ETSI GS CIM 019 V1.2.1 (2025-08)

GROUP SPECIFICATION

Context Information Management (CIM); handling of provenance information in NGSI-LD

Disclaimer

The present document has been produced and approved by the cross-cutting Context Information Management (CIM) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
RGS/CIM-0019ProvenanceV121

Keywords
data, digital signature, provenance, reliability, trust

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Content

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) cross-cutting Context Information Management (CIM).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

The present document specifies a mechanism for embedding W3C® Data Integrity digital signatures into NGSI-LD Entities.

Introduction

In the most generic scenario of a NGSI-LD [i.2] ecosystem, Entities from Context Providers are sent, through multiple Context Brokers, to Clients. In this scenario, the context information creator is the Context Provider, which is trusted by the Clients.

When an Entity typically contains multiple Attributes, it is important to guarantee that these values will not be altered through all its cycles, so that a Client, without further contact with the Context Provider, can be sure of the integrity.

The preferred solution in both literature and industry, to the data integrity problem, is the implementation of an end-to-end digital signature system.

1 Scope

The present document designs a solution to verify integrity and to precisely evaluate attribution and authenticity of NGSI-LD [i.2] Context Information, throughout its lifecycle. It defines technical means for enabling a chain of trust from Context Providers to Context Consumers, by embedding verifiable credentials into NGSI-LD documents, leveraging the W3C® Data Integrity methodology for digital signatures.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found in the ETSI docbox.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] W3C® Proposed Recommendation 11 March 2025: "RDF Dataset Canonicalization. A Standard RDF Dataset Canonicalization Algorithm".

[2] W3C® Recommendation 15 May 2025: "Verifiable Credential Data Integrity 1.0. Securing the Integrity of Verifiable Credential Data".

[3] IETF RFC 8785: "JSON Canonicalization Scheme (JCS)".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] ETSI GR CIM 007 (V1.1.1): "Context Information Management (CIM); Security and Privacy".

[i.2] ETSI GS CIM 009 (V1.9.1): "Context Information Management (CIM); NGSI-LD API".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

Atomic Entity: digitally signed NGSI-LD Entity with only one Attribute

Client: shorthand for NGSI-LD Context Consumer

Context Provider: NGSI-LD Context Source or NGSI-LD Context Producer

Derivation Process: process that transforms NGSI-LD Attributes into Sealed Attributes

NGSI-LD Attribute: reference to both an NGSI-LD Property and to an NGSI-LD Relationship

NGSI-LD Context Broker: architectural component that implements all the NGSI-LD interfaces

NGSI-LD Context Consumer: agent that uses the query and subscription functionality of NGSI-LD to retrieve context information

NGSI-LD Context Producer: agent that uses the NGSI-LD context provision and/or registration functionality to provide or announce the availability of its context information to an NGSI-LD Context Broker

NGSI-LD Context Source: source of context information which implements the NGSI-LD consumption and subscription (and possibly provision) interfaces defined by the present document

NGSI-LD Entity: informational representative of something that is supposed to exist in the real world, physically or conceptually

NGSI-LD Property: description instance which associates a main characteristic, i.e. an **NGSI-LD Value**, to either an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property and that uses the special *hasValue* property to define its target value

Reconstruction Process: opposite process of the Derivation Process

Sealed Attribute: NGSI-LD Attribute with "ngsildproof" sub-property

verification method: method that can be used together with a process to independently verify a proof

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

EdDSA	Edwards-curve Digital Signature Algorithm
JCS	JSON Canonicalization Scheme
JSON	JavaScript Object Notation
JSON-LD	JSON Linked Data
JWS	JSON Web Signature
JWS/CT	JSON Web Signature Clear Text
LD	Linked Data
NGSI	Next Generation Service Interfaces
NGSI-LD	NGSI Linked Data
RDF	Resource Description Format

4 Requirements

ETSI's Industry Specification Group on cross-cutting Context Information Management (ISG CIM) is tasked with specifying security structures for, among other things, ensuring data integrity and provenance of NGSI-LD Entities throughout the typical workflow from data sources, which are the creators of the context information/data, to a federation of Context Providers and NGSI-LD Context Brokers, to NGSI-LD Context Consumers.

Critical requirements for the integrity of data flowing within the NGSI-LD ecosystem, are (adapted from ETSI GR CIM 007 [i.1]):

- INT-1: The NGSI-LD Context Consumers should be able to determine that data integrity has been preserved.

This requirement dictates that data integrity for NGSI-LD Entities is recommended, but not mandatory, when assembling NGSI-LD Entities.

- INT-2: Verification of integrity shall be independent of syntactical re-ordering that may occur when serializing NGSI-LD Entities between peers.

- INT-3: Verification of integrity should be independent of the NGSI-LD serialization format itself, i.e. serialization formats should not strip verification information.

Information for verification of integrity is transported within NGSI-LD Entities, when they are serialized, as specified in clause 5. This requirement acknowledges that some of the output formats supported in NGSI-LD (e.g. the simplified representation, see clause 4.5.4 of ETSI GS CIM 009 [i.2]) may strip information that is vital to verification of integrity.

- INT-4: Preservation of data integrity shall not rely on the Clients trusting the relaying intermediate Context Providers or NGSI-LD Context Brokers, but solely the creators.

5 Specification

5.1 Fulfilling requirements

5.1.0 Foreword

For the sake of brevity and clarity, the terms Entity, Attribute, Property and Context Broker (or simply Broker, all of them capitalized) are used interchangeably with NGSI-LD Entity, NGSI-LD Attribute, NGSI-LD Property and NGSI-LD Context Broker, respectively.

The scenario used throughout the present document is the generation of Entities, by Context Providers, that are then sent, through multiple Context Brokers, to Clients. In this scenario, without loss of generality, the context information creator is the Context Provider, which is thusly trusted by the Clients.

In this scenario, where an Entity typically contains multiple Attributes, it is important to guarantee that these values will not be altered through all its cycles, so that a Client, without further contact with the Context Provider, can be sure of the integrity.

The preferred solution in both literature and industry, to the data integrity problem, is the implementation of a digital signature system.

A digest file of the Entity, cryptographically encoded with the signer private key, bound with it, guarantees the non-corruption of data (integrity) and the association to a specific private key (provenance).

Thus, using a digital signature system fulfils requirements INT-1 and INT-4 described in clause 4.

But cryptographic operations like hashing and signing depend on the fact that the target data does not change during serialization, transport, or parsing.

In the NGSI-LD ecosystem, every time a Context Broker receives Entities, it stores them in terms of the underlying Property Graph structure. On request, the Broker will serialize the Entity, generating its JSON-LD structure anew, in order to share it or send it to Clients. The new structure, though semantically equivalent, can be very different in terms of formatting and ordering of the underlying JSON key+value pairs.

The solution is the implementation of a canonicalization algorithm. Canonicalization is the process of transforming an input dataset to a normalized dataset. Any two input datasets that contain the same information, regardless of their arrangement, will be transformed into identical normalized dataset. This process is sometimes also called normalization.

ISG CIM is thus seeking to apply JSON canonicalization algorithms to serialized JSON-LD data, prior to digitally signing it, in order to fulfil requirement INT-2 described in clause 4.

Table 5.1.0-1 summarizes the status of various JSON canonicalization algorithms.

Table 5.1.0-1: Canonicalization algorithms

Specification Name	Group	Specification Status	Comments
JCS: JSON Canonicalization Scheme	IETF RFC 8785 [3]	Not an Internet Standards Track specification	- Builds on the strict serialization methods for JSON primitives defined by ECMAScript (https://en.wikipedia.org/wiki/ECMAScript), constraining JSON data to the Internet JSON (I-JSON) subset, and by using deterministic property sorting. - Good fit for JSON format. - Ordering of array elements is not managed by the algorithm, thus rearranging the elements within an array will invalidate any digital signature on the original array. - Simple. Possible implementation with JWS standard, through JWS/CT specification (not yet a published standard).
RDF Dataset Canonicalization	W3C® Credentials Community Group; W3C® RDF Dataset Canonicalization and Hash Working Group	It is now a W3C® Recommendation and it is on the W3C® Standards Track	- An algorithm for normalizing RDF datasets such that comparing the differences between sets of graphs, digitally sign them, or generate short identifiers for graphs via hashing algorithms is possible. - Good fit for JSON-LD format. - **Array elements can be reordered without invalidating signature**. - More complex. - Supported by W3C® Verifiable Credential Data Integrity specification [2].

The RDF Dataset Canonicalization [1] is based on Resource Description Framework (RDF), an abstract model with several serialization formats.

The implementation of the RDF Dataset Canonicalization inside the NGSI-LD ecosystem fulfils the INT-2 requirement described in clause 4.

5.1.1 Overview of W3C® Data Integrity specification

The W3C® Verifiable Credential Data Integrity specification [2] describes mechanisms for ensuring the authenticity and integrity of structured digital documents using cryptography.

In order to produce a verifiable digital proof, it supports the usage of different canonicalization algorithms, so that both detection of tampering with the integrity of data and, at the same time, re-ordering or the structured document, is possible.

Following the W3C® Data Integrity specification [2], it is possible to create a data integrity "`proof`" element, which is a set of attributes that represent a digital proof and all parameters required to verify it.

A data integrity proof contains, at least, the following attributes:

- `type`: the fixed string "DataIntegrityProof";
- `cryptosuite`: which indicates the specific type of digital signature used. It is defined as "a specified set of cryptographic primitives bundled together into a cryptographic suite for the purposes of safety and convenience, by cryptographers for developers. A proof type typically consists of a canonicalization algorithm, a message digest algorithm, and a specific corresponding proof algorithm";
- `proofPurpose`: a parameter that ensures that the digital proof is used for the reason it was created for;
- `verificationMethod`: a set of parameters required to independently verify the proof;
- `created`: date and time of the proof generation;
- `proofValue`: the value of the encoded hash.

The verification process is possible through the access to a so-called controller document, a set of data that specify the relationship between a controller, the entity who can change the controller document, and other data sets such as a public cryptographic key.

Whoever wants to verify the data integrity proof shall ensure that a verification method is bound to a specific controller, by going from the verification method attribute in the proof to the controller document, ensuring that this also contains the same verification method and the same proof purpose.

The following signature suites (i.e. verification methods and digital signature types) are contemplated in W3C® Data integrity specification [2]: eddsa-rdfc-2022, eddsa-jcs-2022, ecdsa-jcs-2019, ecdsa-rdfc-2019.

Both JSON Canonicalization Scheme and RDF Dataset Canonicalization are supported by the W3C Data Integrity specification.

5.2 Data integrity and provenance for NGSI-LD

5.2.0 Foreword

Adoption of a W3C® Data Integrity signature mechanism that is based on an RDF Dataset Canonicalization (for example the Ed25519Signature2022 proof type, which produces a verifiable digital proof by canonicalizing the input data using the RDF Dataset Canonicalization algorithm and then digitally signing it using an Ed25519 elliptic curve signature), fulfils requirements INT-1, INT-2 and INT-4, thus guaranteeing data integrity and provenance through the whole NGSI-LD Entity lifecycle.

In order to fulfil the INT-3 requirement, i.e. in order to specify how to serialize and embed the W3C® verifiable digital proof into the NGSI-LD Entity, the following need to be defined and detailed:

- Atomic Entity.
- Sealed Attribute.
- Derivation process.
- Reconstruction Process.

5.2.1 Atomic Entity

Prior to signing, every Entity can be seen as made of two parts:

- The id and type part (head).
- The attributes part (core).

But during the typical NGSI-LD context data lifecycle:

- Multiple Entity aggregation steps can happen, where a Broker merges Entities with the same Entity id, having different Attributes, that come from different Context Brokers or Context Providers, and it serializes them as one bigger Entity at the next step.
- Context Brokers' responses to queries can filter out Attributes, thus serializing just a sub-set of all Attributes of an Entity at the next step.
- For integrity and provenance purposes, Clients only trust the context data creators (requirement INT-4), not the intermediate, relaying Context Brokers. Hence signature schemes involving re-signing, by the intermediate Context Brokers, the newly created Entities at each step (including possible schemes where Broker and creators might collaborate) are not allowed for the purposes of the present document.

Thus, picturing the Entity as a chain and each Attribute as a link of the chain, every manipulation process (merging, selective disclosing) changes only its core, not the head, akin to adding or removing links.

After signing, every Entity can be seen as made of three parts:

- The id and type part (head).
- The attributes part (core).
- The proof part (tail).

But, if one proof tail covers a core composed of more than one Attribute, it is impossible to manipulate the core and retain the signature.

It comes as a consequence that the atomic piece of information that creators can digitally sign in an NGSI-LD ecosystem is each single Attribute of an Entity, i.e. a core with **one Attribute only, together with its head**. The information contained in the head shall be cryptographically signed and bound together with the Attribute, because the information that the Attribute is part of a specific Entity, with its type, shall be verifiable by the Client.

The solution is the implementation of a one-Entity-one-Attribute structure, that is an Entity having one single Attribute and one single proof in it. It is possible to define the **Atomic Entity**:

```
{
  id
  type            head
  attribute       core
  proof           tail
}
```

That is, a signed Entity with only one Attribute, in normalized representation. It represents the first pillar of this model and it is the building-block Entity structure that will keep its signature, and all its content is tamper-evident.

The proof component of the Atomic Entity is the W3C® Data Integrity "`proof`" element (see clause 5.1.1) with all of its properties.

5.2.2 Sealed Attribute

The second pillar is the definition of a signed Attribute structure that will allow all information of one Atomic Entity to be nested in such an Attribute, thus allowing for multiple Atomic Entities (having a common Entity id) to be transported as multiple (Sealed) Attributes of a bigger Entity. Incorporating every information about the Atomic Entity's head (id, type) and tail (proof) inside such an Attribute, will allow the possibility to treat it as a link to be shared among different actors, as it brings its cryptographic signature, and all information needed to verify it at a later time, with it, making it independent and self-standing from the integrity and provenance point of view.

This structure is called **Sealed Attribute** and it is an Attribute with the addition of an "`ngsildproof`" sub-property. "`ngsildproof`" contains the following information:

- `type`: NGSI-LD Property.
- `entityIdSealed`: id value of the originating Atomic Entity. This is a non-reified sub-property of the "`ngsildproof`".

- `entityTypeSealed`: type value of the originating Atomic Entity. This is a non-reified sub-property of the "`ngsildproof`".
- `value`: object containing the W3C® Data integrity "`proof`" structure of the originating Atomic Entity.

EXAMPLE: This is a generic Property that can be considered a Sealed Attribute because it has the "`ngsildproof`" sub-property.

```
"property1" = {
  "type": "Property",
  "value": "value1",
  "ngsildproof": {
    "type": "Property",
    "entityIdSealed": "…",
    "entityTypeSealed": "…",
    "value": {
      "proof": {…}
    }
  },
  "sub-attribute1": "…",
  "sub-attributeN": "…"
}
```

Thus, the only modification needed to make an Attribute become a Sealed Attribute is the addition of the dedicated "`ngsildproof`" sub-property.

5.2.3 Derivation Process

The Sealed Attribute will be created during the **Derivation Process**. The Derivation Process algorithm will take as input an Atomic Entity and it will have a Sealed Attribute as output.

In the following steps:

1) The Sealed Attribute is initialized as being a clone of the single Attribute of the Atomic Entity.
2) The "`ngsildproof`" structure is created and nested inside the Sealed Attribute as a sub-property.
3) Inside the "`ngsildproof`" structure, "`entityIdSealed`" and "`entityTypeSealed`" non-reified sub-properties are created, holding respectively a copy of the id and type values of the Atomic Entity.
4) The "`value`" field is created inside "`ngsildproof`", holding a copy of the "`proof`" object of the Atomic Entity.

After the Derivation Process, the Sealed Attribute can be treated, just like any regular Attribute in the NGSI-LD ecosystem, as a link that can be connected to every chain with the same head part, through any Broker merging processes: every Entity with the same id value, coming to a Context Broker, will be merged into a bigger Entity with a common head, as all received Sealed Attributes become part of its core, but no signatures are invalidated during the merging (or subsequent selective disclosure) process.

EXAMPLE: Entity with two Sealed Attributes, the first a Property, the second one a Relationship.

```
{
  "id": "urn:ngsi-ld:Car123",
  "type": ["Car", "Vehicle"],
  "color": {
    "type": "Property",
    "value": "Red",
    "ngsildproof": {
      "type": "Property",
      "entityIdSealed": "urn:ngsi-ld:Car123",
      "entityTypeSealed": "Car",
      "value": {
        "proof": {…}
      }
      "sub-attribute1": "…",
      "sub-attributeN": "…"
   }
  },
  "parkedAt": {
    "type": "Relationship",
```

```
    "object": "urn:ngsi-ld:Parking123",
    "ngsildproof": {
      "type": "Property",
      "entityIdSealed": "urn:ngsi-ld:Car123",
      "entityTypeSealed": "Vehicle",
      "value": {
        "proof": {…}
      }
    }
  }
}
```

5.2.4 Reconstruction Process

The original Atomic Entity can be recreated trough the **Reconstruction Process**, which is the opposite of the Derivation Process and takes as input a Sealed Attribute and generates an Atomic Entity as output.

It is important to specify that the proof will not verify the Sealed Attribute structure, but only the original Atomic Entity. Only the reconstruction of the original Atomic Entity will allow the validation of the signature.

5.2.5 Workflow

Atomic Entities and Sealed Attributes are the two pillars, together with the two transformation processes, that allow the Sealed Attributes to pass from Context Provider to Client, via any intermediaries, bringing within them their original verifiable signature.

Following the typical NGSI-LD context data lifecycle it is possible to distinguish three main phases:

1) Collection and merging of Entities coming from Context Providers, performed by intermediate relaying Context Brokers.

2) Sharing and selectively disclosing Entities among federated Context Brokers.

3) Presenting merged Entities, as a result of a query, to Context Consumers.

The data integrity of an Entity through these phases will be guaranteed from the Context Producer to the Client following these steps:

1) Context Provider generates and signs one Atomic Entity for each Attribute of an Entity, by removing all other Attributes.

2) Each Atomic Entity is transformed into a Sealed Attribute, through a Derivation process.

3) Context Provider replaces each Attribute of the Entity with its corresponding Sealed Attribute.

4) The Entity undergoes any number of cycles/steps of the three phases above, so that Sealed Attributes may be aggregated with other regular Attributes or may even be removed by selective disclosure policies.

5) When the manipulated Entity reaches the Client, all included Sealed Attributes are transformed to Atomic Entities, following the Reconstruction Process. The recreated Atomic Entities are validated by Client. The Client checks that the current Entity id matches with "`entityIdSealed`".

Annex A (informative): Changes to the NGSI-LD API

To implement the specifications provided in the present document, changes are required to the main NGSI-LD API [i.2], specifically to the following:

- The NGSI-LD Core @context, to introduce URIs and terms identifying the newly introduced non-reified sub-properties.
- The procedures for parsing and serializing NGSI-LD Attributes, since the newly introduced "ngsildproof" Property, and its whole internal structure, is to be recognized and treated differently.

The above required changes are introduced in ETSI GS CIM 009 (V1.9.1) [i.2].

Annex B (informative): Example digital signature workflow

Original Entity

The original Entity contains two Properties: "address" and "location".

EXAMPLE 1: Entity of type "Store" with two Properties.

```
{
  "id": "urn:ngsi-ld:Store:002",
  "type": "Store",
  "address": {
    "type": "Property",
    "value": {
      "streetAddress": ["Tiger Street 4", "al"],
      "addressRegion": "Metropolis",
      "addressLocality": "Cat City",
      "postalCode": "42420"
    }
  },
  "location": {
    "type": "GeoProperty",
    "value": {
      "type": "Point",
      "coordinates": [57.5522, -20.3484]
    }
  },
  "@context": "https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld"
}
```

Atomic Entity with just one Property

The Atomic Entity can only contain one Attribute, for instance he "address" Property.

EXAMPLE 2: Atomic Entity for the "address" Property.

```
{
  "id": "urn:ngsi-ld:Store:002",
  "type": "Store",
  "address": {
    "type": "Property",
    "value": {
      "streetAddress": ["Tiger Street 4", "al"],
      "addressRegion": "Metropolis",
      "addressLocality": "Cat City",
      "postalCode": "42420"
    }
  },
  "@context": "https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld"
}
```

Signature on the Atomic Entity

The Atomic Entity is signed by tools that implement the W3C® Data Integrity specification.

EXAMPLE 3: Signed Atomic Entity above. The signature is created using one Ed25519 instantiation of the Edwards-Curve Digital Signature Algorithm (EdDSA).The used crypto suite is "eddsa-rdfc-2022".

```
{
  "id": "urn:ngsi-ld:Store:002",
  "type": "Store",
  "address": {
    "type": "Property",
    "value": {
      "streetAddress": [
        "Tiger Street 4",
        "al"
      ],
      "addressRegion": "Metropolis",
```

```
      "addressLocality": "Cat City",
      "postalCode": "42420"
    }
  },
  "@context": [
    "https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld",
    "https://w3id.org/security/data-integrity/v2"
  ],
  "proof": {
    "type": "DataIntegrityProof",
    "created": "2025-01-27T21:02:24Z",
    "verificationMethod": "https://example.edu/issuers/565049#z6MkwXG2WjeQnN....Hc6SaVWoT",
    "cryptosuite": "eddsa-rdfc-2022",
    "proofPurpose": "assertionMethod",
    "proofValue": "z3XrH3diVCqpVHXkE7WbnictqyQCkJBGTx....NRTzmuoWU1Y2FyqGfSV9eS"
  }
}
```

Derivation of the Sealed Attribute

The corresponding Sealed Attribute is created by embedding id and type information of the original Entity.

EXAMPLE 4: Derived Sealed Attribute for the "address" Property of the Store:002 Entity.

```
"ngsildproof": {
  "type": "Property",
  "entityIdSealed": "urn:ngsi-ld:Store:002",
  "entityTypeSealed": "Store",
  "value": {
    "type": "DataIntegrityProof",
    "created": "2025-01-27T21:02:24Z",
    "verificationMethod": "https://example.edu/issuers/565049#z6MkwXG2WjeQnN....Hc6SaVWoT",
    "cryptosuite": "eddsa-rdfc-2022",
    "proofPurpose": "assertionMethod",
    "proofValue": "z3XrH3diVCqpVHXkE7WbnictqyQCkJBGTx....NRTzmuoWU1Y2FyqGfSV9eS"
  }
}
```

Original Entity with cryptographic proof

The Sealed Attribute is brought back into the original Entity.

EXAMPLE 5: Entity of type "Store" with two Properties. The "address" Property is digitally signed.

```
{
  "id": "urn:ngsi-ld:Store:002",
  "type": "Store",
  "address": {
    "type": "Property",
    "value": {
      "streetAddress": ["Tiger Street 4", "al"],
      "addressRegion": "Metropolis",
      "addressLocality": "Cat City",
      "postalCode": "42420"
    }
    "ngsildproof": {
      "type": "Property",
      "entityIdSealed": "urn:ngsi-ld:Store:002",
      "entityTypeSealed": "Store",
      "value": {
        "type": "DataIntegrityProof",
        "created": "2025-01-27T21:02:24Z",
        "verificationMethod": "https://example.edu/issuers/565049#z6MkwXG2WjeQnN....Hc6SaVWoT",
        "cryptosuite": "eddsa-rdfc-2022",
        "proofPurpose": "assertionMethod",
        "proofValue": "z3XrH3diVCqpVHXkE7WbnictqyQCkJBGTx....NRTzmuoWU1Y2FyqGfSV9eS"
      }
    }
  },
  "location": {
    "type": "GeoProperty",
    "value": {
      "type": "Point",
      "coordinates": [57.5522, -20.3484]
```

```
    }
  },
  "@context": "https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld"
}
```

Annex C (informative): Change history

Date	Version	Information about changes
August 2022	1.1.1	First publication
January 2025	1.1.2	Draft of novel version 1.2.1

History

Document history		
V1.1.1	August 2022	Publication
V1.2.1	August 2025	Publication