

OAuth WG	G. Fletcher
Internet-Draft	AOL
Intended status: Informational	T. Lodderstedt
Expires: January 12, 2012	Deutsche Telekom AG
	Z. Zeltsan
	Alcatel-Lucent
	July 11, 2011

OAuth Use Cases

draft-zeltsan-oauth-use-cases-02

[Abstract](#)

This document lists the OAuth use cases. The document's objective is to identify the use cases that will be a base for deriving the OAuth requirements. The provided list is based on the Internet-Drafts of the OAUTH working group and discussions on the group's mailing list.

[Status of this Memo](#)

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet- Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 12, 2012.

[Copyright Notice](#)

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

[Table of Contents](#)

- *1. [Introduction](#)

- *2. [Notational Conventions](#)
- *3. [OAuth use cases](#)
 - *3.1. [Web server](#)
 - *3.2. [User-agent](#)
 - *3.3. [In-App-Payment \(based on Native Application\)](#)
 - *3.4. [Native Application](#)
 - *3.5. [Device](#)
 - *3.6. [Client password \(shared secret\) credentials](#)
 - *3.7. [Assertion](#)
 - *3.8. [Content manager](#)
 - *3.9. [Access token exchange](#)
 - *3.10. [Multiple access tokens](#)
 - *3.11. [Gateway for browser-based VoIP applets](#)
 - *3.12. [Signed Messages](#)
 - *3.13. [Signature with asymmetric secret](#)
- *4. [Authors of the use cases](#)
- *5. [Security considerations](#)
- *6. [IANA considerations](#)
- *7. [Acknowledgements](#)
- *8. [References](#)
- *[Authors' Addresses](#)

1. Introduction

The need for documenting the OAuth use cases was discussed at the oauth WG virtual meetings, on the group's mailing list, and at the IETF 77 and IETF 78. This Internet-Draft describes such use cases. The objective of the draft is to initiate discussion that will lead to defining a set of the use cases that the OAuth specifications should support. The following section provides the abbreviated descriptions of the use cases.

Note: The use of the string ".example.com" in the URLs of the example entities does not mean that the entities belong to the same organization.

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#).

3. OAuth use cases

This section lists the use cases that have been discussed by the oauth WG.

3.1. Web server

Description:

Alice accesses an application running on a web server at `www.printphotos.example.com` and instructs it to print her photographs that are stored on a server `www.storephotos.example.com`. The application at `www.printphotos.example.com` receives Alice's authorization for accessing her photographs without learning her authentication credentials with `www.storephotos.example.com`.

Pre-conditions:

- *Alice has registered with `www.storephotos.example.com` to enable authentication

- *The application at `www.printphotos.example.com` has established authentication credentials with the application at `www.storephotos.example.com`

Post-conditions:

A successful procedure results in the application `www.printphotos.example.com` receiving an authorization code from `www.storephotos.example.com`. The code is bound to the application at `www.printphotos.example.com` and to the callback URL supplied by the application. The application at `www.printphotos.example.com` uses the authorization code for obtaining an access token from `www.storephotos.example.com`. The application at `www.storephotos.example.com` issues an access token after authenticating the application at `www.printphotos.example.com` and validating the authorization code that it has submitted. The application at `www.printphotos.example.com` uses the access token for getting access to Alice's photographs at `www.storephotos.example.com`.

Note: When an access token expires, the service at `www.printphotos.example.com` needs to repeat the OAuth procedure for getting Alice's authorization to access her photographs at `www.storephotos.example.com`. Alternatively, if Alice wants to grant the application a long lasting access to her resources at `www.storephotos.example.com`, the authorization server associated with `www.storephotos.example.com` may issue the long-living tokens. Those tokens can be exchanged for short-living access tokens required to access `www.storephotos.example.com`.

Requirements:

- *The server `www.printphotos.example.com`, which hosts an OAuth client, must be capable of issuing the HTTP redirect requests to Alice's user agent - a browser
- *Application at `www.storephotos.example.com` must be able to authenticate Alice. The authentication method is not in the OAuth scope
- *Application at `www.storephotos.example.com` must obtain Alice's authorization of the access to her photos by `www.printphotos.example.com`
- *Application at `www.storephotos.example.com` may identify to Alice the scope of access that `www.printphotos.example.com` has requested while asking for Alice's authorization
- *Application at `www.storephotos.example.com` must be able to authenticate the application at `www.printphotos.example.com` and validate the authorization code before issuing an access token. The OAuth 2.0 protocol [\[I-D.draft-ietf-oauth-v2\]](#) specifies one authentication method that MAY be used for such authentication - Client Password Authentication.
- *Application at `www.printphotos.example.com` must provide a callback URL to the application at `www.storephotos.example.com` (note: the URL can be pre-registered with `www.storephotos.example.com`)
- *Application at `www.storephotos.example.com` is required to maintain a record that associates the authorization code with the application at `www.printphotos.example.com` and the callback URL provided by the application
- *Access tokens are bearer's tokens (they are not associated with a specific application, such as `www.printphotos.example.com`) and should have a short lifespan

*Application at `www.storephotos.example.com` must invalidate the authorization code after its first use

*Alice's manual involvement in the OAuth authorization procedure (e.g., entering an URL or a password) should not be required. (Alice's authentication to `www.storephotos.example.com` is not in the OAuth scope. Her registration with `www.storephotos.example.com` is required as a pre-condition)

3.2. User-agent

Description:

Alice has installed on her computer a gaming application. She keeps her scores in a database of a social site at `www.fun.example.com`. In order to upload Alice's scores, the application gets access to the database with her authorization.

Pre-conditions:

*Alice has installed a gaming application implemented in a scripting language (e.g., JavaScript) that runs in her browser and uses OAuth for accessing a social site at `www.fun.example.com`

*There is no a web site supporting this application and capable of handling the OAuth flow, so the gaming application needs to update the database itself

*The installed application is registered with the social site at `www.fun.example.com` and has an identifier

*Alice has registered with `www.fun.example.com` for identification and authentication

*An auxiliary web server at `www.help.example.com` is reachable by Alice's browser and capable of providing a script that extracts an access token from an URL's fragment

Post-conditions:

A successful procedure results in Alice's browser receiving an access token. The access token is received from `www.fun.example.com` as a fragment of a redirection URL of an auxiliary web server `www.help.example.com`. Alice's browser follows the redirection, but retains the fragment. From the auxiliary web server at `www.help.example.com` Alice's browser downloads a script that extracts access token from the fragment and makes it available to the gaming application. The application uses the access token to gain access to Alice's data at `www.fun.example.com`.

Requirements:

- *Registration of the application running in the Alice's browser with the application running on `www.fun.example.com` is required for identification
- *Alice's authentication with `www.fun.example.com` is required
- *Application running at `www.fun.example.com` must be able to describe to Alice the request made by the gaming application running on her computer and obtain Alice's authorization for or denial of the requested access
- *After obtaining Alice's authorization the application running at `www.fun.example.com` must respond with an access token and redirect Alice's browser to a web server (e.g., `www.help.example.com`) that is capable of retrieving an access token from an URL

3.3. In-App-Payment (based on Native Application)

Description:

Alice has installed on her computer a gaming application (e.g., running as native code or as a widget). At some point she wants to play the next level of the game and needs to purchase an access to the advanced version of the game from her service provider at `www.sp.example.com`. With Alice's authorization the application accesses her account at `www.sp.example.com` and enables her to make the payment.

Pre-conditions:

- *Alice has registered and has an account with her service provider at `www.sp.example.com`
- *The application is registered with the service provider at `www.sp.example.com`. This enables the server provider to provide Alice with all necessary information about the gaming application (including the information about the purchasing price)
- *Alice has a Web user-agent (e.g., a browser or a widget runtime) installed on her computer

Post-conditions:

A successful procedure results in the gaming application invoking the user browser and directing it to the authorization server of the service provider. The HTTP message includes information about the gaming application's request to access Alice's account. The authorization server presents to Alice the authentication and

authorization interfaces. The authorization interface shows Alice the information about the application's request including the requested charge to her account. After Alice successfully authenticates and authorizes the request, the authorization server enables Alice to save the transaction details including the authorization code issued for the gaming application. Then the authorization server redirects Alice's browser to a custom scheme URI (registered with the operating system). This redirection request contains a one-time authorization code and invokes a special application that is able to extract the authorization code and present it to the gaming application. The gaming application presents the authorization code to the authorization server and exchanges it for a one-time access token. The gaming application then uses the access token to get access to Alice's account and post the charges at www.sp.example.com.

Requirements:

- *An authorization server associated with the server at www.sp.example.com must be able to authenticate Alice over a secure transport
- *An authorization server associated with the server at www.sp.example.com must be able to provide Alice with information about the access request that the gaming application has made (including the amount that is to be charged to her account with the service provider, and the purpose for the charge) over a secure transport
- *An authorization server associated with the server at www.sp.example.com must be able to obtain Alice's authorization decision on the request over a secure transport
- *An authorization server associated with the server at www.sp.example.com must be able to generate on demand a one-time authorization code and a one-time access token according to the scope authorized by Alice
- *An authorization server associated with the server at www.sp.example.com must be able to call back to the gaming application with the authorization result over a secure transport
- *An authorization server associated with the server at www.sp.example.com must enable the gaming application to exchange an authorization code for an access token over a secure transport
- ** An authorization server associated with the server at www.sp.example.com must verify the authorization code and invalidate it after its first use

- ** An authorization server associated with the server at `www.sp.example.com` must enable Alice to save the details of the requested transaction, including the authorization code
- ** An authorization server associated with the server at `www.sp.example.com` must keep a record linking the requested transaction with the authorization code and the respective access token
- ** An authorization server associated with the server at `www.sp.example.com` must enable the resource server `www.sp.example.com` to obtain the transaction information that is linked to the issued access token
- ** Resource server at `www.sp.example.com` must verify access token and invalidate it after its first use
- ** A resource server at `www.sp.example.com` must enable the gaming application to post charges to Alice's account according to the access token presented over a secure transport
- *The gaming application must provide a custom scheme URI to the authorization server associated with `www.sp.example.com` (note: it can be preregistered with the authorization server)
- *Alice's manual involvement in the OAuth authorization procedure (e.g., entering an URL or a password) should not be required. (Alice's authentication to `www.sp.example.com` is not in the OAuth scope)
- * The requirements denoted by '*' are not common for the Native Application use cases, but are specific to the In-App-Payment use case

3.4. Native Application

Description:

Alice wants to upload (or download) her photographs to (or from) `storephotos.example.com` using her smartphone. She downloads and installs a photo app on her smartphone. In order to enable the app to access her photographs, Alice needs to authorize the app to access the web site on her behalf. The authorization shall be valid for a prolonged duration (e.g. several months), so that Alice does not need to authenticate and authorize access on every execution of the app. It shall be possible to withdraw the app's authorization both on the smartphone as well as on the site `storephotos.example.com`.

Pre-conditions:

- *Alice has installed a (native) photo app application on her smartphone
- *The installed application is registered with the social site at storephotos.example.com and has an identifier
- *Alice holds an account with storephotos.example.com
- *Authentication and authorization shall be performed in an interactive, browser-based process. The smartphone's browser is used for authenticating Alice and for enabling her to authorize the request by the Mobile App

Post-conditions:

A successful procedure results in Alice's app receiving an access and a refresh token. The app may obtain the tokens by utilizing either the web server or the user agent flow. The application uses the access token to gain access to Alice's data at storephotos.example.com. The refresh token is persistently stored on the device for use in subsequent app executions. If a refresh token exists on app startup, the app directly uses the refresh token to obtain a new access token.

Requirements:

- *Alice's authentication with storephotos.example.com is required
- *Registration of the application running on Alice's smartphone is required for identification and registration and may be carried out on a per installation base
- *The application at storephotos.example.com provides a capability to view and delete the apps' authorizations. This implies that the different installations of the same app on the different devices can be distinguished (e.g., by a device name or a telephone number)
- *The app must provide Alice an option to logout. The logout must result in the revocation of the refresh token on the authorization server

3.5. Device

Description:

Alice has a device, such as a game console, that does not support an easy data-entry method. She also has an access to a computer with a browser. The application running on the Alice's device gets authorized

access to a protected resource (e.g., photographs) stored on a server at `www.storephotos.example.com`

Pre-conditions:

- *Alice uses an OAuth-enabled game console, which does not have an easy data-entry method, for accessing her photographs at `www.storephotos.example.com`. The device starts the OAuth procedure by requesting a token
- *Alice is able to connect to `www.storephotos.example.com` using a computer that provides an easy data-entry method, which is equipped with a browser. This computer is used to authorize access by the application running on the game console to Alice's photographs
- *Application running on Alice's game console has registered with `www.storephotos.example.com` (has been issued an identifier)
- *Alice has registered with the application running at `www.storephotos.example.com` for identification and authentication

Post-conditions: Description:

A successful procedure results in the application running on Alice's game console receiving an access token that enables access to the photographs on `www.storephotos.example.com`.

Requirements:

- *Registration of the application running on the game console with the application running on `www.storephotos.example.com` is required for identification
- *Application running on the game console must be able to poll periodically the application running at `www.storephotos.example.com` while waiting for Alice's authorization of the requested access to her photographs. The repeating requests include the application's identifier and the verification code that has been issued by `www.storephotos.example.com`
- *Alice is required to use her browser for interacting with the web application running on `www.storephotos.example.com`. To that end she has to manually direct her browser to the verification URL that is displayed on her game console
- *Alice's authentication with `www.storephotos.example.com` is required
- *After authentication with `www.storephotos.example.com` Alice, if she wishes to approve the request, which is described in her

browser's window, must enter the user code. (The user code is also displayed on her game console along with the verification URL)

3.6. Client password (shared secret) credentials

Description:

The company GoodPay prepares the employee payrolls for the company GoodWork. In order to do that the application at `www.GoodPay.example.com` gets authenticated access to the employees' attendance data stored at `www.GoodWork.example.com`.

Pre-conditions:

- *The application at `www.GoodPay.example.com` has established through a registration an identifier and a shared secret with the application running at `www.GoodWork.example.com`
- *The scope of the access by the application at `www.GoodPay.example.com` to the data stored at `www.GoodWork.example.com` has been defined

Post-conditions:

A successful procedure results in the application at `www.GoodPay.example.com` receiving an access token after authenticating to the application running at `www.GoodWork.example.com`.

Requirements:

- *Authentication of the application at `www.GoodPay.example.com` to the application at `www.GoodWork.example.com` is required
- *The authentication method must be based on the identifier and shared secret, which the application running at `www.GoodPay.example.com` submits to the application at `www.GoodWork.example.com` in the initial HTTP request
- *Because in this use case GoodPay gets access to GoodWork's sensitive data, GoodWork shall have a pre-established trust with GoodPay on the security policy and the authorization method's implementation

3.7. Assertion

Description:

Company GoodPay prepares the employee payrolls for the company GoodWork. In order to do that the application at

www.GoodPay.example.com gets authenticated access to the employees' attendance data stored at www.GoodWork.example.com.
This use case describes an alternative solution to the one described by the use case Client password credentials.

Pre-conditions:

- *The application at www.GoodPay.example.com has obtained an authentication assertion from a party that is trusted by the application at www.GoodWork.example.com
- *The scope of the access by the application at www.GoodPay.example.com to the data stored at www.GoodWork.example.com has been defined
- *The application at www.GoodPay.example.com has established trust relationship with the asserting party and is capable of validating its assertions

Post-conditions:

A successful procedure results in the application at www.GoodPay.example.com receiving an access token after authenticating to the application running at www.GoodWork.example.com by presenting an assertion (e.g., SAML assertion).

Requirements:

- *Authentication of the application at www.GoodPay.example.com to the application at www.GoodWork.example.com is required
- *The application running at www.GoodWork.example.com must be capable of validating assertion presented by the application running at www.GoodPay.example.com
- *Because in this use case GoodPay gets access to GoodWork's sensitive data, GoodWork shall establish trust with GoodPay on the security policy and the authorization method's implementation

3.8. Content manager

Description:

Alice and Bob are having a chat conversation using a content manager application running on a web server at www.contentmanager.example.com. Alice notifies Bob that she wants to share some photographs at www.storephotos.example.com and instructs the application at www.contentmanager.example.com to enable Bob's access to the photographs. The application at www.contentmanager.example.com, after Alice's authorization, obtains an access token for Bob, who uses it to access Alice's photographs at www.storephotos.example.com.

Pre-conditions:

Alice, Bob the content manager application at `www.contentmanager.example.com`, and the application at `www.storephotos.example.com` have registered with the same authorization server for authentication

Post-conditions:

A successful procedure results in the application at `www.contentmanager.example.com` receiving an access token that allows access to Alice's photographs at `www.storephotos.example.com`. The access token is issued by the authorization server after Alice has authorized the content manager at `www.contentmanager.example.com` to get an access token on Bob's behalf. The access token is passed to Bob by the content manager. Bob uses the access token to view Alice's photographs at `www.storephotos.example.com`.

Requirements:

- *The server at `www.contentmanager.example.com`, must be capable of issuing the HTTP redirect requests to Alice's and Bob's user agents - the browsers
- *The authorization server must be able to authenticate Alice, Bob, and the application at `www.contentmanager.example.com`
- *The authorization server is required to obtain Alice's authorization for issuing an access token to `www.contentmanager.example.com` on Bob's behalf
- *Authorization server must be able to identify to Alice the scope of access that `www.contentmanager.example.com` has requested on Bob's behalf while asking for Alice's authorization

3.9. Access token exchange

Description:

Alice uses an application running on `www.printphotos.example.com` for printing her photographs that are stored on a server at `www.storephotos.example.com`. The application running on `www.storephotos.example.com`, while serving the request of the application at `www.printphotos.example.com`, discovers that some of the requested photographs have been moved to `www.storephotos1.example.com`. The application at `www.storephotos.example.com` retrieves the missing photographs from `www.storephotos1.example.com` and provides access to all requested photographs to the application at `www.printphotos.example.com`. The application at `www.printphotos.example.com` carries out Alice's request.

Pre-conditions:

- *The application running on `www.printphotos.example.com` is capable of interacting with Alice's browser
- *Alice has registered with and can be authenticated by authorization server
- *The applications at `www.storephotos.example.com` has registered with authorization server
- *The applications at `www.storephotos1.example.com` has registered with authorization server
- *The application at `www.printphotos.example.com` has registered with authorization server

Post-conditions:

A successful procedure results in the application at `www.printphotos.example.com` receiving an access token that allows access to Alice's photographs. This access token is used for the following purposes:

- *By the application running at `www.printphotos.example.com` to get access to the photographs at `www.storephotos.example.com`
- *By the application running at `www.storephotos.example.com` to obtain from authorization server another access token that allows it to retrieve the additional photographs stored at `www.storephotos1.example.com`

As the result, there are two access token issued for two different applications. The tokens may have different properties (e.g., scope, permissions, and expiration dates).

Requirements:

- *The applications at `www.printphotos.example.com` and `www.storephotos.example.com` require different access tokens
- *The application at `www.printphotos.example.com` is required to provide its callback URL to the application at `www.storephotos.example.com`
- *Authentication of the application at `www.printphotos.example.com` to the authorization server is required
- *Alice's authentication by the authorization server is required

- *The authorization server must be able to describe to Alice the request of the application at `www.printphotos.example.com` and obtain her authorization (or rejection)
- *If Alice has authorized the request, the authorization server must be able to issue an access token that enables the application at `www.printphotos.example.com` to get access to Alice's photographs at `www.storephotos.example.com`
- *The authorization server must be able, based on the access token presented by the application at `www.printphotos.example.com`, to generate another access token that allows the application at `www.storephotos.example.com` to get access to the photographs at `www.storephotos1.example.com`. In this context the authorization server must validate the authorization of the application at `www.storephotos.example.com` to obtain the token.
- *The application at `www.storephotos.example.com` must be able to validate an access token presented by the application running at `www.printphotos.example.com`
- *The application at `www.storephotos1.example.com` must be able to validate the access token presented by the application running at `www.storephotos.example.com`

3.10. Multiple access tokens

Description:

Alice uses a communicator application running on a web server at `www.communicator.example.com` to access her email service at `www.email.example.com` and her voice over IP service at `www.voip.example.com`. Email addresses and telephone numbers are obtained from Alice's address book at `www.contacts.example.com`. Those web sites all rely on the same authorization server, so the application at `www.communicator.example.com` can receive a single authorization from Alice for getting access to these three services on her behalf at once.

Note: This use case is especially useful for native applications since a web browser needs to be launched only once.

Pre-conditions:

- *The same authorization server serves Alice and all involved servers
- *Alice has registered with the authorization server for authentication and for authorization of the requests of the communicator application running at `www.communicator.example.com`

- *The email application at `www.email.example.com` has registered with the authorization server for authentication
- *The VoIP application at `www.voip.example.com` has registered with the authorization server for authentication
- *The address book at `www.contacts.example.com` has registered with the authorization server for authentication

Post-conditions:

A successful procedure results in the application at `www.communicator.example.com` receiving three different access tokens: one for accessing the email service at `www.email.example.com`, one for accessing the contacts at `www.contacts.example.com`, and one for accessing the VoIP service at `www.voip.example.com`.

Requirements:

- *The application running at `www.communicator.example.com` must be authenticated by the authorization server
- *Alice must be authenticated by the authorization server
- *The application running at `www.communicator.example.com` must be able to get a single Alice's authorization for access to the multiple services (e.g., email and VoIP)
- *The application running at `www.communicator.example.com` must be able to recognize that all three applications rely on the same authorization server
- *A callback URL of the application running at `www.communicator.example.com` must be known to the authorization server
- *The authorization server must be able to issue the separate service-specific tokens (with different, scope, permissions, and expiration dates) for access to the requested services (such as email and VoIP)

3.11. Gateway for browser-based VoIP applets

Description:

Alice accesses a social site on a web server at `www.social.example.com`. Her browser loads a VoIP applet that enables her to make a VoIP call using her SIP server at `www.sipservice.example.com`. The application at `www.social.example.com` gets Alice's authorization to use her account

with `www.sipservice.example.com` without learning her authentication credentials with `www.sipservice.example.com`.

Pre-conditions:

- *Alice has registered with `www.sipservice.example.com` for authentication
- *The application at `www.social.example.com` has established authentication credentials with the application at `www.sipservice.example.com`

Post-conditions:

A successful procedure results in the application at `www.social.example.com` receiving access token from `www.sipservice.example.com` with Alice's authorization.
Requirements:

- *The server at `www.social.example.com` must be able to redirect Alice's browser to `www.sipservice.example.com`
- *The application running at `www.sipservice.example.com` must be capable of authenticating Alice and obtaining her authorization of a request from `www.social.example.com`
- *The server at `www.sipservice.example.com` must be able to redirect Alice's browser back to `www.social.example.com`
- *The application at `www.social.example.com` must be able to translate the messages of the Alice's VoIP applet into SIP and RTP messages
- *The application at `www.social.example.com` must be able to add the access token to the SIP requests that it sends to `www.sipservice.example.com`
- *Application at `www.sipservice.example.com` must be able to authenticate the application at `www.social.example.com` and validate the access token
- *Alice's manual involvement in the OAuth authorization procedure (e.g., entering an URL or a password) should not be required. (Alice's authentication to `www.sipservice.example.com` is not in the OAuth scope)

3.12. Signed Messages

Description:

Alice manages all her personal health records in her personal health data store at a server at `www.myhealth.example.com`, which manages authorization of access to Alice's participating health systems. Alice's Primary Care Physician (PCP), which has a Web site at `www.pcp.example.com`, recommends her to see a sleep specialist (`www.sleepwell.example.com`). Alice arrives at the sleep specialist's office and authorizes it to access her basic health data at her PCP's web site. The application at `www.pcp.example.com` verifies that Alice has authorized `www.sleepwell.example.com` to access her health data as well as enforces that `www.sleepwell.example.com` is the only application that can retrieve that data with that specific authorization.

Pre-conditions:

- *Alice has a personal health data store that allows for discovery of her participating health systems (e.g. psychiatrist, sleep specialist, PCP, orthodontist, ophthalmologist, etc)
- *The application at `www.myhealth.example.com` manages authorization of access to Alice's participating health systems
- *The application at `www.myhealth.example.com` can issue authorization tokens understood by Alice's participating health systems
- *The application at `www.pcp.example.com` stores Alice's basic health and prescription records
- *The application at `www.sleepwell.com` stores results of Alice's sleep tests

Post-conditions:

- *A successful procedure results in just the information that Alice authorized being transferred from the Primary Care Physician (`www.pcp.example.com`) to the sleep specialist (`www.sleepwell.example.com`)
- *The transfer of health data only occurs if the application at `www.pcp.example.com` can verify that `www.sleepwell.example.com` is the party requesting access and that the authorization token presented by `www.sleepwell.example.com` is issued by the application at `www.myhealth.example.com` with a restricted audience of `www.sleepwell.example.com`

Requirements:

- *The application at `www.sleepwell.example.com` interacting with `www.myhealth.example.com` must be able to discover the location of the PCP system (e.g., XRD discovery)

- *The application at `www.sleepwell.example.com` must be capable of requesting Alice's authorization of access to the application at `www.pcp.example.com` for the purpose of retrieving basic health data (e.g. date-of-birth, weight, height, etc). The mechanism Alice uses to authorize this access is out of scope for this use case
- *The application at `www.myhealth.example.com` must be capable of issuing a token bound to `www.sleepwell.example.com` for access to the application at `www.pcp.example.com`. Note that a signed token (JWT) can be used to prove who issued the token
- *The application at `www.sleepwell.example.com` must be capable of issuing a request (which includes the token issued by `www.myhealth.example.com`) to the application at `www.pcp.example.com`
- *The application at `www.sleepwell.example.com` must sign the request before sending it to `www.pcp.example.com`
- *The application at `www.pcp.example.com` must be capable of receiving the request and verifying the signature
- *The application at `www.pcp.example.com` must be capable of parsing the message and finding the authorization token
- *The application at `www.pcp.example.com` must be capable of verifying the signature of the authorization token
- *The application at `www.pcp.example.com` must be capable of parsing the authorization token and verifying that this token was issued to the application at `www.sleepwell.com`
- *The application at `www.pcp.example.com` must be capable of retrieving the requested data and returning it to the application at `www.sleepwell.example.com`

3.13. Signature with asymmetric secret

Description:

Alice accesses an application running on a web server at `www.printphotos.example.com` and instructs it to print her photographs that are stored on a server `www.storephotos.example.com`. The application at `www.printphotos.example.com`, which does not have a shared secret with `www.storephotos.example.com`, receives Alice's authorization for accessing her photographs without learning her authentication credentials with `www.storephotos.example.com`.

Pre-conditions:

- *Alice has registered with `www.storephotos.example.com` to enable authentication
- *The application at `www.printphotos.example.com` has a private and a matching public keys

Post-conditions:

A successful procedure results in the application at `www.printphotos.example.com` receiving an access token from `www.storephotos.example.com` for accessing the Alice's photographs.

Requirements:

- *The application at `www.printphotos.example.com` must be capable of issuing the HTTP redirect requests to Alice's user agent - a browser
- *The application at `www.storephotos.example.com` must be able to authenticate Alice
- *The application running at `www.storephotos.example.com` must be able to obtain the public key of the application at `www.printphotos.example.com`
- *The application running at `www.printphotos.example.com` is required to sign using its private key the requests to the application at `www.storephotos.example.com`
- *The application at `www.storephotos.example.com` must obtain Alice's authorization of the access to her photos by `www.printphotos.example.com`
- *The application at `www.storephotos.example.com` is required to identify to Alice the scope of access that `www.printphotos.example.com` has requested while asking for Alice's authorization
- *The application at `www.storephotos.example.com` must be able to authenticate the application at `www.printphotos.example.com` by validating a signature of its request using the public key of `www.printphotos.example.com`
- *The application at `www.printphotos.example.com` must provide a callback URL to the application at `www.storephotos.example.com` (note: the URL can be pre-registered with `www.storephotos.example.com`)

*The application at `www.storephotos.example.com` must be capable of issuing the HTTP redirect requests to Alice's browser

*Alice's manual involvement in the OAuth authorization procedure (e.g., entering an URL or a password) should not be required. (Alice's authentication to `www.storephotos.example.com` is not in the OAuth scope)

4. Authors of the use cases

The major contributors of the use cases are as follows:

W. Beck, Deutsche Telekom AG
G. Brail, Sonoma Systems
B. de hOra
B. Eaton, Google
S. Farrell, NewBay Software
G. Fletcher, AOL
Y. Goland, Microsoft
B. Goldman, Facebook
E. Hammer-Lahav, Yahoo!
D. Hardt
R. Krikorian, Twitter
T. Lodderstedt, Deutsche Telekom
E. Maler, PayPal
D. Recordon, Facebook
L. Shepard, Facebook
A. Tom, Yahoo!
B. Vrancken, Alcatel-Lucent
Z. Zeltsan, Alcatel-Lucent

5. Security considerations

TBD

6. IANA considerations

This Internet Draft includes no request to IANA.

7. Acknowledgements

The authors thank Igor Faynberg and Hui-Lan Lu for their invaluable help with preparing this document. Special thanks are to the draft reviewers Thomas Hardjono and Melinda Shore, whose suggestions have helped to improve the draft.

8. References

[RFC2119]	
-----------	--

	Bradner, S., " Key words for use in RFCs to Indicate Requirement Levels ", RFC 2119, March 1997.
[I-D.draft-ietf-oauth-v2]	Hammer-Lahav, E., Recordon, D. and D. Hardt, "The OAuth 2.0 Authorization Protocol", .

Authors' Addresses

George Fletcher Fletcher AOL EMail: gffletch@aol.com

Torsten Lodderstedt Lodderstedt Deutsche Telekom AG EMail:
torsten@lodderstedt.net

Zachary Zeltsan Zeltsan Alcatel-Lucent 600 Mountain Avenue Murray Hill, New Jersey USA Phone: +1 908 582 2359 EMail:
Zachary.Zeltsan@alcatel-lucent.com