CodeB Bearer Token Generation API
REST Specification

Contents

Introduction

The CodeB Middleware offers a simple identity layer on top of its Self-Sovereign Identity system. It enables Clients to verify the identity of the End-User based on the authentication performed by a blockchain node, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

Instead of defining new operations the middleware relies on the use of industry standard for identity information such as OpenID Connect (http://openid.net/specs/openid-connect-core-1_0.html) and OAuth2.0 (RFC6749).

As background, the OAuth 2.0 Authorization Framework [RFC6749] and OAuth 2.0 Bearer Token Usage [RFC6750] specifications provide a general framework for third-party applications to obtain and use limited access to HTTP resources. They define mechanisms to obtain and use Access Tokens to access resources but do not define standard methods to provide identity information. Notably, without profiling OAuth 2.0, it is incapable of providing information about the authentication of an End-User. Readers are expected to be familiar with these specifications.

The **CodeB Middleware** implements authentication as a subset of the OAuth 2.0 authorization process. Use of this subset is requested by Clients by including the openid scope value in the Authorization Request. Information about the authentication performed is returned in a JSON Web Token (JWT) called an ID Token.

This specification assumes that the client has already obtained configuration information about the **CodeB Middleware**, including its Authorization Endpoint and Token Endpoint locations. This information is normally obtained via Discovery, as described in **CodeB Discovery 1.0**, or may be obtained via other mechanisms.

Likewise, this specification assumes that the client has already obtained sufficient credentials and provided information needed to use the **CodeB Middleware**. This is normally done via Dynamic Registration, as described in **CodeB Dynamic Client Registration 1.0**, or may be obtained via other mechanisms.

Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Terminology

This specification also defines the following terms:

Authentication
Process used to achieve sufficient confidence in the binding between the Entity and the presented Identity.

Authentication Request
Authorization Request to request that the End-User be authenticated by the Authorization Server, which is a **CodeB Middleware**, to the Client.

Authorization Request
OAuth 2.0 Authorization Request as defined by [RFC6749].

Claim
Piece of information asserted about an Entity.

Claim Type
Syntax used for representing a Claim Value. This specification defines Normal, Aggregated, and Distributed Claim Types.

Claims Provider
Server that can return Claims about an Entity.

Credential
Data presented as evidence of the right to use an identity or other resources.

ID Token
JSON Web Token (JWT) that contains Claims about the Authentication event. It MAY contain other Claims.

Identifier
Value that uniquely characterizes an Entity in a specific context.

Identity
Set of attributes related to an Entity.

Message
Request or a response between a client and the CodeB Middleware.

Subject Identifier
Locally unique and never reassigned identifier within the Issuer for the End-User, which is intended to be consumed by the Client.

IMPORTANT NOTE TO READERS: The terminology definitions in this section are a normative portion of this specification, imposing requirements upon implementations. All the capitalized words in the text of this specification, such as "Issuer Identifier", reference these defined terms. Whenever the reader encounters them, their definitions found in this section must be followed.

For more background on some of the terminology used, see Internet Security Glossary, Version 2 [RFC4949], ISO/IEC 29115 Entity Authentication AssuranceInternational Organization for Standardization, “ISO/IEC 29115:2013 -- Information technology - Security techniques - Entity authentication assurance framework,” March 2013. [ISO29115], and ITU-T X.1252.

Authentication Flow

The CodeB Middleware, in abstract, follows the following steps.

1. The Client sends a request to the CodeB Middleware.
2. The CodeB Middleware the Client and obtains authorization.
3. The CodeB Middleware responds with an ID Token (bearer).

ID Token

The primary mechanism to enable End-Users to be Authenticated is the ID Token data structure. The ID Token is a security token that contains Claims about the Authentication of an End-User by an Authorization Server when using a Client, and potentially other requested Claims. The ID Token is represented as a JSON Web Token (JWT).

The following Claims are used within the ID Tokem:

iss
REQUIRED. Issuer Identifier for the Issuer of the response. The iss value is a case sensitive URL using the https scheme that contains scheme, host, and optionally, port number and path components and no query or fragment components.

sub
REQUIRED. Subject Identifier. A locally unique and never reassigned identifier within the Issuer for the End-User, which is intended to be consumed by the Client, e.g., 24400320 or AItOawmwtWwcT0k51BayewNvutrJUqsvl6qs7A4. It MUST NOT exceed 255 ASCII characters in length. The sub value is a case sensitive string.

aud
*REQUIRED. Audience(s) that this ID Token is intended for. It MUST contain the **client_id** of the Relying Party as an audience value. It MAY also contain identifiers for other audiences. In the general case, the aud value is an array of case sensitive strings. In the common special case when there is one audience, the **aud** value MAY be a single case sensitive string.*

exp
REQUIRED. Expiration time on or after which the ID Token MUST NOT be accepted for processing. The processing of this parameter requires that the current date/time MUST be before the expiration date/time listed in the value. Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time. See RFC 3339 for details regarding date/times in general and UTC in particular.

Iat
REQUIRED. Time at which the JWT was issued. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.

auth_time
Time when the End-User authentication occurred. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time. When a max_age request is made or when auth_time is requested as an Essential Claim, then this Claim is REQUIRED; otherwise, its inclusion is OPTIONAL. (The auth_time Claim semantically corresponds to the OpenID 2.0 PAPE auth_time response parameter.)

nonce
String value used to associate a Client session with an ID Token, and to mitigate replay attacks. The value is passed through unmodified from the Authentication Request to the ID Token. If present in the ID Token, Clients MUST verify that the nonce Claim Value is equal to the value of the nonce parameter sent in the Authentication Request. If present in the Authentication Request, Authorization Servers MUST include a nonce Claim in the ID Token with the Claim Value being the nonce value sent in the Authentication Request. Authorization Servers SHOULD perform no other processing on nonce values used. The nonce value is a case sensitive string.

ID Tokens MAY contain other Claims. Any Claims used that are not understood MUST be ignored.

ID Tokens MUST be signed using JWS and optionally both signed and then encrypted using JWS and JWE respectively, thereby providing authentication, integrity, non-repudiation, and optionally, confidentiality. If the ID Token is encrypted, it MUST be signed then encrypted, with the result being a Nested JWT.

The following is a non-normative example of the set of Claims in an ID Token:

{
"sub": "0x7f2aC5165E1B2287202317992366 47cEB7FE410D",
"aud": "0x7f2aC5165E1B2287202317992366 47cEB7FE410D",
"auth_time": "1607073741",
"iat": "1607073739",
"exp": "1607109736",
"nonce": "d7ff7aed52cf4952b1977997e8250fab",
"client_id": "0x7f2aC5165E1B2287202317992366 47cEB7FE410D",
"grant_type": "password"
}

Authentication

The CodeB Middleware performs authentication to log in the End-User or to determine that the End-User is already logged in. The CodeB Middleware returns the result of the Authentication performed by the Server to the Client in a secure manner so that the Client can rely on it.

The Authentication result is returned in an ID Token. It has Claims expressing such information as the Issuer, the Subject Identifier, when the authentication expires, etc.

Authentication Request

An Authentication Request is an OAuth 2.0 Authorization Request that requests that the End-User be authenticated by the Authorization Server.

Authorization Servers MUST support the use of the HTTP GET and POST methods defined in RFC 2616 at the Authorization Endpoint. Clients MAY use the HTTP GET or POST methods to send the Authorization Request to the Authorization Server. If using the HTTP GET method, the request parameters are serialized using URI Query String Serialization. If using the HTTP POST method, the request parameters are serialized using Form Serialization.

The CodeB Middleware uses the following OAuth 2.0 request parameters with the Authorization Code Flow:

username
REQUIRED. The CodeB Middleware requests MUST contain a username. The username is the blockchain address.

password
REQUIRED. The CodeB Middleware requests MUST contain a password. The password has to be able to unlock the private key of the blockchain address.

client_id
REQUIRED. OAuth 2.0 Client Identifier valid at the Authorization Server. ***Address of Self-Sovereign Identity should be here***

redirect_uri
RECOMMENDED. Redirection URI to which the response will be sent. This URI MUST exactly match one of the Redirection URI values for the Client pre-registered at the OpenID Provider, with the matching performed as described in Section 6.2.1 of [RFC3986] (Simple String Comparison). When using this flow, the Redirection URI SHOULD use the https scheme; however, it MAY use the http scheme, provided that the Client Type is confidential, as defined in Section 2.1 of OAuth 2.0, and provided the OP allows the use of http Redirection URIs in this case. The Redirection URI MAY use an alternate scheme, such as one that is intended to identify a callback into a native application.

State
RECOMMENDED. Opaque value used to maintain state between the request and the callback. Typically, Cross-Site Request Forgery (CSRF, XSRF) mitigation is done by cryptographically binding the value of this parameter with a browser cookie.

nonce
OPTIONAL. String value used to associate a Client session with an ID Token, and to mitigate replay attacks. The value is passed through unmodified from the Authentication Request to the ID Token. Sufficient entropy MUST be present in the nonce values used to prevent attackers from guessing values.

max_age
OPTIONAL. Maximum Authentication Age. Specifies the allowable elapsed time in seconds since the last time the End-User was actively authenticated by the OP. If the elapsed time is greater than this value, the CodeB Middleware MUST attempt to actively re-authenticate the End-User. (The max_age request parameter corresponds to the OpenID 2.0 PAPE [OpenID.PAPE] max_auth_age request parameter.) When max_age is used, the ID Token returned MUST include an ***auth_time*** *Claim Value.*

Token Endpoint

To obtain an ID Token the client sends a Token Request to the Token Endpoint to obtain a Token Response.

Token Request

A Client makes a Token Request by presenting its Authorization Grant (in the form of an Authorization Code) to the Token Endpoint using the grant_type value authorization_code, as described in Section 4.1.3 of OAuth 2.0 [RFC6749]. If the Client is a Confidential Client, then it MUST authenticate to the Token Endpoint using the authentication method registered for its client_id, as described in Section 9.

The Client sends the parameters to the Token Endpoint using the HTTP POST method and the Form Serialization, per Section 13.2, as described in Section 4.1.3 of OAuth 2.0 [RFC6749].

The following is a non-normative example of a Token Request (with line wraps within values for display purposes only):

POST /post.aspx HTTP/1.1
Host: ssi.codeb.io
Content-Type: application/x-www-form-urlencoded

grant_type=password
&scope=openid
&username=0x411fd7d291c185485675d8fadd79341f8921f02b
&password=aloaha123
&client_ID=0xd68608d8e960a7e13f2756f508c7371e03ba176f

To try it out you can post the above to: https://coin.codeb.io/post.aspx

or in GET notation:

GET /post.aspx?grant_type=password&scope=openid&username=
0x411fd7d291c185485675d8fadd79341f8921f02b&password=aloaha123&client_ID=
0xd68608d8e960a7e13f2756f508c7371e03ba176f HTTP/1.1
Host: server.example.com

To try it out call: https://coin.codeb.io/post.aspx*?grant_type=password&scope=openid&username=0x411fd7d291c185485675d8fadd79341f8921f02b&password=aloaha123&client_ID=0xd68608d8e960a7e13f2756f508c7371e03ba176f*

Note: Query parameters are never protected by any encryption. POST is always preferred over GET

Token Request Validation

The CodeB Authorization Server MUST validate the Token Request as follows:

Authenticate the Client with unlocking its server based private key with the supplied username/password

Ensure that the `redirect_uri` parameter value is identical to the `redirect_uri` parameter value that was included in the initial Authorization Request.

Successful Token Response

After receiving and validating a valid and authorized Token Request from the Client, the Authorization Server returns a successful response that includes an ID Token. The parameters in the successful response are defined in Section 4.1.4 of OAuth 2.0 [RFC6749]. The response uses the application/json media type.

The OAuth 2.0 token_type response parameter value MUST be **Bearer**, as specified in OAuth 2.0 Bearer Token Usage [RFC6750]. All nodes and applications SHOULD support the Bearer Token Type.

In addition to the response parameters specified by OAuth 2.0, the following parameters MUST be included in the response:

id_token
ID Token value associated with the authenticated session.

All Token Responses that contain tokens, secrets, or other sensitive information MUST include the following HTTP response header fields and values:

Cache-Control: no-store
Pragma: no-cache

The following is a non-normative example of a successful Token Response.

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

{
"token_type":"Bearer",
"expires_in":35993,
"id_token":"eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCIsImVjY3B1YiI6IlJVTkxNU0FBQUFEay1HZjRZRl9tNUlwcU9KSWs2dkhsNGR3MF9yeTBmNkY3MF8xQ19jT2lfaWhoYWl5SDZhUlA5UHd1bk9lM0d0d3pSZUJjSTB4ZG4xSUlGTldRcHZ6ZiJ9.eyJzdWIiOiIweDdmMmFDNTE2NUUxQjIyODcyMDIzMTc5OTIzNjY0N2NFQjdGRTQxMEQiLCJVc2VyUHViS2V5IjoiMHgwNGNkN2Y2ZDUxMDU1MjQzNzBlNDA4OTZhOTg

wYWEyYmFhZTNjMjQyOGI1MmNlNWVkNDY4NGI0OGZkNDZlZTIzNGFlNjM1MTViMWMwNmRkZD
M3YWM0MTFiNjdkZWEyYTg5ZmE3NDM3NTAyOTZhYWU0NGYyNzBmMTM5MTExZmJjMDU0IiwiY
XVkIjoiMHg3ZjJhQzUxNjVFMUIyMjg3MjAyMzE3OTkyMzY2NDdjRUI3RkU0MTBEIiwiYXV0aF90aW1
lIjoiMTYwNzA3Mzc0MSIsImlhdCI6IjE2MDcwNzM3MzkiLCJleHAiOiIxNjA3MTA5NzM2Iiwibm9uY2Ui
OiJkN2ZmN2FlZDUyY2Y0OTUyYjE5Nzc5OTdlODI1MGZhYiIsImNsaWVudF9pZCI6IjB4N2YyYUM1MTY
1RTFCMjI4NzIwMjMxNzk5MjM2NjQ3Y0VCN0ZFNDEwRCIsImdyYW50X3R5cGUiOiJwYXNzd29yZCIsI
n50aHJlYWQiOnsidGhpZCI6IjEwYzFkZTBiLTY0ZTUtNGExOC1hZDU5LTU2ZDA2OWNlMGMwYSJ9LCJ
AaWQiOiI3MTliMzExYi03Y2Y4LTQ0MWYtOWJhOS00ZTg4ZTZlYTdlZjEifQ.GYeXmtsnfqj32LPtbSSXKV
06ZHgMSXdZFh8YbkGKi701SyEhm2U4KU5QMq9Fp4w2_-J83y1pTQw5q15H1qwjQw"
}

As specified in OAuth 2.0 [RFC6749], Clients SHOULD ignore unrecognized response parameters.

ID Token Validation

Clients MUST validate the ID Token in the Token Response in the following manner:

1. If the ID Token is encrypted, decrypt it using the keys and algorithms that the Client specified during Registration that the node was to use to encrypt the ID Token. If encryption was negotiated with the OP at Registration time and the ID Token is not encrypted, the RP SHOULD reject it.
2. The Issuer Identifier for the Provider (which is typically obtained during Discovery) MUST exactly match the value of the `iss` (issuer) Claim.
3. The Client MUST validate that the `aud` (audience) Claim contains its `client_id` value registered at the Issuer identified by the `iss` (issuer) Claim as an audience.
   The `aud` (audience) Claim MAY contain an array with more than one element.
4. If the ID Token is received via direct communication between the Client and the Token Endpoint (which it is in this flow), the TLS server validation MAY be used to validate the issuer in place of checking the token signature. The Client MUST validate the signature of all other ID Tokens according to JWS using the algorithm specified in the JWT `alg` Header Parameter. The Client MUST use the keys provided by the Issuer.
5. The current time MUST be before the time represented by the `exp` Claim.
6. The `iat` Claim can be used to reject tokens that were issued too far away from the current time, limiting the amount of time that nonces need to be stored to prevent attacks. The acceptable range is Client specific.
7. If a nonce value was sent in the Authentication Request, a `nonce` Claim MUST be present and its value checked to verify that it is the same value as the one that was sent in the Authentication Request. The Client SHOULD check the `nonce` value for replay attacks. The precise method for detecting replay attacks is Client specific.
8. If the `auth_time` Claim was requested, either through a specific request for this Claim or by using the `max_age` parameter, the Client SHOULD check the `auth_time` Claim value and request re-authentication if it determines too much time has elapsed since the last End-User authentication.