SINGLE FACE TO INDUSTRY

SFTI API Authentication

Date: 2021-06-21
Version: 1.0

MYNDIGHETEN FÖR DIGITAL FÖRVALTNING
Agency for Digital Government

Sveriges Kommuner och Regioner

KAMMARKOLLEGIET

Upphandlingsmyndigheten

Executive Summary

This document describes and defines SFTI API Authentication specification from a business and technical point of view.

The purpose of the SFTI API Authentication is to enable a standardized approach for authentication between actors which are providing an API and the clients using the API.

This specification is used in conjunction with other API-specifications for specific business scenarios, such as the Availability Check API.

The authentication process described in this specification implements a subset of the functionality in the OAuth protocol.

Table of Content

1 Introduction

This specification is the result of work within SFTI and is published as part of the SFTI collection of recommendations, and specifications. The document provides a specification for implementing the *API authentication*.

1.1 Structure of specification

The structure of this specification consists of five substantial parts.

1. Governance and management part (section 2) that describes how this document is managed, version handled, etcetera.
2. Business specification part (section 3) that describe the process, parties and roles, and business requirements from business and organisational interoperability perspectives.
3. Technical part that describes the Authentication protocol, with use cases, (section 4) from an ICT interoperability perspective.
4. Technical API Specification, comprising of an introductory description (section 6 of this document) and of the formal Open API 3.0 Specification for the SFTI Item Availability Check Service found in Swagger Hub.

1.2 Intended Audience

The intended audience for this document is organizations wishing to use SFTI specifications, recommendations and standards to exchange electronic business documents.

These organizations may be:

- Service providers
- Contracting Authorities
- Economic Operators
- Software Developers

More specifically the document is addressed towards the following roles:

- Business Analyst and EDI administrator
- ICT Architect and ICT Developer

This document aims to support the interested parties in the following work they do and questions they need answered.

Business Analyst and EDI administrator
Work to be done:

a) Evaluate consequences and requirements for setting up an API integration between trading partners
b) Achieve interoperability from a legal, business concept, organisational, semantic, and business security point of views.
c) Ensure handling of business errors and exceptions.
d) Evaluate and manage business agreements and risks.
e) Develop business requirements for development of EDI and business systems.

f) Manage processes, services and business specifications that are version handled.
g) Decide if the specification, its services and business rules deliver benefits, incur reasonable risks, and sufficiently fit with the organisations security policy.

ICT Architect and Developer

Work to be done:

a) Evaluate technical API specifications and interoperability from technical, technology, data and technical security point of views.
b) Achieve interoperability from a technical, technology, data and technical security point of views.
c) Ensure handling of technical errors and exceptions.
d) Manage technical agreements and risks.
e) Develop technical requirements for development of the technical systems.
f) Develop software using the authentication protocol.
g) Test technical system.
h) Automatically generate software for testing.
i) Manage technical specifications that are version handled.
j) Decide if the specification, its services and technical rules incur reasonable technical risks, and sufficiently fit with the organisations security policy.

2 Governance and Management

The documentation is governed, and lifecycle managed, by Single Face to Industry (SFTI). Questions relating to this specification can be directed to SFTI Technical Secretariat, tekniskt.kansli@skr.se.

2.1 Versions and Releases

This specification is based on major, minor and revision versioning principles, expressed as major.minor.revision, for example “Version 1.1.1”.

2.2 Terms of Use and Statement of Copyright

2.3 References

Link to main site of documentation: http://www.sfti.se/

References used in this specification

Item	Link / Source	Comment
OAuth v2	https://oauth.net/2/	Defines the underlying standard which this specification is a profile of

Other references

Item	Link / Source	Comment
SFTI Technical Secretariat	tekniskt.kansli@skr.se	Email to SFTI Technical Secretariat, may be used for questions about this specification.

3 Business Level Specification

3.1 Benefits

The benefits of using this method of authentication is that the Resource Provider (the organisation which provides the API) can manage users and access rights in a secure and rational way. The authentication specification can be used for many different types of APIs and makes it possible for both Resource Providers and Clients to reuse their implementations and knowledge.

3.2 Parties and Roles

The table below gives the definitions of the parties and roles of this specification. This specification doesn't

Business parties	**Description**
Customer	The customer is the legal person or organization who is in demand of a product or service. A Customer can take the role of both Resource Provider or Client depending on the type of API being used.
Supplier	The supplier is the legal person or organization who provides a product or service. A Supplier can take the role of both Resource Provider or Client depending on the type of API being used.
Role / actor	**Description**
Client	The organization (and application) which are the consumers of the API.
Resource Provider	The organization which provides an API to Clients.

3.3 Business Requirements and Agreements

The SFTI API Authentication requires that several parameters are agreed upon and configured in advance. Necessary parameters which the Customer and Supplier must agree on are:
A ClientID which identifies the organisation accessing the API/resource, the password (client secret) and the end points (URLs for the authorization server and the resource server).

4 Technical Level Specification

4.1 Description

This section of the specification provides technical details on the authentication process that enable ICT architects and developers to design, develop and test ICT solutions and products.

4.2 Technical Security and Risk

The Client ID and Client Secret must be treated as confidential credentials and appropriate means must be considered to ensure they are not disclosed to the user of the client.

During authentication with the Token Endpoint, both the Client ID and Client Secret are exchanged with the Authorization Server as part of the Access Token Request. The OAuth 2.0 specification allows for these parameters to either be included as query parameters of the URL or Base64 encoded as a HTTP Basic Authentication header.

The later approach would the preferred way since it hides the used values from the URL path (which can be recorded by some logs).

4.2.1 Encryption in transmit

Since clear-text credentials are exchanged as part of the authentication process towards both the Authorization Server and Resource Server, it is essential that this communication is encrypted by the means of Transport Layer Security (TLS). This profile expects endpoints to be graded as at least A at all times, in terms of TLS version and configuration, according to SSL Labs grading tool (https://www.ssllabs.com/ssltest/).

4.3 Components and parameters

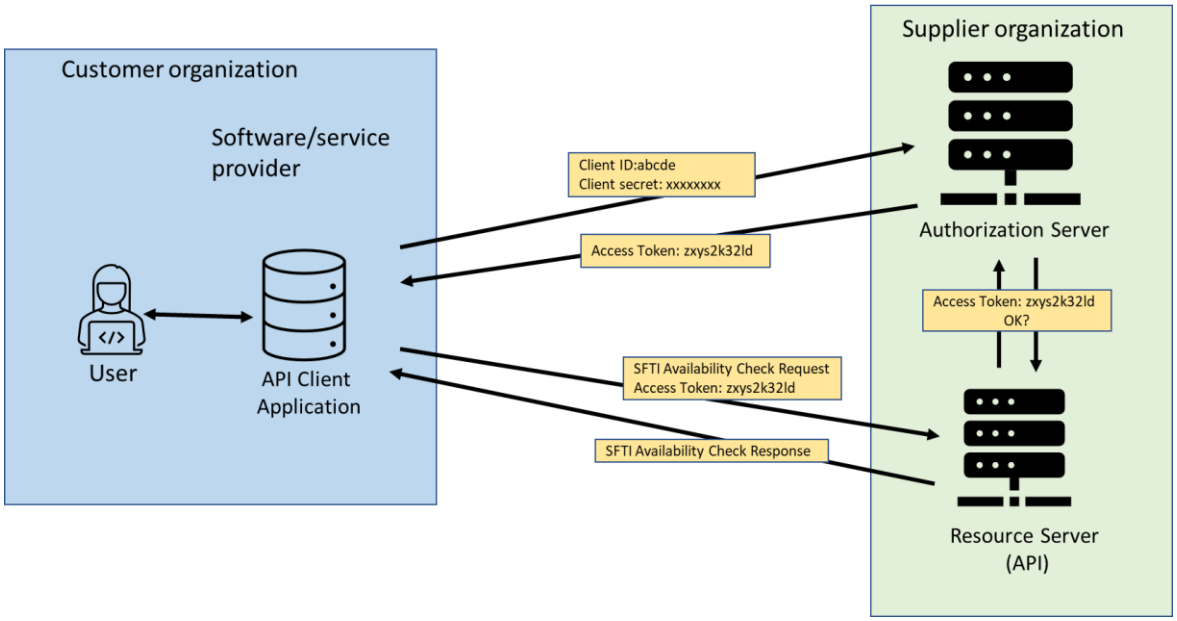
The following components are interacting with each other when authroizing a client application and accessing an API.

Components	**Description**
Client application	The application which is accessing the API .
Authorization Server	The server responsible for implementing the Token Endpoint interface which the client interacts with when requesting an Access Token. Provided by the Resource Provider.
Resource Server	The server responsible for implementing the agreed Resource Endpoint(s) which the client interacts with when performing API operations. Provided by the Resource Provider.

The following parameters are used in the communication between the client application and the Authorization/Resource Server.

Parameter	**Purpose**	**Type**
Token endpoint	URL of the OAuth 2.0 Token service. Authenticates requests from the client and produces access tokens.	URL
Resource endpoint	One URL that provide means of performing agreed API action. The access token authorizes the usage of this endpoint.	URL
Client ID	An identifier used by the client. Pre-agreed during the setup of the integration.	An alphanumeric value with recommended max length of 36 characters.
Client Secret	A secret value used by the client. Pre-agreed during the setup of the integration.	An alphanumeric value with recommended max length of 36 characters.
Grant Type	A parameter that is part of the Access Token request. In this scenario it must always be "client_credentials"	A static alphanumeric value, "client_credentials".
Bearer Token	A value calculated by the Authorization Server on each Access Token request. The value should be included in the Access Token Response at the "access_token" parameter. The value is not intended to have any meaning for the Client, whereas the Resource Server must be able to evaluate the validity of the value.	A string value that allows the usage of following characters: • Alphanumeric characters • "-" • "." • "_" • "+" • "/" The character "=" is also allowed but only as a trailing padding.

The diagram below illustrates, on a conceptual level, an example of the interaction between a Client Application (such as the customer's procurement system) and the Resource provider (such as the supplier's ERP-system)



4.3.1 API Client

The API Client Application is part of the software/service which enables the user to interact with the API, such as a procurement system. It is assumed that the User is authenticated in the application, but specific or technical requirements for the authentication of the User towards the Client application is out of scope of this profile.

The Client requests an Access Token from the Authorization Server by the means of sending an Access Token Request towards the defined Token Endpoint.

Related parameters of an Access Token request:

Parameter Name	Mandatory/Optional	Value
client_id	Mandatory	Agreed with resource provider
client_secret	Mandatory	Agreed with resource provider
grant_type	Mandatory	Fixed: “client_credentials”

4.3.2 Authorization Server

Authenticates the Access Token Request that is originating from the API Client. Upon successful authentication it will return an Access Token Response with a given set of parameters.

Related parameters of an Access Token Response:

Parameter Name	Mandatory/Optional	Value

access_token	Mandatory	The generated access token value that the Resource Server will consume.
token_type	Mandatory	Fixed: “bearer”
expires_in	Mandatory	Value in seconds that indicates for how long the token can be used.

4.3.3 Resource Server

The interfaces and functionality which the Resource Server provides (such as the SFTI Item Availability Check API) is not covered in this specification, only the aspects related to authenticating the Client.

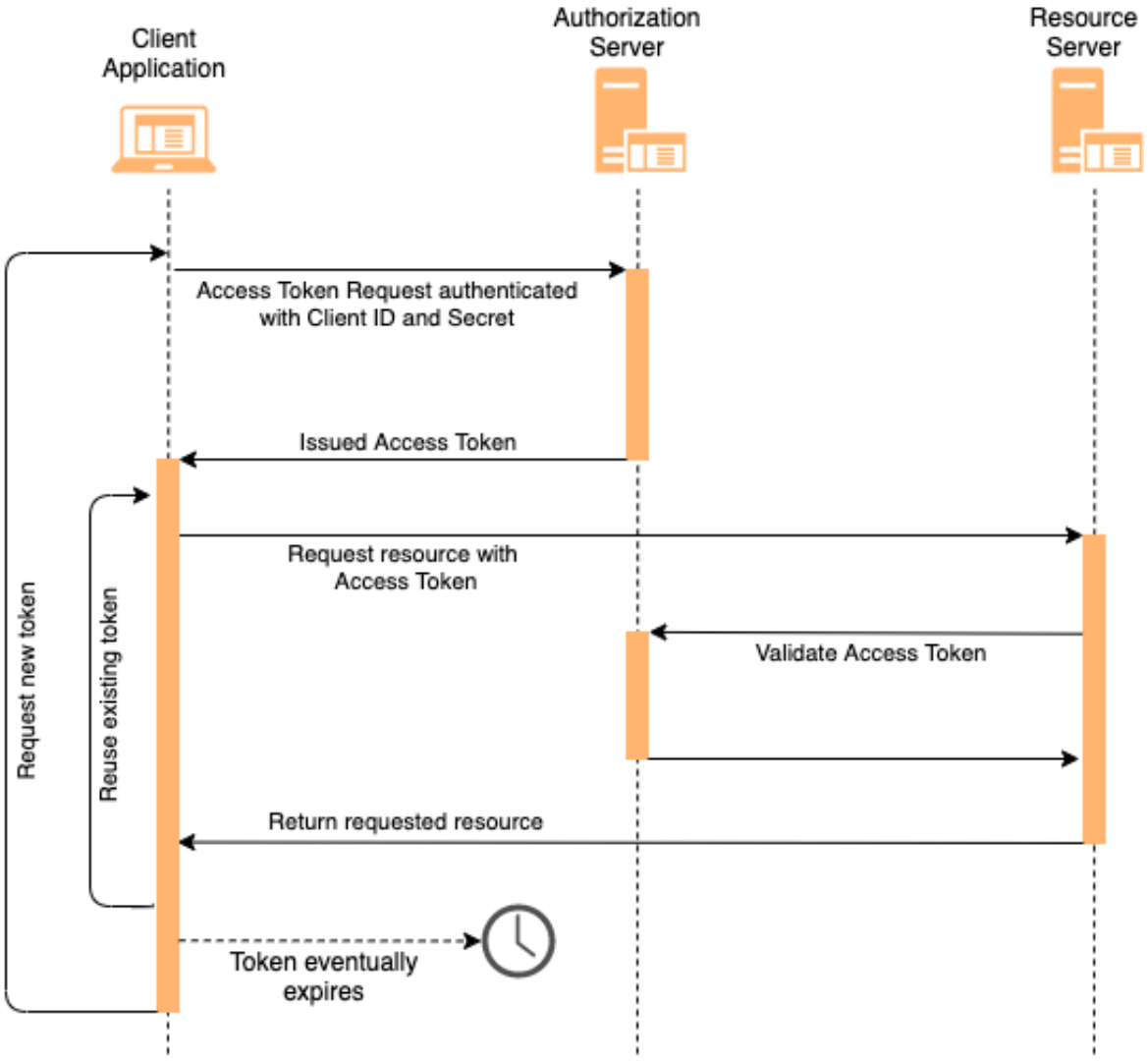
The agreed endpoints of the Resource Server must be able to authenticate the issued “access_token” value from the Access Token Response. The client must include this value in the Authorization header as a Bearer token. Example:

```
Authorization: Bearer <access_token value>
```

4.4 The authentication process flow

This profile is using the OAuth 2.0 Client Credential Flow, the following are the key steps in this flow.

1. Client application requests an Access Token from the Authorization Server by authenticating with the Client ID and Secret.
2. Authorization Server authorizes the request by returning an Access Token Response
3. The client application requests the resource (API-call) by authenticating with the provided access token value from the Access Token Response
4. Resource server validates the access token and if access is granted, returns the expected resource/result.
5. The client application can then repeat the process from step 3 as long as the access token is not expired.
6. Once the access token expires the client application has to revisit the Authorization Server to obtain a new access token.



4.4.1 Access Token Request

Example:

```
POST /sfti-api/oauth2/token HTTP/1.1
Host: www.example.com
Authorization: Basic ${Encoded Client ID and Secret}
Content-Type: “application/x-www-form-urlencoded”

grant_type=client_credentials
```

4.4.2 Access Token Response

Example:

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

{
          "access_token":"v2/4.-W1...Z+C_6Im=",
          "token_type":"bearer",
          "expires_in":600
}
```

4.4.3 Resource Server Request

Example:

```
[GET|PUT|POST|DELETE|OPTIONS] /sfti-api/check-item-availability/1.0
Host: www.example.com
Authorization: Bearer ${access_token value}
```

The above is an incomplete example since other parameters are specific to the API implementation the Resource Server supports.

4.4.4 Access token time to live (TTL) and refresh

The recommended value for an access tokens validity is 600 seconds.

4.4.5 Error handling

The error handling is defined for the Authorization Server and in some areas of the Resource Server. The Resource Server might also need to adhere to the error handling as defined in the API specification it is implementing.

Example:

```
HTTP/1.1 [400|401] Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "error": "invalid_request",
  "error_description": "Request was missing the 'grant_type'
parameter.",
  "error_uri": "Supplementary error information available at
https://www.example.com/documentation"
}
```

Parameters of the JSON formatted response:

Parameter Name	**Mandatory/Optional**	**Description**
error	Mandatory	Refer to list of valid error codes.
error_description	Mandatory	A human readable description of the error.
error_uri	Optional	A reference to a URL that can provide information about how to resolve the error.

Valid error codes:

Error Code	**HTTP Status**	**Usage**
invalid_request	400	The request is missing a mandatory parameter or is including an unknown parameter.
unsupported_grant_type	400	A grant type not recognized by the server was defined.
invalid_client	400	A Client ID not recognized by the server was used.
access_denied	400	When the Resource Server denies access to a specific resource.
token expired	401	When the Resource Server identifies that the provided access token has expired.
invalid_token	401	When the Resource Server identifies the toke as being invalid.
invalid_client_secret	401	When an invalid or unknown Client Secret was used.

5 Release Log

This section provides documentation of changes to this document and underlying standards and other specifications.

Date	Version	Changes
2021-06-21	1.0	First version