации.

retry_in
ОПЦИОНАЛЬНО. При значении ошибки slow_down это поле МОЖНО включить в ответ. Если оно включено, то значение ДОЛЖНО быть числом в числовом формате JSON, которое показывает количество времени, которое клиент должен ждать до повтора запроса, в секундах с момента выдачи ответа.

Пример ответа об ошибке

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "slow_down",
    "retry_in": 10
}
```

9. API провайдера услуг / провайдера авторизации

Этот раздел определяет API, предлагаемый провайдером авторизации провайдеру услуг. Он состоит из единой оконечной точки:

- `/authorized` – проверка токена доступа и возвращение идентификаторов клиента и пользователя

9.1 Создание доверия между провайдером услуг и провайдером авторизации

Предполагается, что между провайдером услуг и провайдером авторизации устанавливаются доверительные отношения, в которых провайдер авторизации присваивает уникальный токен доступа провайдеру услуг, чтобы тот мог делать удостоверенные HTTP запросы провайдеру авторизации.

Способ создания таких отношений и передачи токена доступа провайдеру услуг не входит в рамки данной спецификации.

9.2 /authorized – оконечная точка верификации токена доступа

Оконечная точка верификации токена доступа позволяет провайдеру услуг проверить токен доступа у провайдера авторизации. Эта точка ДОЛЖНА иметь путь URL `/authorized`.

9.2.1 Запрос /authorized

Провайдер услуг делает запрос HTTP POST в оконечную точку верификации токена доступа провайдера авторизации и включает следующие параметры в объект JSON в теле запроса:

access_token
ОБЯЗАТЕЛЬНО. Токен доступа для верификации.

domain
ОБЯЗАТЕЛЬНО. Имя домена провайдера услуг (которое может включать номер порта).

Для аутентификации у провайдера авторизации провайдер услуг включает токен доступа, выданный ему провайдером авторизации, в заголовок Authorization в запросе, как описано в [5], п.2.1.

Пример запроса

```
POST /authorized HTTP/1.1
Host: ap.example.com
Content-Type: application/json
Authorization: Bearer 3028bad800a94789a4b54202123d500a

{
    "access_token": "28b8caec68ae4a8c89dffaa37d131295",
    "domain": "sp.example.com"
}
```

9.2.2 Ответ /authorized

Если токен доступа принадлежит зарегистрированному клиенту, провайдер авторизации отвечает HTTP со статусом 200 и включает следующую информацию в объект JSON в теле ответа:

client_id
ОБЯЗАТЕЛЬНО. Строка, содержащая уникальный идентификатор клиента, который владеет выданным токеном доступа.

user_id
Если клиент ассоциирован с пользовательским аккаунтом, провайдер авторизации ДОЛЖЕН вернуть строку, содержащую уникальный идентификатор пользователя. Если клиент не ассоциирован с пользовательским аккаунтом, провайдер авторизации ДОЛЖЕН исключить поле `user_id` из ответа.

Пример ответа

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "client_id": "1234",
    "user_id": "1"
}
```

Если одно или более требуемых значений параметров отсутствует или недействительно, провайдер авторизации отвечает HTTP со статусом 400 и включает следующую информацию в объект JSON в теле ответа:

error
ОБЯЗАТЕЛЬНО. Это значение должно быть строкой "`invalid_request`".

Пример ответа об ошибке

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "invalid_request"
}
```

Если токен неизвестен провайдеру авторизации, не ассоциирован с доменом провайдера услуг, указанным в параметре domain в запросе, или если токен истек, провайдер авторизации отвечает HTTP со статусом 404 и включает следующую информацию в объект JSON в теле ответа:

error
ОБЯЗАТЕЛЬНО. Это значение должно быть строкой "`not_found`"

Пример ответа об ошибке

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
    "error": "not_found"
}
```

Если провайдер услуг не авторизован для запросов провайдеру авторизации, провайдер авторизации отвечает HTTP со статусом 401 и включает следующую информацию в объект JSON в теле ответа:

error
ОБЯЗАТЕЛЬНО. Это значение должно быть строкой "`unauthorized`"

Пример ответа об ошибке

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json

{
    "error": "unauthorized"
}
```

10. Ссылки

10.1 Нормативные ссылки

[1] **RFC2119** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", Март 1997.

[2] **RFC2818** E. Rescorla, "HTTP Over TLS", Май 2000.

[3] **RFC4122** P. Leach, M. Mealling, R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", Июль 2005.

10.2 Информативные ссылки

[4] **RFC6749** D. Hardt, Ed., "The OAuth 2.0 Authorization Framework", Октябрь 2010.

[5] **RFC6750** M. Jones, D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", Октябрь 2012.

[6] **draft-ietf-oauth-dyn-reg-14** J. Richer, Ed., "OAuth 2.0 Dynamic Client Registration Protocol", 29 Июль 2013.

[7] **draft-recordon-oauth-v2-device-00** D. Recordon, Ed., "OAuth 2.0 Device Profile", Июль 2010.