|  |  |
| --- | --- |
| Project | **Interfacing Cyber and Physical World Working Group**  <<https://sagroups.ieee.org/2888/> **>** |
| Title | **Ticketing system with DID technology for cyber and physical world** |
| DCN | **2888-24-0068-00-0000** |
| Date Submitted | **Oct, 12, 2024** |
| Source(s) | Sang-Kyun Kim, [goldmunt@gmail.com](mailto:goldmunt@gmail.com) (Myongji University)  Younghwan Kim, dudghks021121@gmail.com (Myongji University)  Seonghoon Han, [a01073561977@gmail.com](mailto:a01073561977@gmail.com) (Myongji University) |
| Re: | IEEE 2888 Session #18, Brisbane, Austrailia |
| Abstract | This contribution proposes semantics and examples of pressure sensor capabilty. |
| Purpose | To start discussion on purpose of the standard |
| Notice | This document has been prepared to assist the IEEE 2888 Working Group. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein. |
| Release | The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE’s name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE’s sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that IEEE 2888 may make this contribution public. |
| Patent Policy | The contributor is familiar with IEEE patent policy, as stated in [Section 6 of the IEEE-SA Standards Board bylaws](http://standards.ieee.org/guides/opman/sect6.html#6.3) <[http://standards.ieee.org/guides/bylaws/sect6-7.html#6](http://127.0.0.1:4664/cache?event_id=757737&schema_id=1&s=5X0vID10lu_E6yrIkWkNd4Wz2H8&q=hancock)> and in *Understanding Patent Issues During IEEE Standards Development* <http://standards.ieee.org/board/pat/faq.pdf> |

# **Introduction**

Traditional media content such as plays, operas, concerts, and sports are constantly receiving attention from many people. In particular, the competition for ticketing for performances by singers or bands with large fandoms such as K-POP, or games of sports stars or teams is fierce. As a result, side effects such as group ticketing and black market tickets are increasing, and they are becoming factors that hinder healthy cultural/sports viewing. This proposal introduces DID technology as a means to prevent such side effects, and explains an efficient ticketing method and system using it. In addition, it discusses the possibility of standardization for implementing a ticketing system for the Cyber ​​world as well as the Physical world.

# **What is DID?**

## **DID Architecture Overview and DID**

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

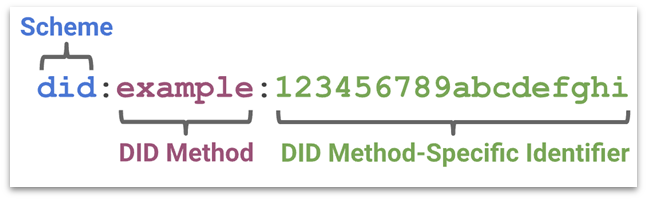
**Figure 1. Overview of DID architecture and the relationship of the basic components [1].**

DID stands for decentralized identification. A DID is a self-sovereign identity, i.e., a lifetime, portable, and verifiable identity that does not depend on any centralized authority. The characteristics of DID are as follows:

* Ease of creation: It should be quick and “cheap” to create possibly thousands of DIDs.
* Decentralized: Do not depend on centralized registries, identity providers, authorities, etc.
* Persistent: Once created, it is permanently assigned to the subject.
* Resolvable: It is possible to find out a basic set of information on the subject.
* Cryptographically verifiable: There is a mechanism to prove identity and ownership cryptographically.

In short, DID is a method that individuals can easily create, own, and prove themselves or their subjects at any time (cryptographic method). While the existing SSO (Single Sign On) method is a centralized method that depends on service providers (e.g. Google, Naver, Kakao) or CA institutions, DID is an SSI (Self Sovereign Identification) method in which individuals are the main actors from ID issuance to use.

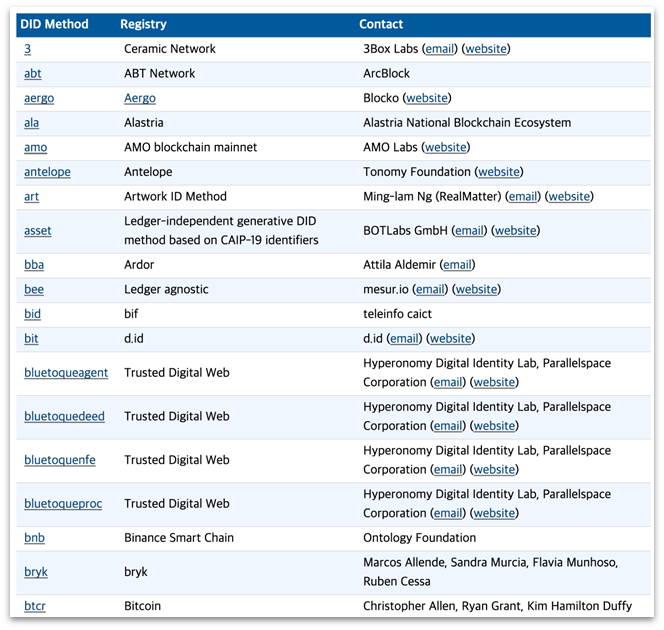
Figure 1 shows an overview of this DID architecture. Individuals create and store a DID and a DID document that proves it in a specific VDR (Verifiable Data Registry) through a DID registrar program (e.g., App or Web). The DID document includes a means to prove the DID (e.g., public key) and a proof method. When requesting proof for a DID, the DID resolver program can refer to the DID document through the location information of the DID document stored in the DID location within the VDR (e.g., blockchain). The DID requester verifies the validity of the DID through the proof method in the DID document.



**Figure 2. Structure of DID [2].**

Figure 2 shows the structure of DID. DID starts with “did”, followed by the DID method according to the storage location, and finally consists of the method-specific identifier provided by the DID method. Figure 3 shows the DID methods registered in the standard. The following is an example of DID.

* **did:btcr:xyv2-xzpq-q9wa-p7t**: Built on top of the Bitcoin blockchain. The method-specific identifier is generated from the Bitcoin transaction position reference.
* **did:sov:mnjkl98uipsndg2hdjdjuf7**: Based on a dedicated distributed ledger (Sovrin-Hyperledger Indy). The method-specific identifier is generated from either a simple UUID or the subject’s public key.



**Figure 3. List of repositories supported by DID Method [3].**

## **DID document**

### **DID document example**

A DID document contains an authentication means that can prove ownership of a DID. The main items are context, id, publicKey, authentication, and service.

Figure 4 shows an example of a DID document.

1. @context: It clearly defines what data items such as id, authentication, and service in a DID document mean and what type of data should be included, and it is defined according to JSON-LD grammar.
2. id: The id field in the third line of Figure 4 contains the DID of the object identified by the id. If the purpose is to identify a user, the user's DID is included, and if the purpose is to identify an object, the object's DID is included. Typically, the DID of the person who created and registered the DID and DID document is included in the id field.
3. publicKey: Used to authenticate DID ownership.
   1. id: Used to authenticate DID ownership.
   2. type: Various authentication methods can be included, such as asymmetric key authentication (RSA), biometric authentication, and elliptic curve asymmetric key authentication.
   3. controller: Indicates who has the authentication authority for publicKeyPem 🡪 In the case of Ethereum-based RSA asymmetric key authentication, the DID of the person who has the private key paired with the stored public key (Line 8 in Figure 4) is entered.
   4. publicKeyPem: Storing data to be used for ownership authentication 🡪 Line 8 of Figure 4 stores the public key required for asymmetric authentication.
4. authentication: Indicates the ownership authentication method provided by the corresponding DID document 🡪 Users can authenticate DID ownership by selecting one of the three methods in lines 21 to 23 of Figure 4.



**Figure 4. DID document example**



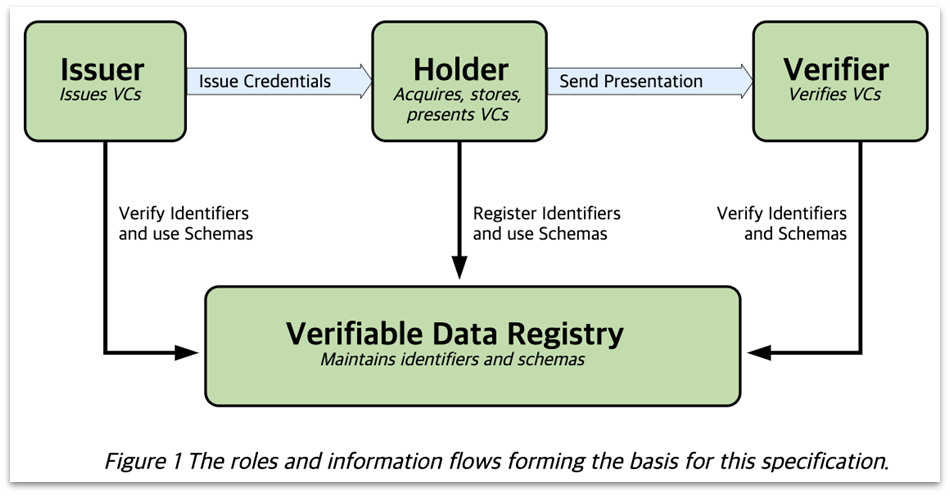
**Figure 5 DID ownership authentication example.**

Figure 5 is an example of DID ownership authentication when the user has the authentication authority for the DID. Figure 6 is an example of when the DID owner does not have the authentication authority for the DID.



**Figure 6. An organization other than the DID owner has the authority to authenticate the DID.**

## **Verifiable Credential and Verifiable Presentation**

****

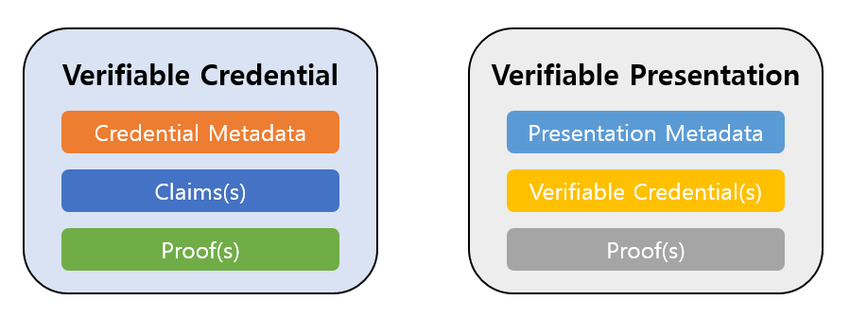
**Figure 7. The roles and information flows forming the basis of this specification**

Figure 7 shows the information flow between the issuer, holder, and verifier.

### **Verifiable credential data model**

Figure 8 shows the structure of the verifiable credential and the verifiable presentation. The verifiable credential is composed of three components as follows:

* Credential metadata: It defines who issued the VC, the entity (Credential subject) that the VC is referring to, the VC expiration period, and the VC disposal method.
* Claim(s): Information about the ID attribute of the credential subject is stored in Subject-Property-Value format.
* Proof(s): Contains values ​​required to verify the authenticity of the VC, and various cryptographic techniques such as RSA, ECDSA, and biometric authentication can be used for verification. Contains the signature of the VC issuer.



**Figure 8. Structure of verifiable credential and verifiable presentation.**

The data items included in VC by default are @context, id, type, issuer, issuanceDate (validFrom), expirationDate (validUntil), credentialSubject, credentialSchema, credentialStatus, proof, etc. (Figure 9).

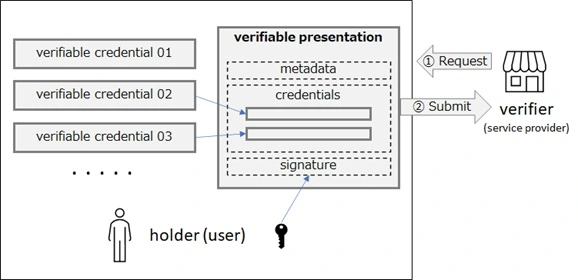
1. @context: It plays a role in defining what value the data has. The third line of Figure 9, “https://www.w3.org/2018/credentials/v1”, is the official VC context, which must be included. The fourth line of Figure 9, “https://www.example.edu/context”, is a context that we created ourselves when additional context is needed for the service we are developing.
2. Id: Figure 9 The sixth line, “http://example.edu//credential/yoon”, is the identifier of the VC, and the DID in the 12th line is the identifier of the object identified by the VC.
3. Type: “VerifiableCredential” in line 7 of Figure 7 means that VC will be created according to the basic VC data structure defined in the VC official context, and “KoreanUniversityCredentia” means that data required for graduation proof will be created according to the data structure defined in the self-created context.
4. Issuer: Refers to the person or institution that issued the VC.
5. issuanceDate 🡪 validFrom: Defines when a VC is valid.
6. expirarionDate 🡪 validUntil: Defines how long a VC is valid.
7. **credentialSubject: An item containing claim(s) data.**
8. **credentialSchema: You can define a schema for specific attributes in a claim or VC. You can restrict non-mandatory attributes in a VC to required attributes.**
9. credentialStatus: This item indicates the status of the credential (valid or expired).
10. proof: Information for VC authentication, “type” is the type of proof, “proofPurpose” prevents misuse other than the intended purpose. “VerificationMethod” is the url where the public key is stored. “created” is the time the proof was created. “proofValue” contains the value in which the digital signature binary data is encoded.



**Figure 9. Verifiable credential example**

Figure 10 shows the structure of Verifiable Presentation and an example of its use.

* Presentation metadata: It includes data that can be referenced for VP verification, such as type, terms of use, and evidence.
* Verifiable credential(s): Privacy protection is possible because the verifier can selectively include the VC with the ID attributes required and the required ID attributes among the Claims within the VC.
* Proof(s): While the proof item of VC contains the issuer's signature, the VP Proof(s) contains the user's signature, and various cryptographic techniques can be used.



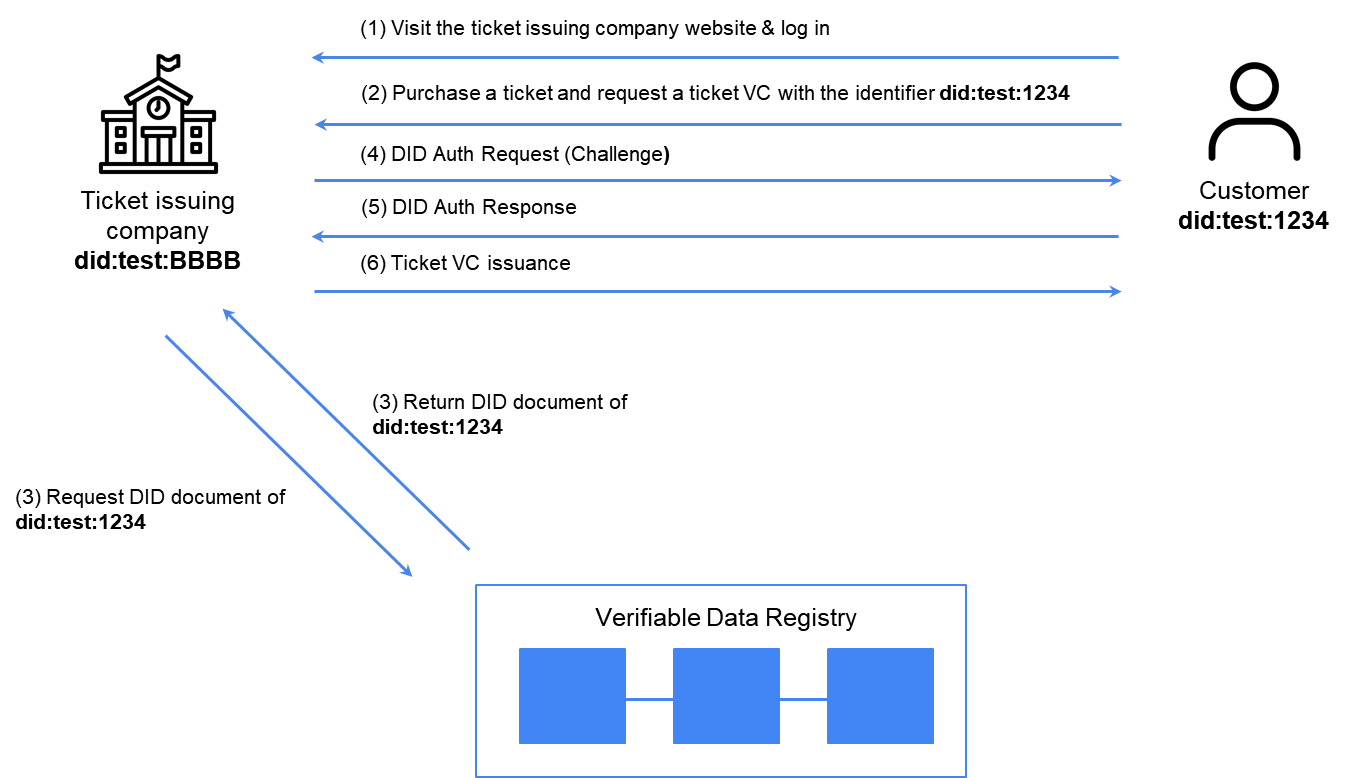
**Figure 10. Usage of verifiable presentation.**

# **Use case of ticketing systems with DID**

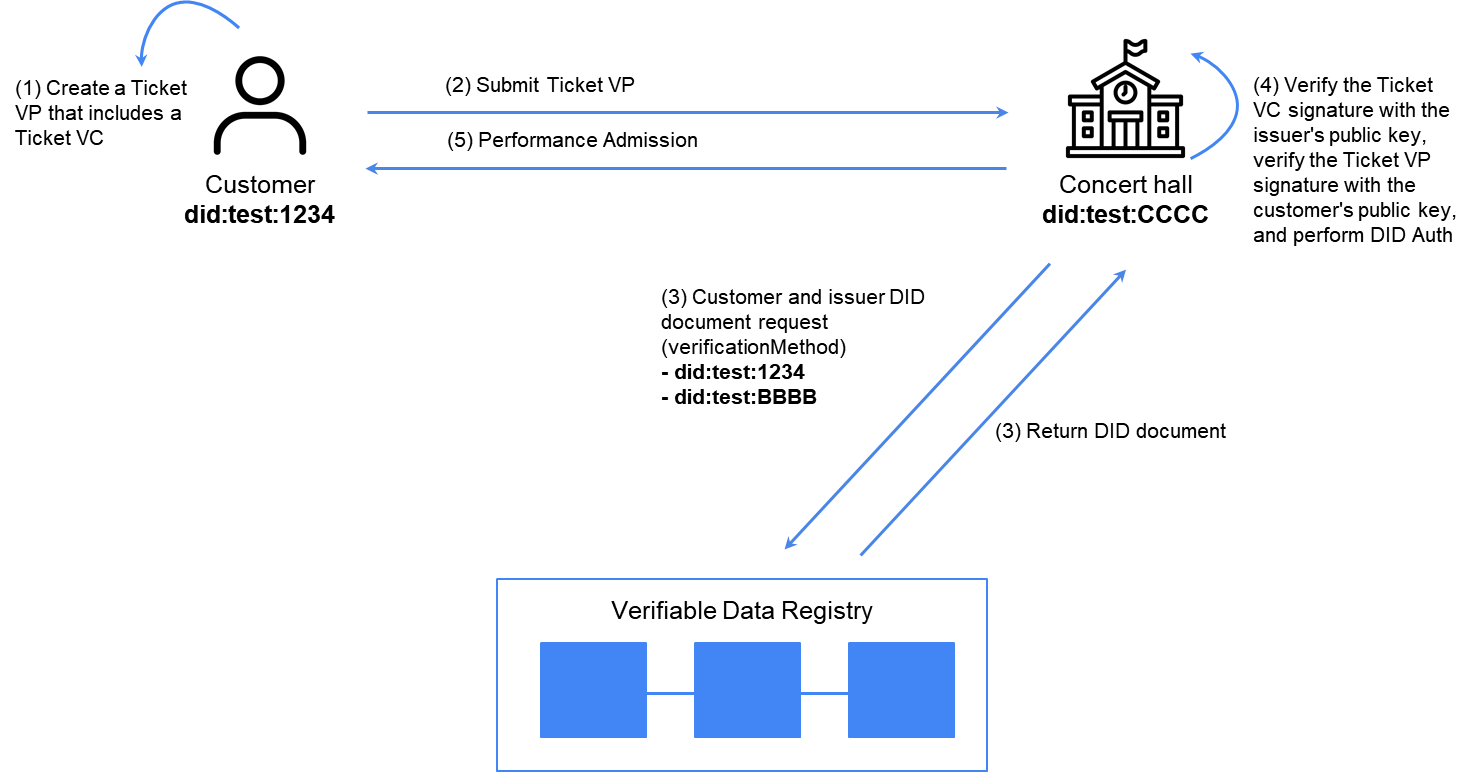
## **Use case**

Use case 1: In a ticket purchasing environment using DID technology in the physical world, the VC holder can be a ticket purchaser, the issuer can be a ticket sales company, and the verifier can be a concert hall management company. Ticketing in the physical world using DID goes through the following process.

1. The buyer creates their own DID through the DID registrar app.
2. The buyer uses their DID to purchase tickets to the concert they are interested in (selecting the seat, performance date, etc.). If all three members of their family are going to the concert, they enter the DIDs of all family members and purchase the tickets.
3. The concert ticket issuing company authenticates the purchaser's DID and generates and issues a Ticket VC based on this information. The purchaser stores the issued Ticket VC in their mobile phone wallet.
4. On the day of the concert, the purchaser presents the Ticket VP (Ticket VC converted to Ticket VP through the purchaser's signature) in their wallet to the inspector at the entrance to the concert venue.
5. At the entrance of the concert hall, the ticket inspector receives the purchaser's ticket VP on their mobile phone. The ticket inspectors use the camera or fingerprint recognition device of their mobile phone to authenticate the purchaser by obtaining the purchaser's face and fingerprint information (the purchaser's private key) corresponding to the DID registered in the ticket VP. If the concert hall has a ticket VP recognition device with facial recognition or fingerprint recognition, automatic entry is possible without a ticket inspector.



**Figure 11. Process of issuing a ticket VC.**



**Figure 12. Process of verifying a ticket VP.**

Use case 2: There is a BTS concert in Seoul. ARMYs around the world who cannot come to Seoul want to watch BTS's concert remotely and in real-time. Just like the process above, buyers who purchase remote tickets keep the Ticket VC they receive in their wallets and convert it into Ticket VP. On the day of the concert, they access the concert with their HMD, submit the Ticket VP, and then verify their identity with the pupil recognition device or fingerprint recognition device attached to the HMD before entering the concert. Users can watch the concert realistically from their desired point of view by receiving 3D real-time images reconstructed through hundreds of cameras installed in the concert hall.

## **Expected Effects**

Tickets using DID technology allow only the person entered at the time of purchase (the DID of an acquaintance including the purchaser) to enter the concert hall. DID technology allows the effect of fundamentally blocking illegal transfers of the ticket since only the DID holder registered with the ticket can enter. Only the DID holder registered with the ticket can be authenticated using their private key (e.g. fingerprint, face, pupil). In addition, it is expected to open up new markets in the industries of DID issuance and management, ticket VC issuance, VC wallet software on personal mobile devices, and VP verification devices for VP recognition (such as personal key authentication sensors in mobile devices and HMD devices).

# **Standards for the ticketing system with DID?**

## **JSON Schema [4 -6]**

{

"$id": "https://example.com/schemas/ticket.json",

"$schema": "https://json-schema.org/draft/2020-12/schema",

"title": "VerifiableTicketCredential",

"description": "Credential data schema of ticket claim",

"type": "object",

"properties": {

"credentialSubject": {

"type": "object",

"properties": {

"id": {

"type": "string",

"format": "uri"

},

"ticketNumber": {

"type": "string"

},

"ticketToken": {

"type": "string",

"format": "uri"

},

"dateIssued": {

"type": "string",

"format": "date"

},

"issuedBy": {

"type": "object",

"properties": {

"name": {

"type": "string"

},

"id": {

"type": "string",

"format": "uri"

}

}

},

"priceCurrency": {

"type": "string"

},

"totalPrice": {

"type": "number"

},

"ticketedSeat": {

"type": "array",

"items": {

"type": "object",

"properties": {

"seatRow": {

"type": "string"

},

"seatNumber": {

"type": "string"

},

"seatSection": {

"type": "string"

},

"seatingType" : {

"type" : "string"

}

},

"required" :["seatingType"]

}

},

"underName": {

"type": "array",

"items": {

"type": "object",

"properties": {

"name": {

"type": "string"

},

"id": {

"type": "string",

"format": "uri"

}

}

}

}

},

"required": ["ticketNumber", "ticketToken", "issuedBy", "underName"]

}

}

## **Semantics**

| *Name* | *Definition* |
| --- | --- |
| id | The URI of the credential subject uniquely identifies the owner. |
| ticketNumber | Unique number of the ticket |
| ticketToken | Represents a token associated with a ticket, usually in the form of a QR code or link. |
| dateIssued | Date the ticket was issued. |
| issuedBy | As an issuing authority, for example, an organization that issues permits, tickets or certificates |
| priceCurrency | Currency unit of ticket price (e.g. USD, KRW) |
| totalPrice | The total price of the ticket |
| ticketedSeat | Seat Information |
| underName | Ticket holder information |

## **Example**

John Doe purchases a concert ticket online. The ticket issuer, “Event Organization Inc.”, issues a Verifiable Credential (i.e., ticket VC) that contains his name, ticket number, and seat information. When entering the concert, John Doe presents the VP (which is generated from the ticket VC), and the concert organizer scans his face through an app and authenticates it with the public key of the facial data contained in the Jone Doe’s DID (i.e., DID document). Alternatively, he scans his fingerprint and verifies a hash of his fingerprint data. If one or more biometric authentications are completed, John Doe is granted entry.

{

"@context": [

"https://www.w3.org/ns/credentials/v2",

"https://www.w3.org/ns/credentials/examples/v2"

],

"id": "https://example.com/credentials/3732",

"type": ["VerifiableCredential", "VerifiableTicketCredential"],

"issuer": "https://example.com/issuers/14",

"credentialSubject": {

"id": "did:example:ebfeb1f712ebc6f1c276e12ec21",

"ticketNumber": "ABC123456789",

"ticketToken": "https://example.com/tickets/qr/ABC123456789",

"dateIssued": "2024-01-01T12:00:00Z",

"issuedBy": {

"name": "Event Organization Inc.",

"id": "https://example.com/organizations/14"

},

"priceCurrency": "USD",

"totalPrice": 99.99,

"ticketedSeat": {  
 "seatingType" : "standing"

},

"underName": {

"name": "John Doe",

"id": "did:example:abc123",

}

},

"credentialSchema": {

"id": "https://example.com/schemas/ticket.json",

"type": "JsonSchema",

"digestSRI": "sha384-1C7ACB24E20BD41F2990048ACDC1F56E595B25AAC0D5DDF70D24A03631D606D04A854F661381C03E41D2A97401FA50E1"

},

"proof": [

{

"type": "DataIntegrityProof",

"cryptosuite": "eddsa-rdfc-2022",

"created": "2020-11-05T19:23:24Z",

"verificationMethod": "https://ldi.example/issuer/1#z6MkjLrk3gKS2nnkeWcmcxiZPGskmesDpuwRBorgHxUXfxnG",

"proofPurpose": "assertionMethod",

"proofValue": "z4oey5q2M3XKaxup3tmzN4DRFTLVqpLMweBrSxMY2xHX5XTYVQeVbY8nQAVHMrXFkXJpmEcqdoDwLWxaqA3Q1geV6"

}

]

}

John Doe purchases a concert ticket online with Jane Doe under his name. The ticket issuer, “Event Organization Inc.”, issues a Verifiable Credential (i.e., ticket VC) that contains their name, ticket number, and seat information. When entering the concert, John Doe presents the VP (which is generated from the ticket VC), and the concert organizer scans his face through an app and authenticates it with the public key of the facial data contained in the Jone Doe’s DID (his DID document). Then Jane scans her fingerprint to verify the hash of her fingerprint data in her DID document. If John and Jane's biometric authentication is successful, John and Jane are granted entry.

{

"@context": [

"https://www.w3.org/ns/credentials/v2",

"https://www.w3.org/ns/credentials/examples/v2"

],

"id": "https://example.com/credentials/3732",

"type": [

"VerifiableCredential",

"VerifiableTicketCredential"

],

"issuer": "https://example.com/issuers/14",

"credentialSubject": {

"id": "did:example:ebfeb1f712ebc6f1c276e12ec21",

"ticketNumber": "ABC123456789",

"ticketToken": "https://example.com/tickets/qr/ABC123456789",

"dateIssued": "2024-01-01T12:00:00Z",

"issuedBy": {

"name": "Event Organization Inc.",

"id": "https://example.com/organizations/14"

},

"priceCurrency": "USD",

"totalPrice": 99.99,

"ticketedSeat": [

{

"seatRow": "B",

"seatNumber": "12",

"seatSection": "Orchestra",

"seatingType": "seating"

},

{

"seatRow": "B",

"seatNumber": "13",

"seatSection": "Orchestra",

"seatingType": "seating"

}

],

"underName": [

{

"name": "John Doe",

"id": "did:example:abc123",

},

{

"name": "Jane Doe",

"id": "did:example:dfe456",

}

]

},

"credentialSchema": {

"id": "https://example.com/schemas/ticket.json",

"type": "JsonSchema",

"digestSRI": "sha384-1C7ACB24E20BD41F2990048ACDC1F56E595B25AAC0D5DDF70D24A03631D606D04A854F661381C03E41D2A97401FA50E1"

},

"proof": [

{

"type": "DataIntegrityProof",

"cryptosuite": "eddsa-rdfc-2022",

"created": "2020-11-05T19:23:24Z",

"verificationMethod": "https://ldi.example/issuer/1#z6MkjLrk3gKS2nnkeWcmcxiZPGskmesDpuwRBorgHxUXfxnG",

"proofPurpose": "assertionMethod",

"proofValue": "z4oey5q2M3XKaxup3tmzN4DRFTLVqpLMweBrSxMY2xHX5XTYVQeVbY8nQAVHMrXFkXJpmEcqdoDwLWxaqA3Q1geV6"

}

]

}

# **Conclusion**

We recommend this new use case and application as a new standard project in the IEEE 2888 working group.

# **References**

1. DID Architecture Overview: [https://w3.org/TR/did-core/#architecture-overview](https://w3.org/TR/did-core/)
2. DID Simple Example: [https://www.w3.org/TR/did-core/#a-simple-example](https://www.w3.org/TR/did-core/)
3. DID Methods: [https://w3c.github.io/did-spec-registries/#did-methods](https://w3c.github.io/did-spec-registries/)
4. <https://www.w3.org/TR/vc-data-model-2.0/>
5. <https://www.w3.org/TR/vc-json-schema/>
6. <https://schema.org/Ticket>