exegy

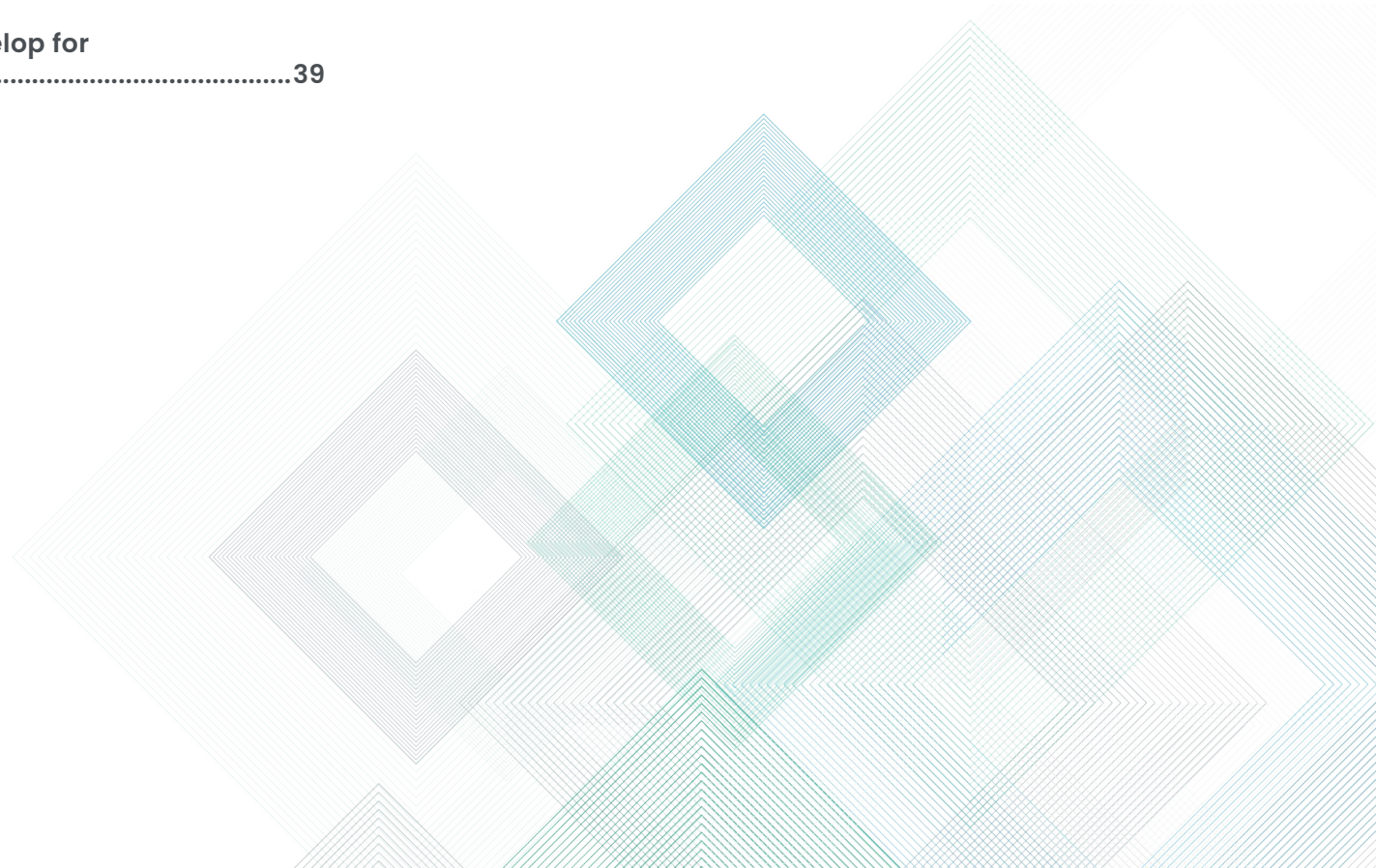
The True Cost of Real-Time Market Data Infrastructure: Quantifying the Build vs. Buy Debate for Capital Markets Firms

PART I: A Cost and Resource Analysis on Building and Maintaining Market Data Processing Technology in Software

SEPTEMBER 2025

TABLE OF CONTENTS

TABLE OF CONTENTS



Executive Summary

In today's capital markets, real-time market data processing is mission-critical for trading, risk management, and compliance. Firms face a fundamental strategic decision: **Build in-house or buy from specialized vendors.** This paper quantifies the true cost of developing and maintaining market data feed handlers using software-based solutions.

Our findings, based on real-world data from a Tier 1 global bank and Exegy's more than 20 years of market data expertise, reveal that building in-house software-based market data infrastructure is significantly more expensive, time-consuming, and operationally burdensome than vendor solutions.

Key Findings

In-House Software Development Is Expensive

- The first in-house software feed handler costs **$1.8 million**, while full North American equities and commodities coverage (20 markets) exceeds **$4.7 million**.
- In-house builds are **~8x** more expensive than Exegy's for full North American equities coverage.

Software Maintenance Is a Recurring Cost Drain

- Annual in-house maintenance costs exceed **$3.5 million**, while Exegy's fully managed solution **reduces these recurring expenses by ~2.6x**.
- In 2023 alone, Exegy managed **8,356 exchange notifications and 481 exchange-directed changes (EDCs), with 35% requiring code updates** that demand extensive development and QA efforts.
- Managing EDCs in-house adds **hidden operational risks** due to regulatory and exchange-driven protocol updates.

Vendor Solutions Offer Time Savings & Resource Optimization

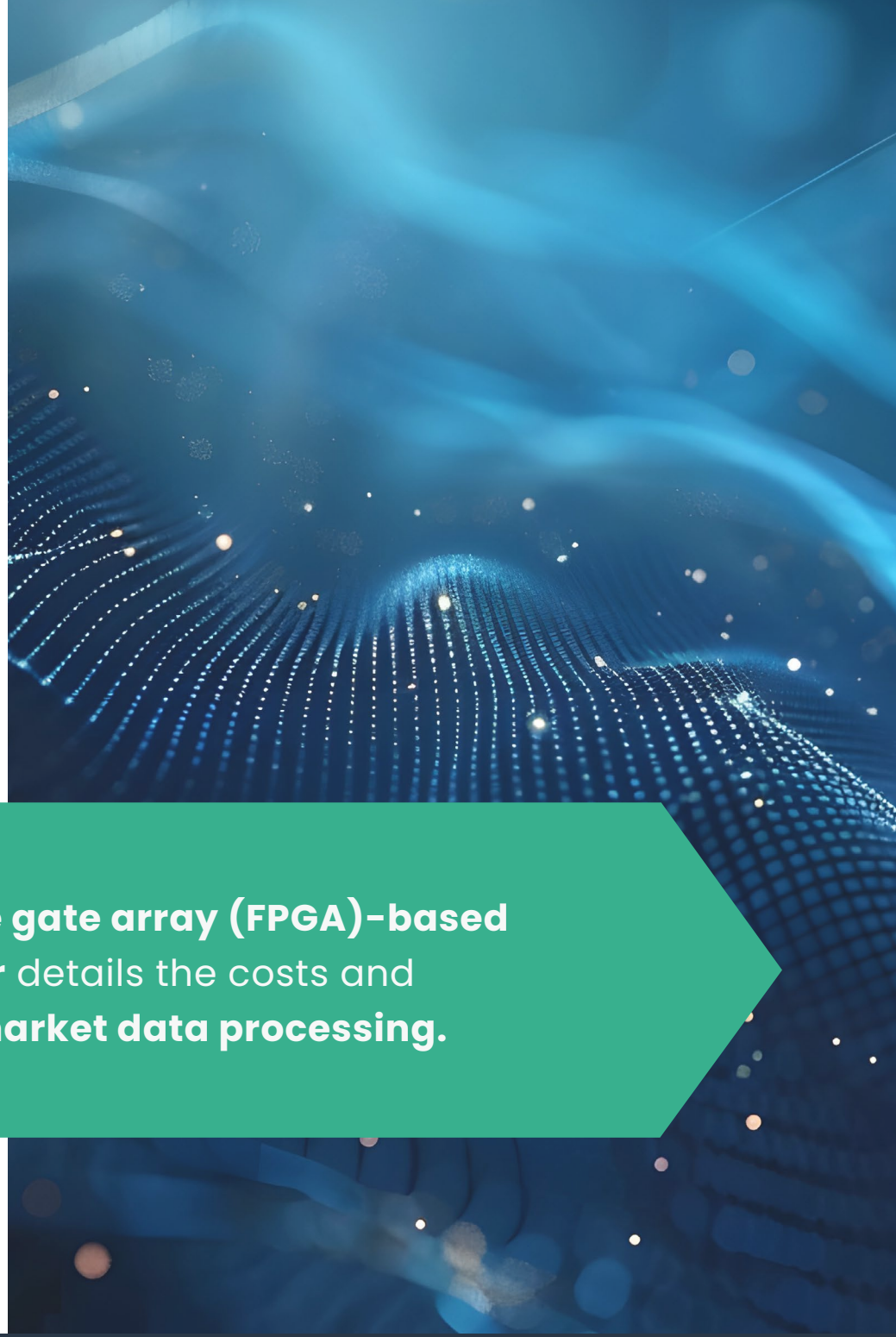
- Exegy delivers production-ready solutions in **4-6 months**, compared to **3.5 years** for in-house teams—a **7x** speed advantage.
- In-house development diverts skilled engineers and resources from strategic projects, increasing overhead and opportunity costs.
- Partnering with Exegy reduces internal workloads, allowing firms to focus on core business innovations.

Strategic Takeaways

Building market data feed handlers **in-house may seem like a way to retain control, but the true cost in time, resources, and ongoing maintenance is substantial.**

Vendor solutions provide:

- **Cost Savings:** Lower total cost of ownership (TCO) by reducing development and maintenance expenses
- **Faster Time to Market:** Immediate access to fully managed market data solutions
- **Reduced Operational Overhead:** Vendors absorbing the burden of ongoing exchange-driven updates and system monitoring



For firms evaluating **field-programmable gate array (FPGA)-based alternatives**, our **companion white paper** details the costs and complexities of **hardware-accelerated market data processing.**

1. Introduction

The ability to process real-time market data efficiently and reliably is a mission-critical capability for capital markets firms, including investment banks, hedge funds, and proprietary trading firms. As trading volumes grow and latency requirements become increasingly stringent, firms must decide whether to build their own market data processing infrastructure or buy from specialized vendors. This decision carries significant implications for cost, operational efficiency, and business agility.

Historically, many firms have built custom software-based market data processing systems to meet their trading and risk management needs. However, the true cost of building and maintaining these systems in-house is often underestimated. Engineering and operational expenses, ongoing compliance with Exchange-Directed Changes (EDCs), and the cost of scaling to new markets all contribute to significant long-term financial and resource burdens.

This paper provides a quantitative cost analysis of in-house software-based market data infrastructure, based on real-world benchmarks from a Tier 1 global bank and Exegy's 20+ years of experience in market data technology. By breaking down the development, scaling, and maintenance costs of in-house software feed handlers, this paper offers critical insights into the long-term viability of in-house solutions versus vendor-managed alternatives.

Our Analysis Draws On

- **Tier 1 Global Bank Insights:** Real-world data from a Tier 1 global bank's experience with building in-house software feed handlers
- **Exegy's Expertise:** More than 20 years of specialized knowledge in developing market data processing solutions using both software and FPGA technology
- **Comparative Cost Data:** In-depth analysis of salary, overhead, and productivity metrics, providing a comprehensive view of total cost of ownership
- **Operational Insights:** Advanced data on managing Exchange-Directed Changes (EDCs), reflecting real-world resource demands and performance outcomes

For firms evaluating hardware-accelerated FPGA-based solutions, a companion white paper details the cost and complexity of building FPGA-based market data infrastructure, offering a side-by-side comparison of software and hardware approaches.

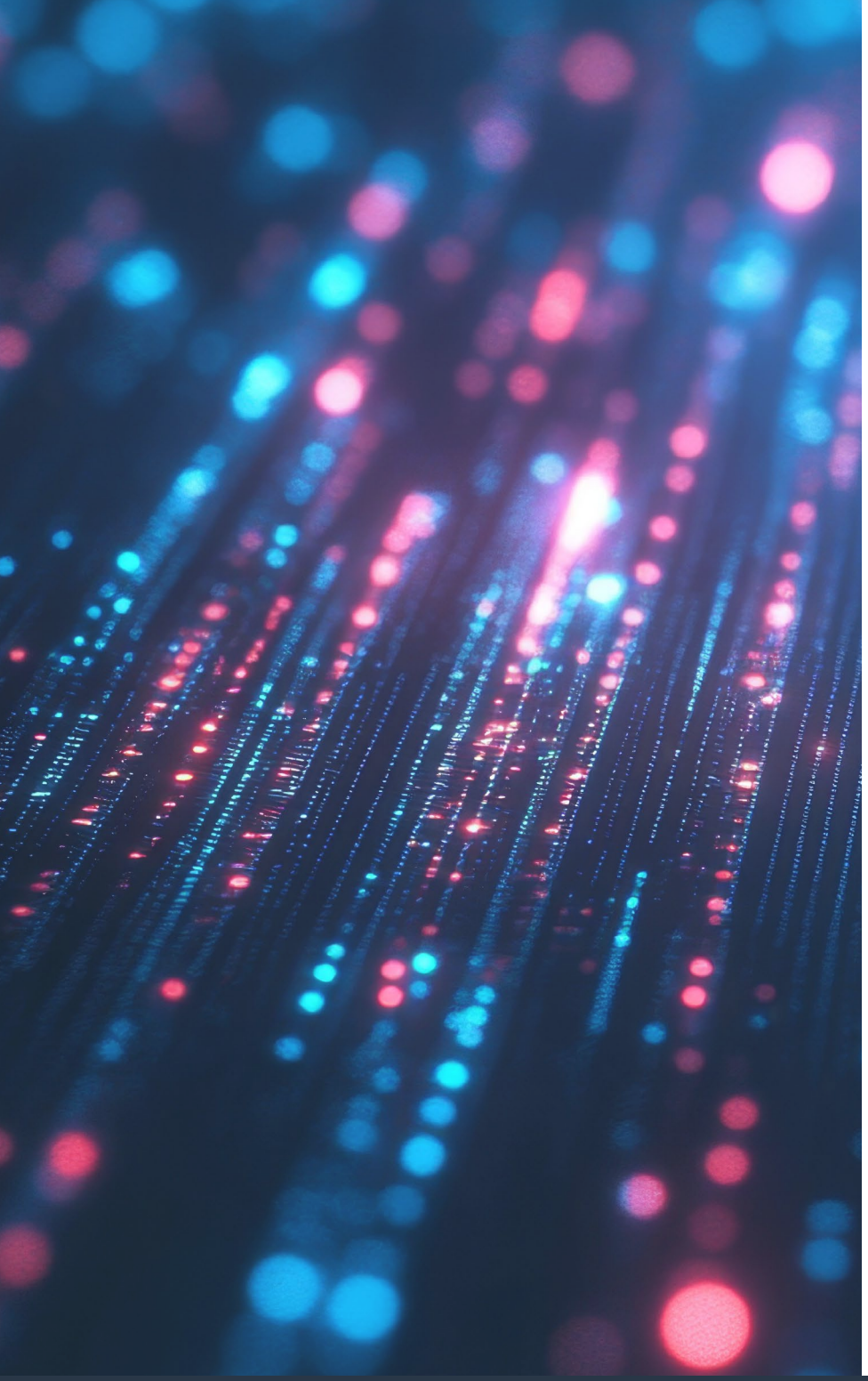
1.1 Industry Challenges Driving Market Data Infrastructure Decisions

Before diving into the challenges of building in-house, it is important to understand the broader industry dynamics compelling firms to reassess their market data infrastructure. Several key factors are placing unprecedented demands on existing systems:

- **Rising Market Data Volumes:** The rapid growth in trading activity and data generation is outpacing infrastructure upgrades, increasing the risk of outages and system failures.
- **Constrained Budgets & Data Center Space:** Firms face budgetary pressures and limited physical space for data center expansion, making it difficult to scale traditional infrastructures cost-effectively.
- **Heightened Competition & Tightening Spreads:** Increased market competition is driving tighter spreads, reducing fill rates, and forcing firms to optimize latency and data processing capabilities to maintain an edge.

Quantitative Industry Trends

- The volume of derivative contracts has increased **5x since 2020.**
- **Options contracts doubled** compared to the previous year.
- Derivatives contracts surpassed **150 billion** in 2024.
- The average daily trade volume of U.S. equities has risen **57% since 2019**, from **6 billion to 11 billion shares.**
- The upcoming **tick size reduction** is expected to significantly increase data volumes.



2. Addressing the Challenges of Building In-House

As firms look for strategic solutions to the above industry macro trends, those opting to develop in-house will face a range of operational, financial, and strategic hurdles.

These challenges are amplified by the growing complexity of market dynamics, the rapid pace of technological change, and the need to remain competitive in a data-driven landscape. Understanding these obstacles is critical for firms to evaluate whether to build, buy, or use a hybrid approach for their market data infrastructure.

Operational Complexity

Managing Exchange-Directed Changes (EDCs) is an ongoing, resource-intensive process. In 2023 alone, Exegy processed **8,356 inbound exchange notifications**, with **481 EDCs** requiring analysis. Approximately **35% of these changes** demanded code updates, testing, and deployment—consuming an average of **140 hours per EDC**. This workload strains engineering teams, often diverting focus from strategic projects to routine maintenance.

BUSINESS IMPACT:

Frequent EDCs introduce operational risk, increasing the potential for system outages or latency issues that can disrupt trading operations.

Inefficient Resource Allocation

In-house development consumes significant engineering time and expertise. Maintaining real-time market data infrastructure requires highly specialized talent, including low-latency software developers, FPGA engineers and architects, and QA specialists specialists. This resource allocation pulls valuable personnel away from revenue-generating activities, such as algorithm development or trading strategy optimization.

BUSINESS IMPACT:

Firms face higher personnel costs, increased turnover risk due to burnout, and reduced capacity for innovation in core business areas.

Increased Risk Exposure

Building and maintaining proprietary infrastructure exposes firms to operational and systemic risks. In-house systems are more vulnerable to outages if not continuously optimized, especially as market data volumes grow. Budget constraints and limited data center space further exacerbate these risks, making it difficult to scale infrastructure effectively.

BUSINESS IMPACT:

Downtime during peak trading periods can result in significant financial losses, regulatory scrutiny, and reputational damage.

Scalability Challenges

As firms expand their market coverage, scaling in-house infrastructure becomes increasingly complex. Each new data feed or market requires added development, testing, and maintenance efforts. Unlike vendor solutions that offer plug-and-play scalability, in-house systems face bottlenecks that delay time to market and limit growth potential.

BUSINESS IMPACT:

Delayed market access reduces competitive advantage, especially in high-frequency trading environments where any delays in getting your strategy into production can have a significant impact on the business.

3. The Cost to Build Market Data Processing Infrastructure in Software

In this section, we break down the comprehensive costs associated with building real-time market data processing infrastructure. The goal is to provide capital markets firms with a clear view of the resources, time, and expenses required for in-house development—initially in software and then using FPGA technology, **which can be found in our follow-up paper on FPGA-based market data processing.**

By quantifying these costs, we aim to help decision makers understand the financial implications of building versus buying and how these decisions impact time-to-market, scalability, and operational efficiency.

Building market data processing infrastructure from the ground up is a complex and resource-intensive endeavor.

3.1 Building the First Market Data Feed Handler: The Foundation

Building market data processing infrastructure from the ground up is a complex and resource-intensive endeavor. Before addressing the costs, it's critical to understand the engineering effort required to build the first market data feed handler.

The initial build sets the stage for subsequent feed handlers and represents the largest investment due to the need to establish core architecture and foundational components.

3.1.1 Why Focus on the Initial Feed Handler?

The initial feed handler represents the most substantial engineering investment because it requires:

- **Establishing the Core Architecture:**
  This includes defining data models, APIs, and system integration points.
- **Building Foundational Components:**
  Key modules, such as line handlers, normalizers, book builders, and distribution frameworks must be developed from scratch.
- **Addressing Latency and Throughput Requirements:**
  Optimizing for low-latency, high-throughput processing is more resource-intensive in the initial build.

Subsequent market feed handlers can leverage the frameworks and templates established during the initial build, reducing both development time and costs.

3.1.2 Key Components Required to Develop for an Initial Feed Handler

To contextualize the cost of building market data infrastructure, it's essential to understand the foundational components involved in developing the first feed handler.

This includes core systems, such as exchange line handlers, normalization, book building, and distribution layers—each with unique engineering requirements and varying degrees of reusability across markets.

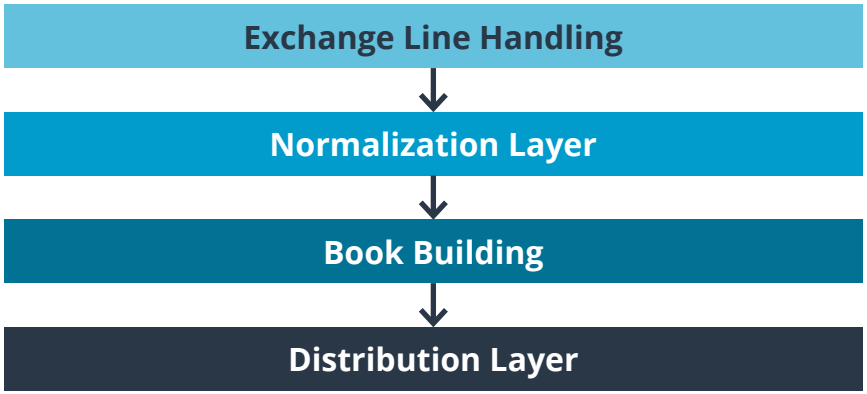
A detailed breakdown of these components and their implementation considerations can be found in ***Appendix A: Key Components of Initial Feed Handler Development.***

3.1.3 Breakdown of Estimated Engineering Effort

In the following sections, we will quantify the costs associated with these engineering efforts, drawing on data from a Tier 1 global bank and Exegy's internal benchmarks. We'll begin with the cost of building software feed handlers, followed by FPGA-based systems, and conclude with a comparative analysis of in-house builds versus Exegy's vendor solutions.

Understanding the true cost of building a market data feed handler requires a granular view of the engineering effort involved. This section breaks down the core components, detailing the specialized expertise, time investment, and hidden complexities that drive development costs.

Visual Timeline of Development Phases



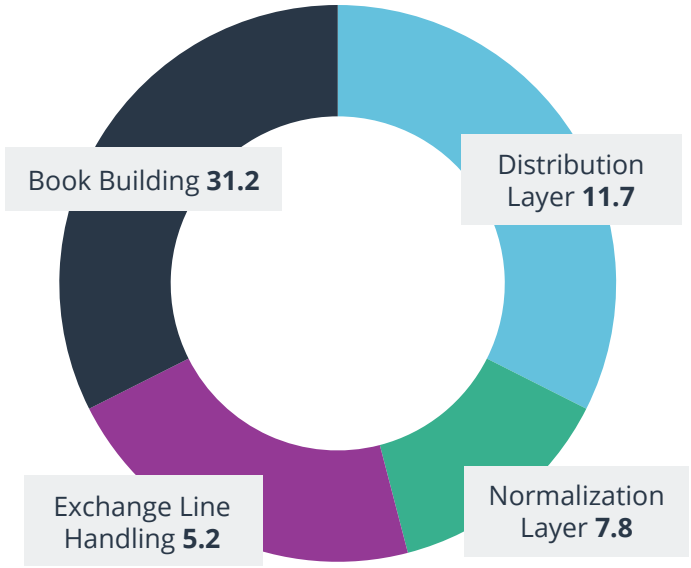
Context for Development Effort Estimates

The development effort accounts for not only the core engineering work required to build each component but also factors in the time spent on testing, quality assurance (QA), project management, and documentation. These elements are critical to ensuring the reliability, performance, and maintainability of the feed handler:

- **Testing & Quality Assurance:** Each development phase includes time allocated for comprehensive unit testing, integration testing, and end-to-end validation. This ensures that the feed handler performs reliably under real-world conditions.
- **Project Management:** Time estimates incorporate project oversight activities, such as sprint planning, status reporting, and cross-team coordination, which are essential for managing complex development workflows.
- **Documentation:** Proper technical documentation, including architecture diagrams, code comments, runbooks, and user guides, is embedded into the development timeline to support long-term maintainability and knowledge transfer.
- **Productivity Buffer:** A 30% buffer is applied to account for real-world challenges, such as unexpected integration issues, cross-functional coordination delays, and extended quality assurance cycles common in complex capital markets infrastructure projects. This buffer significantly impacts overall timelines and resource allocation.

Development Effort per Component:

Component	Number of Developers	Development Time (Months)	Core Development Effort (Person-Months)	Productivity Buffer (30%) (Person-Months)	Total Effort (Person-Months)
Exchange Line Handling	2	2	4	1.2	5.2
Normalization Layer	1	6	6	1.8	7.8
Book Building	4	6	24	7.2	31.2
Distribution Layer	3	3	9	2.7	11.7
Total	**10**	**17**	**43**	**12.9**	**55.9**



1. **Exchange Line Handling**
   - **Engineering Time:** 2 full-time employees (FTEs) × 2 months = 4 person-months (~640 hours)
   - **30% Buffer:** Adds 1.2 person-months (~192 hours) to account for cross-team dependencies and QA cycles
   - **Total Effort:** 5.2 person-months (~832 hours)

2. **Normalization Layer (Greenfield Development)**
   - **Engineering Time:** 1 FTE × 6 months = 6 person-months (~960 hours)
   - **30% Buffer:** Adds 1.8 person-months (~288 hours)
   - **Total Effort:** 7.8 person-months (~1,248 hours)

3. **Book Building (Order Book Management)**
    - **Engineering Time:** 4 FTEs × 6 months = 24 person-months (~3,840 hours)
    - **30% Buffer:** Adds 7.2 person-months (~1,152 hours)
    - **Total Effort:** 31.2 person-months (~4,992 hours)

4. **Distribution Layer**
    - **Engineering Time:** 3 FTEs × 3 months = 9 person-months (~1,440 hours)
    - **30% Buffer:** Adds 2.7 person-months (~432 hours)
    - **Total Effort:** 11.7 person-months (~1,872 hours)
    - **Complexity Factors:** High-throughput data distribution, backpressure management, and API performance tuning

Total software development effort: 55.9 person-months (~8,640 hours), approximately **5.09 years** of combined engineering effort

Key Metrics & Takeaways

Total Development Effort: 55.9 person-months (~8,640 hours), highlighting the extensive engineering commitment required

Primary Cost Driver: Book building, accounting for 56% of total effort due to its complex, real-time data processing requirements

Impact of Real-World Productivity Delays: When starting a new project, firms can encounter significant delays when recruiting skilled engineers in addition to the common development inefficiencies that impact time to market, such as cross-team dependencies, QA cycles, and unexpected technical issues.

Reusability Advantage: Core components, such as networking libraries and normalization frameworks, reduce costs for future market expansions.

3.2 Calculating the Cost of Building the First Software Feed Handler

Understanding the development effort allows us to estimate the total cost of building the first software feed handler. This calculation incorporates salary data, bonuses, and overhead costs to reflect the true financial commitment.

3.2.1 Salary and Overhead Calculations

Base salaries are based on New York City market rates. In addition to base salaries, our calculations include a bonus equal to **15% of the base salary** and staff overhead costs equal to **30% of the base salary**.

"Each full-time staff member picks up significant overhead from many nonrevenue-producing support groups (legal, compliance, HR, etc.). This is estimated to average around 30% of base compensation."
— Former executive at a Tier 1 investment bank

3.2.2 Total Cost Calculation

- **Developer base salary:** $250,000 per year
- **Bonus:** 15% of base salary
- **Overhead:** 30% of base salary

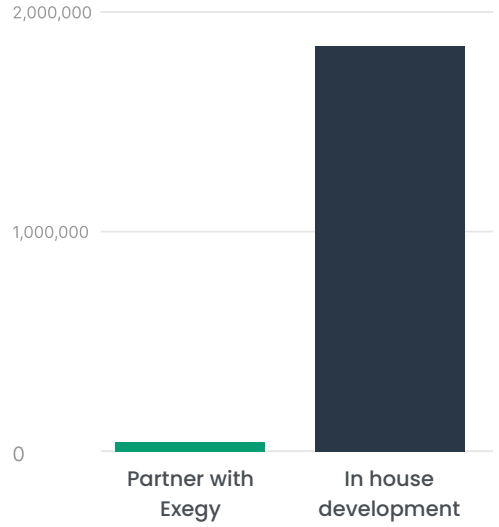
Cost Component	Amount (USD)
Base Salary (5.09 years)	$1,272,500
Bonus (15%)	$190,875
Overhead (30%)	$381,750
Total Cost	**$1,845,125**

- **Base Cost Calculation:** 5.09 years × $250,000 = $1,272,500
- **Total Cost:** $1,845,125 to build the initial feed handler

3.2.3 In-House vs. Vendor Cost Comparison

- **Cost to Partner with Exegy to Deploy 1 Feed Handler:** $30,000 per market
- **Comparison:** Building in-house is approximately **61.5x more expensive** than partnering with Exegy.

Comparative Cost Chart



KEY INSIGHT

Building in-house is **61.5x more expensive** than partnering with Exegy for a single feed handler. This reflects both direct costs (salaries and overhead) and hidden operational inefficiencies (delayed time to market and increased maintenance burden).

Real-World Implications

- **Opportunity Cost:** Delayed deployments result in lost revenue opportunities.
- **Resource Allocation Inefficiencies:** Engineering time diverted from revenue-generating projects
- **Operational Risk:** Higher costs associated with maintaining proprietary systems, especially as EDCs increase in frequency and complexity

This comparison highlights the significant cost efficiencies achievable through vendor partnerships, emphasizing the financial burden of in-house development not just in engineering effort but also in direct and indirect costs.

3.3 Building Additional Software Feed Handlers: Scaling with Efficiency

Once the initial feed handler is developed, building subsequent feed handlers requires a smaller time investment because core data models and foundational functionalities are already in place. However, unique exchange requirements and the need for efficient scaling still demand significant engineering resources.

3.3.1 Development Process Overview

The development process for additional feed handlers follows a structured workflow, ensuring efficiency while maintaining high performance standards:

1. Business Requirements & Analysis

- A **business analyst (BA)** reviews exchange documentation and writes development requirements for engineers.
- **Estimated Time:** 2 weeks → **with 30% buffer:** 2.6 weeks of effort

2. Software Development

- A **software engineer (SWE)** implements the feed handler based on business requirements, coding the necessary protocols and ensuring system compatibility.
- **Estimated Time:** 8 weeks → **with 30% buffer:** 10.4 weeks of effort

3. Quality Assurance (QA) Testing

- An **A software engineer (SWE)** specializing in QA testing rigorously tests the feed handler to ensure performance, reliability, and compliance with exchange standards.
- **Estimated Time:** 6 weeks → **with 30% buffer**: 7.8 weeks of effort
- Development Effort Breakdown

Development Effort Breakdown

Task	Role	Number of People	Time (Person-Weeks)	30% Buffer (Person-Weeks)	Total Time (Person-Weeks)	Total Time (Person-Years)
Business Requirements & Analysis	Business Analyst	1	2	0.6	2.6	0.06
Software Development	Software Engineer	1	8	2.4	10.4	0.22
Quality Assurance	Software Engineer	1	6	1.8	7.8	0.17
Total	**-**	**3**	**16**	**4.8**	**20.8**	**0.45**

3.3.2 Cost Calculation

Base Salaries

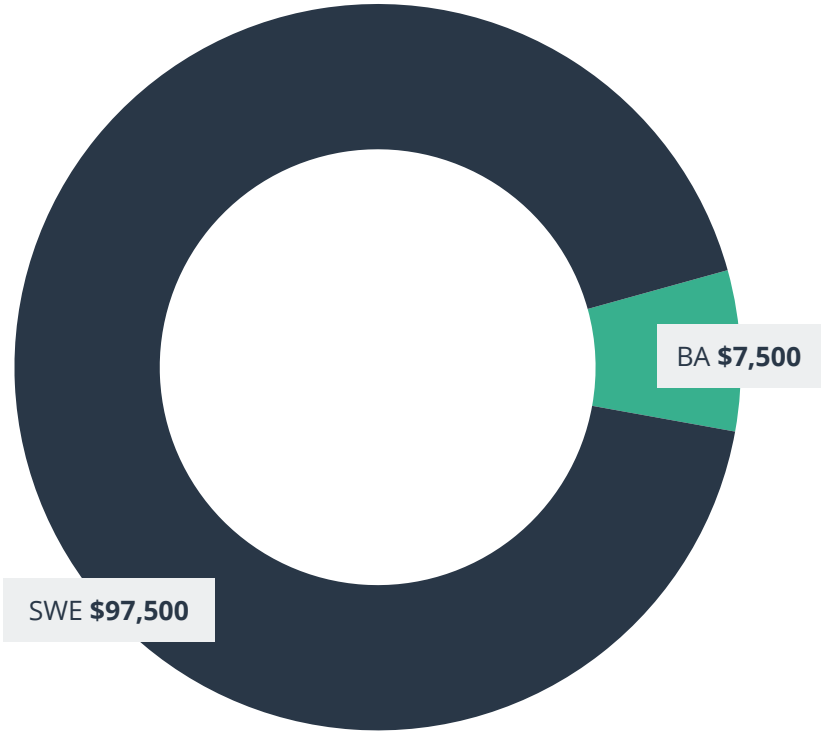
- **Business Analyst:** $125,000 per year
- **Software Engineer:** $250,000 per year

Total Net Cost

- **Business Analyst:** $125,000 × 0.06 years = **$7,500**
- **Software Engineers:** $250,000 × (0.22 years + 0.17 years) = **$97,500**
- **Total:** $105,000

Final Cost (Including Bonus & Overhead)

- **Bonus (15%)** = $15,750
- **Overhead (30%)** = $31,500
- **Total Cost per Additional Feed Handler: $152,250**



Calculation Note: The total net cost for software engineers is derived by combining the effort for both software development (0.22 years) and QA testing (0.17 years), resulting in a total engineering effort of 0.39 years.

KEY INSIGHTS

- **Total Cost Per Additional Feed Handler:** The calculated cost to build an additional feed handler in-house is $152,250, reinforcing the significant cost burden of scaling without vendor support.
- **Engineering Effort Dominates:** More than 90% of the cost comes from development and QA efforts, reflecting the technical complexity of market data feed handlers.
- **Scalability Benefits:** While each feed handler has unique requirements, reusing core components reduces time to market compared to the first build.
- **In-House vs. Vendor Costs:** Even with efficiencies, in-house builds remain significantly more expensive than partnering with vendors such as Exegy.

This section demonstrates how development efficiencies improve with scale, but the cost remains substantial when compared to vendor solutions.

3.4 Cost Calculation for Building Full North American Equities & Commodities Coverage

3.4.3 Visual Comparisons & Key Insights

Expanding market data feed handler coverage to all North American equities and commodities markets significantly increases costs and development efforts. Given that the first feed handler has been built, an additional **19 feed handlers** are required to complete coverage for the **20 market protocols** in this region.

Development Effort Calculation

From previous calculations, we determined that building an additional feed handler requires **0.45 years** (or **5.4 months**) of development effort.

- **Development time per feed handler:** 0.45 years (5.4 months)
- **Total development time for 19 additional feed handlers:**
  - 19 × 0.45 years = 8.55 years of total engineering effort
- **Total development time for full market coverage (20 feed handlers):**
  - 8.55 years (for 19 additional feed handlers) + 5.09 years (for the initial feed handler) = **13.64 years** of total engineering effort

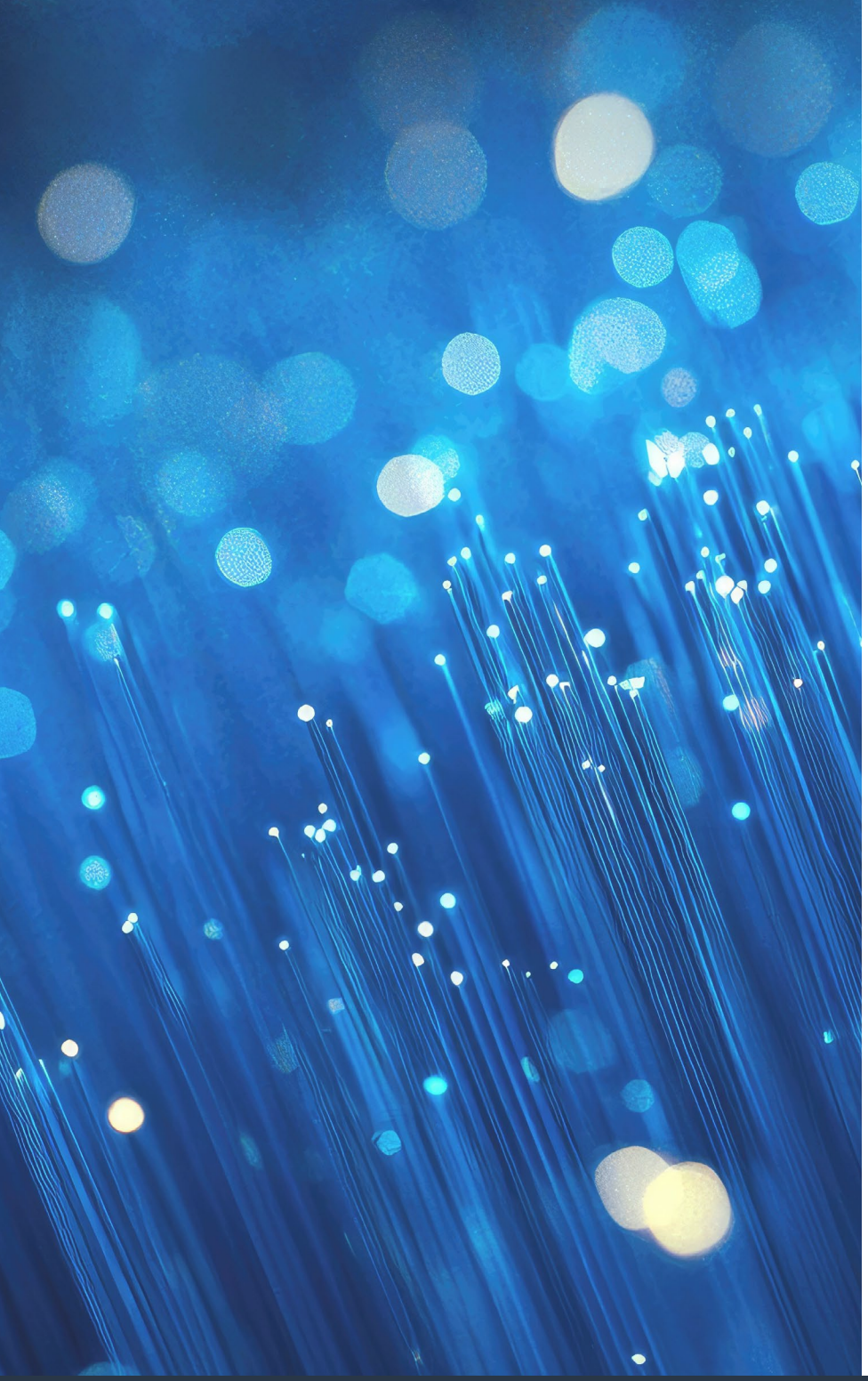
Total Cost Calculation

Using the previously established cost of **$152,250 per additional feed handler**, we calculate the total cost of covering **all 20 markets**:

- **Cost Per Additional Feed Handler:** $152,250
- **Total Cost for 19 Additional Feed Handlers:**
  - 19 × $152,250 = **$2,892,750**
- **Total Cost for Full Market Coverage (Including First Feed Handler):**
  - $1,845,125 (Initial Feed Handler) + $2,892,750 (Additional 19) = **$4,737,875**

KEY INSIGHTS

- **Significant Scaling Costs:** Expanding to full North American equities and commodities coverage requires nearly **$4.74M** in total development costs.
- **Time-to-Market Challenge:** Developing an additional 19 feed handlers would take over 9 years of total engineering effort—reflecting the cumulative workload required rather than calendar time, as multiple development streams would run in parallel.
- **Cost Comparison With Vendor Solutions:** Exegy provides fully operational feed handlers at a fraction of the cost, significantly reducing both up-front and long-term financial commitments.
- **Strategic consideration:** The substantial financial and engineering resources required highlight the challenge of scaling in-house solutions efficiently. Firms must consider how diverting this level of effort from other strategic initiatives impacts overall business agility and innovation.



4. The Cost to Maintain Market Data Processing Infrastructure

Maintaining market data processing infrastructure involves continuous updates, proactive support, and resource-intensive efforts, particularly when managing EDCs and ensuring system reliability. This section breaks down the key cost drivers associated with maintaining both software and FPGA-based feed handlers and evaluates the benefits of vendor-managed solutions.

Exchange-Driven Change Management

Exchange-Directed Changes (EDCs) are critical updates that exchanges mandate to ensure ongoing compliance and the functional integrity of market data feed handlers. These updates can range from minor documentation adjustments to significant protocol modifications requiring software changes.

Example

A major stock exchange may issue a protocol update to its market data feed. Failure to implement this change promptly could result in noncompliance, operational disruptions, and delayed trading activities.

Effective EDC management ensures that updates are assessed, implemented, or dismissed efficiently, minimizing downtime and maintaining system reliability.

4.1 Maintaining Software Feed Handlers

4.1.1 EDC Management Process

Before diving into the formulas, it is crucial to understand their relevance. These calculations simplify resource allocation by quantifying the workload associated with different types of EDCs, allowing teams or vendors to prioritize tasks effectively and optimize operational efficiency.

Exegy Internal EDC Data Overview

Based on Exegy's internal data from **2023**, the scale of EDC management is significant:

- **Total exchange notifications reviewed:** 8,356
- **EDC tickets created:** 481
  - **Non-software release tickets:** 275 **(57%)**
  - **Software release tickets:** 170 **(35%)**
  - **Tickets under review:** 36 **(8%)**

This data underscores the significant resource allocation required to manage both simple and complex EDCs.

4.1.2 EDC Categories and Workflow

EDCs can be divided into two primary categories based on their complexity and resource requirements:

1. **EDCs Requiring a Software Release**
   Involves code changes to adapt to new protocols or features

Workflow

- **Business Analyst (BA):** Reviews the update and drafts requirements
- **Software Engineer (SWE):** Implements code changes
- **QA Engineer:** Validates the update through rigorous testing

2. **EDCs Not Requiring a Software Release**
   Involves documentation updates or minor changes that don't impact the codebase

Workflow

- **Business Analyst:** Reviews the exchange documentation and validates changes

4.1.3 Estimating EDC Volume per Feed Handler

Require a Software Release

To estimate the number of EDCs requiring a software release for any number of feed handlers, we used 2023 internal data from the Exegy Ticker Plant:

- **73** total EDCs required a software release
- **118** active Ticker Plant feed handlers

The ratio is calculated as:

$$\frac{\text{73 EDCs}}{\text{118 Feed Handlers}} = \text{0.62 EDCs per feed handler}$$

Example Calculations

1 feed handler:

$$\left(\frac{73}{118}\right) x1 = \sim 1\ EDC\ annually$$

20 feed handlers:

$$\left(\frac{73}{118}\right) x20 = \sim 12\ EDCs\ annually$$

This calculation helps estimate the EDC workload based on the number of feed handlers a team maintains.

Do Not Require a Software Release

Based on 2023 data, **35%** of EDC tickets required software development. The remaining **65%** consisted of documentation updates or minor changes that didn't require software development.

Estimation Formula

To calculate the number of non-software release EDCs based on the number of software release EDCs:

- **35%** of total EDCs require a software release.
- **Non-software release EDCs:**

$$\frac{\text{Number of Software Release}}{\text{EDCs0.35}} \text{ - Number of Software Release EDCs}$$

For example, if a team maintains 20 feed handlers, it can be assumed that they will have at least 12 EDCs that will require a release; therefore, they will have ~22 EDCs that will not.

- **Software release EDCs:** 12 EDCs
- **Non-software release EDCs:**

$$\left(\frac{12}{0.35}\right) - 12 = 22\ EDCs$$

4.1.4 Calculating the Internal Team Effort per EDC Ticket

The following is an estimation of hours required for each role to address EDC tickets. The 30% UAT/development buffer accounts for typical inefficiencies due to varying complexities of tickets, team sizes, and unforeseen challenges during implementation.

This buffer ensures sufficient time is allocated to maintain high-quality outcomes while allowing for flexibility in resource management.

The times in the table are based on the time spent maintaining one feed handler.

Effort Requirements for EDCs with a Software Release

- **Business Analyst**
  - Spends an estimated **40 hours** per ticket, adjusted to **52 hours** with a 30% development buffer (~1.3 weeks)
- **Software Engineer (Development & QA)**
  - Combined effort of **140 hours** per ticket (80 hours for development, 60 hours for QA)
  - Adjusted to **182 hours** with a 30% buffer (~4.6 weeks)

Effort Requirements for Non-Software Release EDCs

- **Business Analyst**
  - Spends an estimated **8 hours** per ticket, adjusted to **10.4 hours** with a 30% buffer (~0.5 weeks)

EDC Management Effort Breakdown

- **Non-software release EDCs:** ~10.4 hours per ticket
- **Software release EDCs:** ~234 hours (combined BA, SWE, and QA efforts)

Role	EDC Type	Estimated Hours	With 30% Buffer	Total Weeks
Business analyst	Non-software release	8	10.4	0.5
Business analyst	Software release	40	52	1.3
Software engineer (development & QA combined)	Software release	140	182	4.6

4.1.5 Managing EDCs Across Multiple Feed Handlers

The effort required to manage EDCs does not scale linearly with the number of feed handlers. Factors such as shared resources, overlapping tasks, and team efficiencies influence the scaling process. As more feed handlers are added, incremental resources become necessary to maintain optimal system performance.

Scaling Engineering Resources

Based on internal calculations, additional software engineering and business analyst resources are required at specific scaling points:

- **Software Engineers (SWEs)**
  - A new SWE is typically needed after managing **16**, **35**, and **53** feed handlers.
  - On average, an additional SWE should be added for every **18-20** new feed handlers.
- **Business Analysts (BAs)**
  - A new BA is required after every **40** additional markets to handle documentation reviews and EDC assessments.

This complexity underscores the importance of precise planning and the potential advantages of vendor-managed solutions to optimize resource allocation and handle scaling effectively.

4.2 Calculating Yearly Costs for Market Data Maintenance

This section provides a detailed breakdown of the estimated team sizes and annual costs associated with maintaining market data infrastructure, including salaries, bonuses, and staff overhead for key roles involved in EDC management, system maintenance, and monitoring and support.

4.2.1 EDC Management Team Size and Costs

Managing EDCs demands a focused team of business analysts and software engineers. The table below summarizes the estimated yearly resource requirements and associated costs:

KEY ASSUMPTIONS

- **Working weeks:** Calculations assume **48 working weeks** per year (accounting for four weeks of time off).
- **Business analysts:** Review both software and non-software release tickets.
- **Software engineers:** Focus solely on EDC management, excluding system maintenance tasks.

Role	Count	Base Salary	Total Base Cost	15% Bonus	30% Overhead	Total Annual Cost
Business Analysts	1	$125,000	$125,000	$18,750	$37,500	**$181,250**
Software Engineers	1	$250,000	$250,000	$37,500	$75,000	**$362,500**
Total	**2**					**$543,750**

4.3 System Maintenance: Ongoing Engineering Efforts

Maintaining the underlying market data infrastructure requires consistent engineering effort for system enhancements, bug fixes, and performance optimizations.

Resource Allocation for Maintenance

The following table outlines the baseline engineering resources needed for system maintenance, independent of specific EDC calculations:

Role	Number of Engineers
Software Engineers (Improving Efficiency)	3
QA Engineers (Testing Improvements)	2

Role Descriptions

- **Software Engineers:** Focus on enhancing the feed handlers' performance, including building libraries, error reporting tools, and improving internal APIs. They also support QA engineers by automating testing frameworks.
- **QA Engineers:** Validate system improvements, ensuring performance gains meet required standards.

Note: The engineer count for EDC management has already been calculated separately and is not included in the table above.

4.3.1 System Maintenance Cost Calculation

In addition to EDC management, continuous system improvements require dedicated engineering resources. Below is the estimated team structure and cost for system maintenance:

Role	Count	Base Salary	Total Base Cost	15% Bonus	30% Overhead	Total Annual Cost
Software Engineers (Improving Efficiency)	3	$250,000	$750,000	$112,500	$225,000	**$1,087,500**
QA Engineers (Testing)	2	$250,000	$500,000	$75,000	$150,000	**$725,000**
Total	**5**					**$1,812,500**

4.4 Monitoring and Support: Ensuring Uptime

Maintaining real-time system availability requires continuous monitoring and rapid incident resolution. Dedicated Level 1 (L1) and Level 2 (L2) support teams ensure that issues are detected early and resolved efficiently to minimize downtime.

Role Descriptions

- **Level 1 Support Engineers (L1):** They are the first line of defense for handling routine issues, alerts, and monitoring. They are responsible for proactive system oversight and initial triage of incidents.
- **Level 2 Support Engineers (L2):** They address more complex technical issues requiring deeper system knowledge and expertise. They focus on root-cause analysis and long-term fixes.

4.4.1 Monitoring & Support Team Size and Cost Calculation

Monitoring and supporting market data infrastructure requires 24/7 dedicated L1 and L2 support engineers. The table below outlines the team size and associated costs:

Role	Count	Base Salary	Total Base Cost	15% Bonus	30% Overhead	Total Annual Cost
L1 Support Engineers*	5	$90,000	$450,000	$67,500	$135,000	**$652,500**
L2 Support Engineers**	3	$115,000	$345,000	$51,750	$103,500	**$500,250**
Total	**8**					**$1,152,750**

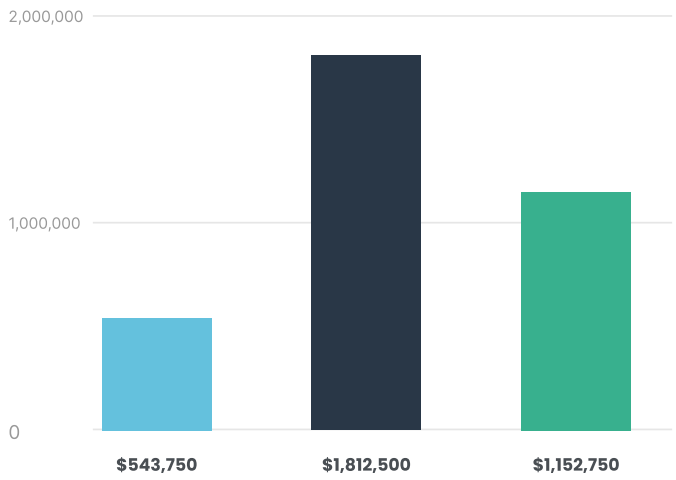
* L1 product monitoring engineers focus on real-time monitoring and alert management.
** L2 product support engineers handle complex issues requiring in-depth system knowledge.

4.5 Summary of Annual Maintenance Costs

Maintaining market data infrastructure incurs substantial recurring costs across different operational areas. This section consolidates the **yearly costs** associated with **EDC management, system maintenance, and monitoring and support** to provide a comprehensive view of the total annual maintenance expenditure.

4.5.1 Annual Maintenance Cost Breakdown

Maintenance Category	Total Yearly Cost
EDC Management	$543,750
System Maintenance	$1,812,500
Monitoring & Support	$1,152,750
Total Yearly Maintenance	**$3,509,000**



The total yearly cost of maintenance is around $3,500,000 for both Exegy and the Tier 1 bank's estimates for maintaining 1-15 markets.

Maintaining 15-35 markets, which would include all North American equities and commodities, is estimated to cost ~$3,871,500. If we look at the average of both figures, we estimate around **$3.7 million in yearly maintenance costs.**

Maintaining market data infrastructure incurs substantial recurring costs across different operational areas.

4.6 Vendor Advantage: Streamlining Market Data Maintenance

Maintaining real-time market data infrastructure—particularly in the face of constant EDCs—is a complex and resource-intensive task. Specialized vendors such as Exegy offer a compelling alternative, delivering cost efficiency, operational resilience, and strategic agility through managed services that scale with your business.

4.6.1 Simplifying EDC Management and Ongoing Maintenance

Vendors significantly reduce the internal burden of maintaining exchange compliance and platform stability:

- **Offloaded EDC Lifecycle:** Vendors manage the full lifecycle of exchange-driven changes, from tracking and interpreting specifications to implementation, testing, and deployment.
- **Proactive Monitoring & Support:** Around-the-clock system monitoring ensures high system availability and rapid issue resolution, backed by SLA-driven performance metrics.
- **Predictable Cost Structure:** Fixed pricing models reduce budget variability and improve forecasting for ongoing support and maintenance.

4.6.2 Strategic & Operational Benefits

By outsourcing to a trusted vendor, firms unlock long-term strategic value:

- **Increased Operational Efficiency:** Vendors accelerate the rollout of EDCs, reducing downtime, deployment risk, and compliance gaps.
- **Resource Optimization:**Internal teams can reallocate effort toward higher-value initiatives rather than managing exchange changes and routine infrastructure maintenance.
- **Scalable Market Coverage:** Vendor platforms make it easier to expand into new markets without the added complexity and cost of scaling internal systems.

4.6.3 Shared Intelligence: The "Crowdsource" Advantage

Exegy's solutions are not only engineered by experts—they are also shaped by insight gathered from hundreds of firms across the financial industry.

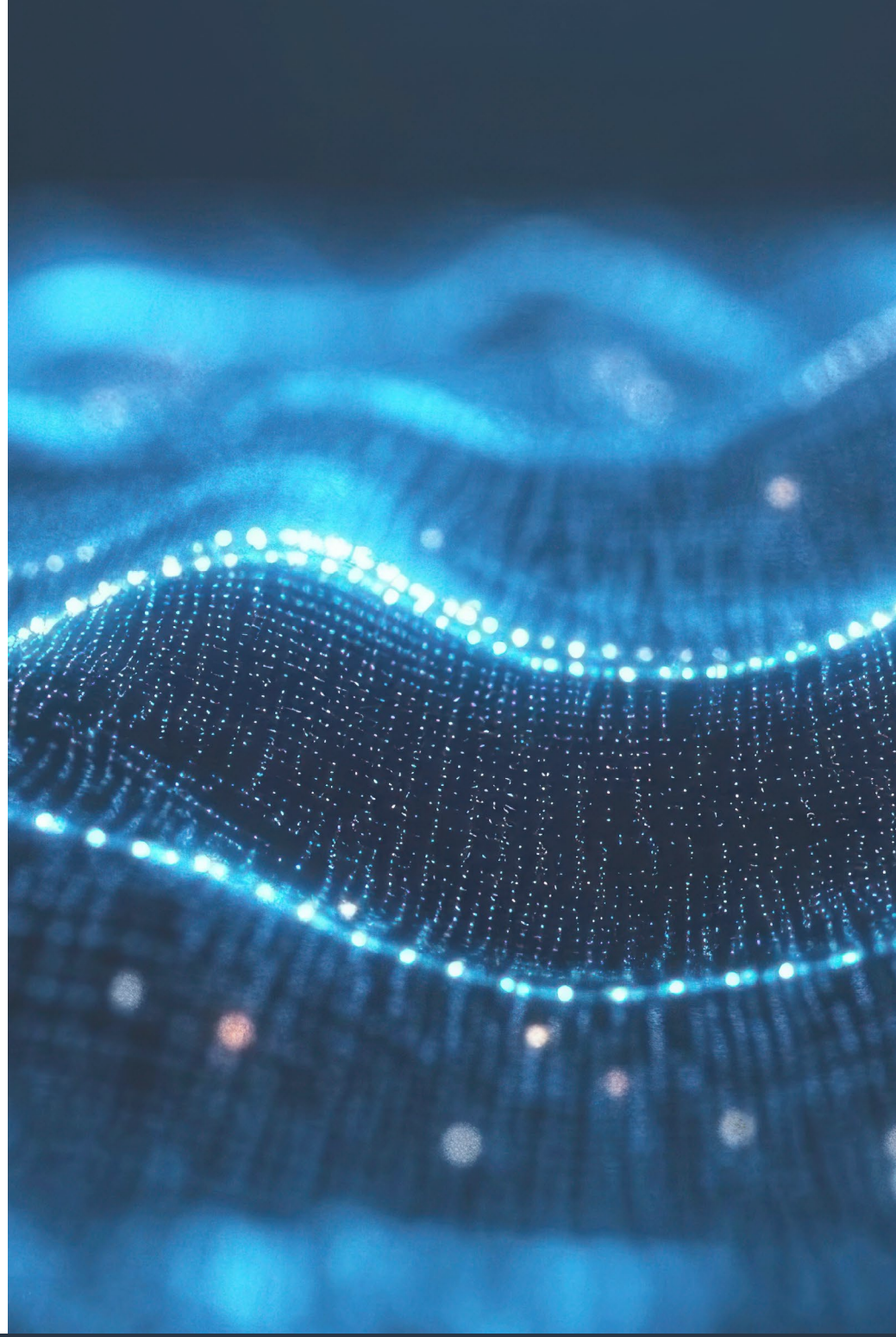
- **Proven Across Diverse Use Cases:** Each feed handler, normalization layer, and platform component is continuously tested in real-world production environments across trading strategies, geographies, and latency sensitivities.
- **Synthesis of Industry-Wide Insight:** Our teams distill feedback, bug reports, performance observations, and optimization opportunities from a broad client base—then apply those learnings to strengthen the platform for everyone.
- **Faster Innovation, Fewer Surprises:** Clients benefit from more resilient solutions, faster updates, and preoptimized configurations—shaped by the collective insight of Exegy's broad client base.

4.7 Strategic Takeaway: Optimizing In-house Management

As illustrated throughout this paper, in-house management of market data infrastructure comes with significant hidden costs, operational risks, and resource constraints—especially when scaling across multiple markets.

Outsourcing to a vendor such as Exegy enables capital markets firms to:

- **Leverage specialized expertise** for faster, more accurate execution of exchange changes while reducing internal workloads.
- **Reduce operational and compliance risk** with always-on support and rigorously tested updates, reducing the risk of outages and noncompliance.
- **Optimize internal resources** by freeing up engineering teams to focus on revenue-generating innovation.
- **total cost of ownership (TCO)** by minimizing maintenance overhead and improving system scalability.



5. In-House vs. Exegy: Comprehensive Cost & Time-to-Market Analysis

This section consolidates the data from previous chapters, providing a clear, concise comparison between in-house development and Exegy's commercial offerings for both software and FPGA-based market data infrastructures. It highlights costs, maintenance, and time-to-market differences, helping firms make informed decisions.

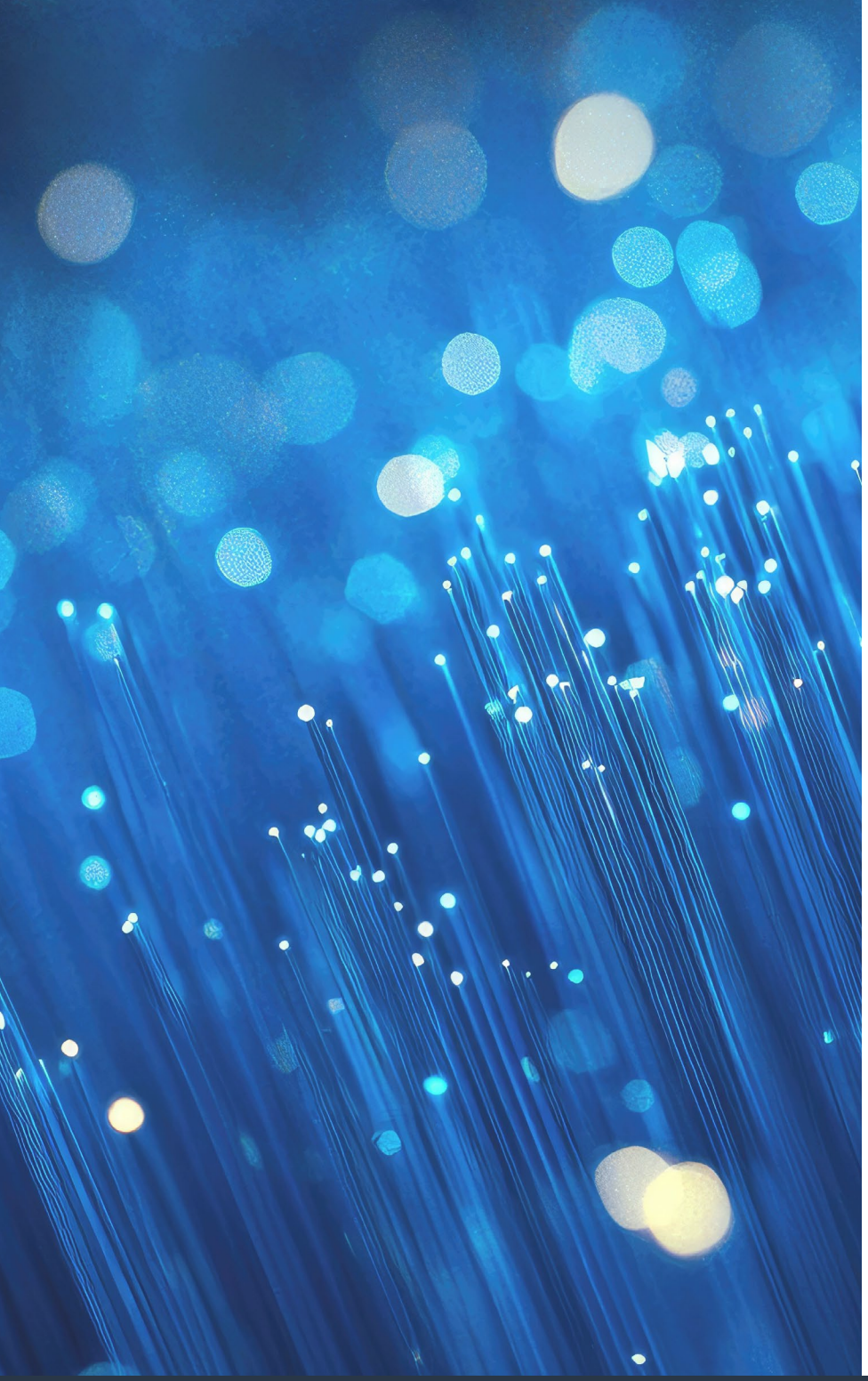
KEY TAKEAWAYS

- **Cost Savings:** Exegy solutions are **~8x cheaper** to build and significantly reduce maintenance overhead.
- **Speed Advantage:** Exegy delivers full market coverage in **4-6 months**, compared to **3.5 years** for in-house teams.
- **Lower Maintenance Burden:** Exegy includes maintenance in service costs, saving more than **$3.5 million annually.**

5.1 Software Feed Handlers: In-House vs. Exegy

Metric	In-House	Exegy	
Cost to Build (Initial Market)	$1,845,125	**$30,000 per market**	61.5x more costly with in-house
Cost to Build (Additional Market)	$152,250	**$30,000 per market**	5.1x more costly with in-house
Cost to Build (20 Markets)	$4,737,875	**$600,000**	8x more costly with in-house
Annual Maintenance Cost (20 Markets)	$3,871,500	**$1,470,000***	2.6x more costly with in-house
Time to Market (Build Initial Feed with a Team of Four Engineers)	~1.5 years	**4-6 months**	3x faster with Exegy
Time to Market (20 Markets)**	~3.5 years	**4-6 months**	7x faster with Exegy

* Yearly cost of Exegy's fully managed appliance for North American equities and commodities coverage
** Time to build North American equities and commodities coverage with a team of four engineers



6. Conclusion

The findings in this paper reveal that **building and maintaining software-based market data processing in-house is significantly more expensive, time-consuming, and operationally burdensome** than partnering with a specialized vendor.

Key takeaways from this analysis include:

- **The cost to build** the first in-house software feed handler exceeds **$1.8 million**, while full North American market coverage (20 protocols) **surpasses $4.7 million** or **~8x more expensive** than the equivalent coverage from Exegy ($600K).
- **Scaling in-house software solutions is slow and expensive,** taking more than **3.5 years to reach full market coverage** compared to only 4-6 months with Exegy, a **7x reduction** in time to market.
- Annual maintenance costs **reach $3.7 million annually**, primarily driven by **Exchange-Directed Changes (EDCs)** and system upkeep, increasing operational risk.
- Vendor-managed solutions reduce the internal resource burden, allowing firms to focus on revenue-generating activities.

6.2 Strategic Recommendation

For capital markets firms, the decision to build versus buy real-time market data infrastructure carries significant cost, time, and risk implications.

Maintaining in-house software feed handlers requires **substantial operational overhead** because firms must dedicate engineering resources to **EDC compliance, monitoring, and support**—diverting talent from revenue-generating trading and strategy development initiatives.

The data presented in this white paper clearly demonstrates that partnering with a specialized vendor such as Exegy offers:

- **Faster time-to-market** with lower up-front and ongoing costs.
- **Operational efficiency** through reduced internal resource demands
- **Long-term cost savings**, with maintenance costs slashed by more than half
- **Scalability and flexibility** to adapt to evolving market needs

For firms seeking a **cost-effective, scalable, and lower-risk approach**, vendor solutions provide clear advantages in both TCO and operational efficiency.

Strategic takeaway: Partnering with Exegy provides a faster, more cost-effective, and lower-risk path to achieving market data infrastructure goals—empowering firms to innovate, scale, and compete more effectively.

For firms considering FPGA-based alternatives, our companion white paper explores the cost implications of hardware-accelerated market data processing, comparing FPGA development costs, maintenance challenges, and vendor solutions.

Appendix A

A. Key Assumptions Underpinning the Cost Analysis

Our cost-benefit analysis is grounded in a set of key assumptions designed to ensure consistency, accuracy, and relevance. These assumptions draw on industry benchmarks, internal data from Exegy, and insights from a Tier 1 global bank.

Team Structure & Expertise

- **Dedicated market data team:** A specialized team is responsible for building, testing, and maintaining market data infrastructure, separate from application development teams.
- **Team composition:** An in-house engineering team specializes in market data environments and exchange processing with average annual salaries of ~$250,000, excluding bonuses or overhead costs.
- **Experience level:** Engineers have 2–10+ years of market data experience, primarily based in the greater New York area.
- **Global coordination:** Teams are globally coordinated for core processes but regionally specialized to address local market nuances.
- **Production support model:** Development teams combine with frontline support to ensure operational resilience.

Operational Scope

- **Data focus:** The analysis covers real-time, direct-feed market data normalization and distribution for North American equities.
- **System requirements:** Teams build and manage data processing and distribution to systems with varying latency needs, emphasizing low-latency performance where possible.
- **Consolidation objective:** The focus is on consolidating multiple data feeds for optimized, low-latency access across diverse systems and consumers.
- **Standardization approach:** While some exchange feed handlers are custom-made, there is increasing use of templated designs for efficiency.

Cost & Productivity Assumptions

- **Compensation:** All salaries are in USD and include a bonus of 15% of base salary, with overhead costs set at 30% of base compensation to account for support functions (legal, compliance, human resources, etc.).
- **Workload:** Cost calculations are based on 48 working weeks per year, assuming four weeks of leave.
- **EDC management effort:** Each EDC that requires code updates consumes an average of **140 hours**, based on Exegy's 2023 operational data.
- **Productivity rates:** A 30% productivity buffer was added to reflect real-world performance and operational inefficiencies.

Methodological Assumptions

- **Development model:** Initial build timelines cover full development, testing, and delivery cycles. For additional markets, efforts are split between the development of new feeds and ongoing maintenance.
- **Development timelines:** Assumes primarily sequential development across different teams due to the wide variability in resource constraints, engineering team organization, and workflows. Total times are used for each task within one engineering function with parallelization baked in.
- **Cost basis:** Includes fully loaded costs, such as salaries, infrastructure, software licensing, data center space, and ongoing maintenance.
- **Vendor benchmarking:** Exegy's cost estimates are based on historical data from similar client deployments over the past 20 years.

B. Key Components Required to Develop Initial Feed Handler

Exchange Line Handlers

Purpose: Responsible for connecting to and ingesting raw market data over network protocols (UDP Multicast or TCP) directly from the exchange.

Engineering Requirements

- **Protocol parsing:** Protocol decoding for specific market data formats (binary, FIX, proprietary formats)
- **Network stack optimization:** Low-latency UDP/TCP packet processing (e.g., using kernel bypass techniques, such as Solarflare Onload or DPDK)
- **Message framing:** Handling packet fragmentation, sequencing, and retransmission requests
- **Error handling:** Packet loss detection, heartbeat monitoring, and failover mechanisms for redundant feeds (A/B lines)
- **Performance tuning:** Memory management, CPU affinity, and lock-free queues

Reusability

- **Reusable:** Core networking libraries, threading models, gap recovery framework, packet capture, and transport layer abstraction
- **Unique per market:** Exchange-specific protocol parser (due to unique message formats), sequencing logic, and recovery mechanisms

Normalization Layer

Purpose: Convert raw exchange-specific data formats into a normalized, internal data model for downstream processing.

Engineering Requirements

- **Schema mapping:** Translate exchange-specific fields to a universal schema.
- **Data type conversion:** Handle different time stamp formats, price precision, and data encodings.
- **Latency optimization:** Minimize serialization/deserialization overhead for low-latency applications.
- **Time stamps and sequence numbering:** Normalize time stamps to a unified time reference (UTC, GMT, etc.) and assign sequence numbers for consistent event ordering.
- **Trade and quote message handling:** Process market data updates (bid/ask quotes, trades, auction states) and handle specific quirks in exchange behaviors (e.g., implied orders and auction logic).

Reusability

- **Reusable:** Normalization framework, common data models, and internal APIs
- **Unique per market:** Mapping logic, specific data conversions, and handling market-specific behaviors

Book Building (Order Book Management)

Purpose: Construct the real-time order book (Levels 1, 2, 3) from normalized data.

Engineering Requirements

- **Order book construction:**
  - Maintain price levels (L2) or individual orders (L3) for each security.
  - Apply exchange-specific rules for add, modify, and delete order events.
- **Concurrency control:** Make multithreaded book updates while maintaining data consistency.
- **Error handling:** Handle crossed book or erroneous conditions that can be detected.
- **Latency optimization:** Implement efficient data structures for rapid lookups and updates (e.g., balanced trees and hash maps).

Reusability

- **Reusable:** Core order book engine, data structures, and snapshot/recovery framework
- **Unique per market:** Edge-case logic for exchange-specific book behaviors (e.g., iceberg orders and hidden liquidity)

Distribution Layer

Purpose: Delivers processed market data to internal consumers (trading apps and analytics) and external clients with varying latency requirements.

Engineering Requirements

- **Publish/subscribe framework:**
  - Support for multicast, TCP, or messaging middleware (e.g., RocE and InfiniBand)
  - Build a high-performance messaging layer allowing for dynamic filtering of data
  - Dynamic subscription and resynchronization layer
  - Backpressure management and flow control mechanisms
- **Internal API or message bus:**
  - Push processed market data to internal consumers via a low-latency API compatible with multiple languages (C, C++, Java, etc.)
  - Optional compatibility with other API or messaging bus formats (MAMA, TREP, etc.)
- **Latency & throughput optimization:**
  - Batching, compression, and data throttling mechanisms
  - Low-latency serialization (data compression and avoidance of data duplication)
- **Latency monitoring & logging:** Track end-to-end processing latency and provide logging for debugging and compliance
- **Scalability:** Horizontal scaling to distribute load across servers, including fault tolerance and redundancy

Reusability

- **Reusable:** Distribution framework, messaging infrastructure, APIs, and performance optimizations
- **Custom for each market:** Minimal; mostly configuration changes unless data format requires modification

Additional Considerations

Testing & QA

- Unit and end-to-end testing with exchange and consumer simulators
- Latency benchmarking and failover testing

Project Management & Documentation

- Technical documentation, runbooks, and project oversight

exegy

exegy.com

St. Louis HQ	Chicago	New York	Montréal
Belfast	London	Paris	Manila