zapthink white paper

High-Performance Information Aggregation Using XML-Based Operational Data Servers

Empowering Flexible Data Aggregation in the Financial Services Markets

ZapThink, LLC • 11 Willow Street, Suite 200 • Waltham, MA 02453 • info@zapthink.com • www.zapthink.com

HIGH-PERFORMANCE INFORMATION AGGREGATION USING XML-BASED OPERATIONAL DATA SERVERS

EMPOWERING FLEXIBLE DATA AGGREGATION IN THE FINANCIAL SERVICES MARKETS

May 2003
Analyst: Ronald Schmelzer

Abstract

Financial Service Providers are essentially information-based businesses: their primary asset is the information they store and share. These companies are struggling today with finding the most flexible and cost-effective means to integrate and aggregate information from a wide range of unstructured and semi-structured enterprise data sources. Today's integration solutions are either targeted only at structured sources of information such as databases, or are too rigid and expensive to handle enterprise information integration and aggregation needs. Furthermore, the unique requirements of Financial Services firms for real-time, scalable access to disparate information has not been successfully met by today's vendors.

This paper presents a solution in the form of a mid-tier information aggregation server that provides an XML-based operational data store as a way of providing aggregated access to multiple data sources. Raining Data's TigerLogic XDMS is used as a strong example of an XML operational data server that provides a way to store and aggregate semi-structured and structured data from a wide variety of sources, XML-enables access to aggregated information and does not require the user to know the structure of data in advance or make any modifications, mapping, or transformations to those data.

Targeted at line-of-business users, corporate architects, as well as IT managers at Financial Services firms, this paper helps readers find a better way to integrate and aggregate disparate systems in the enterprise while lowering costs and off-loading performance from back-end operational servers. Readers will gain a critical understanding of how XML operational data stores can be used in the mid-tier to aggregate data from the heterogeneous sources in the Financial Services enterprise.

Table of Contents

I. Simplifying Data Integration and Aggregation in Financial Services

Rapid and continuous change is a constant for Financial Services companies. Financial Services Providers (FSPs), including a wide range of businesses and industries including Equity and Fixed Income Trading, Commodities and Currencies, Investment Banking, Retail and Commercial Banking, and various financial-related fields, are constantly exploring new channels, increased operational efficiency, and greater visibility into processes. At the same time, mergers and acquisitions, the accelerating the pace of competition, e-Commerce initiatives, fraud prevention, globalization, and the move towards real-time processing of financial transactions are creating new challenges and business opportunities for FSPs. Complex, heterogeneous integration environments and the dependency on aging, legacy systems further challenge implementations of solutions to these problems. While the specific nature of each FSP's business differs, the industry's main concerns are the same: how to improve informational representation and flow in order to improve profitability and business agility.

The Financial Services industry's main concern is how to improve informational representation and flow in order to improve profitability and business agility.

First and foremost, Financial Services firms are information-centric businesses. Other than the cash that comes out of the ATM machine and various pieces of print material, there are few physical products in the financial services world. At the end of the day, account balances, customer portfolios, stock positions, currency values, loan balances, and the vast majority of other financial products exist solely as information stored in a variety of databases, repositories, and file formats. Not only are Financial Services companies dependent on information for their day-to-day operations, but the information itself is also highly valuable. Network or system outages, inefficient processes, or human-intensive operations can dramatically and critically injure a Financial Services business. On the flip side, any improvements in data storage, exchange, representation, and manipulation can greatly add to an FSP's bottom-line.

The Need to Integrate and Aggregate Information

Since FSPs are so reliant on information for their day-to-day operations, they have implemented almost every known data storage, middleware, and exchange technology that has been created since the 1950's. These systems span many different operating system, application, and exchange architectures and are typically cobbled together to produce aggregated results that firms use to drive decision making and financial products and services. As a result, one of the primary costs in the Financial Service enterprise is that of integration. The

Financial Services sector, like many other industry groups, has two primary integration challenges to deal with:

- Internal, system-to-system integration
- External, business-to-business integration

The need for internal integration is spurred by the desire to link multiple systems to provide a cohesive view of data that can power decision-making, customer interaction, and the delivery of integrated products and services. Due to the greatly inter-related nature of financial systems and products, internal integration is a perennial challenge for FSPs. Counterbalancing the urgent desire to integrate systems is the fact that most of these enterprises consist of a mish-mash of heterogeneous systems and architectures that span decades of legacy systems, multiple hardware platforms, operating system versions, database storage technologies, network protocols, component object models, middleware platforms, programming languages, and file formats. Of course, what is needed is to turn these individual ingredients into a cohesive stew without fouling up the whole concoction.

Matched with the complexity of internal integration is the desire to connect supply chain partners, affiliate networks, distribution channels, and even customers. FSPs have been pioneers in the area of *business-to-business integration* (B2Bi) since the early days of EDI. To this day, most financial transactions occur using long-standing network protocols such as SWIFT and FIX along with Financial EDI. The need to standardize integration protocols is tremendous, as the cost of supporting multiple exchange protocols, or even adding a new one, adds an inordinate amount of cost – especially since almost every component in the heterogeneous environment needs to support these protocols. Any significant change in communication protocol requires a cascading set of changes to back-end systems, resulting in very high support and development costs.

Cutting across external and internal integration needs, companies need to integrate data and information from a wide variety of data sources. These data sources might be structured, as in the case of databases and enterprise applications, or might be semi- or unstructured such as web pages, PDF documents, Office files, email, media content, or a wide variety of data feeds and formats. The need to access information of so many disparate types from so many disparate sources and locations forms the integration challenge that most companies must deal with today.

XML: Empowering Data Interchange and Aggregation

The desire to simplify access to aggregated information is one of the primary pressures and drivers for the adoption of the Extensible Markup Language (XML).

This desire to simplify integration is one of the primary pressures and drivers for the adoption of the *Extensible Markup Language* (XML). XML, and system-to-system integration technologies based on XML such as Web Services, are fast becoming a pervasive part of nearly all enterprise data sharing today. XML provides a standardized, versatile, cross-platform way to describe and encode data, enabling business communications that had previously been too complicated, expensive, or infeasible to automate.

This primary benefit of XML allows users to realize a potent solution to today's integration challenges: standardizing data access and encoding among heterogeneous systems of all types. Most current integration solutions encompass a total of at least six systems, and the increasing movement towards greater use of data in real-time enterprise environments, B2B systems, and efforts to make the most of legacy systems is driving a need to integrate with

dozens, if not hundreds of systems in a single environment. As the number of connection points increase, so do the complexities and inefficiencies of data transformation, manipulation, and exchange.

The main reason why XML is well suited to solving the problem of the different classes of integration challenges is that it is capable of representing data and processing information in an application neutral, open, and extensible manner. This means that vendors aren't locked into proprietary choices for integration technology. Vendor and platform "lock-in" are the primary reasons for much of the integration nightmare that exists today. If there was a single open, extensible mechanism for data interchange, there wouldn't be as much of an integration challenge.

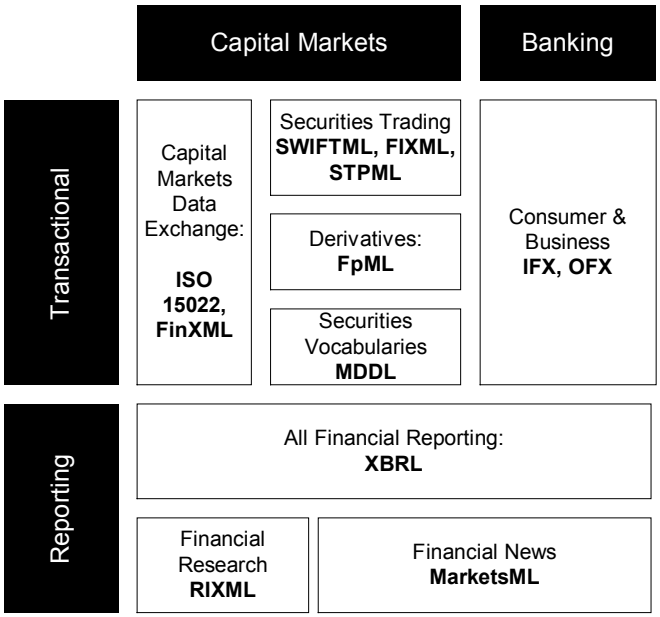
Once information is aggregated and transformed into an XML representation, companies can minimize, if not entirely eliminate, the need for unnecessary data extraction and transformation to satisfy subsequent inquiries. Enterprises are thus able to leverage persisted "views" of disparate data sources, maximizing the efficiency of their information storage and retrieval needs.

Simplifying integration and aggregated data access challenges are not only on the cost-saving side of the ROI curve, but also presents tremendous opportunity for revenue enhancement. Simplified integration and data aggregation allow FSPs to quickly develop and deliver partnerships, more easily respond to emerging opportunities, and rapidly deliver product offerings or variations on existing product offerings. Thus, XML-enabled information aggregation can compress time to market, providing FSPs with greater profitability and competitive advantage.

Key XML-based Specifications for Financial Services Providers

A single integration architecture or approach is not sufficient to guarantee interoperability among businesses and systems. In order to facilitate interoperability, XML-based standards are needed to guarantee inter-organizational information exchange. As a result, a number of key industry initiatives have emerged to standardize information interchange among Financial Services Providers. These standards are illustrated and described below:

Financial XML Standards Landscape



Source: Copyright © 2002, ZapThink, LLC

- *Extensible Business Reporting Language (XBRL)* – An open, freely licensed specification that uses XML to describe financial statements for both public and private companies. The format provides a standard format in which users can prepare business and financial reports that can be subsequently presented in a variety of ways.
- *Research Information Exchange Markup Language (RIXML)* – An XML-based format that aims to provide tags for any piece of research content, in any form or media, with enough meta-data information for consumers to search, sort and filter through publisher research and quickly provide highly relevant information to their decision makers.
- *ISO 15022 XML* - ISO 15022 replaces older standards for securities industry electronic messages (such as SWIFT for financial settlement and FIX for financial trades), including ISO 7775 for securities messages and ISO 11521 for inter-depository messages. The format provides a general framework by which different communities can define message types specific to their processes.
- *Financial Products Markup Language (fpML)* – A format for the trade of derivatives and other financial instruments. fpML provides an XML-based specification that is focused initially on over-the-counter (OTC) financial instruments directly traded between banks and other financial institutions. The specification includes trade execution syntax for OTC instruments such as forward rate agreements, interest rate swaps and options, equity derivatives, and foreign exchange, including spots, forwards, swaps, options, and option strategies.
- *Market Data Definition Language (MDDL)* – MDDL seeks to standardize an architecture for defining community vocabularies for securities prices, volume, bids and offers, and other trading concepts.

Clearly, there is a significant amount of attention on the challenge of simplifying integration both internal and external to the FSP organization and using XML to simplify and streamline access to aggregated information. However, it is far from clear how to actually go about implementing an integration approach that leverages the power of XML without sacrificing any of its promised business benefits.

Companies need to leverage the power of XML without sacrificing any of its promised business benefits.

II. Challenges with Existing Information Integration Approaches

Many of today's organizations look to solve their integration challenges using "point-to-point" integration approaches where systems that need to communicate are connected directly to each other. In this scheme, the number of integration or interconnection pathways that must be established grows geometrically with the number of systems to be integrated. The end result is a tangled web of point-to-point integrations that fails to meet business requirements over time. With traditional integration approaches, the costs for both maintaining and changing systems can become exorbitant, since developers must recode all applications that are impacted by changes. Enterprises are thus faced with an integration problem that grows at a rapidly increasing rate.

In addition, many enterprises' core operations are enabled and powered not only by electronic applications but also by paper-based processes. Insurance industry firms depend on paper claims; banks would cease to operate without paper forms and documents; financial services firms rely on print documents for archival purposes. Even today, many of the supposed electronic trading environments depend on print documents as the sole means of connecting disparate systems. Indeed, for many industries, the promise of the paperless office has failed to materialize. However, there is still a tremendous desire to move beyond human-intensive, paper-based processes to automated, electronic transactions.

There is still a tremendous desire to move beyond human-intensive, paper-based processes to automated, electronic transactions.

Enterprise Information Integration Approaches: ETL

One mechanism for accessing system data is to directly access the underlying databases and file structures that store the application information. This form of data access incorporates three major concepts: *extract, transform and load* (ETL). However, ETL systems are brittle and hard to change. Once a mapping from a source data to destination format has been defined, any changes to the source file will invalidate the mapping and cause the integration process to fail. Furthermore, managing the mappings and conversion steps is often a complex, cumbersome, and error-prone process. ETL approaches are also by definition point-to-point, tightly coupled, and asynchronous, since the data load process can occur at any time – most likely in batch processes when the data is already "stale."

Data Integration Middleware: EAI and B2Bi

Companies have sought to improve the mechanisms for integration between systems by implementing a middleware tier such as those provided by traditional *Enterprise Application Integration* (EAI) or *Business-to-Business Integration* (B2Bi) solutions. These solutions are built primarily on proprietary or system-specific messaging platforms that aim to provide a complete, end-to-end platform for integrating and communicating with various business components, primarily focused on structured data transformations through application API's.

The typical method for accessing these structured systems is through a wide assortment of pre-built adapters that provide bi-directional connectivity to many types of applications and data sources, such as enterprise software applications, databases, file systems, directories, as well as mainframe and other legacy applications. In simple terms, the way these integration solutions work is by extracting or inserting mostly structured data from these various adapter-enabled systems, transforming the data and converting their representation or schema to a different format, and then shipping the data to their destination.

The primary downside to EAI is that up-front costs are much higher than other integration approaches. In a typical EAI solution, end-users must spend from tens of thousands to millions of dollars on software licenses and server systems prior to completing any integration. The actual integration project itself typically ends up costing many times more than the initial costs and can easily dwarf the costs of custom integration projects. Furthermore, when systems, business processes, or major assumptions change, EAI system costs can spike. In fact, we say that EAI systems "pour concrete on business processes," since they tend to solidify existing processes rather than enable an IT environment that allows companies to deal easily with change.

EAI systems "pour concrete on business processes," since they tend to solidify existing processes rather than enable an IT environment that allows companies to deal easily with change.

In addition, EAI doesn't work well across more than one company, because there is no single programming team that can control all the interfaces. Traditional EAI and B2Bi solutions have sought to solve many integration headaches by presenting an architecture that efficiently manages and maintains connections among systems and between enterprises. And, for systems that are not accessible on the network, EAI-type solutions fail to provide any solution. EAI fundamentally is a centralized integration process – which doesn't work well for decentralized data.

The Requirements for Data Aggregation

What companies need to solve these integration cost and complexity issues is a loosely coupled yet scalable approach to data integration.

What companies need to solve these integration cost and complexity issues is a *loosely coupled* approach to integration—one that scales and does not require control of systems on both ends nor an intimate connection between the requester of information and the provider. However, for loose coupling to be a reality, there must be standard, established ways of handling integration so that any company that follows the standards can be confident that their systems will interoperate with other systems that the company wishes to communicate with. XML represents the standards-based way of representing information, and *Web Services* have come to signify the overall movement toward standards-based approaches to distributed computing. This standards-based approach to loosely coupled integration is what the promise of XML and Web Services are all about.

What is needed is a real-time, accurate, integrated, and efficient method for aggregating information from multiple sources, bringing together both structured and unstructured data to feed operational data requirements of Web Services, Server Oriented Architectures (SOA) and service-oriented processes. This method needs to connect to many different back-office financial systems as well as customer-facing front-office and B2B systems. An intelligent, XML-based data aggregation solution is needed to provide an effective means for integration, while also providing a platform neutral and "future-proof" technology base on top of which an FSP's business can run.

III. The XML-based Mid-Tier Operational Data Server as a Flexible Data Aggregation Solution

There are two primary ways that organizations can use XML to facilitate data integration and aggregation: the use of an XML-based middleware layer that queries individual data sources and merges the results in real time, or the use of an XML store to merge data results from multiple sources and persist them in the mid-tier for subsequent reuse. The first approach is also known as the XML "virtual" DBMS system, sometimes known as Enterprise Information Integration (EII), which exposes a database interface to end users, abstracting the actual final location of the data. In this approach, a middleware system processes inbound queries and then makes requests for information across a wide range of end systems, processing the result and returning an aggregated response to the querying system.

While this mechanism certainly serves to solve many needs for heterogeneous data access, the performance of the middleware data integration solution depends entirely on the response time of the *least* efficient data source. Basically, if one system in the aggregated query operates very slowly, the overall query time will become very slow. Rather than just mediating queries for the purpose of integrating disparate data sources, users need an efficient means of gaining access to available data without having to be bogged down by the bottlenecks of the aggregated systems.

Rather than just mediating queries for the purpose of integrating disparate data sources, users need an efficient means of gaining access to available data without having to be bogged down by the bottlenecks of the aggregated systems.

Most RDBMS treat XML as another relational data type and require development efforts to map and flatten XML hierarchies into tables. This results in too many restrictions on XML structure at the expense of scalability.

XML-Enabled Aggregation Requires Native XML Storage

There are a number of alternative platforms to store XML-enabled data, including XML repositories, native XML data stores and XML-enabled Relational Databases (RDBMS). XML repositories tend to treat XML as text documents and, while offering extensible support for XML semantics, lack scalability and transactional integrity of databases. Conversely, RDBMS provide proven data management platforms. Most of these treat XML as another relational data type and require significant development efforts to map XML hierarchies into two-dimensional tables. This results in far too many restrictions on XML structure at the expense of scalability.

While each offers their respective set of features, the requirements for storing XML-enabled aggregated data in the mid-tier are best met by native XML data stores, particularly ones that can provide both the scalability and transactional integrity of databases as well as extensibility and flexibility of repositories. While many such offerings are relatively immature, Raining Data's TigerLogic XML Data Management Server (XDMS) provides high-performance XML profiling and indexing features as well as information aggregation and persistent storage that are based on the company's time-proven Pick Universal Data Model (UDM).

The XML Operational Data Server for Scalable Information Aggregation

The second approach to mid-tier data integration uses a data storage system called an *Operational Data Store (ODS)* to physically store aggregated data and associated metadata in a data store or repository. An operational data store is a short-term data repository that contains information of an "operational" nature that is used for enterprise-wide tactical decision-making. As a result, a mid-tier ODS contains live, or near real-time, data, not information snapshots, and retains minimal history. The operational data store can then be queried for the required aggregated data results. This approach allows users to avoid the problem of system inefficiency due to slow data sources that would pose problems in the virtual DBMS approach. Some of the features and capabilities of an ODS include

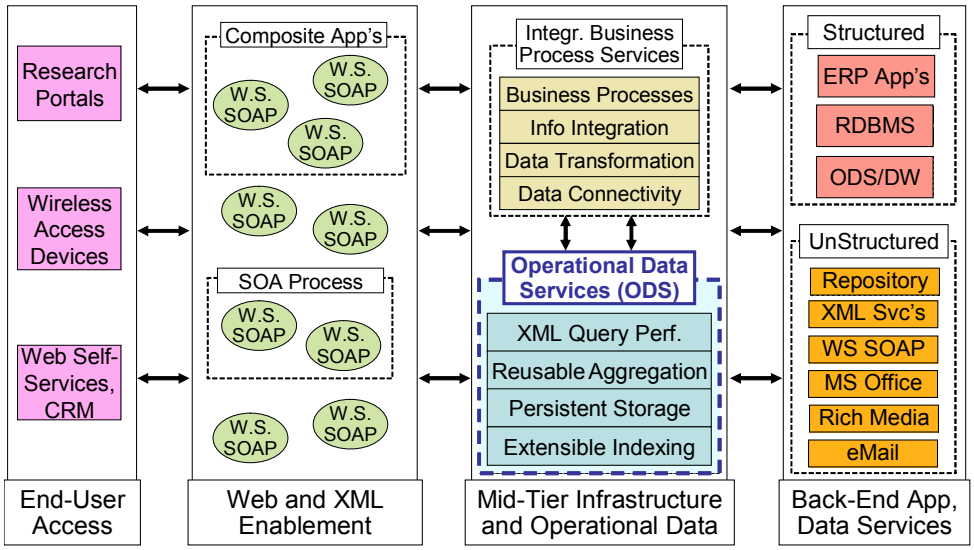
An XML operational data store provides a system that can quickly index, query and retrieve disparate data using XML as the intermediating format.

the ability to support a wide range of source application and data types, the ability to map source elements between documents of different types, the provision of a single API for queries across multiple system interfaces, support for multi-system transactions, and support for asynchronous queries on otherwise synchronous systems.

One of the best ways to implement a mid-tier ODS is through the use of an XML data store. Rooted in SGML, HTML, text, object-oriented, relational, and hierarchical storage, the *"Native" XML Data Store* (NXD) provides a way to store and aggregate semi-structured and structured data from a wide variety of sources, represented in XML, and stored without requiring the user to know the data structure in advance or make any modifications, mapping, or transformations.

Such an XML ODS's most valuable attribute is its ability to store arbitrary and highly variant XML documents. An XML ODS also makes comprehensive use of format-specific XML standards and specifications such as XPath, XQuery, XLink, XML Schema, and XSLT. Using these capabilities, XML data stores allow developers to retrieve entire documents, sub-document trees, and update part or all of these documents locally within the database. In this context, data integration and aggregation is accomplished by managing XML data at the element level rather than at the document level. The XML ODS can also manage collections of documents, allowing developers to query and manipulate XML documents as a set, similar to the relational concept of the table. However, unlike the requirements of relational databases, the XML data store doesn't require the user to pre-define schemas on the collection of XML documents. Regardless of document schema, users can construct queries that cut across all documents in the collection. The diagram below shows how a mid-tier ODS can be used in practice:

How a Mid-Tier ODS Boosts Scalability and Agility of Service-Oriented Architectures



As a result, a mid-tier XML ODS provides a system that can quickly index, query and retrieve disparate data using XML as the intermediating format. The XML ODS is a schema-independent XML data store that can correlate data described by widely variant data structures. By using data mining techniques that combine

the best of structured and unstructured queries, the store allows users to submit data of any schema, without requiring changes to existing applications or schemas. Through the use of notification events, the system can see changes as they happen in remote system through a push or pull mechanism. XML-based operational data stores allow sophisticated data rules to be associated with these events and permit administrators to recognize and take action on relevant trends.

However, not all XML data stores can be used for data integration. In particular, an XML-based operational data store must support arbitrary XML data formats without requiring schemas as well as non-XML data source types. In addition, if the system is to be used in a B2B environment, the XML-based ODS must be able to support long-lived, asynchronous transactions, deal with a wide range of data formats that are not in the control of the parties in a transaction, and handle insecure exchange environments.

IV. Raining Data's XML-Enabled Data Aggregation Solution: TigerLogic XDMS

Developed by Raining Data, the *TigerLogic XML Data Management Server* (XDMS) is a flexible and scalable XML operational data store that aggregates information across structured and unstructured data sources and delivers significant improvements in XML query performance over alternative XML data management approaches, including XML-enabled relational or object databases.

TigerLogic XDMS acts as a "mid-tier" operational data store that can manage and store data of any type persistently, transform it into XML while offloading processing from back-end servers.

Designed to sit on the middle tier between front-end data gathering applications and back-end data sources, TigerLogic XDMS acts as a "mid-tier" operational data store that can manage and store data of any type persistently and transform it into XML. Whereas relational databases parse, shred and then map XML into tables and fields, TigerLogic XDMS stores XML in its native format and reduces the processing load of an infrastructure by creating a data-neutral environment that substantially reduces mappings and transformations and offloads common data requests from overloaded operational data stores.

TigerLogic XDMS leverages Raining Data's experience in providing reliable, scalable and self-managing data management software with the extensible *Pick Universal Data Model* (Pick UDM) that uniquely enables optional XML schema validation of incoming content. This feature, along with the product's patent-pending XML profiling and indexing technology, are the key ingredients that allow TigerLogic XDMS to leverage Pick UDM to deliver high levels of performance without sacrificing critical ODS capabilities.

TigerLogic XDMS provides a facility to persist aggregated views of commonly requested data and leverages XML to satisfy the real-time operational data needs of critical applications and enable them to perform at maximum levels. The product can work with Enterprise Application Integration (EAI) or Enterprise Information Integration (EII) systems to allow enterprises to off-load application requests from the back-end, whether the data source is a relational ODS, enterprise resource planning (ERP) server or other system of record. In addition, the flexibility of TigerLogic's Pick UDM and XML query engine facilitate the dynamic addition of data sources to be included into aggregated data views.

TigerLogic XDMS architecture consists of a Connectivity Client, XML Query Engine and XML Storage Engine layers. Applications written in Java, C, C++, Visual Basic or other languages can access TigerLogic XDMS either remotely or locally. In addition, TigerLogic XDMS leverages over 25 years of Raining Data's experience in developing reliable and scalable data management systems. The product

includes on-line backups, XA-compliant transactions with granular locking and concurrency management, automatic allocation and dynamic extension of disk space and other enterprise platform software features.

TigerLogic XDMS complements EAI and EII approaches by providing a rich set of mid-tier data management and queriable information for reusable data aggregation.

TigerLogic XDMS Complements EAI and EII for Better Performance and Flexibility

As illustrated in the table below, the TigerLogic XDMS complements EAI and EII approaches by providing a rich set of mid-tier data aggregation, management, and query features not provided by other approaches.

	TigerLogic XDMS	EAI	EII
Data Sources Supported	Structured, Semi-Structured, Unstructured	Structured	Structured, Unstructured
Business Rules and Workflow		✓	
Data Transformation	Some	✓	✓
Connectivity Adapters		✓	✓
Tightly-Coupled Integration		✓	✓
Loosely-Coupled Integration	✓		Some
XML Queries	✓		Some
Reusable Data Views	✓		Some
Persistent Storage	✓		
XA-Transaction Support	✓		
Back-End Off-loading	✓		

V. Conclusions

Integrated access to aggregated information has been a headache and challenge for most Financial Services enterprises for decades. Traditional data and application integration solutions have offered some relief, but at great expense, complexity, and rigidity. FSPs clearly need a solution that not only can integrate a wide variety of data in the enterprise, but also embrace their heterogeneity – and do so without sacrificing the critical performance FSPs require.

The XML-based mid-tier operational data server provides many of the capabilities that meet these critical requirements. Rather than requiring a middleware layer that seeks to integrate systems in way that introduces new performance bottlenecks, the ODS provides a reliable, efficient, and cost-effective means to aggregate and integrate data through the use of an intelligent, XML-based data store. These solutions not only provide the ability to integrate a wide range of disparate data, but also support long-lived, asynchronous transactions, deal with a wide range of data formats that are not in the control of the parties in a transaction, and handle insecure exchange environments.

Companies looking to implement this solution should investigate the Raining Data TigerLogic XDMS product.

VI. Raining Data Profile

Profile: Raining Data	(May 2003)

Date Founded:

2000: Raining Data Corp., a merger of Pick and Omnis

1973: Pick Systems Established

Funding: Publicly-traded (Nasdaq: RDTA)

Chairman: Geoffrey P. Wagner

CEO: Carlton H. Baab

CFO: Brian C. Bezdek

VP/GM, XML: Boris Geller

Employees: 145+

Address:

17500 Cartwright Road
Irvine, CA 92614-5846, USA

URL: www.rainingdata.com

Main Phone: +1-800-367-7425

Contact:

Boris Geller (boris.geller@rainingdata.com)

Copyright, Trademark Notice, and Statement of Opinion

About ZapThink, LLC

ZapThink is an IT market intelligence firm that provides trusted advice and critical insight into XML, Web Services, and Service Orientation. We provide our target audience of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink's role is to help companies understand these IT products and services in the context of SOAs and the vision of Service Orientation. ZapThink provides market intelligence to IT vendors who offer XML and Web Services-based products to help them understand their competitive landscape and how to communicate their value proposition to their customers within the context of Service Orientation, and lay out their product roadmaps for the coming wave of Service Orientation. ZapThink also provides implementation intelligence to IT users who are seeking guidance and clarity into how to assemble the available products and services into a coherent roadmap to Service Orientation. Finally, ZapThink provides demand intelligence to IT vendors and service providers who must understand the needs of IT users as they follow the roadmap to Service Orientation.

ZapThink's senior analysts are widely regarded as the "go to analysts" for XML, Web Services, and SOAs by vendors, end-users, and the press. They are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted industry analysts in the IT industry.

ZapThink was founded in October 2000 and is headquartered in Waltham, Massachusetts. Its customers include Global 1000 firms, public sector organizations around the world, and many emerging businesses. ZapThink Analysts have years of experience in IT as well as research and analysis. Its analysts have previously been with such firms as IDC and ChannelWave, and have sat on the working group committees for standards bodies such as RosettaNet, UDDI, CPExchange, ebXML, EIDX, and CompTIA.

Call, email, or visit the ZapThink Web site to learn more about how ZapThink can help you to better understand how XML and Web Services impact your business or organization.

ZAPTHINK CONTACT:
ZapThink, LLC
11 Willow Street, Suite 200
Waltham, MA 02453
Phone: +1 (781) 207 0203
Fax: +1 (786) 524 3186
info@zapthink.com

ZapThink, LLC • 11 Willow Street, Suite 200 • Waltham, MA 02453 • info@zapthink.com • www.zapthink.com