DCXchange.net

Platform Technology White Paper

DCXchange.net — A TooziT LLC Platform — Version 1.0 — 2026

Confidential — For Informational Purposes Only — Patent Pending
Available upon request — licensing@dcxchange.net

This document is the DCXchange.net Platform Technology White Paper. It provides a complete technical description of the platform's architecture, component stack, data model, Patent Pending Schema Engine, security infrastructure, and replacement cost analysis. It is a companion document to the DCXchange.net Platform Pro Forma and is intended for technical reviewers, technology acquirers, and licensing candidates conducting detailed technical due diligence.

This document does not constitute legal advice, investment advice, or a guarantee of any technical specification. All platform capabilities described herein reflect the platform's design and implementation as of the document date.

SECTION 1 — PLATFORM OVERVIEW AND TECHNICAL PHILOSOPHY

DCXchange.net is a multi-tier, database-driven financial instrument marketplace platform built on the Microsoft .NET technology stack. Its architecture follows a deliberate philosophy: use standard, widely-supported, well-documented technology at every layer, and concentrate architectural innovation at the data and configuration level rather than at the framework or language level.

This philosophy produces a platform that is technically sophisticated at the design level — the Schema Engine, the permission architecture, the real-time event infrastructure, the audit system — and operationally straightforward at the maintenance level. A competent .NET development team can understand, maintain, extend, and operate the platform without specialized knowledge of proprietary frameworks or unusual architecture patterns.

The platform is delivered as a complete, production-ready system. It includes the web application, the mobile applications, the admin interface, the database schema, all stored procedures and functions, complete API documentation, administrator reference documentation, and deployment procedures. Everything required to stand up a fully operational instance of the platform is included in the delivery package.

> **TECHNOLOGY STACK SUMMARY**
> Database: Microsoft SQL Server. Web Framework: ASP.NET. Server-Side Languages: C# and Visual Basic. Client-Side: HTML, CSS, JavaScript. Real-Time: ASP.NET SignalR (WebSocket). Mobile iOS: Swift (Native). Mobile Android: Kotlin (Native). API: RESTful ASP.NET Web API. All components use widely adopted, commercially supported Microsoft-ecosystem technology.

SECTION 2 — COMPONENT ARCHITECTURE

The platform is composed of ten primary components. Each component has a defined responsibility boundary and communicates with other components through well-defined interfaces. This component separation enables independent maintenance and upgrade of each component without requiring changes to others.

Component	Architecture and Function
SQL Server Database Engine	The data persistence layer. Stores all platform data: participant accounts, instrument listings, field definitions, auction bid records, message threads, due diligence document metadata, access control records, audit logs, analytics aggregations, and Schema Engine configuration. ACID-compliant, row-level security, encrypted at rest using Transparent Data Encryption.
ASP.NET Web Application	The primary application server. Handles all HTTP and WebSocket traffic, routing, session management, authentication, authorization, and server-side rendering. Runs on Windows Server with IIS. Stateless application tier designed for horizontal scaling behind a load balancer.
RESTful API Layer (ASP.NET Web API)	A dedicated API surface that serves all data requests from web clients, mobile clients, and authorized third-party consumers. All permission evaluation occurs at the API layer before any database query executes. The API is the single authoritative data access point for every client across every access method.
Configurable Instrument Schema Engine	The Patent Pending core of the platform. A database-driven architecture that stores instrument type definitions as configuration records and renders all instrument-specific platform behavior — listing forms, search filters, API responses, analytics dimensions — dynamically from those records at runtime. Described in full detail in Section 3.
JavaScript Client Layer	Client-side logic executed in the participant's browser. Handles conditional field visibility in the listing form (driven by Schema Engine field definition rules), real-time bid chart updates via WebSocket connection, market activity ticker rendering, form validation before server submission, and analytics dashboard interactivity.
SignalR WebSocket Hub	Persistent bidirectional communication channel between the ASP.NET server and all connected clients. Broadcasts auction bid events, market activity events, and messaging notifications to all subscribed clients simultaneously in real time without polling or page refresh.
Native iOS Application (Swift)	Purpose-built native application for iPhone and iPad. Communicates with the RESTful API. Renders all UI using native UIKit components. Integrates with iOS Keychain for secure token storage, Face ID and Touch ID for biometric authentication, APNs for push notification delivery, and the native camera and file system for document upload.
Native Android Application (Kotlin)	Purpose-built native application for Android phones and tablets. Communicates with the RESTful API. Renders all UI using native Android components. Integrates with Android Keystore for secure token storage, Android BiometricPrompt for biometric authentication, Firebase Cloud Messaging for push notification delivery, and the native camera and file system for document upload.
Progressive Web App (PWA)	The DCXchange.net website itself, built to the W3C Progressive Web App specification. Service worker enables offline access to saved searches and recently viewed listings, and powers Add to Home Screen installation on iOS and Android without App Store involvement. The PWA provides full platform feature access from any modern browser without any download or installation requirement.
Admin Interface	A secure, permission-controlled administrative interface for platform operators. Manages participant accounts and verification status, instrument taxonomy and field definitions, subscription tier

	configuration, auction monitoring and intervention, data licensing client access, and platform-wide analytics reporting. All Schema Engine configuration is performed through the admin interface without developer involvement.

SECTION 3 — THE PATENT PENDING CONFIGURABLE INSTRUMENT SCHEMA ENGINE — TECHNICAL DESCRIPTION

The Configurable Instrument Schema Engine is the architectural innovation that separates DCXchange.net from every competitive platform in the private real estate finance market. Its technical description follows. This description is provided at the level of an architecture overview — not at the level of implementation source code, which is proprietary.

The Core Architectural Insight

Every financial instrument listing, regardless of type, shares a common set of platform attributes: a seller, a price, a status, a due diligence package, and a communication channel. These common attributes are handled by the platform's core listing infrastructure — a single master listing table with universal fields that apply to every instrument type on the platform.

What differs between instrument types is the set of descriptive fields a buyer requires to evaluate the instrument. A performing mortgage note requires UPB, interest rate, payment amount, maturity date, lien position, property type, and borrower performance history. A tax lien certificate requires certificate number, issuing jurisdiction, statutory interest rate, redemption period, and delinquency year. These are completely different field sets, and there are 150+ such field sets across 15 instrument categories.

The conventional engineering approach to this problem is to build a separate form, a separate database table structure, and a separate API endpoint for each instrument type. That approach works for three instrument types. It becomes unmanageable at fifteen. It is impossible at 150. It also means that adding a new instrument type requires a code deployment — developer time, testing, staging, release management — for every new category.

The Schema Engine solves this by treating the field set as data rather than code. The field definitions for each instrument type are stored as configuration records in the database. The form renderer reads those records and builds the appropriate form at runtime. The API layer reads those records and constructs instrument-specific response objects. The search infrastructure reads those records and indexes the appropriate fields. Adding a new instrument type is a database operation performed through the admin interface — no code deployment, no developer involvement, no platform disruption.

Database Structure

The Schema Engine rests on two primary database tables. The Instrument Type table stores one record per supported instrument type, containing the type identifier, category assignment, display name, description, and active status. The Field Definition table stores one record per field per instrument type, containing the field identifier, instrument type reference, field name, data type (text, numeric, date, currency, dropdown, boolean, multi-select), display label, help text, required or optional status, conditional visibility rules stored as a JSON configuration object, validation rules stored as a JSON configuration object, search index flag, display group, and display order.

Listing field values — the actual data entered by a seller for a specific listing — are stored in a normalized Listing Field Values table containing one record per completed field per listing, keyed to the field definition ID. This design means the master listing table has a fixed structure that never changes as new instrument types are added. The instrument-specific data lives in the normalized values table, not in instrument-specific columns of the master listing table.

Dynamic Form Rendering

When a seller selects an instrument type during listing creation, the platform queries the Field Definition table for that instrument type and returns the ordered set of field definitions. The server-side renderer constructs the HTML form from those definitions — rendering text inputs, numeric inputs, date pickers, currency fields, dropdowns, boolean toggles, and multi-select checkboxes as appropriate to each field's data type specification. The client-side JavaScript layer processes the conditional visibility rules from the field definition records and applies them as the seller completes the form — showing and hiding field groups based on prior field selections without additional server requests.

Search and Filter Integration

Field definition records marked with the search index flag are automatically included in the platform's search and filter infrastructure. When a new instrument type is activated in the admin interface, its indexed fields are immediately available as search filter options for buyers. The search infrastructure does not require code changes, index rebuilds, or configuration file updates when new instrument types are added.

API Response Generation

The API layer uses the field definition records to construct instrument-specific listing response objects for all clients. A request for a performing mortgage note listing returns only the fields defined for that instrument type. A request for a tax lien certificate listing returns only the fields defined for that type. Clients receive clean, instrument-specific response objects rather than generic objects with 170 nullable columns, most of which are irrelevant to the requested instrument.

SECTION 4 — DATA MODEL OVERVIEW

The platform's data model is organized around eleven primary table groups. The following table describes each group at the conceptual level. Detailed entity-relationship diagrams and complete table specifications are included in the platform delivery package.

Table Group	Description
Participants	Account records for all sellers, buyers, and administrators. Fields: account ID, account type (seller/buyer/admin), entity type, verification status, verification tier, subscription tier, subscription status, geographic market, contact information (encrypted), created date, last active date, session tokens.
Instrument Types	Schema Engine configuration table. One record per supported instrument type. Fields: instrument type ID, category ID, display name, description, active status, search index configuration, display order.
Field Definitions	Schema Engine field configuration table. Many records per instrument type. Fields: field ID, instrument type ID, field name, data type, display label, help text, required flag, conditional visibility rules (JSON), validation rules (JSON), search index flag, display order, display group.
Listings	Master listing table. One record per active or historical listing. Fields: listing ID, seller account ID, instrument type ID, listing status, sale method, asking price, fractional flag, fractional unit count, fractional unit price, accepted payment methods (JSON), created date, activated date, expiration date, closed date, closed price.
Listing Field Values	Instrument-specific field data for each listing. One record per completed field per listing. Fields: value ID, listing ID, field definition ID, field value (typed storage by data type). This table is the normalized storage layer for all instrument-specific listing data.

Due Diligence Documents	Document metadata for all seller-uploaded files. Fields: document ID, listing ID, uploader account ID, document type, file name, file size, storage reference (blob URL), upload date, access log.
Auction Bid History	Complete bid record for all auction events. One record per qualifying bid. Fields: bid ID, listing ID, bidder account ID, bid amount, bid timestamp (millisecond precision), bid status (accepted/rejected/superseded), reserve met flag at time of bid, auto-extend triggered flag.
Message Threads	Internal messaging thread records. One thread per buyer-listing contact initiation. Fields: thread ID, listing ID, buyer account ID, seller account ID, thread status, created date, last message date.
Messages	Individual message records within threads. Fields: message ID, thread ID, sender account ID, message body (encrypted at rest), sent timestamp, read timestamp, message type (text/offer/counter/document request).
Audit Log	Tamper-evident record of all significant platform events. Fields: log ID, event type, actor account ID, target object type, target object ID, event timestamp, IP address, event metadata (JSON). Append-only table — no update or delete operations permitted.
Subscription Records	Billing and subscription history. Fields: subscription ID, account ID, tier, billing cycle, amount, payment method token (not payment data), status, start date, renewal date, cancellation date.

SECTION 5 — SECURITY ARCHITECTURE

Transport Security

All data in transit between any client and any platform endpoint is encrypted using TLS 1.2 or higher. HTTP requests are permanently redirected to HTTPS at the infrastructure level. WebSocket connections are established over WSS (WebSocket Secure). There is no unencrypted data path in the platform under any operating condition.

Authentication

Participant credentials are validated against the database at the API layer using a standard username and password authentication flow. Passwords are never stored in readable form. The platform uses bcrypt or Argon2 — industry-standard cryptographic one-way hashing algorithms — to process passwords before storage. Authentication produces a signed JSON Web Token (JWT) with a configurable expiration window. The JWT is stored in secure, HttpOnly cookie storage for web clients, iOS Keychain for iOS clients, and Android Keystore for Android clients. Biometric authentication on native apps delegates all biometric processing to the device's native secure enclave — no biometric data is transmitted to or stored on any platform server.

Authorization

All authorization decisions are made at the API layer before any database query executes. The permission framework evaluates the requesting participant's account role and subscription tier against the resource and operation requested. Client-side UI hiding of tier-restricted features is a user experience optimization, not a security control. A participant cannot circumvent tier restrictions by manipulating the client-side interface because all enforcement occurs server-side at the API boundary.

Data Security

The SQL Server database is encrypted at rest using Transparent Data Encryption (TDE). Blob storage containing due diligence documents is encrypted at rest using storage-level encryption with platform-managed keys. Personally identifiable information fields — including contact information, financial

terms, and identity verification data — are encrypted at the column level using SQL Server Always Encrypted, ensuring that the data is decryptable only by authorized application processes with the correct encryption certificates.

Audit Infrastructure

The audit log table is an append-only record of all significant platform events. Audit records are written by the application layer and cannot be modified or deleted through any standard application operation. The audit table is segregated at the database permission level — the application service account has INSERT permission only on the audit table, with no UPDATE or DELETE permissions. This technical constraint, not policy alone, enforces the immutability of the audit record.

SECTION 6 — BUILD COMPLEXITY AND REPLACEMENT COST ANALYSIS

This section provides a component-by-component analysis of the development effort required to build a platform functionally equivalent to DCXchange.net from scratch. This analysis is provided for technology acquirers, licensing candidates, and institutional buyers evaluating the platform's acquisition valuation against the alternative of internal development.

The following estimates are based on standard enterprise software development rates in the United States ($100 to $200 per hour for senior developers, $75 to $150 per hour for mid-level developers) and documented timelines for comparable financial marketplace platform development projects.

Component	Dev Cost	Timeline	Notes
Database Architecture and Schema Design	$75,000–$150,000	2–3 months	Senior SQL Server architect. Normalized schema design for 150+ instrument types, performance indexing, security model, audit infrastructure. Most teams underestimate this phase significantly.
Schema Engine — Core Architecture	$100,000–$200,000	3–4 months	Senior architect plus two backend developers. Designing and implementing the configuration-driven field system, the dynamic form renderer, the conditional visibility engine, and the API response generator. This is the technically hardest component to get right.
API Layer	$80,000–$150,000	2–3 months	Two backend developers. RESTful API design, authentication, permission enforcement, rate limiting, versioning.
Listing Form and Admin Interface	$60,000–$120,000	2–3 months	One to two frontend developers. Complex conditional form with 100+ fields across 15 categories, real-time validation, due diligence upload, admin configuration interface.
Auction Engine	$50,000–$100,000	1–2 months	One backend developer. Timer logic, bid acceptance, reserve evaluation, auto-extend, immutability enforcement, real-time broadcast.
Internal Messaging System	$40,000–$80,000	1–2 months	One backend developer. Threaded messaging, document requests, offer workflow, audit trail, immutability architecture.

Real-Time Infrastructure (WebSocket)	$30,000–$60,000	1 month	One backend developer. SignalR hub, event subscription management, client synchronization.
Analytics Dashboard	$50,000–$100,000	1–2 months	One backend and one frontend developer. Nine visualization types, geographic heat maps, tier-gated access, daily aggregation queries.
Native iOS Application	$80,000–$150,000	2–3 months	One dedicated iOS developer. Purpose-built application, not a web wrapper. Full API integration, biometric auth, APNs, offline capability.
Native Android Application	$80,000–$150,000	2–3 months	One dedicated Android developer. Purpose-built application. Full API integration, biometric auth, FCM, offline capability.
Security Architecture	$40,000–$80,000	1 month	Security architect. Encryption implementation, penetration testing, vulnerability assessment, compliance documentation.
QA and Testing	$60,000–$120,000	2–3 months	Two QA engineers. End-to-end test suite, load testing, cross-browser and cross-device testing, security testing.
Project Management and Documentation	$50,000–$100,000	Throughout	Dedicated project manager. Architecture documentation, API documentation, admin reference documentation, deployment procedures.
TOTAL REPLACEMENT COST ESTIMATE	**$745,000–$1,510,000**	**18–24 months**	Plus patent risk: any schema-driven instrument marketplace architecture risks infringement of the DCXchange.net patent portfolio.

THE PATENT RISK FACTOR

The replacement cost analysis above addresses only the financial and time cost of replication. It does not address the legal cost. The DCXchange.net patent portfolio — 14 provisional applications covering the Schema Engine, the permission architecture, the messaging system, the auction engine, and eight additional component technologies — means that any company building a comparable platform using similar architectural approaches faces patent infringement risk. A well-advised technology company building a competing platform would need to design around the patent claims, which requires a different and less efficient architecture, increasing both the development cost and the ongoing operational cost. The acquisition of DCXchange.net eliminates that risk entirely and acquires the patent protection as a competitive asset.

SECTION 7 — PATENT APPLICATION PORTFOLIO

The following table summarizes the fourteen provisional patent applications filed or pending filing with the United States Patent and Trademark Office. Patent No. 1 has been filed and assigned a USPTO application number. Patents No. 2 through 14 are pending filing within the twelve-month priority window

established by Patent No. 1. Full application abstracts are contained in Appendix H of the Pro Forma document set.

No.	Patent Title	Subject Matter
Patent No. 1 — FILED	Complete DCXchange.net Exchange Architecture	The complete integrated platform architecture as a unified system. Umbrella application establishing priority date for all component technologies.
Patent No. 2	Configurable Instrument Schema Engine	The database-driven, configuration-record-based system for dynamic instrument field definition across a multi-vertical financial instrument marketplace without code deployment.
Patent No. 3	Multi-Platform Synchronization Framework	Single API layer architecture maintaining complete real-time data consistency across web, iOS, and Android access points simultaneously.
Patent No. 4	Tiered Access Control Framework	API-layer-enforced, role-and-tier-based access control preventing client-side permission circumvention in a multi-tier financial instrument marketplace.
Patent No. 5	Buyer-Initiated Contact Protocol with Bilateral Identity Protection	Architecturally enforced buyer-only contact initiation with structural prevention of seller-initiated unsolicited contact and bilateral identity protection until voluntary disclosure.
Patent No. 6	Immutable Audit-Linked Internal Messaging System	Tamper-evident, listing-associated, permanently archived messaging system with structured offer workflow and immutable communication record.
Patent No. 7	Instrument Performance Status Lifecycle Management System	Automated status progression management for financial instrument listings through defined lifecycle states with event-driven transition logic.
Patent No. 8	ZIP-Code-Based Contextual Service Provider Discovery Engine	Dynamic geographic matching of listing collateral location to relevant service provider categories with contextual placement at point-of-need in the participant workflow.
Patent No. 9	Portfolio and Tape Aggregation Display System	Multi-instrument portfolio packaging and tape aggregation display architecture for bulk instrument listings with aggregate metric computation and display.
Patent No. 10	Dual-Role Participant Classification System	Architecture supporting simultaneous seller and buyer classification for the same participant account with independent tier management for each role.
Patent No. 11	Anonymous Buyer Preference Filtering System	Saved search and alert architecture enabling buyers to define and store instrument preference profiles without disclosing those preferences to sellers or the platform.
Patent No. 12	Multi-Vertical Instrument Taxonomy with Dynamic Field Schema Inheritance	Hierarchical instrument taxonomy with category-level field inheritance enabling new instrument types to inherit base field sets from their category without redundant configuration.

Patent No. 13	Saved Search Alert Priority Tiering System	Subscription-tier-based alert delivery priority system ensuring higher-tier participants receive matching listing alerts before lower-tier participants.
Patent No. 14	Open Ascending-Bid Auction Engine with Access Fee Gating, Date Lock Enforcement, and Real-Time Bid Transparency	Complete auction engine architecture including access fee validation, parameter immutability after first bid, auto-extend logic, and real-time bid broadcast to all subscribed participants.

SECTION 8 — PLATFORM DELIVERY PACKAGE

The DCXchange.net platform delivery package includes every asset required to stand up a fully operational instance of the platform. The contents of the delivery package vary by commercial arrangement as follows.

Asset	License to Operate / Platform Business Sale / IP Acquisition
Running platform instance	Included in all arrangements. Delivered as a configured, operational platform instance on the purchaser's designated hosting environment.
Complete source code	Included in Technology/IP Acquisition only. Full source code for all platform components — web application, API layer, Schema Engine, auction engine, messaging system, analytics engine, admin interface, and mobile applications.
Database schema and seed data	Included in all arrangements. Complete SQL Server database schema, all stored procedures and functions, all indexes, the complete instrument taxonomy (15 categories, 150+ instrument types), and all field definitions.
API documentation	Included in all arrangements. Complete RESTful API specification covering all endpoints, request formats, response formats, authentication requirements, and error codes.
Administrator reference documentation	Included in all arrangements. Complete guide to all admin interface functions including participant management, instrument taxonomy management, auction monitoring, data licensing client management, and platform configuration.
Deployment procedures	Included in all arrangements. Step-by-step procedures for deploying the platform on a new hosting environment, including database setup, application server configuration, and third-party integration setup.
Mobile application builds	Included in all arrangements. Production-ready iOS and Android application builds configured for the purchaser's domain and branding. App Store submission assets included.
14 provisional patent applications	Included in Technology/IP Acquisition only. All patent application files transfer to the acquiring party. TooziT retains no ongoing interest in the patent portfolio following an outright acquisition.

Built on Standard Technology. Engineered for Permanence. Designed to Be Owned.

DCXchange.net — A TooziT LLC Platform

Technical Inquiries: licensing@dcxchange.net — Telephone: 866-4TOOZIT (866-486-6948)