PDL ISG - PoC Proposal Template

1 PoC Project Details

1.1 PoC Project

PoC Number (assigned by ETSI): PoC#04

PoC Project Name: IoT-based access control system using NFT ticket to prevent scalping

PoC Project Host: Sejong University

Short Description: Traditional access authentication systems face multiple security vulnerabilities, including forgery, tampering, and server hacking. For example, issues with scalpers who steal identities to buy and illegally transfer bulk tickets for cultural performances highlight the need for a system that ensures access only with legitimately issued tickets. This PoC introduces a new ticket issuance and anti-counterfeiting solution using blockchain's Non-Fungible Tokens (NFTs) to combat scalping and counterfeit tickets. Integrated with an IoT access control system, it allows entry solely to individuals holding validly issued tickets. Utilizing a decentralized distributed ledger enhances security by preventing unauthorized access from forged passes or compromised servers. The system employs blockchain standard PDL and IoT standard oneM2M technologies, enhancing interoperability and facilitating future expansion with other IoT devices.

1.2 PoC Team Members

Table A.1

	Organization name	ISG PDL participant (yes/no)	Contact (Email)	PoC Point of Contact (see note 1)	Role (see note 2)	PoC Components
1	Sejong University (SJU)	Yes	jssong@sejong.ac.kr	X	Application /infrastructure provider	Blockchain development, oneM2M development
2	Guarantable	No	kyle@guarantable.com		Application /infrastructure provider	Blockchain development
3	ETRI	No	pjs@etri.re.kr		Application provider	Blockchain development
…						
NOTE 1: Identify the PoC Point of Contact with an X. NOTE 2: The Role will be network operator/service provider, infrastructure provider, application provider or other as given in the Definitions of ETSI Classes of membership.						

All the PoC Team members listed above declare that the information in this proposal is conformant to their plans at this date and commit to inform ETSI timely in case of changes in the PoC Team, scope or timeline.

1.3 PoC Project Scope

1.3.1 PoC Goals

The PoC will demonstrate:

- Utilization of PDLs in oneM2M using standardized and interoperable interfaces, related to ETSI GR PDL 028 (Study on Utilizing PDLs to Standardized IoT Service Layer Platform oneM2M)
- Developing a secure, verifiable, and immutable ticket system using smart contracts, related to ETSI GS PDL 004 (Smart Contracts System Architecture and Functional Specification)

1.3.2 PoC Topics

PoC Topics identified in this clause need to be taken for the PoC Topic List identified by ISG ENI and publicly available, i.e. the three topics identified in clause 4.5 of the ENI PoC Framework. PoC Teams addressing these topics commit to submit the expected contributions in a timely manner.

Table A.2

PoC Topic Description (see note)	Related WI	Expected Contribution	Target Date
Utilization of PDLs in oneM2M using standardized and interoperable interfaces	related to ETSI GR PDL 028 (Study on Utilising PDLs to Standardized IoT Service Layer Platform oneM2M)	Validation of PDL utilization in oneM2M	2024-06
Developing a secure, verifiable, and immutable ticket system using smart contracts	related to ETSI GS PDL 004 (Smart Contracts System Architecture and Functional Specification)	Smart contract application and validation	2024-06
Integrating the PDL-based ticket system with oneM2M access control system	related to ETSI GR PDL 028 (Study on Utilising PDLs to Standardized IoT Service Layer Platform oneM2M)	Interworking between oneM2M and PDL	2024-10
NOTE: This column should be filled according to the contents of table 1.			

1.3.3 Other topics in scope

List here any additional topic for which the PoC plans to provide input/feedback to the ISG ENI.

Table A.3

PoC Topic Description	Related WI	Expected Contribution	Target Date
Validate use cases that utilize PDL in oneM2M	related to ETSI GR PDL 028 (Study on Utilising PDLs to Standardized IoT Service Layer Platform oneM2M)	Demonstration to validate PDL utilization in oneM2M	2024-06

1.4 PoC Project Stages/Milestones

Table A.4

PoC Milestone	Stages/Milestone description	Target Date	Additional Info
P.S	PoC Project Start	2024-05	
P.D1	PoC Demo 1	2024-06	PDL#18, F2F Develop an NFT-based ticket system to prevent ticket scalping
P.D2	PoC Demo 2	2024-10	PDL#19, F2F Develop an NFT-based ticket system and gate access control system using PDL and oneM2M
P.C1	PoC Expected Contribution	2024-06	PDL features to support NFT-based ticket system – New Work - Item
P.C2	PoC Expected Contribution 2	2024-10	oneM2M interworking proxy application – Revision of the System architecture
P.R	PoC Report	2025-01	
P.E	PoC Project End	2025-01	

2 PoC Technical Details

2.1 PoC Overview

A oneM2M access authentication system based on a Permissioned Distributed Ledger (PDL) can address many vulnerabilities by leveraging the security and immutability features of blockchain technology. IoT platforms that utilize a PDL and NFT to prevent scalping—using macros to buy tickets in bulk and resell them at a profit—and to prevent ticket tampering can provide enhanced access control policies. Additionally, implementing two-factor authentication (2FA) through the PDL and the oneM2M standard can effectively prevent scalping and ticket tampering, ensuring that access control systems operate securely and reliably.

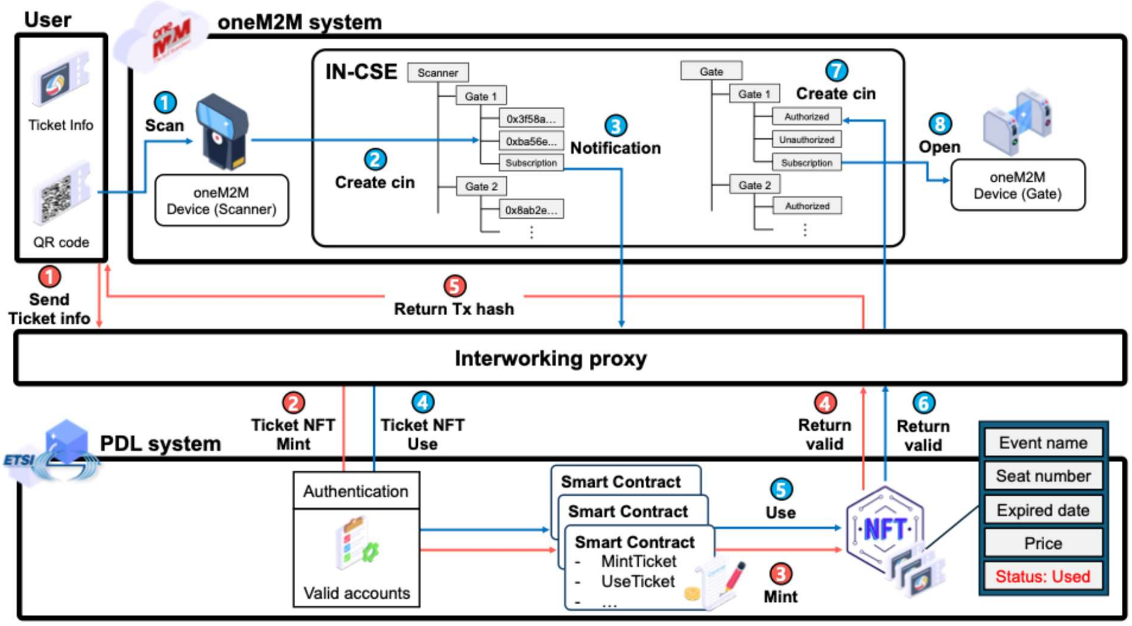

Figure 1. PDL-based oneM2M access authentication system overview

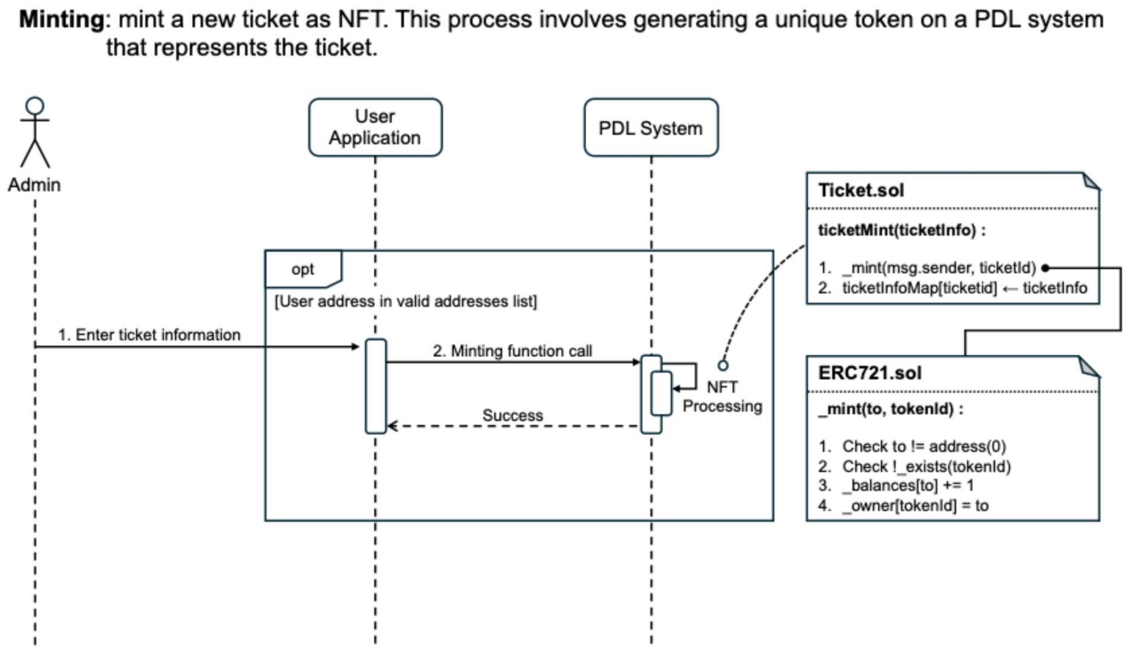
Minting: mint a new ticket as NFT. This process involves generating a unique token on a PDL system that represents the ticket.
Admin
User Application
PDL System
opt
[User address in valid addresses list]
1. Enter ticket information
2. Minting function call
Success
NFT Processing
Ticket.sol
ticketMint(ticketInfo) :
1. _mint(msg.sender, ticketId)
2. ticketInfoMap[ticketid] ← ticketInfo
ERC721.sol
_mint(to, tokenId) :
1. Check to != address(0)
2. Check !_exists(tokenId)
3. _balances[to] += 1
4. _owner[tokenId] = to

Figure 2. Ticket minting sequence diagram

Figure 2 shows the ticket NFT minting sequence diagram in the proposed system. The process involves generating a unique token on a PDL system that represents the ticket. After the minting process, validate tickets can be created with NFTs through the PDL system to access.

Figure 3 provides details on NFT minting and how to create QR tickets.

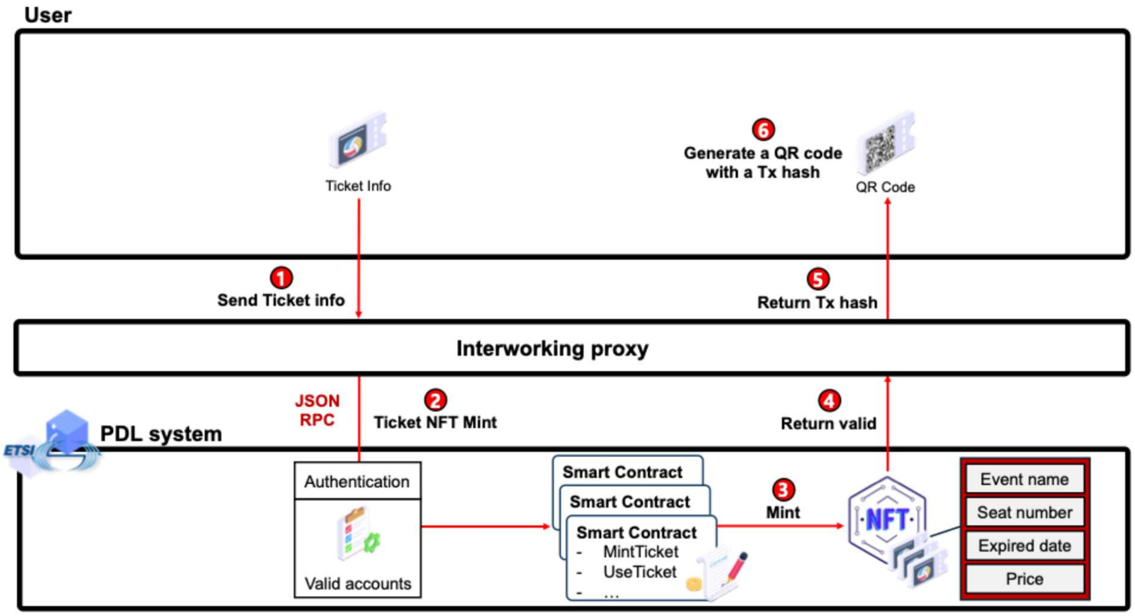

Figure 3. Ticket minting sequence details

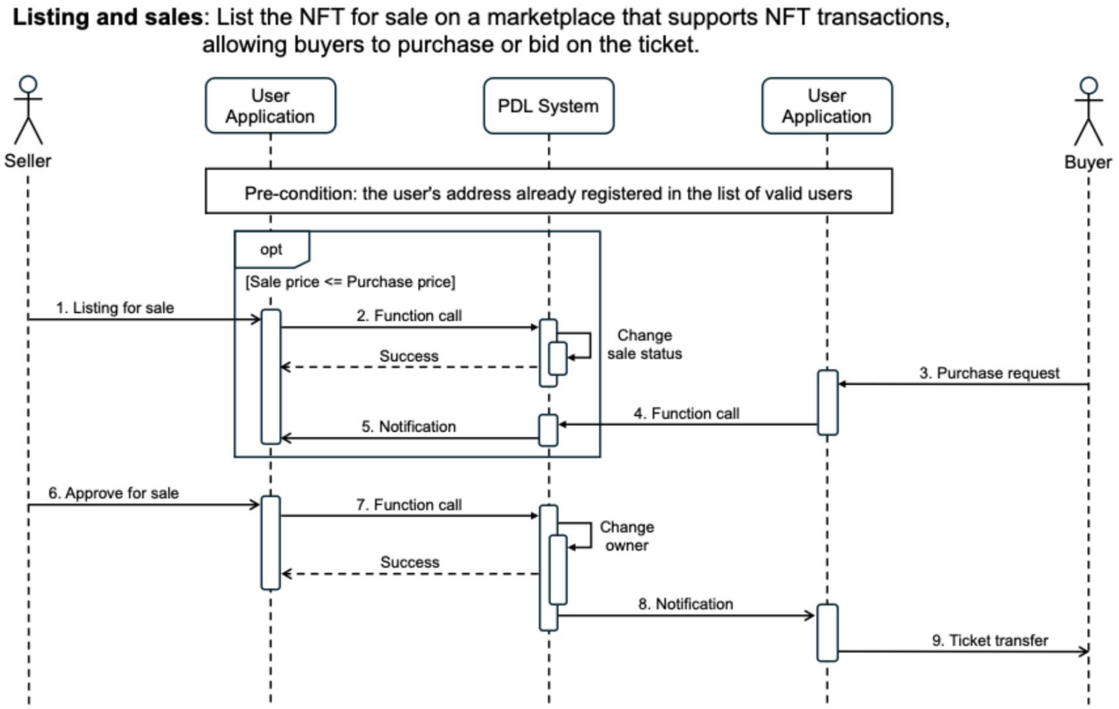

Figure 4. Ticket listing and sales sequence diagram to prevent scalping

Figure 4 shows a sequence diagram of the ticket listing and sales function. The procedure describes how to list the created NFTs for sale on a marketplace or ticket applications that support NFT transactions, allowing buyers to purchase or bid on the ticket. To prevent ticket scalping, the system does not allow NFT tickets to be set higher than the purchase price when reselling them.

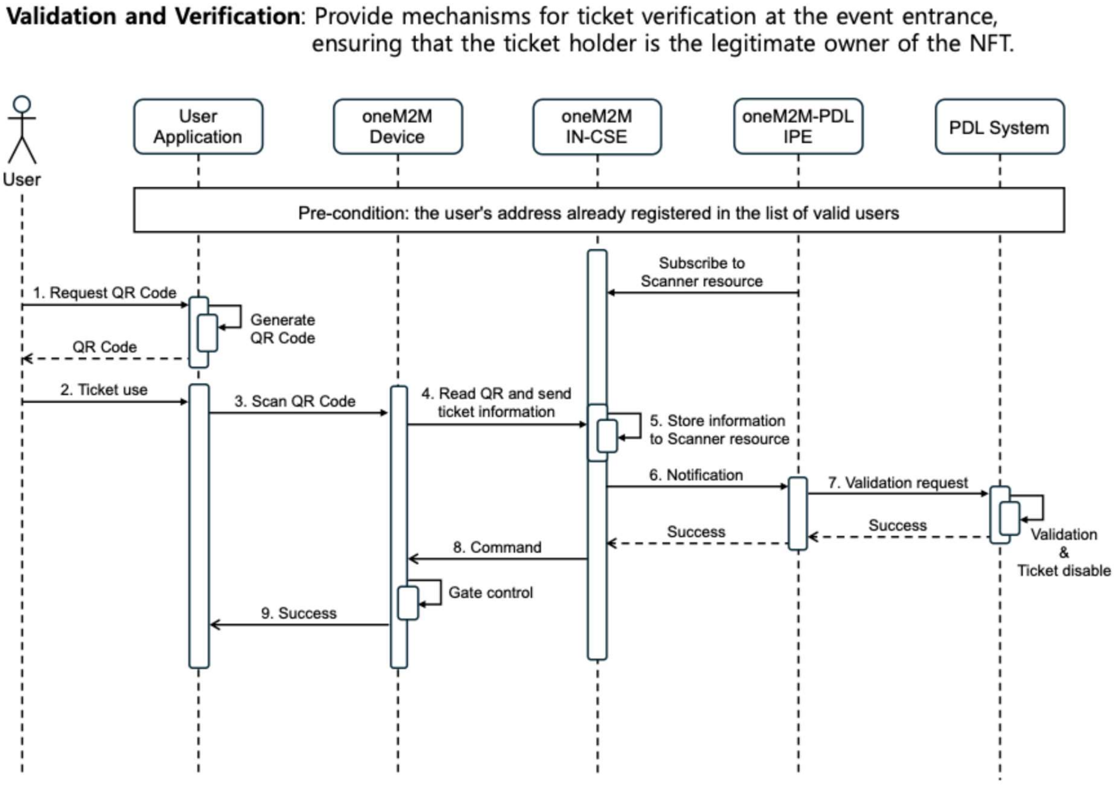
Validation and Verification: Provide mechanisms for ticket verification at the event entrance, ensuring that the ticket holder is the legitimate owner of the NFT.
User
User Application
oneM2M Device
oneM2M IN-CSE
oneM2M-PDL IPE
PDL System
Pre-condition: the user's address already registered in the list of valid users
Subscribe to Scanner resource
1. Request QR Code
Generate QR Code
QR Code
2. Ticket use
3. Scan QR Code
4. Read QR and send ticket information
5. Store information to Scanner resource
6. Notification
7. Validation request
Success
Success
Validation & Ticket disable
8. Command
Gate control
9. Success

Figure 5. Ticket validation and verification sequence diagram for PDL-based oneM2M access authentication

Figure 5 shows a sequence diagram of the PDL-based access authentication scheme in oneM2M. oneM2M can verify the validation of a ticket through PDL.

Figure 6 provides details on how to use NFT tickets, verify the validation, and access control.

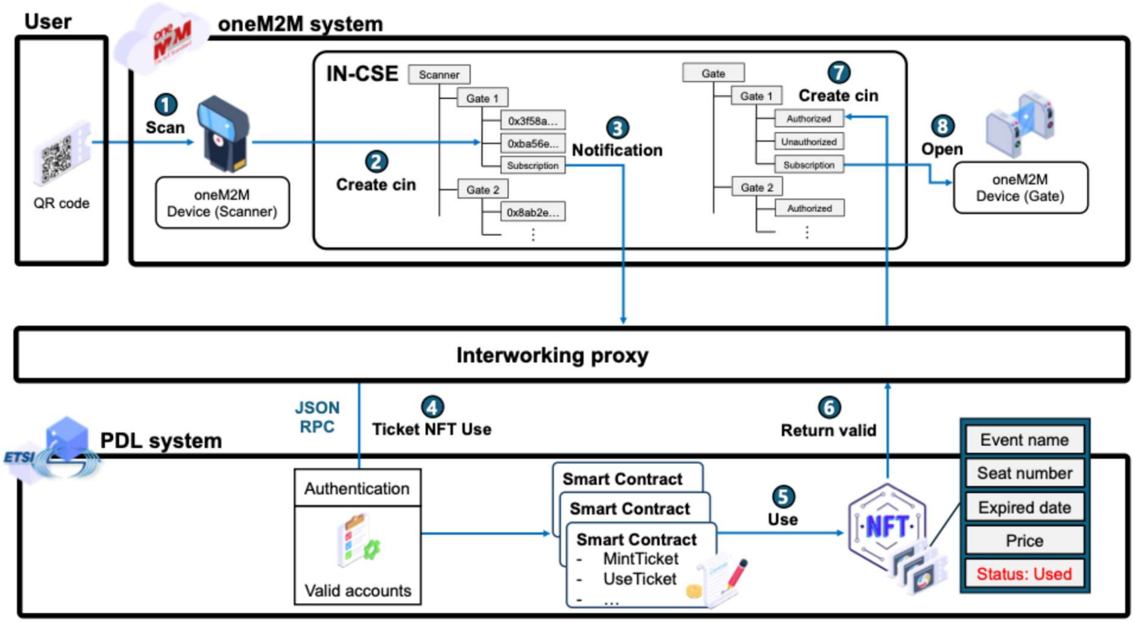

Figure 6. Ticket validation and verification sequence details for PDL-based oneM2M access authentication

2.2 PoC Architecture

The detailed PDL-based access authentication system PoC simple architecture is shown below.

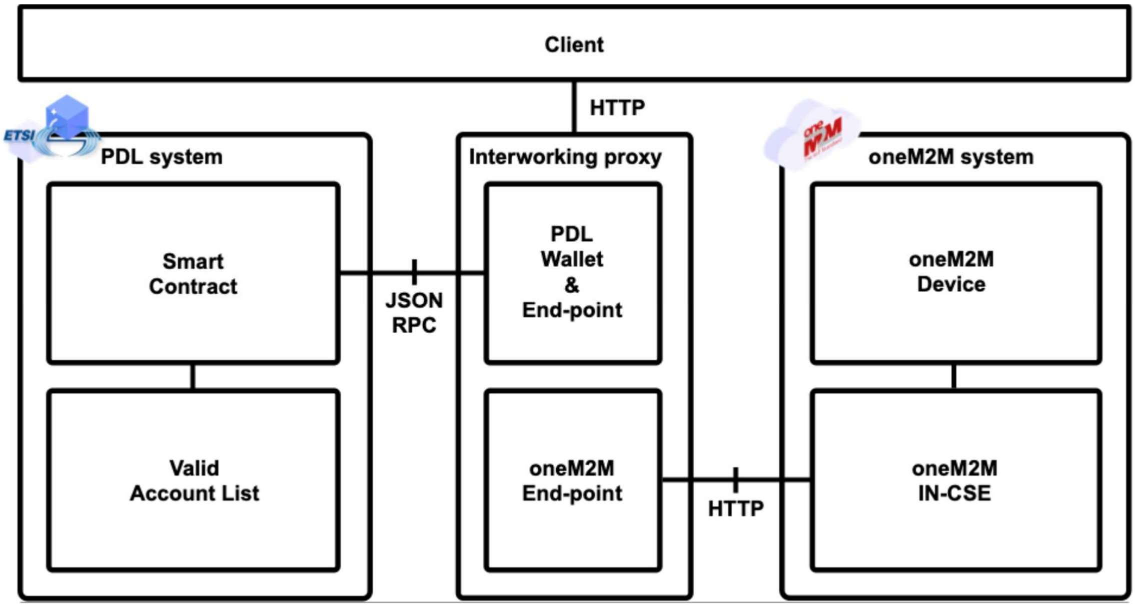

Figure 7. PDL-based access authentication system simple architecture

The oneM2M-PDL Interworking Proxy Entity (IPE) acts as an intermediary that connects the oneM2M platform and the PDL platform, facilitating bidirectional message exchange. The IPE can communicate with the connected oneM2M platform using HTTP, CoAP, MQTT, or WebSocket, which are methods defined by the oneM2M standard. It also acts as an endpoint to receive notifications by applying a subscription to the oneM2M system. For communication with the PDL platform, it uses standard libraries provided by the target PDL platform to exchange messages.

The IPE promptly sends data received from oneM2M entities to the PDL, ensuring the integrity and reliability. The PDL uses smart contracts to automate the validation and execution of the data, returning the results to IPE. During this process, the IPE monitors the success of the smart contract in real time, controlling the opening and closing of the physical gate based on the results. In particular, the IPE achieves high throughput and low latency by employing asynchronous data processing and an event-based architecture, enabling it to maintain stable performance even in large-scale IoT environments.

With this comprehensive design and implementation, the oneM2M-PDL IPE playa a crucial role in building the oneM2M-PDL service by efficiently and reliably managing the collection, verification, execution, and control of IoT data. It is expected to contribute to the construction of various smart infrastructures such as smart cities and smart factories by integrating more functions and services in the future.

Table 1. oneM2M-PDL IPE API list

API name	Linked PDL smart contract functions	Parameter	Short description
/ticket/minter/remove	removeMinter	account	Revokes the MINTER_ROLE from a specified address, effectively removing their permission to mint new tickets. Only accounts with the DEFAULT_ADMIN_ROLE can execute this function.
/ticket/minter/add	addMinter	account	Grants the MINTER_ROLE to a specified address, allowing them to mint new tickets. This function can only be called by accounts holding the DEFAULT_ADMIN_ROLE.
/ticket/use	ticketUse	ticketId	Marks a specific ticket as used. Only the owner of the ticket can invoke this function. It ensures that the ticket is not expired and hasn't been used previously before updating its status to prevent reuse.
/ticket/sell	sellTicket	ticketId, price	Allows the owner of a ticket to list it for sale at a specified price. The function verifies that the ticket is valid, not expired, not already sold, and that the new sale price does not exceed the original price. Upon successful validation, it updates the ticket's sale status and price.
/ticket/mint	mintTicket	eventName, seatNumber, expiredDate, price	Creates (mints) a new ticket with the provided event details, seat number, expiration date, and price. Only addresses with the MINTER_ROLE can execute this function. It ensures that the ticket's expiration date is set in the future and initializes the ticket's status as available for sale and usable.
/ticket/has-role	hasRole	role, address	Checks if a given address possesses a specific role within the contract. Returns true if the address has the role, otherwise false. This function is inherited from the AccessControlUpgradeable contract and is used to manage role-based access control.
/ticket/change-role	changeRole	role, oldAddress, newAddress	Transfers a specific role from one address (oldAddress) to another (newAddress). It first verifies that the oldAddress currently holds the role and that the newAddress does not already have it. Only accounts with the DEFAULT_ADMIN_ROLE can perform this role reassignment.
/ticket/buy-request	ticketRequestToBuy	ticketId	Allows a user to express interest in purchasing a specific ticket. It ensures that the ticket is listed for sale and that the requester is not the current owner. Upon successful request, it records the buyer's address in the ticketPurchaseRequests mapping for further processing.
/ticket/buy-confirm	ticketBuy	ticketId	Completes the purchase of a ticket that a user has previously requested. It verifies that the caller has indeed requested to buy the ticket and is approved to do so. Upon validation, it safely transfers ownership of the ticket to the buyer, updates the ticket's sale status, and removes the purchase request record.
/ticket/approve-sale	approveSale	ticketId	Approves a pending sale request for a specific ticket. Only the current owner of the ticket can call this function. It ensures that there is an active purchase request before approving the sale

			by setting the approved address, allowing the buyer to proceed with the purchase.
/ticket/info	getTicketInfo	ticketId	Retrieves detailed information about a specific ticket, including the event name, seat number, price, expiration date, sale status, and usage status. This function allows users to query the contract for the current state of a particular ticket.
/pdl-ipe/noti	ticketUse	Transaction hash	Receive a json-formatted notification from oneM2M and run the ticketUse function with the corresponding ticketId via the transaction hash value.
/wallet	X	name, address, privateKey	It takes your name, address, and private key and saves the wallet address. The stored address will be used for future API calls.
/wallet/{name}	X	name	Returns information about the corresponding address based on the name.
/qr/generate	X	Transaction hash	Generate a QR code with the transaction hash value.

2.3 PoC Success Criteria

The success of the PoC is demonstrated by the verification of the interaction between oneM2M-PDL through the demonstration. It does not include non-functional requirements such as performance or availability goals. Consequently, validation of the functional use case via a demo is sufficient.

2.4 Additional information

No additional information