



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 Sep 2019
Ending date	30 Nov 2019
Programme	Connecting Europe Facility (CEF) CEF-TC-2018-2

Deliverable D2.2

Solution design

Related WP	Activity 2 - New service-oriented architecture
Deliverable number	2.2
Due date	30/11/2019
Revision date	30/11/2019
Actual date	30/11/2019

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	The sole responsibility of this publication lies with the author(s). The European Union is not responsible for any use that may be made of the information contained therein.
Confidentiality	The information in this document is confidential and restricted only to the members of the SCALES consortium (including the Commission Services).
Note	-

Change Log

Version	Date	Author	Organisation	Description
0.1	29/11/2019	EC	InfoCert	First draft
0.2	09/01/220	EC	Infocert	<ul style="list-style-type: none"> • Updated figures 9 and 10 • Updated figures 12 • Minor typos fix
0.3	3/2/2020	EC	Infocert	<ul style="list-style-type: none"> • Added cap. 4.2 and sub par.
0.4	29/3/2020	EC	Infocert	<ul style="list-style-type: none"> • Refined elvoice Data Model (4.3) • Vas API Flows (5.4) • Revised Initial Deployment Scenario (6.1)
0.5	1/4/2020	EC	Infocert	<ul style="list-style-type: none"> • Revision of cap 4.2
0.6	16/6/2020	EC	Infocert	<ul style="list-style-type: none"> • Introduction of new Cap 7 – generalisation and future development

Table of contents

Table of contents	2
List of figures	3
1. Introduction	4
1.1. Purpose	4
1.2. Scope.....	5
1.3. Overview	6
1.4. Definitions, Acronyms and Abbreviations	7
1.5. References.....	8
2. Scales DLT Network	9
2.1. Constraints and Assumptions	9
2.2. Actors.....	10
2.3. Functional Requirements	11
2.4. Non-functional requirements	12
2.5. General architecture approach	13
2.6. eInvoicing Architecture views.....	14
3. Scales Nodes.....	16
3.1. Corda business Network features	16
3.2. Scales Node Description	16
3.3. Scales Common Services.....	16
3.3.1. Doorman	16
3.3.2. Network Map.....	17
3.3.3. Notary Pool	17
3.3.4. Identity provider	18
3.3.5. Dispatcher	18
3.4. OnBoarding Service.....	18
3.4.1. Onboarding of Intermediary-HUB.....	18
3.4.2. Onboarding of End Entities.....	18
3.4.3. Onboarding of VAS.....	19
3.4.4. EndEntity – VAS agreement	19
3.5. “as is” versus “to be” flows	19
3.5.1. “as is” flow for B2B Invoicing.....	19
3.5.2. “to be” flow for B2B invoicing – Direct Flow.....	20
3.5.3. “to be” flow for B2B invoicing – Indirect Flow	21
4. Data Integration Layer Component	22
4.1. Data integration layer features.....	22

4.2.	Data Integration Layer Logical Flow (B2B)	22
4.2.1.	SFTP Integration (indirect mode)	22
4.2.2.	Web Service Integration (indirect mode).....	24
4.3.	Data models	24
4.3.1.	eInvoice.....	24
4.3.2.	eOrder.....	25
5.	Scales federated APIs.....	26
5.1.	Third party registration	26
5.2.	VAS-End entity authorization	26
5.3.	Open API Specification	27
5.4.	Vas API Flow	28
5.5.	Reason to adopt OpenID Connect standard protocol.....	28
5.6.	3rd Party Transaction Model	28
6.	Scales Network initial deployment	30
6.1.	Initial Deployment Scenario	30
6.2.	Consortium to run Common Services	30
7.	Generalisation and future development.....	31
7.1.	Abstract Architecture Pattern.....	31
7.2.	Self-Sovereign Identity	31
7.3.	Smart legal Contract.....	32
7.4.	eDelivery (ZCM)	32

List of figures

Figure 1 - basic parties and roles in a purchase-to-pay process	10
Figure 2 – Architecture Principles	13
Figure 3 - eInvoicing logical architecture view.....	14
Figure 4 – scales node software stack.....	16
Figure 5 - Commons Service nodes: Doorman	16
Figure 6 – Commons Service nodes: Newtork Map	17
Figure 7 - Commons Service nodes: Notary poll	17
Figure 8 - "as is" B2B invoicing flow	20
Figure 9 – “to be” flow for B2B invoicing – Direct Flow	20
Figure 10 - “to be” flow for B2B invoicing – Indirect Flow	21
Figure 11 - Integration Channel between Hub/Intermediary and Scales Node	22
Figure 12 - Vas API Flow	28
Figure 13 – 3rd Party Transaction Model.....	29
Figure 14 - initial Deployment Scenario	30
Figure 15 – abstract architectural design pattern.....	31

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.

This document is jointly written by the Domain Architect, Solution Architect and Solution Manager, each working on his view on the same matter which is functionally and technically driven.

In the end, all the three parties have to agree on the content.

This document forms the base of subsequent phases of the project like Implementation, Platform Validation and Valorization.

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 Architecture and Design of the Scales network.

This document shall specify:

- The general architecture of the network, taking in consideration at least one of the main use case as a driving case (for instance, B2B invoicing)
- The details of the software layer included in the Scales node and in the Scales Common Services
- The protocols to be adopted for the federation of external services to the scales network

Further on, the document aims to describe what could be a possible initial deployment of the network.

1.3. Overview

This document is organized as follows:

Chapter 2 concerns the Scales DLT-Based Network description. It describes:

- Requirements and Constraints of the network and its participant nodes,
- Actors of the system,
- Logical Architecture,

Chapter 3 concerns the Scales Nodes description, in particular:

- Corda business Network features,
- Scales Node,
- Scales Common Services,
- AS IS vs TO BE Flow (invoice B2B).

Chapter 4 concerns the Data integration layer, through which hubs can integrate to Scales in order to upload and make available to the network their asset (eInvoicing and eProcurement asset and metadata). It describes:

- Data Integration Layer feature,
- Invoice and eProcurement data model

Chapter 5 concerns the authentication and authorization framework, allowing a VAS Partner to subscribe to scales in order to provide its services to hubs and their final users. It describes:

- OpenID federation of third parties with Scales network
- VAS transaction model

Chapter 6 provide the initial deployment scenario of the Scales network. It describes:

- Consortium to run common services, and onboarding procedure
- First Deployment scenario
- Possible further developments

1.4. 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

1.5. 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. Constraints and Assumptions

Here are detailed some considerations regarding the major technical and functional constraints that have some impact on the software design.

Constraint	Description
Architectural Principles	<p>Infrastructure independent:</p> <ul style="list-style-type: none"> - Possibility to deploy the system in on-premise & cloud mode, to respond to heterogeneous technical and compliance requirements. <p>Privacy:</p> <ul style="list-style-type: none"> - Being SCALES a solution based on DLT, privacy of information and correct sharing of data among different network nodes types is a critical constraint. <p>Instant Finality:</p> <ul style="list-style-type: none"> - Since SCALES aims at two main objectives, i.e. <ul style="list-style-type: none"> o build a more efficient and resilient architecture in order to manage large volumes of data o supporting the implementation of enhanced invoicing and eProcurement services Instant finality is necessary. <p>Identity of participant nodes:</p> <ul style="list-style-type: none"> - in order to comply with privacy regulation and establish the correct accountability of transactions on data, the Scales node shall be run by legally identified parties. <p>Software Licensing:</p> <ul style="list-style-type: none"> - Build a software stack based on Open Source components, leaving to the specific instance provider the freedom to upgrade community / open source version to commercial edition, preserving backward compatibility of software.
Timeline	First draft of solution design: 30/11/2019
Existing Landscape	<p>The project involves the development of a DLT-based Network to allowing decentralization of critical service for the invoicing and eProcurement.</p> <p>The present document, also on the base of task 2.1 results, identified Corda-R3 DLT technology (community version 4.3) as the best opportunity to design the Network.</p> <p>This decision shall establish the base of the Scales Node software stack, allowing the design of the synergic integration of new component and inherited ones, such as the EeISi mapper.</p>
Geographical constraints	SCALES DLT network flows are based on SDI elnvoicing and NSO specification according to national SDI specifications

Risk Willingness	Low: <ul style="list-style-type: none"> - SCALES Project aims at decentralizing the architecture of systems and process which are critical from both confidentiality/privacy and availability perspectives. - Authentication and Authorization of VAS to read, retain and acquire business information must be accurately handled. -
Complexity	High: <ul style="list-style-type: none"> - The software has to be designed infrastructure independent. - Current legislation and regulation compliance in both Europe and Italy requires complex implementation rules. - The software has to be entirely designed

2.2. Actors

This paragraph focuses on the actors involved in the Scales network.

Scales Actor may be defined starting from the model showed in [1], where, at par. 5.1 the typical set of parties and roles involved in a basic purchase-to-pay process are described (see figure below).

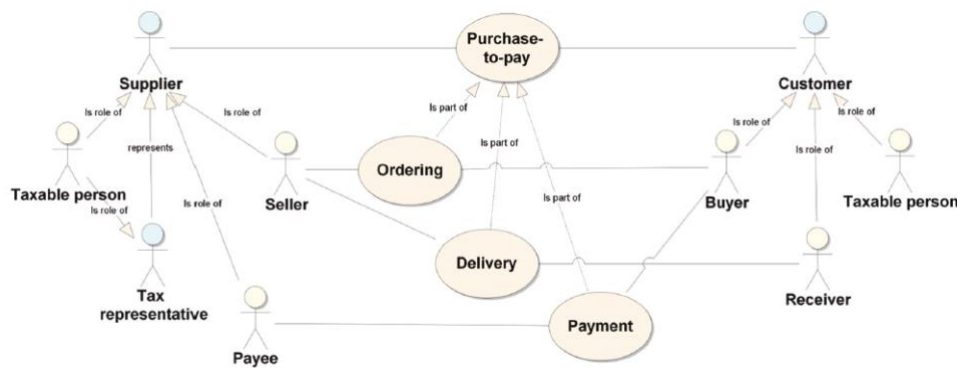


Figure 1 - basic parties and roles in a purchase-to-pay process

We define HUB as collectors of data from end entities parties such as supplier, buyer, defined in the above schema.

HUB are the collectors of data that interfaces scales nodes in order to upload the relevant information they are already exchanging with the centralized components in the current processes, such as SDI and NSO.

HUB are intermediaries that relieve the end entity (such as a company who buys or sells goods) from the burden of managing integration with SDI and NSO.

2.3. Functional Requirements

The Scales DLT network functional requirements offered to the stakeholder and users are:

<i>Functional REQ ID</i>	<i>Functional REQ Name</i>	<i>Functional REQ Description</i>
FREQ1	Smart Contract	<p>Support Smart Contract - it must be possible to develop smart contract representing digital assets in the form of Fungible and NFT. For instance:</p> <ul style="list-style-type: none"> - a fungible token (in the style of an ERC20 Ethereum token) to regulate the microtransactions happening in the platform - a non-fungible token (in the style of ERC721) to represent digital assets like orders, invoices and other documents whose value may be negotiated in the context of VA Services. - Digital Asset Tokenization, in order to allow a number of different subjects to concur in the purchase and sell of a specific number of subunit of a contract (order, invoice, etc) for financial purposes such as invoice Factoring, liquidity polling, etc
FREQ2	Privacy	Support Privacy - only the transaction involved parties may be able to disclose (i.e. decrypt) the transaction content even in the case of multi-recipient transactions (i.e. one sender to many recipients).
FREQ3	Trusted Identity	Support identity and trust - the platform must allow the identification, credential issuing and credential verification for both full node accounts and externally owned accounts
FREQ4	Lighthouse SDI/NSO	<p>Allow government components like current SDI and NSO to participate in the process without the need to maintain its own storage, or to duplicate data (invoices, order, etc.) for the sole purpose of carry on their tasks.</p> <p>Once an information is uploaded by a HUB to a Scales node, the Scales node become the single source of truth concerning the status of the information, its metadata and the flow that applies to it.</p>
FREQ5	VAS Authorization	Onboarding process for the end entity must allow it to authorize VAS partner to access, via VAS Partner API, to its information.
FREQ6	Data Model	Scales supported data model and validation schemas to represents registered object (invoice, orders, etc.) must be maintained by the Scales consortium, voted and committed to the registry as a standard reference for both Hub and external actors (vas, partner, etc.)
FREQ7	Data Model Extension	Scales supported data model and validation schemas shall be extended in order to meet the needs to enrich the registered object life cycle.
FREQ8	Flows	Every use case currently implemented in a centralized paradigm (such as, fatturazione PA, fatturazione B2B) shall be transformed in a distributed flow, based on the cooperation of different Scales nodes, in order to carry out the same result but without the need of a central orchestration.
FREQ9	Data segregation	Each Scales node shall be the single source of truth and sole repository for the data uploaded by its subscriber Hubs. External Actors, providing additional services shall access those data if and only if authorized by its owner (i.e. the hub customer) and via a dispatching component in charge of mapping the vat number of the owner with a specific Scales node.

2.4. Non-functional requirements

The Scales DLT network functional requirements offered to the stakeholder and users are:

<i>Non-Functional REQ ID</i>	<i>Non-Functional REQ Name</i>	<i>Non-Functional REQ Description</i>
NFREQ1	Scalability	Support performance scalability - consensus algorithm must enable a scalable number of transactions. Proof of Stake and Proof of Authority, as specialized BFT consensus are recommended
NFREQ2	Open Source	Scales node full software stack must be based on Open Source components.
NFREQ3	Programming language support	platform must support - at least - one popular language for the smart contract development
NFREQ4	Transaction Instant Finality	in case of a real blockchain data structure (instead of a distributed db) platform must support instant finality of block validation. Probabilistic finality isn't compatible with the platform business scenario
NFREQ5	Optional vendor Support	Enterprise support - platform must be supported by or more official certified company. Platform must provide a certification path.
NFREQ6	Transition	The solution must be designed in order to support both the legacy and new processes. This means that, while a number of end entities and respective hubs shall continue to work with centralized components under the current specifications, some other shall start uploading their data also on their reference scales nodes, allowing a parallel regime.
NFREQ7	Opt-in	While pursuing the Grant Agreement objective, the designed architecture must grant the centralized component the choice to opt-in in the scales network at any time, without this to block in any way the project and the piloting of the solution.

2.5. 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 from 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 2, 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 2, the green node, Competent Authority and the red node, Notary, are involved in every workflow, as they are respectively asked to:
 - o Check the correctness of data (Competent Authority)
 - o 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)

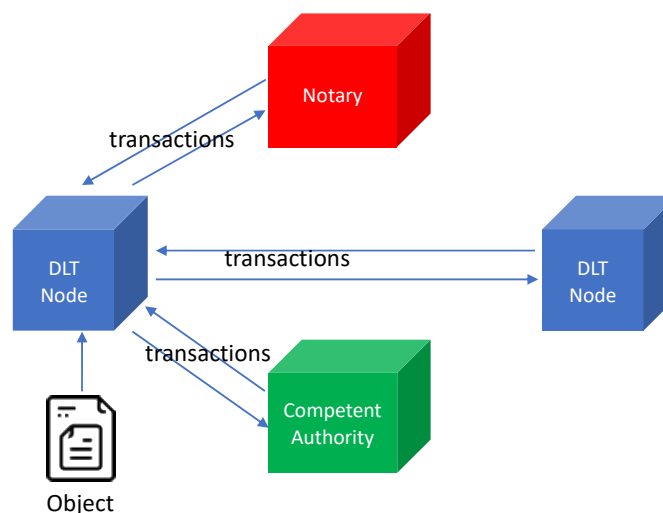


Figure 2 – Architecture Principles

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

2.6. 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 2.5 in the context of eInvoicing.

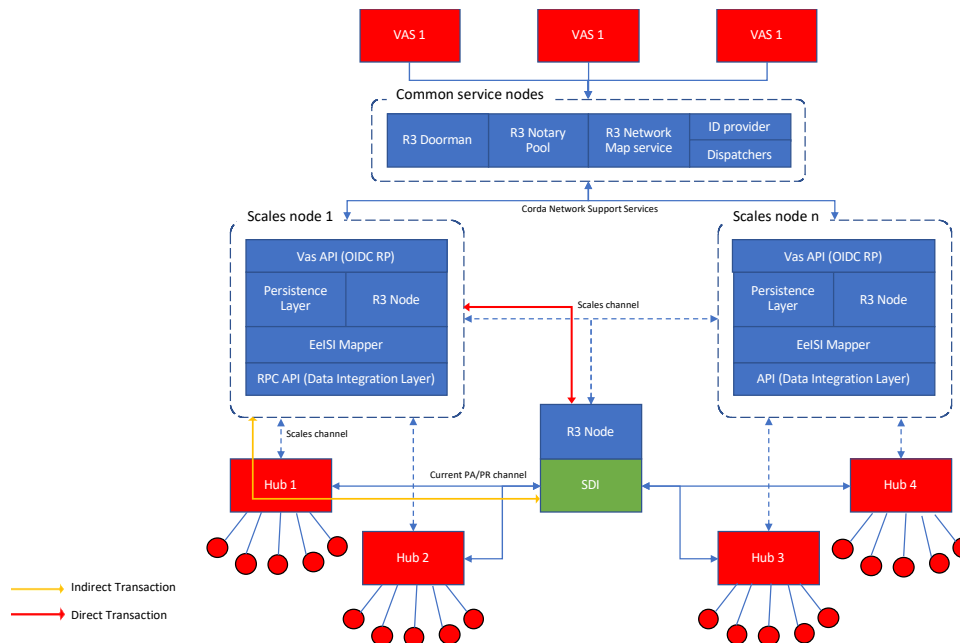
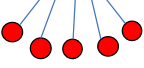
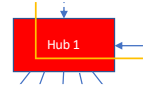
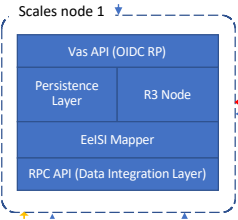
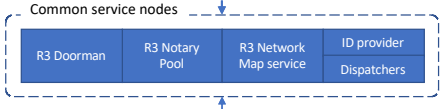
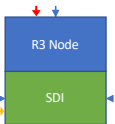



Figure 3 - eInvoicing logical architecture view

Let's describe every component shown in the schema and depict how they interact together.

First of all, the above figure shows the Scales network logical architecture paying attention to a particular use case: Decentralizing the SDI supported processes, while introducing the possibility for Vas partner to provide services to Hub customers (aka End Entities). The same architecture principles apply in the case of NSO and other channels to Scales Networks.

Component	Description
End Entity 	<p>End Entities are the label we shall use throughout the document to identify the legal persons that act as supplier and buyer in the purchase-to-pay process supported by Scales.</p> <p>They are the information object owner and they can choose to authorize third party to access their own data and provide value added services.</p>
Hub 	<p>Hubs are intermediaries between End Entities and Scales network.</p> <p>For instance, in the case of B2B invoicing, they are the ones who interact with the SDI on behalf of their customers.</p> <p>In the mixed architecture, hubs that subscribe to scales shall manage their customer invoice following both the current centralized protocol and the scales one.</p>

<p>Scales Node</p> 	<p>The Scales Node is the main component of the network and it is in charge to:</p> <ul style="list-style-type: none"> - 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
<p>Scales common services</p> 	<p>Scales Common Services are components that are needed to support distributed flows among the network. They are in charge of:</p> <ul style="list-style-type: none"> - 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.
<p>SDI</p> 	<p>SDI</p> <p>In this schema, which is focused on the invoicing use case, the SDI is the Competent Authority, i.e., the nodes that is involved in every transaction in order to fetch government relevant data and provide validation services.</p> <p>This role may be also played by NSO, in the eProcurement use case.</p>
<p>Vas Partners</p> 	<p>Var partner are the components that provide value added service to end entities through the Scales network Services.</p> <p>They are explicitly authorized by end entities to access to the VAS apis on their behalf, using a identity and authorization protocol (OpenID Connect).</p>

3. Scales Nodes

3.1. Corda business Network features

3.2. Scales Node Description

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.
- EelSI 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. EelSI mapper shall be engaged when a transformation of the object is required, but this shall not substitute the original of the stored data.
- Data Integration Layer:

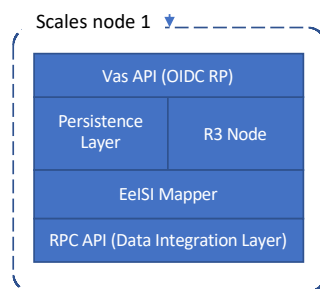
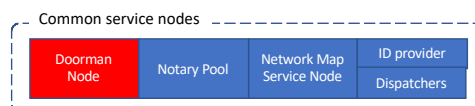


Figure 4 – scales node software stack

3.3. Scales Common Services

3.3.1. Doorman

Doorman component is in charge of KYC and Enrollment of x509 credentials to nodes and clients that performs transactions in Corda Flow



Host	Individual identity	Service identity
notary-1	O=Worker 1, L=London, C=GB	O=HA Notary, L=London, C=GB
notary-2	O=Worker 2, L=London, C=GB	O=HA Notary, L=London, C=GB
notary-3	O=Worker 3, L=London, C=GB	O=HA Notary, L=London, C=GB

Figure 5 - Commons Service nodes: Doorman

3.3.2. Network Map

Network map service is the common component that is in charge of maintaining the mapping between nodes name and IPS. It is a sort of private DNS for the network.

Each node carrying out a flow shall use the network services to correctly address the signature request to the proper counterparty,

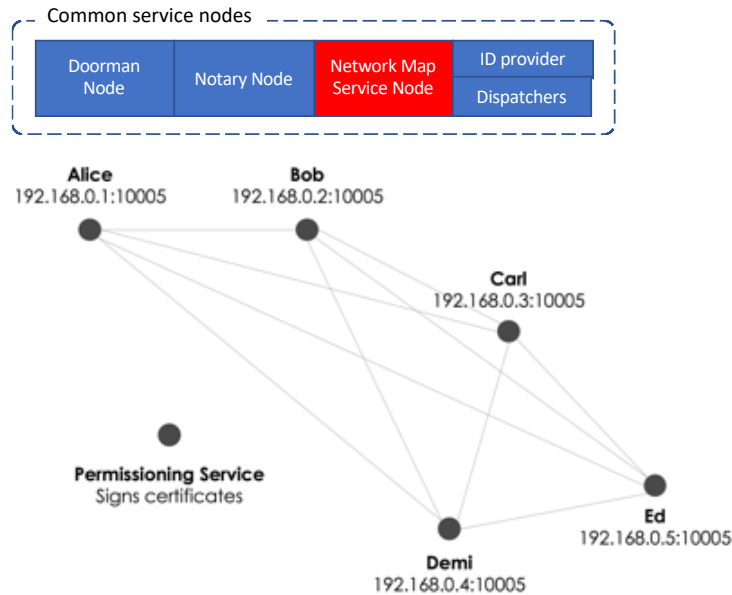


Figure 6 – Commons Service nodes: Network Map

3.3.3. Notary Pool

Notary Pool component is a standard Corda Network element and it is in charge of:

- Preventing double-spends if the transaction has inputs
- Serving as a timestamping authority if the transaction has a time-window

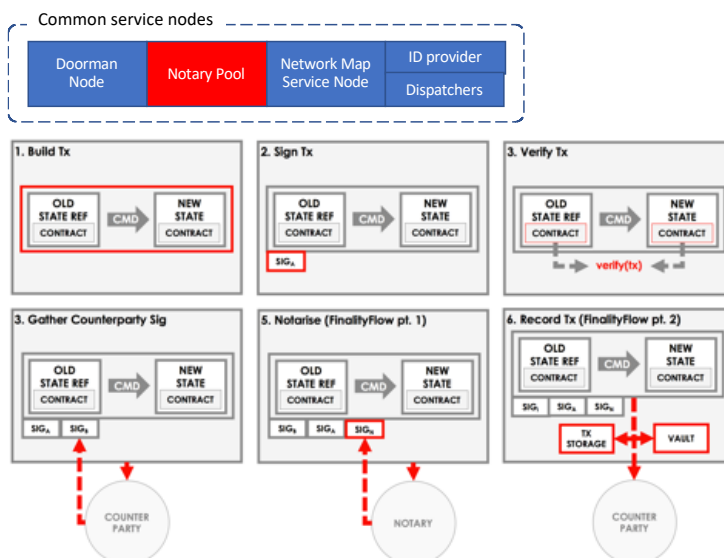


Figure 7 - Commons Service nodes: Notary pool

As provided by the standard Corda Flow paradigm, the notary pool is engaged to add a final signature to a flow, in order to grant that that flow is uniquely executed.

A Flow is a set of transaction that changes the state of an object from an initial one to a final one. This status change is similar to bitcoin UTXO or Ethereum Smart Contract execution, where the consensus algorithm grant that an account can spend its tokens just once, or that a state variable in a smart contract can be uniquely updated. Notary pool has the same responsibility and it is the component through which the Corda consensus algorithm is implemented.

3.3.4. Identity provider

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.

3.3.5. Dispatcher

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 an 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.

3.4. OnBoarding Service

Among the common service we include an onboarding service that shall allow the end entity to have a scales Identity and use it to authorize 3rd party player to access and transact on its own data.

3.4.1. Onboarding of Intermediary-HUB

Since the relative restricted number of candidates in this role, intermediaries / Hubs shall be onboarded via a manual process, on a case by case.

3.4.2. Onboarding of End Entities

End Entity shall be able to upload invoices to Scales Node only through intermediary/hub.

The project assumes that If an Intermediary is able to provide Invoice and SDI notification for a specific End Entity (Vat), the intermediary has already absolved to the End Entity KYC obligation and agreement.

Intermediary must provide evidence of a delegation agreement between him and the End Entity related to the upload of invoices on the intermediary Scales Node of reference.

This evidence must be supported by a dedicated Corda flow, provide, as attachment, a document digitally signed by the End Entity.

At the end of the process, the End Entity shall be able to create an identity on scales identity management system, in order to access to all the pertaining data (invoices, orders, vas proposal), via APIs and Web Interface.

End Entity onboarding shall be carried out only by a legal representative of the End Entity itself.

Operations can be carried out by the Scales node legal entity and must assure the identity of the legal representative prior the identity creation.

3.4.3. Onboarding of VAS

Since the relative restricted number of candidates in this role, VAS shall be onboarded via a manual process, on a case by case.

3.4.4. EndEntity – VAS agreement

This agreement is out of scope of the present document. Yet we consider useful to keep a reminder for this topic.

3.5. “as is” versus “to be” flows

In order to get acquainted with the typical transformation of centralized processes in decentralized / distributed workflow, such as the case of this project, we take in consideration as initial single case, the B2B invoicing process.

In the next paragraph we'll be showing how the current B2B invoicing process is transformed in two alternative types of flow, accordingly to what has been described in par. 2.5, i.e. two possible way to involve the central component (SDI or NSO) in the new flow: direct or indirect transaction.

Another assumption to be underlined is that invoices may be submit to scales nodes from both the active or passive subject; for the sake of clarity, the flows described in the following paragraphs shall refer to the Active case.

3.5.1. “as is” flow for B2B Invoicing

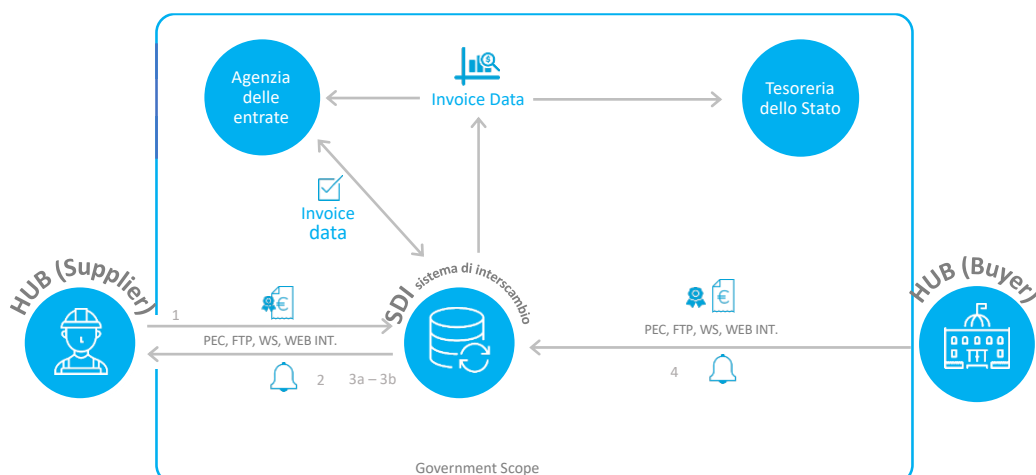


Figure 8 - "as is" B2B invoicing flow

Here follows the description of the main flow, depicted in Figure 8:

1. Supplier HUB sends Invoice to SDI
2. SDI return an invoice ID
3. SDI notifies Supplier that
 - a. the invoice has been successfully delivered to buyer HUB
 - b. the invoice delivery to Buyer HUB has failed

3.5.2. “to be” flow for B2B invoicing – Direct Flow

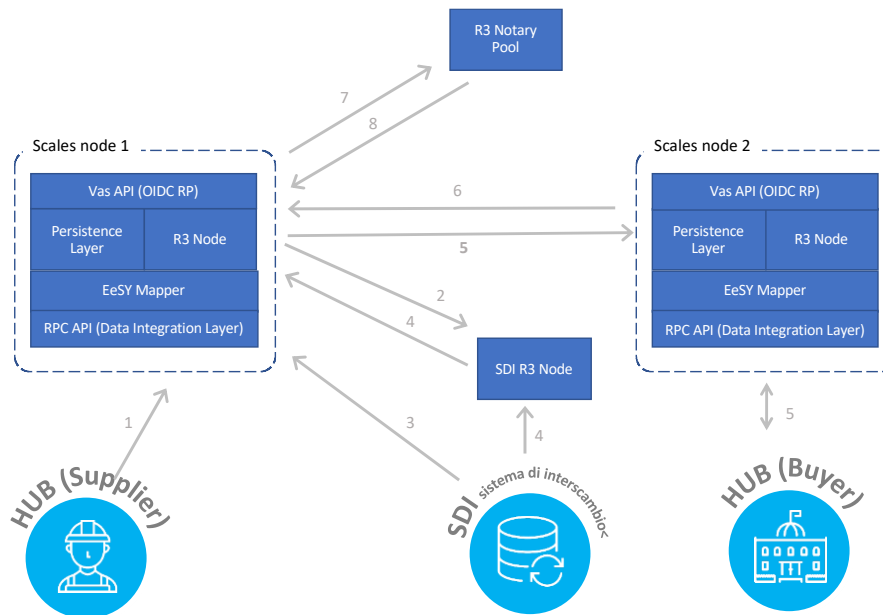


Figure 9 – “to be” flow for B2B invoicing – Direct Flow

Here follows the description of the main flow, depicted in Figure 9:

1. Supplier HUB create Invoice on its Scales node (1) and starts B2B Corda Flow
2. Scales Nodes signs transaction to change invoice state (*) and sends it to SDI note for countersign
3. SDI node gets the transaction signing request and get the invoice metadata from the RPC scales node
4. SDI validate the state change and signs the transaction back
5. Scales Node send transaction signing request to Buyer Scales node (2), which notify request and invoice metadata to Buyer HUB
6. Scales Node 2 Signs transaction back to Supplier Scales Node
7. Scales Node ask notary node chosen from the pool to sign and timestamp the transaction
8. notary node verifies transaction uniqueness and signs it back to the Scales node

3.5.3. “to be” flow for B2B invoicing – Indirect Flow

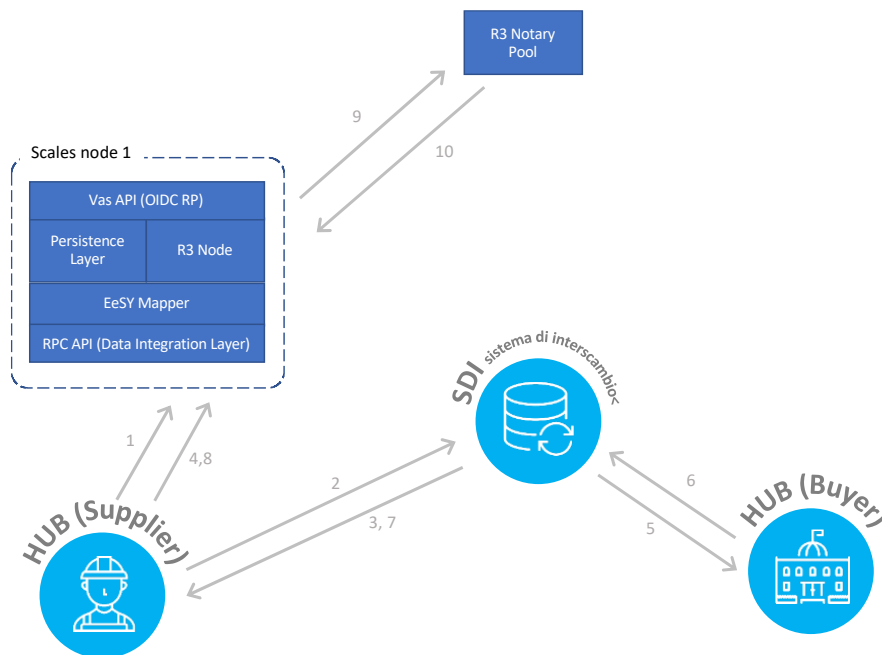


Figure 10 - “to be” flow for B2B invoicing – Indirect Flow

Here follows the description of the main flow, depicted in Figure 10:

1. Supplier HUB create Invoice on its Scales node and starts B2B Corda Flow
2. Supplier submit Invoice to SDI with via the current standard protocol
3. SDI executes its business logic to the invoice and responds to Supplier HUB with the standard protocol
4. Supplier HUB update invoice state by signing a transaction including, as attachment, the signed notification received by SDI
5. SDI deliver Invoice to Buyer Hub
6. Buyer HUB acknowledges the Invoice
7. SDI notifies to Supplier the delivering of invoice to Buyer Hub
8. Supplier Hub updates the invoice status on the base of the notification received from SDI
9. Scales node validate attachments signs the transaction and asks a notary to sign and timestamps the transaction
10. notary node verifies transaction uniqueness and signs it back to the Scales node

4. Data Integration Layer Component

4.1. 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

4.2. 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.

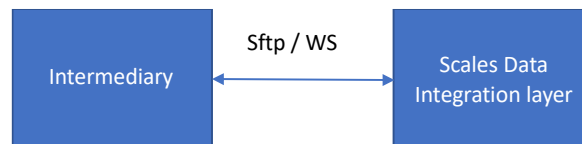


Figure 11 - Integration Channel between Hub/Intermediary and Scales Node

As shown in Figure 3, 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

4.2.1. 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 an 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 from 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
 - The invoice

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

- 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.

4.2.2. Web Service Integration (indirect mode)

Please refer to SCALESAPI1.0.yaml, in particular:

- POST: /api/upload-invoice
- POST: /api/upload-order

4.3. Data models

4.3.1. 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

4.3.2. 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)

5. 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.)

5.1. Third party registration

The third party enroll its client through 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.

5.2. 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 allow 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": "U8pN88yPtUjXElQkTjQ3F7M4LGN_yG1z",
  "id_token":
    "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJjbGllbnRJRCI6ImtieXVGRGlkTExtMjgwTEl3VkZpYXpPcWpPM3R5
    OEtllwiY3JlYXRlZD9hdCI6IjIwMTk0MTU0ZDg1N30uZXksoRhdYaTSN-K5tAUOkIOFZaChfaqxAbcyFck8nyo",
  "scope": "openid profile ",
  "expires_in": 86400,
  "token_type": "Bearer"
}
```

5.3. Open API Specification

SCALES Api are specified in the SCALESAPI1.0.Yaml open Api specification.

5.4. Vas API Flow

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

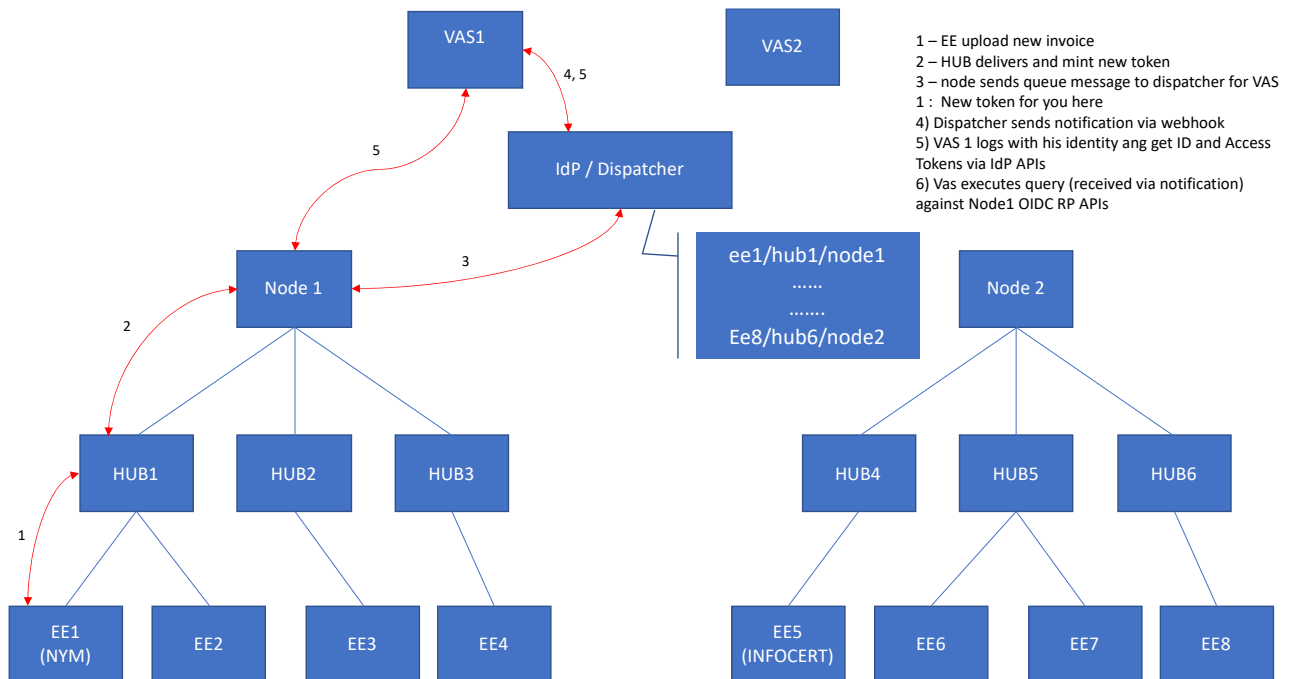


Figure 12 - Vas API Flow

Vas-api are detailed in SCALES1.0.yaml open api specification.

5.5. Reason to adopt OpenID Connect standard protocol

There are a number of reasons 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 heterogeneous 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**

5.6. 3rd Party Transaction Model

In this paragraph we describe the high-level sequence of transaction involving a 3rd party that interacts 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

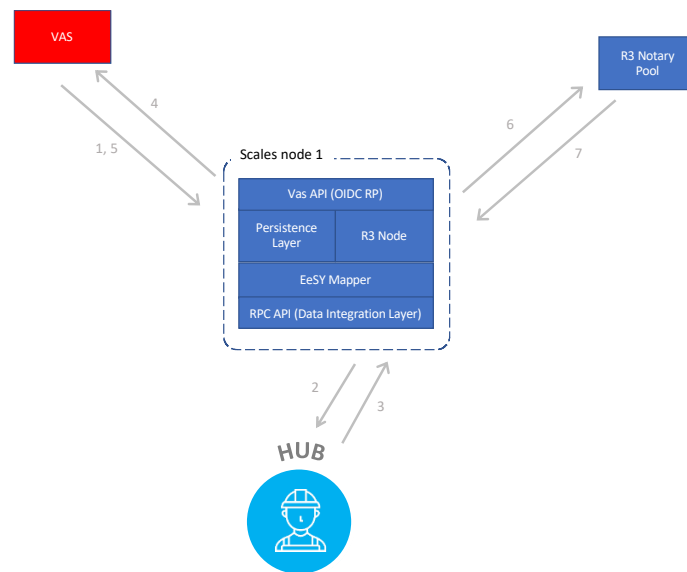


Figure 13 – 3rd Party Transaction Model

6. Scales Network initial deployment

6.1. Initial Deployment Scenario

The figure below describes the proposed initial scenario, on the base of the current Scales consortium participants.

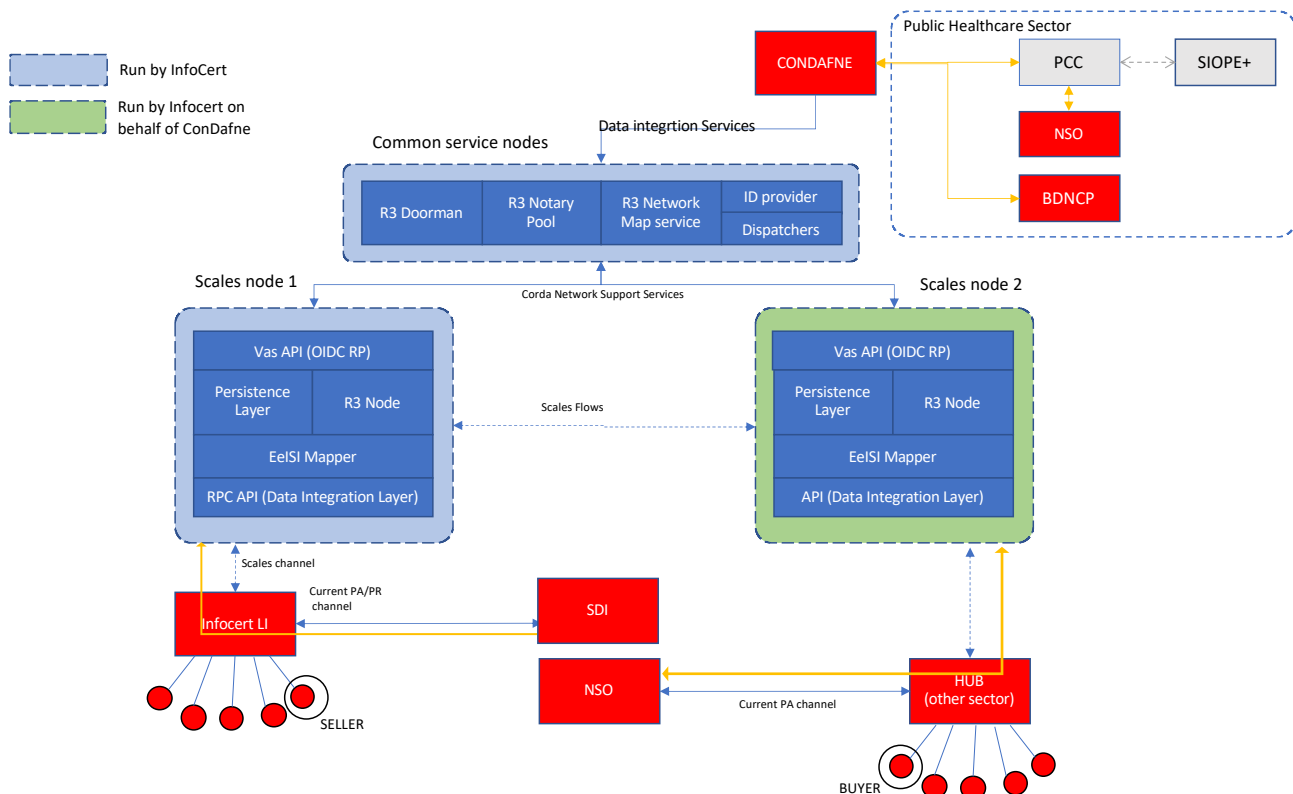


Figure 14 - initial Deployment Scenario

With the two partners Infocert and Dafne Consortium, the initial deployment could cover the major part of the Scales use case, particularly:

- B2B and B2G eInvoicing
- eOrder

further on Infocert could provide support to run Common Services, including onBoarding of end entities, while Dafne Consortium could act as a VAS Partner for the CCP information to suppliers.

6.2. Consortium to run Common Services

Since the Scales architecture is based on DLT technology and particularly Corda R3, and mixes different processes and stakeholders that are potentially competing in their respective market, it seems appropriate to envision a Scales Consortium to be active since the initial deployment of the network and in charge of governing:

- the standards and technology rules shared with the members. For instance, adoption of particular schema, or changes to the flows. This would result in committing to the DLT the rule changes voted by members, make those rules always available to members and their partners.
- The costs needed to run common service nodes

7. Generalisation and future development

7.1. Abstract Architecture Pattern

The design of Scales DLT led to an abstract architectural design pattern, shown in the below figure:

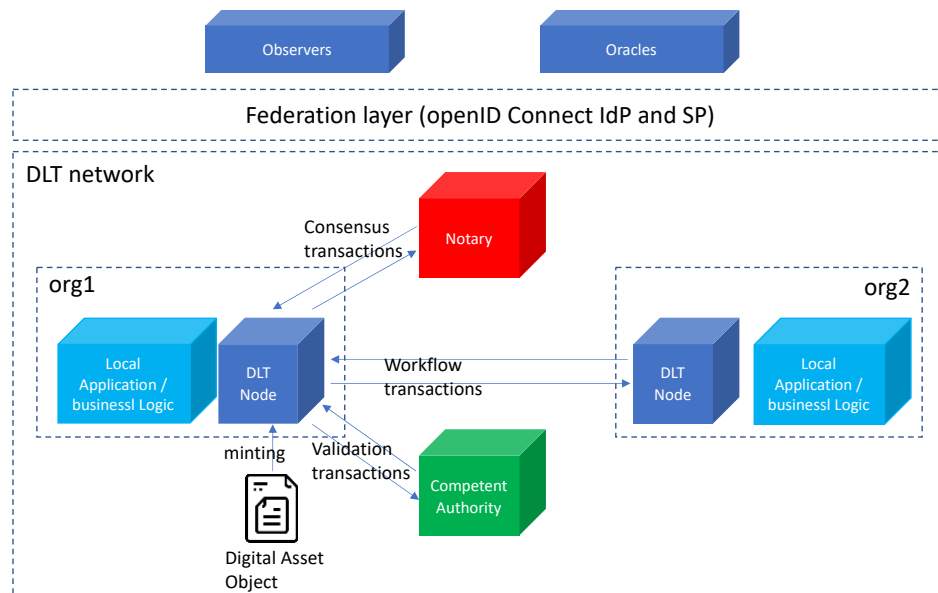


Figure 15 – abstract architectural design pattern

The schema shows the following components:

- A DLT core network, composed of:
 - o A notary, in charge of implementing a consensus protocol based of BFT algorithm
 - o A competent authority, in charge of data asset validation
 - o At least two DLT full nodes, in charge of implementing the smart contract flows
 - o The digital asset object, that through a minting operation, is introduced in the system as a representation of a real-world object or a digitally native object
- A federation layer, based on Open ID Connect protocol, allowing the external entities to observe or actively interact with the DLT core network
- Two different class of components that may interact with the network though the federation layer, i.e.:
 - o Observers:
 - o Oracles:

The above schema could represent a useful starting point for the design of asset tokenization initiatives, particularly in the case of regulated context.

7.2. Self-Sovereign Identity

Self-Sovereign identity (SSI) constitutes the next paradigm of on-line digital identity, evolving the current federated identity model.

SSI is a strategic and core component of ESSIF and EBSI EU initiatives.

SSI brings several advantages, even when applied in a B2B context like Scales, particularly:

- Portability of identity, allowing a password-less/direct authentication to service
- Minimal disclosure of identity attributes by the holder to a verifier
- Prevention of data correlation thanks to pairwise decentralised identifier exchange
- Privacy preserving Zero Knowledge proof-based attribute presentations

The introduction of SSI in Scales DLT could lead to a series of enhancement like, for instance:

- Standardisation of the onboarding procedure for the end entity
- Full decentralisation of accounts: thanks to Self-Issued Open Id Connect credentials, the end entities could interact with the federation layer APIs using DID authentication, yet on an OPEN ID based protocol.

Last but not least, as soon as EBSI/ESSIF project will launch and reach full regime, identities already enrolled could be used to subscribe and access to Scales.

7.3. Smart legal Contract

Scales project uses a Smart contract-based approach for the tokenisation of invoice: using the r3 Corda token SDK the invoice was modelled as a non-fungible token.

This sets the base upon which a real legal contract framework could be built, mixing the contract code with prose attachments giving full legal effect to the resulting minted token.

In this way, every asset and every transaction in the network would be uniquely associated to the contractual foundation, for each specific participant.

7.4. eDelivery (Zero Corner Model)

As described in paragraph 2.6, Scales implements two possible flows: Direct and Indirect flows.

The Direct flow represents an initial implementation of a Zero Corner Model schema, where the network is in charge of dynamically distributing contents on the base of the document and not a predefined/static interconnection schema.

This leads to a possible evolution of Scales in order to match compliance with certified delivery service requirements for trust services.