dcsa

Digital Signatures for Transport Documents

Implementation Guide

Ensuring integrity and non-repudiation in the eBL ecosystem.

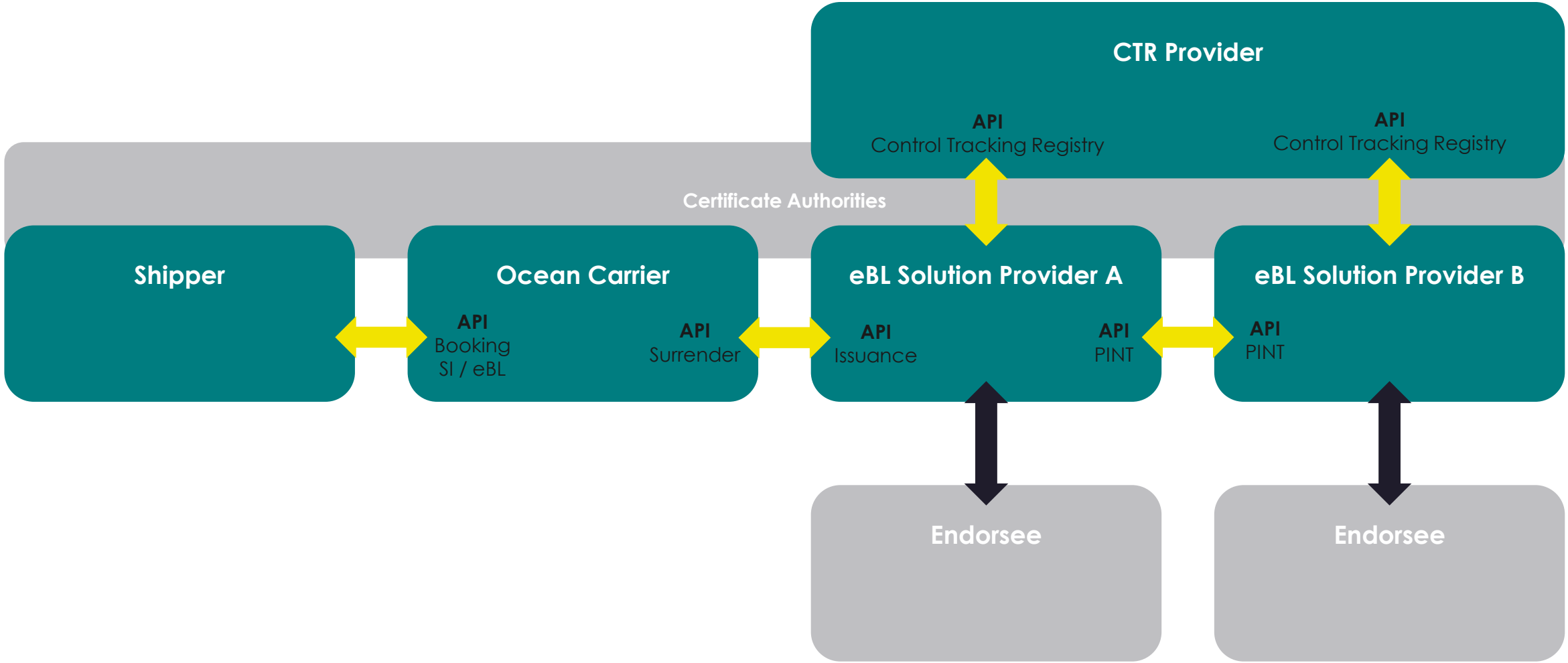
Version 03

01 October 2024

01

Overview and Scope of DCSA

End-to-End System Overview of API Providers



Prerequisites for a secure API ecosystem.

DCSA does not play a role in these underlying activities.

Customer onboarding. Parties arrange commercial contracts to provide service, perform due diligence processes etc.

Credentials. Parties issue credentials to gain access to each others' APIs.

TLS. Parties obtain digital certificates from certificate authorities to provide secure communications using TLS (Transport Layer Security). These are often referred to as "SSL Certificates".

Authentication & Authorization. Parties secure API endpoints, ensuring that authentication and authorization are correctly implemented.

Areas within DCSA API standardisation scope.

Shipper ←→ Ocean Carrier
Booking, Shipping Instructions, Transport Document

Ocean Carrier → eBL Solution Provider
Transport Document Issuance

eBL Solution Provider ←→ eBL Solution Provider
PINT (Platform Interoperability)

eBL Solution Provider → Control Tracking Registry
CTR (Control Tracking Registry)

eBL Solution Provider → Ocean Carrier
Transport Document Surrender

Areas outside DCSA API standardisation scope.

The **digital certificate life cycle** (request, issue, exchange between counterparties, revoke, expire, renew) is facilitated by **certificate authorities** (CAs). This is a prerequisite for a secure API ecosystem, due to the need to provide **TLS**.

CAs inform parties about **certificate revocation** using **OCSP**.

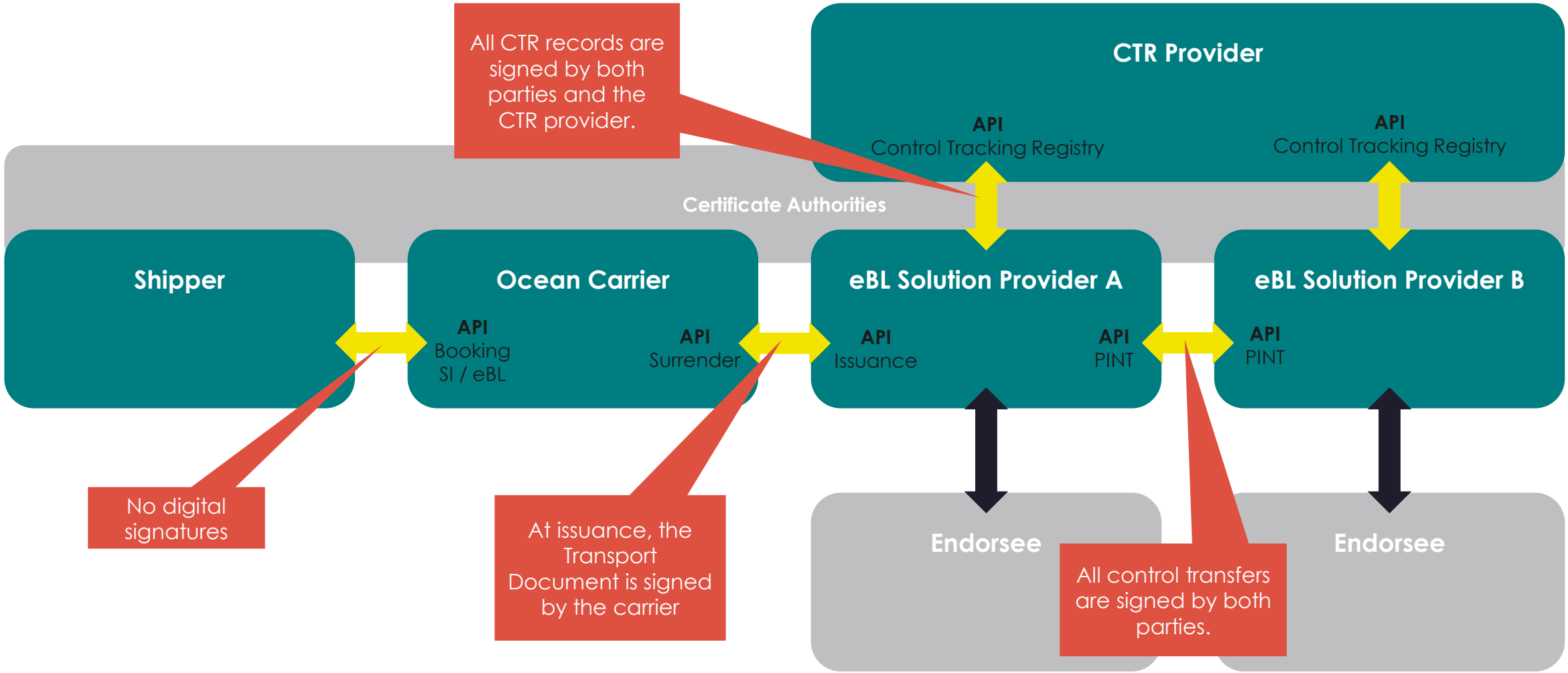
Keys are only valid when backed by valid (not expired and not revoked) digital certificates, otherwise they do not provide clear **non-repudiation**.

All parties will need a **public key infrastructure (PKI)** solution for managing **cryptographic keys**. This is a prerequisite for a secure API ecosystem, due to the need to provide **TLS**.

02

Digital Signature Positioning in the End-to-end System

Digital Signatures in Context



03

Obtaining and Passing Digital Certificates

Carrier / CA / eBL Solution Provider:

Obtaining and passing digital certificates

The purpose of this process is to provide the eBL solution provider with the means to validate digitally signed requests for transport document issuance.

To provide a valid key to the eBL solution provider, the ocean carrier must obtain a **digital certificate** from a **certificate authority**. This digital certificate is passed to the eBL solution provider at onboarding, and then periodically refreshed.

A **digital certificate** is a means to provide a **public key** strongly bound to an **identity,** where this binding is backed by **trust** in a **certificate authority**. It is the way that **non-repudiation** is implemented in this ecosystem (a party cannot reasonably deny that they are the owner of a key).

When the eBL solution provider receives the digital certificate, they validate it against the known public key of the certificate authority. This allows them to determine that it is a **genuine certificate**.

Carrier / CA / eBL Solution Provider:

Obtaining and passing digital certificates

Step 1.
Carrier generates a key pair with sufficient cryptographic strength

Step 2.
Carrier generates a Certificate Signing Request(CSR) that includes the public key

Step 3.
Carrier sends the Certificate Signing Request to a Certificate Authority

Step 4.
Certificate Authority Issues a certificate to the carrier that includes the public key

Step 5.
Carrier provides the certificate to the eBL solution provider. They agree on how to identify the key

dcsa

Carrier / CA / eBL Solution Provider:

Obtaining digital certificates

Every party in a secure API ecosystem must already be able to generate key pairs and obtain digital certificates in order to provide Transport Layer Security **TLS**.

In the case of digital certificates, **Extended Validation (EV)** or **Organization Validated (EV)** certificates can be useful. This is because digital signatures are used to provide **non-repudiation.** Additional validation strengthens the binding of the **identity** to the certificate.

The use of EV or OV certificates is recommended by DCSA for digital signatures, but it is not a conformance requirement. This is a question to be decided between the parties involved.

It is appreciated that EV or OV certificates are falling out of favour for TLS. However, this is a different use case. TLS does not provide non-repudiation of messages.

Carrier / CA / eBL Solution Provider:

Passing digital certificates between parties

The method by which parties pass digital certificates is outside DCSA standardisation scope.

It is expected that initial certificate passing would be part of the **onboarding process** between ocean carriers and eBL solution providers. During this process it is necessary to exchange other security-related information such as credentials for API access. It will be necessary to periodically renew certificates using similar processes.

When the certificates are exchanged, the parties must agree on how to identify the key pair in question. This is the **key id (kid)** used in the digital signature. A common approach (e.g. in OpenID Connect) is to generate a JWK thumbprint of the public key. Therefore, this is a recommendation of DCSA, but it is not a conformance requirement.

04

Signing a Transport Document for Issuance

Carrier / eBL Solution Provider:

Signing a transport document for issuance.

Process steps, performed by the ocean carrier:

i) Generate canonical JSON serialization for two properties of the issuance request.

ii) Calculate checksums for the three properties of the issuance request.

iii) Put the checksums into an Issuance Manifest JSON object.

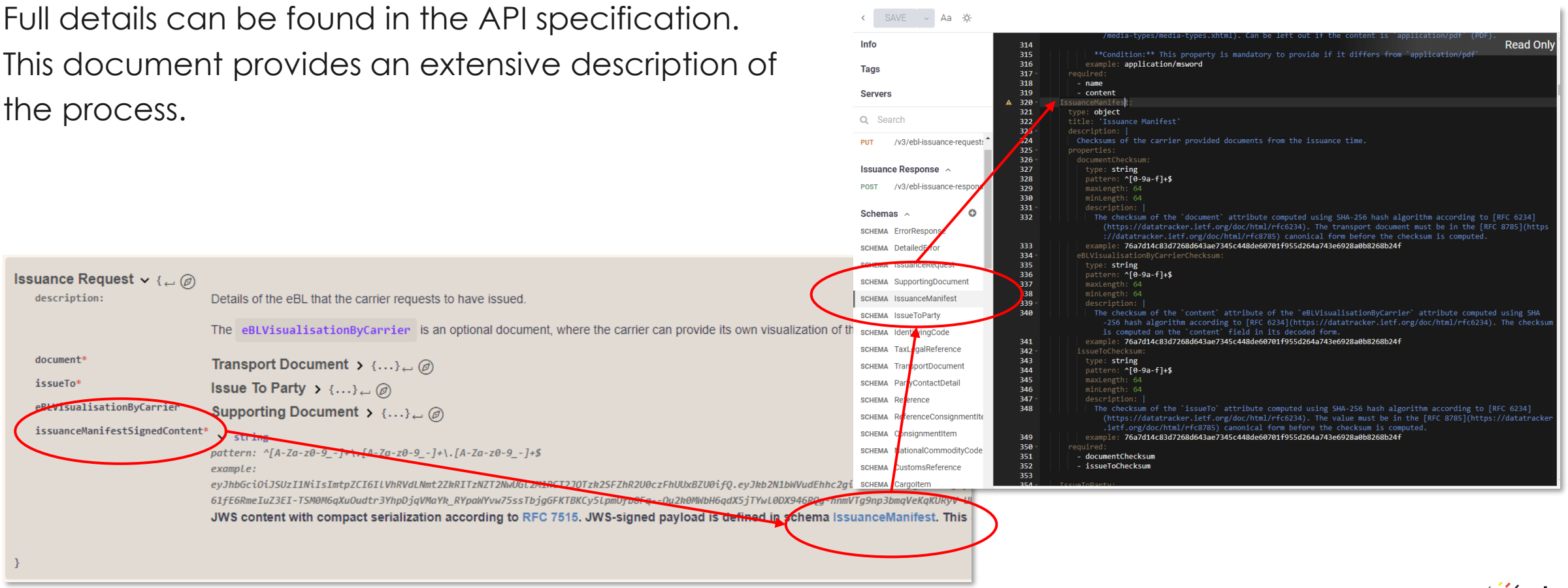
iv) Generate a JWS using this JSON object and carrier private key.

v) Put the JWS into the Issuance Request sent via the API to the eBL Solution Provider.

Carrier / eBL Solution Provider:

Signing a transport document for issuance.

Full details can be found in the API specification.
This document provides an extensive description of the process.



Carrier / eBL Solution Provider:

i) Generate canonical JSON serialization of the two properties.


```
{
  "transportDocumentReference": "HHL71800000",
  "shippingInstructionsReference": "e0559d83-00e2-438e-afd9-fdd610c1a008",
  "transportDocumentStatus": "DRAFT",
  "transportDocumentTypeCode": "SWB",
  "isShippedOnBoardType": true,
  "freightPaymentTermCode": "PRE",
  "isElectronic": true,
  "isToOrder": false,
  "numberOfCopiesWithCharges": 2,
  "numberOfCopiesWithoutCharges": 2,
  "numberOfOriginalsWithCharges": 1,
  "numberOfOriginalsWithoutCharges": 1,
  "displayedNameForPlaceOfReceipt": [
    "Strawinskylaan 4117"
  ],
  "displayedNameForPortOfLoad": [
    "Strawinskylaan 4117"
  ],
  "displayedNameForPortOfDischarge": [
    "Strawinskylaan 4117"
  ],
  "displayedNameForPlaceOfDelivery": [
    "Strawinskylaan 4117"
  ],
  "shippedOnBoardDate": "2020-12-12",
  "displayedShippedOnBoardReceivedForShipment": "Received for Shipment CMA
  "termsAndConditions": "Any reference in...",
  "receiptTypeAtOrigin": "CY",
  "deliveryTypeAtDestination": "CY",
  "cargoMovementTypeAtOrigin": "FCL",
  "cargoMovementTypeAtDestination": "FCL",
  "issueDate": "2020-12-12",
  "receivedForShipmentDate": "2020-12-12",
  "serviceContractReference": "HHL51800000",
  "contractQuotationReference": "HHL1401",
  "declaredValue": 1231.1,
  "declaredValueCurrency": "DKK",
  "carrierCode": "MMCU",
  "carrierCodeListProvider": "NMFTA",
  "carrierClauses": [
    "It is not allowed to..."
  ],
  "numberOfRiderPages": 2,
  "transports": {
    "plannedArrivalDate": "2024-06-07",
    "plannedDepartureDate": "2024-06-03",
    "preCarriageBy": "RAIL",
    "onCarriageBy": "TRUCK",
```


Status:	Informational
More info:	Errata exist | Datatracker | IPR | Info page
Stream:	Independent Submission
RFC:	8785
Category:	Informational
Published:	June 2020
ISSN:	2070-1721
Authors:	A. Rundgren *Independent* · B. Jordan *Broadcom* · S. Erdtman *Spotify AB*

RFC 8785
JSON Canonicalization Scheme (JCS)

Abstract

Cryptographic operations like hashing and signing need the data to be expressed in an invariant format so that the operations are reliably repeatable. One way to address this is to create a canonical representation of the data. Canonicalization also permits data to be exchanged in its original form on the "wire" while cryptographic operations performed on the canonicalized counterpart of the data in the producer and consumer endpoints generate consistent results.

This document describes the JSON Canonicalization Scheme (JCS). This specification defines how to create a canonical representation of JSON data by building on the strict serialization methods for JSON primitives defined by ECMAScript, constraining JSON data to the Internet JSON (I-JSON) subset, and by using deterministic property sorting.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8785.


```
{"cargoMovementTypeAtDestination":"FCL","cargoMovementTypeAtOrigin":"FCL","carrierClau
"value":2.4},"cargoGrossWeight":{"unit":"KGM","value":2400},"cargoNetVolume":{"unit":"
"competentAuthorityApproval":"{Name and reference...}","emergencyContactDetails":{"con
"isCompetentAuthorityApprovalRequired":false,"isEmptyUncleanedResidue":false,"isExcept
"value":2.4},"netWeight":{"unit":"KGM","value":2.4},"packingGroup":3,"properShippingNa
"referenceNumber":"12234"},"endOfHoldingTime":"2021-09-03","fumigationDateTime":"2024-
"isReportableQuantity":false,"isSalvagePackings":false,"isWaste":false,"limits":{"SADT
"subsidiaryRisk1":"1.2","subsidiaryRisk2":"1.2","technicalName":"xylene and benzene"}]
["1515","2106","2507","2512"]}],"references":[{"type":"CR","values":["HHL00103004"]}],
"displayedNameForPortOfDischarge":["Strawinskylaan 4117"],"displayedNameForPortOfLoad"
"streetNumber":"100"},"partyContactDetails":[{"name":"Henrik","phone":"+45 51801234"},
"codeListProvider":"W3C","partyCode":"MSK"}],"partyContactDetails":[{"name":"Henrik","
stCode":"1047 HM","stateRegion":"North Holland","street":"Ruijggoordweg","streetNum
Box":"123","UNLocationCode":"NLAMS","city":"Amsterdam","countryCode":"NL","floor":
ype":"PAN","value":"AAAAA0000A"}]},"notifyParties":[{"address":{"POBox":"123","UNLoc
"phone":"+45 51801234"}],"partyName":"IKEA Denmark","reference":"HHL007","taxLegalRefe
"codeListProvider":"W3C","partyCode":"MSK"}],"partyContactDetails":[{"name":"Henrik","
Holland","street":"Ruijggoordweg","streetNumber":"100"},"displayedAddress":["Strawinsk
"typeOfPerson":"NATURAL_PERSON"}},"freightPaymentTermCode":"PRE","invoicePayableAt":{"
"placeOfIssue":{"UNLocationCode":"NLAMS","locationName":"Port of Amsterdam"},"receiptT
"transportDocumentStatus":"DRAFT","transportDocumentTypeCode":"SWB","transports":{"onC
"portOfLoading":{"UNLocationCode":"DEHAM","locationName":"Hamburg"},"preCarriageBy":"R
"isDrainholesOpen":true,"isVentilationOpen":true,"o2Setpoint":25,"temperatureSetpoint"
"source":"CUS"}],"shippingMarks":["Made in China"]}]}
```

Carrier / eBL Solution Provider:

i) Generate canonical JSON serialization of the two properties.

Ensures that exactly the **same document** produces exactly the **same checksum**.

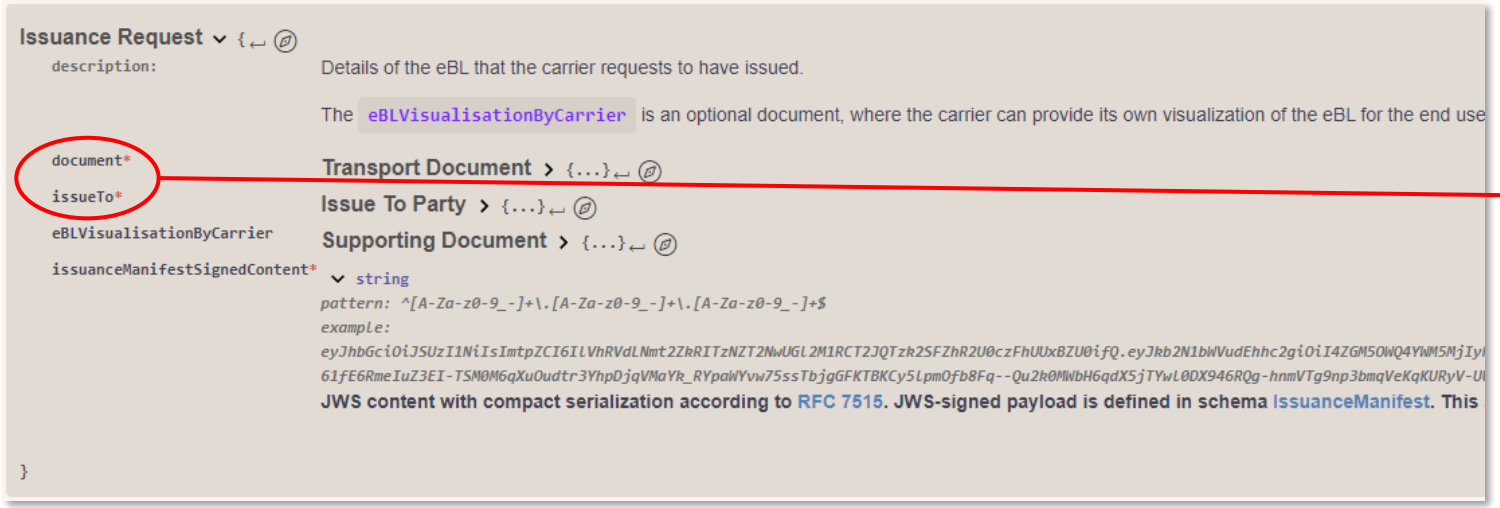
Recognised as a best practice when performing cryptographic operations.

SHOULD be implemented using a library.

Candidate components → https://github.com/cyberphone/json-canonicalization
DCSA does not guarantee the quality of these free open-source components and has not exhaustively tested them.

Carrier / eBL Solution Provider:

i) Generate canonical JSON serialization of the two properties.



Two of the three JSON objects in the Issuance Request must be serialized in canonicalized form:

- **document**
- **issueTo**

Carrier / eBL Solution Provider:

i) Generate canonical JSON serialization of the two properties.

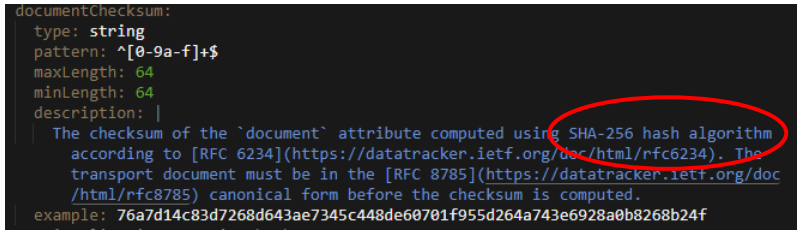
	JSON	Canonicalized JSON
document		
issueTo		

Each of two properties corresponds to a canonicalized JSON object.

dcsa

Carrier / eBL Solution Provider:

ii) Calculate checksums for the three parts of the issuance request.



Checksums are generated in order to have a unique short string that corresponds to each of the three properties.

The three properties are each hashed to produce three SHA-256 checksums. This MUST be performed using libraries such as **`java.security.MessageDigest`**.

The principle of signing a hash is widely accepted as a best practice when working with digital signatures.

Carrier / eBL Solution Provider:

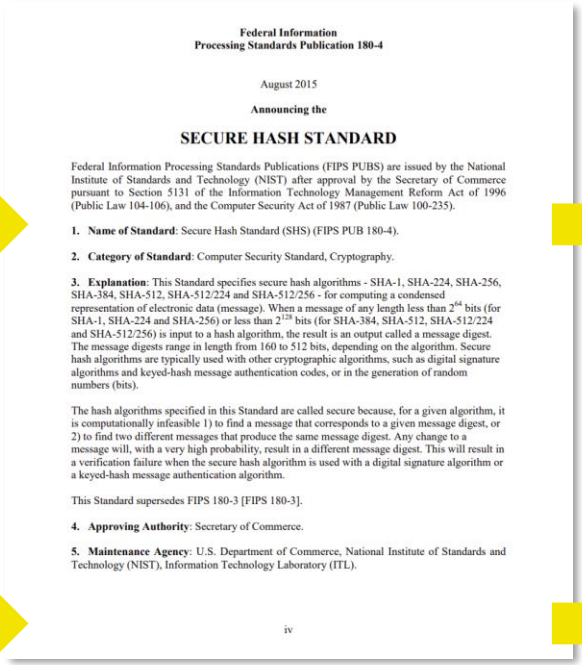
ii) Calculate checksums for the two JSON properties of the issuance request.

Canonicalized JSON document


```
{"cargoMovementTypeAtDestination":"FCL","cargoMovementTypeAtOrigin":"FCL","carrierClau
"value":2.4},"cargoGrossWeight":{"unit":"KGM","value":2400},"cargoNetVolume":{"unit":"
"competentAuthorityApproval":"{Name and reference...}","emergencyContactDetails":{"con
"isCompetentAuthorityApprovalRequired":false,"isEmptyUncleanedResidue":false,"isExcept
"value":2.4},"netWeight":{"unit":"KGM","value":2.4},"packingGroup":3,"properShippingNa
"referenceNumber":"12234"},"endOfHoldingTime":"2021-09-03","fumigationDateTime":"2024-
"isReportableQuantity":false,"isSalvagePackings":false,"isWaste":false,"limits":{"SADT
"subsidiaryRisk1":"1.2","subsidiaryRisk2":"1.2","technicalName":"xylene and benzene"}]
["1515","2106","2507","2512"]}],"references":[{"type":"CR","values":["HHL00103004"]}],
"displayedNameForPortOfDischarge":["Strawinskylaan 4117"],"displayedNameForPortOfLoad"
"streetNumber":"100"},"partyContactDetails":[{"name":"Henrik","phone":"+45 51801234"},
"codeListProvider":"W3C","partyCode":"MSK"}],"partyContactDetails":[{"name":"Hen
"postCode":"1047 HM","stateRegion":"North Holland","street":"Ruijggoordweg","str
{"POBox":"123","UNLocationCode":"NLAMS","city":"A                    NL",
"type":"PAN","value":"AAAAA0000A"}]},"notifyParti                    123","UNLoc
"phone":"+45 51801234"}],"partyName":"IKEA Denmark","reference":"HHL007","taxLegalRefe
"codeListProvider":"W3C","partyCode":"MSK"}],"partyContactDetails":[{"name":"Henrik","
Holland","street":"Ruijggoordweg","streetNumber":"100"},"displayedAddress":["Strawinsk
"typeOfPerson":"NATURAL_PERSON"}},"freightPaymentTermCode":"PRE","invoicePayableAt":{"
"placeOfIssue":{"UNLocationCode":"NLAMS","locationName":"Port of Amsterdam"},"receiptT
"transportDocumentStatus":"DRAFT","transportDocumentTypeCode":"SWB","transports":{"onC
"portOfLoading":{"UNLocationCode":"DEHAM","locationName":"Hamburg"},"preCarriageBy":"R
"isDrainholesOpen":true,"isVentilationOpen":true,"o2Setpoint":25,"temperatureSetpoint"
"source":"CUS"}],"shippingMarks":["Made in China"]}]}
```


Canonicalized JSON issueTo


```
{"identifyingCodes":[{"codeListName":"DID","codeListProvider":"W3C","partyCode":"MSK"}]
```

Federal Information
Processing Standards Publication 180-4

August 2015

Announcing the

SECURE HASH STANDARD

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106), and the Computer Security Act of 1987 (Public Law 100-235).

1. **Name of Standard**: Secure Hash Standard (SHS) (FIPS PUB 180-4).

2. **Category of Standard**: Computer Security Standard, Cryptography.

3. **Explanation**: This Standard specifies secure hash algorithms - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 - for computing a condensed representation of electronic data (message). When a message of any length less than 2^{64} bits (for SHA-1, SHA-224 and SHA-256) or less than 2^{128} bits (for SHA-384, SHA-512, SHA-512/224 and SHA-512/256) is input to a hash algorithm, the result is an output called a message digest. The message digests range in length from 160 to 512 bits, depending on the algorithm. Secure hash algorithms are typically used with other cryptographic algorithms, such as digital signature algorithms and keyed-hash message authentication codes, or in the generation of random numbers (bits).

The hash algorithms specified in this Standard are called secure because, for a given algorithm, it is computationally infeasible 1) to find a message that corresponds to a given message digest, or 2) to find two different messages that produce the same message digest. Any change to a message will, with a very high probability, result in a different message digest. This will result in a verification failure when the secure hash algorithm is used with a digital signature algorithm or a keyed-hash message authentication algorithm.

This Standard supersedes FIPS 180-3 [FIPS 180-3].

4. **Approving Authority**: Secretary of Commerce.

5. **Maintenance Agency**: U.S. Department of Commerce, National Institute of Standards and Technology (NIST), Information Technology Laboratory (ITL).

iv

SHA-256 Hash of document

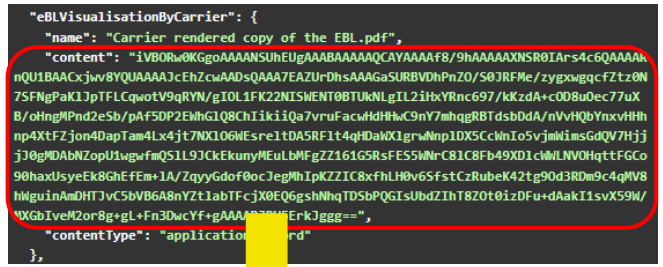
```
0a71bc86d65ba0ea59dbc117a2861912789bf4e8a9cd5c7e3dc09d513b21d17a
```

SHA-256 Hash of issueTo

```
aabdf0e26f7097b2c2615fbad0003bf21d8c24d5b0cd3b0a27d06da544db4159
```

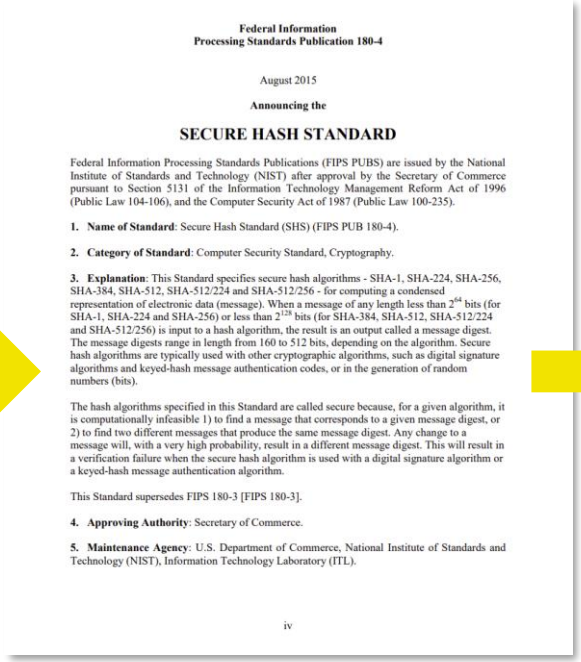
Carrier / eBL Solution Provider:

ii) Calculate checksum for the content of the eBL visualisation of the issuance request.



Only process the content property of the eBLVisualisationByCarrier object. Do not include the other properties, which are only metadata.

Decode from BASE64 to binary.



Federal Information
Processing Standards Publication 180-4

August 2015

Announcing the

SECURE HASH STANDARD

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106), and the Computer Security Act of 1987 (Public Law 100-235).

1. **Name of Standard**: Secure Hash Standard (SHS) (FIPS PUB 180-4).

2. **Category of Standard**: Computer Security Standard, Cryptography.

3. **Explanation**: This Standard specifies secure hash algorithms - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 - for computing a condensed representation of electronic data (message). When a message of any length less than 2^{64} bits (for SHA-1, SHA-224 and SHA-256) or less than 2^{128} bits (for SHA-384, SHA-512, SHA-512/224 and SHA-512/256) is input to a hash algorithm, the result is an output called a message digest. The message digests range in length from 160 to 512 bits, depending on the algorithm. Secure hash algorithms are typically used with other cryptographic algorithms, such as digital signature algorithms and keyed-hash message authentication codes, or in the generation of random numbers (bits).

The hash algorithms specified in this Standard are called secure because, for a given algorithm, it is computationally infeasible 1) to find a message that corresponds to a given message digest, or 2) to find two different messages that produce the same message digest. Any change to a message will, with a very high probability, result in a different message digest. This will result in a verification failure when the secure hash algorithm is used with a digital signature algorithm or a keyed-hash message authentication algorithm.

This Standard supersedes FIPS 180-3 [FIPS 180-3].

4. **Approving Authority**: Secretary of Commerce.

5. **Maintenance Agency**: U.S. Department of Commerce, National Institute of Standards and Technology (NIST), Information Technology Laboratory (ITL).

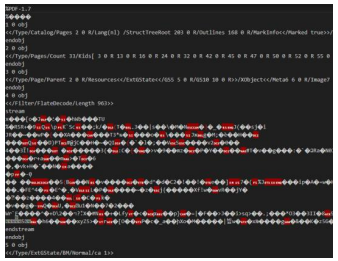
iv

SHA-256 Hash of unencoded content

```
d4ac1ca327cea4f323e5a971f8e091690ed86609ac039b294c7bd1722d931fc0
```

Carrier / eBL Solution Provider:

ii) Calculate checksums for the three parts of the issuance request.

	Canonicalized JSON	SHA-256 Checksum
document		0a71bc86d65ba0ea59dbc117a2861912789bf4e8a9cd5c7e3dc09d513b21d17a
issueTo		aabdf0e26f7097b2c2615fbad0003bf21d8c24d5b0cd3b0a27d06da544db4159
eBLVisualisationByCarrier	Unencoded Content 	d4ac1ca327cea4f323e5a971f8e091690ed86609ac039b294c7bd1722d931fc0

Each one of the three properties corresponds to a SHA-256 checksum.

Carrier / eBL Solution Provider:

iii) Put the checksums into an Issuance Manifest JSON object.

document

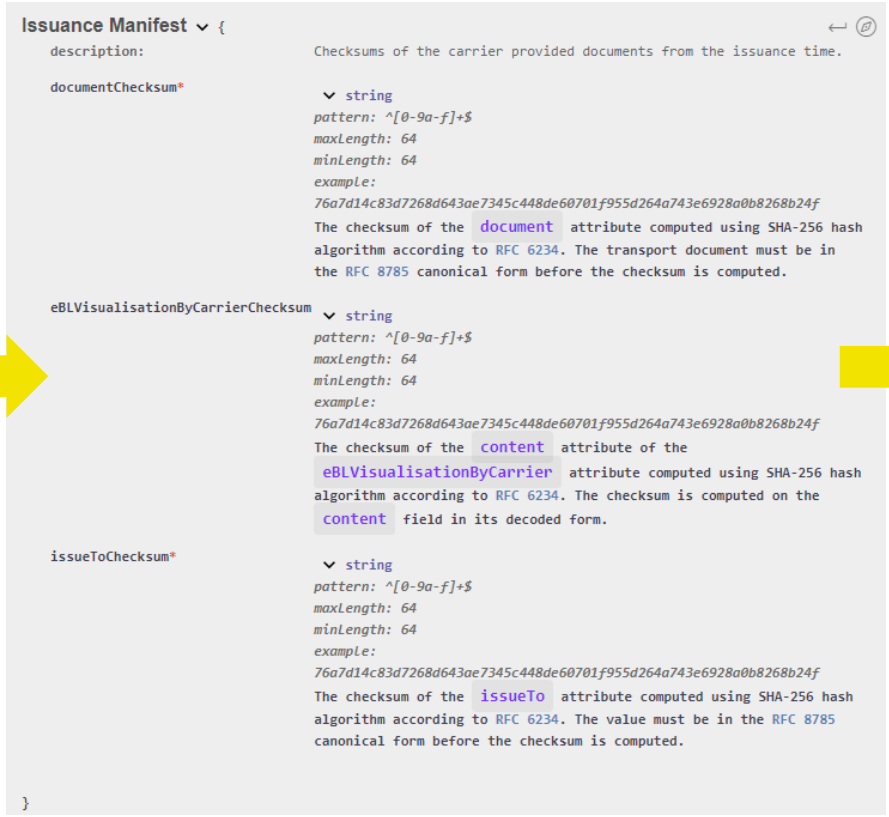
`0a71bc86d65ba0ea59dbc117a2861912789bf4e8a9cd5c7e3dc09d513b21d17a`

eBLVisualisationByCarrier

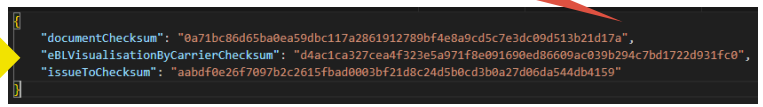
`d4ac1ca327cea4f323e5a971f8e091690ed86609ac039b294c7bd1722d931fc0`

issueTo

`aabdf0e26f7097b2c2615fbad0003bf21d8c24d5b0cd3b0a27d06da544db4159`



The Issuance Manifest JSON object.



Carrier / eBL Solution Provider:

iv) Generate a JWS using this JSON object and carrier private key.

Once the Issuance Manifest object is created, it is digitally signed.

The digital signature is in the form of a **JSON Web Signature (JWS)** with **compact serialization**.

The signature is generated using the **private key** (pair of the public key in the digital certificate) and **algorithm**, the **Issuance Manifest object**, and the **key id** agreed when the certificate was provided to the eBL solution provider. This input is fed into a library to produce the JWS.

The signing process MUST be performed using a library. For Java development a possible candidate is Nimbus JOSE + JWT → https://github.com/Connect2id/Nimbus-JWT

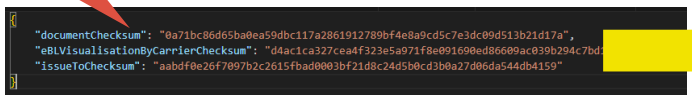
Carrier / eBL Solution Provider:

iv) Generate a JWS using this JSON object and carrier private key.

The private key that is paired with the public key in the digital certificate provided by the carrier to the eBL solution provider.

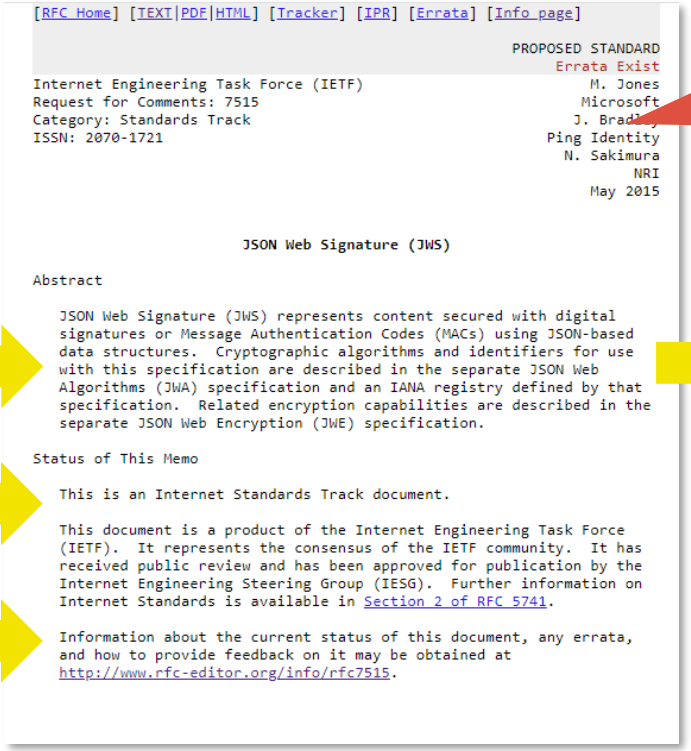
Private Key

The Issuance Manifest JSON object.



Key id agreed with the eBL solution provider during onboarding. Typically the "thumbprint" of the public key.

Key id (kid) that is Known to the eBL Solution Provider



[RFC Home] [TEXT|PDF|HTML] [Tracker] [IPR] [Errata] [Info page]

PROPOSED STANDARD
Errata Exist

Internet Engineering Task Force (IETF) — M. Jones, Microsoft
Request for Comments: 7515 — J. Bradley, Ping Identity
Category: Standards Track — N. Sakimura, NRI
ISSN: 2070-1721 — May 2015

JSON Web Signature (JWS)

Abstract

JSON Web Signature (JWS) represents content secured with digital signatures or Message Authentication Codes (MACs) using JSON-based data structures. Cryptographic algorithms and identifiers for use with this specification are described in the separate JSON Web Algorithms (JWA) specification and an IANA registry defined by that specification. Related encryption capabilities are described in the separate JSON Web Encryption (JWE) specification.

Status of This Memo

This is an Internet Standards Track document.

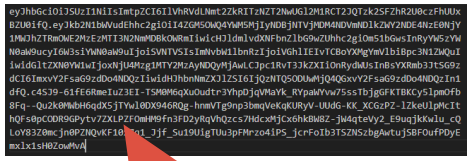
This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7515.

Use a **cryptographic library** recommended by your organisation to implement JWS.

Do not "roll your own" implementation.

JWS
JSON Web Signature



JWS with **compact serialization**.

Carrier / eBL Solution Provider:

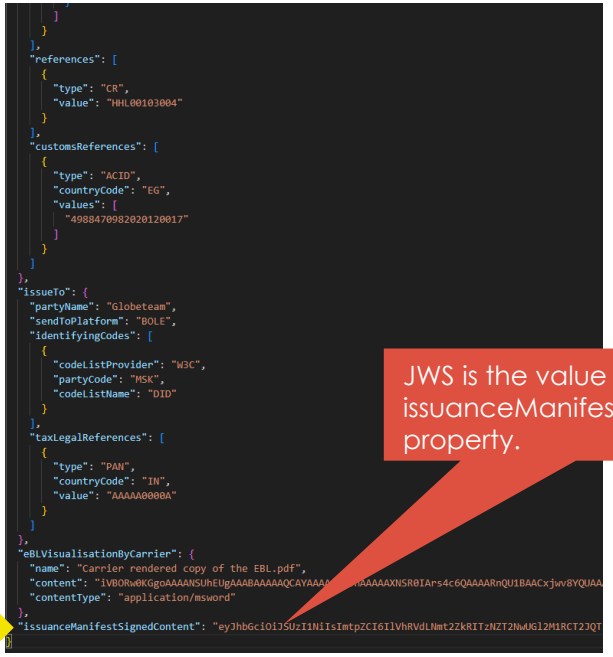
v) Put the JWS into the Issuance Request sent via the API to the eBL Solution Provider.

The JWS contains the following, in encoded form –

- Header – specifies the **signature algorithm** to use and the **key id.** Not standardised by DCSA but RS256 is typical.
- The JSON object with the three checksums - the **IssuanceManifest.**
- The **signature** generated using the private key that corresponds to the key id.

It does **NOT** contain any cryptographic keys.

JWS
JSON Web Signature



JWS is the value of the issuanceManifestSignedContent property.

05

Validation of the digital signature for eBL issuance.

Carrier / eBL Solution Provider:

Validation of the digital signature for eBL issuance.

Process steps, performed by the eBL solution provider:

i) Decode the JWS.

ii) Match the key id to an existing digital certificate.

iii) Check that the digital certificate is from the correct party (API authentication matches with identity in the certificate) and that the certificate is valid (not expired and not revoked).

iv) Check that the signature matches with the public key in the certificate.

v) Validate that the checksums in the Issuance Manifest in the JWS match with checksums calculated from the properties of the received API request.

06

Q&A.

Q&A

Q. Does DCSA have a hashing algorithm proposal?

A. Yes, the algorithms for hashing are specified in this document. SHA-256 is used to create the properties of the Issuance Manifest object.

Q. Will DCSA standardize the algorithms to be used across all ocean carriers?

A. The hashing algorithm to generate the properties of the Issuance Manifest object is standardised. The algorithms to generate the public / private key pair, and to generate the JWS signature are not standardised.

Q. Is DCSA planning to standardise the kid value and signature strategy, or will each carrier be able to implement its own?

A. DCSA is not planning to standardise the kid value or the digital signature algorithm.

Q&A

Q. How are keys revoked or expired?

A. Public keys are made available in digital certificates. These have an expiry date. The key is only valid until this expiry date. Revocation occurs via the certificate authority, which can publicise this using the **OCSP protocol**. It is expected that parties would also directly inform each other in the event of a breach that invalidated keys; this process occurs independently of DCSA.

Q. Can an ocean carrier have more than one active certificate, for example close to the expiry date of a certificate?

A. This is not determined by DCSA, but there is no fundamental reason that this should not be allowed. It is determined by the onboarding / certificate passing mechanism between the ocean carrier and the eBL solution provider.

Q&A

Q. Why is there no standard for the key id, key generation algorithm or signature algorithm?

A. DCSA is active in a large ecosystem. The participants have their own corporate security teams who set policy in this area. DCSA does not wish to interfere with this. For example, perhaps one party has a policy to use RSA-2048, whilst another party insists on ECC-256. The types of keys in turn determine the algorithms for the signature.

Q. Why is there not a standard for the certificate authority?

A. Different parties will have different policy on acceptable CAs. Some parties may even wish to use self-signed certificates from a corporate CA (not recommended, but possible). The API could even work without certificates (unclear how **non-repudiation** would be achieved). The standard makes all options available. It is up to each pair of ocean carrier / eBL solution provider to determine what is acceptable.