OAuth authentication

Developer's guide

24.09.2014

OAuth authentication. Developer's guide. Version 1.3
Document build date: 24.09.2014.
This volume is a part of Yandex technical documentation.
Yandex helpdesk site: http://help.yandex.ru
© 2008—2014 Yandex LLC. All rights reserved.

Copyright Disclaimer

Contact information

Yandex LLC
http://www.yandex.com
Phone: +7 495 739 7000
Email: pr@yandex-team.ru
Headquarters: 16 L'va Tolstogo St., Moscow, Russia 119021

Contents

Introduction

This guide is intended for application developers using the Yandex APIs to access users' personal data.

It provides an overview of how the OAuth 2.0 protocol is used for authenticating applications on Yandex services.

Sections in this document:

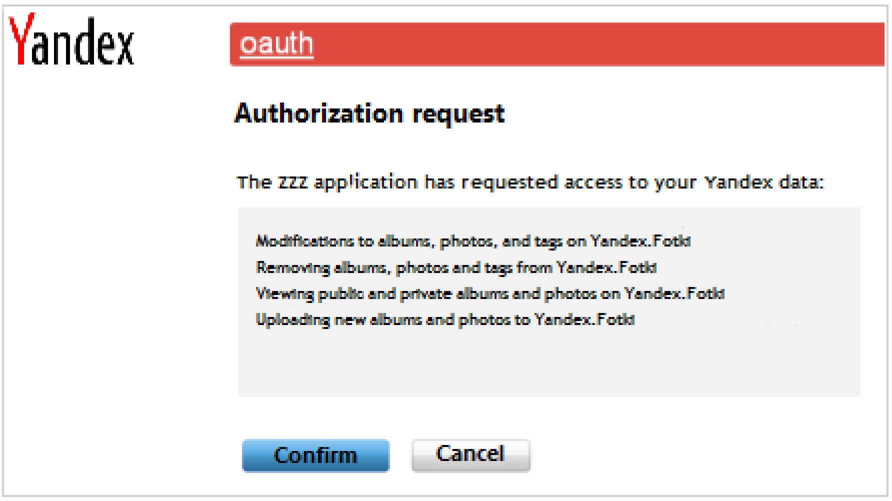
- Authorization flow — describes the interaction between an application and the Yandex OAuth server, different ways to get a token, application properties, and parameters for requests to the server.
- Samples — choosing how to get a token depending on the type of application (web service, mobile application, or other).
- Manually obtaining a debugging token — how to quickly get a token manually. A debugging token can be used in requests to the APIs of Yandex services during application development.
- Managing applications — how to view and edit properties of applications you have registered on the Yandex OAuth server; how users can control applications' access to their personal data.

For background information and details on parameters in the protocol, please refer to the official OAuth 2.0 Specification (currently a draft).

About the OAuth protocol

A third-party client application must get authorization to access a user's personal data in Yandex services (profiles and blogs in Ya.ru, photos on Yandex.Fotki, subscriptions, etc.).

The OAuth protocol makes authorization secure and convenient. With OAuth authentication, applications don't need to ask users for their Yandex credentials, because they can ask users for permission to access their data directly on the Yandex web site.



Authorization request

The ZZZ application has requested access to your Yandex data:

Modifications to albums, photos, and tags on Yandex.Fotki
Removing albums, photos and tags from Yandex.Fotki
Viewing public and private albums and photos on Yandex.Fotki
Uploading new albums and photos to Yandex.Fotki

After getting authorization from a user, the Yandex OAuth server generates an **access token**; the client application can use the token to access this user's personal data.

OAuth can be used by any type of application, such as web applications, mobile applications, desktop applications, and others.

What is an access token?

An access token is a string containing permission settings, meaning a scope that specifies which operations a particular application may perform with a particular user's personal data.

A token may have a limited lifespan.

The user can revoke authorization to access personal data at any time, and the token that was issued becomes invalid.

Advantages of OAuth authentication

Authentication that doesn't require the user's login credentials has many advantages:

- **Security**.

  When developing an application, you don't need to worry about the confidentiality of the user's credentials. The login and password are not sent to the application, so they can't get into the wrong hands.
- **User loyalty**.

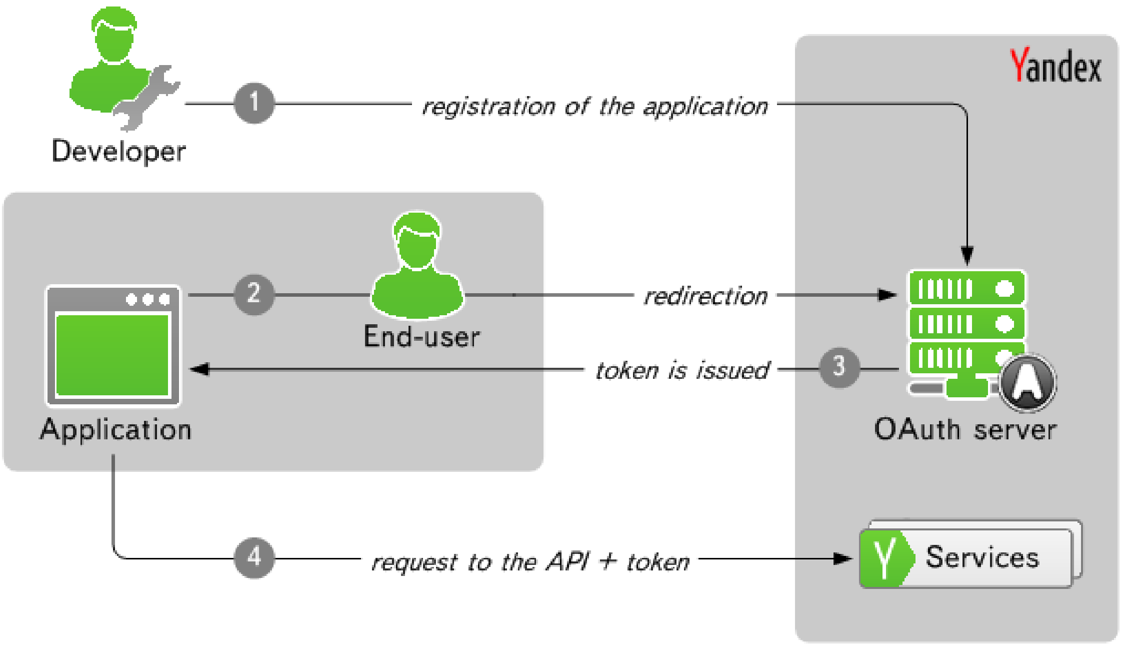
  Users are more likely to trust an application when they are sure that their personal data is not vulnerable to unauthorized access. Since the application doesn't have the user's login and password, it can only access personal data exactly as the user authorized, and nothing more.
- **User convenience**.

  Users who are already signed in to Yandex (for example, in Mail) do not need to re-enter their login and password on Yandex when granting authorization to an application. Not having to enter credentials frequently is a real benefit for mobile users.

See also
The official OAuth web site
OAuth (content from Wikipedia)
OAuth 2.0 protocol specification (draft)

Authorization flow

This diagram shows a simplified illustration of the interaction between an application, the OAuth server and Yandex services.



1. The developer manually registers an application on the Yandex OAuth server.

   See the section Registering an application.
2. The application redirects the user to the access request page. The user authorizes the application to access personal data.

   See the section Obtaining an access token.
3. The OAuth server implements a redirect to the URI specified by the developer, and appends a parameter containing an access token or an authorization code that allows the application to get a token.

   See the section Obtaining an access token.
4. The application specifies this token in all API requests that access the user's personal data. The access token is used for identifying and authenticating both the user and the application.

   See the section Using an access token to access personal user data.

Tip:
During the application development stage, you can use a debugging token for accessing the APIs of Yandex services. You can get a debugging token manually by granting permission to access data on behalf of a test user; see the section Manually obtaining a debugging token. A debugging token lets you develop and test the main features of your application before implementing a mechanism for obtaining a token.

Registering an application

You must sign in to Yandex to complete these steps.

To register an application on the Yandex OAuth server:

1. Navigate to https://oauth.yandex.com/client/new and go to the **Register new application** page.
2. Specify application properties:

   Name
   The title of the application.

   Permissions
   A set of access rights (the same as the `scope` parameter in the OAuth protocol). Select the operations that the application will be allowed to perform after getting a token.

   Note:
   Some permissions show a period of validity. All other permissions do not expire. The lifespan of a token corresponds to the shortest lifespan of its individual permissions.

   Icon link
   The link to the application's icon. Optional property.

   Application link
   The link to the application's web site. Optional property.

   Callback URI
   The URI that the OAuth server redirects to after getting the user's authorization (corresponds to the `redirect_uri` parameter in the OAuth protocol).

   For examples of values for this parameter for various types of applications, see the section Samples.

   Client for development
   Select this option if you need to manually get an access token via the oauth.yandex.com service web interface for debugging purposes (see the section Manually obtaining a debugging token).

   Note:

   - In the Yandex implementation of the OAuth protocol, the `redirect_uri` and `scope` parameter values are set when you register the application. You can't send these parameters when requesting user authorization.
   - The **Name**, **Icon link** and **Application link** properties are used for displaying application information on user pages in the oauth.yandex.com service.

3. Click the **Create** button.

After registration, you will see the application information page. This page displays the application's **ID** and **password** generated by the Yandex OAuth server (these are the `client_id` and `client_secret` parameters in the OAuth protocol), which must be provided when requesting an access token.

Attention!
The security of the application's password can only be guaranteed for a web application that runs on its own server. If the application runs on the user's computer (either desktop or mobile), there is the possibility that the password could be extracted from the application.

See also
Managing applications
Authorization flow
Samples

Obtaining an access token

To get authorization from the user to access personal data, the application has to redirect the user to the following URL:

```
https://oauth.yandex.com/authorize?response_type=<token|
code>&client_id=<client_id>[&display=popup][&state=<state>]
```

Note:
In the Yandex implementation of the OAuth protocol, the value of the `redirect_uri` parameter is set when registering the application (instead of being passed when requesting user authorization).

Request parameters:

response_type
The type of response required by the OAuth server, meaning the parameter name that the OAuth server will add to the `redirect_uri`:

- `token` — The access token.

  The token is passed as a fragment component (appended to the URI after the # symbol).
- `code` — The authorization code.

  The authorization code is passed as a query parameter (appended to the URI after the **?** symbol).

An authorization code is used when it's either not possible or not optimal to send a token as a fragment component. In this case, a token can be obtained using a POST request to the OAuth server, specifying the authorization code as a parameter. The code is valid for one hour from the moment it was issued, and can only be used once to obtain a token.

client_id
The application ID that was assigned during registration. See the section Registering an application.

display=**popup**
A page with a lightweight layout that is convenient to open in a narrow popup window.

Optional parameter. If it is not specified, the authorization request page is displayed in normal mode.

state
An arbitrary state parameter, which you can use for passing any type of information to the `redirect_uri`. It will be appended to the `redirect_uri` as-is by the OAuth server. Optional parameter.

After getting a response from the user, the Yandex OAuth server performs a redirect to the `redirect_uri`.

If the user granted the application access to personal data, the set of parameters to be passed by the OAuth server depends on the value of `response_type`. See the sections Obtaining an access token directly (response_type=token) and Using an authorization code (response_type=code) below.

If the user denied the application access, the OAuth server adds to `redirect_uri` the parameter `error=access_denied` (as well as the `state` parameter, if it was specified in the original request).

Obtaining an access token directly (response_type=token)

After the user confirms that the application may access personal information, the Yandex OAuth server performs a redirect to the `redirect_uri`, which has parameters added to it (after the # symbol) in the following format:

```
#access_token=<token>[&expires_in=<seconds>][&state=<state>]
```

access_token
An access token that the application can use to work on the user's behalf.

expires_in
The lifespan of the token. The value is shown in seconds.

This parameter is used only when the token has a limited lifespan.

state
The state parameter. This parameter is used only if it was specified in the original request. The value is copied from the request.

Using an authorization code (response_type=code)

After the user confirms that the application may access personal information, the Yandex OAuth server performs a redirect to the `redirect_uri`, which has the following query parameters added to it (after the **?** symbol):

code
Authorization code.

state
The state parameter. This parameter is used only if it was specified in the original request. The value is copied from the request.

To obtain a token, an application must send a POST request to **https://oauth.yandex.com/token**. The request must include the authorization code, along with the application's ID and password.

The ID and password can be sent using either of the following methods:

- In the request body
- Using basic HTTP authentication

Attention!
All OAuth requests must be sent to the server over HTTPS.

In the request body

```
POST /token HTTP/1.1
Host: oauth.yandex.com
Content-type: application/x-www-form-urlencoded
Content-Length: <length>

grant_type=authorization_code&code=<code>&client_id=<client_id>&client_secret=<c
lient_secret>
```

Request parameters:

- `grant_type = authorization_code`
- `code = <code>` — The authorization code.
- `client_id = <client_id>` — The application ID.
- `client_secret = <client_secret>` — The application password.

Using basic HTTP authentication

```
POST /token HTTP/1.1
Host: oauth.yandex.com
Content-type: application/x-www-form-urlencoded
Content-Length: <length>
Authorization: Basic <base64-encoded-string>

grant_type=authorization_code&code=<code>
```

The `Authorization` header is used for passing the *`<client_id>`*`:`*`<client_secret>`* string, which is base64 encoded. Here, *`<client_id>`* is the application ID and *`<client_secret>`* is the application password.

Parameters are specified in the body of the request:

- `grant_type = authorization_code`
- `code = <code>` — The authorization code.

You don't need to specify the `client_id` parameter in the body of the request, since the application ID is already included in the `Authorization` header.

Server response

If successful, the OAuth server passes the data as a JSON document. Example:

```
200 OK
Content-type: application/json
Content-Length: 76

{"access_token": "ea135929105c4f29a0f5117d2960926f", "expires_in": 2592000}
```

Response parameters:

access_token
An access token that the application can use to work on the user's behalf.

expires_in
The lifespan of the token. The value is shown in seconds.

This parameter is used only when the token has a limited lifespan.

If there are errors, the OAuth server passes the parameter

error
Error code:

- `invalid_request` — Incorrect request format.
- `invalid_grant` — Invalid or expired authorization code.
- `unsupported_grant_type` — Incorrect value for the `grant_type` parameter.

See also
Authorization flow
Samples
Sections from the OAuth 2.0 protocol specification:
Obtaining Authorization
Client Password
The section Basic Authentication Scheme in the RFC 2617 specification

Using an access token to access personal user data

The access token must be used in all requests sent to Yandex APIs that access the user's personal data:

- in an `Authorization` header:

  ```
  Authorization: OAuth <token>
  ```

- in a query parameter:

  ```
  https://api-metrika.yandex.ru/...?oauth_token=<token>
  ```

Attention!
When using OAuth authentication, we recommend sending requests to Yandex services over the HTTPS protocol whenever possible.

How to get a user's personal base URL

In many of the Yandex service APIs (Yandex.Fotki, Ya.ru and others), user data is in the form of resources that have addresses containing the user's identifier (login name or uid).

With OAuth authorization, the application only gets the access token, not the user's login name or uid. To solve this, the APIs have a special `/me/` URL, which is used for identifying a user by token. Having gotten a token, the application must send a request containing the token to this URL. The API server performs a redirect to the user's personal URL.

Example request to the Yandex.Fotki server:

```
GET http://api-fotki.yandex.ru/api/me/ HTTP/1.1
Host: api-fotki.yandex.ru
Authorization: OAuth eb1c55f...
```

Example server response:

```
HTTP/1.1 302 FOUND
Content-Type: text/plain; charset=utf-8
Location: http://api-fotki.yandex.ru/api/users/testuser/
```

The personal URL is a base address for obtaining URLs for all the resources associated with this user. The application can save the personal URL for further use.

Note that you do not need to extract the user ID from the personal URL. The complete URL of any resource can be obtained from the server. A resource that is accessible via a personal URL contains a link to other resources; by following the links, you can get to any resource.

You can find descriptions of resources and their request formats in the documentation for the corresponding APIs.

Samples

The following sections summarize recommended ways to obtain an access token for different types of applications, as well as suggested values for the OAuth parameters `redirect_uri` and `response_type`:

- Web service

  This example considers the case when a token can't be added to the URI as a fragment component after the # symbol, since many server environments do not pass these fragments to a web service. So we use an authorization code and send it as a query parameter. The application intercepts the redirect to the `redirect_uri`, extracts the authorization code from the URI, and then obtains a token using a POST request to the OAuth server.
- Mobile applications, Desktop applications

  Here we look at cases when a token can be passed directly.
- Console applications

  This example demonstrates a case when the redirect can't be intercepted. The OAuth server displays the authorization code on a web page where the user can copy it manually, and an application can later request the code from the user.

See also
Obtaining an access token
Client Profiles section of the OAuth 2.0 protocol

Web service

If the application is a web service, we recommend using the following values for OAuth parameters:

redirect_uri
When registering your application, set the web service's address (web page) as the redirection address, so the OAuth server can pass the authorization code to this address.

For example:

```
http://www.service.ru/tokens
```

response_type
When requesting authorization from the user, specify `response_type=code`. In other words, redirect the user to the following URL:

```
https://oauth.yandex.com/authorize?response_type=code&client_id=<client_id>
```

After the user authorizes the application to access personal data, intercept the redirect to the `redirect_uri` and copy the authorization code from the URI:

```
http://www.service.ru/tokens?code=<code>
```

Then obtain a token using a POST request.

See also
Authorization flow
Samples

Mobile applications

For mobile applications, we recommend using the following values for OAuth parameters:

redirect_uri

Use your own URI scheme. When registering your application on the OAuth server, set a link to this scheme as the redirection address. For example:

```
myapp://token
```

Register your application in the mobile OS as a handler for this scheme, so that your application is launched when the user clicks the link with this scheme (the link starting with `myapp://`).

response_type

When requesting authorization from the user, specify `response_type=token`. In other words, redirect the user to the following URL:

```
https://oauth.yandex.com/authorize?response_type=token&client_id=<client_id>
```

After the user grants the application access to personal data, the application will open with a URI that looks like this:

```
myapp://token#access_token=<token>&expires_in=<seconds>
```

See also
Authorization flow
Samples

Desktop applications

For desktop applications, we recommend using the following values for OAuth parameters:

redirect_uri

Use your own URI scheme. When registering your application on the OAuth server, set a link to this scheme as the redirection address. For example:

```
myapp://token
```

response_type

When requesting user authorization, specify `response_type=token`.

Use the OS browser control tools to embed a browser window in your application. In the browser, open the following URL:

```
https://oauth.yandex.com/authorize?response_type=token&client_id=<client_id>
```

After the user authorizes the application to access personal data, intercept the redirect to the `redirect_uri` and copy the token from the URI:

```
myapp://token#access_token=<token>&expires_in=<seconds>
```

See also
Authorization flow
Samples

Console applications

Tip:
The recommendations in this section apply both to console applications and to other types of applications as well.

Follow these recommendations if your development tools do not allow you to get an access token in a more user-friendly way.

redirect_uri

During application registration, for the redirect address setting, enter the address of the page that will display the authorization code for the user. You can use this oauth.yandex.com page:

```
https://oauth.yandex.com/verification_code
```

or create your own.

response_type

When requesting authorization from the user, specify `response_type=code`. In other words, redirect the user to the following URL:

```
https://oauth.yandex.com/authorize?response_type=code&client_id=<client_id>
```

After the user grants the application access to personal data, the authorization code will be displayed on the page with the address from the `redirect_uri`. Make sure the application provides a way to get the authorization code from the user, such as a command-line request, configuration file, etc.

Then obtain a token using a POST request.

See also
Authorization flow
Samples

Manually obtaining a debugging token

You can get an authorization code and debugging token manually via the web interface for the oauth.yandex.com service.

To get a token manually:

1. Register or edit your application and specify the following URL in the **Callback URI** field:

   ```
   https://oauth.yandex.com/verification_code?dev=true
   ```
2. Authenticate on Yandex with the user account that the application will be working on behalf of, then go to the link

   ```
   https://oauth.yandex.com/authorize?
   response_type=token&client_id=<application ID>
   ```

   The ID is marked with the label **Application ID** on the application's page.
3. On the page that opens, click the **Confirm** button to grant the application access to your account. You do not need to grant access if a token has already been issued for this application and has not expired yet.

The service redirects you to a URL containing the token in the value of the `access_token` parameter:

```
https://oauth.yandex.com/verification_code?dev=True#access_token=<access
token>&token_type=bearer&state=
```

See also
Authorization flow
Registering an application

Managing applications

For developers

The following interface is provided for developers to manage applications:

- The https://oauth.yandex.com/client/my page displays a list of applications the developer has registered on the Yandex OAuth server. Click the link for an application name to go to the application information page.
- The `https://oauth.yandex.com/client/<client_id>` page provides information about the application:

  - Application ID (`client_id`).
  - Application password (`client_secret`).
  - List of access permissions.
  - The number of token authentications that have been performed for this application.

  The application information page also contains a link to the page for editing application properties, and a button for canceling application registration.
- The page `https://oauth.yandex.com/client/<client_id>/edit` is for editing application properties, and also allows you to generate a new application password when necessary.

For users

On the https://oauth.yandex.com/list_tokens web page, users can view a list of applications that are using access tokens to access their personal information in Yandex services. For each application, there is a list of permissions that were selected by the developer during registration of the client application.

The user can deny access to personal information, if there is any doubt about an application's security. Access rights for this token will be revoked.

See also
About the OAuth protocol
Registering an application

Change history

October 2011

Added application passwords as the `client_secret` request parameter on `oauth.yandex.com/token`.

The application password is generated by the OAuth server during application registration and is displayed on the application information page. A password can also be generated on the page for editing application properties.

April 2011

Added the `display` parameter for requests on `oauth.yandex.com/authorize`.

Added the ability to edit properties of client applications.

February 2011

Added the `state` parameter for requests on `oauth.yandex.com/authorize`.

November 2010

Launched.

Index

Yandex

OAuth authentication

Developer's guide

24.09.2014