Co-financed by the European Union
Connecting Europe Facility

SCALES - Supply Chain Architecture Leading to Enhanced Services

Project number	INEA/CEF/ICT/A2018/1759084 2018-IT-IA-0053
Project acronym	SCALES
Project title	**Supply Chain Architecture Leading to Enhanced Services**
Starting date	1 March 2020
Ending date	30 May 2020
Programme	Connecting Europe Facility (CEF) CEF-TC-2018-2

Deliverable D3.3

Data integration services design and implementation (service HUB)

Related WP	Activity 3 - **Enhanced e-invoicing and e-procurement services**
Deliverable number	**3.3**
Due date	30/05/2020
Revision date	30/05/2020
Actual date	30/05/2020

Deliverable Info

Responsible organisation	Infocert
Editor	Infocert
Contributors	Infocert
Reviewers	Infocert
Abstract	This document describes the technical specification and high-level Architectural design of the envisaged solution, keeping into account the outcomes of Task 2.1 both for the state-of-the art analysis and for the requirements of the Italian eInvoicing infrastructure (B2B and B2G) and end-to-end eProcurement system.
Keywords	Architecture, Design
Acknowledgement	This work was partially supported by the European Commission (EC) through the Connecting Europe Facility (CEF) programme under project SCALES (grant agreement no. INEA/CEF/ICT/A2018/1759084 2018-IT-IA-0053)
Disclaimer	
Confidentiality	
Note	-

Change Log

Version	Date	Author	Organisation	Description
0.1	30/5/2020	EC	InfoCert	First Version

Table of contents

List of figures

1. Introduction

1.1. Purpose

The purpose of this document is to describe the technical specifications and high-level Architecture for the SCALES distributed ledger network and standard node and particularly for the Data integration Services.

This has two main use cases:

- minting of an Invoice NFT, by an intermediary/hub
- Accessing (and possibly updating) the Invoice status (and its metadata) by and external subject (VAS)

The intended audiences for this document are:

- Solution Architect
- Domain Manager/Sub-Domain Manager
- Project/Program Manager
- Stakeholders of the project

1.2. Scope

Scope of this document is the definition of the Data integration services of the Scales network.

This document shall specify:

- The general architecture of the network, taking in consideration the two main channel that are used to mint and to update the documents in the network
- The details of the uploading / minting process
- The protocols to be adopted for the federation of external services to the scales network

1.3. Definitions, Acronyms and Abbreviations

B2B	Business to Business
B2C	Business to Consumer/Citizen
B2G	Business to Government
BII	Business Interoperability Interfaces
DLT	Distributed Ledger Technology
C2G	Citizen to Government
CCTS	Core Component Technical Specification
CEF	Connecting Europe Facility
CEM	Certified Electronic Mail – Legal Mail (PEC Posta Elettronica Certificata in Italy)
CEN	European Committee for Standardisation
CII	Cross Industry electronic Invoice
CIUS	Core Invoice Usage Specification
DSI	Digital Service Infrastructures
EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport
EMSFEI	European Multi-Stakeholder Forum on eInvoicing
e-SENS	Electronic Simple European Networked Services
FatturaPA	Public administration electronic invoice framework (FatturaPubblica Amministrazione)
FT	Fungible Token
G2G	Government to Government
INEA	Innovation and Networks Executive Agency
JWT	JSON Web Token (cfr. RFC 7519)
NFT	Non-Fungible Token
OASIS	Organization for the Advancement of Structured Information Standards
OAuth2.0	Open Authorisation protocol version 2.0
OIDC	OpenID Connect – an identity and authorization federation protocol, specified at https://openid.net/specs/openid-connect-core-1_0-final.html
OP	OpenID Connect Provider
PEPPOL	Pan-European Public Procurement Online
PEPPOL-BIS	Pan-European Public Procurement Online Business Interoperability Specifications
RP	Relying Party
SDI	Electronic exchange system in Italy (Sistema Di Interscambio)
UBL	Universal Business Language
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
UNTDID	UN Trade Data Interchange Directory
URI	Uniform Resource Identifier
URL	Uniform Resource Location
URN	Uniform Resource Name
VAS	Value Added Service
XML	Extensible Mark-up Language

1.4. References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application.

[1]	EN 16931-1:2017 Electronic invoicing - Part 1: Semantic data model of the core elements of an electronic invoice
[2]	CEN/TS 16931-2:2017 Electronic invoicing - Part 2: List of syntaxes that comply with EN 16931-1
[3]	CEN/TS 16931-3-1:2017 Electronic invoicing - Part 3 - 1: Syntax bindings of the core elements of an electronic invoice - Syntax binding methodology
[4]	CEN/TS 16931-3-2:2017 Electronic invoicing - Part 3 - 2: Syntax bindings of the core elements of an electronic invoice - Binding to ISO/IEC 19845 (UBL 2.1)
[5]	CEN/TS 16931-3-3:2017 Electronic invoicing - Part 3 - 3: Syntax bindings of the core elements of an electronic invoice - Binding to UN/CEFACT XML
[6]	CEN/TS 16931-3-4:2017 Electronic invoicing - Part 3 - 4: Syntax bindings of the core elements of an electronic invoice - Binding to ISO/IEC 9735 (UN/EDIFACT)
[7]	ISO 3166 1, Codes for the representation of names of countries and their subdivisions — Part 1: Country codes
[8]	ISO 4217, Codes for the representation of currencies
[9]	ISO 639 2, Codes for the representation of names of languages
[10]	ISO 8601, Data elements and interchange formats — Information interchange — Representation of dates and times
[11]	ISO 15000-5, Electronic Business Extensible Markup Language (ebXML) — Part 5: Core Components Specification (CCS)
[12]	ISO 6523, Information technology — Structure for the identification of organizations and organization parts
[13]	ISO/IEC 19845, Information technology -- Universal business language version 2.1 (UBL v2.1)
[14]	OpenID Connect Specification – Core specifications, version 1.0

2. Scales DLT Network

2.1. General architecture approach

The general approach in designing the Scales architecture provides that the current centralized service, typically based on a central authority shifts towards a decentralized network where:

- The object is created on a specific node of the DLT network so that it shall be visible only to that node and all the counterparty nodes involved in a workflow
- The Workflow is distributed among different nodes, where each node participates via signed transaction, once it has completed its own task
- The only nodes that holds the data is the one where the data has been uploaded to. All the other nodes, involved in the workflow, participate by getting the data form the source node, doing some work on it and signing back their own transaction in order to let the workflow progress.
- Some nodes are involved on a need-to-know basis. For instance, with reference to Figure 1, the blue colored DLT nodes are involved on a need-to-know basis: one is the source node, where the data is uploaded, the other is asked to participate to the workflow.
- Some nodes are always involved. For instance, with reference to Figure 1, the green node, Competent Authority and the red node, Notary, are involved in every workflow, as they are respectively asked to:
    - Check the correctness of data (Competent Authority)
    - Sign the last step of the workflow, in order to avoid double spending of it (i.e. repeating the workflow on the same instance of data)

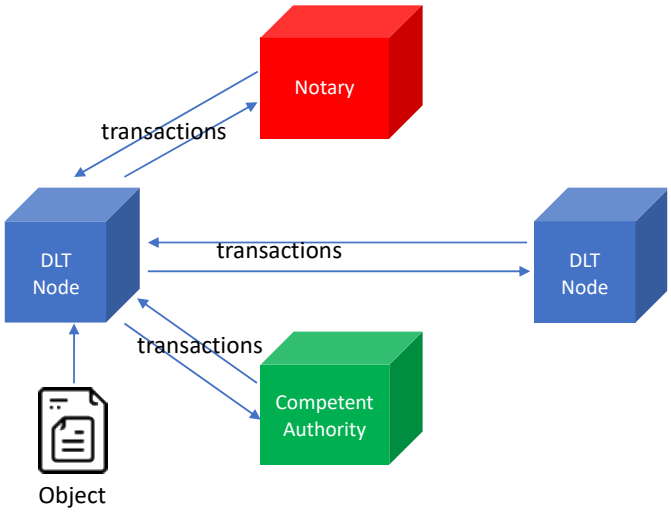

Figure 1 – Architecture Principles

This approach shall be applied to all user cases supported by SCALES project and shall comply with RegRep Standard Approach.

2.2. eInvoicing Architecture views

In this paragraph we focus on the logical architectural view of the Scales Decentralized Network, specializing the general architecture described at paragraph 2 in the context of eInvoicing.

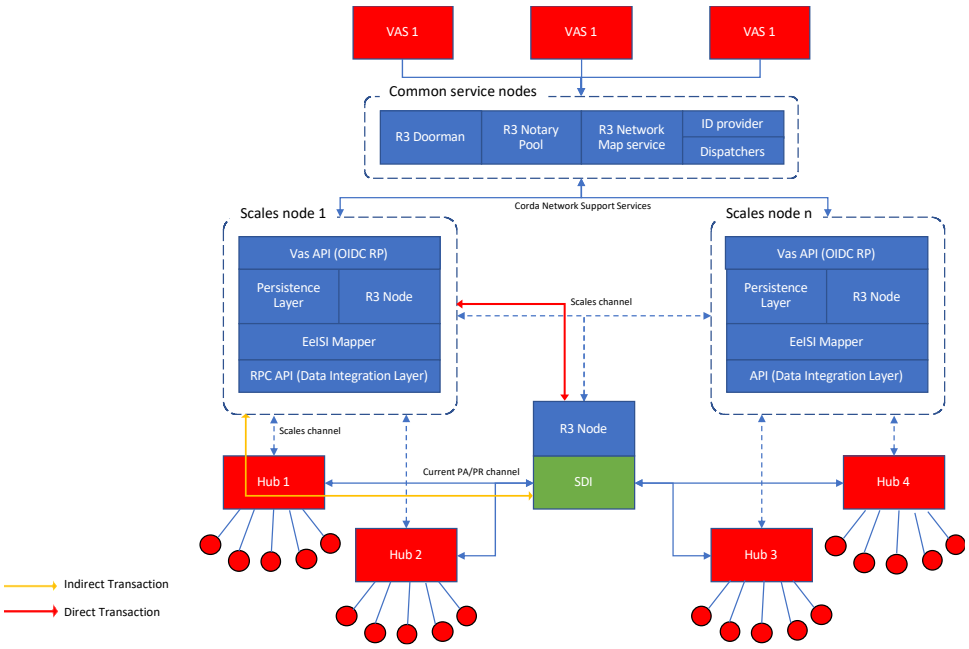

Figure 2 - eInvoicing logical architecture view

Limiting the scope of the architecture description to the components that are relevant to this document, let's describe in detail the Scales corda nodes and how this integrates with the external systems.

1.1.1 Scales node

The Scales Node is the main component of the network and it is in charge to:

- offer a data integration layer allowing HUBS to upload (and basically create digital object (i.e. invoices, order, etc) in the DLT.
- Offer a VAS partner set of APIS in order to allow external parties to fetch and possibly take control of object once they have completed their primary life cycle (for instance, once the invoice has been validated, delivered and accepted by the counterparty, VAS partner may have access to it and decide to propose a service to the involved parties).
- Offer support services for the mapping between formats
- Implement the persistence layer of all the data and metadata of objects
- Implement the Corda R3 flows

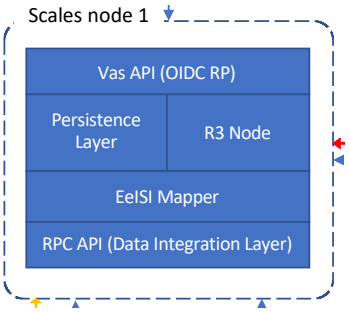

Figure 3 - Scales Node

A Scales full node is composed of the following layers:

- R3 Corda node: this is the DLT component in charge of executing corda Flows and manage uniquely represented digital asset in the system (i.e. Invoices, orders, etc.). Transactions carried out by the corda flow works on metadata of the object without moving the original object around the network. Once the object is created on the Nods, it resides in its view of the global registry and its persistence layer permanently.
- Persistence layer: this is a storage services, implemented through the cooperation of a Database and a Network Storage, front-ended by proper APIs, in order to let an external component to access to the original object and its metadata.
- EeISI mapper is the layer in charge of performing transformation of the objects uploaded to a Scales node. Particularly, an object is uploaded in a specific – original – format, that will be persisted in the nodes. EeISI mapper shall be engaged when a transformation of the object is required, but this shall not substitute the original of the stored data.

1.1.2 Common Services

Scales Common Services are components that are needed to support distributed flows among the network. They are in charge of:

- Provide x.509 Pki credentials to Scales node, i.e. a legal identity to organization running scales nodes
- Map IPs onto Scales node names, in order to let the Corda R3 flows happen.
- Notarize transaction in order to avoid double spending of status changes
- Provide Federated identity service to External parties (vas partner, other countries hubs to integrate with the network)

Provide dispatching service in order to address an external party to the proper scales node when it needs to get or change an object and its status.

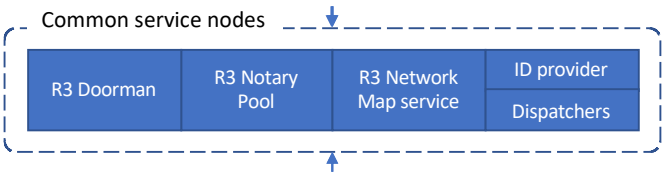

Figure 4 - Scales common services

The Identity provider component is an OpenID Connect Provider in charge of authenticating both the data owners and the third parties that requires access to their metadata.

OpenID Connect is the proper protocol to allow complex use cases where, for instance, the owner of some data wants to grant access to a client, and still both need to pass through a rigorous authentication process in order to set up the delegation.

OpenID protocol is flexible as it is json / rest based and it has all the security features already available in SAML.

The Dispatcher component is a DLT common component in charge of persisting the mapping between an end entity and the scales node that hold its data.

Once the third party (for instance the VAS partner) has been authenticated to the system, though the dispatcher service it shall be able to access and, possibly change status, of all the data for which it received a grant from the respective owners.

The dispatcher maps the following relation:

data owner (vat number or other unique identifier) / intermediary-hub / scales node id

The dispatcher shall notify, via callback, the VAS that were previously granted access to and end entity data (invoices, order, etc) in order to let them know that there are new data available.

The callback payload shall include the uri where to get the data.

The access to that uri shall be protected via federated OpenID Connect authorization code flow.

3. Data Integration Layer Component

3.1. Data integration layer features

Data integration layer shall be based on an opensource component allowing ETL (Extract, transform and Load) operation on data.

Further, the same component shall allow hub to upload the data using the same protocol as:

- Webservice (Restful API)
- FTP

3.2. Data Integration Layer Logical Flow (B2B)

In this paragraph we specify the technical details of the Integration process between the Intermediary (Hub) and the scales node.

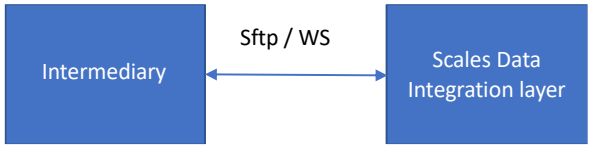

Figure 5 - Integration Channel between Hub/Intermediary and Scales Node

As shown in Figure 2, each hub/intermediary shall refer to a Scales node.

Intermediary shall communicate with its Scales Node of reference via two different protocols:

- SFTP
- Web Services

3.2.1. SFTP Integration (indirect mode)

Set up: The Intermediary and its Scales node shall exchange x.509 v3 certificate to set up the SFTP Channel. The procedure shall be the same of the SFTP channel set up between and intermediary and the SDI.

On the scales node, for each hub, there will be 2 folders, namely:

- Input folder (<hub-id>_input)
- Output folder (<hub-id>_output).

Hubs shall write in the input folder and read form the output folder.

Each hub shall have a different SFTP account, so that it shall be able to access only to its folders.

Data format: intermediary shall upload invoices on the basis of the following format:

- 1 zip file for each invoice
- The zip file shall include:[1]

[1] The aggregation of invoices in a single lot shall not be supported.

- The invoice
- The SDI notification

Notification: we have to distinguish the case in which the Active subject (or supplier) is uploading the invoice form the case where the passive subject (or buyer) is uploading. In fact, the two subjects have a different notification form SDI.

Active/Supplier notification: the notification includes

- Invoice HASH
- SDI identifier
- File name of the invoice
- Positive Uniqueness check

Passive/Buyer notification: the notification includes

- Invoice HASH
- SDI identifier
- File name of the invoice
- Positive Uniqueness check

Logic Flow of Scales Node Data Integration Layer (DIL): when receiving an invoice (a new zip file) form and intermediary, the flow of checks and operation that the DIL must carry out are the following:

1. Check if the Zip file is complete (it includes both einvoice and SDI notification)
2. Check that Invoice file hash match with SDI notification included hash string
3. Check that Invoice hasn't been already submitted. At technical design level it may be convenient for the Scales node to keep record of all the hashes of the submitted invoices, in order to quickly execute a query to check invoice uniqueness.
4. In case of Signed Invoice, in format P7M, extract original xml from envelope is necessary.
5. Check SDI notification signature verification against trusted root / cert
6. Extract and prepare token data and original data to be written on the ledger:
    a. Check invoice format type and version:
        i. Check invoice header (root element)
        ii. (optional) apply schema to validate invoice
        iii. Save invoice original format and version in Token metadata
    b. Apply extractor in order to:
        i. Map invoice data on token metadata
        ii. Calculate extended metadata (for instance: total amount, taxes, etc) (n.b. Token metadata are expressed in business terms)
7. Start Minting flow:
    a. Mint invoice (i.e. create the NFT invoice token on the ledger)
    b. Notarize token metadata: (invoice hash, owner)

8. Create SFTP response Flow:
    a. For each received ZIP File, create an output file that shall be named:
        i. OK-<original-filename>.out, in case of success. The file can be empty
        ii. KO-<original-filename>.out in case of error. It's a text file including the error message.

3.2.2. Web Service Integration (indirect mode)

APIS provided by Scales Nodes are described in the following schema

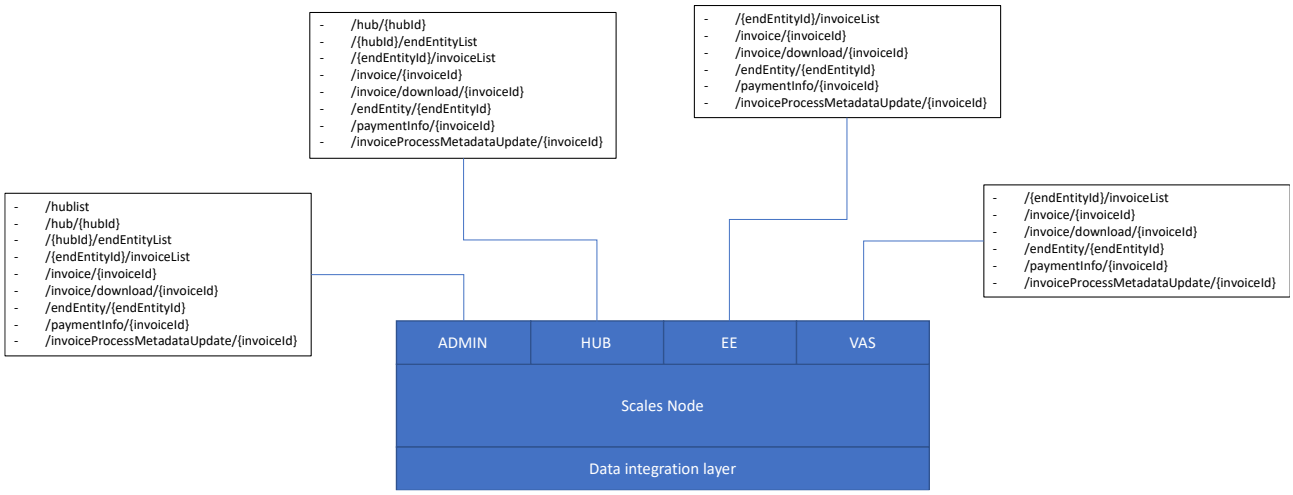

Figure 6 - Api for HUB and VAS

3.3. Data models

3.3.1. eInvoice

Data-bound metadata

- InvoiceFormat
- InvoiceNumber
- InvoiceIssueDate
- InvoiceTypeCode
- InvoiceCurrencyCode
- ProjectReference
- PurchaseOrderReference
- TenderOrLotReference
- SellerVATIdentifier
- SellerTaxRegistrationIdentifier
- BuyerVATIdentifier
- BuyerTaxRegistrationIdentifier

- BuyerTaxRegistrationIdentifierSchemeIdentifier
- BuyerElectronicAddress
- BuyerElectronicAddress SchemeIdentifier
- PaymentDueDate
- InvoiceTotalAmountWithVAT
- AmountDueForPayment

Process-bound metadata:

- CompetentAuthorityUniqueIdentifier
- CurrentState
- FileName
- MessageId
- DateTimeReceipt
- DateTimeDelivery
- NotificationNotes
- NotificationSignature
- InvoiceHash
- InvoiceOwner
- InvoiceSignature
- ApprovedSubject

3.3.2. eOrder

eOrder Data Model:

- + sellerVatNumber (core) or sellerTin (core)
- orderDate (core)
- orderID (core)
- orderIssuerEndpoint (core)
- buyerVatNumber (core)
- buyerTin (core)
- senderEndpoint (core)
- receiverEndpoint (core)
- buyerEndpoint (core)
- invoiceTypeCode (core)
- IdT (NSO)

- currentState (NSO)

4. Scales federated APIs

This chapter describes the strategy through which Scales network allows third party to register and access private data in order to fetch them and decide to perform transaction, offering value added services.

OpenID Protocol is the technology that unlocks such a critical use case, as it adds to OAuth2.0 protocol all the Identity and signature elements that are fundamental for a privacy preserving, secure and auditable system.

The process may be split in two main use case:

- Third party registration,
- End entity onboarding.

Also, it describes the proposed model regarding how a third party (particularly a VAS Partner) transact to the network in order to take ownership of Scales objects (invoice, etc.)

4.1. Third party registration

The third party enroll its client though the OnBoarding common service component.

Through this process it obtains two critical information that are:

- Client_ID
- Client_secret

The third party shall securely persist this information in its application.

4.2. VAS-End entity authorization

Each End Entity shall explicitly authorize the third party to access its own data on Scales.

The OpenID proposed flow is “authorization code flow” with scope “openid”.

The id_token, resulting from the flow shall:

- refer to the client_id as audience
- include all the relevant primary keys to allows the third party to access the scales node.

Following, a sample VAS Authorisation Request:

1. A GET method

https://dispatcher.scales.eu/authorize?

client_id=<vas_client_id>

&redirect_uri=<vas_callback_uri>

&scope=<vas_type_scope>

&response_type=code

&state=<transaction_id>

You get a webpage where the End Entity representative logs in and get a <authorization_code> back

Then follow a Token Request Method (POST)

2. Post https://dispatcher.scales.eu/oauth/token
   grant_type=authorization_code
   &client_id=<vas_client_id>
   &client_secret=<vas_application_client_secret>
   &redirect_url=<vas_callback_uri>
   &code=<autorisation_code>

The third party application receive a response consisting of a json like the following:

{
"access_token": "U8pN88yPtuJxEIQkTjQ3F7M4LGN_yG1z",
"id_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJjbGllbnRJRCI6ImtieXVGRGlkTExtMjgwTEl3VkZpYXpPcWpPM3R5OEtIIiwiY3JlYXRlZF9hdCI6IjIwMTktMTEtMzBUMTc6MDk6MzQuNDEwWiIsImVtYWlsIjoiZWdpZGlvLmNhc2F0aUBnbWFpbC5jb20iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwiZmFtaWx5X25hbWUiOiJDYXNhdGkiLCJnaXZlbl9uYW1lIjoiRWdpZGlvIiwiaWRlbnRpdGllcyI6W3sicHJvdmlkZXIiOiJnb29nbGUtb2F1dGgyIiwidXNlcl9pZCI6IjExMTkwNzMwNTE4MDc3NTE2MzA4MiIsImNvbm5lY3Rpb24iOiJnb29nbGUtb2F1dGgyIiwiaXNTb2NpYWwiOnRydWV9XSwibG9jYWxlIjoiZW4iLCJuYW1lIjoiRWdpZGlvIENhc2F0aSIsIm5pY2tuYW1lIjoiZWdpZGlvLmNhc2F0aSIsInBpY3R1cmUiOiJodHRwczovL2xoMy5nb29nbGV1c2VyY29udGVudC5jb20vYS0vQUF1RTdtQ3hJN19taG5zMGMxTEdNVGNHRFEyTXRIQndkY0NZUFlpSnluaE9kQSIsInVwZGF0ZWRfYXQiOiIyMDE5LTEyLTAyVDA1OjAwOjQ3LjIyNloiLCJ1c2VyX2lkIjoiZ29vZ2xlLW9hdXRoMnwxMTE5MDczMDUxODA3NzUxNjMwODIiLCJpc3MiOiJodHRwczovL3NhbXBsZXMuYXV0aDAuY29tLyIsInN1YiI6Imdvb2dsZS1vYXV0aDJ8MTExOTA3MzA1MTgwNzc1MTYzMDgyIiwiYXVkIjoia2J5dUZEaWRMTG0yODBMSXdWRmlhek9xak8zdHk4S0giLCJpYXQiOjE1NzUyNjI4NTcsImV4cCI6MTU3NTI5ODg1N30.uxksoRhdYaTSN-K5tAUOklOFZaChfaqxAbcyFck8nyo",
"scope": "openid profile ",
"expires_in": 86400,
"token_type": "Bearer"
}

4.3. Vas API Flow

The following schema describes the Vas API flow, based on OIDC federation.

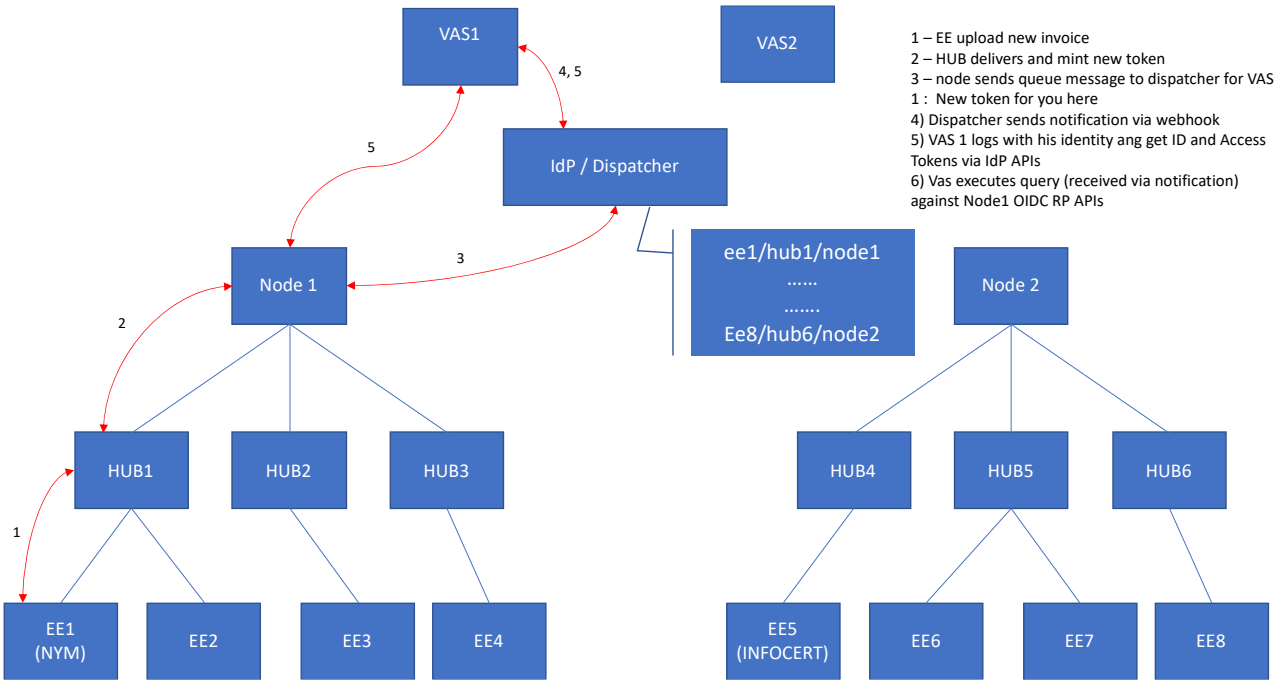

Figure 7 - Vas API Flow

4.4. Reason to adopt OpenID Connect standard protocol

There are a number of reason OpenID Connect protocol could benefit the Scales Architecture; these can be summarized as the following:

- Popularity: there is a wide amount of opensource components (libraries, sdk, plug in) that supports the OpenID Connect Protocol
- Interoperability: using OpenID Connect allows heterogenous systems to easily federate together
- Modern: is a lightweight protocol, it is also cloud / mobile friendly.
- Avoid Lock-in it can be implemented indifferently through a large number of different platforms
- Consolidate: Widely used by tech giants (Google, Facebook, etc.)
- Security proofed

4.5. 3rd Party Transaction Model

In this paragraph we describe the high-level sequence of transaction involving a 3rd party that interact with the network.

As a premise:

- the object (invoice, order, etc) has completed its normative life cycle.
- The 3rd party already registered its client_id and client_secret to Scales common service (particularly to the ID provider)
- The 3rd party already determined on which object he intends to operate

Transaction sequence:

1. The 3rdParty starts a flow: it signs a “bid” offer for a specific object (for instance, an invoice), requesting a specific hub to approve

2. Hub is notified about the signature request. Hub notify end entity and get (off-chain) its approval
3. Hub approves bid offer on behalf of end entity
4. 3rd party is notified
5. 3rd party confirm approval and becomes the new “owner” of the object
6. 3rd party ask notary to countersign the flow
7. Notary countersigns the flow, completing the process

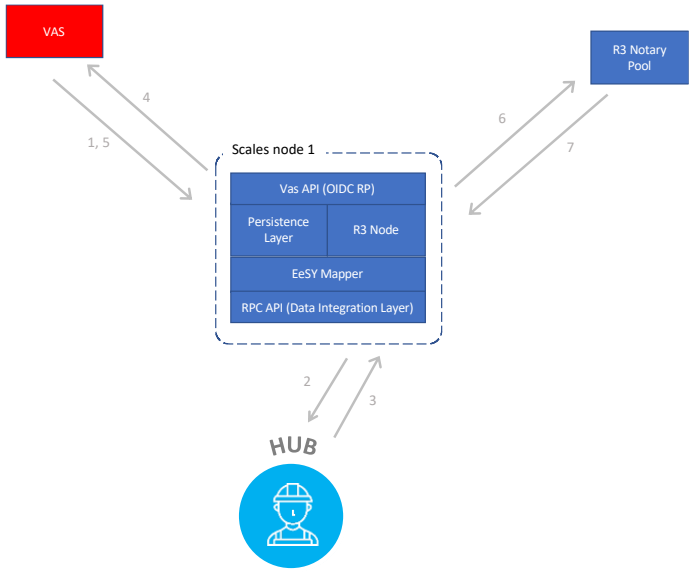

Figure 8 – 3rd Party Transaction Model