

PolarLake

Abstract: Few would deny that XML has made a dramatic impact on the way we develop software for both enterprise and domestic users. The tools used by developers are for the most part standard programming languages and web servers. PolarLake have looked at the issues faced by companies building complex XML-based systems and built a flexible and scalable platform for application development. PolarLake provides development productivity, performance, re-use, and ease of maintenance.

By Jonathan Stephenson

PolarLake provides a graphical design tool where you literally draw the system, chaining together processors and rules to build up a process pipeline.

Introduction

PolarLake is a Dublin based company that develops on the Java platform and have used their experience in the finance sector and CORBA middleware to develop a powerful platform for XML document based commerce. The key benefits are:

- Increased development productivity
- Built-in transport layer supporting synchronous and asynchronous messages
- Optimal XML processing performance
- Flexibility of the solution
- Ease of maintenance

As the de facto standard for business-to-business information exchange, XML document processing will be involved in most projects undertaken today. In particular, PolarLake understands the performance issues associated with large XML documents and also the routing and transport issues. Synchronous and asynchronous

messaging is supported using either PolarLake's own transport protocols or via an industry standard messaging middleware.

By modeling an application as a number of discrete steps that are chained together using rules stored in an XML configuration document rather than being hard-coded in the Java processors, a system can be assembled, reconfigured and its components re-used to benefit the business. PolarLake provides a graphical design tool where you literally draw the system, chaining together processors and rules to build up a process pipeline. Java components that you develop and PolarLake system components are slotted together so that decision trees are constructed based on the content of the XML documents received. This not only provides a graphical overview of the system but also adds flexibility at the business and technical level. Deployment, debugging and system management tools complete the development toolkit.

Productivity Gains through Orchestration

In the current climate of carefully controlled IT spending, PolarLake justifies its price tag on three main counts: performance, productivity and ease of maintenance. The development environment is particularly easy to use. Circuits are built up and configured using a highly graphical drag and drop workspace. A circuit comprises a set of event processors, each processor responds to incoming SAX events, processing XML and feeds the next processor. In the graphical workspace, circuits are assembled into the processing pipeline by specifying XPATH queries, which govern the conditional routing of documents through the paths of the circuit, and by configuring the individual processors through their property sheets.

The result is a highly graphical picture showing how the application is structured. New applications can be rapidly assembled from existing circuits or by extending existing applications by adding new XPATH conditions and new processors.

Application event processors can be written using any of the following:

- Java (created using plug-ins for familiar IDEs). Classes that conform to SAX or JDOM are supported.
- Script based (using XSLT or BeanShell).
- PolarLake applications can also bridge to outside Java, EJB or COM systems.
- With the Database integrator, they can also be mapping components into database tables, or calls to stored procedures.

Once you have constructed each circuit, you can test it using PolarLake's graphical monitor/debugger. This allows you to create test documents and trace them as they flow through the circuit. Break points are supported and audit trails can be saved for the purposes of regression testing.

Since PolarLake hides all of the infrastructure and middleware programming, developers need only focus on writing the application's business logic within event processors.

XML Processors

PolarLake provides SAX and DOMs processors for document manipulation. PolarLake has addressed the performance issues associated with parsing large XML documents by implementing an event-driven system based on the SAX specification. The architecture is streams based, this means that a step in the pipeline does not have to complete before the next step sees the document. The PolarLake SAX processor is compatible with the standard Java SAX class definition. The documents are streamed through the parser, triggering events in your handler as the nodes in the document are located. This is in complete contrast to the DOM (document object model) approach that loads the whole document into an object representation and then navigates the object tree in order to extract information. Any system that is based solely on the DOM model will be slow and memory inefficient because the DOM tree has to be completely built and torn down at each step of the process. PolarLake supports both SAX and DOM approaches and there are cases where you may choose to use the SAX model to iterate through a list of transactions and

then for each one, load a small sub-tree into the DOM parser to process the transaction. Developers can choose their own document processing tools as long as the interfaces conform to the expected interface types. Above all the PolarLake approach avoids the pitfalls associated with XSLT based transformation which is too slow for large documents.

Reusable Components

Having established the platform in the first release, work with customers has driven the production of ready-built modules from which systems can be constructed without cutting any Java. The XSLT, BeanShell and SOAP components are part of the set shipped with the base product; the database integrator and message integrator are licensed separately.

SOAP module

This handles Web Service requests (both incoming and outgoing) in a transparent manner.

BeanShell

The BeanShell scripting component is used for simple string manipulation for example, stripping and adding a new envelope to an incoming XML document.

XSLT Transform

The transformation component provides the circuits with the ability to apply an XSLT document to an XML document.

Logger

Logs messages.

Validator

Compares XML input against a schema.

Database Integrator

This allows the non-programmer to take an XML document and map nodes to

columns in a relational database and to insert/update/delete records. Stored procedures are also supported. Enterprises are generally happier with mission critical information, such as an equity trade request, sitting in their database than in a queue of documents. Once in the database traditional RDBMS skills can be used to finish the integration with the back-office systems.

Messenger Integrator

Despite the high media profile for Web Services over HTTP, most mission critical XML projects in the world of financial services are being done well behind the firewall using tried and tested transports like MQ Series. The message integrator component provides interfaces for transports based on JMS (MQ-Series, Tibco Enterprise, SpiritSoft SpiritWave, and Sonic MQ), plus HTTP(S), SMTP, and file system. MS-MQ and Tibco Rendezvous support are available as add-ons to the PolarLake Messaging Integrator.

Message Integrator allows you to define transaction boundaries when processing XML from JMS queues. Messages are deleted from the queue only when the processor commits the transaction.

You construct systems by chaining together components and setting the parameters in the property sheets; the circuits can include components you build yourself using Java and one of the supported IDEs (Sun ONE, Borland or Eclipse).

DIY Java Components

Building your own components for PolarLake requires knowledge of Java. Integration with JBuilder, Sun ONE, and in the future IBMs Eclipse development environments, means that the basic outline of the code is done for you. The

component uses the SAX or JDOM interfaces and the class generated assumes XML is arriving and XML will be sent on. There is no special class library to learn because processors conform to either the SAX or JDOM interface. The programming examples are well commented and there is ample documentation.

Support for Services Oriented Architectures

The whole world of software design is currently dominated by the Web Services approach. PolarLake supports XML Web Services in the following ways:

- Wizard based generation of 'thin' wrapping of existing Java classes, EJBs and COM components. These require no coding and generate configuration files which allow PolarLake servers to receive SOAP requests and call the appropriate backend components, and then complete the trip by constructing the expected SOAP response and sending it back. In addition, the appropriate WSDL is also generated and this can be loaded into web servers (also included in a simple web based query tool which allows developers to look at Web Services available on a given web server).
- SOAP, UDDI and WSDL modules allow any PolarLake applications to act as consumers or providers in Web Service based interactions, with minimal effort (The UDDI module supports querying of UDDI registries, PolarLake don't provide a registry in the product). A single PolarLake application can act as a provider and consumer of multiple web services.
- Support for SOAP over JMS

Figure 1 illustrates the service façade approach to building an enterprise services bus. Existing software assets can be exposed as services very easily but this produces fine-grained services that are inefficient when used on external networks and are often meaningless function calls designed for close-coupled intranet systems. SOA requires coarser grained services which map easily to business process. PolarLake circuits provide an elegant solution to the wrapping and transformation of existing components and function calls to provide the business services bus.

In addition to the wrapping and transformation, PolarLake significantly reduces the risks of exposing internal software assets as services through its ability to provide context checks and enforce correct sequencing of method calls. The simplistic approach of exposing methods and component interfaces is a high risk strategy; PolarLake provides a decoupling layer that minimizes risk by enforcing checks and rules in the runtime environment.

The flexibility of the PolarLake circuit designer makes it relatively simple to provide alternative presentations of the services interface so that partner organizations that may have less flexible XML formats can be rapidly accommodated. We believe that as more enterprises and especially SMEs implement their services bus the mismatch of XML schema will be an inhibitor unless tools like PolarLake are in-place.

PolarLake Express

The Express product is a cut-down version of the server, which can be used to expose existing components as Web Services. It consumes COM, EJB and Java classes to provide XML web

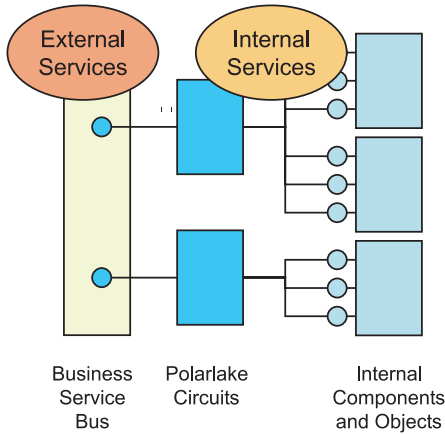


Figure 1: Service Oriented Architectures

services and manages the WSDL and routing of the SOAP message through to the implementation code.

Deploying And Managing Your Solution

PolarLake is a Java application which normally runs as a stand-alone java application, three standard configurations supported are:

- Stand-alone PolarLake Server
- Stand-alone PolarLake Server with embedded Web Server (typically used during development and early testing to avoid the need for setting up and configuring a separate Web Server)
- PolarLake Server embedded in a Servlet Engine (i.e. dedicated to handling requests delivered via the Web)

PolarLake is shipped with a management console, or you can integrate with a SNMP compatible product like HP OpenView or BMC Patrol. Using the supplied management console you can:

- Start/stop PolarLake instances and applications
- Add instrumentation and debug
- Reconfigure instances

- Monitor resource usage and gather statistics
- Hot Swap System and application Management
- Monitor current status and change active rule sets

Initial Impressions

CBDi downloaded the standard evaluation software from the PolarLake web site and installed it on a fairly average laptop. The ‘out-of-the-box’ experience was positive; the installer was informative and gave plenty of feedback and pointers for configuring the system to use an existing Java 1.3 runtime. The application ran fairly quickly considering it was Java. The documentation and tutorial are very clearly laid out and logical and we had no difficulty in running and following the tutorial through. The Getting Started guide covered some fairly simple concepts very well, but kept well clear of difficult programming

content needed for the more complex tasks. The programming guide included a range of source code examples that covered the programming model.

The GUI circuit designer was particularly easy to use and it is easy to understand how much more productive developers would be once the basic components are created. The screenshot in Figure 3 shows a simple circuit that implements a business rule.

Case Studies

The following customer case studies illustrate the type of scenarios PolarLake is being used for.

Financial Services – Fund Management

A leading fund manager has a mixture of platforms: IBM, MQ Series and Microsoft SQL Server. The task was to map the XML documents flowing over MQ from the back-office systems to the SQL server databases. Three key requirements were:

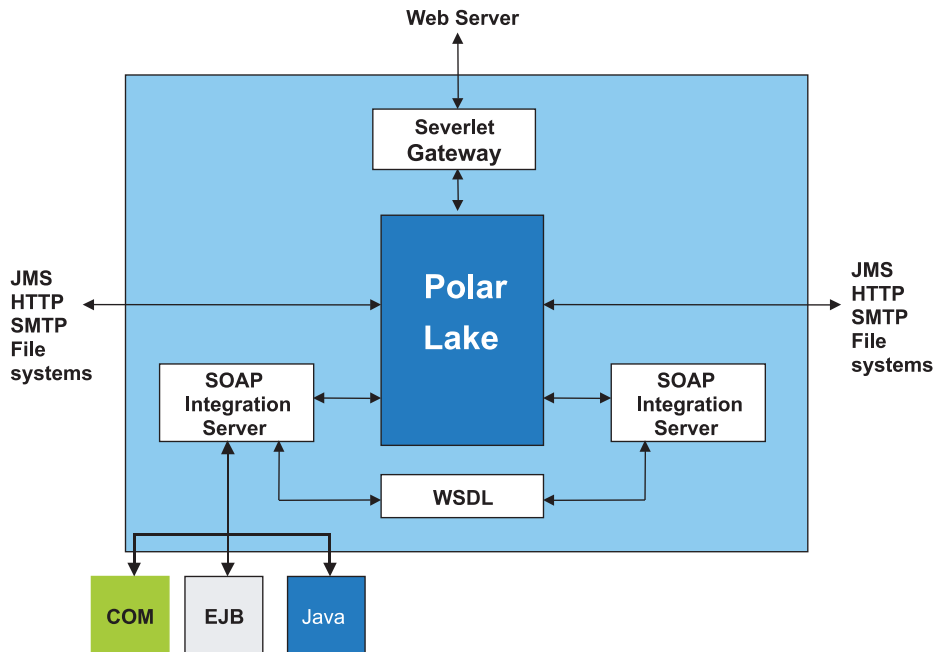


Figure 2: SOAP Integration

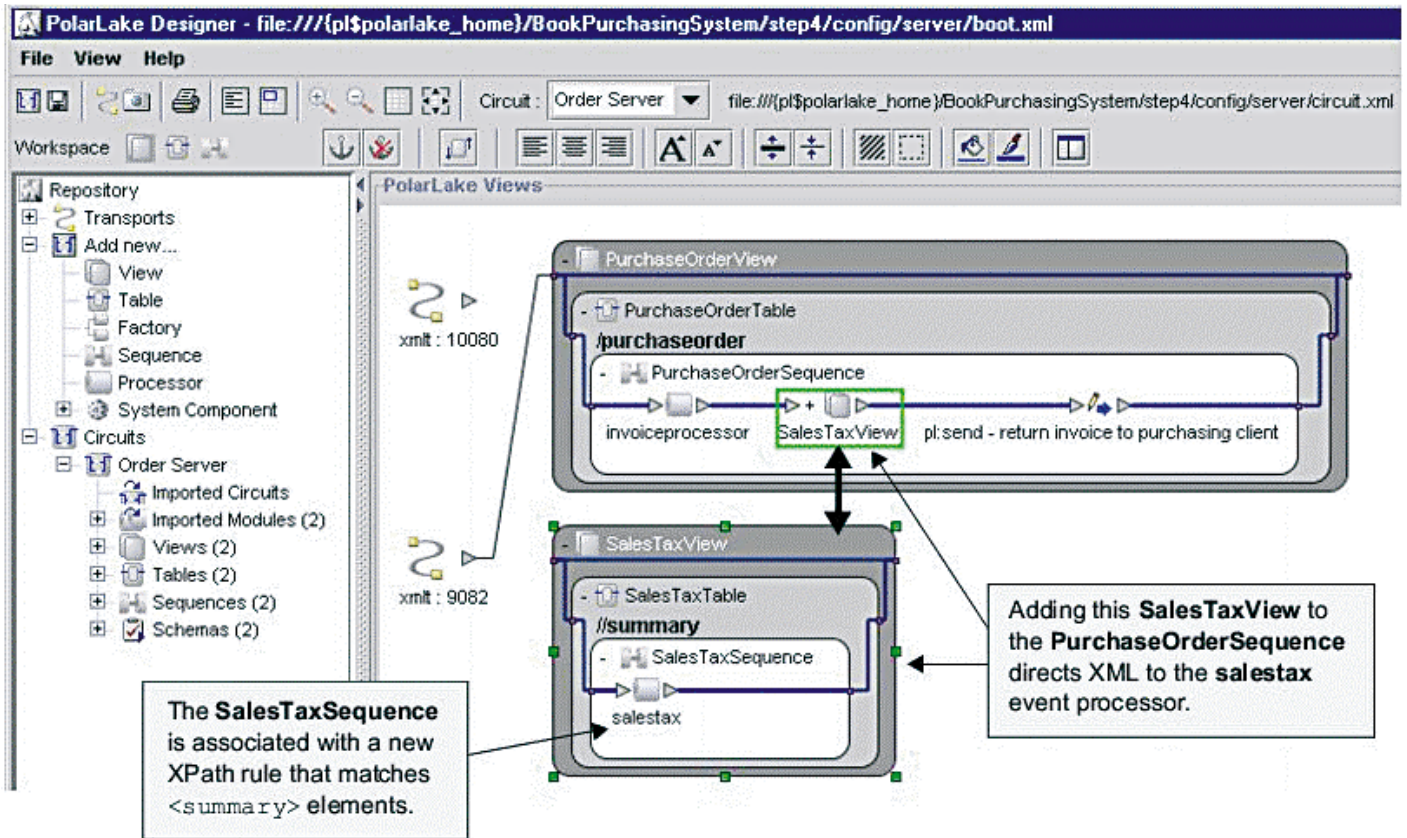


Figure 3: Circuit Designer

- Ability to deploy into a mission critical environment, with set performance, scalability requirements
- Integrate with existing database application (Microsoft SQLserver), selected management platform (BMC Patrol) and messaging infrastructure (MQ-series)
- Provide an easy to use development platform to allow the system to be extended to process new types of documents without disruption and zero coding.

The Solution

Two PolarLake servers were deployed as a fail-over cluster. The application receives XML from MQ Series queues and sends it to the relational database.

The XML documents being processed are long and complicated, reflecting the structure of the financial instruments in each fund. PolarLake receives and validates each document within the transactional boundaries of an MQ Series transaction. Mail alerts and error logging is implemented at each stage of the processing chain. Once unwrapped

the XML content is used to update the SQL Server database using the Database Integrator component.

Telecomms – Information Services Provision

A leading mobile operator in the U.S. with subscribers in three states has focused on data services such as email,

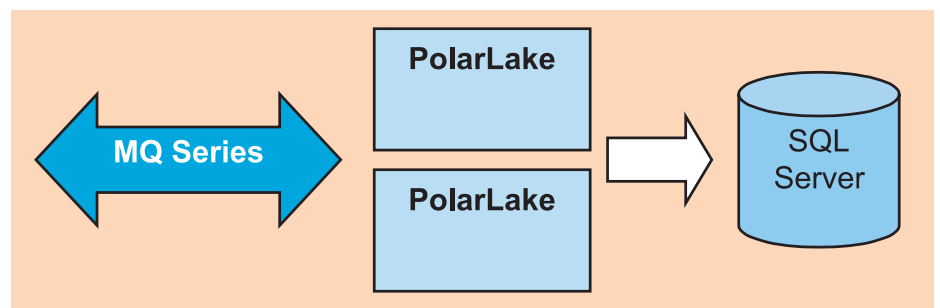


Figure 4: Funds Management Pipeline

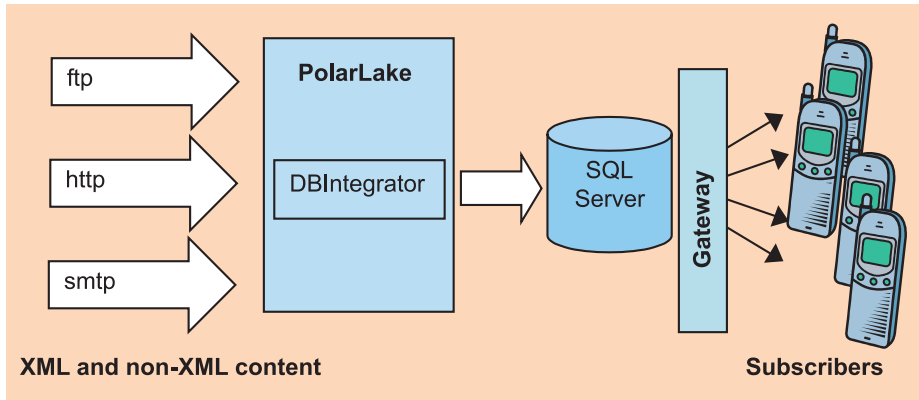


Figure 5: Mobile Data Services Solution

weather, financial and sport. These are made available to its mobile subscribers and to standard phones. In order to increase the range of services available and make the infrastructure more adaptable to new information feeds and XML-based architecture was required.

The existing SQL Server application stores the subscriber preferences and is a staging post for the information to be provided. The data sources are varied and include XML and non-XML content, as well as Web Services. Some of the information feeds include XML documents that run to several megabytes and data such as sports results must be available instantly. PolarLake was identified as having the features needed for the new system, i.e.

- Scalability and reliability: to support their rapidly increasing subscriber base without downtime and without delays.
- Flexibility: to allow the company to define, implement and deployment new services in days
- Performance: to allow time critical information to be received, processed and forwarded to subscribers in real-time.

The Solution

A PolarLake server was configured using the Database Integrator to receive content from a number of different transport connections, assemble new content documents and update the SQL Server database.

Market Analysis

PolarLake stands out from the current clutch of Web Services software products in the way it was focused on the real-world customer requirements of XML and non-XML content processing rather than specializing on the emerging Web Services paradigm. PolarLake does of course support Web Services and uses the Express product, which exposes Java, EJB and COM as Web Services, as an entry level 'product-lite'. But the emphasis is on XML, SOAP is regarded as another form of XML content.

The XML document exchange paradigm is well established in the world of financial services and over the next year the industry is going through some major re-engineering to support standards such as FpML, XBRL, MDDL, swiftML, FIXML, RIXML, NewsML, and so on. There is clearly going to be a vast amount of XML data to process and

PolarLake have responded directly to customers to provide a platform for dealing with it in a flexible and scalable way. Other industries are following on in their adoption of XML standards and we expect to see more activity in telecoms., retail and construction. The imminent widespread adoption of XML for patient records in the UK National Health Service and the e-government initiatives are further examples of growing markets.

Over the last 9 months we have seen the PolarLake product develop to provide more ready-built capability such as the Database Integrator and the Message Integrator and this type of code-free system development will further enhance their potential value to users.

Product Summary

PolarLake is an extremely polished and well produced product. It has a wide range of applications and its potential market is growing fast. We found it easy to install and use and were particularly impressed with the quality and depth of the documentation. It will appeal to:

- **System Architects** looking for a flexible architecture for efficient processing of XML content;
- **Companies** who, having adopted Java, need to jump start their XML document-oriented development;
- **Systems integrators** who need to re-use XML Java components in different applications and who can develop highly optimized industry-specific components.

Above all, they have customers, which is always a good sign!

Jonathan Stephenson
jonathan.stephenson@cbdiforum.com

More information at: <http://www.polarlake.com/>