November 14, 2006

White Paper

Promoting platform connectivity and openness: the Xenon Integration Platform (XIP) and the Xenon Developer's Kit (XDK)

Synopsis

This paper describes Sakonnet's new Xenon Integration Platform, XIP, an application or middleware for advanced and flexible connectivity between Xenon®, Sakonnet's energy trading and risk application, and external systems such as exchanges or customers' internal systems. The paper also outlines the Xenon Developer's Kit (XDK), a component Sakonnet uses and is making available to customers, to facilitate the process of creating new interfaces (using XIP or other means) between Xenon and other systems.

Timothy High
Technical Architect
Sakonnet Technology, LLC

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

Contents

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

Introduction

A company's trading floor needs to be connected closely to exchanges, brokers, information providers, and the company's whole system landscape. These connections have to pass information in both directions in real time, and be easy to set up, maintain and upgrade. New connections must be added frequently.

To address this fundamental need in the energy trading markets, Sakonnet Technology is releasing today the Xenon Integration Platform (XIP), an enterprise service bus for flexible messaging between Xenon®, Sakonnet's energy trading and risk application, and other platforms and systems. This middleware application is based on the Apache ServiceMix open source solution for messaging. XIP will be used for robust interfacing to standard energy industry platforms, including Trayport, the ICE and Reuters, and also for creating new interfaces to meet client-specific needs. XIP uses state-of-the art approaches and technologies to ensure excellent quality of services, e.g. for monitoring and extending interfaces.

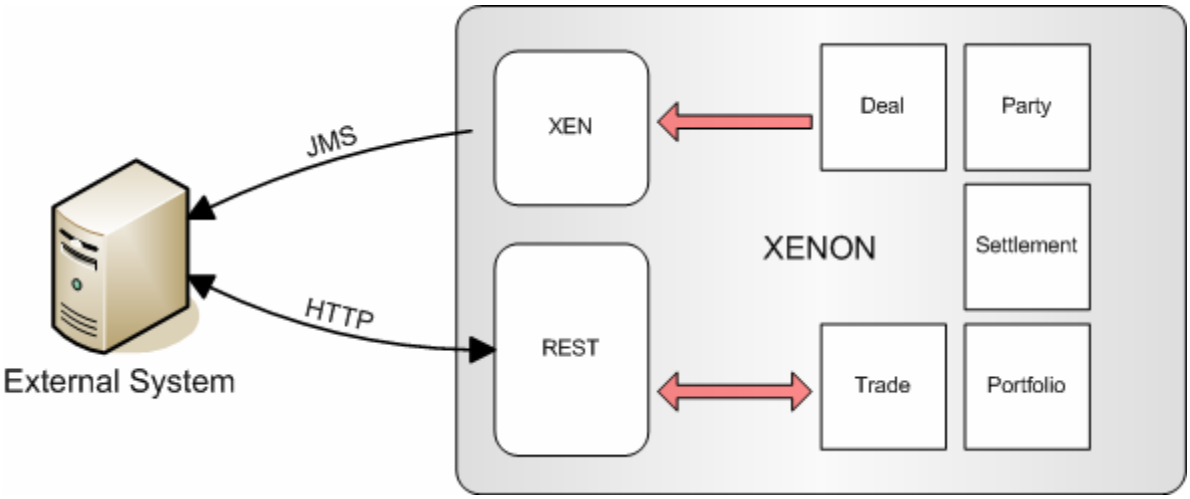
Sakonnet is simultaneously releasing the Xenon Developer's Kit (XDK), a Java API or component which will be used by Sakonnet and made available to its customers, to assist in building new interfaces, either using XIP or independently.

In this white paper, we first describe Sakonnet's interfaces, then outline the role and functionality of XDK. Then XIP and the ServiceMix platform are described. The paper ends with notes on XIP's pluggable architecture, how XIP can be deployed, and how it handles monitoring and error handling.

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

Xenon Interfaces

Xenon has traditionally offered a stable, domain-driven interface for access by external systems. This interface is composed of two principal components: the JMS-based XEN interface for event notifications, and the Xenon REST interface for querying and importing of Xenon data over HTTP(S).



XEN ("Xenon Event Notifications") is a JMS Topic which Java applications may use to subscribe for notifications of Xenon events. Notifications are provided for trade and deal events, including trade creation and status changes, reference data changes, and other workflow-related events. In order to reduce unnecessary traffic, the notifications do not include the complete set of data related to the event, but offer enough information for the subscriber to retrieve the data via the REST interface should they require. Subscribers have the option of choosing the type of notifications they wish to receive through the use of JMS message selectors.

Xenon data is made available to external systems via the REST ("Representational State Transfer") interface[1]. REST allows applications to query for data over the HTTP(S) protocol using simple URLs, without the burden of SOAP or other higher-level protocols. This allows REST to be accessible by any type of application, from scripts and web browsers to enterprise applications.

[1] http://www.xfront.com/REST-Web-Services.html

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

REST services are designed around the essential business entities in Xenon. Functions are currently available for querying on reference data, trades and deals, and settlement information. REST also allows external systems to import new deals using an HTTP POST. There are specialized exportation interfaces as well, providing commonly requested calculations, such as the aggregated physical flows reports. All business entities are represented using Xenon-specific XML formats.

As Xenon functionality evolves, it is our goal to provide complete coverage in REST of all business entities. We intend to keep the REST interface as simple as possible by avoiding the creation of special purpose functions except when there is an obvious benefit for all. We will also continue our efforts to provide complete documentation of these interfaces. As a result, Xenon will have an external interface that is both easy to maintain and to understand.

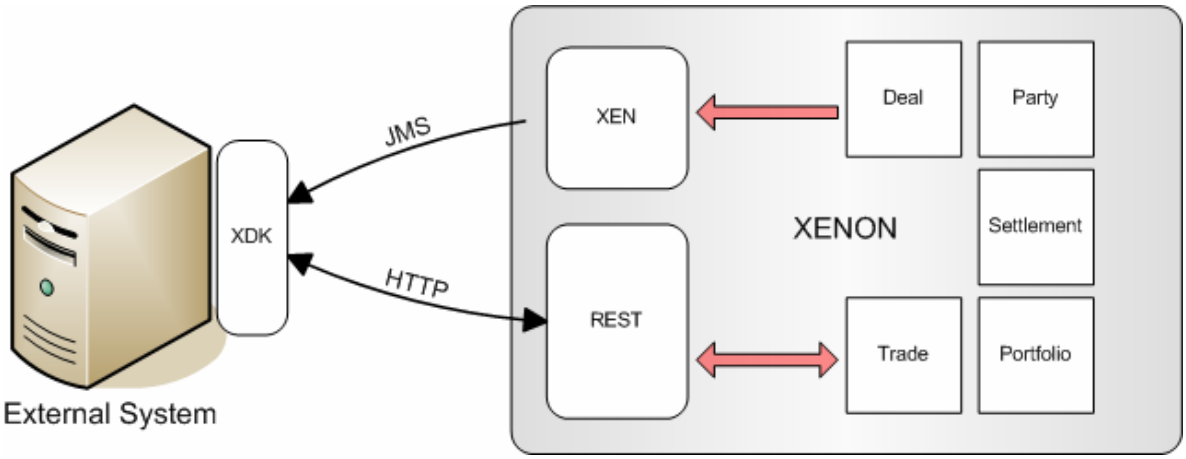
Copyright © Sakonnet Technology, LLC 2006. All rights reserved

The Xenon Developer's Kit (XDK)

While it is important for Xenon to maintain a simple interface, in many cases this is not sufficient. The solution in these cases is to build the necessary functionality outside of Xenon in a custom component designed to act as a bridge to other systems. This approach allows for much greater flexibility and control over the data formatting, content and flows. It also enables customers to deploy new interfaces in their environment on a timeframe independent of Xenon releases (compatibility issues notwithstanding).

While it is true that the REST interface provides a very simple method for accessing Xenon data, there are still many common problems that must be solved with each new interface. The component must connect to Xenon over an authenticated HTTP connection, it must properly form URLs for performing REST queries, and it must know how to deal with Xenon responses.

In order to facilitate the implementation of integrations with Xenon, Sakonnet Technology has developed the Xenon Developer's Kit (XDK), a Java API designed to solve many of these common problems. The XDK is now available as an optional JAR file to customers wishing to implement their own interfaces.



The XDK provides developers with an RPC-style interface with Xenon. Simple calls to plain Java methods are translated into the proper Xenon URL. The XDK handles the details of user authentication, retrieves the requested data, and then returns the information to the caller in the form of an XdkResponse object. This object contains the original response data, plus information regarding the HTTP response code, whether or not the operation was successful, and details regarding the errors in the case that it was not.

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

The first release (1.0) of the XDK, which is compatible with Xenon release 4.7.1, offers functions to:

- Export deal information based on an external or a Xenon ID
- Import and update deals in Xenon using the Xenon-proprietary XML deal format
- Parse XML responses and error messages using the XdkResponse object
- Assist testing and comparing of XML documents

More enhancements are planned for the 4.8 release, including:

- Methods for subscribing to Xenon event notifications
- Object representations of events
- Methods for querying on and exporting Xenon reference data

Many more facilities are planned for the long term. The XDK will provide complete access to REST functionalities, while reducing its complexity. Additional features will include:

- Ability to specify fetch levels to retrieve multiple related entities in a single request
- Both synchronous and asynchronous call modes
- Offline test mode for mocking Xenon behavior in automated tests
- Sample XML files for use in testing and development

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

The Xenon Integration Platform (XIP)

Overview

The XDK is designed to solve the complexities involved in communicating with Xenon from external systems. However, the Xenon interface is just one piece to the puzzle of a working integration between systems. The integration logic and workflow must be implemented in a component which supports the quality of service requirements for the system. Once an integration component is developed, you must decide how and where to deploy it. For this, we have developed XIP: the Xenon Integration Platform.

XIP is an Enterprise Service Bus (ESB) based on the Apache ServiceMix open source solution for messaging[2]. An ESB is at its core a unified platform designed to standardize the way all applications in an enterprise communicate with one another. This may be done by enforcing a single protocol (such as JMS), or by supporting multiple transport methods (for example JMS, Web Services and file transfers) and providing abstractions to hide the transport details. ESBs traditionally offer additional messaging utilities and APIs, including out-of-the-box adapters so that applications can easily "plug in" to the bus, message translation utilities, and easily configurable messaging components, including routers, splitters and others.

ServiceMix is no different in this respect. It uses ActiveMQ, a very popular open source JMS provider, for its messaging backbone. It comes with adapters for many of the most common transport methods, including HTTP, SOAP-based web services, JMS, file system, FTP, and others. It also offers methods for configuring complete message flows, including message transformation, without writing a single line of Java code.

```
<!-- JMS Receiver - CONSUMER -->
<sm:activationSpec componentName="myJmsReceiver"
   service="xip:myJmsReceiver" destinationService="xip:router">
   <sm:component>
      <bean class="org.apache.servicemix.components.jms.JmsReceiverComponent">
         <property name="selector" value="xis_subject='TradeDataChanges'"/>
         <property name="template">
            <bean class="org.springframework.jms.core.JmsTemplate102">
               <property name="connectionFactory">
                  <ref local="jmsTopicConnectionFactory"/>
               </property>
               <property name="defaultDestinationName" value="XIS.ACME"/>
               <property name="pubSubDomain" value="true"/>
            </bean>
         </property>
      </bean>
   </sm:component>
</sm:activationSpec>
```

[2] http://servicemix.org/site/home.html

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

```
<!-- Content-based router -->
<sm:activationSpec componentName="router" service="xip:router">
  <sm:component>
    <bean class="org.apache.servicemix.components.xslt.XsltComponent">
      <property name="xsltResource" value="./router.xsl"/>
      <property name="disableOutput" value="true"/>
    </bean>
  </sm:component>
</sm:activationSpec>

<!-- Query Trade information -->
<sm:activationSpec componentName="queryTrade" service="xip:queryTrade"
  destinationService="xip:tradeFileSender">
  <sm:component>
    <bean class="com.sknt.xip.core.esb.servicemix.components.XenonBasicAuthenticationHttpInvoker">
      <property name="urlExpression">
        <bean class="org.apache.servicemix.expression.JaxenStringXPathExpression">
          <constructor-arg
            value="concat('http://localhost/xenon/acme/deals/', /xenon-notification/deal-change/@deal-id)"/>
        </bean>
      </property>
      <property name="login" value="login"/>
      <property name="password" value="password"/>
      <property name="company" value="ACME"/>
      <property name="timeout" value="15000"/>
    </bean>
  </sm:component>
</sm:activationSpec>
```

Sakonnet has had great success with open source projects in the past, most notably with the JBoss Application Server. ServiceMix is backed by a large open source community. However, as with JBoss, we have contracted professional support for the software. This support is offered by LogicBlaze[3], the same people that built and offer professional support for ActiveMQ. The LogicBlaze team counts among their members the main developers and architects for the ServiceMix project.

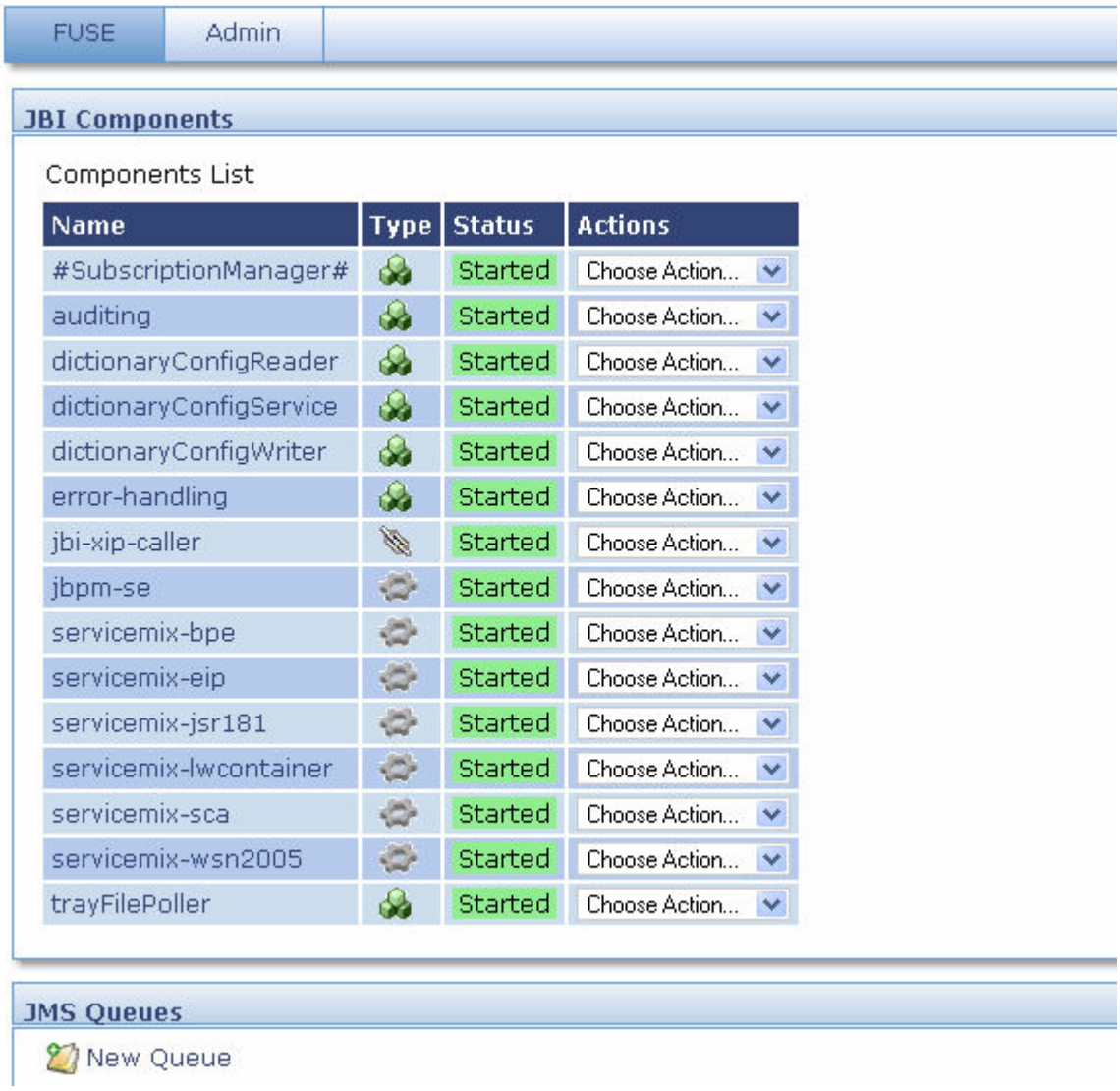
The architecture of ServiceMix is very flexible, and there are many possible deployment configurations. We have selected the LogicBlaze FUSE 1.2 bundle[4] of ServiceMix as our primary deployment platform. It offers a broad set of features, beyond what is available with ServiceMix alone, and all components are tested and certified together by LogicBlaze.

[3] http://www.logicblaze.com/
[4] http://devzone.logicblaze.com/site/fuse.html

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

XIP Services

XIP can be deployed with all the basic services that are packaged with FUSE. This includes the full ServiceMix suite for messaging, an LDAP server for users and roles, and the LifeRay web portal, which hosts the FUSE web administration site. This site offers a view of all components that have been installed in ServiceMix and allows administrators to start and stop them as needed.



Copyright © Sakonnet Technology, LLC 2006. All rights reserved

The ServiceMix component provides all the ESB functions mentioned above. This includes adapters for Web Services and SOAP, email, files, FTP, HTTP, JMS, Jabber, RSS and VFS. The configurable message flows are available via a component known as the ServiceMix "Lightweight Container," which uses a proprietary XML format for declaring endpoints and message destinations.

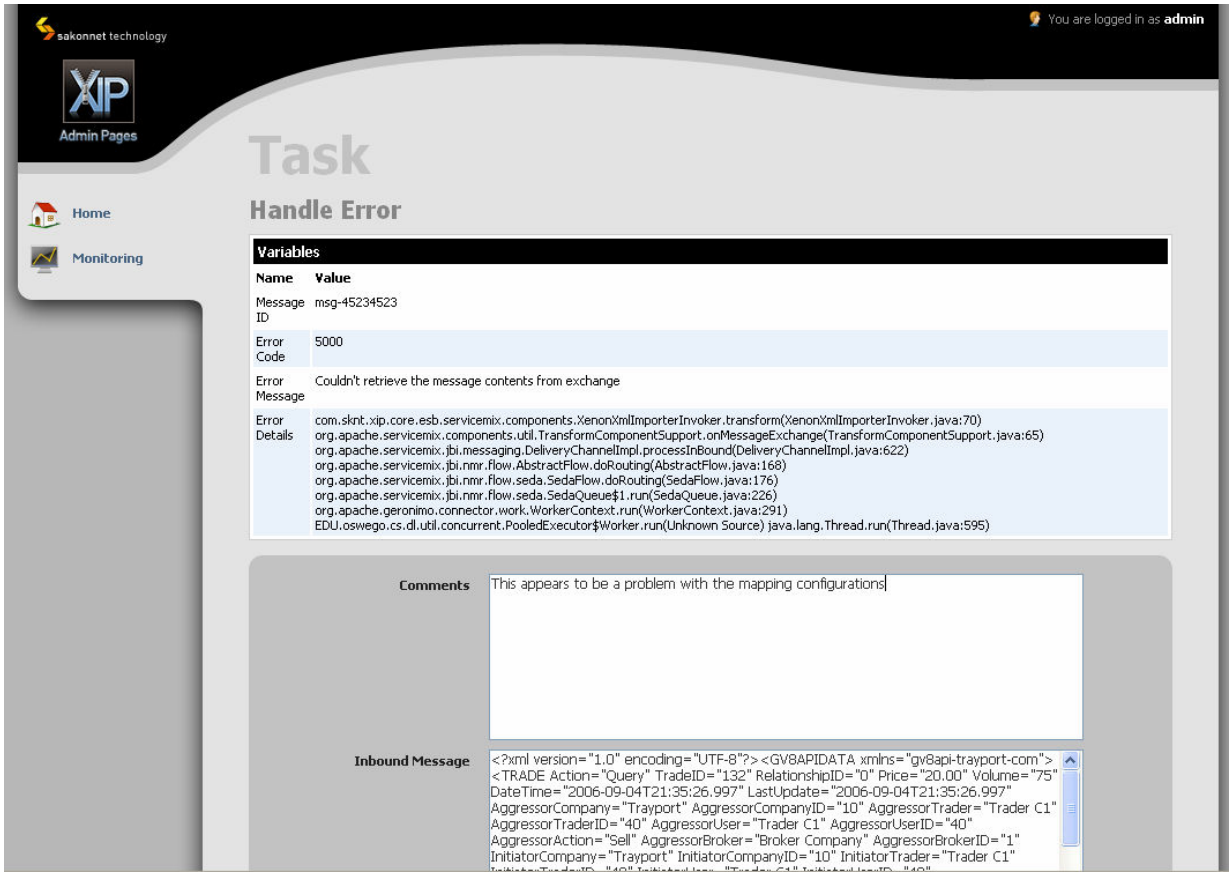
For XIP, we have added several more components to FUSE:

- XIP Basic Services – this is a bundle of standard services that can be used by all feeds which are deployed on XIP. These services include destinations for error handling and for auditing messages. Messaging errors and audit logs are written to the file system currently. Future enhancements will probably include storage of these messages in a relational database and email notifications.
- Message Mapping – a very important part of any integration is the need to translate message data from one system into a format that is readable and meaningful to the other. While ServiceMix provides components for performing simple transformations via XSLT, in the majority of cases this is not sufficient. XIP offers a message mapping engine that is configurable and goes beyond simple transformations to data mapping (translation) and data enrichment (configurable default values and calculated values). The data mapping component is based on a "dictionary" of values, which translates the value provided by one system into a value recognizable by the other. The dictionary can be configured at runtime by placing a formatted Excel spreadsheet into a special folder in the filesystem. The final values for the transformed document are calculated using configurable rules via an API designed specifically for XIP.
- jBPM Workflow Engine[5] – jBPM is a workflow engine that forms part of the suite of enterprise JBoss projects. XIP integrates jBPM with ServiceMix to provide a visual way to build and run integrations. Feeds that use the jBPM component benefit from a complete audit trail of the steps executed within the integration process, and can easily include human interaction as part of the workflow.
- XIP Caller – XIP provides a framework for building custom connectors to systems that are not accessible via the standard transport protocols covered by the ServiceMix components. This includes systems that provide their own API for connectivity, such as Trayport, EEX, or even Xenon. Implementing a new connector involves creating an implementation for the ConnectionManager interface and then configuring (or implementing your own) strategies for receiving messages from the external system. The XIP Caller currently provides strategies for event-driven subscriptions, and for periodic polling of the external system.

[5] http://www.jboss.com/products/jbpm

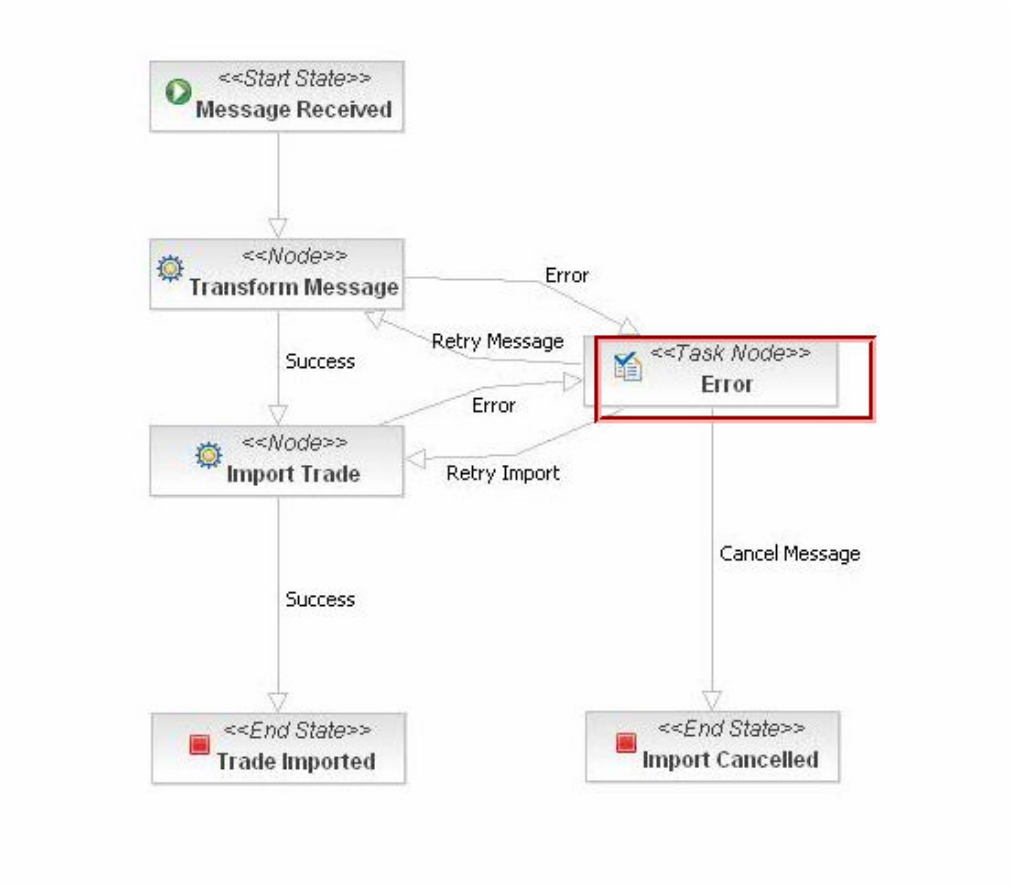
Copyright © Sakonnet Technology, LLC 2006. All rights reserved

- XIP Web Administration – a second administration page is available in XIP for viewing XIP-specific messaging activity, including outstanding error tasks. The site is based on the jBPM web application, and allows administrators to see graphical representations of the import workflow processes. In the future, this web application will be further customized to show messaging information beyond the scope of jBPM.



An example of a task screen for error handling in the XIP Web Administration Site

Copyright © Sakonnet Technology, LLC 2006. All rights reserved



A jBPM process caught in an error state, as viewed from the XIP Web Administration Site

As XIP evolves, more effort will be focused on improving the basic services. Both audit logs and all error events will be visible from the XIP Web Administration console. XIP will also provide options for configuring email notifications for different error conditions.

Although XIP is an open integration platform, effort will also be made towards maintaining a close but loosely-coupled relationship between XIP and Xenon. Plans are underway to provide a new channel in Xenon for XIP to report on the status of its feeds. Traders and Xenon administrators will receive notifications of feed events, such as trade importations, invoicing exports, and error events. These notifications will be available directly in the Xenon GUI Client, so Xenon users will not need to open a separate application to monitor their activity.

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

A Pluggable Architecture

XIP offers a highly-pluggable architecture for integration with Xenon. It is based on the recently approved Java JBI standard[6] for application integration, which describes the concept of a "Service Assembly", what we like to call a "Feed". A Service Assembly is essentially a group of configurations and artifacts that define an integration flow, and can be deployed together as a single package. By the JBI standard, this package can then be hot-deployed to a JBI container at runtime, thus activating the flow.

This concept of hot-deployed components extends to the services themselves, such that you can create a new service and deploy it at runtime, then install an assembly which uses the service, all without having to restart the server. JBI also defines a mechanism for avoiding one of the biggest problems with pluggable architectures: library version conflicts. Each JBI service (or "Component") has its own classpath, which includes classes and JARs found in its own internal packaging, but doesn't include JARs packaged with other components.

This translates into a much more "plug and play" approach to feeds. When a new interface is ready for production, it may be deployed to the XIP server without any need to re-deploy the server itself. Given the number of tools available to developers for the XIP platform, we expect to see a fairly quick turnaround in the time it takes to develop and release new interfaces. Customers will be able to develop their own XIP integrations and deploy them side-by-side with interfaces provided by Sakonnet. Both XIP and its components are only loosely tied to Xenon, and may follow a separate test and release schedule.

Industry Interfaces

Sakonnet has begun working in earnest to provide out-of-the-box implementations for standard industry interfaces. The first one, which is included in XIP Release 1.0, is with Trayport, the leading European power and gas broker electronic platform. Sakonnet plans to implement shortly interfaces with:

- The ICE (Intercontinental Exchange) and EEX for straight-through processing of trades
- Reuters and Bloomberg for market prices
- enerbility for automated reconciliation of bilateral trade confirmations

[6] http://jcp.org/aboutJava/communityprocess/final/jsr208/index.html

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

XIP Deployment

XIP runs with the Java 5.0 JDK, and uses an Oracle 10g database for persistence of the dictionary mappings and for jBPM process information. Currently, all XIP services, including third-party components, use Hibernate for their persistence layer. While no other RDBMS platforms have been fully tested with XIP, it should be possible to deploy them with XIP with only minor adjustments.

The first release of XIP has only been certified against Windows XP. However, there are FUSE 1.2 installations for Windows 2000, Windows XP, Linux, MacOS, and Unix systems that support Java. Note that deployment options may be restricted by the requirements of components which are installed with XIP. Due to limitations imposed by the Trayport API, the Trayport feed must be deployed with XIP on a Windows server.

As an external client of Xenon, XIP should be run on a separate server from Xenon. However, depending on the customer deployment for Xenon (hosted or intranet), the placement of XIP may vary. The location of the XIP server should be carefully chosen based on firewall issues and the requirements of all interfaces deployed on the server.

It is possible to have more than one installation of XIP in a given customer's production environment. However, some services are not easily compatible with being installed in multiple servers. In particular, the jBPM component and the Trayport connector service have special limitations. The jBPM component can run two separate instances, but they will not be able to store all process data in the same context. Instead, they will each have their own separate set of log files, error tasks, and so on. This would complicate administration tasks considerably. The Trayport connector works by subscription to event notifications, and installing it in two separate servers would result in Xenon receiving duplicate trade information.

Monitoring and Error Handling

XIP aims to provide complete visibility for feed messaging activity. This will enable system administrators to easily identify and handle error conditions, and to track system performance to identify bottlenecks.

The first release of XIP offers many options for monitoring the system. As previously mentioned, the FUSE Web Administration site provides visibility into the status of the JBI components, and into messaging activity on the ActiveMQ JMS backbone. The XIP Web Administration site provides auditing of the jBPM processes, in which the majority of interface business logic is executed. When a business or system error occurs during one of its processes, the error appears as an administrator on the "Pending Tasks" list in the XIP Web Administration home page. The XIP Auditing service logs all audit messages to feed-specific files, so that all incoming data can be tracked before the

Copyright © Sakonnet Technology, LLC 2006. All rights reserved

process begins. Any errors that occur outside of the jBPM processes are logged with the XIP Error Handling service, which currently also logs to a feed-specific file, and may send email notifications to a system administrator.

Since XIP is deployed with Java 5.0, it also benefits from the built-in JMX console[7] for monitoring JVM status. This console, which is viewable via tools such as JConsole, offers views on current and historical memory usage, system information and other JVM statistics.

The use of the JBI paradigm for messaging offers the ability to send messages robustly with very little effort. Therefore, message delivery can be guaranteed between components when necessary. This mechanism is used extensively in trade importation feeds, for example.

The current recommended strategy for failover of XIP is to use a manual failover task. The database should be deployed on a separate server, ideally in a clustered environment. Under these conditions, if the original XIP server crashes for any reason, another may be started to replace it. Additional work to improve fault tolerance is planned in 2007.

[7] http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html

Copyright © Sakonnet Technology, LLC 2006. All rights reserved