EUROPEAN COMMISSION

DIRECTORATE-GENERAL FOR INFORMATICS

Piloting Domibus integration with CEF EBSI (blockchain): Implementation Overview

Published on 8 September 2022

Document Author(s)

Author Name	Role
Joze Rihtarsic	Software Developer

Document Reviewer(s)

Reviewer Name	Role
Bogdan Dumitriu	eDelivery Project Officer

Document Approver(s)

Reviewer Name	Role
Bogdan Dumitriu	eDelivery Project Officer

This document was created as a deliverable in the 2020 ISA^2 Innovative Public Services (IPS) action.

Table of Contents

1 Introduction

The purpose of piloting Domibus integration with CEF EBSI (blockchain) was to explore several areas where the CEF eDelivery message exchange could be enriched with functionality relying on the blockchain technology. In a practical sense, the exploration consisted of modifying the **Domibus** sample software to use the **CEF EBSI infrastructure** to pilot two different scenarios that with the potential to provide significant added value to the users of CEF eDelivery.

Domibus is an Access Point implementation that is conformant to the CEF eDelivery AS4 profile, maintained by the European Commission, with the purpose to ensure the safe, non-repudiable and reliable message exchanges.

The European Blockchain Services Infrastructure (EBSI) is a joint initiative from the European Commission and the European Blockchain Partnership (EBP) established to deliver cross-border public services using blockchain technology in the EU.

1.1 Purpose of the document

This document provides an overview of the technical implementation of the pilot. It describes the components that were developed, along with insight into their logical design and the concrete use of EBSI services.

The implementation serves as proof of concept of the selected scenarios and can be used as a basis for further prototyping and development of enriched message exchange using blockchain technology.

1.2 Limitations

Before the results can be used for production purposes, additional quality and reliability assurance work must be carried out and, at the minimum, a service level agreement would need to be established between the concerned eDelivery network and EBSI.

2 Use case implementation

The two main scenarios that were implemented in the pilot are described in document (ISA2).(eDelivery).(Piloting Domibus integration with CEF EBSI).(Software Functional Specification).(v1.0).docx:

- Story 02 – Timestamping of eDelivery AS4 messages using the EBSI notarisation
- Story 04 – Using the EBSI DID Registry for authentication/authorisation in an eDelivery network

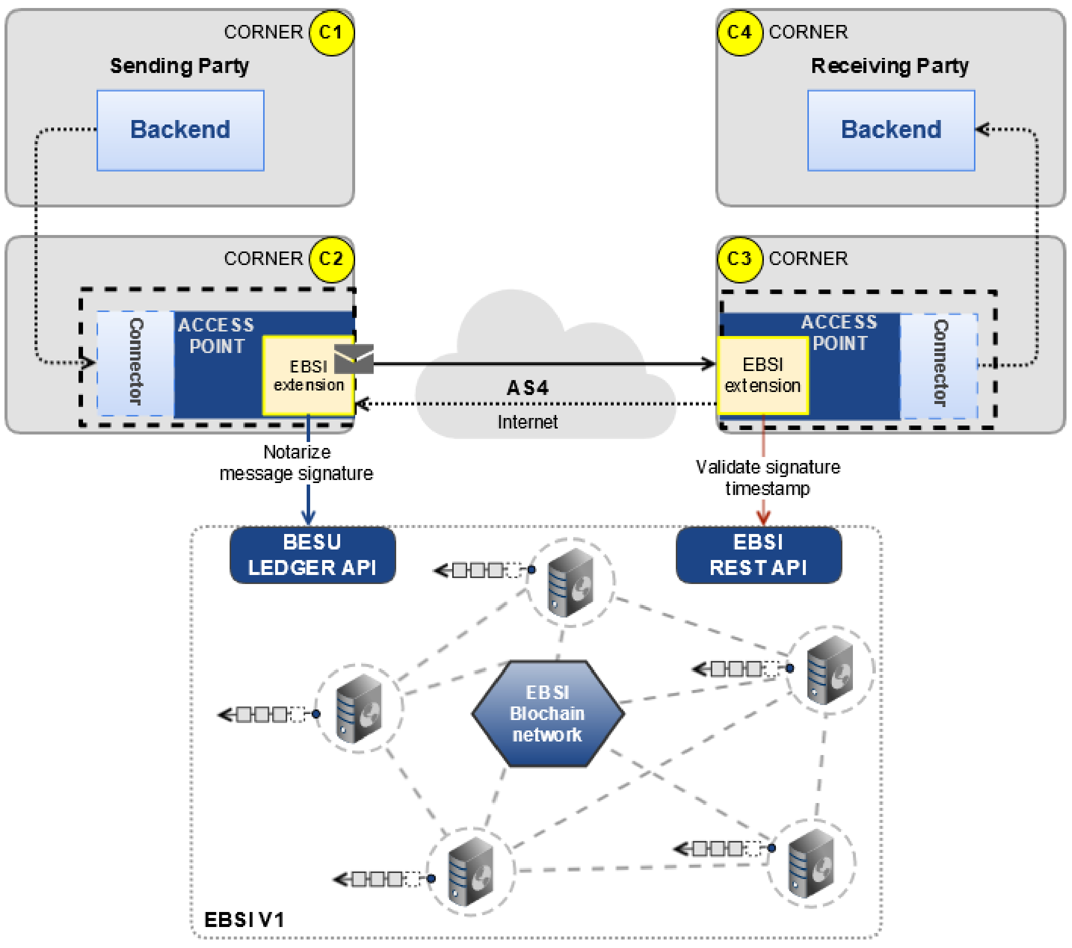
Given the limitations of EBSI at the time of the implementation of the pilot, the prerequisite "Story 01 – Enabling of a Domibus AP to access the authenticated services of the EBSI infrastructure" could not be implemented in its described form due to lacking support in the EBSI API. The preparatory steps needed for for authentication were conducted manually.

The remainder of this document assumes familiarity with the scenarios as described in the referenced document.

2.1 Implementation of "Timestamping of eDelivery AS4 messages using the EBSI notarisation"

The **timestamp** of the signature of an AS4 message serves to prove that the message existed before that particular time. For example:

- On the **sending side**, the timestamp of an AS4 message can serve to prove that the message was already prepared, ready to be sent at a particular time and, importantly, that it could no longer be modified after that time. In certain business scenarios, the existence rather than the receipt of a message could be relevant. This would allow parties to establish a given shared reality even in the absence of a timely delivery of the message, e.g., due to a network or technical issue. One example is that of a public procurement procedure with a firm deadline. The public authority could avoid disqualifying bidders in case of technical issues on its side by relying on the timestamp to assert which bids were available before the deadline.
- On the **receiving side**, the timestamp of the AS4 acknowledgement can serve to prove it had received the message before that particular time. This can strengthen the legal value of the acknowledgement, since it is no longer the local server time which is provided, but the EBSI network time.



2.1.1 Implementation Overview

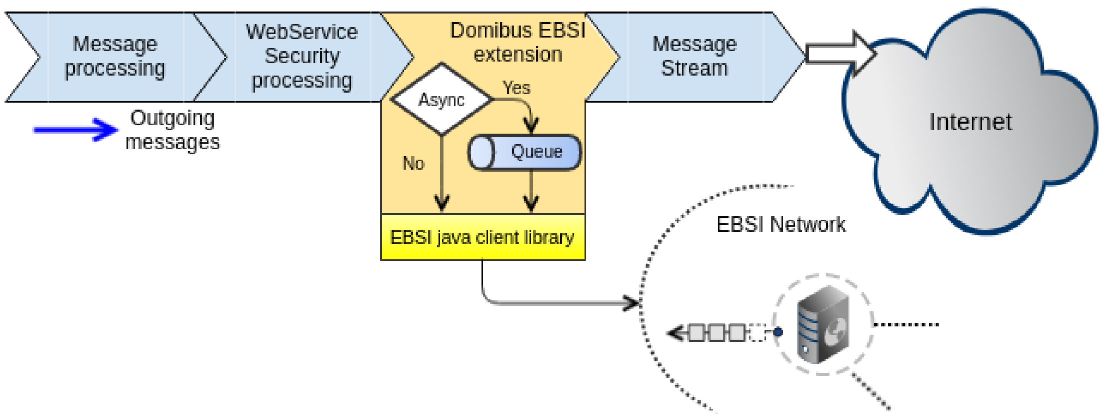
To use the EBSI notarisation service for timestamping AS4 messages, the following components were developed:

- **Domibus interceptor SPI**: proposal for a Service Provider Interface (SPI) for Domibus. The SPI enables the development and the deployment of a custom plugin for SOAP messages. The plugin processes received and sent AS4 Messages without modifying the code of Domibus.
- **Domibus interceptor SPI EBSI extension**: is the Domibus extension implementing the Domibus interceptor SPI for notarising outgoing and incoming AS4 messages using the **EBSI Notarisation service.**
- **EBSI Java client library:** common EBSI integration tools for blockchain smart contract invoking as well as a simple EBSI REST service client are extracted to a common library. The library can be independently developed and used by third party AP solutions.
- **"EBSI enhanced" Domibus:** the pilot implementation is based on Domibus version 4.2.1. This version of Domibus cannot enable extensions to add their own UI elements to the user interface. Therefore, for demo purposes, the existing Domibus Admin Console has been custom extended with UI features for a Domibus administrator to be able to validate the EBSI timestamp.

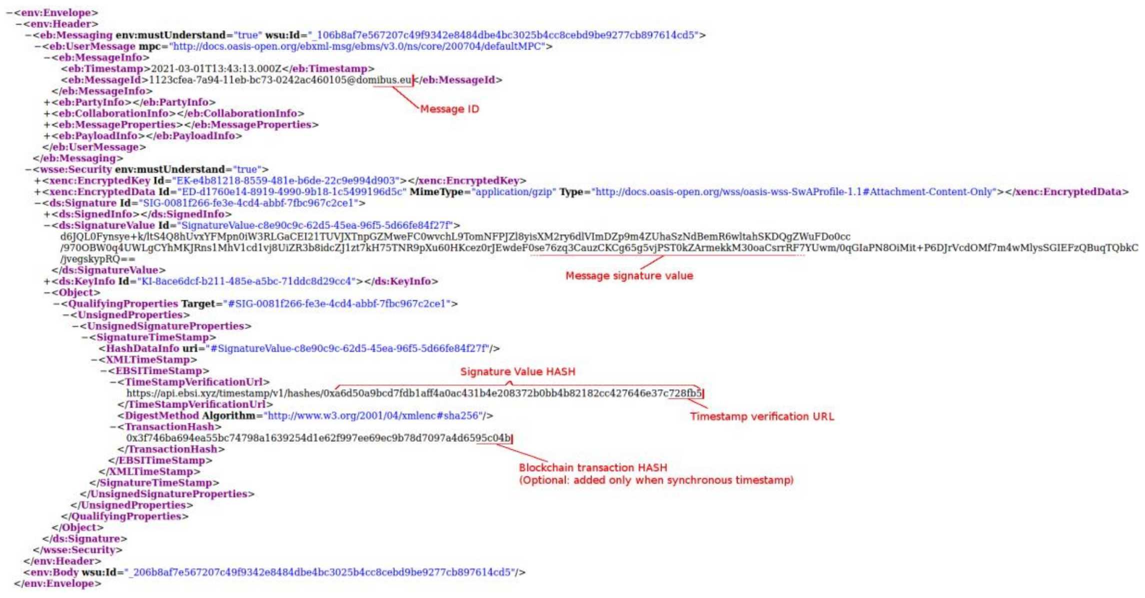
2.1.2 Domibus interceptor SPI EBSI extension

The current implementation of the Domibus interceptor SPI EBSI extension has features to invoke EBSI signature notarisation synchronously and asynchronously. The synchronous notarisation ensures the timestamp of the signature is present before receiving the message. In the asynchronous mode, the timestamp is executed via the JMS queue with no time delay on the message flow. The drawback is that the party verifying the timestamp should delay the verification by a reasonable amount.

The Domibus interceptor SPI EBSI extension intercepts the message flow for outgoing messages and executes the EBSI notarisation/signature timestamp after the message WebService security processing and before streaming the message to the outbound channel (see the image below). Intervening in this step enables adding the timestamp verification URL to the signature. The receiver can use the URL to verify the EBSI signature timestamp.

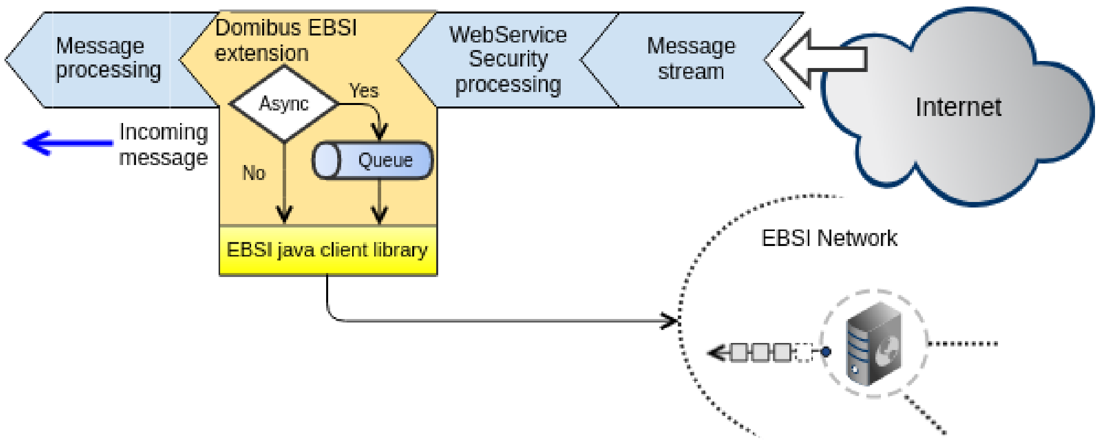


The timestamp verification URL is added to the signature in the XMLDSig object without invalidating the signature. In the pilot phase, we follow the structure of the XADES-T timestamp. However, the timestamp is NOT XADES-T compliant due to a lack of specifications for blockchain timestamps.



```
−<env:Envelope>
  −<env:Header>
    −<eb:Messaging env:mustUnderstand="true" wsu:Id="_106b8af7e567207c49f9342e8484dbe4bc3025b4cc8cebd9be9277cb897614cd5">
      −<eb:UserMessage mpc="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/defaultMPC">
        −<eb:MessageInfo>
           <eb:Timestamp>2021-03-01T13:43:13.000Z</eb:Timestamp>
           <eb:MessageId>1123cfea-7a94-11eb-bc73-0242ac460105@domibus.eu</eb:MessageId>
         </eb:MessageInfo>                                       Message ID
        +<eb:PartyInfo></eb:PartyInfo>
        +<eb:CollaborationInfo></eb:CollaborationInfo>
        +<eb:MessageProperties></eb:MessageProperties>
        +<eb:PayloadInfo></eb:PayloadInfo>
        </eb:UserMessage>
      </eb:Messaging>
    −<wsse:Security env:mustUnderstand="true">
      +<xenc:EncryptedKey Id="EK-e4b81218-8559-481e-b6de-22c9e994d903"></xenc:EncryptedKey>
      +<xenc:EncryptedData Id="ED-d1760e14-8919-4990-9b18-1c5499196d5c" MimeType="application/gzip" Type="http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-Only"></xenc:EncryptedData>
      −<ds:Signature Id="SIG-0081f266-fe3e-4cd4-abbf-7fbc967c2ce1">
        +<ds:SignedInfo></ds:SignedInfo>
        −<ds:SignatureValue Id="SignatureValue-c8e90c9c-62d5-45ea-96f5-5d66fe84f27f">
           d6JQL0Fynsye+k/ltS4Q8hUvxYFMpn0iW3RLGaCEI21TUVJXTnpGZMweFC0wvchL9TomNFPJZl8yisXM2ry6dlVImDZp9m4ZUhaSzNdBemR6wltahSKDQgZWuFDo0cc
           /970OBW0q4UWLgCYhMKJRns1MhV1cd1vj8UiZR3b8idcZJ1zt7kH75TNR9pXu60HKcez0rJEwdeF0se76zq3CauzCKCg65g5vjPST0kZArmekkM30oaCsrrRF7YUwm/0qGIaPN8OiMit+P6DJrVcdOMf7m4wMlysSGIEFzQBuqTQbkC
           /jvegskypRQ==
         </ds:SignatureValue>                                    Message signature value
        +<ds:KeyInfo Id="KI-8ace6dcf-b211-485e-a5bc-71ddc8d29cc4"></ds:KeyInfo>
        −<Object>
          −<QualifyingProperties Target="#SIG-0081f266-fe3e-4cd4-abbf-7fbc967c2ce1">
            −<UnsignedProperties>
              −<UnsignedSignatureProperties>
                −<SignatureTimeStamp>
                   <HashDataInfo uri="#SignatureValue-c8e90c9c-62d5-45ea-96f5-5d66fe84f27f"/>
                  −<XMLTimeStamp>
                    −<EBSITimeStamp>                              Signature Value HASH
                      −<TimeStampVerificationUrl>
                         https://api.ebsi.xyz/timestamp/v1/hashes/0xa6d50a9bcd7fdb1aff4a0ac431b4e208372b0bb4b82182cc427646e37c728fb5
                       </TimeStampVerificationUrl>
                       <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>         Timestamp verification URL
                      −<TransactionHash>
                         0x3f746ba694ea55bc74798a1639254d1e62f997ee69ec9b78d7097a4d6595c04b
                       </TransactionHash>
                     </EBSITimeStamp>                                  Blockchain transaction HASH
                   </XMLTimeStamp>                                     (Optional: added only when synchronous timestamp)
                 </SignatureTimeStamp>
               </UnsignedSignatureProperties>
             </UnsignedProperties>
           </QualifyingProperties>
         </Object>
       </ds:Signature>
     </wsse:Security>
   </env:Header>
   <env:Body wsu:Id="_206b8af7e567207c49f9342e8484dbe4bc3025b4cc8cebd9be9277cb897614cd5"/>
</env:Envelope>
```

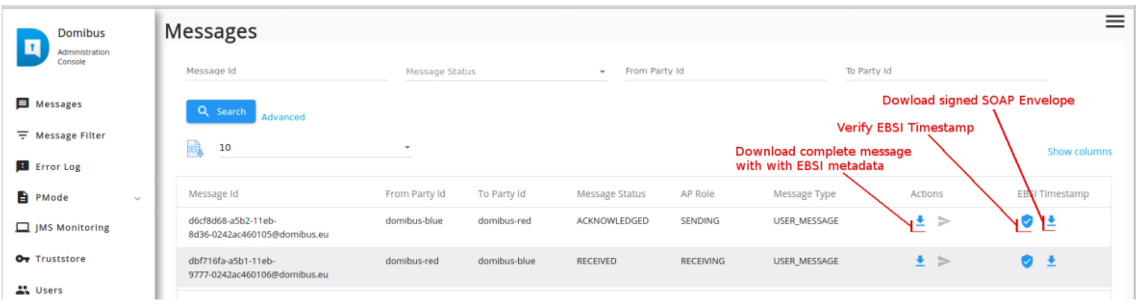
The Domibus interceptor SPI EBSI extension intercepts the message flow for incoming messages and executes the EBSI notarisation/signature timestamp after the WebService security processing (see the image below). If the message signature already contains the EBSI verification URL, the message signature is not notarized for a second time. This restriction comes from the EBSI Notarisation service, where the same hash (in this case of the signature) can only be notarised once. (The added value of a second notarisation performed by the receiver could serve as an additional, out-of band signed acknowledgement of receipt.)



2.1.3 "EBSI enhanced" Domibus

To enable administrators to verify timestamps, the message list of the (pilot) Domibus Admin Console has been extended with the following actions:

- **Verify EBSI timestamp:** this action verifies the timestamp and opens the dialog with the EBSI timestamp metadata.
- **Download signed SOAP Envelope:** this actions downloads the SOAP Envelope message expecting the SOAP Envelope signature.
- **Download complete message:** this existing action has been upgraded to download the complete zipped soap message (SOAP Envelope + attachments) with the EBSI timestamp metadata. The ZIP bundle enables a third party user to build tools to verify message XML signatures and to validate EBSI timestamps.

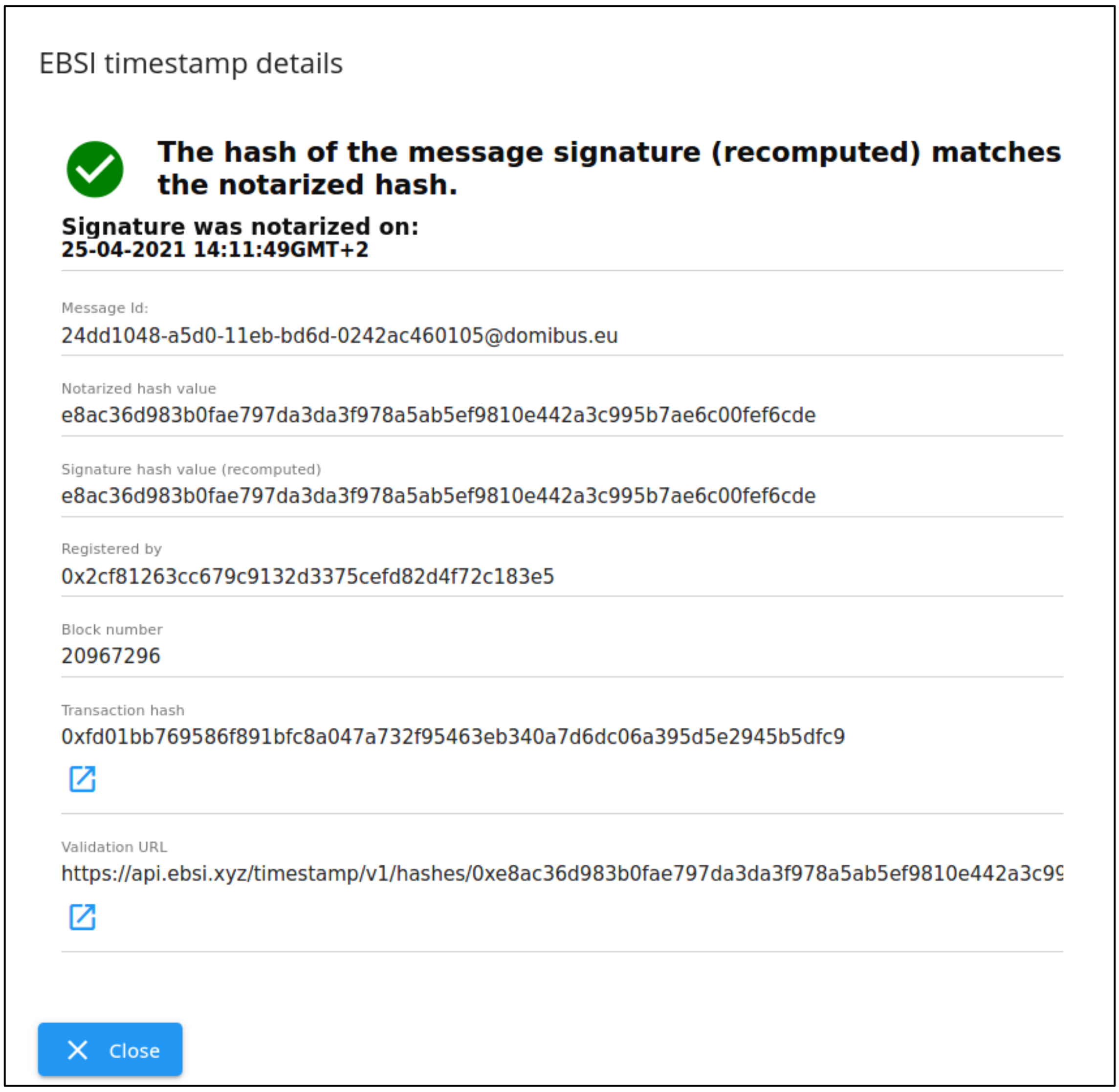


Messages

Message Id	From Party Id	To Party Id	Message Status	AP Role	Message Type	Actions	EBSI Timestamp
d6cf8d68-a5b2-11eb-8d36-0242ac460105@domibus.eu	domibus-blue	domibus-red	ACKNOWLEDGED	SENDING	USER_MESSAGE		
dbf716fa-a5b1-11eb-9777-0242ac460106@domibus.eu	domibus-red	domibus-blue	RECEIVED	RECEIVING	USER_MESSAGE		

2.1.3.1 Action: Verify EBSI timestamp

When this action is triggered, the timestamp verification URL is extracted from the SOAP Message signature, and the timestamp data is retrieved from the EBSI Notarisation service. The response from the service contains the following data: notarisation date, notarised hash value, blockchain transaction hash, block number, and "registered by" address. The Domibus interceptor SPI EBSI extension calculates the hash from the signature value and compares it to the notarised hash value to ensure that notarisation data belongs to the signature. In case the signature value hash matches the notarised hash value, the notarisation data is shown, otherwise the following error message is displayed: "*The hash of the message signature doesn't match the notarized hash!"*



EBSI timestamp details

The hash of the message signature (recomputed) matches the notarized hash.

Signature was notarized on:
25-04-2021 14:11:49GMT+2

Message Id:
24dd1048-a5d0-11eb-bd6d-0242ac460105@domibus.eu

Notarized hash value
e8ac36d983b0fae797da3da3f978a5ab5ef9810e442a3c995b7ae6c00fef6cde

Signature hash value (recomputed)
e8ac36d983b0fae797da3da3f978a5ab5ef9810e442a3c995b7ae6c00fef6cde

Registered by
0x2cf81263cc679c9132d3375cefd82d4f72c183e5

Block number
20967296

Transaction hash
0xfd01bb769586f891bfc8a047a732f95463eb340a7d6dc06a395d5e2945b5dfc9

Validation URL
https://api.ebsi.xyz/timestamp/v1/hashes/0xe8ac36d983b0fae797da3da3f978a5ab5ef9810e442a3c9

2.1.3.2 Action: Download complete message

The purpose of this action is to download a zip bundle that includes the message and all the necessary items that would enable a third party to validate the message and the timestamp. In addition, the timestamp metadata is extracted in a file called **timestamp-result.txt** containing:

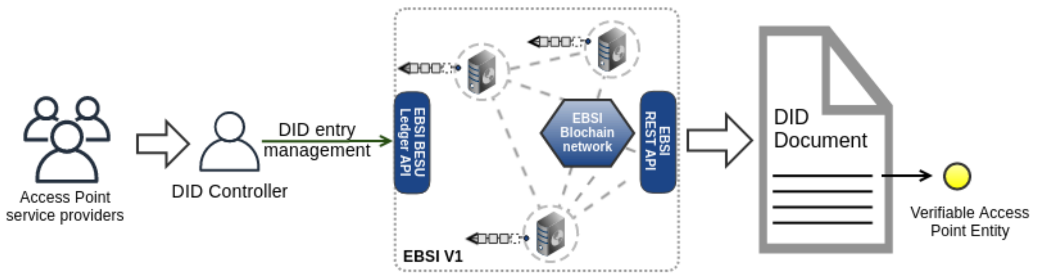
```
Message id: d6cf8d68-a5b2-11eb-8d36-0242ac460105@domibus.eu
Hash: 0a00a54a0c9191e112d4ef766b266aa0175c066819231cf0c2d82d456038e7fa
Timestamp: 2021-04-25T10:41:51
RegisteredBy: 0x2cf81263cc679c9132d3375cefd82d4f72c183e5
TxHash: 0xdbdfed1b1a082dc6f69a6e6e12f5a24bda2cfb6b66c82d148f306e271230beb8
ValidationUrl:
https://api.ebsi.xyz/timestamp/v1/hashes/0x0a00a54a0c9191e112d4ef766b266aa
0175c066819231cf0c2d82d456038e7fa
```

Code Block 1 TimestapResult

2.2 Implementation of "Using the EBSI DID Registry for authentication/authorisation in an eDelivery network"

EBSI DID Registry is based on the W3C Decentralized Identifiers (DIDs) v1.0 specification. The specifications describe a new type of identifier that enables verifiable, decentralized digital identity. The identity uses distributed ledger technology to ensure the safe use and trust of the identity.

In the case of the pilot, the DID identifier used is a hash value of the Access Point Identifier and its certificate. Using the hash of certificates alone would not be sufficient as it would not protect against impersonation (i.e., an Access Point with a certificate already trusted by the network claiming to be another Access Point). The DID identifier can only be published in the DID document by the DID controller(s) using the distributed ledger technology. The DID controller can be a business domain owner, a business domain administrator, an identity service provider or another business entity.



Access Points can use the DID Document to verify if the counterparty certificate is authorised and bound to the expected (counter)party identifier.
As an example: when an Access Point (C3) receives a message from another Access Point (C2), it initially validates the message signature and runs preliminary checks of the C2 certificate (e.g., to check that it was issued by a trusted issuer). The DID document can then be used by C3 to also validate that the certificate was authorised for the Access Point C2 in the specific business domain network.

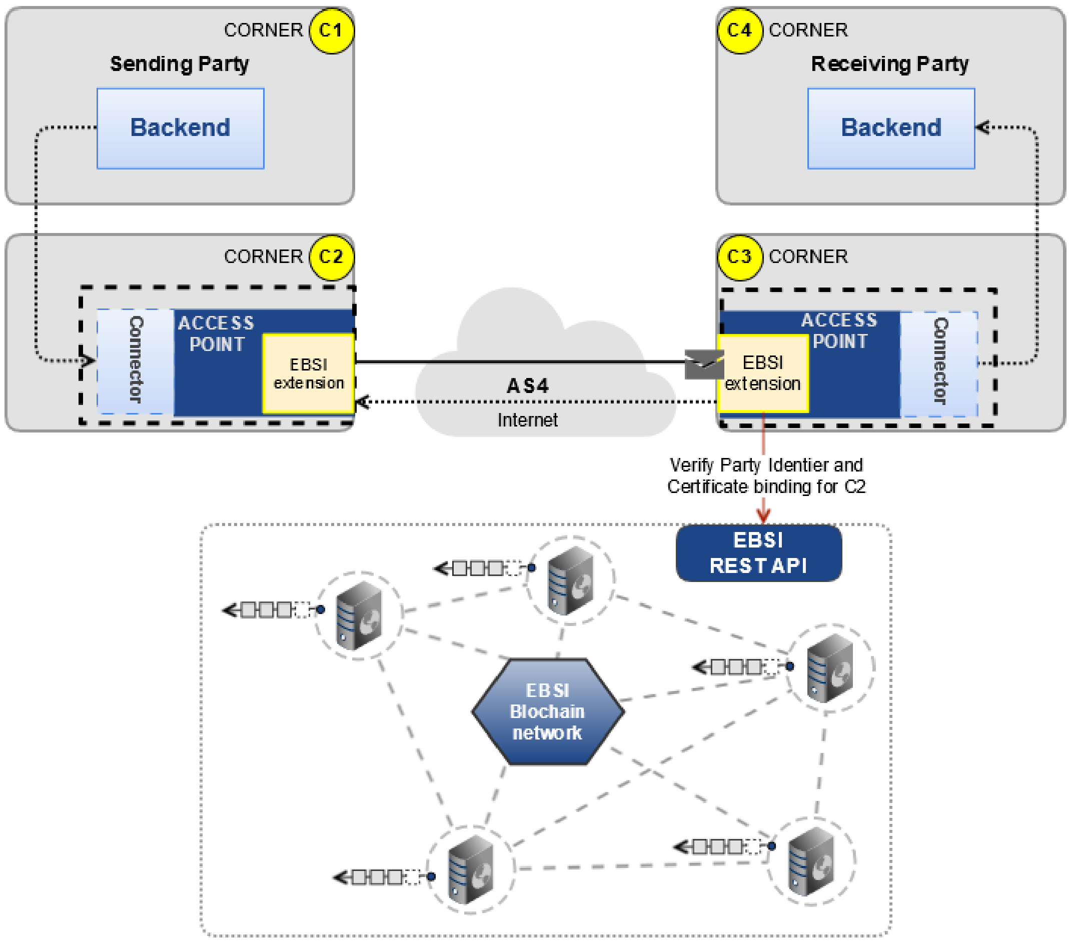
2.2.1 Implementation Overview

To use an eDelivery DID document, the following components were developed:

- **Domibus IAM SPI EBSI extension:** is the Domibus extension implementing the Domibus IAM SPI that verifies if the certificate used to signed the incoming AS4 message is authorised in the DID Document. (The Domibus IAM SPI is the standard Domibus SPI which enables custom implementation of the authorisation for the incoming messages.)
- **eDelivery EBSI DEMO application**: a simple web application for administering the DID document used for testing and demonstration.

2.2.2 Domibus IAM SPI EBSI extension

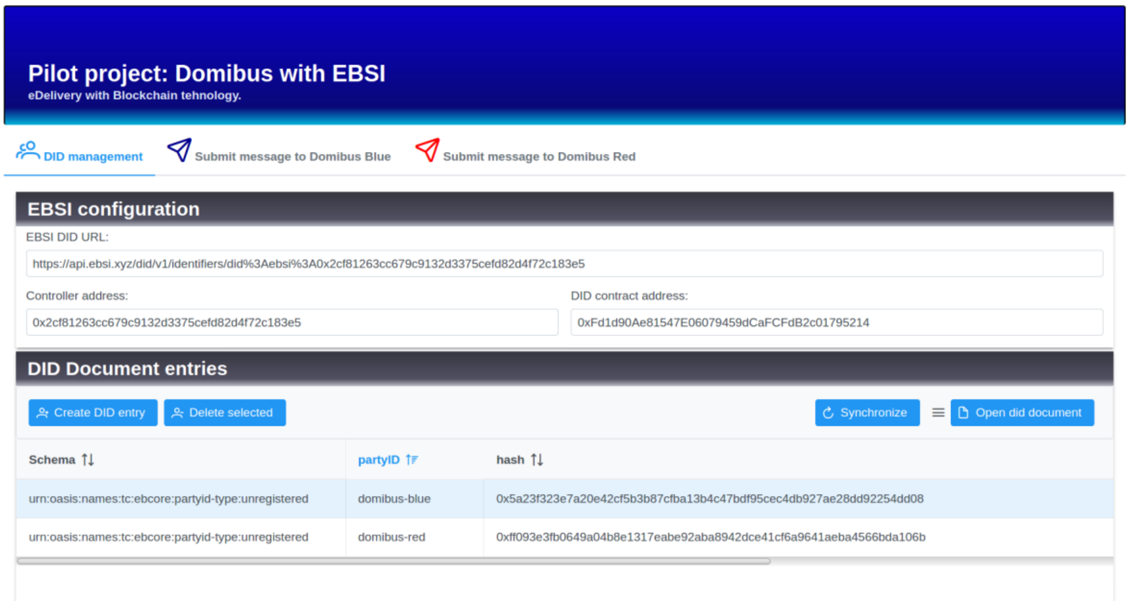
The extension performs the standard authentication steps to verify the certificate's trust and then uses the EBSI DID document to validate if the certificate is authorised to be used in the network by the identifier of the sender party of the incoming message.



2.2.3 eDelivery EBSI DEMO application

The purpose of the application is to build a simple tool for testing and demonstrating the authorisation process, including:

- Managing the EBSI DID document for the eDelivery network
- Submitting messages to the Access Points C2 and C3


Pilot project: Domibus with EBSI
eDelivery with Blockchain tehnology.
DID management
Submit message to Domibus Blue
Submit message to Domibus Red
Create DID entry
Delete selected
Synchronize
Open did document

EBSI configuration

EBSI DID URL: https://api.ebsi.xyz/did/v1/identifiers/did%3Aebsi%3A0x2cf81263cc679c9132d3375cefd82d4f72c183e5

Controller address: 0x2cf81263cc679c9132d3375cefd82d4f72c183e5

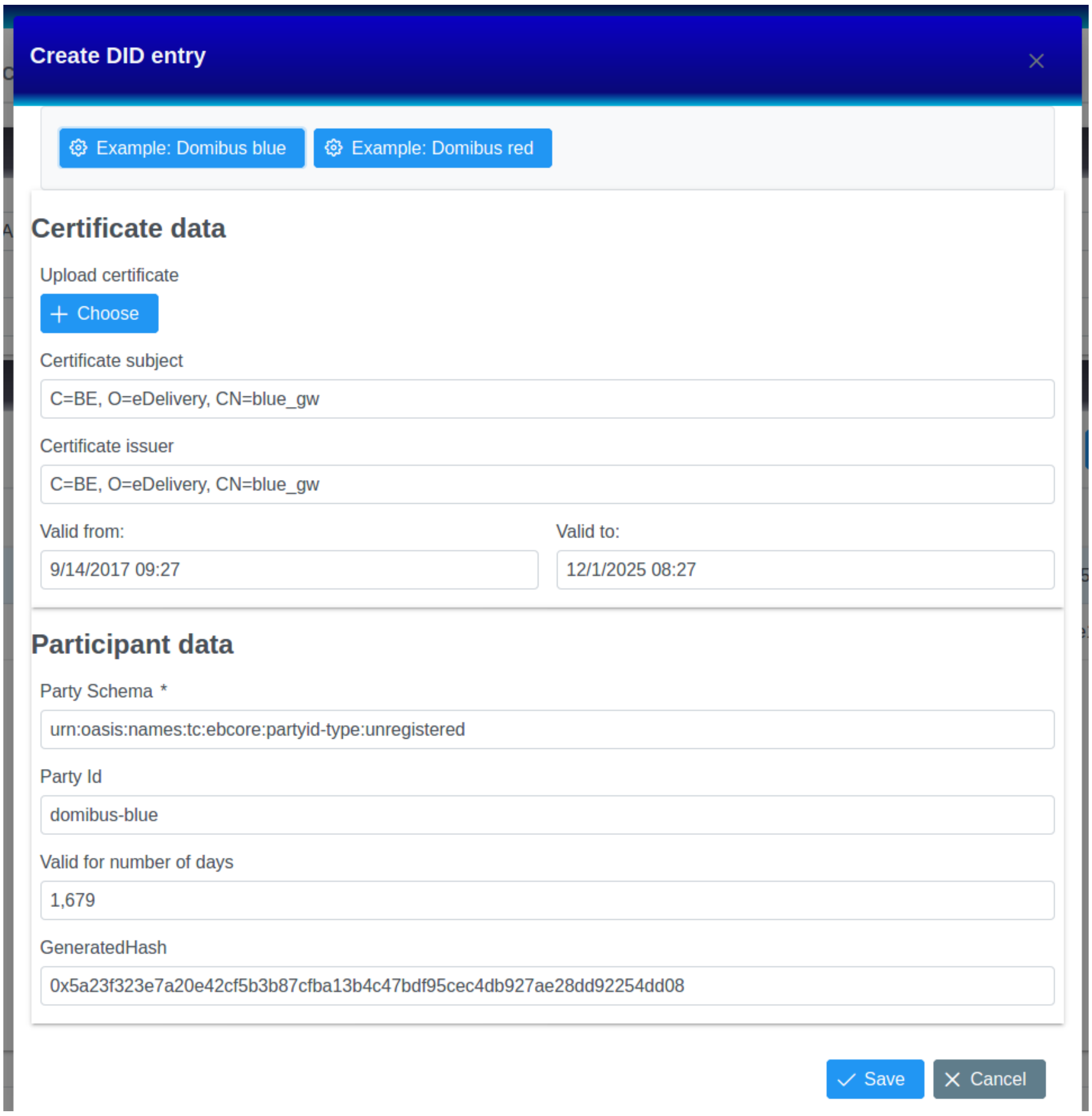
DID contract address: 0xFd1d90Ae81547E06079459dCaFCFdB2c01795214

DID Document entries

Schema	partyID	hash
urn:oasis:names:tc:ebcore:partyid-type:unregistered	domibus-blue	0x5a23f323e7a20e42cf5b3b87cfba13b4c47bdf95cec4db927ae28dd92254dd08
urn:oasis:names:tc:ebcore:partyid-type:unregistered	domibus-red	0xff093e3fb0649a04b8e1317eabe92aba8942dce41cf6a9641aeba4566bda106b

Managing the EBSI DID document involves adding a new party identifier with the certificate, calculating the hash from the concatenation of the party identifier and the X.509 certificate byte array and registering it to the EBSI document by invoking the smart contract method.

For demonstration and testing purposes only, when clicking the "Example: Domibus blue" button (or "Example: Domibus red" button), the UI Dialog shows the C2 (or C3) filled with pre-configured (party identifier and certificate) data.



Certificate data

Certificate subject: C=BE, O=eDelivery, CN=blue_gw

Certificate issuer: C=BE, O=eDelivery, CN=blue_gw

Valid from: 9/14/2017 09:27

Valid to: 12/1/2025 08:27

Participant data

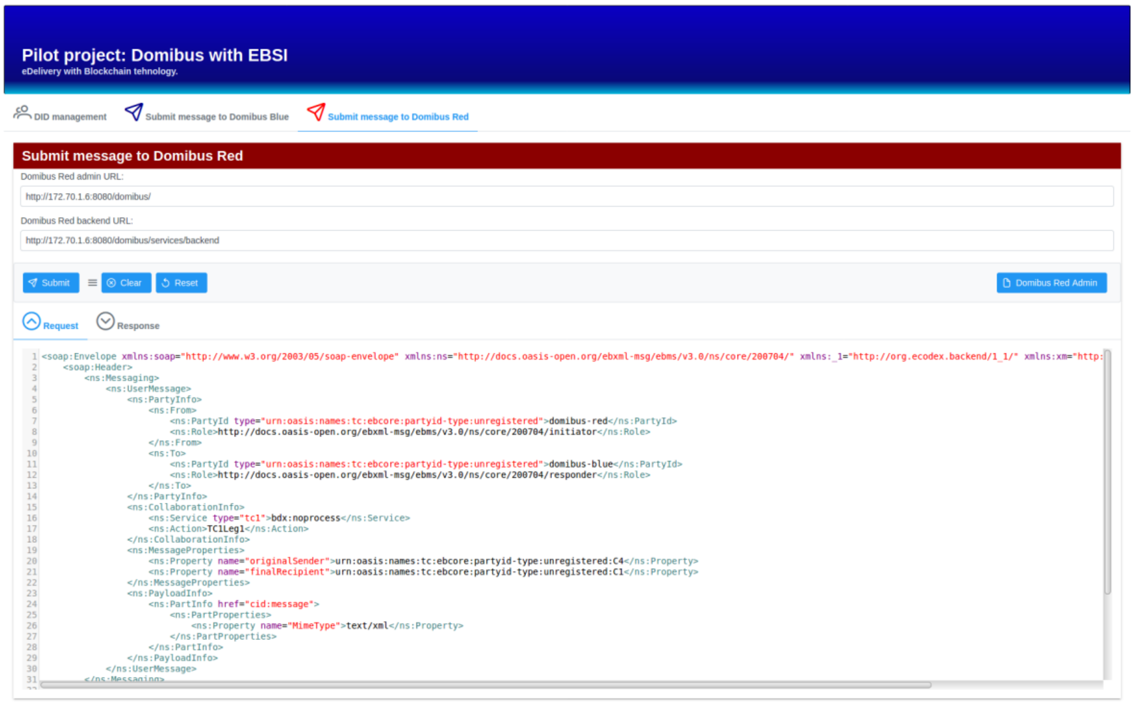
Party Schema *: urn:oasis:names:tc:ebcore:partyid-type:unregistered

Party Id: domibus-blue

Valid for number of days: 1,679

GeneratedHash: 0x5a23f323e7a20e42cf5b3b87cfba13b4c47bdf95cec4db927ae28dd92254dd08

The Demo application also contains simple features to submit messages to C2 (Domibus Blue) or C3 (Domibus Red), emulating the backend system (C1 or C4, respectively). Messages submitted to C2 are sent by the latter to C3 and vice-versa.


Pilot project: Domibus with EBSI
eDelivery with Blockchain tehnology.
DID management
Submit message to Domibus Blue
Submit message to Domibus Red
Submit
Clear
Reset
Domibus Red Admin
Request
Response

Submit message to Domibus Red

Domibus Red admin URL: http://172.70.1.6:8080/domibus/

Domibus Red backend URL: http://172.70.1.6:8080/domibus/services/backend

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:ns="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/" xmlns:_1="http://org.ecodex.backend/1_1/" xmlns:xm="http:
    <soap:Header>
        <ns:Messaging>
            <ns:UserMessage>
                <ns:PartyInfo>
                    <ns:From>
                        <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-red</ns:PartyId>
                        <ns:Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator</ns:Role>
                    </ns:From>
                    <ns:To>
                        <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-blue</ns:PartyId>
                        <ns:Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/responder</ns:Role>
                    </ns:To>
                </ns:PartyInfo>
                <ns:CollaborationInfo>
                    <ns:Service type="tc1">bdx:noprocess</ns:Service>
                    <ns:Action>TC1Leg1</ns:Action>
                </ns:CollaborationInfo>
                <ns:MessageProperties>
                    <ns:Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-type:unregistered:C4</ns:Property>
                    <ns:Property name="finalRecipient">urn:oasis:names:tc:ebcore:partyid-type:unregistered:C1</ns:Property>
                </ns:MessageProperties>
                <ns:PayloadInfo>
                    <ns:PartInfo href="cid:message">
                        <ns:PartProperties>
                            <ns:Property name="MimeType">text/xml</ns:Property>
                        </ns:PartProperties>
                    </ns:PartInfo>
                </ns:PayloadInfo>
            </ns:UserMessage>
        </ns:Messaging>
```

3 Contact information

For any questions regarding this document please contact EC-EDELIVERY-SUPPORT@ec.europa.eu.