SEA2SEE

INNOVATIVE BLOCKCHAIN TRACEABILITY TECHNOLOGY AND STAKEHOLDERS' ENGAGEMENT STRATEGY FOR BOOSTING SUSTAINABLE SEAFOOD VISIBILITY, SOCIAL ACCEPTANCE AND CONSUMPTION IN EUROPE

DELIVERABLE D3.1 – SEA2SEE BLOCKCHAIN MODEL WITH ALL THE CONNECTOR TOOLS

Lead Partner Organization	Tilkal
Due date	30-Jun-24
Issue date	7-Jul-24

Co-funded by
the European Union

Document information

Settings	Value
Deliverable Title	Sea2See Blockchain model with all the connector tools
Work Package Number & Title	WP3 - Traceability technologies development
Deliverable number	D3.1 D9
Description	Conception and development of a set of tools to install and operate blockchain nodes in all data capture locations.
Lead Beneficiary	Tilkal
Lead Authors	Adeline Caffin, Sébastien Gaïde
Contributors	/Person /s name/s/
Submitted by	Carlos Mazorra

Review History

Version	Date	Reviewer	Short Description of Changes
1	5-Jul-24	Carlos Mazorra	Minor changes

Document Approval

Name	Role	Action	Date
Carlos Mazorra	Project Coordinator	Approved	7-Jul-24

Nature of the deliverable

R	**Document, report (excluding the periodic and final reports)**	☐
DEM	Demonstrator, pilot, prototype, plan designs	☐
DEC	Websites, patents filing, press & media actions, videos, etc.	☐
DATA	Data sets, microdata, etc.	☐
DMP	Data management plan	☐
Ethics	Deliverables related to ethics issues.	☐
SECURITY	Deliverables related to security issues	☐
Other	Software, technical diagram, algorithms, models, etc.	☒

Dissemination level

PU	**Public — fully open (automatically posted online on the Project Results platforms)**	☒
SEN	Sensitive — limited under the conditions of the Grant Agreement	☐

ACKNOWLEDGEMENT

This report forms part of the deliverables from the project Sea2See which has received funding from the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101060564.

Current seafood traceability tools and services have the potential to take advantage of novel blockchain technologies to obtain a wide range of data making sustainable seafood practices more visible to consumers. Sea2See project will fill in existing seafood traceability gaps through development and demonstration of an innovative end-to-end blockchain traceability model throughout the seafood value chain and professional and consumer applications to increase trust and social acceptance of sustainably fished and farmed seafood.

The project will provide technological solutions to answer the need of a valuable source of data collected throughout the whole seafood value chain, verified, and covering inputs from diverse stakeholders. For that purpose, a specific focus will be put on active commitment of stakeholders and real empowerment of consumers through the implementation of societal and sectoral strategies for co-creation, communication and awareness raising.

The project runs from July 2022 to June 2026. It involves 14 partners from 6 EU countries, and is coordinated by SMARTWATER PLANET SL, Spain.

More information about the project can be found at: http://www.sea2see.eu/

COPYRIGHT

EXECUTIVE SUMMARY

This document describes the conception and development of the Sea2See blockchain based traceability platform tools, used to install, operate and leverage a blockchain node in all data capture locations of the project.
Beyond explaining how to become a traceability stakeholder in detail, it presents the advantages of becoming an actor and the motivation of the project to make the onboarding as easy as possible.

ACRONYMS AND ABBREVIATIONS

ACRONYM	DEFINITION
JSON	JavaScript Object Notation
JWT	JSON Web Token
EPCIS	Electronic Product Code Information Services
GLN	Global Location Number
GTIN	Global Trade Item Number
API	Application Programming Interface
SaaS	Software as a Service
LTS	Long-Term Service
XML	Extensible Markup Language
TCP	Transmission Control Protocol
HTTP(S)	Hypertext Transfer Protocol (Secure)
TLS	Transport Layer Security
DNS	Domain Name Server

PROJECT PARTNERS

#	Partners full name	Short	Country	Website
1	SMARTWATER PLANET SL	SmartWater	ES	www.smartwaterplanet.com
2	TILKAL	Tilkal	FR	www.tilkal.com
3	PAGE UP	PAGE UP	FR	www.pageup.fr
4	SUBMON	SUBMON	ES	www.submon.org
5	CENTRO DE CIENCIAS DO MAR DO ALGARVE	CCMAR	PT	www.ccmar.ualg.pt
6	ASOCIACION NACIONAL DE FABRICANTES DE CONSERVAS DE PESCADOS Y MARISCOS-CENTRO TECNICO NACIONAL DE CONSERVACION	ANFACO	ES	www.anfaco.es

	DE PRODUCTOS DE LA PESCA			
7	IOANNA N.ARGYROU SIMBOULOI EPICHEIR ISIAKIS ANAPTYXIS ETAIREIA PERIORISMENIS EYTHYNIS	NAYS	EL	www.nays.gr
8	SEAENTIA-FOOD, LDA	SEAentia	PT	www.seaentia.com
9	LANDLNG AQUACULTURE BV	LA	NL	www.landingaquaculture.com
10	UNIVERSIDADE DE AVEIRO	UAVR	PT	www.ua.pt
11	VITAGORA POLE	VITAGORA	FR	www.vitagora.com
12	ETHIC OCEAN	Ethic Ocean	FR	www.ethic-ocean.org
13	EVROPROJECT OOD	EP	BG	www.europroject.bg
14	ANP - ASSOCIACAO NATUREZA PORTUGAL	ANP	PT	www.natureza-portugal.org

TABLE OF CONTENTS

TABLES AND FIGURES

1. HOW TO BECOME A TRACEABILITY NETWORK PLAYER

1.1. WHY

To make a traceability project a success, data is paramount. In a supply chain of any product, a lot of data is generated, stored and used by all the actors. But usually, not shared amongst the actors. Confidentiality, process secrets, security and privacy are valid motivations not to share data.

Heterogeneous data format and data sources is also an obstacle to interoperability. Each actor may use a different system to manage its data. To share it usually means to be able to extract it and convert it to a common format. A tedious process that needs some technical skills, not always available.

Cost, engagement, project management, compliance to local laws, all these obstacles make it even more complex to succeed in a traceability project.

The Sea2See traceability platform includes a set of tools and connectors, to make this process easier:

- Easier onboarding by leveraging blockchain network.
- Easier data collection by providing a common language, based on a globally recognized standard (GS1/EPCIS).
- Easier data sharing made possible by using well known mechanism like “channels”.
- Easier communication with consumers by exposing chosen information about the product they have in hands.

1.2. HOW

Becoming a traceability network player means to be willing to be more transparent, to make the good practices you have adopted more visible, and this comes with responsibilities:

- Ensuring the accuracy, consistency, and completeness of data you share with other actors.
- Involving your personnel into a data collection process, your IT team into operating a blockchain node.
- Involving your suppliers to collect data related to business you have with them.

All these responsibilities, having different complexity levels, are alleviated by using the Sea2See traceability platform:

- Installing a blockchain node is a “few minutes” process, thanks to low technical requirements and new application deployment technologies (docker).
- Data collection process is eased by using wizard-based web forms or mobile applications on the field, making data typing less error prone.

- Data restitution by the Tilkal Suite SaaS application encourages the actor to share accurate data.
- Campaign management provided by the platform makes it easy to involve actor's suppliers, making them benefit the same collection tools set.

Traceability actor gets all this just by onboarding the project installing a blockchain node.

2. BLOCKCHAIN NODE INSTALLATION

2.1. PREREQUISITES

HARDWARE REQUIREMENTS

The default server configuration typically suitable to be a Sea2See blockchain node is low level. Table 1 shows two different hosting services examples :

Table 1

Hosting service	Instance type	Instance description
OVH	VPS Essential	2 vcpu, 4 GB ram, 80 GB disk
AWS	T3a.medium	2 vcpu, 4GB ram, disk size on demand

As explained in D3.6, small servers are enough in the context of a permissioned network, like the Sea2See one.

NETWORK REQUIREMENTS

The node needs to communicate with other nodes, so network connections must be allowed on some ports as shown in Table 2:

Table 2

Port range	Protocol	Direction	Description
6301	TCP	In/out	Peer-to-peer link with other nodes
443	TCP	In/out	Access to/from external (https)
80	TCP	In/out	Access from external (http used by Let's Encrypt)

For installation or update of the distribution packages and docker images, a temporary connection to Internet is required. When the node is in service, it will access the API of the Tilkal environment of the actor. No other connections will be made.
If the default node web API are to be used, then the instance must have a valid public DNS name, so that a TLS certificate can be requested and obtained using Let's Encrypt service (this is done

automagically by the http service component). If this name is given before the node installation, then the configuration file will already use it, otherwise the http service configuration file must be edited before starting the component (see below).
If the actor does not want to expose the instance publicly, then the TLS certificate must be provided by its IT service, matching a DNS name he controls internally.

SOFTWARE REQUIREMENTS

The Linux distribution can be one of these, minimum version is specified, but latest long-term service (LTS) is always recommended:

- Ubuntu (>= 16), server edition, with all patches installed.
  - Do not install docker using snap, but by following official docker Ubuntu installation
- Debian (>= 9), server edition, with all patches installed
- CentOS (>= 7), server edition, with all patches installed
- Other distributions known to be compatible with docker

To update all system packages launch (Ubuntu/Debian example):

```
sudo apt update && sudo apt upgrade -y
```

Then proceed to docker installation by following the official docker documentation, this will install docker itself and docker compose plugin.

USER AND GROUP CONFIGURATION

For the rest of this document, and unless otherwise specified, all docker and docker-compose commands should be launched with a user having docker privileges. Please see official Docker post install documentation to learn more about how to do this and why.

MONITORING

Optionally a monitoring system like netdata can be installed: various netdata plugins are available to monitor the node components. To install netdata as a root user:

```
wget -O /tmp/netdata-kickstart.sh https://get.netdata.cloud/kickstart.sh && sh /tmp/netdata-kickstart.sh
```

2.2. INSTALLATION

CONFIGURATION FILES

All the configuration files will be transmitted by Tilkal, using a single archive file (tgz). Once received, the archive must be uncompressed, and unarchived, using this command:

```
tar xzf node.tgz
```

The configuration files can be placed in the folder of your choice. In the rest of this document, we will use the folder /home/user/node in the command lines.

STARTING THE NODE COMPONENTS

Please note that after starting the services for the first time, if you restart the Linux instance, docker will restart all the services at boot time, as requested in the docker-compose.yml configuration files. So, you can reboot your instance after a maintenance operation, with nothing to do about the Tilkal services.

HTTPS ENDPOINT

An http(s) endpoint may be needed to send the data you want to notarize, depending on the infrastructure the node is placed in.
A default http(s) server is provided, but you can use you own http component to provide this functionality, to ensure that the security level of your company is fulfilled.
To enable https connections using the default server, please make sure that the node is reachable using a public DNS name, and that ports 80/tcp and 443/tcp are opened. The TLS certificate will be requested calling the Let's Encrypt service by default. To use you own certificate please edit the configuration file placed in /home/user/node/http/config/Caddyfile.
The public DNS name of the instance is set in the first line of this configuration file, for example:
https://node.example.com
Please verify that this name is the one you want to use for your node.

If you want to use your own tls certificate, you must edit the line beginning with the word `tls` : for exemple, instead of

```
tls webmaster@example.com
```

type in

tls /opt/caddy/config/ssl/my-certificate.crt /opt/caddy/config/ssl/my-key.key

then copy your certificate and key files in the http/config/ssl folder:
/home/user/node/http/config/ssl/my-certificate.crt and /home/user/node/http/config/ssl/my-key.key
(Adapt file names accordingly)

LAUNCH THE SERVICES

Once the configuration files are set up, please proceed to the installation by using these commands:

```
chmod u+x install.sh
./install.sh
docker-compose up -d
```

All services should start, if needed the tls certificate will be requested to Let's encrypt and will be available shortly.

3. NODE MONITORING

3.1. DOCKER MONITORING

BASICS

Since docker uses technologies embedded in the Linux kernel, you can use the usual system tools to monitor what is going on the server.
You can of course use some more specialized tools, like ctop or sysdig to go further.
docker-compose itself provides some basic tools, for example, in the `/home/user/node /pipeline folder the command:

```
docker-compose ps
```

will list the status of the services configured in the docker-compose.yml file.

GENERAL METRICS

The node components are running as docker containers, so that any tools dedicated to monitor a docker environment could be used.
As noted during the installation steps, you can use NetData, a versatile, very efficient tool that is docker aware. You can also use some other tools, like Sysdig or Datadog if you like. All these tools are compatible with Prometheus, a well spread open-source systems monitoring and alerting toolkit. Hence if your existing monitoring system is compatible with Prometheus, you can directly build additional dashboards dedicated to the node.
NetData is also a statsd server, so that you can aggregate all the collected metrics into an already existing monitoring application.

All these tools can be configured to notify you in case of a problem, for example when some thresholds (cpu usage, network errors, ram usage or disk free space) are crossed.

LOGS

By default, the logs produced by the Docker containers are sent to the `json-file` driver, so that they can be accessed from the `/home/user/node/<component name>` folder with a command like :

```
docker-compose logs <service>
```

where <service> is the name of a Tilkal service running inside the container. Services names can be obtained using this command in the component folder:

```
docker-compose ps
```

EXAMPLES

In /home/user/node:

```
docker-compose logs multichaind
```

If you want events to be prefixed with a timestamp, use:

```
docker-compose logs -t multichaind
```

If you want only the last 10 events, use:

```
docker-compose logs --tail=10 multichaind
```

Logs are rotated, so that they won't fill up the disk. This is configured in the docker-compose.yml configuration files:

```
logging: &logging
    driver: "json-file"
    options:
      max-size: "200k"
      max-file: "20"
```

If you use a log collection app, such as loggly or logentries/insightOps or datadog, you can set up an agent to collect the logs. The easiest way to do so is to redirect the logs to a system service like syslog or rsyslog; this can be configured directly in the /root/pipeline/docker-compose.yml file:

```
logging: &logging
    driver: syslog
    options:
      tag: "{{.Name}}"
```

This should be done only for the logging part of the first service, to apply this to all services.
Regarding the logs produced by the Tilkal components of the node, you can configure the log level, in the /home/user/node/pipeline/config/config.js file, in the winston.configure call:

```
winston.configure({
 transports: [
  new Console({ level: 'warn' }),
  new File({
    level: 'info',
    filename: 'pipe.log',
    maxsize: 20 * (1024 ** 2),
    maxFiles: 10,
    tailable: true,
    zippedArchive: true,
    format: winston.format.combine(
       winston.format.colorize(),
       winston.format.timestamp(),
       winston.format.printf(info => `${info.timestamp} ${info.level}: ${JSON.stringify(info.message)}`)
    )
  }),
 ]
})
```

In the above example two logs destinations are configured: the console, with a warn (warning) level, and a file, reachable at /pipe.log in the container, with a info level. Please note that only the logs sent to the console are seen by Docker and then redirected to json-file or syslog depending on the docker-compose.yml configuration file.

3.2. APPLICATION MONITORING

Tilkal node components expose some metrics to allow node monitoring. Especially, the message queues expose metrics about the number of messages that have been received, processed, rejected and their processing durations (min, max, avg).
A few netdata plugins are available to monitor these metrics easily: please contact Tilkal if you are interested to use them.

MESSAGE QUEUES

Internal node components are using message queues to process incoming data, these queues expose some endpoints to be monitored, by default a service named nsqadmin is configured to be started by docker-compose and is accessible using a web browser at http://localhost:4171.
A netdata plugin can also be used to collect the same information and be displayed in some charts.

TILKAL COMPONENTS

Internal components expose metrics that can be retrieved in a simple json format using standard http requests:

```
wget http://localhost:<service_port>/pipe/stats
```

will print :

```
{
 "ReceivedMessages":<number of received messages>,
 "ProcessedMessages":<number of processed messages>,
 "RequeuedMessages":<number of requeued messages>,
 "DiscardedMessages":<number of discarded messages>,
 "MinimumHandlingTime":<minimum handling time (ms)>,
 "AverageHandlingTime":<average handling time (ms)>,
 "MaximumHandlingTime":<maximum handling time (ms)
}
```

All service ports are configured in the /home/user/node/pipeline/docker-compose.yml configuration file.
A netdata plugin is also available to draw charts from this data.

BLOCKCHAIN

You can monitor the internals of the blockchain engine (number of transactions, number of blocks created, etc...). Here are typical useful commands to be run from /home/user/node/multichain:

```
source .env
docker-compose exec multichaind /usr/local/bin/multichain-cli $MC_CHAIN -rpcport=$MC_RPC_PORT -rpcuser=$MC_RPC_USER -rpcpassword=$MC_RPC_PASSWORD getinfo
docker-compose exec multichaind /usr/local/bin/multichain-cli $MC_CHAIN -rpcport=$MC_RPC_PORT -rpcuser=$MC_RPC_USER -rpcpassword=$MC_RPC_PASSWORD getmeminfo
docker-compose exec multichaind /usr/local/bin/multichain-cli $MC_CHAIN -rpcport=$MC_RPC_PORT -rpcuser=$MC_RPC_USER -rpcpassword=$MC_RPC_PASSWORD getpeerinfo
docker-compose exec multichaind /usr/local/bin/multichain-cli $MC_CHAIN -rpcport=$MC_RPC_PORT -rpcuser=$MC_RPC_USER -rpcpassword=$MC_RPC_PASSWORD liststreams
```

These commands allow to retrieve the following information as shown in Table 3:

Table 3

Command	Information
getinfo	general info about the node (number of blocks, protocol versions)
getmempoolinfo	size of transactions still being in memory
getpeerinfo	list the nodes connected to this node
liststreams	streams stats, the important stream is `event`, collected data are sent to this stream

A netdata plugin is also available to represent these metrics in a set of charts.

3.3. DISK MONITORING

In case of a heavy activity on the platform, monitoring the disk free space is a must. Please note that no collected data is stored as-is in the blockchain, but rather fingerprints of collected data are stored. Fingerprints all have the same size, so it is not the size of the data but the frequency of the data events that is critical to monitor.

It is highly recommended to use storage devices with a logical volume manager (like lvm), so that it will be easy to extend storage when needed. Monitoring and notifications about disk free space can be done using usual system tools, and of course using netdata.

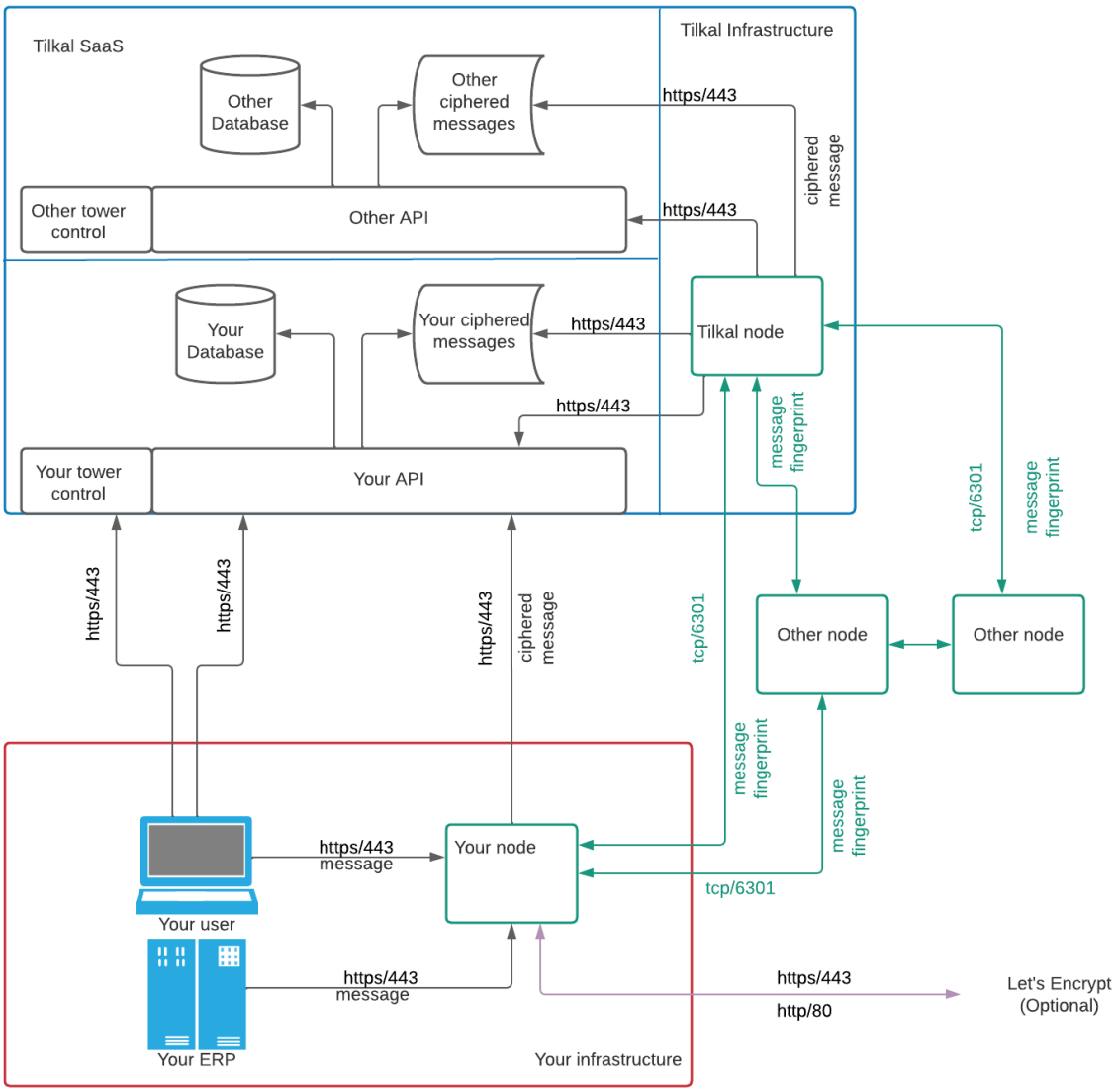
3.4. NODE INTERACTIONS / FLOWS MAPPING

Figure 1 shows a big picture schema to have a better comprehension of the interactions between your node, the rest of the traceability network actors and the Tilkal SaaS platform.

In this schema, the red rectangle represents your infrastructure with your own systems (ERP, employee's computers...), the blue rectangle represents the Sea2See Platform hosted by Tilkal and provided as a SaaS application.

The green part of the schema represents the blockchain network, and some of its nodes. Traceability actors will send data through their node, and the node will send the data to SaaS platform. This mechanism is used by SmartWater CLOUD aquaculture management software and MEDUSA water quality monitoring platform, to send aquaculture production data. In the same manner Page Up mobile applications are able to leverage the platform API to request and display existing data, and send new data by calling their node.

Figure 1



As represented:

- your node must be accessible from computers used by employees involved in the Tilkal Project (black arrow in red rectangle)
- your node must be accessible from you own systems to send data automatically, if planned in your project (black arrow in red rectangle)
- your node must be able to connect to at least the Tilkal node (green arrows)
- your node may be able to connect to other nodes, depending on your security rules (green arrows)
- your node exposes an HTTPS API, so that either you provide a TLS certificate associated with the node DNS name, or by default Let's Encrypt services are used (making it mandatory to open ports tcp 80 and 443, purple arrow)
- your node must be able to access your Tilkal SaaS application to retrieve the Tilkal SaaS web app and access your Tilkal SaaS API (black arrows from red rectangle to blue rectangle).

4. WHAT ARE THE VARIOUS OPTIONS FOR SHARING DATA

4.1. HTTP ENDPOINT

SECURITY

The entry point to send some raw data is the HTTP endpoint. Either called from your own internal infrastructure, or from an external service, this HTTP endpoint can be called either by using HTTP or HTTPS, depending on your security context.

SSL/TLS certificates can be generated automatically, free of charge, by leveraging the Let's Encrypt service. If this service is not compliant with your own internal policies, you can also provide a certificate of your own. Specifically, by default, to use the Let's Encrypt service the node must be accessible from the Let's Encrypt servers on ports 80. This could be not permitted by your company security rules, so that in this case you must provide your own TLS certificate.

To be able to expose the https service, the constraint are:

- The node must be reachable using a DNS name (TLS certificates cannot be linked to raw ip address)
- Either: ports tcp/80 and tcp/443 must be opened inbound publicly, to allow Let's Encrypt requests to reach your service to be secured (the node).
- Or: your DNS provider must expose an API that is supported by the Caddy server, and can be used to generate the certificate
- Or: you provide your own TLS certificate, then you can open only the needed ports from authorized services.

The HTTP endpoint can be configured to require an authentication when called, http basic and jwt token are supported right out of the box by the provided server.

EXAMPLES

When you send raw data to be notarized, two pieces of information must be provided beside the data: the channel through which the data is sent (the channel specify which other actors of the traceability network will receive the data), and the type of the data. Channels are configured using the Tilkal Suite application.
For a node, reachable using the DNS mynode.example.com, with the default configuration provided by Tilkal, using the wget command line tool :

```
wget --http-user=user --http-password=password --post-file data1.json
https://mynode.mydomain.com/event?type=application/x.gs1.epcis%2Bjson&channel=channel1
wget --http-user=user --http-password=password --post-file data2.json
https://mynode.mydomain.com/event?type=application/x.gs1.epcis%2Bjson&channel=channel2
```

The two calls use an http basic authentication with login=user, and password=password. Each call uses the same message type application/x.gs1.epcis+json.
URLs, authentication method, credentials are all configurable. If you decide to use the default http server (Caddy), then the configuration file is in <node installation path>/http/config/Caddyfile

Other authentication methods are available (e.g JWT token) and are explained in detail in the D3.4 deliverable of the Sea2See project.

4.2. SUPPORTED DATA FORMATS

Three raw data formats are supported:

- GS1/EPCIS over JSON
- GS1/EPCIS over XML
- CSV

4.3. CSV

Even if JSON or XML formats are easily produced and manipulated, it may be difficult for a data provider to build messages using these formats. This is why the Sea2See platform is also able to interpret CSV files to receive data.
Using CSV files, you will be able to share data about: product (trade item), locations (actors and sites), supply chain events.

PRODUCT

Products or assets can be created using a CSV file and specifying the application/x.global.asset+csv type.

Here are the main columns to provide, to create an asset:

Table 4

CSV column name	Description
Asset Name	Name of the asset (descriptionShort attribute in EPCIS model)
Unique Identifier GTIN or EAN(14)	GTIN 14 of the asset
Asset Reference	Reference of the asset (refProduct attribute in EPCIS model)
Type	Type of the asset (use Composition to specify the asset is a component)
Composition	Composition of the asset in one string
Weight	Weight of the asset (combined with Unit column)
Unit	Unit of the weight of the asset
Product line	productLine in EPCIS model
Category	Category of the asset
Colors	Colors of the asset
Number of sizes	sizeNumber in EPCIS model
Weight of the packaging	packagingWeight attribute in EPCIS model
Raw material of the packaging	packagingRawMaterial attribute in EPCIS model

For example, this CSV extract:

```
Asset Name,Unique Identifier GTIN or EAN(14),Asset Reference,Type,Composition,Weight,Unit,Product line,Category,Colors,Number of sizes,Weight of the packaging,Raw material of the packaging
Savon noir,4260305540034,34,Composition,,3,KGM,,,,,,
```

Will produce this object:

```
{
  "kind": "MasterData",
  "vocabulary": "urn:epcglobal:epcis:vtype:EPCClass",
  "id": "urn:epc:idpat:sgtin:426030554.0003.*",
  "attributes": {
    "urn:epcglobal:cbv:mda#descriptionShort": "Savon noir",
    "urn:tilkal:platform:mda#weight": "3KGM",
    "urn:tilkal:platform:mda#type": "Composition",
    "urn:tilkal:platform:mda#isComponent": true,
    "urn:tilkal:platform:mda#refProduct": "34"
  },
```

```
  "locale": "fr-FR"
}
```


LOCATIONS

Locations can be injected by submitting a CSV file having application/x.global.supplychain+csv as a type.

Here are the main columns to provide to create a correct CSV location file:

Table 5

CSV column name	Description
Unique Identifier GLN / ID	Location GLN
Supplier / Partner Name	Location name
Address	Location street address
Latitude	Latitude of the location
Longitude	Longitude of the location
Category	category attribute in EPCIS model
Country	countryCode attribute in EPICS model
City	city attribute in EPCIS model

For example, this CSV extract:


```
Supplier / Partner Name,Unique Identifier GLN / ID,Category,Address,Latitude,Longitude,Activity,Country,Region,City,Production Capacity,Use of green power,Wastewater treatment
Farm,0001.01,Farms,"Zone Artisanale, 50850 City",41.84094000000008,-2.9357999999999265,finishing,,,,,,
```


will produce this object:


```
{
  "kind": "MasterData",
  "vocabulary": "urn:epcglobal:epcis:vtype:Location",
  "id": "urn:tilkal:platform:id:sgln:loc:GLOBAL.0001.01",
  "attributes": {
    "urn:epcglobal:cbv:mda#name": "Farm",
    "urn:epcglobal:cbv:mda#latitude": 41.84094000000008,
    "urn:epcglobal:cbv:mda#longitude": -2.9357999999999265,
    "urn:epcglobal:cbv:mda#streetAddressOne": "Zone Artisanale, 50850 City",
    "urn:tilkal:platform:mda#category": "Farms",
    "urn:tilkal:platform:mda#activity": ["finishing"]
  },
  "locale": "fr-FR"
},
```

EVENTS

Events data can be shared using the Sea2See platform using a CSV file having the application/x.global.event+csv type. Here are the main columns to be provided:

Table 6

CSV column name	Description
Event	Type of the event (bizstep in EPCIS model)
Activity	activity attribute in EPCIS model
Event Date	Event date
Event Hour	Event time
AT/FROM Supplier / Partner Identifier	Source location id
TO Supplier / Partner Identifier	Destination location list
INPUT Asset GTIN	GTIN of input access in a transformation
INPUT Asset ID	Item reference of the input access in a transformation
Input Batch quantity	Quantity of input in a batch transformation
Input Unit	Unit of quantity of input batch in transformation
OUTPUT Asset GTIN	GTIN of output access in a transformation
Transaction Number	Id of a transaction associated to this event
Batch quantity	Quantity of output batch
Unit	Unit of quantity of output batch

For example:

```
Event,Activity,Event Date,Event Hour,AT/FROM Supplier / Partner Identifier,TO Supplier / Partner Identifier,INPUT Asset GTIN,INPUT Asset ID,Input Batch quantity,Input Unit,OUTPUT Asset GTIN,OUTPUT Asset ID,Transaction Number,Batch quantity,Unit
Shipping,Mining,23/01/2023,00:00:00,brandx.101931,brandx.nt21,,,,,brandx.0010008,1397961,,20000.000,TNE
```

Will produce :


```
{
  "kind": "Event",
  "eventType": "ObjectEvent",
  "eventTime": "2023-01-23T00:00:00.000Z",
  "eventTimeZoneOffset": "+00:00",
  "quantityList": [
    {
      "epcClass": "urn:tilkal:platform:id:lgtin:class:brandx.0010008.1397961",
      "quantity": 20000,
```

```
    "uom": "TNE"
   }
  ],
  "action": "OBSERVE",
  "bizStep": "urn:epcglobal:cbv:bizstep:shipping",
  "bizLocation": "urn:tilkal:platform:id:sgln:loc:brandx.101931",
  "sourceList": [
   {
    "type": "urn:epcglobal:cbv:sdt:owning_party",
    "source": "urn:tilkal:platform:id:sgln:loc:brandx.101931"
   }
  ],
  "destinationList": [
   {
    "type": "urn:epcglobal:cbv:sdt:owning_party",
    "destination": "urn:tilkal:platform:id:sgln:loc:brandx.nt21"
   }
  ],
  "extensions": {
   "comments": ["Mining"]
  }
}
```


ATTRIBUTES

For each kind of object (product, location and event), it is possible to add some columns to pass some attributes attached to the object.
A convention is to be used to specify the type and name of the attribute. Here are a few examples:

Table 7

CSV column name	Attribute name	Attribute type	Description
attribute:string:attr1	attr1	string	
attribute:attr2	attr2	string	Default type is string
attribute:boolean:attr3	attr3	boolean	true to pass true, anything else for false
attribute:number:attr5	attr5	number	Any parsable number, can use '.' or ','
attribute:integer:attr4	attr4	integer	
attribute:date:attr6	attr6	date	Date is specified in DD/MM/YYYY format

4.4. FORMS

Calling the node using an https request is not always the best method to use, especially when you are requesting data from a supplier or a non IT equipped company. This is why the Sea2See platform is also relying on the Tilkal Suite application to expose some web forms, to be filed by an operator. Tilkal Suite will make the needed https calls to the node once the web form is filled.

4.5. TILKAL OBJECT MODEL

The native format to send data to the Sea2See platform is built on GS1/EPCIS, in JSON messages. This is described in detail in the D3.4 deliverable of the Sea2See project.

CONCLUSION

In conclusion, the development and implementation of the blockchain-based Sea2See Traceability Platform has been pivotal in creating a robust network for data sharing through the use case demonstrators in the project, and in the coming months will demonstrate its functioning as a full traceability platform for the complete seafood value chain . This document outlines the comprehensive process of installing a Sea2See blockchain node and utilizing it to facilitate data exchange within a traceability project, enhancing the system's overall integrity and transparency.

By integrating well-established technologies such as Linux and Docker, the project ensures that deploying and operating a Sea2See blockchain node is straightforward and accessible. These widely adopted tools not only streamline the installation process but also empower stakeholders to become active participants in the traceability platform with minimal technical barriers.

The deployment of open-source blockchain nodes across various data partners allows authenticated users seamless access to the network. This facilitates the reliable transmission of data and enhances the traceability of data exchanges, thereby building trust among partners through verifiable data transactions.

Furthermore, the project includes the creation of a dedicated cloud platform for each data partner, tailored to store, analyze, and visualize both their data and shared data from others. This platform, configured to meet specific project needs, is hosted on a secure cloud infrastructure, providing a robust environment for data management and collaborative analysis.

Overall, Sea2See's innovative approach, combining blockchain technology with user-friendly tools and secure cloud solutions, marks a significant advancement in ensuring data integrity and enhancing cooperation within the network. This project not only simplifies the onboarding process for new partners but also provides them with the necessary tools to become active contributors to a transparent and accountable data ecosystem. The ease of deployment and operation via familiar technologies underscores the project's commitment to making participation in the Sea2See Traceability Platform as accessible and efficient as possible.