

OFFICE EUROPÉEN DE LUTTE ANTI-FRAUDE

ANTI-FRAUD INFORMATION SYSTEM (AFIS)

B2B

Technical Specifications

DOCUMENT HISTORY

Ver.	Release Date	Description	Author (Function, Party)
3.0	08/06/2026	Initial version	Dario Savarese Pedro do Vale Frank Crijns Upasana Khanna Roman Sasiada

REFERENCE DOCUMENTS

Title	Reference	Version	Date
The OAuth 2.0 Authorization Framework	https://tools.ietf.org/html/rfc6749	-	10/2012
OpenID Connect Core	https://openid.net/specs/openid-connect-core-1_0.html	1	8/11/2014
JSON Web Token (JWT)	https://tools.ietf.org/html/rfc7519	-	05/2015
JSON Web Key (JWK)	https://tools.ietf.org/html/rfc7517	-	05/2015
JSON Web Signature (JWS)	https://tools.ietf.org/html/rfc7515	-	05/2015
JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants	https://tools.ietf.org/html/rfc7523	-	05/2015
Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)	https://tools.ietf.org/html/rfc7800	-	04/2016
Resource Indicators for OAuth 2.0	https://tools.ietf.org/html/rfc8707	-	02/2020
OAuth 2.0 Token Introspection	https://tools.ietf.org/html/rfc7662	-	10/2015
The 'Basic' HTTP Authentication Scheme	https://tools.ietf.org/html/rfc7617	-	09/2015
Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing	https://datatracker.ietf.org/doc/html/rfc7230	-	06/2014
Spring Security Framework OAuth2 capabilities	https://docs.spring.io/spring-security/site/docs/current/reference/html5/#oauth2	5.4.5	17/02/2021
Spring Boot	https://spring.io/projects/spring-boot	2.4.4	2021
Forgerock Access Management Documentation	https://backstage.forgerock.com/docs/am/7	7	24/03/2021
Java Platform, Standard Edition (Java SE) 8	JDK 8 and JRE 8 Installation	-	-

GLOSSARY

AFIS	**Anti-Fraud Information System**
alg	**(JWT) Algorithm**
AM	**Forgerock Access Management**
AS	**OAuth2.0 Authorization Server**
AT	**Access Token**
aud	**JWT Audience**
B2B	**Business-to-Business**
CS	**Client System**
exp	**JWT Expiration time**
HTTP	**Hypertext Transfer Protocol**
iat	**JWT Issued At time**
iss	**JWT Issuer**
JSON	**Javascript Object Notation**
jti	**JWT Identifier**
JWK	**JSON Web Key**
JWKS	**JSON Web Key Set**
JWS	**JSON Web Signature**
JWT	**JSON Web Token**
OAuth2.0	**Open Authorization 2.0**
RS	**OAuth2.0 Resource Server**
RT	**Refresh Token**
SAFE	**Service Authentication Featuring Encryption**
SBCP	**OAuth2.0 Security Best Current Practices**
sub	**JWT Subject**
typ	**JWT Type**
URI	**Uniform Resource Identifier**
UUID	**Universal Unique Identifier**

Contents

DOCUMENT HISTORY 2

REFERENCE DOCUMENTS 3

GLOSSARY 4

1 Accessing AFIS B2B services 6

1.1 Introduction 6

1.2 Protocol 6

1.2.1 Prerequisites 7

1.2.2 Steps 7

1.2.3 Authorization Server Exchange 8

1.2.4 Resource Server Exchange 10

2 Annexes 12

2.1 AFIS Security Best Practice for B2B 12

2.1.1 Abstract 12

2.1.2 Recommendations 12

1 Accessing AFIS B2B services

1.1 Introduction

An AFIS B2B service is a service meant to be consumed by an application rather than by a human user. To consume an AFIS B2B service, the calling application must be trusted and entitled. This involves authentication (i.e. determining who the caller is) and authorization (i.e. assessing what the caller can do).

To provide an adequate level of security, the process relies on the OAuth 2.0 protocol using the Client Credentials grant type with the following additional security practices:

- authentication is performed by means of the private_key_jwt method as specified in OpenID Connect Core 1.0;
- The connection should use the standard OAuth2 protocol.
- the Access Token is Audience-Restricted by means of Resource Indicators, as specified in RFC 8707.

This document details the protocol that must be used for accessing AFIS B2B services and provides guidance to use the demonstrator implementing the protocol and distributed under the European Union Public License (EUPL).

1.2 Protocol

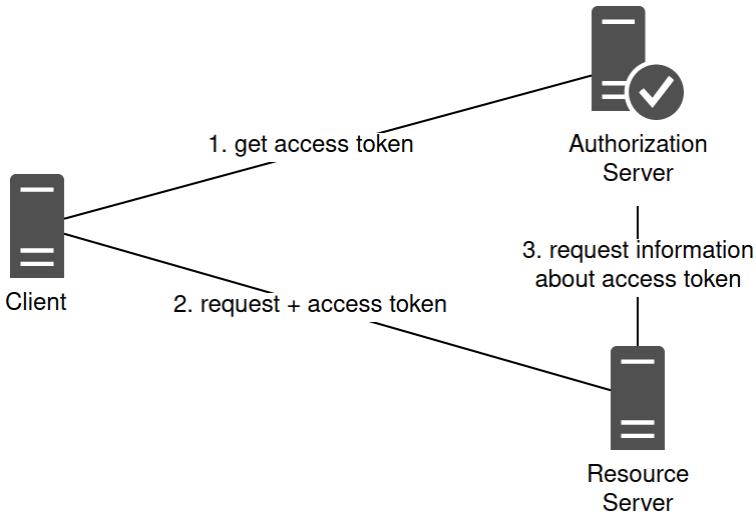
OAuth2.0 is an open standard that addresses authorization and, to some extent, authentication. It supports several usage scenarios for which specific flows are specified, known as grant types.

The Client Credentials grant type specifically targets the use case where an application invokes another application. The calling application, known as the **Client**, obtains an **Access Token** from a trusted third-party, known as the **Authorization Server**, to consume a service exposed by a B2B application, known as the **Resource Server**. This requires that the Client trusts the Authorization Server.

The **Resource Server** can obtain information on the **Client** to perform access control by providing the **Access Token** to the **Authorization Server**. This requires that the Resource Server trusts the Authorization Server.

The Authorization Server therefore acts as a trusted third-party allowing the Resource Server to establish trust with the Client.

The figure below provides a high-level view of the OAuth2.0 Client Credentials grant type.



1.2.1 Prerequisites

The prerequisites to implement the protocol are:

- An RSA public/private key pair with a 2048-bit modulus to authenticate the caller - this is further referred to as **AuthN Key Pair**.
- Registration of the public key of the AuthN Key Pair at AFIS level – as a result of this process, a client identifier is provided (e.g. example-client), further referred to as **client_id**. Optionally, the public key can be dynamic and served as a JSON Web Key Set (JWKS) from a URL.

- Knowledge of the resource indicator of the Authorization Server to be invoked. It corresponds to the absolute URI of the AFIS Authorization Server. This is further referred to as **Authorization Audience**.

- Knowledge of the resource indicator of the Resource Server to be invoked. It corresponds to the absolute URI of the AFIS B2B Service called by the Client and is provided as part of the definition of the AFIS B2B Service (e.g. https://resource-server.sample/protected-resource). This is further referred to as **Resource Audience**.
- Knowledge of the scopes to be bound to the Access Token. They are used to limit what the Access Token grants access to. It is defined for each Resource Server and it can be obtained as part of the definition of the AFIS B2B Service. This is further referred to as **Resource Server Scopes**.

1.2.2 Steps

The protocol consists of two steps:

1. obtaining an access token from the Authorization Server — this is further referred to as the **Authorization Server Exchange**;
2. using the access token to invoke the Resource Server — this is further referred to as the **Resource Server Exchange**.

1.2.3 Authorization Server Exchange

The Authorization Server Exchange is initiated by the Client that sends a request to the Authorization Server. It allows the Authorization Server to authenticate the client.

The Authorization Server answers with a response containing an Access Token that the client can use to invoke an AFIS B2B Service.

1.2.3.1 Request

The figure below describes the request sent by the Client using HTTPS.

```
POST /token HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded;charset=UTF-8Host:
authorization-server.sample

grant_type=client_credentials
&scope=example-scope-read
&client_id=example-client&resource=https%3A%2F%2Fresource-
server.sample%2Fprotected-resource
&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-
type%3Ajwt-bearer
&client_assertion=eyJraWQiOiI3NDQ...
```

The first line of the request contains the method that must be set to **`POST`**. It is followed by the headers above the blank line and the body under the blank line.

The following parameters are mandatory and must be encoded using percent-encoding, also known as URL encoding (e.g. `http://foo` becomes `http%3A%2F%2Ffoo`):

- **`grant_type`** – The OAuth 2.0 grant type used to obtain the Access Token. For accessing AFIS B2B Services, the value must always be `client_credentials`.
- **`scope`** – The Resource Server Scopes to which the Access Token is bound (see Prerequisites). If it is multi-valued then the values must be space-delimited, e.g. ‘`example-scope-read example-scope-write`.
- **`client_id`** – The client_id obtained when registering the public key of the AuthN Key Pair (see Prerequisites).**`resource`** – The Resource Audience corresponding to the AFIS B2B Service that will be invoked (see Prerequisites).
- **`client_assertion_type`** – The type of the client assertion. For accessing AFIS B2B Services, the value must always be `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
- **`client_assertion`** – The JSON Web Token (JWT) used to authenticate the Client, encoded using Base64 (see below). It must be signed with the private key of the AuthN Key Pair (see Prerequisites).

The figure below describes the JWT used to authenticate the Client (client_assertion):

```
{
  "kid": "c8f20dcf-fcd4-4200-b32a-bcf8721cf68d",
  "typ": "jwt",
  "alg": "RS256"
}
{
  "aud": "https://authorization-server.sample/token",
  "sub": "example-client",
  "iss": "example-client",
  "exp": 1613561484,
  "iat": 1613560584,
  "jti": "b50fdb74-1bc0-4aff-9a0e-aa0b40dbaebe"
}The first part is a Javascript Object Signing and Encryption (JOSE) header
with the following elements:
```

- `kid` – The identifier of the AuthN Key Pair. See the figure below how to compute this value.
- `typ` – The media type of the complete JWT. For accessing AFIS B2B Services, the value must always be `jwt`.
- `alg` – The digital signature algorithm. For accessing AFIS B2B Services, the value must always be `RS256`. It corresponds to signing using a SHA-256 hash and RSA encryption.

How to compute the kid:

How to compute the kid:
kid = UUID_V3(publicKey.encoded)

The kid is a UUID Version 3 and it is based on the public key material.
A UUID_V3 is obtained using a hashing so for every byte array input the output will always output the same UUID.
To compute the kid, the byte array to hash is the public key in its primary encoded format.

E.g.: Using Java, the kid can be computed in the following way:

```
import java.util.UUID;
import java.security.PublicKey;

PublicKey publicKey = getPublickJey();
UUID kid = UUID.nameUUIDFromBytes(publicKey.getEncoded());
```

The second part, commonly referred to as claims, has the following elements:

- `aud` – The Authorization Audience (see Prerequisites).
- `sub` – The subject of the JWT. It must be the client_id (see Prerequisites).
- `iss` – The issuer of the JWT. It must be the client_id (see Prerequisites).
- `exp` – The time when the JWT expires, expressed as an Epoch time (i.e. number of seconds elapsed since Thursday January 1st 1970 at 0:00:00 GMT/UTC).
- `iat` – The time when the JWT was issued, expressed as an Epoch time (i.e. number of seconds elapsed since Thursday January 1st 1970 at 0:00:00 GMT/UTC).

- `jti` – The unique identifier of the JWT. It is recommended to use a randomly non-sequential value, such as a Universally Unique Identifier (UUID).

As value for the **exp** claim we recommend setting it to a timestamp representing **1 minute** after it was created (iat). It should be a reasonable time window to complete any further computation happening from the moment the value is set and the moment the request is sent, including potential latency. Clock skew is taken into account on the authorization server. **In any case the gap between the iat and the exp value must not exceed 30 minutes!**

The signature of the JWT is a JSON Web Signature (JWS), as defined in RFC 7515, performed using the private key of the AuthN Key Pair.

1.2.3.2 Response

The figure below describes the response returned by the Authorization Server.

```
HTTP/1.1 200
Content-Type: application/json;charset=UTF-8

{
    "access_token": "6DXJ2uof37yPvhCtoRO1luBOD1g",
    "scope": "read",
    "token_type": "Bearer",
    "expires_in": 59
}
```

The first line of the response contains the status code. It is followed by the headers above the blank line and the body under the blank line.

The body is a JSON document containing the following elements:

- `access_token` – The Access Token is opaque and therefore does not have any structure or expected content. It must be retrieved and stored in a character string.
- `scope` – The scope is expected to match the Resource Server Scopes and is provided for information.
- `token_type` – When accessing AFIS B2B Services, the value is always `Bearer`.
- `expires_in` – The number of seconds during which the Access Token remains valid.

1.2.4 Resource Server Exchange

The Resource Server Exchange is initiated by the Client that sends a request to the Resource Server to consume a service. It must include a valid Access Token obtained using the Authorization Server Exchange described above.

The Resource Server answers with a response containing business specific data or with an Unauthorized HTTP status code. For more details on these types of errors, check Resource Server Exchange.

It must be noted that an Access Token is only valid for a short duration and can be used until it expires. A new Access Token must therefore be obtained once the previous one has expired, unless the AFIS B2B Service implements a mechanism to maintain trust, such as session or state.Request

The figure below describes the request sent by the Client using HTTPS.

```
GET /protected-resource HTTP/1.1
host: resource-server.sample
accept: */*
Authorization: 6DXJ2uof37yPvhCtoRO1luBOD1g
...
```

The first line of the request contains the method that is dependent on the AFIS B2B Service invoked. It is followed by the headers above the blank line and the body under the blank line.

1.2.4.1.1 Headers

The headers are dictated by the AFIS B2B Service being invoked and are outside the scope of this document (including the headers in the example: host, accept.)

In addition to the headers specific to the AFIS B2B Service, the request must include:

- The **Authorization header**, containing the access token obtained using the Authorization Server Exchange (according to RFC6750).

1.2.4.1.2 Body

The content of the body is dictated by the AFIS B2B Service being invoked and is outside the scope of this document.

1.2.4.2 Response

When access is denied, the Resource Server should reply with the specific HTTP status code 401 (Unauthorized). This could happen if the request sent to the Resource Server does not contain an Access Token or if the Access Token is invalid or has expired.

For security reasons, the AFIS F5 proxy provides a first validation of the payload to verify its compliance with the yaml definition. In case of error, there is no error message returned but a support id. The caller needs to contact the AFIS Helpdesk to get detailed information about what is not compliant in the payload.

In other situations, such as a successful invocation, the response sent by the Resource Server is dictated by the AFIS B2B Service being invoked and is outside the scope of this document.

2 Annexes

2.1 AFIS Security Best Practice for B2B

2.1.1 Abstract

This section describes security best practice for OAuth 2.0 adopted by AFIS. It is based on recommendations published in OAuth 2.0 Security Best Current Practice (SBCP) and explains which best practice have been implemented.

2.1.2 Recommendations

2.1.2.1 Client Authentication

From **2.5 Client Authentication of SBCP**:

Authorization servers SHOULD use client authentication if possible.

It is RECOMMENDED to use asymmetric (public-key based) methods for client authentication such as mTLS [RFC8705] or "private_key_jwt" [OpenID]. When asymmetric methods for client authentication are used, authorization servers do not need to store sensitive symmetric keys, making these methods more robust against a number of attacks.

The client authentication method implemented by AFIS is: **private_key_jwt**.

This method requires the Authorization Server to have access to the public key of the Client.

The Client needs to provide either the value of the public key or the JWKS_URI into the client registration info in the Authorization Server.

Resource Servers may use the same authentication method (**private_key_jwt**) when connecting to the introspect endpoint of the Authorization Server.

2.1.2.2 Audience Restricted Access Tokens

From **4.9.1.1.3. Audience Restricted Access Tokens**

Audience restriction essentially restricts access tokens to a particular resource server. The authorization server associates the access token with the particular resource server and the resource server SHOULD verify the intended audience. If the access token fails the intended audience validation, the resource server must refuse to serve the respective request.

AFIS implements audience restriction according to RFC 8707: Resource Indicators for OAuth 2.0.