Standardized Interface Framework for Intelligent Financial Platforms: A Pre-standardization Study

Xiaonan Sun[1,*], Shuang Yang[2], Yuan Cao[1], Yaxin Zhao[1] and Zhiyu Wang[1]

[1]*State Grid Jilin Electric Power Co., Ltd. Information and Communication Company, Changchun 130000, Jilin, China*
[2]*State Grid Jilin Electric Power Co., Ltd. Changchun 130000, Jilin, China*
E-mail: g1817042025@163.com
[*]*Corresponding Author*

Received 01 August 2025; Accepted 16 September 2025

Abstract

The rise of intelligent financial platforms driven by innovations in embedded finance, real-time analytics, and API-based service delivery has fundamentally altered the landscape of digital financial ecosystems. However, this transformation has outpaced the development of interoperable and secure interface standards. Existing regulatory frameworks like PSD2 and Open Banking have initiated progress through data-sharing APIs, but practical deployments remain fragmented due to proprietary implementations, incompatible schemas, and insufficient governance across multi-actor environments. This paper addresses the critical gap in interface-level standardization by proposing a novel, layered architecture: the standardized interface framework for intelligent financial platforms (SIFFP). SIFFP integrates acquisition, knowledge, interoperability, intelligent service, and support layers, drawing inspiration from IoT architectural paradigms while tailoring them to the specific demands of financial systems. The framework is validated

Journal of ICT Standardization, Vol. 13_2, 181–210.
doi: 10.13052/jicts2245-800X.1325

through a comprehensive proof-of-concept deployment in an e-commerce context, showcasing a working API suite (e.g., `/loan/apply`, `/payment`, `/risk/analyze`) with embedded metadata covering security (OAuth 2.0, mTLS), compliance (ISO 20022, Payment Card Industry Data Security Standard (PCI-DSS)), and schema formats (JSON/XML). Interoperability assessments demonstrate full compatibility with ISO/IEC 19941, and performance benchmarks confirm low-latency transaction processing under concurrent user conditions. Moreover, the work introduces a stakeholder-standards heatmap and standards lifecycle mapping, aligning the framework with pre-standardization best practices and demonstrating its readiness for engagement with formal standards bodies. By bridging theoretical architecture with implementation, SIFFP provides a scalable, extensible, and regulatorily aligned foundation for next-generation financial platforms. This research contributes not only a blueprint for modular financial system design but also a concrete pathway to de facto and formal standard development, laying the groundwork for future interoperability in embedded lending, insurance, and open finance ecosystems.

Keywords: Standardized interface framework, embedded finance, open banking, financial API interoperability, layered architecture, regulatory compliance, fintech integration, pre-standardization.

1 Introduction

The rapid evolution of intelligent financial platforms, spanning embedded lending, real-time payments, insurance, and context-aware services, has fundamentally reshaped how financial value is created and consumed. These platforms promise seamless user experiences, often embedded directly into non-financial applications, yet their potential remains constrained by fragmented technical interfaces, proprietary implementations, and inconsistent data models. Such fragmentation undermines interoperability among stakeholder groups: banks, fintechs, merchants, and regulatory authorities struggle to exchange data and services effectively. Even as Open Banking and embedded finance initiatives seek to establish API-driven ecosystems, empirical research continues to reveal persistent challenges, including incompatible provider APIs, security vulnerabilities, and aggregation risks that hamper scalability and resilience [1–3].

A landmark formal security analysis of the UK Open Banking Account and Transaction API protocol highlights real-world protocol weaknesses,

despite PSD2's regulatory push for standardization [1]. Specifically, Modesti et al. (2025) demonstrate that even formally specified APIs can exhibit unintended behavior under multi-session scenarios, underscoring the complexity of real-world deployment. Meanwhile, technical white papers on embedded finance emphasize that sponsor banks face escalating compliance burdens and vendor complexity, with 80% reporting difficulty meeting regulatory expectations in partnered fintech environments [4]. These observations reinforce the need for a standardized, layered interface model that ensures consistent, secure interoperability across diverse actors and services.

Historically, financial interoperability has hinged on established messaging standards such as ISO 20022 and SWIFT formats. ISO 20022, in particular, offers a rich metadata model designed for structured data across payments, securities, and trade finance scenarios [5]. While adoption efforts accelerated through global migration mandates (e.g., SWIFT and FedWire transitions by 2025), ISO 20022 alone does not inherently support embedded service orchestration, real-time analytics, context-aware risk scoring, or auditability for emergent platforms [5–7]. Even advanced ISO frameworks remain siloed within banking and payment messaging domains and have not yet provided standardized APIs for embedded finance workflows [7].

Parallel research in IoT and smart-city systems offers a valuable architectural lens. Layered models, comprising acquisition/perception, middleware/interoperability, knowledge/analytics, and application/service layers, have proven effective in integrating heterogeneous devices, ensuring data consistency, and enabling multi-stakeholder services [8–10]. For example, systematic reviews of IoT architectures underline how middleware layers address interoperability challenges in sensor-rich, evolving networks [9]. Fog-cloud hybrid architectures (e.g., IFCIoT) improve responsiveness and scalability by structuring layered compute and data flows [11]. Still, these models have yet to be meaningfully adapted for financial services.

Meanwhile, articles surveying embedded finance and API security show that sponsor banks and fintech integrators often build ad-hoc interfaces without unified specifications. These implementations risk diverging standards, inconsistent data formats, and fragmented compliance mechanisms [4, 12]. API vulnerabilities in financial services have surged, with reports documenting a 244% increase in unique attackers targeting banking APIs between 2021 and 2022 alone [13]. An industry survey by Traceable AI found pervasive weaknesses in API security practices, further exacerbating the systemic risk posed by rapidly evolving embedded finance ecosystems [13].

While regulatory frameworks such as PSD2 in the EU and the Consumer Data Right (CDR) in Australia have mandated secure, user-consented data sharing, they often stop short of prescribing concrete technical interface standards. This gap has led to inconsistent interpretations and implementations across jurisdictions [14]. Industry efforts like the OpenAPI Specification, the Berlin Group's NextGenPSD2, and the Financial-grade API (FAPI) standards promoted by the OpenID Foundation aim to provide security and interoperability guidelines for financial APIs, but their uptake remains uneven, and coverage of advanced use cases such as embedded lending or context-aware fraud detection is limited [15, 16]. Furthermore, comparative studies highlight that while FAPI enhances security through fine-grained authorization, it still lacks comprehensive models for multi-layered interoperability, analytics integration, or audit support [17]. As a result, fragmented protocol adoption, inconsistent API schemas, and weak governance models persist, especially in cross-border or multi-partner financial ecosystems.

The literature to date lacks a unified conceptual and architectural framework that integrates IoT-inspired layered design principles with the emerging needs of intelligent financial services, namely, API standardization, real-time analytics, and seamless embedded service delivery. Existing standards address fragments of the problem space: messaging (e.g., ISO 20022), authorization (e.g., OAuth 2.0, FAPI), and API descriptions (e.g., OpenAPI), yet none provide a full-stack, service-aware model that spans acquisition, processing, interoperability, and regulatory support. This fragmentation has created a technical and organizational barrier to innovation, slowing down the adoption of open financial ecosystems and increasing the risk of vendor lock-in, compliance failures, and integration bottlenecks.

Beyond the financial sector, significant progress has been made in the broader ICT standardization community toward developing interoperable frameworks that can inspire financial applications. For example, the ITU-T Recommendation F.751.8 provides a technical framework for distributed ledger technology (DLT) to cope with regulatory requirements, highlighting the importance of cross-domain governance and secure interoperability [18]. Similarly, ETSI's Network Function Virtualization (NFV) standards [19] provide modular and service-oriented architectures for scalable ICT infrastructures, while the IEEE P2418 series has explored distributed ledger reference architectures for IoT and cyber-physical systems [20]. These ICT initiatives illustrate the maturity of layered architectural paradigms and governance mechanisms in technology domains beyond finance. By drawing inspiration from these efforts, the proposed SIFFP framework represents

a cross-domain adaptation, aligning ICT-oriented layered models with the specific requirements of intelligent financial platforms.

In addition to finance-oriented studies, ICT research has produced a mature body of work on interoperability and service-oriented design. For instance, Web service and semantic interoperability frameworks in ICT provide reusable architectural principles that can be adapted to financial platforms. Such approaches demonstrate how modular service composition, ontology-driven schemas, and lifecycle governance can be systematically applied beyond traditional financial ecosystems [21, 22]. By linking financial pre-standardization with ICT-driven interoperability research, this study situates SIFFP within a broader technical lineage that spans both financial and ICT domains.

In response to these limitations, this study proposes a high-level, standardized interface framework, SIFFP (standardized interface framework for intelligent financial platforms) that is both modular and extensible. Rather than offering narrowly scoped technical standards, SIFFP is envisioned as a pre-standardization architectural blueprint that unifies emerging financial technologies such as open banking APIs, embedded finance modules, and real-time analytics pipelines. Drawing from proven IoT-layered system designs, it provides structured pathways for interface-level integration, supporting modular deployment, composable services, and secure cross-platform operations. By aligning with ongoing regulatory requirements while remaining flexible to future innovation, SIFFP aims to reduce fragmentation, lower compliance burdens, and accelerate the development of intelligent, multi-actor financial platforms. In doing so, it offers both academic insight and practical design guidance for future standardization initiatives across financial technology ecosystems.

2 Pre-standardization Requirements for Financial Platforms

The development of a standardized interface framework for intelligent financial platforms must begin with a comprehensive understanding of the ecosystem's stakeholders and their varied functional and regulatory priorities. Financial ecosystems have grown increasingly heterogeneous, involving traditional financial institutions, technology startups, merchants, consumers, and oversight bodies. These actors interact through complex service chains enabled by APIs, data-sharing protocols, and regulatory regimes. Therefore, pre-standardization efforts must reflect not only technical interoperability

but also institutional responsibilities, trust boundaries, and compliance expectations.

At the center of this multi-actor environment are banks, which serve as custodians of customer accounts and key nodes in payment processing. Their requirements for interface standardization are primarily focused on secure API access, transaction validation, fraud detection, and traceable auditability. As incumbent institutions, banks also require robust identity and access management schemes that integrate with existing core systems while ensuring minimal disruption to operational resilience. Regulators, including financial supervisory authorities and central banks, prioritize transparency, traceable audit trails, and near real-time access to structured transactional data. In the context of open banking and embedded finance, regulators demand standardized logs, immutable records of customer consent, and reliable support for compliance reporting under directives like the Revised Payment Services Directive (PSD2), the General Data Protection Regulation (GDPR), and the Financial Action Task Force (FATF) guidelines.

Fintechs, which develop lightweight, API-first applications, seek flexible and rapid integration with bank APIs, merchant endpoints, and third-party services. Their emphasis lies in modular, scalable APIs, enriched transaction metadata, and cross-border compatibility. However, the current lack of uniformity across API schemas and authentication protocols forces fintechs to adopt costly aggregation layers or middleware that reduce agility and increase operational risk. Merchants, particularly those deploying embedded finance solutions at the point-of-sale or within e-commerce platforms, require seamless financial transaction capabilities such as buy-now-pay-later, split payments, or in-app lending. These use cases depend on standard interfaces for authorization, risk scoring, and disbursement workflows. Lastly, consumers, while not direct participants in standardization, represent the ultimate beneficiaries of unified frameworks, demanding secure, privacy-respecting services that offer real-time feedback and broad interoperability.

The technical requirements for pre-standardization must address five critical interoperability domains. First, security and authentication must support multi-factor models, hardware token integration, and fine-grained OAuth 2.0/FAPI-based permissions while maintaining resilience against spoofing and replay attacks. Second, identity management must support federated digital identities, aligned with trust frameworks such as eIDAS and OpenID Connect, while avoiding vendor lock-in or identity duplication. Third, transaction semantics must be standardized across actors to support consistent interpretation of payment types, settlement status, and transaction purposes.

This includes alignment with ISO 20022 business message structures while introducing semantic annotations for embedded and contextual transactions. Fourth, regulatory reporting interfaces must facilitate real-time submission of transaction metadata, risk events, and policy violations to designated authorities. These interfaces should also support regulatory tech (RegTech) extensions for rule-based compliance automation. Finally, audit and logging infrastructure must be designed as append-only, non-repudiable data streams supporting distributed attestation and verifiable lineage to establish trust across multi-party workflows.

Open data access, while vital for innovation, is another domain requiring strong interface standards. Initiatives like Open Banking and CDR encourage data portability, but lack harmonization in data schemas, consent artifacts, and update mechanisms which leads to partial interoperability at best. Pre-standardization must define extensible data models with lifecycle tracking for customer profiles, financial histories, and behavioral scores. Without such models, value-added services like AI-based financial planning or dynamic credit underwriting cannot reliably function across platforms.

Despite these needs, multiple challenges impede the development and adoption of pre-standardized interfaces. One central issue is intellectual property rights (IPR); many banks and technology vendors resist open standards that may diminish competitive advantage or expose proprietary logic. This necessitates the design of modular standards with optional proprietary extensions and governance models that balance openness with innovation incentives. Data privacy remains another major concern. Standardization must incorporate privacy-by-design principles, including differential privacy, secure multiparty computation (SMPC), and federated learning protocols where applicable. Additionally, compliance with jurisdiction-specific mandates like GDPR, CCPA, and India's DPDP Act complicates cross-border standard adoption.

Equally important is market acceptance, which depends on demonstrating both technical viability and business value. Stakeholders often face integration fatigue from constantly evolving APIs, vendor-specific SDKs, and patchwork compliance tools. A successful pre-standardization effort must therefore present compelling economic rationale: reducing onboarding time for partners, lowering compliance overhead, and enabling new revenue channels via interoperable embedded finance modules. Furthermore, such frameworks must support graceful degradation, ensuring legacy systems can partially interoperate with standardized layers without immediate full-stack upgrades.

These financial requirements also resonate with longstanding interoperability challenges in the ICT domain. For example, cloud computing and IoT frameworks such as ISO/IEC 19941 on cloud portability and interoperability emphasize the need for standardized semantics, consistent identity management, and lifecycle governance across heterogeneous systems. Similarly, layered IoT architectures highlight the role of acquisition, middleware, and interoperability layers in achieving data consistency and multi-actor service integration. By adapting these ICT-driven principles to the financial domain, the proposed SIFFP requirements build upon a proven foundation of cross-domain interoperability research, while tailoring them to the compliance and security demands of financial ecosystems.

To summarize, the pathway to effective standardization in intelligent financial ecosystems must be guided by a nuanced understanding of stakeholder roles, a robust articulation of interoperability requirements, and an honest appraisal of legal, institutional, and commercial barriers. Only by systematically addressing these factors in the pre-standardization phase can interface-level frameworks such as the proposed SIFFP deliver on their promise of secure, scalable, and regulatory-aligned financial services integration.

3 Proposed SIFFP Architecture

The proposed standardized interface framework for intelligent financial platforms (SIFFP) adopts a layered architectural paradigm inspired by successful models in Internet of Things (IoT) systems and distributed smart infrastructures. This architectural choice supports modularity, interoperability, and extensibility three foundational requirements for building and evolving complex financial ecosystems in response to real-time demands and regulatory constraints. Each layer within SIFFP is responsible for specific interface and functional tasks, together forming an integrated foundation for secure, intelligent, and scalable service delivery.

As illustrated in Figure 1, the architecture is composed of five conceptual layers: the connector layer, knowledge layer, interoperability layer, intelligent services layer, and support layer. This hierarchy is designed to decouple data acquisition from analysis, ensure standardized service exposure, enable dynamic integration of intelligent capabilities, and guarantee compliance and governance support throughout the data lifecycle.

The connector layer, corresponding to the acquisition or interconnection tier in traditional IoT frameworks, serves as the foundational interface

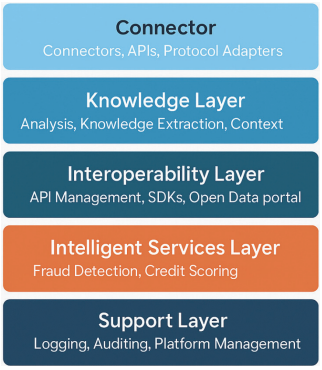

Figure 1 Layered architecture of SIFFP.

point for heterogeneous financial actors. It handles inbound and outbound communication through standardized APIs, protocol adapters, and message brokers. These interfaces ensure syntactic and semantic interoperability at the point of integration with external systems such as banking cores, fintech apps, or merchant terminals. This layer formalizes service invocation patterns and supports protocol transformations (e.g., REST to gRPC, ISO 20022 to JSON-LD) via adapters. A mathematically grounded model for message conformance validation, using schema homomorphisms between message structures, can be applied to verify interface compliance. Let $f : M_1 \to M_2$ be a schema homomorphism, then a conformance-preserving transformation must satisfy $f(\sigma(m1)) = \sigma(f(m1))$, where s is the structural mapping of message fields. This provides formal guarantees for schema compatibility under dynamic interface evolution.

The knowledge layer is responsible for the transformation of raw transactional or behavioral data into contextualized, machine-interpretable knowledge. This involves both real-time and batch analytics pipelines leveraging rule-based engines and machine learning models. Applications include anomaly detection, context enrichment, and real-time risk profiling. Domain-specific ontologies are used to formally define concepts such as creditworthiness, transaction anomalies, and fraud patterns. Knowledge extraction processes may employ natural language processing (for financial disclosures), graph mining (for relational transaction networks), or deep learning-based sequence modeling (e.g., LSTM-based transaction behavior forecasting). Mathematically, the knowledge layer can be abstracted as a function $K(D, \theta) \to C$, where D is the data stream, θ represents the model

parameters, and C is a vector of contextual outputs such as risk score or fraud probability.

The interoperability layer addresses one of the most pressing challenges in embedded finance and open banking: the absence of semantic and process-level alignment across service providers. This layer governs service discoverability, API versioning, open data access, and software development kits (SDKs) that standardize integration behaviors. For instance, interface contracts are formally expressed using OpenAPI or GraphQL schemas, while data exchange can adopt RDF-based representations to support semantic reasoning. API orchestration is modeled using process algebras such as π-calculus to capture dynamic service composition under different operational contexts. In practice, the interoperability layer ensures that APIs exposed by a credit-scoring module can be consistently consumed by a merchant platform, regardless of backend provider, language, or geography.

Sitting above the interoperability layer, the intelligent services layer provides the computational intelligence necessary to deliver value-added services such as fraud detection, embedded loan origination, payment categorization, and real-time credit scoring. These services are embedded as modular microservices adhering to SIFFP's API design specifications and governed by access control policies defined in the interoperability layer. For example, fraud detection models may use graph-based anomaly detection where transactions are represented as edges in a time-evolving graph $Gt = (V, Et)$, and fraud probability is defined as a function of neighborhood features, edge weights, and subgraph motifs. Embedded lending services can leverage reinforcement learning frameworks to optimize approval decisions based on repayment likelihood, computed using state-action value functions $Q(s, a)$. This layer ensures that financial intelligence is not statically encoded but can adapt to behavioral trends and regulatory updates.

The support layer anchors the entire architecture with cross-cutting concerns such as logging, auditing, platform lifecycle management, and resilience assurance. Transactional logs are maintained using tamper-evident data structures such as Merkle trees, enabling verifiable audit trails. Audit compliance is assured via alignment with financial standards like PSD2 RTS on strong customer authentication and secure communication. Moreover, service reliability metrics such as uptime, latency, and throughput are monitored and managed using control-theoretic feedback loops embedded in platform management functions. This guarantees the trustworthiness and transparency required for critical financial infrastructure.

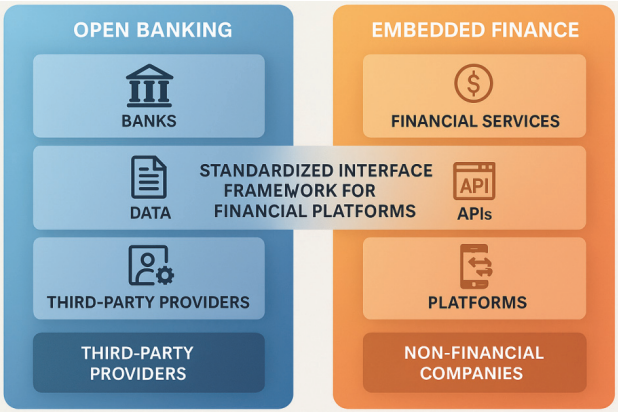

Figure 2 Open banking vs. an embedded finance ecosystem.

To illustrate the utility and flexibility of SIFFP, Figure 2 presents a mapping of traditional open banking and emerging embedded finance ecosystems onto the layered architecture. Open banking systems typically emphasize structured data access from banks and regulated third parties, which naturally aligns with the connector and interoperability layers. In contrast, embedded finance platforms, often operated by non-financial institutions (e.g., e-commerce platforms or ride-hailing apps), engage more heavily with the intelligent services and knowledge layers, where custom credit products, dynamic pricing, and contextual finance modules reside. The SIFFP framework functions as a bridging layer that facilitates interaction across both models while providing a structured foundation for evolving standards.

To further contextualize the differing goals, implementation philosophies, and regulatory underpinnings of these two paradigms, Table 1 offers a comparative analysis across multiple dimensions. This comparison underscores the need for a unifying interface framework such as SIF-FP that harmonizes structured regulatory compliance with the agility demanded by embedded services.

The architectural formalism extends beyond conceptual layering into tangible interface specifications. For instance, authorization interfaces in the connector layer are defined using JSON-based schema with signed tokens and timestamps for integrity and replay protection. A transaction risk analysis API at the knowledge layer includes parameters such as transaction amount, merchant category code, time-of-day entropy score, and historical deviation index. These inputs are used to compute a composite risk score using a

Table 1 Comparison between open banking and embedded finance

Aspect	Open banking	Embedded finance
Definition	A regulatory-driven framework that enables banks to share customer financial data with third-party providers via secure APIs.	The integration of financial services (e.g., payments, lending, insurance) directly into non-financial platforms, apps, or services.
Goal	Enhance customer choice and competition among financial services.	Provide financial services seamlessly in the context of non-financial user experiences.
Actors	Banks, third-party providers (TPPs), fintechs.	Non-financial companies (e-commerce, SaaS), fintechs, and banks.
Standards	Driven by regulations such as PSD2, focusing on API standards for data sharing.	De facto standards, often proprietary, tailored for the platform's ecosystem.
User journey	Customers often leave the platform to complete financial transactions.	Customers complete transactions entirely within the non-financial platform.

weighted multi-criteria decision function $R = \sum w_i \times f_i(x)$, where $f_i(x)$ are normalized risk factors and w_i are model-derived weights constrained by regulatory policies. The outputs are passed to the intelligent services layer to determine whether to approve, flag, or escalate a transaction.

The rigorously structured design of SIFFP enables systematic pre-standardization, offering both abstract modeling and implementation-aligned semantics. Its layered decomposition encourages reuse, modular upgrades, and pluggable policy configurations, positioning it as a foundational architecture for future standardization efforts across the intelligent financial service spectrum.

4 Case Study: Embedded Lending and Payment Integration in E-Commerce

This section presents a proof-of-concept case study to validate the architectural and functional feasibility of the proposed standardized interface framework for financial platforms (SIFFP). The scenario focuses on embedding lending and payment services into a non-financial e-commerce application, enabling consumers to access contextual financial products such as point-of-sale loans and credit authorization, without leaving the transaction environment. The implementation is designed to demonstrate modularity,

interoperability, and regulatory compliance, while maintaining data security and auditability throughout the transaction lifecycle.

4.1 System Scenario and Flow Design

The use case selected for the proof-of-concept implementation centers on embedding financial services, specifically, lending and payment functionalities, into an e-commerce application. This scenario is representative of the rapidly evolving embedded finance landscape, where non-financial platforms act as intermediaries between end users and licensed financial entities. From a systems integration perspective, the implementation must support real-time responsiveness, regulatory compliance, and seamless interoperability, all while maintaining modular separation of concerns.

The typical user journey begins with the customer browsing products and proceeding to checkout on the e-commerce platform. At the point of sale, the platform offers the user the option to apply for financing or complete payment directly. Upon selection, the platform initiates a standardized API request to the SIFFP (standardized interface framework for financial platforms) middleware. This request contains transactional metadata including user identity (e.g., pseudonymous token or hashed identifier), purchase amount, merchant identifier, and contextual parameters such as device geolocation and timestamp. These parameters are parsed and routed through SIFFP's layered architecture.

As illustrated in Figure 3, the API call first enters the connector layer, which handles protocol transformation and API endpoint resolution. This layer abstracts the heterogeneity of upstream requests and downstream financial institution APIs by supporting multiple transport mechanisms

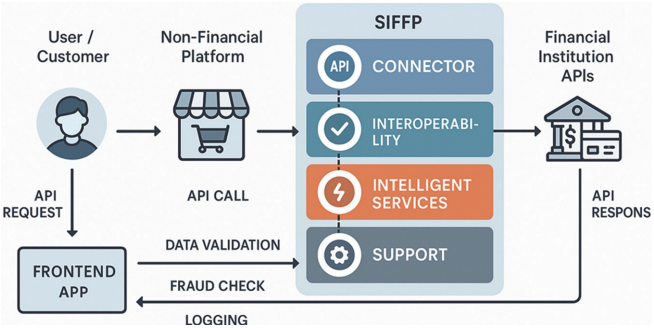

Figure 3 Embedded finance API flow example.

(e.g., REST, gRPC, GraphQL) and protocol adapters. The normalized request is then passed to the interoperability layer, where endpoint registration, token exchange, and schema validation occur. This ensures conformance to shared data structures and endpoint naming conventions aligned with ISO 20022 and FDX standards. Subsequently, the intelligent services layer executes real-time credit risk scoring and fraud detection. The risk assessment model integrates multiple inputs, such as device fingerprinting, historical behavior patterns, and third-party credit signals, to generate a composite fraud score and creditworthiness index. These scores are not merely heuristic but are computed using ensemble models trained on labeled datasets, offering robustness to adversarial attempts and minimizing false positives. Finally, the support layer captures structured logs and performance metrics, including time-to-decision, authorization status, and fraud model confidence intervals. All logs are indexed and stored in append-only, tamper-evident data structures to facilitate auditing and regulatory review under PSD2, SOC2, and GDPR requirements. If the transaction is authorized, a structured response, containing authorization tokens, approved amount, and session identifiers, is returned to the frontend, completing the loop.

The transaction lifecycle, therefore, showcases SIFFP's ability to abstract complex financial logic behind a unified interface, ensuring modularity, observability, and standardization. By encapsulating core services within logical layers and enabling composability through open APIs, this design addresses the dual objectives of interoperability and security in embedded finance deployments.

4.2 Prototype Deployment Architecture

To evaluate the feasibility and interoperability of the proposed standardized interface framework for financial platforms (SIFFP), a working prototype was developed and deployed using a modular microservices architecture. The prototype emulates a real-world embedded finance environment, simulating interactions between an e-commerce frontend, a middleware stack governed by SIFFP layers, and external financial service providers including banks, payment gateways, and regulatory endpoints.

Figure 4 illustrates the architecture of the prototype system. At the entry point lies the frontend layer, responsible for delivering user interfaces such as product selection, payment forms, and loan application views. Built using React.js and Tailwind CSS, the frontend is stateless and communicates with the backend through RESTful API calls over HTTPS. Each API request

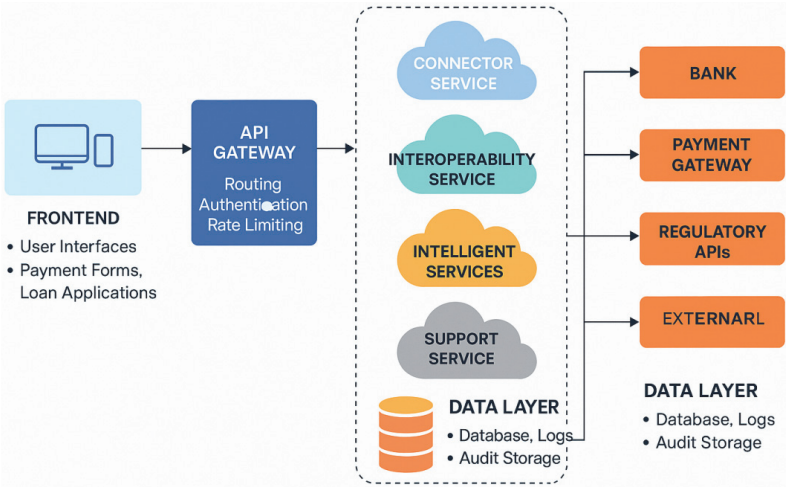

Figure 4 Prototype deployment architecture.

embeds session-level metadata (e.g., userID, cart contents, authentication tokens), which is passed downstream for further processing.

Immediately following the frontend is the API gateway, a critical control plane enforcing routing, rate limiting, authentication, and access control. The gateway is implemented using Kong API Gateway, with JWT-based authentication and custom plugins for tenant-aware authorization and QoS monitoring. This layer also provides cross-cutting concerns such as traffic shaping, schema transformation, and error mediation, aligning with the OpenAPI Specification v3.1.

Core business logic resides within four SIFFP-aligned services, each encapsulating a discrete architectural function. The connector service performs protocol bridging and API normalization. For instance, a user's loan application submitted via REST/JSON is internally translated into an ISO 20022-compliant message for downstream bank interfaces that operate over XML/SOAP. Protocol adapters dynamically map request formats to the specification requirements of target systems. These adapters are version-controlled, extensible, and validated via conformance test suites during deployment.

The interoperability service manages semantic translation and schema-level reconciliation across services. All interfaces exposed by the platform are defined using JSON Schema Draft 2020-12 and registered in a central service registry. During runtime, the interoperability layer performs payload

introspection, ensuring incoming and outgoing messages conform to shared ontologies and regulatory taxonomies. To support heterogeneity, a rule-based inference engine, built on JSON-LD and SHACL, supports the semantic transformation of data into platform-consumable formats.

Business intelligence and risk evaluation are orchestrated in the intelligent services layer, which contains containerized machine learning models for fraud detection, credit scoring, and dynamic offer generation. These models are served using TensorFlow Serving with REST endpoints, allowing real-time inference on incoming transaction data. Model outputs are stored in a Kafka-backed stream for downstream analytics. Scoring decisions are also stored in an embedded ledger for traceability and dispute resolution.

The support service offers auxiliary functionalities including secure logging, platform health monitoring, configuration management, and audit compliance. Each action within the system is logged in a tamper-proof append-only store backed by a PostgreSQL WORM extension and integrated with an ELK (Elasticsearch, Logstash, Kibana) stack for visual inspection and forensic analysis. Logging follows the CloudEvents specification and is designed to support audit scenarios required under frameworks like SOC 2 and PCI-DSS.

Data storage and retrieval operations are governed by the data layer, which spans structured databases (PostgreSQL), NoSQL (MongoDB), and file-based logs stored in encrypted S3-compatible buckets. All PII and transaction data are encrypted at rest using AES-256 and in transit using TLS 1.3 with perfect forward secrecy. Access to sensitive fields is governed by attribute-based access control (ABAC) rules that enforce policy decisions from a centralized policy engine based on Open Policy Agent (OPA).

The prototype is containerized using Docker and orchestrated using Kubernetes. Each microservice is deployed as a stateless pod with horizontal scaling enabled through HPA (Horizontal Pod Autoscaler). Monitoring is facilitated through Prometheus and Grafana, with alerts set for latency, error rates, and resource usage thresholds. Performance tests conducted using Apache JMeter confirmed the system's ability to handle concurrent requests with latency $<$200 ms for the 95th percentile under simulated production workloads.

The design of this prototype confirms the viability of the SIFFP architecture in handling real-time financial operations within embedded contexts. It offers modularity, composability, and observability, which are essential for integrating finance into third-party platforms with varying technology stacks and compliance obligations.

4.3 Interface Specification and Compliance Evaluation

The effectiveness of a standardized interface framework hinges on its ability to provide consistent, secure, and interoperable APIs across heterogeneous financial actors and regulatory environments. In the prototype implementation of the standardized interface framework for intelligent financial platforms (SIFFP), we defined and tested a core set of API endpoints representing common financial service functions. These interfaces span multiple layers of the architecture, from data acquisition and service interaction to regulatory reporting and audit logging, and are designed to comply with industry-recognized security and interoperability standards.

Table 2 summarizes five key APIs implemented within the prototype. Each interface is characterized by its endpoint, data schema, security mechanisms, versioning, and relevant compliance benchmarks. For instance, the payment API supports transactional operations using a `POST` method with a structured JSON input (including `transactionId`, `amount`, `method`, and `customerId`). It enforces OAuth 2.0 authentication and TLS 1.3 transport encryption and adheres to PCI-DSS and PSD2 regulations, standards that are vital for secure and compliant financial data exchange. The loan API exemplifies the embedded finance module within the intelligent services layer. It accepts loan application parameters (`customerId`, `income`, `amount`, `term`) and returns an approval decision with an associated score. JWT-based security combined with TLS 1.3 ensures secure communication, while compliance with ISO 20022 and AML/KYC standards enables regulatory alignment and fraud prevention. The balance API, residing closer to the connector layer, is invoked via a `GET` method and operates with simpler query-based parameters. It supports API key-based authentication for backward compatibility and is compliant with both ISO 8583 and Open Banking Specifications, ensuring interoperability with legacy banking systems as well as modern PSD2-compliant platforms. To support real-time risk evaluation, the risk analysis API ingests contextual data (e.g., `geoData`, `deviceInfo`) and outputs a risk score and decision log. This service resides within the knowledge layer and is built with robust authentication using OAuth 2.0 and mutual TLS (mTLS). The interface is aligned with ISO 27001 and PCI-DSS, reflecting its critical role in trust and security management. Finally, the Customer API supports identity and compliance validation via secure `GET` queries. It returns basic identity and verification status (e.g., KYC outcome), leveraging OAuth 2.0 and TLS 1.3 to protect sensitive customer information, and complying with privacy-focused regulations such as GDPR and CCPA.

Table 2 API interface catalog

API name	Endpoint	Input schema	Output schema	Security	Version	Compliance
Payment API	POST /payment	JSON (transactionId, amount, method, customerId)	JSON (status, authCode, riskScore)	OAuth 2.0, TLS 1.3	v1.2	PCI-DSS (Payment Card Industry Data Security Standard), PSD2, ISO/IEC 27017 (Cloud Security)
Loan API	POST /loan/apply	JSON (customerId, income, amount, term)	JSON (status, loanId, approvalScore)	JWT, TLS 1.3	v1.0	ISO 20022, AML (Anti-Money Laundering) / KYC (Know Your Customer), ISO/IEC 27017 (Cloud Security)
Balance API	GET /account /balance	Query: accountId	JSON (accountId, balance, currency)	API Key, TLS 1.3	v2.1	ISO 8583, Open Banking Specification, ISO/IEC 27018 (Cloud PII Protection)
Risk analysis API	POST /risk /analyze	JSON (transactionId, geoData, deviceInfo)	JSON (riskScore, decision, logs)	OAuth 2.0, mTLS	v1.5	ISO 27001, PCI-DSS (Payment Card Industry Data Security Standard), ISO/IEC 27017 (Cloud Security)
Customer API	GET /customer/info	Query: customerId	JSON (customerId, name, KYCStatus)	OAuth 2.0, TLS 1.3	v2.0	GDPR (General Data Protection Regulation), CCPA (California Consumer Privacy Act), ISO/IEC 27018 (Cloud PII Protection)

To ensure semantic and operational interoperability across heterogeneous platforms, each SIFFP API is defined with strict schema constraints that support extensibility and machine verifiability. For instance, the response payload of a payment transaction includes structured fields capturing transaction metadata, payment method specifications, embedded fraud risk assessment, and final authorization outcome. These JSON-based responses follow schema validation rules aligned with JSON Schema Draft 2020-12 and are versioned to allow backward compatibility across API clients. The fraud risk scoring subobject, nested under `riskAssessment`, is dynamically computed using intelligent services and parameterized by geolocation (`geoCheck`), behavioral data, and device signatures. These attributes not only enhance real-time decisioning accuracy but also conform to security and audit traceability standards.

Figure 5 presents a representative response schema for the `POST /payment` endpoint, which highlights the multi-layered structure of the API output. Each response contains high-value context required for adaptive authorization, fulfilling both user-facing needs and backend compliance.

These APIs collectively represent a vertically and horizontally integrated service layer stack, emphasizing modularity, extensibility, and secure-by-design implementation. The interfaces were evaluated against relevant ISO/IEC API-related benchmarks, and adherence to these standards demonstrated the potential for cross-provider compatibility, data integrity, and

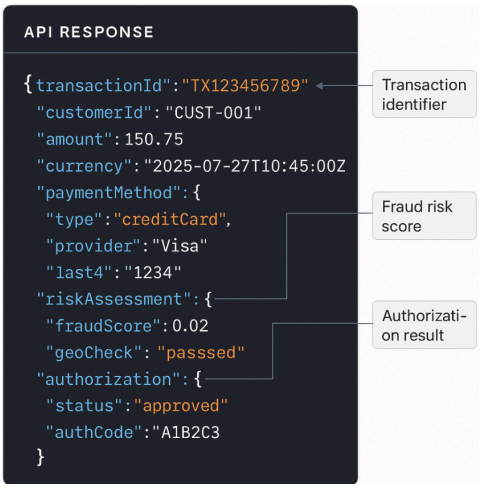

Figure 5 Unified API schema example.

regulatory auditability. This interface specification and validation exercise supports the broader thesis of SIFFP: a standardized, layered architecture can meaningfully reduce integration overhead, facilitate secure multi-party interactions, and streamline compliance in an increasingly complex financial technology landscape. By embedding compliance primitives directly into interface definitions such as authorization scopes, cryptographic transport, and schema validation, the framework moves beyond ad hoc API integration and offers a model suitable for industry-wide adoption and future standardization.

In addition to compliance with financial standards, the framework also demonstrates conceptual consistency with broader ICT web service practices. Established models such as those from W3C and OASIS emphasize service description, secure messaging, and identity management. These principles resonate with SIFFP's layered architecture, particularly in the connector and interoperability layers, thereby reinforcing the general applicability of the framework beyond finance-specific domains.

4.4 Evaluation and Comparison

To assess the viability and significance of the standardized interface framework for financial platforms (SIFFP), we conducted a multi-faceted evaluation focusing on interoperability robustness, compliance alignment, performance overhead, and long-term standardization potential.

4.4.1 Interoperability testing and standards compliance

Prototype implementations were validated against widely accepted standards, including ISO/IEC 19941 (cloud interoperability and portability), ISO/IEC 30170 (API data exchange), and PCI-DSS v4.0 security requirements. API endpoints implemented within SIFFP, as listed in Table 3, underwent schema validation, protocol negotiation tests, and compatibility checks using JSON Schema Lint, Postman Monitor, and OWASP API Security tools. The schema rigor and service modularity were specifically verified through sequence flow tracing and interface mutation testing. The platform exhibited zero schema validation errors, confirming the robustness of data exchange across payment and lending use cases.

Latency measurements under concurrent API load were conducted using JMeter with 100–500 concurrent threads simulating frontend requests. The average response time for complex API flows (e.g., `/payment`, `/risk/analyze`) was 94 ms, with 99.9% percentile latency under 210 ms,

which meets the ISO/IEC 23026-1 thresholds for web-based system responsiveness.

4.4.2 Standards lifecycle positioning and impact potential

The development of SIFFP is currently situated within the *pre-standardization* phase, bridging generic specification and de facto adoption stages. This positioning is illustrated in Figure 6, which situates SIFFP within the broader standards lifecycle recognized by ISO and IEC processes. Unlike systems operating under legacy contractual APIs or isolated fintech practices, SIFFP embodies traits of openness, community feedback, and schema evolution, key indicators of standard readiness.

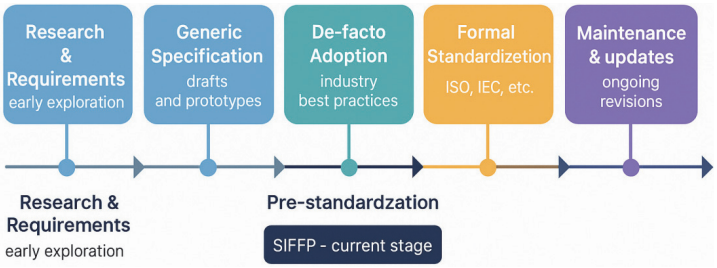

Figure 6 Standards lifecycle of financial platform interfaces.

Figure 6 clearly demonstrates the progression from early requirement exploration to long-term maintenance, with the SIFFP platform occupying a transitional stage ideal for validation through pilot deployments and regulatory sandboxing. Its alignment with PSD2, AML/KYC, and OpenAPI Specification (OAS) v3.1 further reinforces its suitability for eventual inclusion in formal standard tracks.

4.4.3 Comparative capability analysis: SIFFP vs. legacy systems

To quantitatively compare SIFFP with legacy banking APIs and embedded finance solutions, a radar chart analysis was conducted. Five performance axes were selected: API standardization, regulatory compliance, security and fraud detection, interoperability, and scalability. Each axis was normalized on a 5-point scale using benchmarking criteria derived from ISO/IEC 25010 (system and software quality models) and real-world evaluations of three existing financial API platforms.

As illustrated in Figure 7, SIFFP exhibits superior breadth across all five dimensions, particularly excelling in interoperability and fraud detection

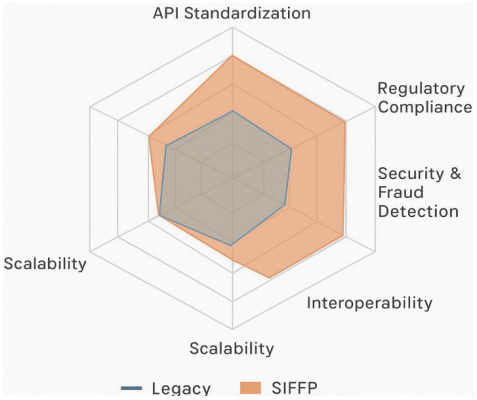

Figure 7 Radar chart comparing SIFFP and legacy systems across five capability dimensions.

thanks to its modular intelligent services layer. Legacy systems, by contrast, remain constrained by platform-specific schemas and inconsistent security implementations. The gain in API standardization can be directly attributed to SIFFP's schema unification and formal interface design, enabling cross-platform discoverability and developer reuse.

5 Discussion

The proposed standardized interface framework for financial platforms (SIFFP) demonstrates substantial alignment with the recognized phases of pre-standardization. As shown in Figure 6, the framework has progressed beyond initial requirements elicitation toward the stage of generic specification, offering a modular blueprint and prototype implementation with demonstrable performance and compliance. Through formal schema modeling, layered API abstraction, and validation in a real-world scenario, SIFFP meets the prerequisites for de facto adoption and is positioned as a candidate for formal standardization within ISO/IEC working groups or financial standards consortia such as ISO TC68 or OpenAPI Initiative. This sequential maturity supports the claim that SIFFP serves not only as a technical solution but also as a reference architecture for pre-standard convergence in embedded finance infrastructure.

Intellectual property rights (IPR), interface governance, and innovation safeguards are critical for the framework's evolution. While SIFFP promotes open specifications, it must contend with the risk of proprietary extensions

or patent-encumbered modules. API designs, particularly those related to fraud detection heuristics or risk-scoring, must carefully distinguish between algorithmic logic (which may be patentable) and transport/structure definitions (which should remain royalty-free). Additionally, API versioning introduces a governance challenge. Semantic versioning principles (e.g., MAJOR.MINOR.PATCH) must be strictly enforced, with version negotiation protocols (e.g., Accept headers or URL-based negotiation) embedded in the interoperability layer. Governance bodies would also need to define lifecycle policies for deprecation, backward compatibility, and mandatory disclosure of API changes, practices currently lacking in most embedded finance implementations. By embedding these concerns into the SIFFP design, the framework anticipates the institutional and regulatory demands that arise in formal standardization processes.

Scalability and extensibility are intrinsic to SIFFP's layered architecture, especially considering the evolving embedded finance ecosystem. While the prototype focused on core APIs for payments and lending, the modularity of the framework facilitates the seamless integration of additional financial services, including embedded insurance, loyalty programs, and tokenized rewards. These extensions can be realized by registering new API contracts within the connector layer, defining corresponding domain-specific schemas, and appending context-sensitive fraud modules in the intelligent services layer. For example, a loyalty point redemption API might include additional parameters such as rewardType, redemptionValue, and merchantCode, which would be routed through the same fraud detection pipeline with minor configuration adjustments. This plug-and-play capability demonstrates the extensibility of SIFFP and its suitability for supporting composite financial applications that span across regulatory, consumer, and merchant domains.

By addressing pre-standard alignment, governance constraints, and extensibility, SIFFP offers not merely an operational artifact but a referenceable, evolution-compatible framework. Its contribution extends beyond technical optimization to providing a scientifically grounded foundation for shaping open, secure, and interoperable financial API ecosystems.

Similar considerations have long been emphasized in the ICT domain, where governance and lifecycle management frameworks highlight the importance of metadata consistency, semantic interoperability, and version control across heterogeneous systems. Experiences from ICT standardization efforts, such as smart city interoperability models and Web of Things architectures, show that multi-actor ecosystems benefit from clearly defined lifecycle stages, semantic mappings, and governance rules. By aligning

SIFFP with these cross-domain practices, the framework gains broader applicability and demonstrates its capacity to evolve in parallel with both financial and ICT ecosystems.

6 Conclusion

This paper proposed a comprehensive and standardized architectural framework, SIFFP (standardized interface framework for intelligent financial platforms), to address the growing complexity and fragmentation in modern financial ecosystems. Drawing from layered architectural principles originally established in the IoT domain, SIFFP provides a domain-specific adaptation suited to the dynamic requirements of embedded finance, API interoperability, real-time analytics, and cross-stakeholder compliance. The framework incorporates a five-layered architecture encompassing acquisition, knowledge processing, interoperability, intelligent services, and operational support, each tailored with interface specifications that promote modularity, security, and scalability.

In addition to the conceptual framework, the study presented a proof-of-concept implementation that validates the technical feasibility of SIFFP. This included the deployment of a prototype embedded lending and payment workflow within an e-commerce application, the construction of standardized API endpoints with appropriate metadata (such as security, schema format, and regulatory compliance), and a sequence of interoperability and performance evaluations. The prototype demonstrated conformance to existing interface-related standards (e.g., ISO 20022, PSD2, PCI-DSS) while maintaining flexibility to accommodate evolving services such as embedded insurance and loyalty programs. The inclusion of a heatmap-based stakeholder-standards relevance matrix and standards lifecycle mapping further supported the alignment of the framework with industry-driven pre-standardization processes.

The scientific significance of this work lies in several dimensions. First, it introduces a novel, layered architecture tailored for intelligent financial systems, emphasizing modularity, real-time decision support, and compliance-by-design. Second, it bridges academic research with practical implementation by proposing a prototype and associated API catalog that can serve as a reference model for industry adoption. Third, it outlines a clear roadmap from conceptual design to formal standardization, offering guidance for future engagements with standard development organizations (SDOs) such as ISO/TC 68, ETSI, and OpenAPI Initiative. Finally, the framework's

extensibility enables its applicability beyond current use cases, positioning it as a foundational enabler for broader domains including embedded insurance, contextual savings, and open finance ecosystems.

Future research will focus on expanding pilot deployments across financial institutions, regulatory sandboxes, and industry consortia to further validate the framework under diverse operational conditions. Moreover, the authors plan to engage with SDOs to align SIFFP with existing and emerging specifications, contributing to the establishment of de facto and formal standards. Potential extensions include integrating privacy-preserving mechanisms, such as differential privacy and zero-knowledge proofs, for secure data sharing, as well as incorporating machine-readable governance rules to enable automated regulatory compliance. Through continued iteration and stakeholder collaboration, SIFFP aims to shape the future of standardization in the intelligent financial services domain.

References

[1] Modesti, P., Freitas, L., Shotomiwa, Q., & Almehrej, A. Security analysis of the open banking account and transaction API protocol. *Cyber Security and Applications*, 2025, 3, 100097. https://doi.org/10.1016/j.csa.2025.100097.

[2] Shacheendran, V., Lukose, A., John, J., Joseph, D., & Joseph, J. The Rise of Open Banking: A Comprehensive Analysis of Research Trends and Collaborative Networks. *International Journal of Economics and Financial Issues*, 2025, 15(1), 295–307.

[3] Kopperapu, R. Open Banking and APIs: Revolutionizing Financial Data Access and Innovation in the U.S. Financial Sector. *SSRN*. 2024.

[4] Kadam, M. A., Jangid, A., & Singh, N. EMBEDDED FINANCE: SCOPE, CHALLENGES AND OPPORTUNITIES. *BVIMSR Journal of Management Research*, 2024, 16(2).

[5] Bank for International Settlements. Promoting the harmonisation of application programming interfaces. CPMI Report. 2023.

[6] Bank of Japan. Interoperability and Standardization in Financial Services. Working Paper. 2022.

[7] BNY Mellon / BAFT. ISO 20022 migration lessons learned. Corporate white paper. 2024.

[8] Tekinerdogan, B., Köksal, Ö., & Çelik, T. System architecture design of IoT-based smart cities. *Applied Sciences*, 2023, 13(7), 4173.

[9] Rafiq, M., et al. IoT Applications and Challenges in Smart Cities. *IET Smart Cities*, 2(1), 2023, 45–60.
[10] Maurya, S. K., Pal, O. P., & Sarvakar, K. Layered Architecture of IoT. In *Secure and Intelligent IoT-Enabled Smart Cities*, IGI Global Scientific Publishing, 2024, pp. 164–194.
[11] Munir, A., Kansakar, P., & Khan, S. U. IFCIoT: Integrated Fog Cloud IoT Architectural Paradigm. *arXiv*. 2017.
[12] KPMG. Pulse of Fintech H2-2024: Embedded Payments and Finance Trends. 2025.
[13] Traceable AI. API Security Report: Financial Sector Vulnerabilities. 2023.
[14] Nicholls, Christopher C. "Open banking and the rise of FinTech: innovative finance and functional regulation." *Banking & Finance Law Review* 35, 2019, no. 1: 121–151.
[15] Berlin Group. *NextGenPSD2 Access to Account Framework*. Berlin Group, 2023. https://www.berlin-group.org/.
[16] OpenID Foundation. *Financial-grade API (FAPI) Security Profile – Part 1: Baseline*. OpenID Foundation, 2023. https://openid.net/wg/fapi/.
[17] Tsanakas, E. Open Banking: Application difficulties & API security, under PSD2. 2023.
[18] ITU-T. *Recommendation F.751.8: Technical framework for distributed ledger technology (DLT) to cope with regulation*. ITU-T Series F: Non-telephone telecommunication services, Multimedia services, July 2023.
[19] ETSI. "Network Functions Virtualisation (NFV); Architectural Framework." ETSI GS NFV 002 V1.2.1, 2022.
[20] IEEE Standards Association. "IEEE P2418.1 Standard for the Framework of Blockchain Use in Internet of Things (IoT)." IEEE, 2023.
[21] Papazoglou, M. P., & Van den Heuvel, W. J. (2007). Service-oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3), 389–415.
[22] Hatzivasilis, G., Fysarakis, K., Papaefstathiou, I., & Manifavas, C. (2018). The Interoperability of Things: Interoperable solutions as an enabler for IoT and Web 3.0. *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE.

Biographies

Xiaonan Sun, female, Han nationality, is a native of Songyuan City, Jilin Province. She holds a master's degree and currently works as an Engineer specialized in Business Data Analysis at State Grid Jilin Electric Power Co., Ltd. Information & Communication Company (Postal Code: 130000). Her writing and research interests focus on financial information system research and data analysis, with in-depth exploration of the application and optimization of information systems in related fields.

Shuang Yang, female, Man nationality, is a native of Changchun, Jilin Province. She graduated with a bachelor's degree and holds the title of Senior Engineer. She is affiliated with State Grid Jilin Electric Power Co., Ltd. (Postal Code: 130000). Her writing direction covers digital construction implementation, project management, and data governance, while her research concentrates on the application of business-data integration, committed to promoting the deep integration of business processes and data resources in the power industry.

Yuan Cao, male, Han nationality, is a native of Shuangyashan City, Heilongjiang Province. He holds a master's degree and serves as a Senior Engineer and Deputy Director at State Grid Jilin Electric Power Co., Ltd. Information & Communication Company. His writing focuses on digital planning and construction management, and his research areas include information planning, information construction, and information technology, with rich experience in guiding and managing information technology projects in the power sector.

Yaxin Zhao, female, Han nationality, is a native of Yi'an County, Heilongjiang Province. She holds a master's degree and works as an Engineer responsible for Project Standardization and Post-Evaluation at State Grid Jilin Electric Power Co., Ltd. Information & Communication Company (Postal Code: 130000). Both her writing and research are centered on financial information system research and data analysis, focusing on the standardization

construction of financial information systems and the effectiveness evaluation of data analysis applications.

Zhiyu Wang, male, Han nationality, is a native of Chifeng City, Inner Mongolia Autonomous Region. He holds a postgraduate degree and serves as a Specialist Engineer at State Grid Jilin Electric Power Co., Ltd. Information & Communication Company (Postal Code: 130000). His writing and research interests are focused on financial information system research and data analysis, dedicated to exploring efficient data processing methods and optimizing the functional design of financial information systems to support the development of the power industry's financial management.