



W H I T E P A P E R

Standards-based Integration in Financial Services

The promise of XML, Web Services
and the Enterprise Service Bus

Contents

Introduction	3
The Promise of XML & Web Services	4
XML: the Integration Standard	4
Adoption and Benefits in Financial Services	5
The Technology Reality Gap	6
The PolarLake Solution	6
The CIO's Dilemma	7
The PolarLake Solution	8
The PolarLake Product Set	9
XML & the Financial Services Industry Today	11
The Extensible Markup Language (XML)	11
Web Services and XML	12
The Enterprise Service Bus	13
Service Oriented Architectures	14
Some Business Uses for XML & Web Services	15
Key Issues and Problems Encountered	17
Resolving Key Issues with PolarLake	18
Performance and Scalability	18
Semantic Validation, Transformation & Enrichment	19
Intelligent Exception Handling	20
Business Activity Modelling	21
Intelligent Routing	21
Integration with Key Technologies and Message Formats	22
J2EE™ Integration	22
Database Integration	22
Messaging Oriented Middleware Integration	23
Integration with Packaged and Mainframe based Applications	23
Integration with Non-XML formats	23
Deploying and Managing PolarLake Applications	26
Conclusions	28

Introduction

Traditionally, application or data integration required the use of Enterprise Application Integration (EAI) products or Application Servers. The promises offered by moving to XML and Web Services standards for this class of project include greatly reduced development and deployment costs, as well as an ability to respond much more quickly to changing business requirements by incrementally developing their solution. The size and scale of these benefits are dramatic, with customers reporting minimum savings of 25% rising to 75% on development times as well as significant reductions to deployment and on-going maintenance costs. The challenge is to unlock those benefits while minimizing the risk and investment.

PolarLake™, a leader in standards-based incremental integration, provides a complete suite of products for implementing XML and Web Services-based solutions, including those based on the Enterprise Service Bus architecture. PolarLake's products deliver rapid Return on Investment by focusing on solving high value business problems with a standards-based approach designed to evolve and expand to address the longer-term objectives of the organization. PolarLake's global network of offices and solution partners (including Sun Microsystems and Hitachi Systems and Services) support customers in delivery of the most complex projects. PolarLake's involvement with leading standards organizations such as SWIFT and the EAI Consortium ensure that PolarLake solutions support the latest standards and solution architectures.

PolarLake™ products achieve the following:

- Enable the integration of applications within and between organizations to automate new and existing business processes.
- Fit easily within the existing investment in enterprise infrastructure (e.g. messaging systems), handling integration with common application architectures (e.g. databases and J2EE) as well as common infrastructures such as system management platforms and transaction managers.
- Exposing application infrastructure as a distributed (Web) services oriented architecture.

- Provide a “small footprint”, high performance solution, necessary in today’s need to focus on delivering projects on time, to budget and with a tangible and attractive return on investment.

PolarLake has a proven track record in delivering the benefits of incremental integration since 2000. Deployed customers include leading corporations in financial services such as Pioneer Investments (Ireland), Man Financial Ltd (UK), Nissay Dowa (Japan) as well as leading telecommunications and government organizations.

The Promise of XML & Web Services

XML: the Integration Standard

Using XML* as a common data format across all systems can significantly reduce the costs, complexity, project risk and development timescales associated with integration projects. Building on XML and the principles of a Service Oriented Architecture (SOA), Web Services allow applications to provide *software services* to other applications in an infrastructure neutral manner, where the sending and receiving of specific XML documents represent each service.

While XML and Web Services were originally conceived for a B2B environment, it is now clear that they can also provide cost and time benefits when deployed *within* the enterprise to integrate systems and business processes. One approach to achieving this that is rising to prominence is the **Enterprise Service Bus** (ESB). This provides a distributed message-oriented architecture, supporting traditional EAI features such as message routing and transformation, within the context of Web Services and Application Servers.

Later in this white paper the issues associated with handling ever more complex XML messages and an ever-increasing number of messaging standards will be addressed. However, prior to that it is worth identifying the key drivers behind the interest in XML in financial services from both a standards and business problem perspective.

* As they are strongly linked references to “XML” will refer to both XML and Web Services in this document.

Adoption and Benefits in Financial Services

Within the financial services industry there are additional drivers for XML adoption and the use of XML has been steadily growing over the last three to four years. Over the next two years, the rate of growth is likely to accelerate as XML becomes increasingly pervasive and in some cases mandatory across a wide-range of business and regulatory functions.

The spread of XML can be seen particularly clearly in financial messaging as standards bodies and industry groups have committed to XML based standards. For instance, in November 2001 the International Swaps and Derivatives Association (ISDA), the global trade association for privately negotiated derivatives such as currency, commodity, and equity swaps, adopted FpML as its XML standard. In 2002, major organizations such as SWIFT and FIX started to move to XML-based messaging systems. With the release of FIX4.4 at the end of 2003, FIX now releases both its non-XML and XML based message standards close to simultaneously. Also at the end of 2003, SWIFT piloted its first XML based business solution, for real-time cash management, which goes live in 2004, and will start pilots of its investment funds messages later this year.

On the reporting side, the XML business reporting language (XBRL) will become a supported submission format for a wide range of government agencies worldwide including the UK Inland Revenue, Edgar Online, the US Federal Deposit Insurance Corporation (FDIC), and The Tokyo Stock Exchange. At the start of 2004, the Financial Services Authority (FSA) in the UK has announced that it will be mandating the use of XBRL for electronic regulatory reporting from 2005.

XML is also becoming important for a number of specific business areas such as financial research. The reason in this case is clear: reduced cost and greater efficiency. An Accenture report quoted in STP Magazine in 2003 estimated that a bank with a five per cent market share could save as much as \$35m, gain \$60m in productivity and accrue between \$200m and \$400m in revenue benefits by modernizing its research systems with XML and the research specific standard RIXML. The same report predicted a saving of between \$50m and \$100m for banks and brokerages using XML-based standard in research.

All of this led to 81% of leading financial services organizations seeing XML as absolutely critical to their overall business strategy, as found by a *Wall Street and Technology* survey published in May 2002. This is further highlighted by

the number of organizations adopting their own XML standard for communication both internal and between partners and customers.

The Technology Reality Gap

In sharp contrast to the high expectations associated with the move to XML and Web Services is the reality of the pain of actually developing solutions. Adopters found that there were two main options, each with significant drawbacks:

- Build your own XML processing frameworks rather than focusing on developing business logic. These frameworks require a significant level of investment and rely on a high level of scarce technical expertise. This means that the resulting solutions are often expensive to develop and maintain. In addition, many organizations have found that the frameworks suffer from poor performance and high latency, which make them unsuitable for business critical deployment in financial services. Finally, as code-based frameworks, they are often hard to extend in line with changing business requirements.
- Implement Enterprise Application Integration (EAI) products, which typically require propriety skill sets that scarce and do not deliver a flexible solution. In many cases these products are not XML centric, pre-dating the widespread use of the standard. This further reduces their power and flexibility within an XML-centric project. Reflecting their original market, these products are not suitable for departmental or even divisional projects with tight delivery timescales and the cost of ownership makes a Return on Investment hard to justify. All of this is reflected in the Gartner statistic that 65% of EAI projects over-run, go over budget or otherwise fail.

The PolarLake Solution

PolarLake bridges this reality gap by providing an enterprise grade XML and Web Services platform. It dramatically reduces the cost and complexity associated with integrating existing systems - delivering completely new capabilities based on those systems and evolving the delivered systems, as business needs change. Systems created with PolarLake are fundamentally more flexible and can be easily adapted to address new business problems.

Recognizing the value in existing IT investments, PolarLake can leverage existing EJB™, Java, COM and relational database systems without code changes. It can also integrate into other enterprise assets, including data feeds, messaging systems such as IBM's WebSphere® and TIBCO's ActiveEnterprise™, and management systems such as HP OpenView and BMC Patrol®. Finally, it can integrate with common enterprise applications from vendors such as SAP and PeopleSoft as well mainframe systems based on CICS and IMS.

The CIO's Dilemma

IT managers and CIOs in the Financial Services industry and elsewhere are faced with a dilemma: the business problems they are trying to solve are changing fast. Some of these changes are driven by the business itself, and others are outside of its control, as with current examples such as the Basel II capital accord and the Sarbanes-Oxley reporting requirements. However, all of these changes require a much higher degree of integration between existing systems: the ability to access the systems in real time and the ability to evolve their capabilities as fast as the business requires.

Over the last few years, the commoditization and sharply reduced margins in trading vanilla financial instruments has put greater emphasis on innovation and flexibility in bringing new products and services to market. Often this means creating variations of existing products, and giving customers direct access to data and capabilities that previously were in-house only. More recently, the rise in popularity of multi-asset class financial products imposes a much more complex problem: integration across the departments responsible for each class of asset within the product; departments that are often independent with their own applications and integration architectures.

There is also an on-going drive towards greater outsourcing and greater sophistication of services. This also places a heavy burden on both the providers and consumers of such services as paper processes are moved into real-time and manual processes are automated. For instance, the services offered by Global Custodians and Transfer Agencies to fund managers are evolving towards real-time information flows and this move may be greatly accelerated by the new SWIFT XML based Investment Funds messages.

All of these innovations fundamentally depend on IT systems and their ability to rapidly evolve and extend in step with business needs.

The PolarLake Solution

PolarLake is designed for use in precisely the business critical environments encountered by CIOs and IT managers in financial services. PolarLake's products match their enterprise requirements, leveraging the company's unique technology:

- **Technology integration** with PolarLake's ***Dynamic XML Runtime™*** provides a highly scalable, high performance runtime server that integrates with enterprise infrastructure such as queuing systems, management infrastructure and legacy applications.
- **Business integration** with PolarLake's data-centric ***XML Circuits™*** approach allows existing developers and business users to rapidly deliver new solutions with minimum disruption to existing systems and maximum leveraging of existing assets and skills.
- **Leverage existing skills** with tools that provide an intuitive XML-centric environment that supports the software life cycle. These integrate with and complement familiar tools environments such as Sun™ ONE Studio, Borland® JBuilder™ for Java™ applications or BMC Patrol® Enterprise Manager Connect SNMP management system.

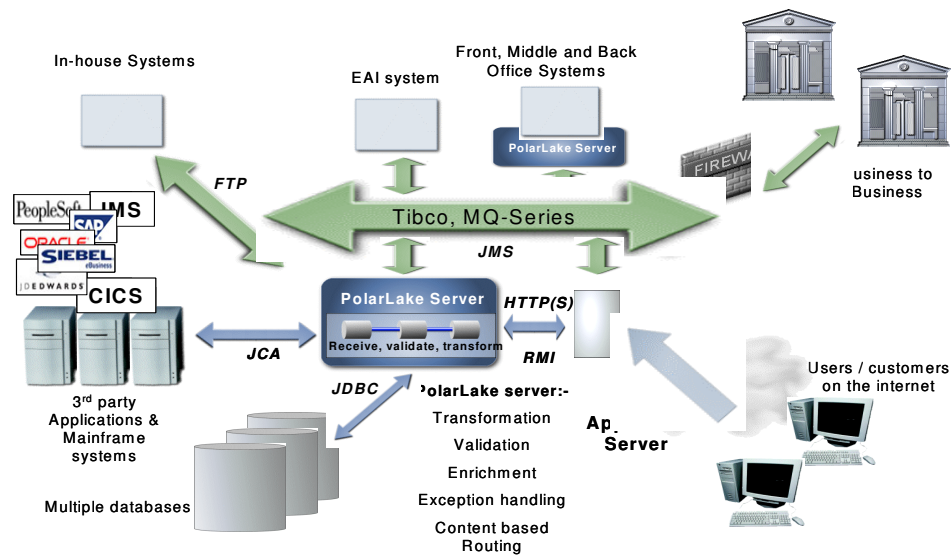


Figure 1 PolarLake integrates with a wide range of technologies, addressing key processing tasks

The PolarLake Product Set

Built on this common technology base, the products that make up the PolarLake Integration Suite can be used separately to address point problems or combined to form a complete platform for enterprise-wide adoption of XML and Web Services.

- **PolarLake JIntegrator** provides a complete platform for creating new, or extending existing, Java and J2EE-based XML and Web Service systems, while meeting the most demanding performance and reliability requirements. The PolarLake JIntegrator can be used as an Enterprise Service Bus (ESB), with support for content-based routing, publish-and-subscribe, and Web Services. Furthermore, the PolarLake JIntegrator is the first **Universal ESB**, with support for the leading messaging products, unlike other ESB products that are limited to supporting the vendors messaging product only. The product can be used in a wide range of solutions from XML-based integration of individual applications to enterprise-wide integration architectures.
- **PolarLake Messaging Integrator** provides complete support for XML-based integration, including routing, validation and processing, across Message Oriented Middleware (MOM) and other

protocols such as HTTP(S), SMTP and FTP. The Messaging Integrator also enables XML and Web Services based integration of existing database-centric applications: providing a zero code/zero disruption approach to the integration of XML and native relational data, as well as other database assets such as stored procedures. It features the PolarLake Mapper, a sophisticated high performance XML transformation engine with a graphical interface, allowing the most complex transformations to be easily defined. Other features include support for mapping between non-XML and XML formats, and business activity monitoring.

- The **PolarLake Adapters** provide rich interaction between PolarLake solutions and the most common enterprise applications such as SAP® R/3®, Siebel, PeopleSoft®, CICS®, IMS, J.D. Edwards™, Oracle® and more. Based on the Java Connector Architecture (JCA), the PolarLake Adapters expose the standard and custom business processes implemented in the application as XML or Web Services.

XML & the Financial Services Industry Today

The Extensible Markup Language (XML)

XML is a markup language that is similar to HTML, the language used in Web browsers, and is becoming commonly used to facilitate interchange and processing of data between computer systems. Gartner has predicted that by 2005, XML, standardized by the World Wide Web Consortium (W3C) in 1998, will have emerged as the common language of electronic commerce.

Part of the power of XML is that it is readily understandable by people as well as machines. The structure of an XML representation of information is hierarchical, consisting of a series of elements, each defined by a start tag and a matching end tag. For example, a famous address can be described in XML by:

```
<Address>
  <Number>1600</Number>
  <Street>Pennsylvania Avenue NW</Street>
  <City>Washington, DC</City>
</Address>
```

Element tags are enclosed in angle brackets ('<', '>') and each begin tag must have a matching end tag. The term 'document' is

XML-related standards are being progressed by a number of organizations. These standards don't provide a complete architecture for encoding financial information, but rather focus on specific sub-sections of the Financial Services industry or particular vertical niches. Each of the organizations is creating frameworks for defining XML documents in its area of focus, and in some cases rules for how the documents should be exchanged. The frameworks are sometimes described as architectures or dictionaries. These definitions are normally in the form of XML Schemas, which describe the structure and sequence of each conversation. In most cases, the definitions are not Web Services-based, reflecting the messaging-oriented nature of much financial services interaction.

While some convergence between the standards is likely, it is probable that any finance organization will need to support multiple standards within its systems as well as others required for B2B and other connections to customers and clients. It is clear that any viable architecture must be flexible enough to manage the evolution of these standards. Gartner summed this up well when it predicted in 2002 that 'Through year-end 2005, the majority of large financial services providers will support multiple XML-based data model standards to connect with their external stakeholders.'

This requirement to coexist with multiple standards must be achieved without significant performance, latency or maintenance overhead, all of which would undermine the typical project goal of real-time response. PolarLake recognizes this and its products support all the emerging schema standards, so systems developed with PolarLake can be easily modified to handle a new XML document type or XML schema standard with minimal cost and disruption to the existing solution.

Web Services and XML

At its most basic level, a Web Service is a component of business processing that may involve one or more business systems. The initial Web Services architecture was B2B-focused and this is still reflected in the standards that many vendors' are offering. Web Services provides an XML-based model for interface-driven system interactions, such as previously available in CORBA, or the EJB model using RMI.

There is a second type of interaction model that is common in financial services, which is *data or message oriented*. This model is typified by data dictionaries defined and used in systems such as IBM's WebSphere®MQ-series. This style of interaction is much more common in enterprise systems for reasons of scalability and match to the underlying business process.

At the moment, the majority of Financial Services industry organizations mentioned above are not using Web Services as the basis of their frameworks. This will change over time as Web Service standards evolve and mature. In the medium term it is likely that most will add some Web Service support, but most will retain the message-oriented mode of interaction as their key focus.

To allow each component in Web Services-based architectures to be integrated in a useful way, two additional elements are needed:

- A standard way of sending and receiving the information required to complete the business processing (XML, with an additional standardized envelope called SOAP, which is also part of the ebXML standard)
- A standard way of publishing descriptions of the conversations that Web Services can be involved in. The leading standards are UDDI (which can provide WSDL as a way of describing actual XML document exchanges within the conversation) although many implementations use more informal mechanisms such as databases or browser based systems.

As these base standards have matured, many more specialized standards have been proposed to handle everything from *security* to *event notification* to *identity management*. Of these more recent standards, most are still not mature enough or widely accepted enough to be seriously considered. One exception is the Business Process Execution Language for Web Service (BPEL4WS), which provides a simple standard for calling sequences of Web Services (known as orchestration).

Web Services provide a layer of abstraction on top of the implementation of a business component - the service definition. This definition is not dependent on any specific infrastructure or middleware. Hence the adoption of Web Services will remove one of the obstacles to the integration by providing a common language to describe the interactions. All of PolarLake's products natively handle both Web Services as well as any other XML document exchange.

The Enterprise Service Bus

Leveraging the XML and Web Services standards and Java, the Enterprise Service Bus (ESB) is a new category of product, first defined by Gartner as "a new architecture that exploits Web Services, messaging middleware, intelligent routing, and transformation. ESB's act as a lightweight, ubiquitous integration backbone through which software services and application components flow."

The key features that are required in a product supporting the ESB architecture are:

- Native Web Services support
- Support for both Synchronous and Asynchronous messaging for inter-application communication
- Message transformation
- Support for the publish / subscribe and content-based routing model of distributing messages
- Support for J2EE-based application servers

An ESB provides the basic functionality required for many integration projects. PolarLake's JIntegrator product provides the complete set of functionality required by an ESB architecture. The additional components with the PolarLake Integration suite extend this to form a complete integration platform.

Service Oriented Architectures

The concept of a Service Oriented Architecture (SOA) is not a new concept; it is also not tied to a specific standard or set of standards. It can be implemented with Web Services or in the context of an Enterprise Service Bus. Moreover, these approaches are generally seen as the most appropriate for a SOA and the most likely to deliver the promised benefits of increased flexibility and reuse and hence reduced cost and risk associated with change.

At the core of the SOA concept is the idea of defining services that any application allows other applications to access. For instance, a settlement system could offer a settlement service for certain classes of financial product. Any other application can access that service without knowing anything else about the service provider except what information to send to it and what to expect back.

This *decoupling* delivers the key benefits of SOA: reuse and flexibility. Reuse is enhanced because new business processes can be created by accessing existing services and only implement new services when an existing service does not match the requirement. Flexibility is enhanced because the service provider or consumer can be modified without impact on the other party (so long as the service definition is not changed). The combination reduces the cost and risk profile: fewer changes are made to existing systems and fewer new systems are required.

Some Business Uses for XML & Web Services

XML is playing a larger and larger role in a number of Financial Services industry initiatives. Web Services is starting to contribute to these initiatives, providing the interface mode of interaction. Examples of problem areas that particularly benefit from XML or Web Services are discussed below. It should be stressed that these benefits are not automatic and require a solution capable of being addressed by a number of technology challenges outlined later in this paper.

Straight Through Processing (STP)

STP requires both multi-system and multi-enterprise integration, coupled with real-time decision-making. A significant problem within any STP environment is the management and synchronization of multiple data sources representing the same underlying data. This problem becomes significantly more complex as multi-asset class financial products become more common and more sophisticated.

A benefit of using XML is that it provides a single data format, which reduces the risk of data error in translation between formats and reduces the processing time. Exception handling and resolution is a key component of STP. As a self-describing and *somewhat* human-readable format, XML can assist in both areas. The first attribute can assist in the identification of problems; the second in easing the task of human intervention in resolution.

A second benefit of XML, as an infrastructure neutral format, is that it can reduce the cost of implementation by potentially freeing up each department or organization to make their own technology choices with a lowered cost of integration across disparate technologies.

Real-Time Risk Management

Real-time risk management requires similar levels of integration as STP with an additional requirement for real-time integration across a wider range of systems and support for derived decision-making. Complex transactions, with disastrous negative consequences if mishandled, must be managed for risk exposure. As many enterprises are attempting to reduce costs and risk through automation of financial business operations, the ability to provide real-time statements of cash and exposure is becoming a key requirement on financial organizations.

The Basel II capital accord makes this even more important. Under this accord, a bank's ability to meet well-defined standards of risk management directly impacts on its capital requirements, and hence its profitability. The potential value of this is reflected in the level of investment in this area over the next two years, which IDC expects to exceed \$4b by European banks alone.

As with STP, XML can provide the framework for organizations within which to integrate systems and support their customers' business needs. Specifically, XML provides an application and infrastructure neutral format that can be used to gather information across a wide-range of systems in real-time. Because it is a neutral format, it can be introduced at a much lower level of disruption than any other approach would allow.

Client connectivity and B2B Portals

This requires integration of each enterprise's systems with its suppliers and customers to support the delivery of new products and services over the Internet. Suppliers in financial services provide a wide-range of services across the whole spectrum of the organization's business. For instance in fund management, as has already been mentioned, these can range from fund administration to transfer agency. A more general example is in the area of real-time cash management which can deliver significant cost savings to international banks through more efficient control of the cash balances held in other banks.

As these services move from paper to batch to real-time, there are additional problems that need to be addressed. In many cases, the organization must comply with a message format defined by the other party. This has the potential to add significant cost to the implementation project and can undermine the business case.

XML provides the neutral format for both the internal systems and the external systems. Passing XML documents around the network significantly reduces integration and translating effort within each system. By taking an incremental approach, such as that provided by PolarLake, the initial investment can be controlled to a level in line with the business case, and evolved as business drivers emerge.

Business Process Integration

This is a generic problem, sometimes overlapping with the areas described above. It requires integration of each enterprise's systems, predominantly within the enterprise. Examples range widely between organizations and departments. A sell-side organization may wish to automate its research authoring and management processes to improve efficiency and reduce cost. A

buy-side organization may wish to streamline reporting processes. As with e-Business integration, XML provides a neutral format for the participating systems. In addition, it can act as a useful abstraction of the underlying business process being implemented.

All of the above

For many organizations, these requirements overlap and will actually involve a number of the categories covered above. STP needs to feed into real-time risk management; E-business integration with clients may require SWIFT messaging and will certainly access the same internal systems.

In order to handle these overlapping requirements, any solution must be able to handle a wide range of message types, interact with multiple software infrastructures, and be capable of handling different qualities of service and security requirements. The solution must also be able to handle a high-rate of change as requirements change, as new service providers are engaged, as new clients are signed, or as the standard message formats evolve.

PolarLake's solution is uniquely capable of handling each of these business areas and evolving to solve multiple changing requirements, in a cost effective and low risk manner.

Key Issues and Problems Encountered

Even though XML and Web Services are used in a very diverse set of projects there are a number of common issues that are emerging. If these issues are not addressed, they become serious roadblocks on the way to successfully using XML and Web Services. They are:

- The system does not match the performance and scalability capabilities of existing systems. Using XML should not be an excuse for poor performance and scalability.
- The integration is not achieved without changing the application and without breaking the functional capabilities and integrity of the application. To fix this, the XML arriving must be validated to ensure integrity and also transformed to match the existing interfaces of the application.
- The XML-based integrations are not capable of handling exceptions in an intelligent manner. The exceptions are not only technical problems, but also the more common business related problems, such as inappropriate trades or unknown transaction

types. Exception handling includes not only capturing the exception but also analyzing and resolving it in an as automated manner as possible. Without effective exception handling, complex integration projects, such as STP, will be much less successful or even fail.

- The new infrastructure does not easily fit within the enterprise environment, handling integration with common application architectures (such as databases and J2EE) as well as providing expected message routing capabilities.

Resolving Key Issues with PolarLake

Performance and Scalability

PolarLake overcomes the performance issues often associated with processing XML by employing a number of innovative technologies, typically increasing throughput by 30-50 times compared with other servers. Some of the key factors that contribute to this performance boost are:

- XML-streaming: XML documents are processed as each element arrives, thereby ensuring very low latency. PolarLake is equally efficient at handling large documents as small ones.
- Multi-threading: multiple sections of the same document (and multiple documents of multiple types) are processed in parallel.
- Single scan: PolarLake ensures a document is scanned once only within each PolarLake server.
- Selective processing: frequently an application only requires a subset of an XML document for its purposes. Within the XML Circuit™, PolarLake isolates and processes just the subset – often with dramatic performance results.
- Distributed and service orientated: PolarLake can be used to expose applications and databases as services and distribute those services across an enterprise where necessary to eradicate bottlenecks and improve scalability.

The PolarLake architecture requires less hardware to meet the scalability requirements at each node than alternative approaches that rely on XSLT scripts or EJB-based approaches. In addition, PolarLake supports clustered deployment, leveraging capabilities provided by hardware, operating system and other infrastructure components (e.g. IP clustering and operating system-provided clustering).

Semantic Validation, Transformation & Enrichment

It is easy to create XML that is technically correct and will pass any schema validation but is nonetheless nonsense from an application perspective. Similarly, transformation and enrichment need to go beyond simply recombining or augmenting the XML document. In each case, the business meaning and context of the XML document needs to be the basis for validation, transformation or enrichment. This reflects differences in the information model between the incoming document and the outgoing document.

- *Semantic Validation* ensures the XML content makes sense, both at the level of individual fields and between fields (simple example: "start date" is earlier than "end date").
- *Semantic Transformation* ensures the data conforms to internal formats and therefore can be used by internal systems (this is likely to mean transformation into *non-XML* formats as well as between XML formats).
- *Semantic Enrichment* ensures the data is complete and is likely to involve other applications (e.g. databases, Web Services, etc.) as well as internal calculations (e.g. aggregation, derivation and

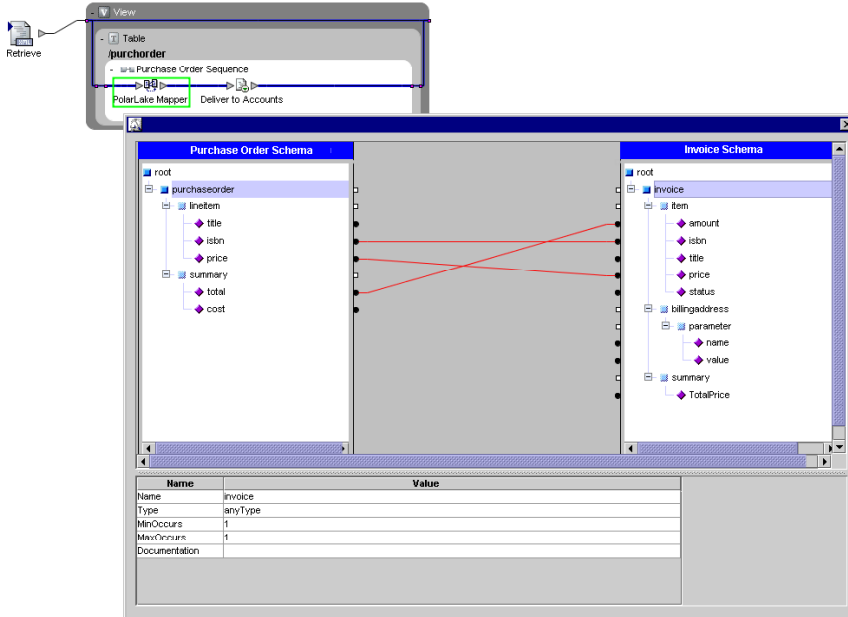


Figure 2 Defining a transformation using the PolarLake Mapper.

All of these are accomplished within the XML Circuit, which defines the way a document is processed based on XPath selection rules and sequences of components that act on the selected fragments of XML. XSLT or BeanShell may be used for simple transformations when performance is not an issue. For more complex transformations, the high performance PolarLake Mapper engine supports simple, multi-element, contextual and functional transformations, and includes out-of-the-box functions to handle common problems such as date format conversions.

Intelligent Exception Handling

Exception identification and resolution is key to most financial processing, and is central to STP in particular. The Hurwitz Group has estimated that nearly 80% of time spent in building business processes is spent in exception management. All of PolarLake's products provide a complete and flexible exception handling architecture within the definition of the XML Circuit™ that spans both system and application level exceptions. An exception can be raised at any point in the circuit and passed to an exception-handling segment of the circuit that has access to the full power of the PolarLake platform to carry out the required recovery or alerting procedures.

Business Activity Modelling

PolarLake includes a pre-defined set of system level statistics that can be remotely monitored from the PolarLake Management Console, any SNMP compatible management systems or a custom-built management console.

It is also possible to define application level statistics, which can collect information about any aspect of the XML documents flowing through the system or other system behaviours. Examples of how this can be used include capturing the number of documents of specific types flowing through a PolarLake Messaging Integrator node or the total value of any document of a particular type, such as a purchase order.

Intelligent Routing

The PolarLake architecture supports a number of document routing models including:

- **Simple Message Routing**, based on the type of document, on the origination queue or other defined parameters such as application defined flags.
- **Content-Based Routing**, based on the XML content of the document, with routing decisions made with XPath rules.
- **Publish & Subscribe Based Routing**, which allows a list of destinations (the subscribers) interested in receiving documents from a sender (the publisher) based on selection criteria (the topic). If a JMS-based queuing system is used, PolarLake can integrate with the defined topic mechanism.

Integration with Key Technologies and Message Formats

J2EE™ Integration

PolarLake is a 100% Java-based system that can be deployed within a J2EE application server. PolarLake allows Java and J2EE developers to easily XML-enable and Web Service-enable existing deployed applications without sacrificing performance, reliability or scalability. Furthermore, a PolarLake enabled J2EE application can be easily extended to process many different types of XML document and participate in complex XML-based interactions or workflows, without changing or disrupting the deployed application.

In addition to being capable of deployment within a J2EE server, PolarLake can also be deployed as a standalone server running within its own Java Virtual Machine. It can also be deployed within a servlet engine, such as the popular open source Tomcat (PolarLake v2.0 includes the Tomcat servlet engine), or within a web server.

Database Integration

Database centric applications are central to most enterprises' IT infrastructure, and integrating with them is key to any effective use of XML. PolarLake provides a complete integration of XML with the leading relational databases by:

- Automating the storing of XML documents in a database *and* the mapping of XML directly into and out of native types in existing rows and columns. This mapping capability uses the PolarLake Mapper Engine, enabling the most complex transformation to be completed within the graphical configuration environment.
- Integrating with the *business logic* held in databases as stored procedures and providing these capabilities within a complete XML-centric environment.
- Providing the ability to detect changes in the data stored in the database and generate XML based on those changes.

Significantly, PolarLake achieves all of this without changes or disruption to the deployed database, and without compromising the database's performance, reliability or scalability.

Messaging Oriented Middleware Integration

Asynchronous messaging oriented middleware (MOM) products deliver better scalability and adaptability than alternative approaches. Enterprises already rely on (often several) messaging technologies to reliably route mission critical information between systems.

PolarLake's products are integrated with the leading MOM products from IBM and Tibco and are compatible with the transactions, clustering, and publish/subscribe capabilities provided by these products. Because PolarLake supports all of these as well as other protocols such as HTTP(S) and SMTP, it is ideally suited to the role of bridging between multiple domains based on different protocols and messaging systems.

Integration with Packaged and Mainframe based Applications

The PolarLake Adapters allow deployed packaged applications from leading vendors such as Oracle®, PeopleSoft® and Siebel and mainframe applications based on IMS and CICS, to be seamlessly integrated within PolarLake solutions. Based on the Java Connector Architecture (JCA) standard, the PolarLake Adapters expose the standard and custom business processes within these applications as Web Services or as XML messages. The PolarLake Adapters allow the full function of the applications to be exposed including support for bi-direction synchronous and asynchronous communication.

Integration with Non-XML formats

In most integration problems, there will be a mix of XML and non-XML based formats. In financial services in particular, the migration to XML based message formats by organizations such as SWIFT or FIX will take a number of years. Within organizations, pre-XML formats may continue to be used for

some tasks indefinitely period. This means that there will continue to be a mix of XML and legacy formats flowing through the network.

While the PolarLake architecture is XML-centric, it is straightforward to integrate a wide range of non-XML formats. The exact approach taken will reflect the requirements of the organizations and their timescales for XML migration.

At a technology level, PolarLake can easily forward non-XML messages to another destination, such as a FIX engine, or convert them into XML. In the former case, many of the routing and processing capabilities are available, although with restrictions that are inherent in the message formats. In the later case, once processing is complete, PolarLake can output documents into the required non-XML format.

The most common non-XML formats are supported as standard capabilities of PolarLake. These include comma-separated and tab-separated files and Microsoft Word and Excel formats. To handle the automatic translation of any unsupported formats, PolarLake can be extended with easy to build Java-based components called Request Processors. These components are customized to handle the formats required and then deployed into the PolarLake server to automatically convert to and from the non-XML formats when necessary.

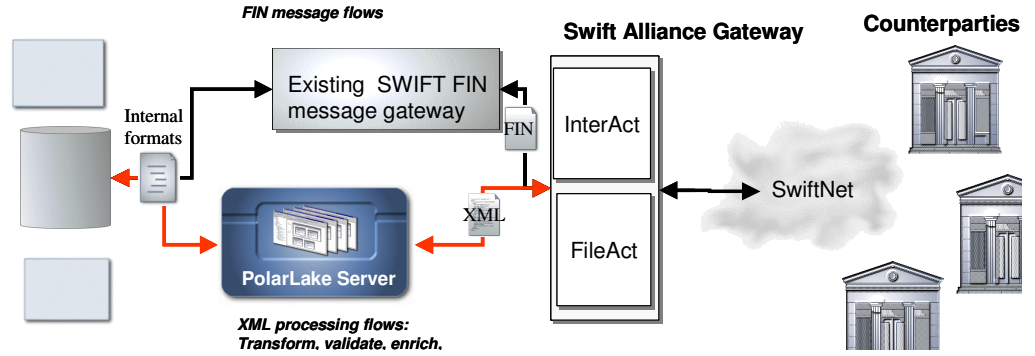
An example: Migration to SWIFT Investment Fund Messages

In 2004, SWIFT is launching a new business solution for investment funds based on XML messages. Some organizations may already be using some SWIFT FIN messages for this purpose and will need to migrate away from these messages before May 2005. Many organizations also use SWIFT for other purposes and will continue to do so after the migration of the investment funds messages.

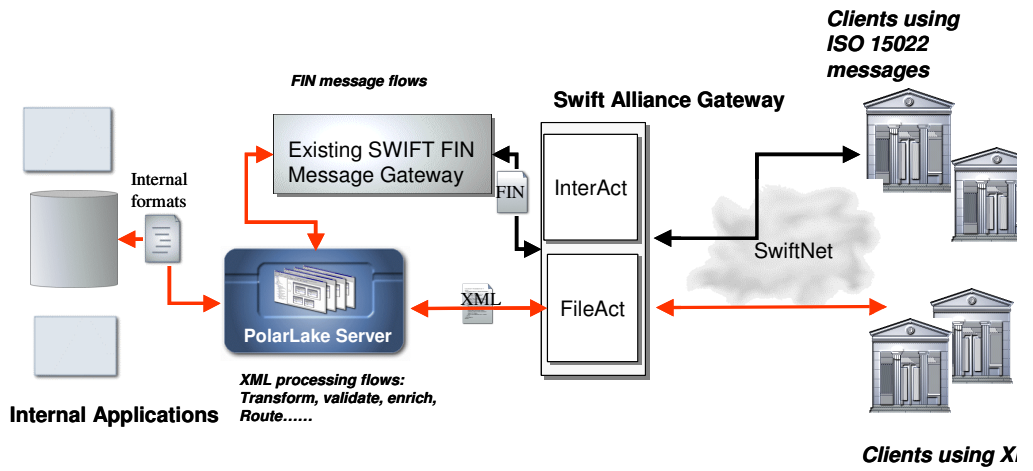
The rollout of the messages and eventual migration can be done in a number of different ways. The exact manner will depend on each organization's needs and timescales. The easiest case is for an organization that does not have an existing SWIFT installation, in which case PolarLake can provide the complete XML-only solution. In the more common case, when some SWIFT messages are already in use, two possible migration strategies are shown below. In both cases, the existing investment in SWIFT gateway is preserved and PolarLake is used as a migration path towards the XML based messages.

In the first example, the internal application will make the routing decision to either the FIN based gateway or PolarLake on the basis of the counterparty and message format. In the second example, PolarLake acts as a router, routing

the FIN messages to the existing gateway, while handling the XML messages itself.



A migration architecture showing FIN based SWIFT gateway working in parallel with the PolarLake server processing XML based messages

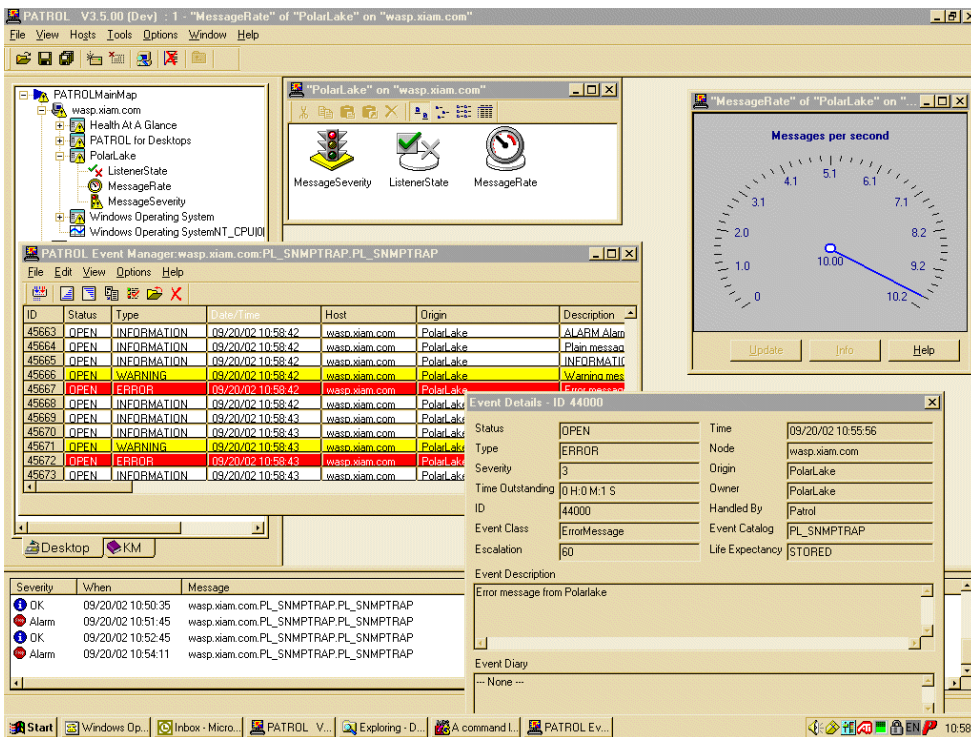


A migration architecture showing PolarLake routing to either the FIN based SWIFT gateway to the SWIFT Alliance Gateway.

In both cases, PolarLake provides the flexibility for the organization to pick both the approach and the speed with which the migration is carried out.

Deploying and Managing PolarLake Applications

The PolarLake platform can be deployed in a number of configuration types, including as a stand-alone server, or within a J2EE application server. Whichever configuration type is chosen, there are two management options: using the PolarLake Management Console (PMC), or using a SNMP compatible system such as HP OpenView or BMC Patrol® (shown below). In the latter case, all the usual features of those systems can be used with PolarLake deployments.



The BMC Patrol® management console showing a PolarLake deployment under management, including alarms and load monitors.

Using the PMC, developers can start and stop applications, view system and error messages and launch the PolarLake monitor utility to examine application behaviour in greater detail, either locally or remotely. The PMC also supports hot swap application and configuration updates, allowing the deployed XML circuits or any server configurations to be modified or completely changed without system downtime.

Conclusions

In the Financial Services industry, XML adoption is underway. The combination of regulatory requirements and the business need for better integration, both inside and across organisations, will see XML become pervasive over the next two to three years. A number of finance-specific standards are emerging to deal with different aspects of the industry's requirements. Web Services, with its promise of greatly increased programming efficiency, is also likely to be used in the financial services industry. Over time all of these standards will evolve, and will need to co-exist with other standards and company-specific XML standards.

In order to deploy an XML-based integration system successfully, organizations will need a platform that is productive, and flexible. PolarLake delivers a set of products based on an enterprise grade XML platform, incorporating a full set of development and deployment tools as well as Web Services support, that is both productive and has a significantly reduced learning curve for developers. This, coupled with its Dynamic XML Runtime™ technology and its XML-centric component assembly framework, provides PolarLake developers with the flexibility to meet business requirements today and to evolve and extend solutions over time.



PolarLake™, a leader in standards-based incremental integration, provides a complete suite of products for implementing XML and Web Services-based solutions, including those based on the Enterprise Service Bus architecture. PolarLake's products deliver rapid Return on Investment by focusing on solving high value business problems with a standards-based approach capable of evolving and expanding to address the longer term objectives of the organization.

PolarLake has a proven track record in delivering the benefits of incremental integration with a technology that leverages existing IT investments in standards, skills and systems to reduce both initial investment and total cost of ownership. Deployed customers include leading corporations in financial services such as Pioneer Investments* (Ireland), Man Financial Ltd (UK), Nissay Dowa (Japan) and in telecommunications such as Midwest Wireless (USA).

PolarLake' solutions are provided by partners such as Hitachi Systems and Services and Sun Microsystems. PolarLake is private company, headquartered in Dublin, Ireland, with offices in London, New York and Tokyo.

Technology integration with PolarLake's *Dynamic XML Runtime™* which provides a highly scalable, high performance runtime server that integrates with enterprise infrastructure such as queuing systems, management infrastructure and legacy applications.

Business integration with PolarLake's data-centric *XML Circuit™* approach which allows existing developers and business users to rapidly deliver new solutions with minimum disruption of existing systems and maximum leveraging of existing assets and skills.

Leveraging of existing skills with tools that provide an intuitive XML based environment, support the complete software life cycle and are entirely open standards-based.

CONTACT DETAILS

PolarLake Ireland (HQ)
Block F1
East Point Business Park
Dublin 3
Ireland

PolarLake Japan
13 F Ebisu Business Tower
1-19-19 Ebisu
Shibuya-ku
Tokyo, Japan

PolarLake USA
1001 Avenue of the Americas,
Suite 1121
New York, NY 10018
USA

PolarLake UK
No. 78 Cannon Street
London
EC4N 6NQ
UK

T: +353 (1) 449-1010

T: +81-3-4360-3965

T: +1 (212) 813 2965

T: +44 (0) 20 7618-6426

F: +353 (1) 449-1011

F: +81-90-1421-6486

F: +1 (212) 790 9072

F: +44 (0) 20 7618-8001

E: info@polarlake.com

E: japan@polarlake.com

E: usa@polarlake.com

E: uk@polarlake.com

www.polarlake.com

