Unifiedpost, UAB

EuroConnector

A solution for data exchange via PEPPOL network

IT specification

CONTENTS

1 General

1.1 History of this document

Version	Date	Author	Comments
1.0	2023-10-13	Nerijus Abraitis	
1.0.1	2023-11-06	Nerijus Abraitis	Adjustments, fixed urls, more explanations
1.1	2024-02-09	Nerijus Abraitis	BIS3 responses added (OpenAPI v1.1)
	2024-03-28	Nerijus Abraitis	Clarifications added

1.2 Points of Contact

Here is the list of contacts in the project:

No.	Name / E-mail	Title / Role	Organization
1.	Paulius Augulis paulius.augulis@unifiedpost.com	Project Manager, PMP	Unifiedpost, UAB
2.	Žilvinas Kaziukonis zilvinas.kaziukonis@unifiedpost.com	Product manager, ERP Expert	Unifiedpost, UAB
3.	Nerijus Abraitis nerijus.abraitis@unifiedpost.com	IT Project manager, Software architect	Unifiedpost, UAB

1.3 Project Description

1.3.1 Terminology

The terminology to be used in this document:

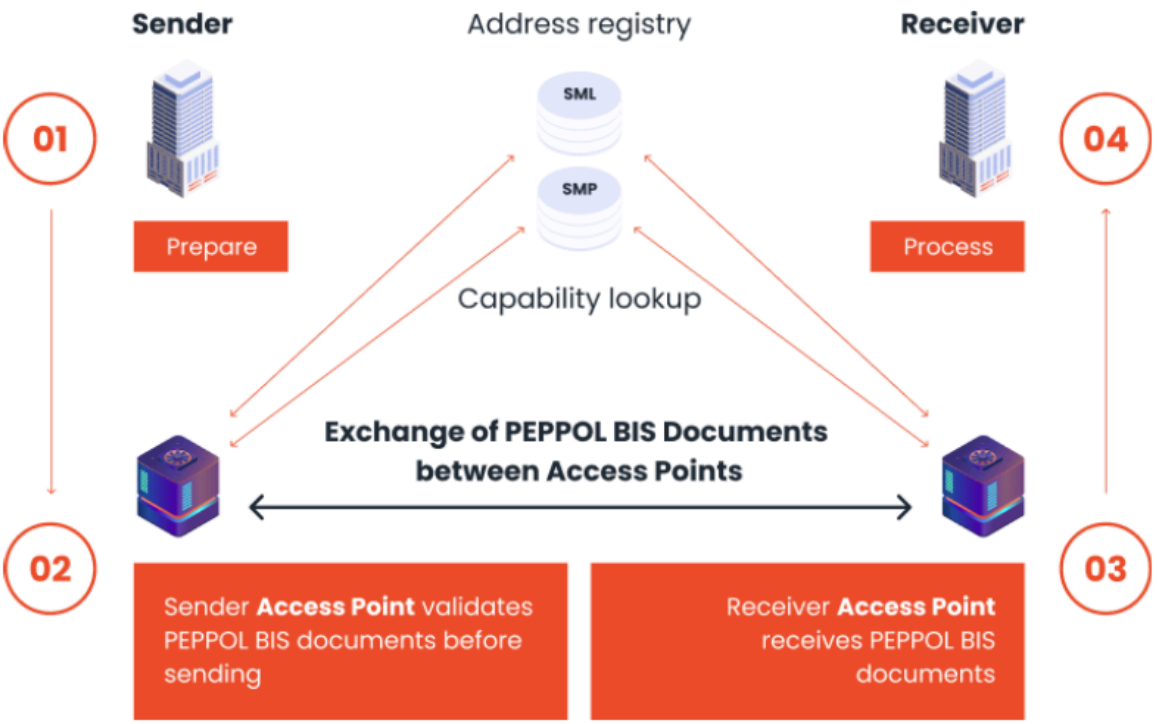
Peppol	Peppol (Pan-European Public Procurement Online) is a set of specifications for establishing and also the primary implementation of a federated electronic procurement system for use across different jurisdictions. Through Peppol, participant organizations can deliver procurement documents to each other including electronic invoices in machine readable formats, avoiding the labour of data entry.
EuroConnector	This solution to exchange invoices via Peppol network
AP	A Peppol Access Point is a software, offered or owned by a company, that can connect others to the Peppol network and exchange documents via the required standards/protocols and in accordance with the necessary regulations.
SMP	All organizations that receive e-documents via Peppol publish their electronic receiving addresses and supported document types. Service Metadata Publisher (SMP), like an address book, contains a business' address (URL) and receiving capabilities. An SMP can be bundled with an Access Point or provided as an independent service.
SML	Service Metadata Locator (SML) is a centralized component that stores the location of all SMPs for each participant (receiver) within the Peppol Network. When sending electronic documents to a business on the network, the SML returns the address to the SMP that contains all necessary details for a correct transfer.
BIS (eInvoice)	Business Interoperability Specifications (BIS) are formal requirements to ensure pan-European interoperability of procurement documents, such as eInvoices. https://docs.peppol.eu/poacc/billing/3.0/
SME	Small and medium-sized enterprises (SMEs) represent 99% of all businesses in the EU. The definition of an SME is important for access to finance and EU support programs targeted specifically at these enterprises.
KYC	KYC means "Know Your Customer." It describes the process of verifying the identity of (new) customers. The KYC process is performed to prevent illegal activities such as money laundering or fraud, in return protecting both company and client.
ERP	Enterprise Resource Planning refers to a type of software that organizations use to manage day-to-day business activities such as accounting, procurement, project management, risk management and compliance, and supply chain operations.
EDI	Electronic Data Interchange (EDI) is the electronic interchange of business information using a standardized format - a process which

	allows one company to send information to another company electronically rather than with paper.
Entity	Business or public organization which is subject to issue or receive invoices
UPG	Unifiedpost Group - the creator of this Euroconnector solution

1.3.2 Background of Peppol infrastructure

Peppol four-corner model: modern electronic transactions

Peppol runs on the four-corner model, which can be summarized as four components working together to enable an information exchange for procurement data: a sender, a recipient, and two access points (one for each of them):



Picture 1

When a message is sent through a four-corner model, the following steps occur:

- The sender communicates with its access point and creates a new message in the agreed-upon format of the network.
- The sending access point references the service metadata locator to find the Peppol ID of the recipient, which has been previously registered in the network.
- The sender's access point pushes the message to the recipient's access point, which

then forwards the message to the intended recipient.
- The two access points connect once more to confirm that the message was sent successfully. A transport acknowledgment may be involved.

Standardized e-invoicing

The European Union works with its own e-invoicing standard codenamed EN 16931. This systematic approach to invoicing supports multiple EU laws and directives, such as the Principle VAT Directive for tax purposes.

The specific implementation of this standard is the Peppol BIS Billing 3.0 format, which any European public authority has an obligation to follow by law.

1.3.3 Purpose

There is a potential demand in the Lithuanian market to send and receive e-invoices via PEPPOL network as it is becoming a trend and kind of a requirement all over Europe.

Meanwhile, it's not an easy task for an ERP / EDI provider or SME to implement it, because it requires a lot of IT knowledge and programming resources.

For that reason, an invoice exchange tool EuroConnector is intended to be created with the main goals of making the solution simple, secure, easy-to-implement and for it to become widely used.

1.3.4 Assumptions and Constraints

The created IT solution must be placed in a publicly accessible storage (e.g. github or so), an OpenSource code should be available.

- Specification documents (instructions) must be prepared and published, explaining how the solution is to be implemented and used;
- the software code of the solution must be published;
- deployment-ready version must be provided.

The essential IT requirements for the solution:

- the solution must meet the Peppol Network requirements (AS4 profile, Business Message Envelope (SBDH));
- electronic records sent and received must comply with Peppol BIS 3.0 Post-Award requirements;
- IT solution created must ensure the Peppol KYC (know your customer) requirements;
- Testing of the solution with at least 3 ERP systems or eInvoicing vendors using a PEPPOL-connected Access Point and SMP/SML* must have been carried out, and documentation of successful testing must be provided (*SML - Service Metadata Locator; SMP - Service Metadata Publisher)

2 FUNCTIONAL REQUIREMENTS

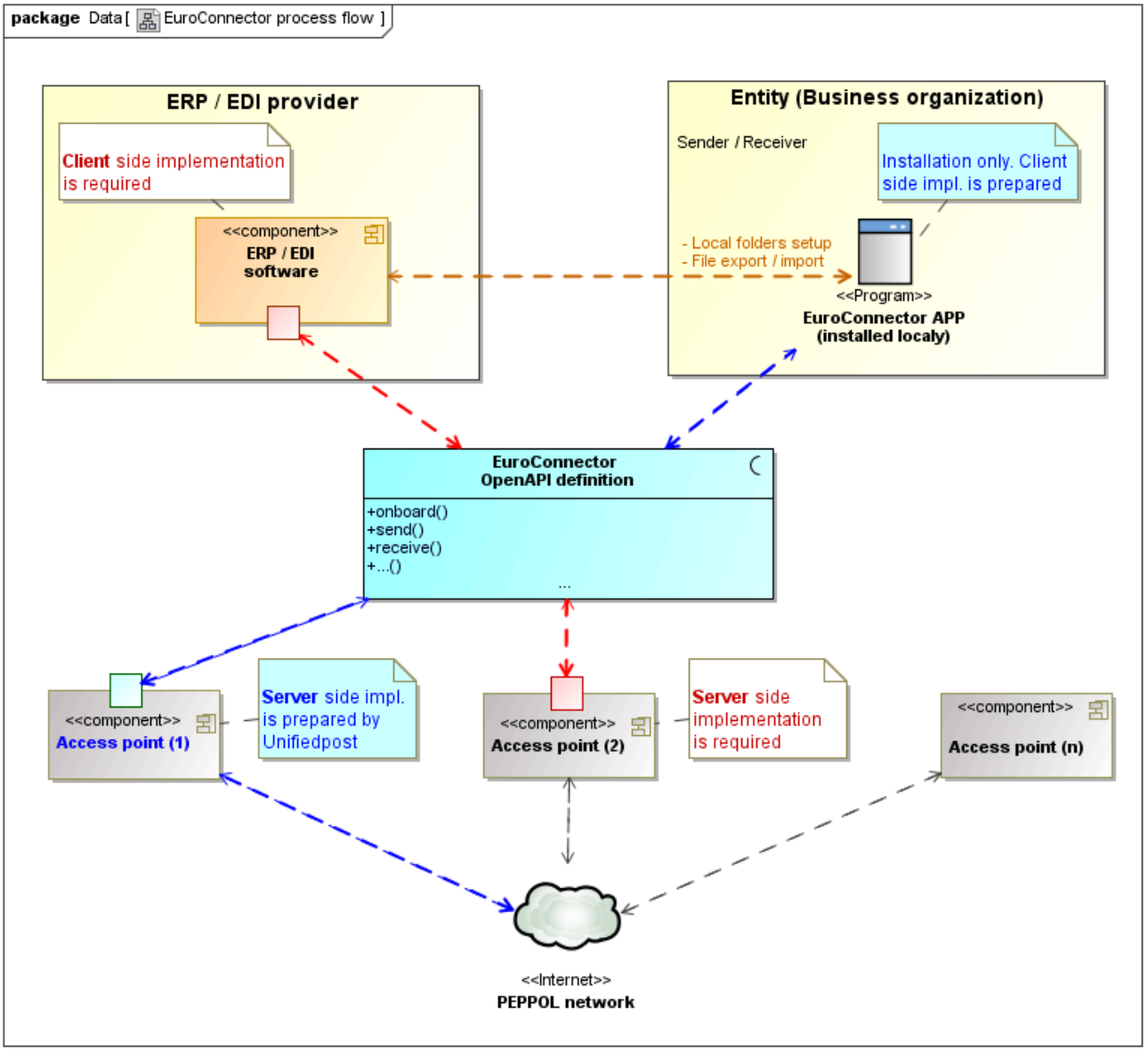
2.1 The scope of the solution

2.1.1 Model of use

EuroConnector solution in general is a group of several elements:

- EuroConnector OpenAPI definition with programming code samples of client and server sides. It's **important to note** that ERP or the Entity should implement the client side and the Access points (AP) - the server side.
- Access points (AP) which implement the server part of EuroConnector OpenAPI definition;
- EuroConnector client-side implementations:
    - OpenAPI client-side implementation by ERP / EDI provider who onboards the Entities to the Peppol network and allows them to exchange documents.
    - Already prepared client-side app which could be installed locally at the Entity part. The programming code of the app is OpenSource, so it can be adjusted according to the Entity or ERP / EDI requirements

The Euroconnector process flow schema is presented below:



Picture 2

According to the picture, the blue arrows represent already implemented features and the red ones – the features which require additional implementation at the Access point or the ERP/EDI or the Entity side.

It's important to know that Euroconnector client-side app or ERP/EDI providers are able to connect only to those Access points which are a part of the EuroConnector ecosystem. Connections to different Access points can be utilized using the same client side implementation, having just an argument which refers the URL to the Access point. To change the Access point means only to change the URL in the client's implementation setup.

2.1.2 IT infrastructure

The solution has a shared and transparent infrastructure. All the participants mostly use their own IT resources. The main infrastructure components are listed below:

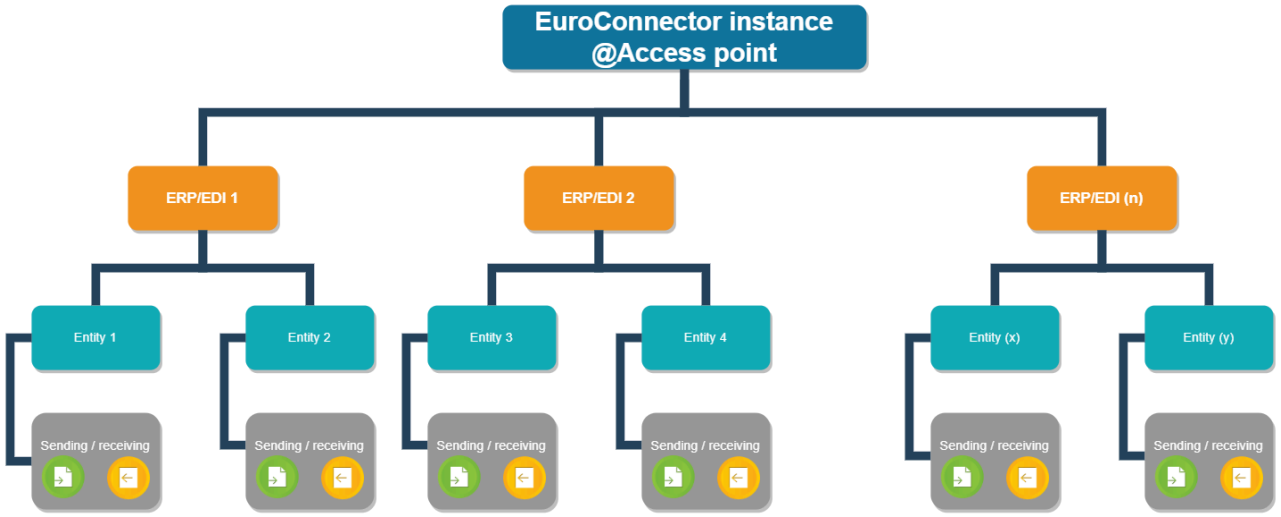
EuroConnector OpenAPI definition	- Collaborative documentation at SwaggerHub platform (https://app.swaggerhub.com/apis-docs/euroconnector/econ-def) - The documentation can be connected directly to several server-side implementations (working solutions). - Code samples of the server and client parts are provided - Limited open-source code of OpenAPI at GitHub storage by Unifiedpost Group (https://github.com/UnifiedPost/econ-doc)
EuroConnector client-side app	- Documentation at GitHub storage by Unifiedpost Group (https://github.com/UnifiedPost/econ-client-app/wiki). - Client-app open-source code at GitHub storage by Unifiedpost Group (https://github.com/UnifiedPost/econ-client-app) - The app should be downloaded, installed and executed locally at the ERP/EDI or Entity side. - Recommended for those who want to start quick and easy
Server-side implementations	- Can be launched at any Peppol Access Point site. - Additional and more custom programming resources are required. - At least one implementation will be prepared by Unifiedpost Group and can be used immediately.
Client-side implementations	- Can be launched at any ERP/EDI or even the Entity side. - Additional programming resources are required for OpenAPI integration. - The EuroConnector client-side app will be prepared and can be installed and used immediately.

2.2 Functional process requirements

2.2.1 Getting started

The picture below displays the business structure of the platform. There are a few major steps:

- Platform administrator of Euroconnector instance registers an ERP/EDI provider to the platform
- When an ERP/EDI provider is up and running they can onboard their customers (Entities) to the Peppol network.
- When the Entity is successfully onboarded, they can send and receive invoices or execute other actions available on the platform.



Picture 3

2.2.2 Registering new ERP/EDI provider

When an ERP/EDI decides to use the platform, its representative person should contact the Platform administrator of the implemented Euroconnector instance (in most cases it would be Access point) and sign an agreement. The platform administrator is responsible to make sure that KYC ("Know Your Customer") requirements are met.

Once KYC requirements are met, the Platform administrator creates an account for ERP/EDI and safely transfers the connection credentials to the ERP/EDI representative.

With those credentials, ERP/EDI connects to the EuroConnector API and manages their customers (Entities).

2.2.3 Onboarding a new Entity to Peppol and obtaining a Peppol ID

Normally, to get a Peppol ID, you should register at a Peppol access point and provide them with information about your business entity. Your chosen access point service provider will verify this information. By the way, the verification may take several working days.

If you're using a software to maintain invoices (e.g. of some ERP/EDI provider), you may be able to use the software in conjunction with the Access point.

After the registration, you should also take some time to browse the Peppol directory https://directory.peppol.eu/public/ or https://peppol.helger.com/public . They are useful resources to check the document types supported by different users on the network. The list will also provide additional information such as Entity name, location, Peppol ID, and relevant website address.

So, according to the Euroconnector model, when an Entity (invoice sender/receiver) decides to use the platform, their representative person could contact the ERP/EDI provider only (instead of contacting Access point) and pass the local KYC procedure which is implemented on ERP/EDI's side.

After KYC is met, the ERP/EDI provider onboards the Entity into the Peppol network, creates an account for the Entity, and hands over the connection credentials. Those actions could be done by using the standard Euroconnector features.

And then, with those credentials, the Entity connects to the EuroConnector API and sends/receives invoices. Some extra API actions could be available as well.

2.2.4 EuroConnector OpenAPI features by design

EuroConnector OpenAPI definition includes these standard functions:

- Authentication and authorization
- Access to Peppol SMP/SML Registry (onboarding an Entity to Peppol)
- Validation of the BIS3 standard messages
- Sending of the BIS3 standard messages
- Receiving of the BIS3 standard messages

BIS3 standard messages include Invoice, Credit note, Message level response and Invoice response documents.

2.2.5 Extra features that depend on EuroConnector implementation instance

There are some extra functions which are not a part of the standard, but would be useful in the scope of this solution. The functions could be included into the EuroConnector Server side implementation by the particular Access point:

- The validation of the most popular e-invoice standards
- The transformations of the most popular e-invoice standards to/from BIS3

2.2.6 Features of the EuroConnector client application

The EuroConnector client application supports the most documents' management functions which are listed in the standard definition:

- Authentication and authorization
- The transformations of the most popular e-invoice standards to/from BIS3
- Validation of BIS3 standard messages
- Sending of the BIS3 standard messages
- Receiving of the BIS3 standard messages
- Responding to the received Invoice or Credit note with the Invoice response message

BIS3 standard messages include Invoice, Credit note, Message level response and Invoice response documents.

Open-source programming code is provided with the samples of web calls to EuroConnector API.

2.2.7 Typical e-invoice sending procedure via application

When an e-invoice is sent through a locally installed application, the following steps occur:

- The Supplier (invoice sender) is authorized to access the EuroConnector web API resources
- The app validates the e-invoice for BIS requirements
- Then the app sends a BIS e-invoice via the EuroConnector web API
- Web API redirects the message to the Access point and the e-invoice is sent out to the Peppol network
- After a few moments the Supplier can check the status of the sent invoice document in the Outbox folder view
- The Supplier is also able to receive the response messages if the invoice receiver part (Customer) supports responding functionality

2.2.8 Typical e-invoice receiving procedure via application

Application performs the following actions in order to receive new invoices from the Peppol network:

- The Customer (invoice recipient) is authorized to access EuroConnector web API resources;
- The Customer triggers the Inbox folder view to check if there are any new documents;
- If there are, they should be presented with the status “New”;
- The Customer triggers the download process for a selected document in the Inbox folder view;
- The document is marked as “received” when download is finished.
- When the Customer verifies the invoice content then he/she sends the response (accept, reject, etc.) to that invoice.
- After a few moments the Customer can check the status of the sent response document in the Outbox folder view.

2.2.9 Authentication and authorization

The API-key token-based authentication is used in this solution.

A token is a piece of data that has no meaning or use on its own, but combined with the correct tokenization system, becomes a vital player in securing your application. This ensures granting the access to a set of resources (e.g., remote APIs or user data).

After a successful onboarding procedure, the Client receives a *Client Secret* credential to identify and authenticate themselves and request an *Access Token*.

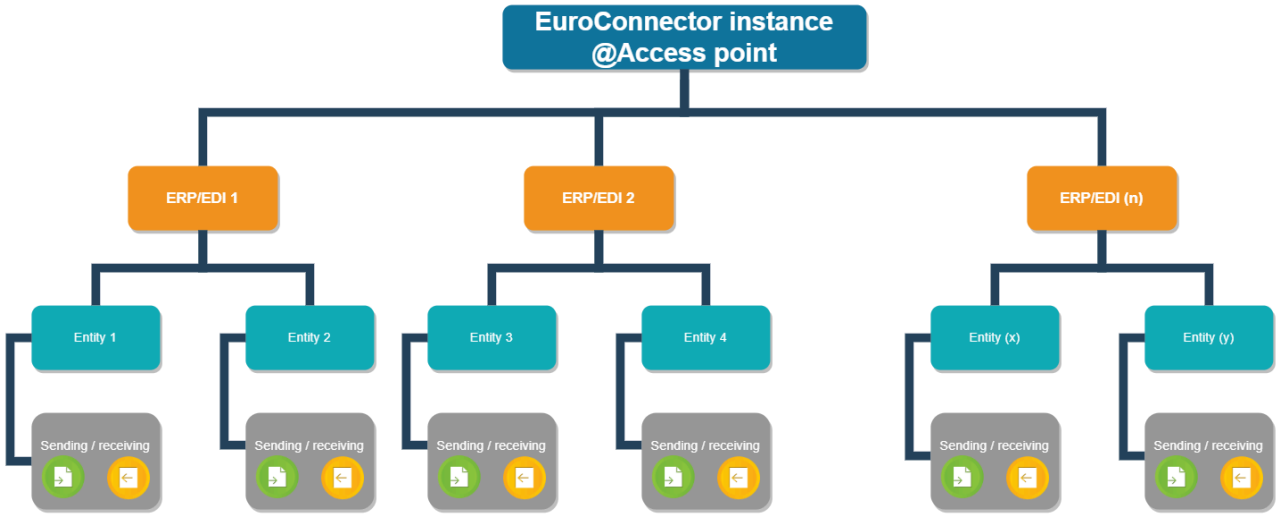
With the *Access Token*, the Client requests access to the resources from the EuroConnector web API server.

In this case, the token-based authentication works by ensuring that each request to a server is accompanied by a signed token which the server verifies for authenticity and only then responds to the request.

3 EUROCONNECTOR OPEN API DEFINITION

3.1 Web API structure

There are two main parts in this API hierarchy which are intended for the ERP/EDI provider and an Entity (Company). See once again this picture:



3.1.1 Features available for an ERP/EDI provider

ERP/EDI provider can:

- get authorized to access the system
- manage owned Entities (onboard to Peppol, update information, delete them)
- get billing information
- access information about the system status

3.1.2 Features available for an Entity (Company)

An Entity is able to:

- get authorized to access the system
- manage documents' content (validate, transform*)
- send and receive documents, also get or update the status of a document
- access information about the system status

*Transformation services depend on client-side applications and it is not a part of the standard definition.

3.2 Standard API features

Full and detailed version of the API definition can be found at this url:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def

- **Authorization section**

3.2.1 Authentication and authorization

The system has two types of users: an ERP and an Entity.

The Admin of the implementation instance creates an account for an ERP. And then, the ERP Admin creates an account for an Entity.

When an ERP/EDI or an Entity has valid credentials (userName/secretKey) they can authenticate themselves in the system.

Check the roles by calling:

```
GET /api/public/v1/Authorization/roles
```

https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.0#/Authorization/get_api_public_v1_Authorization_roles

- *Bearer authentication*

A bearer authentication (also called token authentication) would be a standard authentication scheme for Euroconnector that involves security tokens called bearer tokens. The name "Bearer authentication" can be understood as "give access to the bearer of this token." The bearer token is a cryptic string, usually generated by the server in response to a login request. The client must send this token in the Authorization header when making requests to protected resources:

```
GET <some end-point url>
Authorization: Bearer <token>
```

The bearer usage in the Euroconnector solution:

- First the caller should retrieve the token by providing a UserName and a SecretKey:

```
POST /api/public/v1/Authorization/token-create
{
  "userName": "string",
  "secretKey": "string"
}
```

Look here for more details:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.0#/Authorization/post_api_public_v1_Authorization_token_create

- Then the caller uses this token in the rest of the operations of the API, providing the token in the Authorization header of the HTTP message, e.g.:

```
GET /api/public/v1/Documents/2233446655
Authorization: Bearer ed8192d3b93d4604b9a196a62386c01d
```

- When the access token has expired it should be replaced with a new one. A refresh token is a special token that is used to obtain more access tokens. This allows you to have short-lived access tokens without having to collect the credentials every time one expires. Refresh tokens have a longer lifetime than access tokens. The recommended lifetime for a refresh token is 24 hours.

  Create the token from a refresh token by using the Euroconnector method:

```
GET /api/public/v1/Authorization/token-refresh
Authorization: Bearer {refreshToken}
```

  Look here for more details:
  https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.0#/Authorization/get_api_public_v1_Authorization_token_refresh

- **The Entities section for ERP/EDI providers**

This section is handled by the ERP/EDI providers and allows them to manage their Entities (customers).

3.2.2 Peppol lookup of an Entity

This function allows us to check if the Entity already exists in the Peppol network.

Web call examples:

```
GET /api/public/v1/Entities/peppol-lookup/9937:LT01234567890

GET /api/public/v1/Entities/peppol-lookup/0200:132456789
```

- The response (200) example, when the participant exists:


```
{
  "peppolService": {
    "accessPoint": "crossinx GMBH Access Point Provider",
    "EndpointSchemeId": "0200",
    "ParticipantId": "0200:132456789",
    "ParticipantIdentifierSchemeId": "iso6523-actorid-upis",
    "Status": "active",
    "IsEnabled": true,
    "createdAt": "2023-10-31",
    "updatedAt": null
  },
  "customInformation": "MIIFxDCCA6ygAwIBAgIQOQucDayhbPyv9j...d8udCX"
}
```


- The response (400) example, when the participant was not found:


```
{
  "statusCode": 400,
  "message": "The requested Peppol participant 9937:LT01234567890 was not found in
SML edelivery.tech.ec.europa.eu"
}
```


Also, check here for more information:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Entities/get_api_public_v1_Entities_peppol_lookup__id_

3.2.3 Onboarding the Entity to the Peppol network

The ERP/EDI providers can create access to this API for their customers (Entities) and also onboard them to Peppol. During the onboarding process it's also important to understand how the Peppol ID works in the network.

The Peppol ID format consists of three parts:

- BusDox Participant Identifier prefix (ParticipantIdentifierSchemeId), default: "iso6523-actorid-upis";
- The identification scheme identifier of the Seller electronic address or Electronic Address Scheme (EAS), more known as EndpointSchemeId. Sample: "0200". The full list of codes is provided in Peppol documentation: https://docs.peppol.eu/poacc/billing/3.0/codelist/eas/
- the GLN/VAT of the organization, e.g. "123456789". It is the ID or VAT number of the country's Tax Authority

Sample of a full participant ID: `iso6523-actorid-upis::0200:123456789`

Based upon the real practice it uses simplified Peppol ID version which contains only

EndpointSchemeId and GLN/VAT, e.g. 0200:123456789

If the company is identified in the Peppol network using the Peppol Participant Identifier, for example VAT or GLN (Global Location Number), you're required to maintain this information in the master data for the company and the customer accounts.

Currently supported Identifier codes in Lithuania are:

- **0200**: LT Entity code schema (most popular)
- **9937**: LT VAT number schema

The ERP / EDI provider must provide the identifier and VAT/GLN values when onboarding its customer.

Also, the onboarded Entity should support a few document types which are handled by Access points mostly:

- Peppol BIS Billing UBL Invoice V3
- Peppol BIS Billing UBL CreditNote V3
- Peppol Message Level Response transaction 3.0 (optional)
- Peppol Invoice Response transaction 3.0 (optional)

All the information above could be simply checked by making a query to the public Peppol directories and specifying the Peppol ID in there, e.g.:
https://directory.peppol.eu/public/locale-en_US/menuitem-search?q=0200%3A111629419

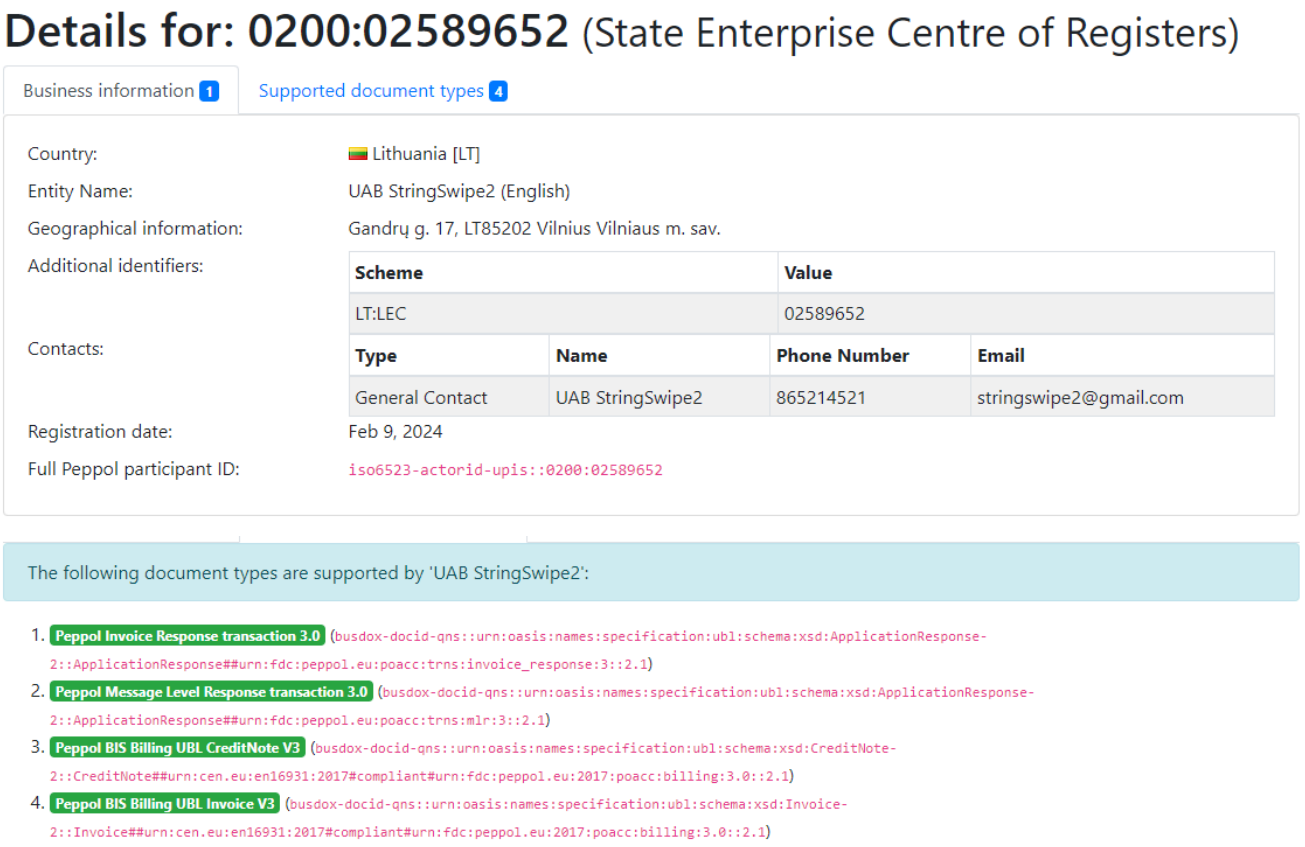
or

https://peppol.helger.com/public/locale-en_US/menuitem-tools-participant?scheme=iso6523-actorid-upis&value=0200%3A111629419&sml=autodetect&querybc=true&__querybc=true&__showtime=true&xsdvalidation=true&__xsdvalidation=true&verifysignatures=true&__verifysignatures=true&action=perform

- Euroconnector also supports the onboarding feature by using the end-point **/Entities/create** of an API definition:

  https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Entities/post_api_public_v1_Entities_create

The sample below represents the onboarded Entity with all the supported document types:



Details for: 0200:02589652 (State Enterprise Centre of Registers)

Business information 1 | Supported document types 4

Country:	Lithuania [LT]
Entity Name:	UAB StringSwipe2 (English)
Geographical information:	Gandrų g. 17, LT85202 Vilnius Vilniaus m. sav.
Additional identifiers:	Scheme: LT:LEC — Value: 02589652
Contacts:	Type: General Contact — Name: UAB StringSwipe2 — Phone Number: 865214521 — Email: stringswipe2@gmail.com
Registration date:	Feb 9, 2024
Full Peppol participant ID:	iso6523-actorid-upis::0200:02589652

The following document types are supported by 'UAB StringSwipe2':

1. Peppol Invoice Response transaction 3.0 (busdox-docid-qns::urn:oasis:names:specification:ubl:schema:xsd:ApplicationResponse-2::ApplicationResponse##urn:fdc:peppol.eu:poacc:trns:invoice_response:3::2.1)
2. Peppol Message Level Response transaction 3.0 (busdox-docid-qns::urn:oasis:names:specification:ubl:schema:xsd:ApplicationResponse-2::ApplicationResponse##urn:fdc:peppol.eu:poacc:trns:mlr:3::2.1)
3. Peppol BIS Billing UBL CreditNote V3 (busdox-docid-qns::urn:oasis:names:specification:ubl:schema:xsd:CreditNote-2::CreditNote##urn:cen.eu:en16931:2017#compliant#urn:fdc:peppol.eu:2017:poacc:billing:3.0::2.1)
4. Peppol BIS Billing UBL Invoice V3 (busdox-docid-qns::urn:oasis:names:specification:ubl:schema:xsd:Invoice-2::Invoice##urn:cen.eu:en16931:2017#compliant#urn:fdc:peppol.eu:2017:poacc:billing:3.0::2.1)

3.2.4 Retrieve the information about the Entity

Once the Entity is created it could be reviewed by calling a method with the argument of the entity {id}:

```
GET /api/public/v1/Entities/{id}
```

See the link for more details:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Entities/get_api_public_v1_Entities__id_

Response data sample:


```
{
    "entities": [
        {
            "entityInfo": {
                "entityId": "feca829e-31d9-425e-bdc1-e71e8b8e7d8e",
                "isEnabled": false,
                "createdAt": "2023-10-19T08:51:07.614093Z",
                "updatedAt": "2023-11-30T12:23:11.802492Z",
                "entityCode": "52365102",
```

```
                "vatNumber": "65152302",
                "entityName": "LTUAT ERPENTITY2",
                "emailAddress": "ltuaterpentity02@gmail.com",
                "phoneNumber": null,
                "street": "Dev st. 19",
                "locality": "Vilnius",
                "municipality": "Vilniaus m. sav.",
                "postalCode": "LT54613",
                "countryCode": "LT"
            },
            "userInfo": {
                "userName": "ltuaterpentity2user",
                "firstName": "Harry",
                "lastName": "Josh",
                "emailAddress": "ltuaterpentity02@gmail.com",
                "isEnabled": true,
                "createdAt": "2023-10-19T08:51:07.614093Z",
                "updatedAt": "2023-10-19T08:51:42.051945Z"
            },
            "peppolService": {
                "accessPoint": "crossinx",
                "endpointSchemeId": "0200",
                "participantId": "0200:52365102",
                "participantIdentifierSchemeId": "iso6523-actorid-upis",
                "status": "active",
                "isEnabled": true,
                "createdAt": "2023-10-19T08:51:07.614093Z",
                "updatedAt": "2023-10-19T08:51:41.988818Z"
            }
        }
    ]
}
```


3.2.5 Edit the Entity

Editing the information about the Entity is not an easy task as it requires synchronization between Access point and SML/SMP registry. The main identification arguments, such as registration number (entityNumber) or VAT number could be changed only by removing the Entity from the Peppol registry and registering it again with new details. So, editing information about the Entity is quite a bit limited.

Use the method below to edit the Entity:

```
PUT /api/public/v1/Entities/{id}/edit
```

A more detailed description is provided in the following link:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Entities/put_api_public_v1_Entities__id__edit

3.2.6 Generating (changing) the secret key

The secret key allows you to enter the system (see more details in the Authorization section). If the Entity wants to change or even loses the secret key, then the ERP/EDI provider may want to prepare a new one. This is done by calling the method:

```
PUT /api/public/v1/Entities/{id}/secret-key
```

See the link for more details:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Entities/put_api_public_v1_Entities__id__secret_key

3.2.7 Search the Entities

The Entities could be searched in the system by making more complex queries. The description is provided in the method:

https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Entities/post_api_public_v1_Entities_search

All the filled in arguments are connected with the logical operator “AND”. The argument will be ignored if it’s not set or is null or empty. Also, if the search string contains a wildcard character “%” then the search should be executed as a “LIKE” operator (instead of “EQUAL”).

Example:

- Request:

```
POST ../api/public/v1/Entities/search
Authorization : Bearer eyJh...oXqvDE
Content-Type : application/json
{
  "entityName": "%Good%",
  "createdBetween": ["2023-01-01T00:00:00Z", "2024-02-09T00:00:00Z"],
  "recordsCountLimit": 100
}
```

- Response (200):

```
{
    "entities": [
        {
            "userInfo": {
```

```
            "userName": "GoodComp092223",
            "firstName": "Jacob",
            "lastName": "Michael",
            "emailAddress": "goodcomp0923@gmail.com",
            "isEnabled": true,
            "createdAt": "2023-11-27T11:04:45.766139Z",
            "updatedAt": "2023-11-27T11:05:12.746996Z"
        },
        "peppolService": {
            "accessPoint": null,
            "endpointSchemeId": "9937",
            "participantId": "9937:651023542223",
            "participantIdentifierSchemeId": "iso6523-actorid-upis",
            "status": null,
            "isEnabled": true,
            "createdAt": "2023-11-27T11:04:45.765126Z",
            "updatedAt": "2023-11-27T11:05:12.743746Z"
        },
        "entityInfo": {
            "entityId": "344d3300-f9e6-41a2-8e31-1457969fd377",
            "entityCode": "023651542223",
            "vatNumber": "651023542223",
            "entityName": "UAB Good Company 092223",
            "emailAddress": "goodcompany092223@gmail.com",
            "phoneNumber": null,
            "street": "test st. 962223",
            "locality": "Vilnius",
            "municipality": "Vilniaus m. sav.",
            "postalCode": "LT69332",
            "countryCode": "LT",
            "isEnabled": false,
            "createdAt": "2023-11-27T11:04:45.764194Z",
            "updatedAt": "2024-02-15T13:07:49.347292Z"
        }
    },
    {
        "userInfo": {
            "userName": "GoodComp06",
            "firstName": "Tom",
            "lastName": "Thompson",
            "emailAddress": "goodcomp06@gmail.com",
            "isEnabled": true,
            "createdAt": "2023-11-15T13:11:50.268877Z",
            "updatedAt": "2023-11-15T13:12:18.276643Z"
        },
        "peppolService": {
            "accessPoint": null,
            "endpointSchemeId": "0200",
            "participantId": "0200:02365151",
            "participantIdentifierSchemeId": "iso6523-actorid-upis",
            "status": null,
            "isEnabled": true,
            "createdAt": "2023-11-15T13:11:50.267199Z",
            "updatedAt": "2023-11-15T13:12:18.239773Z"
        },
        "entityInfo": {
            "entityId": "8c59841f-e9b1-40b3-b8b2-44bc52dd5527",
```

```
                "entityCode": "02365151",
                "vatNumber": "65102351",
                "entityName": "UAB Good Company 06",
                "emailAddress": "goodcompany06@gmail.com",
                "phoneNumber": null,
                "street": "test st. 93",
                "locality": "Vilnius",
                "municipality": "Vilniaus m. sav.",
                "postalCode": "LT69326",
                "countryCode": "LT",
                "isEnabled": true,
                "createdAt": "2023-11-15T13:11:50.265988Z",
                "updatedAt": "2023-11-15T13:11:50.265988Z"
            }
        }
    ]
}
```


3.2.8 Deleting the Entity

The ERP/EDI provider can delete the Entity and unboard it from the Peppol network.

See the link for more details:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Entities/delete_api_public_v1_Entities__id_

Example:

- Request:

```
DELETE ../api/public/v1/Entities/344d3300-f9e6-41a2-8e31-1457969fd377
Authorization : Bearer eyJhb...oXqvDE
Content-Type : application/json
```

- Response (200):


```
{
    "statusCode": 200,
    "message": "The Entity 344d3300-f9e6-41a2-8e31-1457969fd377 was deleted
successfully"
}
```


- Response (400) if some errors were raised:


```
{
    "traceId": "299b1497-cdc0-463b-b59e-0324a3cf8d0e",
    "statusCode": 400,
    "message": "Bad request error.",
    "data": [
```

```
        {
            "propertyName": "id",
            "message": "The value '344d3300-f9e6-41a2-8e31-1457969fd377' is not
valid."
        }
    ]
}
```


It is worth noting that the removal of the company from the Peppol network is also not instantaneous, so you need to wait until the information travels between all the registers. After this step, always check the publicly available Peppol catalogs.

- **Documents section**

3.2.9 Available document types

Check the available document types which are provided by the Access Point provider by calling the API method **/Documents/types**:

https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Documents/get_api_public_v1_Documents_types

The list of available document types according the Eurconnector standard:

- Invoice
- CreditNote
- MessageLevelResponse (optional)
- InvoiceResponse (optional)

Those document types should be registered during the onboarding process of an Entity. Keep in mind that the optional ones may not be supported by some Access point providers.

- *Typical use case for invoicing:*
  - **Supplier** sends the Invoice or CreditNote to the **Customer**;
  - **Customer's AP** responds to the **Supplier's AP** with the MessageLevelResponse which includes technical information (invoice validation issues, etc.);
  - After evaluating the content of the invoice, the **Customer** responds to the **Supplier** with an invoice response message, which already contains information related to the business logic. It is important to note that the response should include a reference to the received invoice number, which is found in the cbc:ID element.

3.2.10 Validating a document

It is possible to validate a document without sending it out. Validation engine by default follows the rules of Peppol BIS Billing 3.0 standard. It's also possible to validate other types of standard if the Access point supports them.

- See section **/Documents/validate** in an API definition

https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Documents/post_api_public_v1_Documents_validate

Example:

- Request:

```
POST /api/public/v1/Documents/validate
{
  "documentStandard": "BIS3",
  "documentContent": "PEludm9pY2UgeG1sbnM9InVybjpvYXNp..."
}
```

- Response (200):

```
{
    "statusCode": 200,
    "message": "BIS3 document has been validated successfully."
}
```

- Response (400) with some errors:

```
{
    "TraceId": "97c35cf4-0195-4b5d-8baa-33e4b4337dc3",
    "StatusCode": 400,
    "Message": "Invalid document type.",
    "Data": null
}
```

3.2.11 Sending Invoice or CreditNote documents

The authenticated and authorized Entity can send documents to the recipient via the Peppol network. In that case, the **asynchronous** method described below should be called:

```
POST /api/public/v1/Documents/send
{
  "documentStandard": "BIS3",
  "documentContent": "Base64-string of Invoice/CreditNote xml"
}
```

and the response like in the sample below should be received:

```
{
  "documents": [
    {
      "documentId": "ed8192d3-b93d-4604-b9a1-96a62386c01d",
      "documentStandard": "BIS3",
      "documentType": "Invoice",
      "status": "new",
      "folderName": "outbox"
    }
  ]
}
```

As the request method is asynchronous, the sender should check the document status after a few moments, until it gets "sent" and "delivered" by using the method GET /Documents/{id} and also provide **documentId** as an argument.

It's important to note that the sender should prepare a valid e-invoice UBL data which fits the Peppol BIS Billing 3.0 standard (https://docs.peppol.eu/poacc/billing/3.0/).

The Peppol ID of the recipient should be placed in the UBL section:

```
/cac:AccountingCustomerParty/cac:Party/cbc:EndpointID
```

Example snippet of an Invoice/CreditNote:

```
<cac:AccountingCustomerParty>
 <cac:Party>
  <cbc:EndpointID schemeID="0200">123456789</cbc:EndpointID>
  ...
```

According to the example, the Peppol ID of the recipient is "0200:123456789"

- See more detailed description of a method in an API definition: https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Documents/post_api_public_v1_Documents_send

3.2.12 Sending response documents

Sending the responses technically works in the same way as sending invoices - by calling the end-point **/Document/send**. The only difference is the argument [documentContent] which should be prepared according the standard of MesageLevelResponse or InvoiceResponse:

```
POST /api/public/v1/Documents/send
{
  "documentStandard": "BIS3",
  "documentContent": "Base64-string of MessageLevelResponse/InvoiceResponse xml"
```

```
}
```

Also, keep in mind that the responses should contain the reference to the original document. The example below represents the connection between invoice and the response.

- Received invoice:

```
<Invoice xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2" xmlns:cac="urn:oasis:names:specification:ubl:sch
  <cbc:CustomizationID>urn:cen.eu:en16931:2017#compliant#urn:fdc:peppol.eu:2017:poacc:billing:3.0</cbc:CustomizationID>
  <cbc:ProfileID>urn:fdc:peppol.eu:2017:poacc:billing:01:1.0</cbc:ProfileID>
  <cbc:ID>TEST1234567</cbc:ID>
  <cbc:IssueDate>2022-11-30</cbc:IssueDate>
  <cbc:DueDate>2022-12-30</cbc:DueDate>
  <cbc:InvoiceTypeCode>380</cbc:InvoiceTypeCode>
  <cbc:DocumentCurrencyCode>EUR</cbc:DocumentCurrencyCode>
```

- The reference in the response to the received invoice:

```
    <cac:DocumentReference>
      <cbc:ID>TEST1234567</cbc:ID>
      <cbc:DocumentTypeCode>380</cbc:DocumentTypeCode>
    </cac:DocumentReference>
```

3.2.13 Retrieving document information

When a document is sent out or received, it always gets an **{id}** which is the main identifier of the document. Pass it to the method below in order to retrieve the detailed information of a particular document:

```
GET  /api/public/v1/documents/{id}
```

Example:

- Request:

```
GET ../api/public/v1/Documents/469f73a6-0cf8-49bc-9d66-95d084f9f7b8
Authorization : Bearer eyJhbG...m9ME2EQlNydx4
Content-Type : application/json
```

- Response (200) sample:

```
{
  "documents": [
      {
          "documentId": "469f73a6-0cf8-49bc-9d66-95d084f9f7b8",
          "documentNo": "CLT001202",
          "documentStandard": "BIS3",
          "documentType": "CreditInvoice",
```

```
            "senderEndpointId": "0200:111629419T1",
            "senderName": "Unifiedpost, UAB",
            "senderEntityCode": "111629419T1",
            "senderVatNumber": "LT100426314",
            "recipientEndpointId": "0200:56412801",
            "recipientName": "ERP2ENTITY1 LTUAT",
            "recipientEntityCode": "56412801",
            "recipientVatNumber": "LT887648610",
            "createdAt": "2023-11-03T12:23:08.318059Z",
            "updatedAt": "2023-11-03T12:23:10.280976Z",
            "status": "delivered",
            "statusNotes": null,
            "folderName": "outbox",
            "customFields": [
                {
                    "field": "documentIdExt",
                    "value": "4127926::"
                }
            ]
        }
    ]
}
```


- See more details in an API definition: https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Documents/get_api_public_v1_Documents__id_

3.2.14 Receiving documents

The procedure of receiving documents is a bit more complex than sending them out. The workflow contains several actions:

- collecting new documents
- retrieving documents information
- downloading documents' content
- changing documents status

- *Collecting new documents*

  A collection of the new documents is created by triggering the end-point:

  ```
  GET /api/public/v1/documents/received-list
  ```

  This method looks for the documents with the status "**new**" in the folder "**inbox**" and gets an array of documents in return.

  Sample:
  The request for new documents:

  ```
  GET /api/public/v1/Documents/received-list
  Authorization : Bearer eyJhbGciOiJIUzI1...6Ixfc3oDvS2-4
  Content-Type : application/json
  ```

- Response (200) with an empty array (no new documents were found):

```
{
  "documents": []
}
```

- Response (200), when it returns an array of 2 new documents:

```
{
  "documents": [
      {
          "documentId": "2e316446-d005-4190-b645-23cd32ac488a",
          "documentStandard": "BIS3",
          "documentType": "Invoice",
          "status": "new",
          "folderName": "inbox"
      },
      {
          "documentId": "33a73d80-582d-4243-9897-74eb7072d39b",
          "documentStandard": "BIS3",
          "documentType": "InvoiceResponse",
          "status": "new",
          "folderName": "inbox"
      },
  ]
}
```

More information can be found at API definition:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Documents/get_api_public_v1_Documents_received_list

- *Retrieving information of a document without downloading it*

  Optionally, if you want just to check a metadata of a new document without downloading the content, you can do it by calling:

```
GET /api/public/v1/documents/{id}
```

More information can be found here:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Documents/get_api_public_v1_Documents__id_

Example:

- Request:

```
GET ../api/public/v1/Documents/37afbaec-967f-4bb2-a558-732c16498a66
Authorization : Bearer eyJhbGciOiJIUzI...C1z7m8
Content-Type : application/json
```

- Response:


```
{
    "documents": [
        {
            "documentId": "37afbaec-967f-4bb2-a558-732c16498a66",
            "documentNo": "UPG00000213",
            "documentStandard": "BIS3",
            "documentType": "Invoice",
            "senderEndpointId": ":",
            "senderName": null,
            "senderEntityCode": null,
            "senderVatNumber": null,
            "recipientEndpointId": "0200:304075793",
            "recipientName": "Plieno studija, UAB",
            "recipientEntityCode": "304075793",
            "recipientVatNumber": null,
            "peppolMessageId": null,
            "documentReference": null,
            "createdAt": "2024-02-13T12:35:30.747799Z",
            "updatedAt": "2024-02-13T12:58:38.981726Z",
            "status": "New",
            "statusNotes": "Access point info. Inbound:PROCESSING",
            "folderName": "outbox",
            "customFields": [
                {
                    "field": "documentIdExt",
                    "value": "4133474::"
                }
            ]
        }
    ]
}
```


- *Downloading the content of a document*

  Make a loop through the list of the new documents and download each document separately by the attribute **documentId** by calling the endpoint:

```
GET /api/public/v1/documents/{id}/content
```

More information can be found here:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Documents/get_api_public_v1_Documents__id__content

Example:

- Request:

```
GET ../api/public/v1/Documents/4e06d546-ab4d-4a8e-bd15-655d009c9db5/content
Authorization : Bearer eyJhbGciO..QlNydx4
Content-Type : application/json
```

- Response (200):

```
{
    "documentId": "4e06d546-ab4d-4a8e-bd15-655d009c9db5",
    "documentStandard": "BIS3",
    "documentContent": "PEludm9pY2UgeG1sbnM9...uZT4KICA8L0ludm9pY2U+"
}
```

- *Changing the status of received document*

When a document is successfully downloaded it should be marked as “received” by the receiver in the EuroConnector platform. Also, the status could be changed to “error” or “held” if some exceptions have been raised.

This is done by calling the end-point:

```
PUT /api/public/v1/documents/{id}/status/{status}
```

Important to know that if the status of a document was not changed, then the same document will be collected in the next “received-list” call.

More information can be found here:
https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Documents/put_api_public_v1_Documents__id__status__status_

Example of how to set the particular document status to “error”:

- Request:

```
PUT
../api/public/v1/Documents/9e9fe587-0fea-48ed-ab27-9676f38ead79/status/error
Authorization : Bearer eyJhbG...1z7m8
Content-Type : application/json
```

- Response:

```
{
    "documentId": "9e9fe587-0fea-48ed-ab27-9676f38ead79",
    "documentStandard": "BIS3",
    "documentType": "Invoice",
```

```
    "status": "Error",
    "folderName": "inbox"
}
```

3.2.15 Searching documents

The documents could be searched in the system by making more complex queries. The description is provided in the method:

```
POST /api/public/v1/Documents/search
{
  "documentId": "ed8192d3-b93d-4604-b9a1-96a62386c01d",
  "documentNo": "ABC123%",
  "documentType": "Invoice",
  "senderEndpointId": "0200:123456789",
  "senderName": "Trys paršiuk%",
  "senderEntityCode": "123456789",
  "senderVatNumber": "string",
  "recipientEndpointId": "0200:987654321",
  "recipientName": "Trys paršiuk%",
  "recipientEntityCode": "987654321",
  "recipientVatNumber": "string",
  "status": "new",
  "folderName": "outbox",
  "recordsCountLimit": 100
}
```

https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Documents/post_api_public_v1_Documents_search

All the filled in arguments are connected with the logical operator “AND”. The argument will be ignored if it’s not set or is null or empty. Also, if the search string contains a wildcard character “%” then the search should be executed as a “LIKE” operator (instead of “EQUAL”).

Example:

```
POST /api/public/v1/Documents/search
{
  "documentNo": "ABC123%",
  "documentType": "CreditNote",
  "senderEndpointId": null,
  "folderName": "outbox",     // required
  "createdBetween": ["2023-09-01T00:00:00Z", "2023-09-02T00:00:00Z"],
  "recordsCountLimit": 10     // will be 100 if not present
}
```

According to the example, the query returns the first 10 (ten) “credit note” documents in the “outbox” folder, where their number starts with “ABC123” and documents are created on September 1st, 2023.

The SQL statement below could explain more detail how the search engine should work for that particular example:

```
SELECT TOP(10) *
  FROM documents_table
  WHERE
       documentNo LIKE 'ABC123%'    -- "LIKE" is because it has a wildcard "%"
   AND documentType = 'CreditNote'
   AND createdAt BETWEEN '2023-09-01 00:00:00Z' AND '2023-09-02 00:00:00Z'
   AND folderName = 'outbox'
```

- **Extensions section**

3.2.16 Check the active version of the API instance

You can check the active version of the API implementation by calling a method:

```
GET /api/public/v1/Extensions/version
```

https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Extensions/get_api_public_v1_Extensions_version

3.2.17 The report for ERP/EDI of sent / received documents

The reports of sent/received documents may be necessary for the ERP / EDI providers. This one gives the statistics to the provider on how many documents have been sent out or received by their customers (Entities).

The example below presents the report use.

The request:

```
GET /api/public/v1/Extensions/billing-report/2023-10-01/2023-10-31
Authorization : Bearer eyJhbGciOiJIUzI1...6Ixfc3oDvS2-4
Content-Type : application/json
```

Returns the report with the billing information:

```
{
  "reportId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "billingPeriodFrom": "2023-10-01",
  "billingPeriodTo": "2023-10-31",
  "billingInfo": [
    {
      "participantId": "0200:123456789",
      "documentsSent": 10,
      "documentsReceived": 7
```

```
    },
    {
      "participantId": "0200:987654321",
      "documentsSent": 123,
      "documentsReceived": 58
    }
  ]
}
```


- See section **/Extensions/billing-report** in an API definition: https://app.swaggerhub.com/apis-docs/euroconnector/econ-def/1.1#/Extensions/get_api_public_v1_Extensions_billing_report__periodFrom___periodTo_

4 API IMPLEMENTATION INSTRUCTIONS

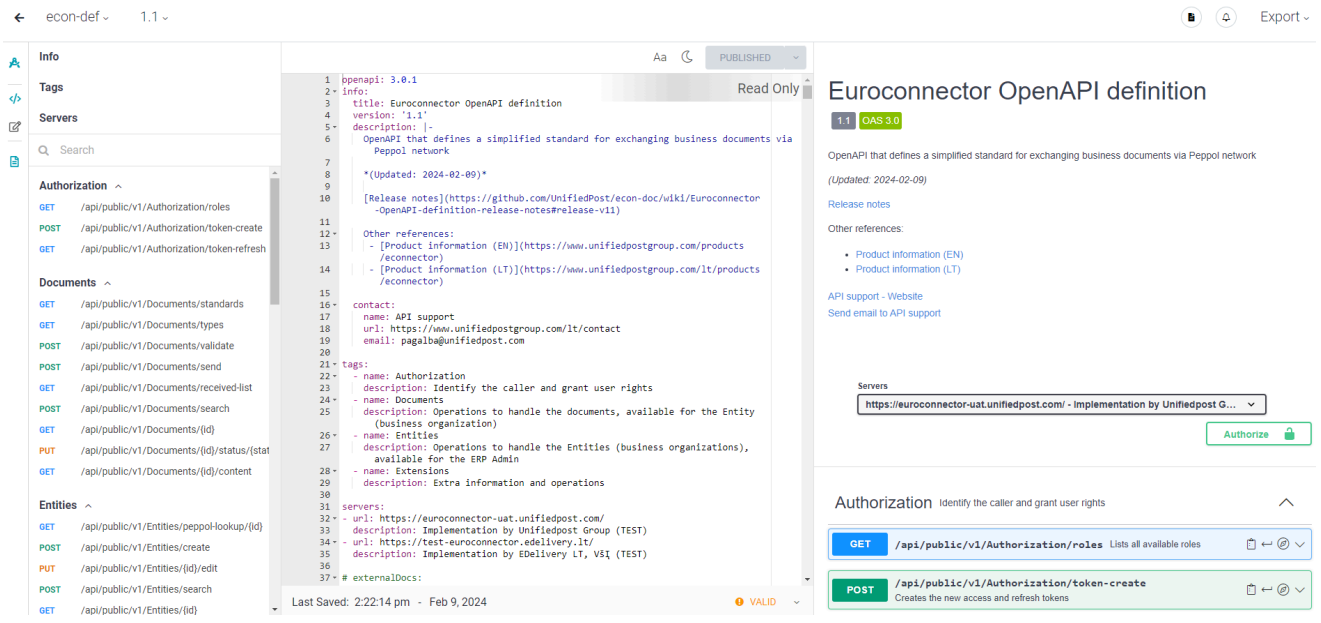
To complete this tutorial, you need a basic knowledge of APIs.

The specification of the Euroconnector OpenAPI definition is developed on the SwaggerHub platform. This is an online platform where you can design your APIs – be it public APIs, internal private APIs or microservices. The core principle behind SwaggerHub is "Design First, Code Later". That means, you start by laying out your API, its resources, operations and data models, and once the design is complete you implement the business logic.

The Euroconnector API definitions are written in the OpenAPI format. They are saved in the SwaggerHub cloud and can be synchronized with external systems like GitHub or Amazon API Gateway. You can also collaborate with your team on SwaggerHub, and maintain multiple versions of APIs as it evolves.

To start with the implementation, please visit the link below
https://app.swaggerhub.com/apis/euroconnector/econ-def

You'll get a view like this:



An API page on SwaggerHub is a split view that shows the YAML code of this OpenAPI definition and reference-style API documentation generated from it. The API documentation allows the users to test API calls directly in the browser. The navigation panel on the left shows a list of operations and models defined in your API and lets you jump to the corresponding place in the YAML code. You can resize the panels by dragging the splitter between them.

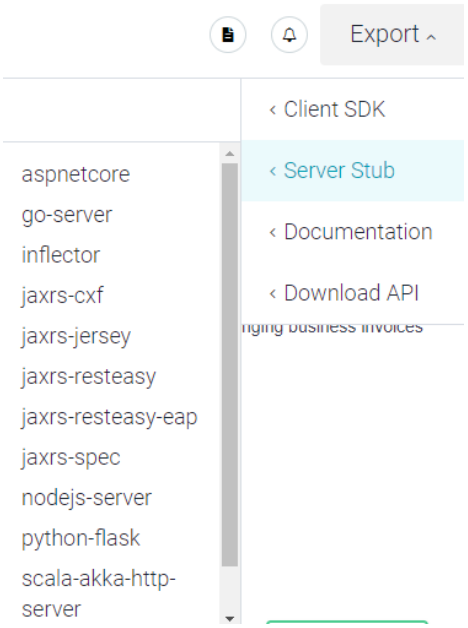
Please visit a link below to know more about Swagger:
https://support.smartbear.com/swaggerhub/docs/tutorials/getting-started.html

4.1 Server side implementation

Server side implementations are dedicated to the Access point providers.

- Implementation using SwaggerHub code packages

To continue with this, please open the same SwaggerHub view. Then you may want to choose the code samples with your preferable programming language. Then select the menu item [Export] > [Server Stub] > [.....] and you'll get various options of code snippets:



Select to download that package of code, put it in your software designing tool, add your custom code into the particular parts of the API methods.

- OpenAPI source code snippets from live implementation by UPG

This chapter provides the code samples which could be used while implementing Euroconnector server-side API. The code is written in C# programming language and designed by the "Visual Studio 2022" tool.

You can start by downloading the code package from the open source repository https://github.com/UnifiedPost/econ-doc

The package contains the folders:

EuroConnector	Root folder of this API
Clients	Connectivity with 3rd parties. Mostly it could be the AP and other supporting clients like web services, ftp or local storage
Controllers	Describes the API method types, structure, routing and Swagger UI setup
DTOs	Describes the Data transfer objects for API requests and responses
Helpers	Some custom actions
Services	Business logic, relations with the databases
XsltTransformer	Separate project to work with Peppol schematrons
Interfaces	the logic to work with schematrons
xslt	schematrons location

Keep in mind that this code is limited and it has no connectivity to the AP and other local services whose structure cannot be disclosed.

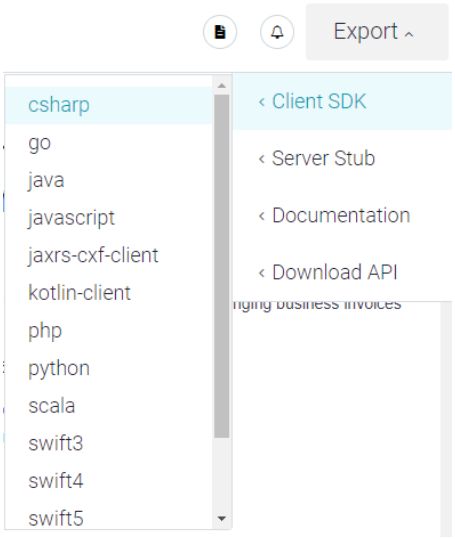
In general, it is recommended to make a mix of both implementations. Use the SwaggerHub platform to load the API structure and include some code snippets from the live implementation by UPG. And of course, add your custom code which connects the API with the AP.

4.2 Client side implementation

Client side implementations are dedicated to the ERP/EDI providers, also the Entities who are sending/receiving invoices.

- Implementation using SwaggerHub code packages

Open the same SwaggerHub view. Select the menu item [Export] > [Client SDK] > [.....] and you'll get various options of code snippets:

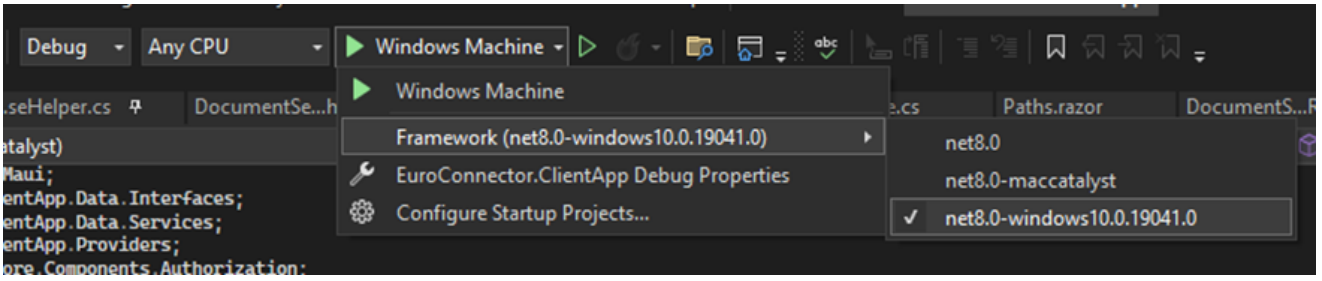


Select to download that package of code. You'll get an empty package with the web calls structure to OpenAPI. Put it in your software designing tool, add your custom code into the particular parts.

■ Open-source code of the client-app

You can download a full version of the the client-app last release (C# programming language) from: https://github.com/UnifiedPost/econ-client-app

In order to start it in a debug mode, please open the solution file (*.sln) using the "Visual Studio 2022" tool. Then choose the framework "net8.0-windows10.0.19041.0" as it is presented in the picture below:

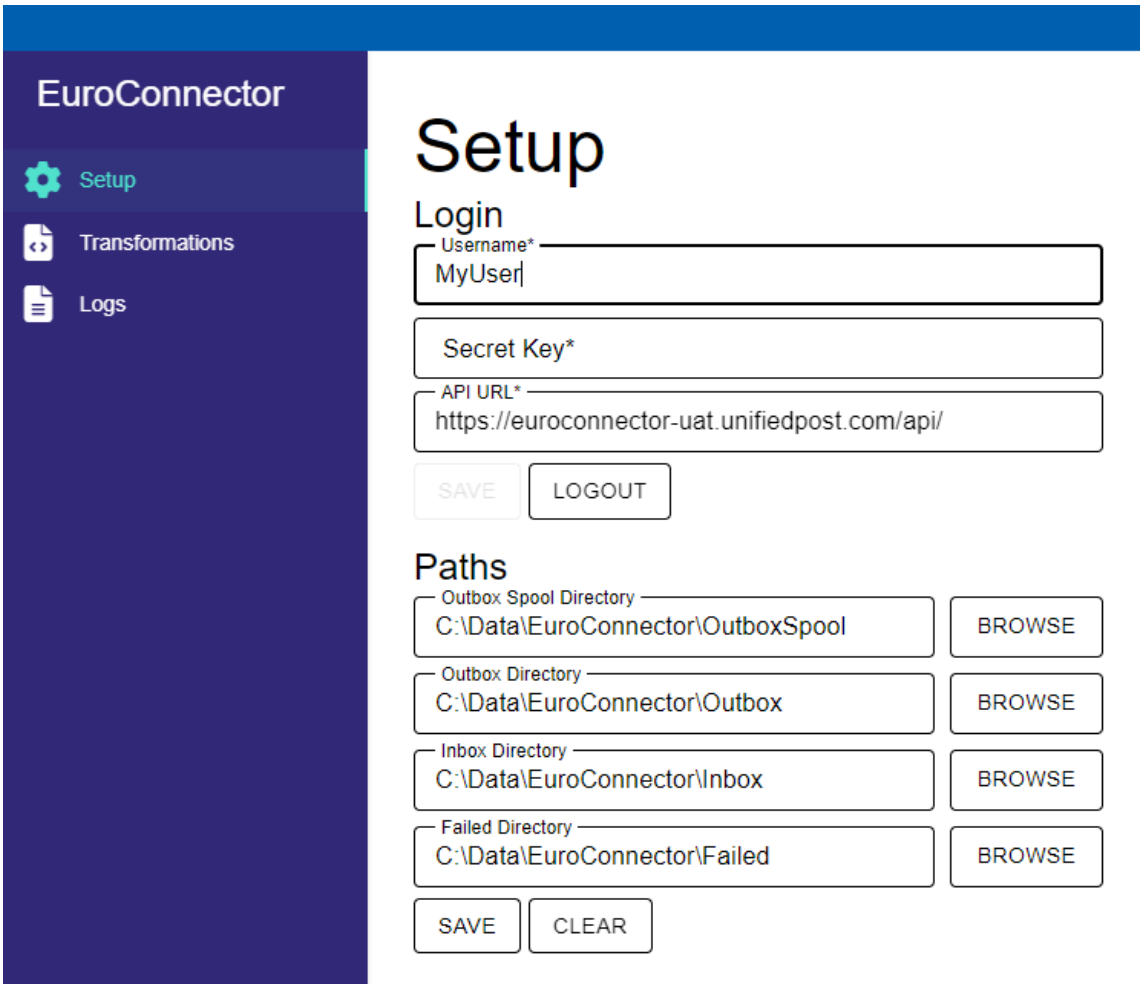


5 CLIENT APP IMPLEMENTATION INSTRUCTIONS

5.1 Executable version

The executable version of the Euroconnector Client app could be downloaded from GitHub url: https://github.com/UnifiedPost/econ-client-app/wiki

Just simply install the application, then set up the main configuration arguments and you can already start using it.



These main arguments are required to setup in order to start working with application:

- Login and API URL information, which should be provided by your ERP / EDI provider if you sign an agreement or pass the provider's KYC procedures.
- Local folders setup, where the sent and received documents files will be saved.

The rest actions in the application are enabled when you login to the API provided by the Access point. You can find out more about how to use the client-app in the detailed user manual: https://github.com/UnifiedPost/econ-client-app/wiki/User-manual