aioauth

Release 1.3.0

Ali Aliyev

CONTENTS

I	Installing	3
2	Supported RFC	5
3	Pages	7
4	Sections 4.1 Plug-and-Play	
	4.2 Configuration	9 10
	4.4 Aiohttp	10 10
	4.6 Collections 4.7 Config	11 11
	4.8 Constances	12 12
	4.10 Grant Type	14 15
	4.12 Requests	18 20
	4.14 Responses	20 22
	4.16 Storage	25 28
	4.18 Utils	
	Indices and tables	33
٠	othon Module Index	35 37
HIL	MUA.	51

aioauth is a spec-compliant OAuth 2.0 asynchronous Python module. aioauth works out-of-the-box with asynchronous server frameworks like FastAPI, Starlette, aiohttp, and others, as well as asynchronous database modules like Motor (MongoDB), aiopg (PostgreSQL), aiomysql (MySQL), or ORMs like Gino, sqlalchemy, or Tortoise.

The magic of aioauth is its plug-and-play methods that allow the use of virtually any server or database framework.

CONTENTS 1

2 CONTENTS

ONE

INSTALLING

To install aioauth at the command line:

\$ pip install aioauth

To install pre-releases:

\$ pip install git+https://github.com/aliev/aioauth

TWO

SUPPORTED RFC

aioauth supports the following RFCs:

- RFC 6749 The OAuth 2.0 Authorization Framework
- RFC 7662 OAuth 2.0 Token Introspection
- RFC 7636 Proof Key for Code Exchange by OAuth Public Clients

THREE

PAGES

- Github Project
- Issues
- Discussion

8 Chapter 3. Pages

FOUR

SECTIONS

4.1 Plug-and-Play

aioauth was designed to be as flexible as possible to allow developers to choose their own server framework, as well as database provider. Since aioauth is written as an asynchronous module it would be to the advantage of the developer to write their applications asynchronously.

4.2 Configuration

All aioauth settings are made through aioauth.config.Settings class.

Defaults

Setting	Default	Description
	value	
TOKEN_EXPIRES_IN	86400	Access token lifetime.
AUTHORIZA-	300	Authorization code lifetime.
TION_CODE_EXPIRES_IN	V	
INSE-	False	Allow connections over SSL only. When this option is disabled server
CURE_TRANSPORT		will raise "HTTP method is not allowed" error.

the default settings can be changed as follows:

```
import os
from aioauth.config import Settings

settings = Settings(
    INSECURE_TRANSPORT=not os.getenv('DEBUG', False)
)
```

this example disables checking for insecure transport, depending on the debug mode of the current environment.

The aioauth.requests.Request consumes an instance of the aioauth.config.Settings class:

```
import os
from aioauth.config import Settings
from aioauth.requests import Request

settings = Settings(
    INSECURE_TRANSPORT=not os.getenv('DEBUG', False)
```

(continues on next page)

(continued from previous page)

```
request = Request(
    settings=settings,
    ...
)
```

4.3 Server & Database

4.4 Aiohttp

4.5 FastAPI

4.5.1 Installing

To install aioauth with FastAPI at the command line:

```
$ pip install aioauth[fastapi]
```

Usage example

```
from aioauth_fastapi.router import get_oauth2_router
from aioauth.storage import BaseStorage
from aioauth.config import Settings
from aioauth.server import AuthorizationServer
from fastapi import FastAPI
app = FastAPI()
class Storage(BaseStorage):
    Storage methods must be implemented here.
storage = Storage()
authorization_server = AuthorizationServer(storage)
# NOTE: Redefinition of the default aloauth settings
# INSECURE_TRANSPORT must be enabled for local development only!
settings = Settings(
    INSECURE_TRANSPORT=True,
# Include FastAPI router with oauth2 endpoints.
app.include_router(
   get_oauth2_router(authorization_server, settings),
   prefix="/oauth2",
   tags=["oauth2"],
```

4.6 Collections

```
from aloauth import collections
```

Collections that are used throughout the project.

class HTTPHeaderDict (dict=None, /, **kwargs)

Parameters

- headers An iterable of field-value pairs. Must not contain multiple field names when compared case-insensitively.
- kwargs Additional field-value pairs to pass in to dict.update.

A dict like container for storing HTTP Headers.

Example:

```
from aioauth.collections import HTTPHeaderDict
d = HTTPHeaderDict({"hello": "world"})
d['hello'] == 'world' # >>> True
d['Hello'] == 'world' # >>> True
d['hEllo'] == 'world' # >>> True
```

4.7 Config

```
from aioauth import config
```

Configuration settings for aioauth server instance.

class Settings ($TOKEN_EXPIRES_IN$: int = 86400, $REFRESH_TOKEN_EXPIRES_IN$: int = 172800, $AUTHORIZATION_CODE_EXPIRES_IN$: int = 300, $INSECURE_TRANSPORT$: bool = False, $ERROR_URI$: str = ", AVAILABLE: bool = True)

Configuration options that is used by the Server class.

Create new instance of Settings(TOKEN_EXPIRES_IN, REFRESH_TOKEN_EXPIRES_IN, AUTHORIZATION_CODE_EXPIRES_IN, INSECURE_TRANSPORT, ERROR_URI, AVAILABLE)

TOKEN EXPIRES_IN: int

Access token lifetime in seconds. Defaults to 24 hours.

REFRESH_TOKEN_EXPIRES_IN: int

Refresh token lifetime in seconds. Defaults to TOKEN_EXPIRES_IN * 2 (48 hours).

AUTHORIZATION CODE EXPIRES IN: int

Authorization code lifetime in seconds. Defaults to 5 minutes.

INSECURE TRANSPORT: bool

Allow connections over SSL only.

Note: When this option is disabled server will raise "HTTP method is not allowed" error when attempting to access the server without a valid SSL tunnel.

4.6. Collections

```
ERROR URI: str
```

URI to redirect resource owner when server encounters error.

AVAILABLE: bool

Boolean indicating whether or not the server is available.

4.8 Constances

```
from aioauth import constances
```

Constants that are used throughout the project.

```
default_headers = {'content-type': 'application/json', 'cache-control': 'no-store', 'practive authorization server must include the HTTP Cache-Control response header field, as per RFC2616, with a value of no-store in any response containing tokens, credentials, or other sensitive information, as well as the Pragma response header field, as per RFC2616, with a value of no-cache.
```

4.9 Errors

```
from aloauth import errors
```

Errors used throughout the project.

```
exception MethodNotAllowedError(request:
                                                         aioauth.requests.Request.
                                                                                      descrip-
                                                 Optional[str]
                                                                    None,
                                        tion:
                                                                                          Op-
                                        tional[aioauth.collections.HTTPHeaderDict] = None)
     The request is valid, but the method trying to be accessed is not available to the resource owner.
                      str = 'HTTP method is not allowed.'
     description:
     status_code:
                      http.HTTPStatus = 405
     error: aioauth.types.ErrorType = 'method_is_not_allowed'
exception InvalidRequestError (request: aioauth.requests.Request, description: Optional[str] =
                                     None, headers: Optional[aioauth.collections.HTTPHeaderDict] =
```

The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed.

Client authentication failed (e.g. unknown client, no client authentication included, or unsupported authentication method). The authorization server may return an HTTP 401 (Unauthorized) status code to indicate which HTTP authentication schemes are supported. If the client attempted to authenticate via the Authorization request header field, the authorization server must respond with an HTTP 401 (Unauthorized) status code, and include the WWW-Authenticate response header field matching the authentication scheme used by the client.

```
status code: http.HTTPStatus = 401
exception InsecureTransportError (request:
                                                           aioauth.requests.Request,
                                                                                        descrip-
                                          tion:
                                                   Optional[str] = None,
                                                                               headers:
                                                                                            Op-
                                          tional[aioauth.collections.HTTPHeaderDict] = None)
     An exception will be thrown if the current request is not secure.
                       str = 'OAuth 2 MUST utilize https.'
     description:
     error: aioauth.types.ErrorType = 'insecure_transport'
exception UnsupportedGrantTypeError(request:
                                                             aioauth.requests.Request,
                                                                                        descrip-
                                              tion:
                                                      Optional[str] = None,
                                                                               headers:
                                                                                            Op-
                                              tional[aioauth.collections.HTTPHeaderDict] = None)
     The authorization grant type is not supported by the authorization server.
               aioauth.types.ErrorType = 'unsupported_grant_type'
exception UnsupportedResponseTypeError(request:
                                                              aioauth.requests.Request,
                                                                                        descrip-
                                                         Optional[str] = None, headers:
                                                  tion:
                                                                                            Op-
                                                  tional[aioauth.collections.HTTPHeaderDict]
                                                  None)
     The authorization server does not support obtaining an authorization code using this method.
     error: aioauth.types.ErrorType = 'unsupported_response_type'
exception InvalidGrantError (request: aioauth.requests.Request, description: Optional[str] =
                                   None, headers: Optional[aioauth.collections.HTTPHeaderDict] =
                                   None)
     The provided authorization grant (e.g. authorization code, resource owner credentials) or refresh token is invalid,
     expired, revoked, does not match the redirection URI used in the authorization request, or was issued to another
     client.
     See RFC6749 section 5.2.
     error: aioauth.types.ErrorType = 'invalid_grant'
exception MismatchingStateError(request:
                                                           aioauth.requests.Request,
                                                                                        descrip-
                                                                     None,
                                                   Optional[str]
                                                                 =
                                                                              headers:
                                                                                            Op-
                                         tional[aioauth.collections.HTTPHeaderDict] = None)
     Unable to securely verify the integrity of the request and response.
     description: str = 'CSRF Warning! State not equal in request and response.'
     error: aioauth.types.ErrorType = 'mismatching_state'
exception UnauthorizedClientError(request:
                                                            aioauth.requests.Request,
                                                                                        descrip-
                                                    Optional[str] = None,
                                                                               headers:
                                                                                            Op-
                                           tional/aioauth.collections.HTTPHeaderDict1 = None)
     The authenticated client is not authorized to use this authorization grant type.
               aioauth.types.ErrorType = 'unauthorized_client'
exception InvalidScopeError (request: aioauth.requests.Request, description: Optional[str] =
                                   None, headers: Optional[aioauth.collections.HTTPHeaderDict] =
                                   None)
     The requested scope is invalid, unknown, or malformed, or exceeds the scope granted by the resource owner.
     See RFC6749 section 5.2.
     error: aioauth.types.ErrorType = 'invalid_scope'
exception ServerError (request: aioauth.requests.Request, description: Optional[str] = None, head-
                            ers: Optional[aioauth.collections.HTTPHeaderDict] = None)
     The authorization server encountered an unexpected condition that prevented it from fulfilling the request. (This
```

4.9. Errors 13

error code is needed because a HTTP 500 (Internal Server Error) status code cannot be returned to the client via a HTTP redirect.)

```
error: aioauth.types.ErrorType = 'server_error'

exception TemporarilyUnavailableError(request: aioauth.requests.Request, description: Optional[str] = None, headers: Optional[aioauth.collections.HTTPHeaderDict] = None)
```

The authorization server is currently unable to handle the request due to a temporary overloading or maintenance of the server. (This error code is needed because a HTTP 503 (Service Unavailable) status code cannot be returned to the client via a HTTP redirect.)

error: aioauth.types.ErrorType = 'temporarily_unavailable'

4.10 Grant Type

```
from aioauth import grant_type
```

Different OAuth 2.0 grant types.

class GrantTypeBase (*storage*: aioauth.storage.BaseStorage, *client_id*: *str*, *client_secret*: *str*) Base grant type that all other grant types inherit from.

```
async create_token_response (request: aioauth.requests.Request, client: aioauth.models.Client) \rightarrow aioauth.responses.TokenResponse Creates token response to reply to client.
```

async validate_request (request: aioauth.requests.Request) \rightarrow aioauth.models.Client Validates the client request to ensure it is valid.

The Authorization Code grant type is used by confidential and public clients to exchange an authorization code for an access token. After the user returns to the client via the redirect URL, the application will get the authorization code from the URL and use it to request an access token. It is recommended that all clients use RFC 7636 Proof Key for Code Exchange extension with this flow as well to provide better security.

Note: Note that aloauth implements RFC 7636 out-of-the-box. See RFC 6749 section 1.3.1.

```
async validate_request (request: aioauth.requests.Request) \rightarrow aioauth.models.Client Validates the client request to ensure it is valid.
```

```
async create_token_response (request: aioauth.requests.Request, client: aioauth.models.Client) \rightarrow aioauth.responses.TokenResponse Creates token response to reply to client.
```

```
class PasswordGrantType (storage: aioauth.storage.BaseStorage, client_id: str, client_secret: str)
```

The Password grant type is a way to exchange a user's credentials for an access token. Because the client application has to collect the user's password and send it to the authorization server, it is not recommended that this grant be used at all anymore. See RFC 6749 section 1.3.3. The latest OAuth 2.0 Security Best Current Practice disallows the password grant entirely.

```
async validate_request (request: aioauth.requests.Request) \rightarrow aioauth.models.Client Validates the client request to ensure it is valid.
```

The Refresh Token grant type is used by clients to exchange a refresh token for an access token when the access token has expired. This allows clients to continue to have a valid access token without further interaction with the user. See RFC 6749 section 1.5.

```
async create_token_response (request: aioauth.requests.Request, client: aioauth.models.Client) \rightarrow aioauth.responses.TokenResponse Validate token request and create token response.
```

```
async validate_request (request: aioauth.requests.Request) \rightarrow aioauth.models.Client Validates the client request to ensure it is valid.
```

The Client Credentials grant type is used by clients to obtain an access token outside of the context of a user. This is typically used by clients to access resources about themselves rather than to access a user's resources. See RFC 6749 section 4.4.

4.11 Models

```
from aioauth import models
```

Memory objects used throughout the project.

Create new instance of Client(client_id, client_secret, grant_types, response_types, redirect_uris, scope, user)

client id: str

Public identifier for the client. It must also be unique across all clients that the authorization server handles.

client_secret: str

Client secret is a secret known only to the client and the authorization server. Used for secure communication between the client and authorization server.

```
grant_types: List[aioauth.types.GrantType]
```

The method(s) in which an application gets an access token from the provider. Each grant type is optimized for a particular use case, whether that's a web app, a native app, a device without the ability to launch a web browser, or server-to-server applications.

```
response_types: List[aioauth.types.ResponseType]
```

A list containing the types of the response expected.

```
redirect uris: List[str]
```

After a user successfully authorizes an application, the authorization server will redirect the user back to the application with either an authorization code or access token in the URL. Because the redirect URL will contain sensitive information, it is critical that the service doesn't redirect the user to arbitrary locations.

scope: str

Scope is a mechanism that limit an application's access to a user's account. An application can request one or more scopes, this information is then presented to the user in the consent screen, and the access token issued to the application will be limited to the scopes granted.

4.11. Models 15

user: Optional[Any]

The user who is the creator of the Client. This optional attribute can be useful if you are creating a server that can be managed by multiple users.

$check_redirect_uri (redirect_uri) \rightarrow bool$

Verifies passed redirect_uri is part of the Clients's redirect_uris list.

check_grant_type (*grant_type: Optional*[aioauth.types.GrantType]) → bool

Verifies passed grant_type is part of the client's grant_types list.

$\begin{tabular}{ll} \textbf{check_response_type} (response_type: Optional[Union[aioauth.types.ResponseType, str]])} \rightarrow \\ begin{tabular}{ll} begin{tabul$

Verifies passed response type is part of the client's response types list.

$get_allowed_scope(scope: str) \rightarrow str$

Returns the allowed scope given the passed scope.

Note: Note that the passed scope may contain multiple scopes seperated by a space character.

```
check\_scope(scope: str) \rightarrow bool
```

class AuthorizationCode (code, client_id, redirect_uri, response_type, scope, auth_time, expires_in,

code_challenge, code_challenge_method, nonce, user)

Create new instance of AuthorizationCode(code, client_id, redirect_uri, response_type, scope, auth_time, expires_in, code_challenge, code_challenge_method, nonce, user)

code: str

Authorization code that the client previously received from the authorization server.

client_id: str

Public identifier for the client. It must also be unique across all clients that the authorization server handles.

redirect_uri: str

After a user successfully authorizes an application, the authorization server will redirect the user back to the application with either an authorization code or access token in the URL. Because the redirect URL will contain sensitive information, it is critical that the service doesn't redirect the user to arbitrary locations.

response_type: str

A string containing the type of the response expected.

scope: str

Scope is a mechanism that limit an application's access to a user's account. An application can request one or more scopes, this information is then presented to the user in the consent screen, and the access token issued to the application will be limited to the scopes granted.

auth_time: int

JSON Web Token Claim indicating the time when the authentication occurred.

expires_in: int

Time delta in which authorization_code will expire.

code_challenge: Optional[str]

Only used when RFC 7636, Proof Key for Code Exchange, is used. PKCE works by having the app generate a random value at the beginning of the flow called a Code Verifier. The app hashes the Code Verifier and the result is called the Code Challenge. The app then kicks off the flow in the normal way, except that it includes the Code Challenge in the query string for the request to the Authorization Server.

code_challenge_method: Optional[str]

Only used when RFC 7636, Proof Key for Code Exchange, is used. Method used to transform the code verifier into the code challenge.

nonce: Optional[str]

Only used when RFC 7636, Proof Key for Code Exchange, is used. Random piece of data.

user: Optional[Any]

The user who owns the AuthorizationCode.

 $check_code_challenge(code_verifier: str) \rightarrow bool$

property is_expired

Checks if the authorization code has expired.

class Token (access_token, refresh_token, scope, issued_at, expires_in, refresh_token_expires_in, client_id, token_type, revoked, user)

Create new instance of Token(access_token, refresh_token, scope, issued_at, expires_in, refresh_token_expires_in, client_id, token_type, revoked, user)

access_token: str

Token that clients use to make API requests on behalf of the resource owner.

refresh_token: str

Token used by clients to exchange a refresh token for an access token when the access token has expired.

scope: str

Scope is a mechanism that limit an application's access to a user's account. An application can request one or more scopes, this information is then presented to the user in the consent screen, and the access token issued to the application will be limited to the scopes granted.

issued at: int

Time date in which token was issued at.

expires_in: int

Time delta in which token will expire.

refresh_token_expires_in: int

Time delta in which refresh token will expire.

client_id: str

Public identifier for the client. It must also be unique across all clients that the authorization server handles.

token_type: str

Type of token expected.

revoked: bool

Flag that indicates whether or not the token has been revoked.

user: Optional[Any]

The user who owns the Token.

property is_expired

Checks if the token has expired.

property refresh_token_expired

Checks if refresh token has expired.

4.11. Models 17

4.12 Requests

```
from aioauth import requests
Request objects used throughout the project.
class Query (client_id: Optional[str] = None, redirect_uri: str = ", response_type: Optional[str] = None,
               state: str = ", scope: str = ", nonce: Optional[str] = None, code_challenge_method: Op-
               tional[aioauth.types.CodeChallengeMethod] = None, code_challenge: Optional[str] = None,
               response mode: Optional[aioauth.types.ResponseMode] = None)
     Object that contains a client's query string portion of a request. Read more on query strings here.
     Create new instance of Query(client_id, redirect_uri,
                                                                 response_type,
                                                                                  state,
                                                                                          scope,
                                                                                                  nonce.
     code_challenge_method, code_challenge, response_mode)
     client_id: Optional[str]
          Alias for field number 0
     redirect uri:
          Alias for field number 1
     response_type: Optional[str]
          Alias for field number 2
     state:
               str
          Alias for field number 3
     scope:
               str
          Alias for field number 4
     nonce:
               Optional[str]
          Alias for field number 5
     code_challenge_method:
                                     Optional[aioauth.types.CodeChallengeMethod]
          Alias for field number 6
     code_challenge: Optional[str]
          Alias for field number 7
                          Optional[aioauth.types.ResponseMode]
     response_mode:
          Alias for field number 8
class Post (grant_type: Optional[aioauth.types.GrantType] = None, client_id: Optional[str] = None,
              client_secret: Optional[str] = None, redirect_uri: Optional[str] = None, scope: str = ", user-
              name: Optional[str] = None, password: Optional[str] = None, refresh_token: Optional[str]
              = None, code: Optional[str] = None, token: Optional[str] = None, token type hint: Op-
              tional[str] = None, code verifier: Optional[str] = None)
     Object that contains a client's post request portion of a request. Read more on post requests here.
     Create new instance of Post(grant_type, client_id, client_secret, redirect_uri, scope, username, password, re-
     fresh token, code, token, token type hint, code verifier)
     grant_type: Optional[aioauth.types.GrantType]
          Alias for field number 0
     client id:
                    Optional[str]
          Alias for field number 1
```

client_secret: Optional[str]

Alias for field number 2

```
redirect_uri: Optional[str]
         Alias for field number 3
     scope:
              str
         Alias for field number 4
     username: Optional[str]
         Alias for field number 5
     password: Optional[str]
         Alias for field number 6
     refresh_token: Optional[str]
         Alias for field number 7
     code: Optional[str]
         Alias for field number 8
     token:
              Optional[str]
         Alias for field number 9
     token type hint: Optional[str]
         Alias for field number 10
     code_verifier: Optional[str]
         Alias for field number 11
class Request (method: aioauth.types.RequestMethod, headers: aioauth.collections.HTTPHeaderDict
                 = {}, query: aioauth.requests.Query = Query(client_id=None, redirect_uri=", re-
                 sponse type=None, state=", scope=", nonce=None, code challenge method=None,
                 code_challenge=None, response_mode=None), post:
                                                                       aioauth.requests.Post =
                 Post(grant_type=None,
                                       client_id=None,
                                                        client_secret=None,
                                                                            redirect_uri=None,
                 scope=", username=None, password=None, refresh_token=None, code=None, to-
                 ken=None, token_type_hint=None, code_verifier=None), url: str = ", user: Optional[Any]
                 = None, settings: aioauth.config.Settings = Settings(TOKEN_EXPIRES_IN=86400, RE-
                 FRESH_TOKEN_EXPIRES_IN=172800, AUTHORIZATION_CODE_EXPIRES_IN=300,
                 INSECURE_TRANSPORT=False, ERROR_URI=", AVAILABLE=True))
     Object that contains a client's complete request.
     Create new instance of Request(method, headers, query, post, url, user, settings)
     method: aioauth.types.RequestMethod
         Alias for field number 0
     headers:
                 aioauth.collections.HTTPHeaderDict
         Alias for field number 1
     query:
              aioauth.requests.Query
         Alias for field number 2
            aioauth.requests.Post
         Alias for field number 3
```

4.12. Requests

url: str

settings:

Alias for field number 4

Alias for field number 6

aioauth.config.Settings

user: Optional[Any]
Alias for field number 5

4.13 Response Type

```
from aioauth import responses
Response objects used throughout the project.
class ResponseTypeBase (storage: aioauth.storage.BaseStorage)
     Base response type that all other exceptions inherit from.
     async validate_request (request: aioauth.requests.Request) → aioauth.models.Client
class ResponseTypeToken (storage: aioauth.storage.BaseStorage)
     Response type that contains a token.
     async create_authorization_response(request:
                                                                           aioauth.requests.Request,
                                                    client:
                                                                     aioauth.models.Client)
                                                    aioauth.responses.TokenResponse
class ResponseTypeAuthorizationCode (storage: aioauth.storage.BaseStorage)
     Response type that contains an authorization code.
                                                                           aioauth.requests.Request,
     async create_authorization_response(request:
                                                                     aioauth.models.Client)
                                                    aioauth.responses.AuthorizationCodeResponse
class ResponseTypeIdToken (storage: aioauth.storage.BaseStorage)
     async validate_request (request: aioauth.requests.Request) → aioauth.models.Client
                                                                           aioauth.requests.Request,
     async create_authorization_response(request:
                                                                     aioauth.models.Client)
                                                    aioauth.responses.IdTokenResponse
class ResponseTypeNone (storage: aioauth.storage.BaseStorage)
     async create_authorization_response(request:
                                                                           aioauth.requests.Request,
                                                    client:
                                                                     aioauth.models.Client)
                                                    aioauth.responses.NoneResponse
```

4.14 Responses

```
Response objects used throughout the project.

class ErrorResponse (error: aioauth.types.ErrorType, description: str, error_uri: str = ")
Response for errors.

Create new instance of ErrorResponse(error, description, error_uri)

error: aioauth.types.ErrorType
Alias for field number 0

description: str
Alias for field number 1
```

```
error uri: str
         Alias for field number 2
class AuthorizationCodeResponse(code: str, scope: str)
     Response for authorization_code.
     Used by aioauth.response_type.ResponseTypeAuthorizationCode.
     Create new instance of AuthorizationCodeResponse(code, scope)
     code:
             str
         Alias for field number 0
     scope: str
         Alias for field number 1
class NoneResponse
     Response for aioauth.response_type.ResponseTypeNone.
     See: OAuth v2 multiple response types,
     Create new instance of NoneResponse()
class TokenResponse (expires_in: int, refresh_token_expires_in: int, access_token: str, refresh_token:
                         str, scope: str, token_type: str = 'Bearer')
     Response for valid token.
     Used by aioauth.response_type.ResponseTypeToken.
     Create new instance of TokenResponse(expires_in, refresh_token_expires_in, access_token, refresh_token,
     scope, token_type)
     expires in: int
         Alias for field number 0
     refresh_token_expires_in:
                                       int
         Alias for field number 1
     access_token:
                       str
         Alias for field number 2
     refresh token: str
         Alias for field number 3
     scope:
               str
         Alias for field number 4
     token_type: str
         Alias for field number 5
class IdTokenResponse(id_token: str)
     Response for OpenID id token.
     Used by aioauth.response_type.ResponseResponseTypeIdTokenTypeToken.
     Create new instance of IdTokenResponse(id_token,)
     id_token: str
         Alias for field number 0
class TokenActiveIntrospectionResponse(scope:
                                                          str.
                                                                client id:
                                                                             str.
                                                                                  token type:
                                                aioauth.types.TokenType, expires_in: int, active:
                                                bool = True
     Response for a valid access token.
     Used by aioauth.server.AuthorizationServer.create_token_introspection_response().
```

4.14. Responses 21

```
Create new instance of TokenActiveIntrospectionResponse(scope, client_id, token_type, expires_in, active)
```

scope: str

Alias for field number 0

client id: str

Alias for field number 1

token_type: aioauth.types.TokenType

Alias for field number 2

expires_in: int

Alias for field number 3

active: bool

Alias for field number 4

class TokenInactiveIntrospectionResponse (active: bool = False)

For an invalid, revoked or expired token.

Used by aioauth.server.AuthorizationServer.create_token_introspection_response().

Create new instance of TokenInactiveIntrospectionResponse(active,)

active: bool

Alias for field number 0

class Response (content: Dict = {}, status_code: http.HTTPStatus = <HTTPStatus.OK: 200>, headers: aioauth.collections.HTTPHeaderDict = {'content-type': 'application/json', 'cache-

control': 'no-store', 'pragma': 'no-cache'})

General response class.

Used by aioauth.server.AuthorizationServer.

Create new instance of Response(content, status_code, headers)

content: Dict

Alias for field number 0

status_code: http.HTTPStatus

Alias for field number 1

headers: aioauth.collections.HTTPHeaderDict

Alias for field number 2

4.15 Server

from aioauth import server

Memory object and interface used to initialize an OAuth2.0 server instance.

Warning: Note that *aioauth.server.AuthorizationServer* is not depedent on any server framework, nor serves at any specific endpoint. Instead, it is used to create an interface that can be used in conjunction with a server framework like FastAPI or aiohttp to create a fully functional OAuth 2.0 server. Check out the *Examples* portion of the documentation to understand how it can be leveraged in your own project.

```
class AuthorizationServer (storage: aioauth.storage.BaseStorage, response_types: Optional[Dict]

= None, grant_types: Optional[Dict] = None)

Interface for initializing an OAuth 2.0 server.

response_types = {<ResponseType.TYPE_CODE: 'code'>: <class 'aioauth.response_type.Resgrant_types = {<GrantType.TYPE_AUTHORIZATION_CODE: 'authorization_code'>: <class 'aio
is_secure_transport (request: aioauth.requests.Request) → bool

Verifies the request was sent via a protected SSL tunnel.
```

Note: This method simply checks if the request URL contains https:// at the start of it. It does **not** ensure if the SSL certificate is valid.

```
Parameters request - aioauth.requests.Request object.
```

Returns Flag representing whether or not the transport is secure.

Returns a response object with introspection of the passed token. For more information see RFC7662 section 2.1.

Note: The API endpoint that leverages this function is usually /introspect.

Example

Below is an example utilizing FastAPI as the server framework.

```
from aioauth_fastapi.utils import to_oauth2_request, to_fastapi_response

@app.get("/token/introspect")
async def introspect(request: fastapi.Request) -> fastapi.Response:
    # Converts a fastapi.Request to an aioauth.Request.
    oauth2_request: aioauth.Request = await to_oauth2_request(request)
    # Creates the response via this function call.
    oauth2_response: aioauth.Response = await server.create_token_
    introspection_response(oauth2_request)
    # Converts an aioauth.Response to a fastapi.Response.
    response: fastapi.Response = await to_fastapi_response(oauth2_response)
    return response
```

Parameters request - An aioauth.requests.Request object.

Returns An aioauth.responses.Response object.

Return type response

 $\texttt{get_client_credentials} \ (\textit{request:} \ aioauth.requests.Request) \ \rightarrow Tuple[\textit{str}, \textit{str}]$

4.15. Server 23

```
async create_token_response(request: aioauth.requests.Request) → aioauth.responses.Response
```

Endpoint to obtain an access and/or ID token by presenting an authorization grant or refresh token. Validates a token request and creates a token response. For more information see RFC6749 section 4.1.3.

Note: The API endpoint that leverages this function is usually /token.

Example

Below is an example utilizing FastAPI as the server framework.

```
from aioauth_fastapi.utils import to_oauth2_request, to_fastapi_response

@app.post("/token")
async def token(request: fastapi.Request) -> fastapi.Response:
    # Converts a fastapi.Request to an aioauth.Request.
    oauth2_request: aioauth.Request = await to_oauth2_request(request)
    # Creates the response via this function call.
    oauth2_response: aioauth.Response = await server.create_token_
    →response(oauth2_request)
    # Converts an aioauth.Response to a fastapi.Response.
    response: fastapi.Response = await to_fastapi_response(oauth2_response)
    return response
```

Parameters request - An aioauth.requests.Request object.

Returns An aioauth.responses.Response object.

Return type response

Endpoint to interact with the resource owner and obtain an authorization grant. Validate authorization request and create authorization response. For more information see RFC6749 section 4.1.1.

Note: The API endpoint that leverages this function is usually /authorize.

Example

Below is an example utilizing FastAPI as the server framework.

```
from aioauth.fastapi.utils import to_oauth2_request, to_fastapi_response

@app.post("/authorize")
async def authorize(request: fastapi.Request) -> fastapi.Response:
    # Converts a fastapi.Request to an aioauth.Request.
    oauth2_request: aioauth.Request = await to_oauth2_request(request)
    # Creates the response via this function call.
    oauth2_response: aioauth.Response = await server.create_authorization_
    →response(oauth2_request)
    # Converts an aioauth.Response to a fastapi.Response.
```

(continues on next page)

(continued from previous page)

response: fastapi.Response = await to_fastapi_response(oauth2_response)
return response

Parameters request - An aioauth.requests.Request object.

Returns An aioauth.responses.Response object.

Return type response

4.16 Storage

```
from aioauth import storage
```

Storage helper class for storing and retrieving client and resource owner information. See the examples on the sidebar to view this in action.

class BaseStorage

async create_token (request: aioauth.requests.Request, client_id: str, scope: str, access_token: $str, refresh_token: str) \rightarrow aioauth.models.Token$ Generates a user token and stores it in the database.

Warning: Generated token *must* be stored in the database.

Note: Method is used by all core grant types, but only used for aioauth.response_type. ResponseTypeToken.

Parameters

- request An aioauth.requests.Request.
- client_id A user client ID.
- scope The scopes for the token.

Returns The new generated aioauth.models.Token.

async get_token (request: aioauth.requests.Request, client_id: str, token_type: Optional[str] = <TokenType.REFRESH: 'refresh_token'>, access_token: Optional[str] = None, refresh_token: Optional[str] = None) \rightarrow Optional[aioauth.models.Token] Gets existing token from the database.

Note: Method is used by aioauth.server.AuthorizationServer, and by the grant type aioauth.grant_types.RefreshTokenGrantType.

Parameters

• request - An aioauth.requests.Request.

4.16. Storage 25

- client id A user client ID.
- access token The user access token.
- refresh_token The user refresh token.

Returns An optional aioauth.models.Token object.

```
async create_authorization_code (request: aioauth.requests.Request, client_id: str, scope: str, response_type: str, redirect_uri: str, code_challenge_method: Optional[str], code_challenge: Optional[str], code: str) \rightarrow aioauth.models.AuthorizationCode
```

Generates an authorization token and stores it in the database.

Warning: Generated authorization token *must* be stored in the database.

Note: This must is used by the response type aioauth.respose_type. ResponseTypeAuthorizationCode.

Parameters

- request An aioauth.requests.Request.
- client_id A user client ID.
- scope The scopes for the token.
- response_type An aioauth.types.ResponseType.
- redirect_uri The redirect URI.
- $\bullet \ \ \textbf{code_challenge_method} An \ \textit{aioauth.types.CodeChallengeMethod}. \\$
- code_challenge Code challenge string.

Returns An aioauth.models.AuthorizationCode object.

```
async get_id_token (request: aioauth.requests.Request, client_id: str, scope: str, response_type: str, redirect\_uri: str, nonce: str) \rightarrow str
Returns an id_token. For more information see OpenID Connect Core 1.0 incorporating errata set 1 section 2.
```

Note: Method is used by response type aioauth.response_type.ResponseTypeIdToken

```
async get_client (request: aioauth.requests.Request, client_id: str, client_secret: Optional[str] = None) \rightarrow Optional[aioauth.models.Client] Gets existing client from the database if it exists.
```

Warning: If client does not exists in database this method *must* return None to indicate to the validator that the requested client_id does not exist or is invalid.

Note: This method is used by all core grant types, as well as all core response types.

Parameters

- request An aioauth.requests.Request.
- client id A user client ID.
- client_secret An optional user client secret.

Returns An optional aioauth.models.Client object.

async authenticate (request: aioauth.requests.Request) \rightarrow bool Authenticates a user.

Note: This method is used by the grant type aioauth.grant_type.PasswordGrantType.

Parameters request - An aioauth.requests.Request.

Returns Boolean indicating whether or not the user was authenticated successfully.

async get_authorization_code (request: aioauth.requests.Request, client_id: str, code: str) \rightarrow Optional[aioauth.models.AuthorizationCode] Gets existing authorization code from the database if it exists.

Warning: If authorization code does not exists this function *must* return None to indicate to the validator that the requested authorization code does not exist or is invalid.

Note: This method is used by the grant type aioauth.grant_type. AuthorizationCodeGrantType.

Parameters

- request An aioauth.requests.Request.
- client id A user client ID.
- code An authorization code.

Returns An optional aioauth.models.AuthorizationCode.

async delete_authorization_code (request: aioauth.requests.Request, client_id: str, code: $str) \rightarrow None$ Deletes authorization code from database.

Note: This method is used by the grant type aioauth.grant_type.

AuthorizationCodeGrantType.

Parameters

• request - An aioauth.requests.Request.

4.16. Storage 27

- client id A user client ID.
- **code** An authorization code.

async revoke_token (request: aioauth.requests.Request, $refresh_token$: $str) \rightarrow None$ Revokes a token's from the database.

Note: This method *must* set revoked to True for an existing token record. This method is used by the grant type aloauth.grant_types.RefreshTokenGrantType.

Parameters

- request An aioauth.requests.Request.
- refresh_token The user refresh token.

4.17 Types

```
from aloauth import types
```

Containers that contain constants used throughout the project.

```
class ErrorType(value)
    Error types.
    INVALID REQUEST = 'invalid request'
    INVALID_CLIENT = 'invalid_client'
    INVALID_GRANT = 'invalid_grant'
    INVALID_SCOPE = 'invalid_scope'
    UNAUTHORIZED_CLIENT = 'unauthorized_client'
    UNSUPPORTED_GRANT_TYPE = 'unsupported_grant_type'
    UNSUPPORTED_RESPONSE_TYPE = 'unsupported_response_type'
    INSECURE_TRANSPORT = 'insecure_transport'
    MISMATCHING_STATE = 'mismatching_state'
    METHOD_IS_NOT_ALLOWED = 'method_is_not_allowed'
    SERVER_ERROR = 'server_error'
    TEMPORARILY_UNAVAILABLE = 'temporarily_unavailable'
class GrantType (value)
    Grant types.
    TYPE_AUTHORIZATION_CODE = 'authorization_code'
    TYPE_PASSWORD = 'password'
    TYPE_CLIENT_CREDENTIALS = 'client_credentials'
    TYPE_REFRESH_TOKEN = 'refresh_token'
```

```
class ResponseType (value)
    Response types.
    TYPE TOKEN = 'token'
    TYPE CODE = 'code'
    TYPE NONE = 'none'
    TYPE_ID_TOKEN = 'id_token'
class RequestMethod(value)
    Request types.
    GET = 'GET'
    POST = 'POST'
class CodeChallengeMethod(value)
    Code challenge types.
    PLAIN = 'plain'
    $256 = '$256'
class ResponseMode(value)
    Response modes.
    MODE_QUERY = 'query'
    MODE_FORM_POST = 'form_post'
    MODE_FRAGMENT = 'fragment'
class TokenType(value)
    An enumeration.
    ACCESS = 'access_token'
    REFRESH = 'refresh_token'
```

4.18 Utils

```
from aioauth import utils
```

Contains helper functions that is used throughout the project that doesn't pertain to a specific file or module.

```
get_authorization_scheme_param (authorization_header_value: str) → Tuple[str, str]
    Retrieves the authorization schema parameters from the authorization header.

Parameters authorization_header_value - Value of the authorization header.

Returns Tuple of the format (scheme, param).

enforce_str(scope: List) → str
    Converts a list of scopes to a space separated string.
```

Note: If a string is passed to this method it will simply return an empty string back. Use <code>enforce_list()</code> to convert strings to scope lists.

4.18. Utils 29

Parameters scope – An iterable or string that contains a list of scope.

Returns A string of scopes seperated by spaces.

Raises TypeError - The scope value passed is not of the proper type.

enforce_list (scope: Optional[Union[str, List, Set, Tuple]]) \rightarrow List Converts a space separated string to a list of scopes.

Note: If an iterable is passed to this method it will return a list representation of the iterable. Use <code>enforce_str()</code> to convert iterables to a scope string.

Parameters scope – An iterable or string that contains scopes.

Returns A list of scopes.

 $\label{eq:generate_token} \textbf{generate_token} \ (\textit{length:} \quad \textit{int} = 30, \quad \textit{chars:} \quad \textit{str} = 'abcdefghijklmnopqrstuvwxyzABCDEFGHI-JKLMNOPQRSTUVWXYZ0123456789'}) \rightarrow \textit{str}$

Generates a non-guessable ÕAuth token. OAuth (1 and 2) does not specify the format of tokens except that they should be strings of random characters. Tokens should not be guessable and entropy when generating the random characters is important. Which is why SystemRandom is used instead of the default random.choice method.

Parameters

- **length** Length of the generated token.
- **chars** The characters to use to generate the string.

Returns Random string of length length and characters in chars.

build_uri (url: str, query_params: Optional[Dict] = None, fragment: Optional[Dict] = None) → str Builds an URI string from passed url, query_params, and fragment.

Parameters

- **url** URL string.
- query_params Paramaters that contain the query.
- **fragment** Fragment of the page.

Returns URL containing the original url, and the added query_params and fragment.

encode_auth_headers ($client_id$: str, $client_secret$: str) \rightarrow aioauth.collections.HTTPHeaderDict Encodes the authentication header using base64 encoding.

Parameters

- client_id The client's id.
- client_secret The client's secret.

Returns A case insensitive dictionary that contains the Authorization header set to basic and the authorization header.

```
decode auth headers (authorization: str) → Tuple[str, str]
```

Decodes an encrypted HTTP basic authentication string. Returns a tuple of the form (client_id, client_secret), and raises a aioauth.errors.InvalidClientError exception if nothing could be decoded.

Parameters authorization – Authorization header string.

Returns Tuple of the form (client_id, client_secret).

Raises ValueError – Invalid authorization header string.

```
\verb|create_s256_code_challenge| (code\_verifier: str)| \rightarrow str|
```

Create S256 code challenge with the passed code_verifier.

Note: This function implements base64url(sha256(ascii(code_verifier))).

Parameters code_verifier - Code verifier string.

Returns Representation of the S256 code challenge with the passed code_verifier.

Decorator that adds error catching to the function passed.

Parameters f - A callable.

Returns A callable with error catching capabilities.

4.18. Utils 31

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

а

```
aioauth.collections, 11
aioauth.config, 11
aioauth.constances, 12
aioauth.errors, 12
aioauth.grant_type, 14
aioauth.models, 15
aioauth.requests, 18
aioauth.response_type, 20
aioauth.responses, 20
aioauth.server, 22
aioauth.storage, 25
aioauth.types, 28
aioauth.utils, 29
```

36 Python Module Index

INDEX

A	aioauth.responses), 21
ACCESS (TokenType attribute), 29	AuthorizationServer (class in aioauth.server), 22
access_token (<i>Token attribute</i>), 17	AVAILABLE (Settings attribute), 12
access_token (<i>TokenResponse attribute</i>), 21	D
active (TokenActiveIntrospectionResponse attribute),	В
22	BaseStorage (class in aioauth.storage), 25
active (TokenInactiveIntrospectionResponse attribute),	build_uri() (in module aioauth.utils), 30
22	
aioauth.collections	C
module, 11	<pre>catch_errors_and_unavailability() (in</pre>
aioauth.config	module aioauth.utils), 31
module, 11	<pre>check_code_challenge() (AuthorizationCode</pre>
aioauth.constances	method), 17
module, 12	check_grant_type() (Client method), 16
aioauth.errors	check_redirect_uri() (Client method), 16
module, 12	check_response_type() (Client method), 16
aioauth.grant_type	check_scope() (Client method), 16
module, 14	Client (class in aioauth.models), 15
aioauth.models	<pre>client_id (AuthorizationCode attribute), 16</pre>
module, 15	<pre>client_id (Client attribute), 15</pre>
aioauth.requests	client_id (Post attribute), 18
module, 18	client_id (<i>Query attribute</i>), 18
aioauth.response_type	client_id (<i>Token attribute</i>), 17
module, 20	<pre>client_id (TokenActiveIntrospectionResponse at-</pre>
aioauth.responses	tribute), 22
module, 20	client_secret (Client attribute), 15
aioauth.server	client_secret (Post attribute), 18
module, 22	ClientCredentialsGrantType (class in
aioauth.storage	aioauth.grant_type), 15
module, 25	code (AuthorizationCode attribute), 16
aioauth.types	code (AuthorizationCodeResponse attribute), 21
module, 28	code (<i>Post attribute</i>), 19
aioauth.utils	code_challenge (AuthorizationCode attribute), 16
module, 29	code_challenge (<i>Query attribute</i>), 18
auth_time (AuthorizationCode attribute), 16	<pre>code_challenge_method (AuthorizationCode at-</pre>
authenticate() (BaseStorage method), 27	tribute), 16
AUTHORIZATION_CODE_EXPIRES_IN (Settings at-	code_challenge_method (<i>Query attribute</i>), 18
tribute), 11	code_verifier (Post attribute), 19
AuthorizationCode (class in aioauth.models), 16	CodeChallengeMethod (class in aioauth.types), 29
AuthorizationCodeGrantType (class in	content (Response attribute), 22
aioauth.grant_type), 14	create_authorization_code() (BaseStorage
AuthorizationCodeResponse (class in	method), 26

create_authorization_response() (AuthorizationServer method), 24	error_uri (<i>ErrorResponse attribute</i>), 20 ERROR_URI (<i>Settings attribute</i>), 11
create_authorization_response() (Respon-	ErrorResponse (class in aioauth.responses), 20
seTypeAuthorizationCode method), 20	ErrorType (class in aioauth.types), 28
create_authorization_response() (Respon-	expires_in (AuthorizationCode attribute), 16
seTypeIdToken method), 20	expires_in (Token attribute), 17
create_authorization_response() (Respon-	expires_in (Token authority, 17 expires_in (TokenActiveIntrospectionResponse at-
seTypeNone method), 20	tribute), 22
create_authorization_response() (Respon-	expires_in (TokenResponse attribute), 21
seTypeToken method), 20	_
create_s256_code_challenge() (in module	G
aioauth.utils), 31	<pre>generate_token() (in module aioauth.utils), 30</pre>
create_token() (BaseStorage method), 25	GET (RequestMethod attribute), 29
<pre>create_token_introspection_response()</pre>	get_allowed_scope() (Client method), 16
(AuthorizationServer method), 23	get_authorization_code() (BaseStorage
<pre>create_token_response() (AuthorizationCode-</pre>	method), 27
GrantType method), 14	<pre>get_authorization_scheme_param() (in mod-</pre>
<pre>create_token_response() (AuthorizationServer</pre>	ule aioauth.utils), 29
method), 23	<pre>get_client() (BaseStorage method), 26</pre>
create_token_response() (GrantTypeBase method), 14	get_client_credentials() (Authorization- Server method), 23
create_token_response() (RefreshTokenGrant-	get_id_token() (BaseStorage method), 26
Type method), 15	get_token() (BaseStorage method), 25
VI	grant_type (Post attribute), 18
D	grant_types (AuthorizationServer attribute), 23
decode_auth_headers() (in module aioauth.utils),	grant_types (Client attribute), 15
30	GrantType (class in aioauth.types), 28
default_headers (in module aioauth.constances),	GrantTypeBase (class in aioauth.grant_type), 14
12	
delete_authorization_code() (BaseStorage	Н
method), 27	headers (Request attribute), 19
description (ErrorResponse attribute), 20	headers (Response attribute), 22
description (InsecureTransportError attribute), 13	HTTPHeaderDict (class in aioauth.collections), 11
description (MethodNotAllowedError attribute), 12	
description (MismatchingStateError attribute), 13	I
E	<pre>id_token (IdTokenResponse attribute), 21</pre>
	IdTokenResponse (class in aioauth.responses), 21
encode_auth_headers() (in module aioauth.utils),	INSECURE_TRANSPORT (ErrorType attribute), 28
30	INSECURE_TRANSPORT (Settings attribute), 11
enforce_list() (in module aioauth.utils), 30	InsecureTransportError, 13
enforce_str() (in module aioauth.utils), 29	INVALID_CLIENT (ErrorType attribute), 28
error (ErrorResponse attribute), 20	INVALID_GRANT (ErrorType attribute), 28
error (InsecureTransportError attribute), 13	INVALID_REQUEST (ErrorType attribute), 28
error (InvalidClientError attribute), 12	INVALID_SCOPE (ErrorType attribute), 28
error (InvalidGrantError attribute), 13	InvalidClientError, 12
error (InvalidRequestError attribute), 12	InvalidGrantError, 13
error (InvalidScopeError attribute), 13	InvalidRequestError, 12
error (MethodNotAllowedError attribute), 12	InvalidScopeError, 13
error (Mismatching State Error attribute), 13	<pre>is_expired() (AuthorizationCode property), 17</pre>
error (ServerError attribute), 14	is_expired() (Token property), 17
error (TemporarilyUnavailableError attribute), 14	is_secure_transport() (AuthorizationServer
error (UnauthorizedClientError attribute), 13	method), 23
error (UnsupportedGrantTypeError attribute), 13	issued_at (<i>Token attribute</i>), 17
error (UnsupportedResponseTypeError attribute) 13	

38 Index

METHOD_IS_NOT_ALLOWED (ErrorType attribute), 28 MismatchInngStatE (ErrorType attribute), 28 MismatchInngStatE (ErrorType attribute), 29 MODE_TORN_POST (ResponseMode attribute), 29 MODE_TORN_POST (ResponseMode attribute), 29 MODE_OLDETY (Respo	M	refresh_token_expires_in (<i>Token attribute</i>), 17
MELHOONDA AllowedError, 19 authority, 19 aut	method (Request attribute), 19	
aioauth.grant_type), 14 Request (class in aioauth.requests), 19 Response (class in aioauth.request), 12 aioauth.config, 11 aioauth.config, 11 aioauth.config, 11 aioauth.grant_type, 14 aioauth.grant_type, 14 aioauth.grant_type, 14 aioauth.grant_type, 14 aioauth.requests, 18 aioauth.requests, 18 aioauth.response_type, 20 aioauth.server, 22 aioauth.storage, 25 aioauth.types, 28 aioauth.utils, 29 Nonnee (AuthorizationCode attribute), 16 monee (AuthorizationCode attribute), 18 NonneResponse (class in aioauth.response_type), 20 ResponseTypeloxen (class in aioa	METHOD_IS_NOT_ALLOWED (ErrorType attribute), 28	
MSISMATCHINGSTATEFORM, 130 MODE_FRACEMENT (ResponseMode attribute), 29 MODE_GUERY (ResponseMode attribute), 29 module aloauth.collections, 11 aloauth.conting, 11 aloauth.constances, 12 aloauth.grant_type, 14 aloauth.models, 15 aloauth.response_Lype, 14 aloauth.responses_type, 20 aloauth.responses_type, 20 aloauth.responses, 20 aloauth.server, 22 aloauth.storage, 25 aloauth.tils, 29 Nonce (AuthorizationCode attribute), 16 monce (Query attribute), 18 NonceResponse (class in aloauth.responses), 21 P Password (Post attribute), 19 Password (Post attribute), 19 Password (Post attribute), 19 Password (Rost attribute), 19 Post (Cass in aloauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 QQ Query (class in aloauth.requests), 18 predirect_uri (AuthorizationCode attribute), 18 redirect_uri (Response attribute), 19 refresh_token (Foken attribute), 19 refresh_token (F	MethodNotAllowedError, 12	· _ · · · · · · · · · · · · · · ·
RequestMethod (class in aioauth.types), 29 Response (class in aioauth.type), 18 response_type (Query attribute), 29 response_type (Query attribute), 29 response_type (Qlass in aioauth.types), 29 ResponseType (Class in aioauth.types), 20 ResponseType (Class in aioauth.types), 29 ResponseType (Cla	MISMATCHING_STATE (ErrorType attribute), 28	
Response (class in aioauth.responses), 22 response (class in aioauth.responses), 22 response (class in aioauth.responses), 23 response (class in aioauth.responses), 23 response (class in aioauth.responses), 24 response type (Query attribute), 18 response type (Query attribute), 15 Response type (Client attribute), 15 Response type (Client attribute), 15 Response type (Lass in aioauth.response), 28 Response type (Client attribute), 15 Response type (Lass in aioauth.response), 28 Response type (AuthorizationCode (class in aioauth.response), 29 Response type (Lass in aioauth.response_type), 20 Response type (Ident attribute), 20 Response type (Lass in aioauth.response_type), 20 Response type (Ident attribute), 20 Response type (Ident attribute	MismatchingStateError, 13	
response_mode (Query attribute), 18 response_type (AuthorizationCode attribute), 16 response_type (AuthorizationCode attribute), 16 response_type (AuthorizationCode attribute), 18 response_type (AuthorizationCode attribute), 13 response_type (AuthorizationCode attribute), 13 response_type (AuthorizationCover attribute), 13 response_type (AuthorizationCover attribute), 13 response_type (AuthorizationCover attribute), 23 response_type (AuthorizationCover attribute), 24 response_type (AuthorizationCover attribute), 25 response_type (AuthorizationCover attribute), 26 response_type (AuthorizationCover attribute), 27 revoke_token (Over attribute), 18 response_type (AuthorizationCover attribute), 27 revoke_token (Ivent attribute), 18 response_type (AuthorizationCover attribute), 27 revoke_token (Ivent attribute), 18 response_type (AuthorizationCover attribute), 27 revoke_token (Ivent attribute), 18 response_type (AuthorizationCover attribute), 27 revoke_token (MODE_FORM_POST (ResponseMode attribute), 29	
response_type (AuthorizationCode attribute), 16 response_type (Query attribute), 18 response_types (Client attribute), 15 response_types (Client attribute), 23 response_types (Client attribute), 23 response_types (Client attribute), 23 response_types (Client attribute), 25 response_types (Client attribute), 26 response_types (Client attribute), 27 response_types (Client attribute), 28 response_types (Client attribute), 29 response_type (Client attribute), 20 response_type (Client attribu	MODE_FRAGMENT (ResponseMode attribute), 29	· · · · · · · · · · · · · · · · ·
aioauth.config. II aioauth.config. II aioauth.config. II aioauth.constances, I2 aioauth.errors, I2 aioauth.errors, I2 aioauth.errors, I2 aioauth.models, I5 aioauth.requests, I8 aioauth.requests, I8 aioauth.response_type, 20 aioauth.server, 22 aioauth.server, 22 aioauth.storage, 25 aioauth.tils, 29 N nonce (AuthorizationCode attribute), I6 nonce (Query attribute), I8 NoneResponse (class in aioauth.response_type), 20 ResponseTypeBatchorizationCode (class in aioauth.response_type), 20 ResponseTypeBase (class in aioauth.response_type), 20 ResponseTypeBase (class in aioauth.response_type), 20 ResponseTypeBase (class in aioauth.response_type), 20 ResponseTypeBose (class in aioauth.response_type), 20 ResponseTypeNone (class in aioauth.response_type), 20 ResponseTypeToken (class in aioauth.response_type), 20 ResponseTypeToken (class in aioauth.response_type), 20 ResponseTypeNone (class in aioauth.response_type), 20 ResponseTypeBose (class in aioauth.response_type), 20 ResponseTypeIoToken (class in aioauth.response_type), 20 ResponseType (class in aioauth.response_t	MODE_QUERY (ResponseMode attribute), 29	
aioauth.config.11 aioauth.config.11 aioauth.config.11 aioauth.config.11 aioauth.config.12 aioauth.grant_type,14 aioauth.grant_type,14 aioauth.models,15 aioauth.requests.18 aioauth.response_type,20 aioauth.responses,20 aioauth.storage,25 aioauth.types,28 aioauth.types,28 aioauth.types,28 aioauth.types,28 aioauth.types,28 aioauth.types,28 aioauth.types,28 aioauth.types,28 aioauth.types,28 aioauth.grantibute),16 monce (Query attribute), 18 PP password(Post attribute), 19 Password(Post attribute), 19 Post (Request attribute), 19 Post (R	module	
aloauth.constances, 12 aloauth.errors, 12 aloauth.grant_type, 14 aloauth.models, 15 aloauth.requests, 18 aloauth.response_type, 20 aloauth.responses, 20 aloauth.server, 22 aloauth.storage, 25 aloauth.storage, 26 ResponseTypeloauthorizationced (lass in aloauth.response_type), 20 ResponseTypeNone (class in aloauth.re	aioauth.collections,11	
aloauth.errors, 12 aloauth.grant_type, 14 aloauth.models, 15 aloauth.response_type, 20 aloauth.response_type, 20 aloauth.responses_type, 20 aloauth.server, 22 aloauth.storage, 25 aloauth.types, 28 aloauth.utils, 29 N monce (AuthorizationCode attribute), 16 monce (Query attribute), 18 NoneResponse (class in aloauth.responses), 21 P password (Post attribute), 19 PasswordGrantType (class in aloauth.responses), 21 P Post (class in aloauth.requests), 18 post (Request attribute), 19 Post (request AuthorizationCode attribute), 29 Post (class in aloauth.requests), 18 post (Request attribute), 19 Post (Request attribute), 19 R redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 19 refresh_token (Post attribute), 19 refresh_token (Token Reponery), 17 refresh_token (Token property), 17 refresh_token (Token Reponer attribute), 17 refresh_token (Token Reponer attribute), 17 refresh_token (Token Reponer attribute), 17 refresh_token (Post attri		
aloauth.grant_type, 14 aloauth.models, 15 aloauth.requests, 18 aloauth.response_type, 20 aloauth.server, 22 aloauth.storage, 25 aloauth.storage, 25 aloauth.types, 28 aloauth.utils, 29 Nonence (AuthorizationCode attribute), 16 monce (Query attribute), 18 NoneResponse (class in aloauth.responses), 21 PasswordGrantType (class in aloauth.responses), 21 Post (class in aloauth.requests), 18 post (Request attribute), 19 Prefiresh_token (Post attribute), 19 Prefresh_token (Token Response attribute), 17 REFFRESH_TOKEN_EXPIRES_IN (Settings attribute), 17 token(Post attribute), 17 Token(class in aloauth.response_type), 20 ResponseTypeBase (class in aloauth.response_type), 20 Resp		
aloauth.models, 15 aloauth.requests, 18 aloauth.response_type, 20 aloauth.server, 22 aloauth.server, 22 aloauth.server, 23 aloauth.server, 24 aloauth.server, 25 aloauth.server, 26 aloauth.server, 27 aloauth.server, 28 aloauth.utils, 29 Nonence (AuthorizationCode attribute), 16 monce (Query attribute), 18 NoneResponse (class in aloauth.responses), 21 P PasswordGrantType (class in aloauth.grant_type), 26 ELAIN (CodeChallengeMethod attribute), 29 Bost (class in aloauth.requests), 18 Bost (Request attribute), 19 POST (RequestMethod attribute), 29 CQ Query (class in aloauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 18 redirect_uri (Client attribute), 19 refresh_token (Token attribute), 19 refresh_token (Token attribute), 19 refresh_token (Token attribute), 21 refresh_token (Token attribute), 21 refresh_token (Token Response attribute), 21 refresh_token (Token Response attribute), 21 refresh_token (Token attribute), 17 refresh_token (Token attribute), 19 refresh_token (Token attribute), 10 refresh_token (Token attribute), 11 refresh_token (Post attribute), 12 refresh_token (Post attribute), 12 refresh_token (Post attribute), 13 refresh_token (Post attribute), 14 refresh_token (Post attribute), 15 refresh_token (Post attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Post attribute), 19 refresh_token (Post attribute), 19 refresh_token (Post attribute), 10 refresh_		
aloauth.response_type, 20 aloauth.response_type, 20 aloauth.server, 22 aloauth.storage, 25 aloauth.types, 28 aloauth.types, 29 aloauth.types, 28 aloauth.types, 29 aloauth.types, 28 aloauth.types, 29 aloauth.types, 28 aloauth.response_type, 20 responseTypeToken (class in aloauth.response_type), 20 aloauth.response_type, 20 revoke_token() (BaseStorage method), 28 revoked (Token attribute), 19 scope (Cuery attribute), 15 scope (Clein attribute), 19 scope (Guery attribute), 19 scope (Token attribute), 19 scope (Token attribute), 19 settings (Request attribute), 19 state (Query attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (MethodNotAllowedError, 14 TemporarilyUnavailableError, 14 Token (class in		
ResponseTypeBase (class in aioauth.response_type), 20 aioauth.server, 22 aioauth.storage, 25 aioauth.ttypes, 28 aioauth.utils, 29 N nonce (AuthorizationCode attribute), 16 nonce (Query attribute), 18 NoneResponse (class in aioauth.response), 20 Password (Post attribute), 19 Password (Post attribute), 19 Password (Post attribute), 19 Password (Request attribute), 19 Post (class in aioauth.requests), 18 post (Request attribute), 19 Post (Response Type IdToken (Response_type), 20 Post (ResponseTypeToken (Class in aioauth.response_type), 20 ResponseTypeToken (class in aioauth.response_type),		
aioauth.responses, 20 aioauth.server, 22 aioauth.storage, 25 aioauth.types, 28 aioauth.utils, 29 N nonce (AuthorizationCode attribute), 16 nonce (Query attribute), 18 NoneResponse (class in aioauth.responses), 21 Password (Post attribute), 19 PasswordGrantType (class in aioauth.grant_type), 14 PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 Post (Request att		
aloauth.server, 22 aloauth.storage, 25 aloauth.types, 28 aloauth.types, 29 ResponseTypeIdToken (class in aloauth.response_type), 20 ResponseTypeToken (class in aloauth.response_type), 20 ResponseTypeToken (class in aloauth.response_type), 20 ResponseTypeToken (class in aloauth.response_type), 20 ResponseTypeIdToken (loase in aloauth.response_type), 20 ResponseTypeIdToken (class in aloauth.response_type), 20 ResponseTypeIdToken (class in aloauth.response_type), 20 ResponseTypeIdToken (loase in aloauth.response_type), 20 ResponseTypeIdToken (class in aloauth.response_type), 20 ResponseTypeIdToken (loase in aloauth.response_type), 20 ResponseTypeIdToken (loase in aloauth.response_type), 20 ResponseTypeIdToken (loase in aloauth.resp		
aioauth.storage, 25 aioauth.types, 28 aioauth.utils, 29 N N nonce (AuthorizationCode attribute), 16 nonce (Query attribute), 18 NoneResponse (class in aioauth.responses), 21 P password (Post attribute), 19 passwordGrantType (class in aioauth.grant_type), 14 ptalin (CodeChallengeMethod attribute), 29 post (class in aioauth.requests), 18 post (Request attribute), 19 post (Reaponse Type), 20 prefech ((Class in aioauth.response_type), 20 prefech ((Roken attribute), 19 post (Repuest attribute), 19 post (Repuest attribute), 19 post (Response TypeToken ((class in aioauth.response_type), 20 prevoke_token(() (BaseStorage method), 28 prevoked (Token attribute), 19 pospe (AuthorizationCode attribute), 19 pospe (AuthorizationCode attribute), 19 pospe (AuthorizationCode attribute), 19 pospe (Query attribute), 19 pospe (Query attribute), 19 pospe (Response attribute), 19 pospe (AuthorizationCode attribute), 19 pospe (Query attribute), 19 pospe (Request attribute), 19 pospe (AuthorizationCode attribute), 19 pospe (AuthorizationCode attribute), 19 pospe (Query attribute), 19 pospe (Request attribute), 19 pospe (RuthorizationCode attribute), 19 pospe (Query attribute), 19 pospe (Rote attribute), 19 pospe (RuthorizationCode attribute), 19 pospe (RuthorizationCode attribute), 20 pospe (Rote attribute), 19 pospe (Rote attribute), 19 pospe (Rote attribute), 19 pospe (Rote attribute), 19 pospe (Rote at	_	
ResponseTypeNone (class in aioauth.response_type), 20 ResponseTypeToken (class in aioauth.response_type), 20 ResponseTypeToken (class in aioauth.response_type), 20 revoke_token() (BaseStorage method), 28 revoked (Token attribute), 17 S \$2556 (CodeChallengeMethod attribute), 29 password (Post attribute), 19 password (Post attribute)		
aioauth.utils, 29 N Nonce (AuthorizationCode attribute), 16 Nonce (Query attribute), 18 NoneResponse (class in aioauth.responses), 21 P Password (Post attribute), 19 PasswordGrantType (class in aioauth.grant_type), 14 PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 Post (Request attribute), 19 Post		
ResponseTypeToken (class in aioauth.response_type), 20 revoke_token () (BaseStorage method), 28 revoked (Token attribute), 17 S S P password (Post attribute), 19 PasswordGrantType (class in aioauth.grant_type), 14 PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 Post (Req		
aioauth.response_type), 20 revoke_token() (BaseStorage method), 28 revoked (Token attribute), 17 S Password (Post attribute), 19 Password (Post attribute), 19 Password (Post attribute), 19 Post (class in aioauth.requests), 18 Post (Request attribute), 19 Post (Request attrib	aioauth.utils,29	
revoke_token() (BaseStorage method), 28 revoked (Token attribute), 17 S S P password (Post attribute), 19 PasswordGrantType (class in aioauth.grant_type), 14 PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Client attribute), 18 redirect_uri (Client attribute), 19 refresh_token (Post attribute), 19 refresh_token (Token Response attribute), 21 refresh_token (Token Response attribute), 17 refresh_token (Token Response attribute), 21 refresh_token (Token Response attribute), 17 refresh_token (Token Response attribute), 21 refresh_token (Token Response attribute), 17 refresh_token (Token Response attribute), 21 refresh_token (Token Response attribute), 22 refr	N	
revoked (Token attribute), 18 NoneResponse (class in aioauth.responses), 21 P password (Post attribute), 19 PasswordGrantType (class in aioauth.grant_type), 14 PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 POST (RequestMethod attribute), 29 Query (class in aioauth.requests), 18 query (request attribute), 19 R R redirect_uri (AuthorizationCode attribute), 18 redirect_uri (Post attribute), 18 redirect_uri (Gient attribute), 18 redirect_uri (Gient attribute), 19 refresh_token (Post attribute), 19 refresh_token (TokenResponse attribute), 21 refresh_token (TokenResponse attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token (Token property), 17 refresh_token (TokenResponse attribute), 17 refresh_token (TokenResponse attribute), 21 refres		
NoneResponse (class in aioauth.responses), 21 Password (Post attribute), 19 PasswordGrantType (class in aioauth.grant_type), 14 PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 Post (class in aioauth.requests), 18 Post (Request attribute), 19 Post (Request attribute), 29 Post (Request attribute), 19 P		
Second (Post attribute), 19 Password (Post attribute), 29 Post (Class in aioauth.requests), 18 Post (Class in aioauth.requests), 18 Post (Request attribute), 19 Post (Request attribute), 18 Post (Request attribute), 19 Post (Request attribute)		revoked (Token duribute), 17
password (Post attribute), 19 password (Post attribute), 29 password (Post attribute), 19 passwo	NoneResponse (class in aioauth.responses), 21	S
scope (AuthorizationCode attribute), 16 PasswordGrantType (class in aioauth.grant_type), 14 PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Query (class in aioauth.requests), 18 query (Request attribute), 19 R R redirect_uri (AuthorizationCode attribute), 18 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Client attribute), 18 refiresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (Token attribute), 21 refresh_token (Token attribute), 21 refresh_token (Exports attribute), 17 refresh_token (Token attribute), 17 refresh_token (Token attribute), 17 refresh_token (Exports attribute), 17 refresh_token (Token attribute), 17 refresh_token (Exports attribute), 17 refresh_token (Token attribute), 17 refresh_token (Exports attribute), 18 redirect_uri (Exports attribute), 19 refresh_token (Exp	P	\$256 (CodeChallengeMethod attribute) 29
PasswordGrantType (class in aioauth.grant_type), 14 scope (Client attribute), 15 scope (Client attribute), 19 scope (Query attribute), 18 scope (Query attribute), 18 scope (Query attribute), 17 scope (Token attribute), 17 scope (Token attribute), 21 scope (Token attribute), 22 scope (Token attribute), 23 scope (Token attribute), 24 scope (Token attribute), 25 scope (Token attribute), 26 scope (Token attribute), 27 scope (Token attribute), 28 scope (Token attribute), 28 scope (Token attribute), 29 scope (Token attribute), 28 scope (Token attribute), 29 scope (Token attribute), 29 scope (Token attribute), 29 scope (Token attribute), 21 scope (Authorization Code Response attribute), 19 scope (Query attribute), 18 scope (Query attribute), 18 scope (Token attribute), 21 scope (Token attribute), 21 scope (Token attribute), 21 scope (Token attribute), 13 scope (Token attribute), 21 scope (Token attribute), 13 scope (Token attribute), 21 scope (Token attribute), 13 scope (Token attribute), 14 scope (Token attribute), 14 scope	nassword (Post attribute) 10	
scope (Client attribute), 15 PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Query (class in aioauth.requests), 18 query (Request attribute), 19 R R redirect_uri (AuthorizationCode attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Client attribute), 16 refiresh_token (Post attribute), 29 refresh_token (Token attribute), 19 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19 refresh_token (Post attribute), 19 roll of the ditribute), 15 scope (Rest attribute), 18 scope (Releast attribute), 18 scope (Token attribute), 17 scope (Token attribute), 17 scope (Token attribute), 17 scope (Token attribute), 17 scope (Query attribute), 18 scope (Query attribute), 18 scope (Token attribute), 17 scope (Token attribute), 18 scope (Token attribute), 19 scope (Token attribute), 18 scope (Token attribute), 18 scope (Token attribute), 18 scope (Token attribute), 18 scope (Token attribute), 19 scope (Token attribute), 18 scope (Token attribute), 19 scope (Token attribute), 19 scope (Token attribute), 18 scope (Token attribute), 19 scope (Token attribute), 19 scope (Token attribute), 18 scope (Token attribute), 18 scope (Token attribut		
scope (Post attribute), 19 scope (Query attribute), 19 scope (Query attribute), 17 scope (Token attribute), 17 scope (Token attribute), 21 scope (Token attribute), 13 scope (Token attribute), 13 scope (Token attribute), 13 scope (Token attribute), 13 scope (Token attribute), 21 scope (Token attribute), 22 scope (Token attribute), 22 scope (Token attribute), 23 scope (Token attribute), 23 scope (Token attribute), 17 scope (Token attribute), 18 scope (Token attribute), 18 scope (Token attribute), 21 scope (Token attribute), 22 scope (Token attribute), 22 scope (Token attribute), 23 scope (Token attribute), 22 scope (Token attribute), 23 scope (Token attribute), 24 scope (Token attribute), 24 scope (Token attribute), 25 scope (Token attribute), 25 scope (Token attribute), 21 scope (Token attribute), 22 scope (Token attribute), 22 scope (Token attribute), 21 scope (Token attribute), 22 scope (Token attribute), 22 scope (Token attribute), 22 sco		
scope (Query attribute), 18 scope (Token attribute), 19 scope (Token attribute), 17 scope (Token attribute), 21 scope (Token ActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 scope (TokenResponse attribute), 22 scope (TokenResponse attribute), 22 scope (TokenResponse attribute), 23 scope (Token attribute), 24 scope (Token attribute), 25 scope (Token attribute), 24 scope (Token attribute), 17 scope (Token attribute), 18 scope (Token attribute), 24 scope (Token attribute), 24 scope (Token attribute), 25 scope (Token attribute), 28 scope (Token attribute), 29 scope (Token attribute), 21 scope (Token attribute), 21 scope (Token attribute), 22 scope (Token attribute), 24 scope (Token attribute), 25 scope (Token attribute), 26 scope (Token attribute), 26 scope (Token attribute), 28 scope (Token attribute), 29 scope (Tok		Scope (Citetti dili ibilic), 13
scope (Token attribute), 17 scope (Token attribute), 17 scope (Token attribute), 22 scope (TokenResponse attribute), 21 scope (TokenResponse attribute), 22 scope (TokenResponse attribute), 23 scope (TokenResponse attribute), 21 scope (TokenResponse attribute), 22 scope (TokenResponse attribute), 21 scope (TokenResponse attribute), 21 scope (TokenResponse attribute), 21 scope (InvalidClientError attribute), 12 scatus_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (Metho		
scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 server_error, 13 settings (class in aioauth.config), 11 settings (Request attribute), 19 settings (Request attribute), 19 settings (Request attribute), 19 settings (Request attribute), 19 state (Query attribute), 18 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 18 refresh_token (Post attribute), 15 refresh_token (Token attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17	PLAIN (CodeChallengeMethod attribute), 29	scope (<i>Post attribute</i>), 19
scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 Query (Class in aioauth.requests), 18 Query (Request attribute), 19 Settings (Class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 redirect_uri (Post attribute), 18 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 15 Refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18	scope (<i>Post attribute</i>), 19 scope (<i>Query attribute</i>), 18
Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Client attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19 SERVER_ERROR (ErrorType attribute), 28 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12 status_code (Response attribute), 22 T TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 TemporarilyUnavailableError, 14 Token (class in aioauth.models), 17 token (Post attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19	scope (<i>Post attribute</i>), 19 scope (<i>Query attribute</i>), 18 scope (<i>Token attribute</i>), 17
Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 22 T T T TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 T emporarilyUnavailableError, 14 Token (class in aioauth.models), 17 token (Post attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22
Redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21
settings (Request attribute), 19 state (Query attribute), 18 redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 18 redirect_uris (Client attribute), 15 refresh_token (Post attribute), 29 refresh_token (Token attribute), 19 refresh_token (TokenResponse attribute), 21 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28
state (Query attribute), 18 redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 22 T T TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 TemporarilyUnavailableError, 14 Token (class in aioauth.models), 17 token (Post attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13
redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uri (Client attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (InvalidClientError attribute), 12 status_code (I	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11
redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 18 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19 status_code (MethodNotAllowedError attribute), 12 status_code (MethodNotAllowedError attribute), 22 T T TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 TemporarilyUnavailableError, 14 Token (class in aioauth.models), 17 token (Post attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19
redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), status_code (Response attribute), 22 T TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 TemporarilyUnavailableError, 14 Token (class in aioauth.models), 17 token (Post attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18
redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 TemporarilyUnavailableError, 14 Token (class in aioauth.models), 17 token (Post attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12
REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19 TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 TemporarilyUnavailableError, 14 Token (class in aioauth.models), 17 token (Post attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12
refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17 REFRESH_TOKEN_EXPIRES_IN (Settings attribute), 19 TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 TemporarilyUnavailableError, 14 Token (class in aioauth.models), 17 token (Post attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (Response attribute), 22
refresh_token (<i>Token attribute</i>), 17 28 refresh_token (<i>TokenResponse attribute</i>), 21 TemporarilyUnavailableError, 14 refresh_token_expired() (<i>Token property</i>), 17 Token (<i>class in aioauth.models</i>), 17 REFRESH_TOKEN_EXPIRES_IN (<i>Settings attribute</i>), token (<i>Post attribute</i>), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Query attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (Response attribute), 22
refresh_token (<i>TokenResponse attribute</i>), 21 TemporarilyUnavailableError, 14 refresh_token_expired() (<i>Token property</i>), 17 Token (<i>class in aioauth.models</i>), 17 REFRESH_TOKEN_EXPIRES_IN (<i>Settings attribute</i>), token (<i>Post attribute</i>), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (Response attribute), 22 T
refresh_token_expired() (<i>Token property</i>), 17 Token (<i>class in aioauth.models</i>), 17 REFRESH_TOKEN_EXPIRES_IN (<i>Settings attribute</i>), token (<i>Post attribute</i>), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uris (Client attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (Response attribute), 22 T TEMPORARILY_UNAVAILABLE (ErrorType attribute),
REFRESH_TOKEN_EXPIRES_IN (Settings attribute), token (Post attribute), 19	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (Response attribute), 22 T TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28
	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (Response attribute), 22 T TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 TemporarilyUnavailableError, 14
11 TOKEN_EXPIRES_IN (Settings attribute), 11	PLAIN (CodeChallengeMethod attribute), 29 Post (class in aioauth.requests), 18 post (Request attribute), 19 POST (RequestMethod attribute), 29 Q Query (class in aioauth.requests), 18 query (Request attribute), 19 R redirect_uri (AuthorizationCode attribute), 16 redirect_uri (Post attribute), 18 redirect_uri (Query attribute), 18 redirect_uris (Client attribute), 15 REFRESH (TokenType attribute), 29 refresh_token (Post attribute), 19 refresh_token (Token attribute), 17 refresh_token (TokenResponse attribute), 21 refresh_token_expired() (Token property), 17	scope (Post attribute), 19 scope (Query attribute), 18 scope (Token attribute), 17 scope (TokenActiveIntrospectionResponse attribute), 22 scope (TokenResponse attribute), 21 SERVER_ERROR (ErrorType attribute), 28 ServerError, 13 Settings (class in aioauth.config), 11 settings (Request attribute), 19 state (Query attribute), 18 status_code (InvalidClientError attribute), 12 status_code (MethodNotAllowedError attribute), 12 status_code (Response attribute), 22 T TEMPORARILY_UNAVAILABLE (ErrorType attribute), 28 TemporarilyUnavailableError, 14 Token (class in aioauth.models), 17

Index 39

```
token_type (Token attribute), 17
token_type (TokenActiveIntrospectionResponse at-
        tribute), 22
token_type (TokenResponse attribute), 21
token_type_hint (Post attribute), 19
TokenActiveIntrospectionResponse (class in
        aioauth.responses), 21
TokenInactiveIntrospectionResponse (class
        in aioauth.responses), 22
TokenResponse (class in aioauth.responses), 21
TokenType (class in aioauth.types), 29
TYPE_AUTHORIZATION_CODE (GrantType attribute),
        28
TYPE_CLIENT_CREDENTIALS (GrantType attribute),
        28
TYPE_CODE (ResponseType attribute), 29
TYPE_ID_TOKEN (ResponseType attribute), 29
TYPE NONE (Response Type attribute), 29
TYPE_PASSWORD (GrantType attribute), 28
TYPE REFRESH TOKEN (GrantType attribute), 28
TYPE_TOKEN (ResponseType attribute), 29
U
UNAUTHORIZED CLIENT (ErrorType attribute), 28
UnauthorizedClientError, 13
UNSUPPORTED_GRANT_TYPE (ErrorType attribute),
UNSUPPORTED_RESPONSE_TYPE
                                  (ErrorType
        tribute), 28
UnsupportedGrantTypeError, 13
UnsupportedResponseTypeError, 13
url (Request attribute), 19
user (AuthorizationCode attribute), 17
user (Client attribute), 15
user (Request attribute), 19
user (Token attribute), 17
username (Post attribute), 19
V
validate_request()
                          (AuthorizationCodeGrant-
        Type method), 14
validate_request()
                              (AuthorizationServer
        method), 23
validate_request() (GrantTypeBase method), 14
validate_request()
                              (PasswordGrantType
        method), 14
validate_request()
                           (RefreshTokenGrantType
        method), 15
validate_request() (ResponseTypeBase method),
        20
validate_request()
                             (ResponseTypeIdToken
        method), 20
```

40 Index