



Introduction to APIs: Power of Interoperability

FDX is dedicated to unifying the financial industry around a common, interoperable, royalty-free standard for the secure and convenient access of permissioned consumer and business financial data: the FDX Application Programming Interface (FDX API).

FDX is a global 501(c)(6) nonprofit organization with no commercial interests operating in the US and Canada.

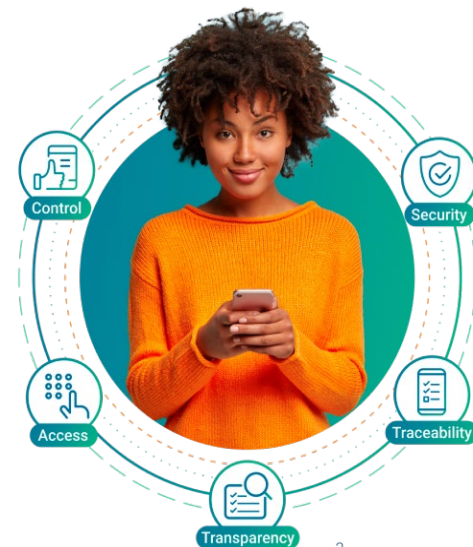
It is an independent subsidiary of the FS-ISAC.

Table of Contents

- Understanding Application Program Interface (API)
 - Consumer Interactions
 - What is an API?
 - What is Not an API?
 - APIs in Context of Developer Interactions
- Implementing APIs
 - API Protocols and Architectures
 - Understanding the Underlying Data Structures
 - Data Sharing Principles in API Integration
 - Introduction to OpenAPI
 - Implementation and Ecosystem of OpenAPI
 - API Design Patterns
 - API Maintenance and Versioning
- API Security
 - APIs and Fraud Prevention
 - Securing Your API
- API Use Cases
 - Financial Use Cases of APIs
 - FDX API's
 - Non-Technical Aspects of API Management
- Non-Technical Considerations
 - Non-Functional Attributes of APIs
 - Non-Technical Considerations for API Integration
- What are some API Opportunities
- References

Financial Data Exchange

- FDX is a non-profit organization that operates in the US and Canada, with the aim of bringing together the financial services industry through an **open, interoperable** and **royalty-free** technical standard for the sharing of **user-permissioned** financial data
- FDX upholds five core principles of financial data sharing to enable and empower users with their financial data and utilize it in a secure and dependable manner. These principles are **Control, Access, Transparency, Traceability and Security**
- 225 members and growing across the financial industry:
 - 76+ million consumer accounts are sourced through FDX API
 - > 5 Billion API calls per month
- FDX plays a pivotal role in advancing secure and standardized data sharing across the financial industry, promoting interoperability and enhancing consumer data protection.



FDX Specifications



API and Data Structures

1. Accounts and Transactions
2. Tax
3. Money Movement
4. Consent
5. Customers
6. Capability and Metrics



User Experience

1. Consent Grant
2. Consent Viewing
3. Consent Revocation
4. Consent Notifications
5. Taxonomy



Security

1. Secure AuthN & AuthZ
2. Security for Sensitive Data
3. Secure App Onboarding



Technical Certification

1. Provider Requirements
2. Recipient Requirements (TBD)
3. Certification Use Cases
4. Certification Model

Understanding Application Program Interface (API)

Consumer Interactions



E-Commerce

When you go to an E-Commerce website the Product information API pulls all the information (images, price, size, IDs, available amount, etc.) about a specific product from the database of all products in the store.



Travel Booking

When you're planning your next vacation and go online to look through the best deals for flights, hotels, car rentals, APIs are working behind the scenes on booking websites.



Financial Institutions

When you go to an ATM machine to withdraw cash from your bank account, APIs work as messengers between you and the bank. APIs check your account balance and communicate with the cash dispenser to hand you the requested amount of money.



Navigation

APIs identify the user's location, their destination and then plots the route. APIs even take constructions zones, blocked roads and traffic jams into account when planning the most efficient route to save time.

What is an API?

Foundations of Modern Financial Data Exchange

- API is the acronym for Application Programming Interface. It is software that allows different software applications to communicate with each other. They are a set of rules defining methods and data formats for information exchange within and across organizations.



What is Not an API?

Clearing Common Misconceptions



Not a Database: While APIs can be used to access databases, they are not databases themselves but rather the intermediary layer facilitating data exchange between applications and databases.



Not a User Interface: APIs operate in the background to connect software applications and do not include a graphical user interface (GUI); they enable functionality rather than provide a visual experience.



Not Exclusive to Web Services: Although commonly associated with web services, APIs encompass a broader range of technologies, including operating systems, databases, and hardware interactions, beyond just internet-based applications.

APIs in Context of Developer Interactions

Facilitating Developer Engagement and Integration



APIs Defined Through Documentation: APIs are typically defined by documentation specifying endpoints, request methods, parameters, response formats, and authentication requirements, serving as a comprehensive guide for developers.



Promoting Standardization and Efficiency: APIs standardize the way applications communicate, significantly enhancing development efficiency by allowing for easy integration, automated testing, and clear documentation of functionalities.

Implementing APIs

API protocols and architectures

Understanding the Spectrum

- RESTful APIs
 - The representational state transfer (REST) architecture is the most popular approach to building APIs. REST relies on a client/server approach that separates front and back ends of the API and provides considerable flexibility in development and implementation.
 - RESTful APIs are designed for web services communication, using standard methods like GET, POST, PUT, and DELETE to interact with resources.
- RPC
 - The remote procedural call (RPC) protocol is a simple means to send multiple parameters and receive results. RPC APIs invoke executable actions or processes, while REST APIs mainly exchange data or resources such as documents. RPC can employ two different languages, JSON and XML, for coding.
- GraphQL
 - GraphQL: Allows clients to request specific data they need, reducing over-fetching or under-fetching of data. Provides a flexible and efficient alternative to traditional RESTful APIs.
- SOAP APIs
 - SOAP (Simple Object Access Protocol) APIs rely on XML for message format and are known for their strict standards and security, often used in enterprise environments for complex transactions.

Understanding the Underlying Data Structures

Foundations of API Data



JSON (JavaScript Object Notation): A lightweight format for data interchange, utilizing key-value pairs and arrays for structure. Predominantly used in RESTful APIs for its ease of use and human readability.



XML (eXtensible Markup Language): A markup language that defines a set of rules for encoding documents in a format both human-readable and machine-readable. Used for complex data hierarchies and document structures.



Arrays and Lists: Represent ordered collections of items, facilitating the management of data sets. Arrays in JSON are denoted by square brackets.



Custom Data Structures: APIs may define bespoke data structures tailored to specific domain needs, representing complex entities or relationships.

Introduction to OpenAPI

Standardizing API Design and Documentation



OpenAPI Specification (OAS): is an API description format for REST APIs, enabling developers and **computers** to understand the capabilities of a service without accessing its source code.

Purpose of OpenAPI: Facilitates clear and precise API documentation, simplifies generation of client and server code, and supports automated testing and API discovery mechanisms.

Benefits: Promotes API usability and interoperability across different systems, enabling more efficient design, integration, and consumption of APIs.

Implementation and Ecosystem of OpenAPI

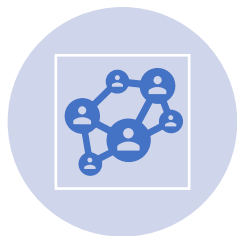
Empowering API Development and Integration



Tooling and Libraries: A rich ecosystem of tools and libraries for designing, testing, and generating client and server code, such as Swagger UI, Swagger Editor, and OpenAPI Generator.



API Gateways and Management: Integration with API gateways and management platforms to enforce policies, rate limiting, and access controls. Leverages OpenAPI definitions for seamless API lifecycle management.



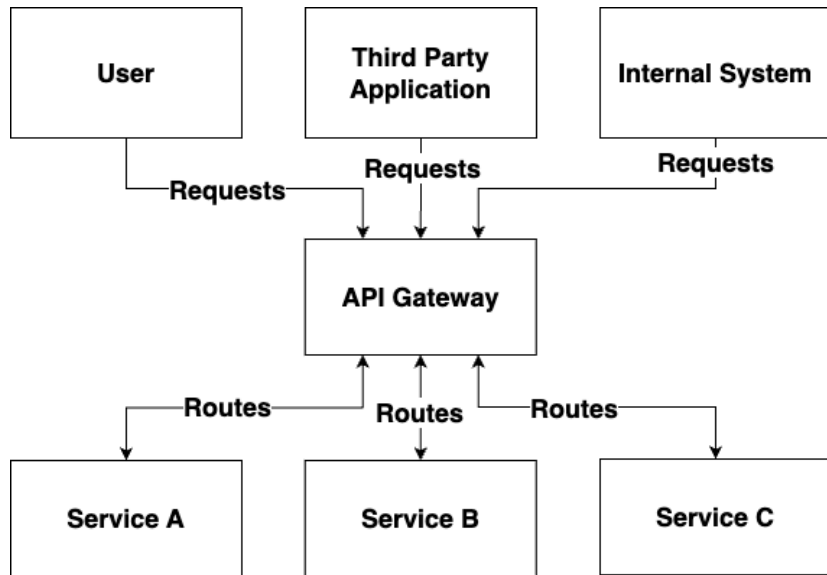
Community and Standards: A vibrant community contributing to the evolution of the OpenAPI Specification, ensuring it remains relevant and effective in addressing new API development challenges.



Use Cases: Widely adopted in industries such as finance, healthcare, and e-commerce for documenting APIs, automating workflows, and facilitating third-party integrations.

API Design Patterns

Strategies for Robust API Architecture



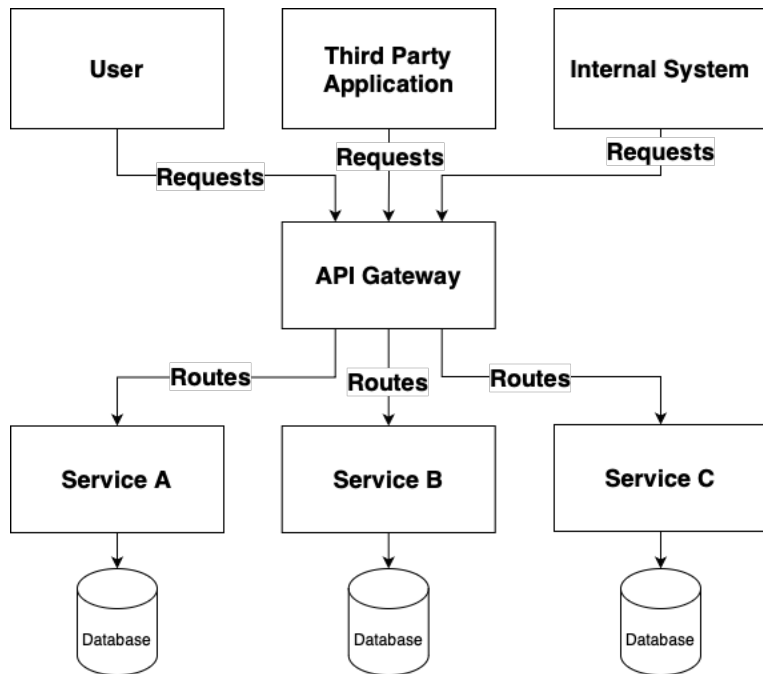
*Simplified view of this pattern.

API Gateway Pattern

Acts as a single entry point for all client requests, routing them to the appropriate microservice, and can also handle cross-cutting concerns like authentication, logging, and load balancing.

API Design Patterns

Strategies for Robust API Architecture



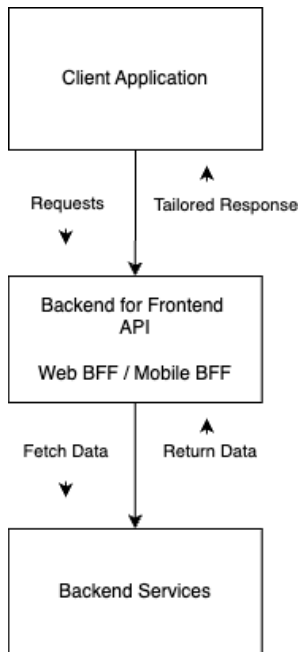
Microservices Architecture

Divides a larger application into small, loosely coupled services, each implementing specific business capabilities, allowing for flexibility, scalability, and independent deployment.

*Simplified view of this pattern.

API Design Patterns

Strategies for Robust API Architecture



*Simplified view of this pattern.

Backend for Frontend (BFF) Pattern

Creates specific backends for different types of clients (e.g., mobile, web) to tailor the API responses to the client's needs, optimizing the user experience by reducing the amount of data transferred.

API Maintenance and Versioning

Best Practices for Longevity and Compatibility

- **Versioning Strategies:** Employ versioning through URI paths, query parameters, or headers to manage changes and ensure backward compatibility.
- **Deprecation Policy:** Clearly communicate the lifecycle of API versions, providing ample notice before deprecating old versions to allow consumers to adapt.
- **Continuous Monitoring and Testing:** Regularly monitor API performance and usage, and implement automated testing to identify and fix issues promptly, ensuring reliability and security.

API Security

APIs and Fraud Prevention

Leveraging Technology to Combat Fraud

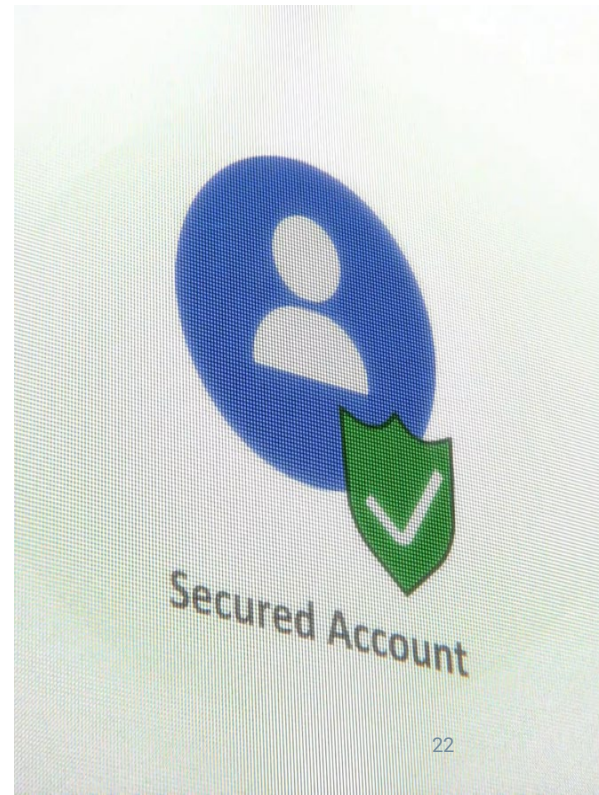
- **Authentication and Encryption:** APIs use strong authentication methods (e.g., OAuth, API keys) and encryption (HTTPS, MTLS) to secure communications and prevent unauthorized access.
- **Anomaly Detection:** Machine learning algorithms analyze API traffic patterns to detect and alert on unusual activities that could indicate fraud, such as abnormal transaction volumes or locations.
- **Transaction Monitoring:** Real-time monitoring of transactions through APIs enables the identification and blocking of suspicious transactions before they are processed, reducing the risk of fraud.
- **Data Sharing for KYC:** APIs facilitate secure data sharing among financial institutions for Know Your Customer (KYC) compliance, helping to verify identities and prevent identity theft.



Securing Your API

Essential Practices for Protection

- **Implement Strong Authentication:** Use robust authentication mechanisms like OAuth, JWT (JSON Web Tokens), to verify the identity of users and services accessing the API.
- **Employ Multifactor Authentication:** Improve confidence of who is performing user activities.
- **Employ HTTPS:** Ensure all API communications are encrypted using HTTPS to protect data in transit against interception and tampering.
- **Input Validation:** Validate all input data to prevent common vulnerabilities such as SQL injection, cross-site scripting (XSS), and other injection attacks.
- **Rate Limiting and Throttling:** Apply rate limiting to prevent abuse and reduce the risk of DDoS attacks. Throttling helps manage the load on the API and maintain service availability.
- **Regular Security Audits:** Conduct periodic security audits and penetration testing to identify and mitigate vulnerabilities, ensuring continuous improvement of security posture



API Use Cases

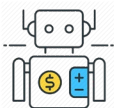
Financial Use Cases of APIs



Data sharing for Personal Financial Management (budgeting), Wealth Management, Robo-Advising, Tax prep)



Account, asset/holdings, balance, status, tenure verifications



Money Movement (Standing Orders/Bill Pay, internal transfers, wires (FedWire/SWIFT), ACH, RTP, other)



Apps and Aggregators can notify FI's of suspicious activity and provide indicators of compromise



Corporate / Treasury APIs for B2B, Accounting and ERP system integration



Payroll for verification of Income and employment

FDX API's



Consent APIs: Grant Consent, Retrieve Consent, Revoke Consent



Account APIs: Query information for a set of accounts



Reward-Program APIs: Get reward program information.



Customer APIs: Retrieve account holders related to consented accounts



Event Notification APIs: Creates and Get notification subscription



Fraud APIs: Notify Data Provider of suspected fraud



Money Movement Extension APIs: Get Payee information, Search Payments, Update Payments, Search Transfers



Tax APIs: Get the tax form image or data as JSON for a single tax document for the customer

*FDX APIs Align to the OPENAPI Standard



Non-Technical Aspects of API Management

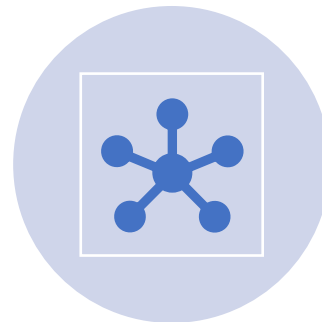
Beyond the Code



STAKEHOLDER ENGAGEMENT: EFFECTIVE COMMUNICATION WITH ALL STAKEHOLDERS, INCLUDING DEVELOPERS, BUSINESS LEADERS, AND END-USERS, TO ALIGN API INITIATIVES WITH BUSINESS OBJECTIVES AND USER NEEDS.



STRATEGIC PLANNING: INCORPORATING APIS INTO THE BROADER BUSINESS AND DIGITAL STRATEGY, CONSIDERING MARKET TRENDS, COMPETITIVE LANDSCAPE, AND POTENTIAL REVENUE MODELS.



ECOSYSTEM DEVELOPMENT: FOSTERING AN API ECOSYSTEM BY ENCOURAGING THIRD-PARTY DEVELOPERS TO BUILD APPLICATIONS ON TOP OF YOUR APIS, ENHANCING SERVICE OFFERINGS AND CUSTOMER EXPERIENCES.

Non-Technical Considerations

Non-Functional Attributes of APIs

Quality Attributes for API Performance



Performance Metrics: Includes response time, throughput, and latency, assessing how quickly and efficiently the API processes requests.



Scalability: Evaluates the API's capability to handle increased load, both vertically (adding resources) and horizontally (adding instances).



Reliability: Focuses on availability and fault tolerance, ensuring the API remains operational and recoverable from failures.



Security: Encompasses authentication, authorization, data encryption, and vulnerability scanning to safeguard against unauthorized access and data breaches.

Non-Functional Attributes of APIs

Quality Attributes for API Performance



Monitoring and Logging:

Involves tracking errors, auditing API interactions, and collecting metrics for performance insights.



Compliance and Standards:

Ensures adherence to API versioning standards and regulatory compliance for industry-specific regulations.



Documentation: Maintains the quality and currency of API documentation for effective use and integration by developers.



Resource Utilization:

Monitors CPU, memory, and bandwidth usage to optimize performance and resource allocation.



Capacity Planning:

Uses predictive analysis based on historical data and trends to forecast future capacity requirements.

Non-Technical Considerations for API Integration

Ensuring Successful Collaboration



Legal and Regulatory Compliance: Understanding and adhering to legal and regulatory requirements for data sharing, defining clear terms for data use, ownership, and responsibilities.



Data Governance and Privacy: Establishing policies for data management and protection, addressing privacy concerns in line with regulations like GDPR and CCPA.



Data Quality and Standards: Setting standards for data quality and agreeing on common data formats for seamless integration and interoperability.



Security Measures: Implementing robust security measures including encryption and access controls to protect shared data.



Data Ownership and Intellectual Property: Clarifying rights over data ownership and intellectual property, setting terms for commercial use and licensing.

Non-Technical Considerations for API Integration

Ensuring Successful Collaboration



Trust and Relationship Building: Fostering trust through transparent communication, understanding mutual benefits, and building strong collaborative relationships.



Liability and Accountability: Defining liability and accountability for data breaches, inaccuracies, or other issues, with contingency plans for data-related incidents.



Data Access and Usage Policies: Establishing policies on data access, usage limitations, and guidelines for data access requests and approvals.



Exit Strategies: Developing procedures for ending data sharing agreements, with plans for data transition and migration.



Communication Protocols: Setting clear protocols for collaboration, including channels for regular communication and conflict resolution.

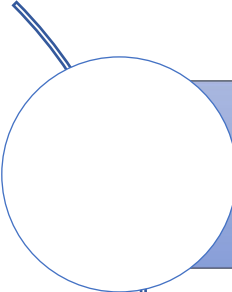


Scalability and Flexibility: Considering the arrangement's scalability for future growth and incorporating flexibility to adapt to business and technological changes.

What are some API opportunities?

Potential Use Cases

Creating Connections aligned to Standards



Data Collection and Reporting API

- APIs that allow parties to streamline data collection by integrating directly with financial institutions, Standard Setting Organizations, or other entities.



Standardized Reporting APIs

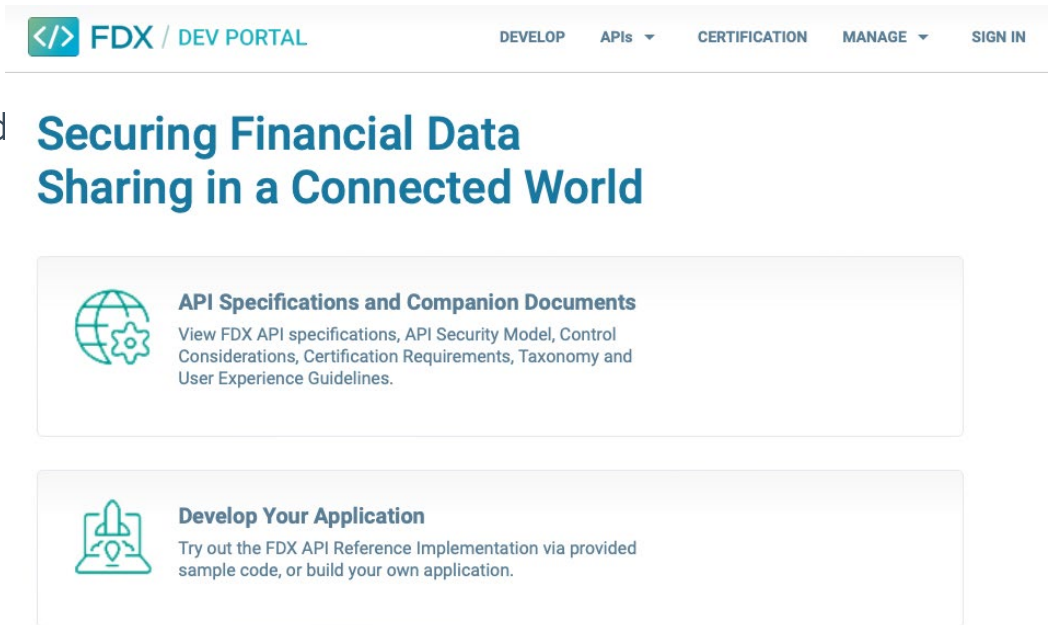
- Standardization of incoming and outgoing report data.
- Standardized APIs for reporting, ensuring consistency, and making it easier to analyze and compare data

References


Developer Portal

Overview: A member-only portal with tools and artifacts to support the ecosystem.


<https://developer.financialdataexchange.org/>




The screenshot shows the top navigation bar of the Developer Portal. It includes a code icon, the text 'FDX / DEV PORTAL', and several menu items: 'DEVELOP', 'APIs' (with a dropdown arrow), 'CERTIFICATION', 'MANAGE' (with a dropdown arrow), and 'SIGN IN'. Below the navigation bar is a large heading: 'Securing Financial Data Sharing in a Connected World'. Underneath this heading are two main content blocks. The first block is titled 'API Specifications and Companion Documents' and features a globe icon with a gear. The second block is titled 'Develop Your Application' and features an icon of a laptop with a lightbulb above it.

 **FDX / DEV PORTAL** DEVELOP APIs ▾ CERTIFICATION MANAGE ▾ SIGN IN

Securing Financial Data Sharing in a Connected World

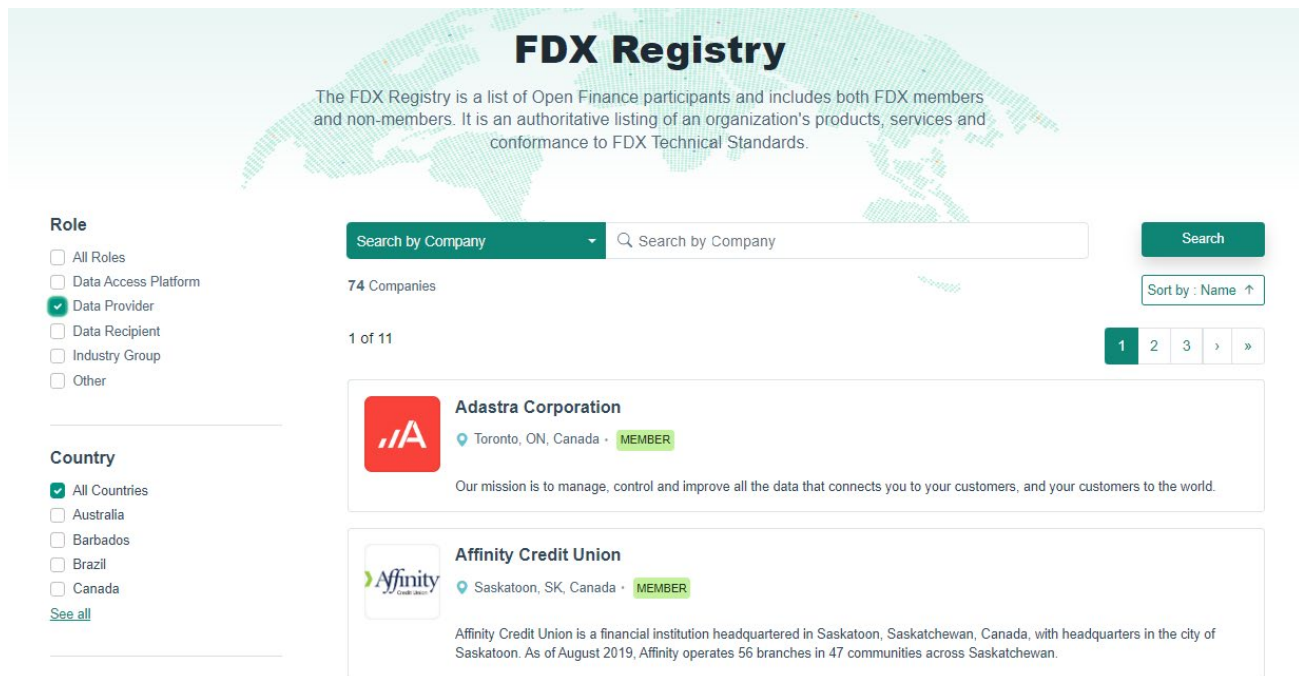
 **API Specifications and Companion Documents**
View FDX API specifications, API Security Model, Control Considerations, Certification Requirements, Taxonomy and User Experience Guidelines.

 **Develop Your Application**
Try out the FDX API Reference Implementation via provided sample code, or build your own application.

FDX Registry

Overview: A publicly viewable registry of all FDX members. Summary information is available to all viewers with organization details available to members.

<https://registry.financialdataexchange.org/>



FDX Registry

The FDX Registry is a list of Open Finance participants and includes both FDX members and non-members. It is an authoritative listing of an organization's products, services and conformance to FDX Technical Standards.

Role

- All Roles
- Data Access Platform
- Data Provider
- Data Recipient
- Industry Group
- Other

Country


- All Countries
- Australia
- Barbados
- Brazil
- Canada

[See all](#)

Search by Company


74 Companies

1 of 11 1 2 3 > »

 **Adastra Corporation**

Toronto, ON, Canada · MEMBER

Our mission is to manage, control and improve all the data that connects you to your customers, and your customers to the world.

 **Affinity Credit Union**

Saskatoon, SK, Canada · MEMBER

Affinity Credit Union is a financial institution headquartered in Saskatoon, Saskatchewan, Canada, with headquarters in the city of Saskatoon. As of August 2019, Affinity operates 56 branches in 47 communities across Saskatchewan.

References

Acknowledgments and Further Reading

- **Source Acknowledgment:** All information and examples provided in this presentation are based on best practices and standards within the API development community and financial services industry.
- **Further Reading:** For more in-depth understanding, refer to the official documentation of RESTful API standards, Financial Data Exchange (FDX) guidelines, and OpenAPI.
- **Additional Resources:**
 - API Design Patterns by Michael Amundsen
 - Web API Design: Crafting Interfaces that Developers Love by Brian Mulloy
 - Articles from reputable technology blogs and financial services forums.
 - Hands-On RESTful API Design Patterns and Best Practices By Harihara Subramanian and Pethuru Raj
- <https://www.FinancialDataExchange.org/>