Introduction

Welcome to IAG Authentication Service API! You can use our API as a Developer, to create `tokens`, and use them later to authenticate with other endpoints which accept our `tokens`.

We support language bindings in `cURL`, `Go` (and many more coming soon)! You can view code examples in the dark area to the right, and you can switch the programming language of the examples with the tabs in the top right.

This API docs page was created with Slate.

Background

OAuth2 and OpenID Connect are fundamental to securing your APIs. At some point, your APIs will need to allow limited access to users, servers, or servers on behalf of users. IAG's API authentication and authorization features allow you to manage the authorization requirements for server-to-server applications.

Server *to* Server API Access

World Mappers is a company that has been collecting geo-spatial data for many years. They are building a new API to offer geo-coding and route planning.

For example, their clients will be able to use these services to find a taxi or the closest restaurant to their current location, or for route planning in a drivers and deliveries management application.

World Mappers has chosen to use a standards-based approach instead of developing their own authentication for their API. Using OAuth 2.0, their services are represented as Resource Owners and their consumers are the Clients. These Clients will interact with IAG Auth Service (the Authorization Server) to get an `access_token` that can be used to interact with the World Mapper API.

World Mapper API

API Authorization

By using the OAuth 2.0 authorization framework, you can give your own applications or third-party applications limited access to your APIs on behalf of the application itself.

Using IAG Auth Service, you can easily support different flows in your own APIs without worrying about the OAuth 2.0/OpenID Connect specification, or the many other technical aspects of API authorization.

PARTICIPANTS

Several participants in the OAuth 2.0 specification can be identified:

- **Authorization Server**: IAG Auth Service, in this case
- **Resource Servers**: your APIs
- **Clients**: the consumers of your APIs, which can include third-party applications or your own customers
- **Resource Owner**: the user of your APIs and of the applications
- **User Agent**: the user's browser or mobile app

Using different `grants` (or flows), these participants will interact to grant Clients limited access to the Resource Servers you are building. As a result, the Client will obtain an `access_token` that can be used to call the Resource Server on behalf of the user or of the Client itself.

SUPPORTED FLOWS

- **Server to Server Applications**: Client Credentials Grant

With Client Credentials Grant, a Client can directly request an `access_token` to the Authorization Server by using its Client Credentials (`client_id` and `client_secret`).
Instead of identifying a Resource Owner, this token will represent the Client itself.

- The Client authenticates with the Authorization Server using its `client_id` and `client_secret`
- The Authorization Server validates this information and returns an `access_token`
- The Client can use the `access_token` to call the Resource Server on behalf of itself

This flow is not redirect based, but is an API call made by the Client to the Authorization Server. And finally the resulting `access_token` can be used by the Client to call the Resource Server. Here are some of the use-cases for Client Credentials Grant.

- Allow the Client to make calls to the Resource Server on its own behalf (machine to machine)
- APIs and services that are not user centric

`client_ssl_certificate` can be used in lieu of `client_id` and `client_secret`, provided the client can utilise `ssl_certificate` for authenticating itself.

Token

Create Token

```
curl -v \
   -X POST \
   https://api.auth.auiag.corp/v1/oauth/tokens \
   -H "Content-Type: application/json" \
   -d '{
      audience: "{YOUR_API_ENDPOINT_IDENTIFIER}",
      grant_type: "client_credentials",
      scope: "{REQUESTED_SCOPE}",
      client_id: "{APP_CLIENT_ID}",
      client_secret: "{APP_CLIENT_SECRET}"
   }'
```

Example HTTP Response:

```
HTTP/1.1 201 Created
```

```
{
   "code": 201,
   "data": {
      "token_type": "bearer",
      "access_token": "{ACCESS_TOKEN}",
      "id_token": "{SIGNED_JWT_TOKEN}"
   }
}
```

This endpoint creates a `token` that you can use later to authenticate with other services that support our `token`. At present this endpoint only supports `tokens` for machine to machine communication, i.e the `tokens` are issued for a service / system account.

HTTP REQUEST

`POST https://api.auth.auiag.corp/v1/oauth/tokens`

Only accepts with Header: `Content-Type: application/json`

PAYLOAD SPECIFICATION

Key	Type	Description
`audience`	*string*	API Endpoint for which the generated token will be used
`grant_type`	*string*	`client_credentials`
`scope`	*string*	Space delimited list of scopes, that are being requested
`client_id`	*string*	issued `client_id` for the service account
`client_secret`	*string*	issued `client_secret` for the service account

REQUIRED PARAMETERS

Key	Description
`audience`	API Endpoint for which the generated token will be used
`grant_type`	`client_credentials`
`client_id`	issued `client_id` for the service account
`client_secret`	issued `client_secret` for the service account

If no `scope` is requested, a `token` is generated with empty scopes.

GENERATED CLAIMS

The response will be a signed JWT (JSON Web Token) containing at least the following claims in the body:

```
# Claims that are returned as part of the `id_token`
```

```
{
   "iss": "https://api.auth.auiag.corp/",
   "sub": "{APP_CLIENT_ID}@clients",
   "aud": "{YOUR_API_ENDPOINT_IDENTIFIER}",
   "exp": // unix timestamp of the token's expiration date,
   "iat": // unix timestamp of the token's creation date,
   "scope": "claims:read claims:create"
}
```

CLAIMS SPECIFICATION

Key	Type	Description
`iss`	*string*	Identifies the principal that issued the JWT
`sub`	*string*	Identifies the principal that is the subject of the JWT
`aud`	*string*	Identifies the recipients that the JWT is intended for
`exp`	*timestamp*	Identifies the JWT expiry time
`iat`	*timestamp*	Identifies the time at which the JWT was issued
`scope`	*string*	Space delimited list of scopes, for which the token is valid

Errors

Auth Service API uses the following error codes:

Code	Meaning
400	Bad Request -- Your request is not valid
404	Not Found -- The specified resource could not be found
405	Method Not Allowed -- You tried to access a resource with an invalid method
406	Not Acceptable -- You requested for a response format that isn't json
415	Unsupported media type -- Content-Type is not supported by our server
500	Internal Server Error -- We had a problem with our server. Try again later.
503	Service Unavailable -- We're temporarily offline for maintenance. Please try again later.

References

This API is based on the following **RFC specs**

1. OAuth v2, draft-31
2. OpenID Connect
3. JSON Web Token (JWT)