Building Interoperable Insurance Systems with .Net 3.0 Technologies

Author: Mike Walker
October 2006

Introduction
The purpose of this whitepaper series is to provide guidance around integration challenges. Through this whitepaper we will use an insurance industry scenario to demonstrate interoperability capabilities of the Microsoft platform. Through maturity of many enterprises, we live in a world where there is more than one stack of technology. These platform stacks range from legacy mainframe-based COBOL or FORTRAN types of applications, to the more modern solutions based on .NET, Mobile Systems, or Java – and everything in the middle. As a result, as enterprises have iterated through technologies and technology trends, there has been more than a few Band Aids applied to the various technologies.

Insurance Interop Series
This whitepaper will serve as a guide for architects that are facing integration challenges in the insurance industry. We will show you how to use Microsoft integration technologies to integrate disparate systems in your enterprise. Additionally, this document will provide pragmatic design guidance for building interoperable solutions using open standards such as WS-*.

Additional documents in this series will include the following:
Architecture Overview to Building Interoperable Insurance Systems
Securing Insurance Solutions
Scaling & Operational Management
Deploying Enterprise Solutions
Developing Composite Applications

Technologies that will be covered include:
1. BizTalk 2006 – The integration technology for this solution. The solution also uses the BizTalk business rules and workflow orchestration.
2. Windows Communication Foundation (WCF) – The programming model to develop Web service messages and manage protocol-level communication by using the WS-* protocols.
3. Windows Workflow Foundation (WF) – To create compelling workflows using smart client technologies.
4. SQL Server 2006 – The repository for all of the application and customer data.

This scenario will give us a glimpse into the business process. Like many businesses, each insurance company has its own unique way of handling its process. However, there are some similarities that these businesses share at the platform level. The purpose here is to demonstrate there is a way to leverage these common platform
services to build Service-Oriented Architectures (SOA) giving an organization more agility with the processes that differentiate their specific business.

**Insurance Industry Forces**

In the insurance industry there are many technologies at play, ranging from mainframe, to UNIX, to Windows. With this wide range of platform technologies, it is increasingly difficult to manage and operate while trying to be agile in an ever-changing financial market. For years, organizations have been building and buying technologies to meet these needs. Interoperability has become a necessary evil after the solution has been built and/or implemented. This has left us with point-to-point integrations that only address very specific problems at the application or system level, but not at the business function level.

![Diagram showing point to point integrations](image)

Figure 1: The result of point to point integrations

If not careful, point-to-point integrations over many years result in:

- **IT Portfolio Management** can become unmanageable given the duplication of systems, multiple variations of integrations, management of dependencies of applications, and so forth.
- **Increased Cost** of IT systems dramatically raises due to the number of custom integrations
- **Loss of Agility** because development of systems are slowed significantly as a result of increased code complexity, limited re-usability and lack of standardization in the enterprise

So what does this mean to many insurance carriers? It means that interoperability is at critical importance – not only as an efficiency issue, but also as a competitive
differentiator. In these days of modern competition, companies must increase the return on investment (ROI) of their IT systems by streamlining the processes and becoming more agile to stay competitive.

Our goal is to address the industry challenges with a set of enterprise-ready technologies on the Microsoft platform. We use the following principles in the examples:

- Enterprise Class Solution
- Standard Communications:
  - Use WS-* Standards
  - ACORD Messages
- Ensuring Interoperability with existing solutions

**Business Terms Used in this Document**

**ACORD** - ACORD (www.acord.org) is a nonprofit association whose mission is to facilitate the development and use of standards for the insurance, reinsurance and related financial services industries.

**Order System** - Creates requests for external data, transmits them to the appropriate third party data provider, manages responses received and matches responses to the appropriate original requestor.

**Third-party service provider** - An external system to fulfill an underwriting requirements request. For example, a credit rating system.


**Broker System** - Possible smart-client front end system for order entry and progress monitoring used by an insurance broker. Other front-end systems are also possible, such as a Web portal to brokers, or a Web UI for self-service order entry by customers.

**Life Insurance Policy Scenario**

The customer, Robert, wants to purchase a platinum-level one-million dollar life-insurance policy. The broker, Tom, enters Robert’s policy application using his smart client application. The policy is sent to the Order system where it is processed and routed to the appropriate systems to begin the underwriting process. While in the Order system, third-party services are kicked off. For this scenario we will use a Paramed, a third party service that verifies insurers’ health insurance and medical records.

The built-in business logic can also generate requests to third parties if a certain condition is met. This could be the broker or another partner of the insurance company.
Figure 2: Business process used for our scenario

**Architecture Overview**

This section will walk through the high-level logical architecture used in the scenario. The details around specific aspects such as: security, messages, development, and deployment will be provided in other papers in this series.
To ensure applicability with real world challenges, we derived a set of high level requirements were derived.

Requirements
The following requirements are for an enterprise-class solution.

- **Must interoperate with existing commercial off the shelf applications** – As discussed above, many organizations purchase and customize software. It is critical to address this.

- **The integration technology must be Web Services** – Many forms of communication such as binary communication are proprietary. Until the emergence of Web services, there was no standardized way to communicate messages. Web services provide a way to communicate across heterogeneous platforms.

- **WS-* standards must be used** – Web services using SOAP and WSDL have been industry integration standard for years. However, these traditional Web services lack the robustness needed for messaging. The WS-* standards provide these necessary features without the usage of binary communication.

- **Long-running workflow** – Management of long-running orchestrations has been difficult, especially when that workflow spawns many smaller external workflows in which case reconciliation and transaction management can become complex.

We use BizTalk as the message hub for this solution, given its rich capabilities and the strong need this insurance solution has with tying multiple systems together and managing multiple external workflows.

![Diagram of message bus technology](image.png)

Figure 3: Using message bus technology
Shown in Figure 3 above is an enterprise view of BizTalk as an enterprise service bus (ESB). Remember that it is not a requirement that this is used as an ESB. This whitepaper refers to this layer as just a message layer, so that you can incorporate it into your solution in either case.

The rationale for using BizTalk is that it provides a centralized platform for the following capabilities:

- **Business process management** – Centralizing re-usable business process not only lends to service orientation but also provides a mechanism for organizations to augment existing or purchased commercial off the shelf (COTS) applications without the complexity of modifying them.

- **Workflow orchestration** – Management of multiple workflows can be simplified through this platform. Instead of coding or reconciling each workflow, solutions can be managed as they should. We do this by creating one workflow to manage the business process from the beginning to the end that is able to orchestrate multiple internal system workflows.

- **Rich adapter support** – Jump-starting development is critical to organizations. BizTalk has a wide range of adapters to support your integration needs. In the insurance space there is an ACORD adapter that can jump start your integrations. In conjunction with the ACORD adapter, the Web Services Adapter and File-Based Adapters are available for BizTalk (see: http://www.microsoft.com/biztalk).

- **Message routing and transformation** – Message routing can be very complex when the messages need to be transformed so that other systems understand the message. BizTalk can provide a platform to reduce the complexity and still align with open standards.

**The Insurance Agent Policy System**

Currently the technology trends in the insurance industry vary from portals, thick clients, 3270 mainframe terminal emulation screens, and smart clients. Given the diverse number of applications and vendors in this space, we choose a smart client user interface to provide the optimal experience for the agent for the following reasons:

- Offline and online modes
- No dependencies on network connectivity.
- Rich user experience with much greater functionality.

A disconnected model for agents makes sense in many situations as brokers an often be mobile or have limited connectivity to network resources. However, since we will be using Web services as the core of our messaging strategy when architecting this solution the manner in which the end broker submits policies should be trivial.

For the client side architecture we used Windows Forms as the user interface, this provides the user interface needed for the agents. There will be several controls such as data grids, text boxes, and command buttons. A data grid on the Windows form will serve as the window into the policy pipeline for the broker. We use Web services to update this data grid to ensure real-time updates.

Since this is a smart client, returning data can be cached for offline viewing and updating. This provides significant benefits to the brokers. In addition to the data, a small layer of business logic would reside on the client application. The majority of
the application logic will reside on the insurance company side. The rationale here is we will have light rules to drive UI functionality.

Smart Client

WinForms

Windows Communication Foundation

Security Management

http://www.acord.org/Standards/

Figure 4: Client Logical Architecture

To make the calls from the client to the messaging tier we will use Windows Communication Foundation (WCF). WCF will send SOAP 1.2 Web services messages using the ACORD messaging schemas. The WCF layer will provide a unified development model for our developers when coding communications. From the protocol perspective we will use a series of WS-* standards. However, this is not enough to ensure interoperability. Usage of the ACORD industry standards is key as well. We should be able to interoperate between "homegrown" applications, COTS applications, and third-party services seamlessly.

**Messaging Architecture**

The use of Web services enables this broad variety of channels to leverage a common Web service that receives new business applications into the underwriting process in the form of an ACORD 103 message that includes a policy number that has been assigned, and that will be used for tracking/correlation purposes throughout this demonstration. This ACORD 103 New Business Submission message
will be based on a SOAP Message Transmission Optimization Mechanism (MTOM/XOP) attachment containing the binary representation of Robert’s signature to authorize release of medical information as required by HIPAA. It is absolutely critical that the ACORD standards are incorporated in our messaging. This will ensure portability of the architecture.

It is also essential that communications are secure and reliable. To achieve this we will use WS-Secure Conversation (WS-SC) for personal information that may pass through an undetermined number of intermediaries. We also use WS-SC for high-volume, frequent requests (such as credit checks) which will be required for all new policy applications. We use WS-Security for less frequent requests such as an Attending Physician’s Statement (APS), where the overhead of session establishment is not justified by the request volume. We also use TLS/SSL (also known as HTTPS) in rare cases where a service is directly processing requests without any intermediate routing.

For messaging where tracking reception is important, such as ensuring receipt of a new policy to claim a commission, we use WS- Reliable Messaging (WS-RM). We also use WS RM for data requests that are expensive to process (typically involving human workflow such as APS queries). This ensures that requests are only delivered once and avoids expensive duplicate requests.

For long-running messages we used WS-Secure Conversation (WS-SC) (See References, below.)
Figure 5: Client message exchange patterns

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Business Processes</th>
<th>WS-* Protocols</th>
<th>Architecture Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submission of new policy (103 Request)</td>
<td>Broker-Client Underwriting Process</td>
<td>WS-Security (WS-S)</td>
<td>WS-S used for personal information that may pass through an undetermined number of intermediaries.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WS-Reliable Messaging (WS-RM)</td>
<td>WS-RM to track message receipt. Since infrequent transactions, no need for session-oriented security mechanisms like WS-Secure</td>
</tr>
</tbody>
</table>
Conversation.

<table>
<thead>
<tr>
<th>Status Queries (122 Request / Response)</th>
<th>Broker-Client Underwriting Process Fulfillment Process</th>
<th>WS-Secure Conversation (WS-SC)</th>
<th>Non-critical and individual request or response messages that can be retried easily, but still contain personal information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underwriting Requirement Order Request (121)</td>
<td>Underwriting Process Fulfillment Process</td>
<td>WS-Secure Conversation (WS-SC) or WS-Security (WSS) or Transport-level security (TLS/SSL)</td>
<td>These messages contain personal information.</td>
</tr>
<tr>
<td>Underwriting Requirement Order Response (1122)</td>
<td></td>
<td>WS-Reliable Messaging (WS-RM)</td>
<td>Use WS-SC for high-volume, frequent requests (such as credit checks). Use WS-Security for less frequent requests, where the overhead of session establishment is not justified by the request volume. Use TLS/SSL where a service is directly processing requests without any intermediate routing. Use WS RM for data requests that are expensive to process.</td>
</tr>
</tbody>
</table>

Table 1: Business Process Messaging Design Decision Matrix

You may ask yourself, after making a submission why is the status returned in a separate transaction? Well the reasons are twofold. First, it is important for this to be asynchronous, and the ACORD standard does not allow an implementation without separating the status from the submission. Second is that the Broker will be getting status returns periodically through the course of the application process by queering the Status Service.

**The Insurance Carrier Systems**

When architecting the server side of the solution there were particular aspects and assumptions considered.

- This architecture accounts for fragmented systems.
- Functional areas are self-contained and need to be managed.
- Operating systems and development environments differ.

As a result of this, there are a significant number of point-to-point integrations with very specific applications, thus causing proprietary implementations. In this solution, the facade layer will be created around these existing applications.
Here you can see how we are using the enterprise service bus (ESB) as a message bus. This layer will serve as the centralized messaging layer that will manage our internal and external messages. Management and orchestration are key benefits of this architecture.

An infrastructure like this can bring order to the chaos of disparate point-to-point integrations by putting intelligent, long-running orchestrations and policies around transactions in one layer instead of many. It would be common to have several distinct COTS-based applications in upwards of five or six systems to accomplish an end to end transaction. We are reducing these systems significantly by consolidating the redundant functions such as workflow and messaging, leaving infrastructure level functionality where it belongs and keeping the business logic applicable applications.

It is important to remember that this message bus is a *logical* representation. The *implementation* view can look very different. For example, the message bus could be several BizTalk servers, or there could be servers in different DMZ environments to manage both internal and external communications.

Figure 6: Insurance Message Bus
The next tier down, which is where specific business functions are performed, contains two different legacy systems wrapped with an interface: the Order System and the Fulfillment System. The reason why we are keeping these as separate systems instead of consolidating them is that the majority of the time these would be two separate COTS-based systems.

A Status system was added for the following reasons:

- Provide centralized way to report status to the agents.
- Reduce the number of interfaces and control logic needed to query multiple systems.
- It fits nicely with the orchestration capabilities with our ESB for our long-running workflow.

The Ordering system and the Fulfillment system have been converted into course-grained services. By doing so we have removed the dependencies of independent implementations. All communication that occurs to these systems now goes through our message hub. The exposed Web service end points that are managed from the message bus can then be managed with orchestration technologies built in BizTalk.
Now that these applications are exposed as Web services any technology that can accept Web services XML can integrate with these applications. This removes the tight coupling of other technology protocols that would limit interoperability. For example, you could just as easily use existing Java-based systems if those were your legacy systems.

SQL Server is used here to store application data in the database layer. Since the core focus of this paper is integration and composite applications, we will not highlight this.

The third-party services referenced are external services that are called by the fulfillment service. These services have varying protocol needs. However, this paper will show how WS-* standards can provide increased functionality for your services. It is important to note that many of the real-world insurance third-party services only support XML-based communications, not the more advanced SOAP-based Web services. The messaging architecture sections, below, will have more on the third-party services.
Insurance Carrier Messaging Architecture

This section walks through a Basic Life Policy that is processed by the Insurance Carrier. Based on the information Robert supplied, the business rules/heuristic logic defined in the Underwriting process decides that an Attending Physician Statement APS (that is, a physical) is also required.

Since another provider must fulfill this request, the order system builds an ACORD XML TransType 121 General Requirements Order Request transaction (TXLifeRequest) and transmits it to a secondary external ordering system for Robert’s physician (the APS System). This message also contains the MTOM/XOP attachment of Robert’s signature that was originally carried on the ACORD 103 New Business Submission, authorizing his physician to release his medical information to the insurance company.

At some point, Robert’s physician will process the APS order by verifying that Robert’s signature matches the one he has on file, and then examining Robert’s medical history, filling in the necessary information required on the APS report.

After the physician has completed the APS report, an 1122 General Requirements Status/Results Transmittal message is generated and transmitted back to the endpoint reference specified in the WS-Addressing ReplyTo specified in the ACORD 121 request above. This message will also be delivered reliably using WS-Reliable Messaging.

The rest of the business process runs, including any automated-actuary decision. However, in this case, because there is an APS and possibly some additional information that cannot be processed automatically, the case is flagged for an Underwriter’s review and approval.
In the insurance industry a fulfillment system or service is very different than the process of fulfillment:

- **Fulfillment System** – A system or service that receives a request and fulfills it. Think of a fulfillment service as an integration component for gathering data. In this scenario, the fulfillment system is responsible for pull the various reports from third-party providers.
- **Fulfillment Process** – The process in which a policy is issued by the insurance carrier.

---

**Figure 9: Underwriting process message exchange pattern**
You may ask why we kept the fulfillment service. For this scenario we are assuming that systems such as these are purchased as black-box solutions. This is not to say that you could not remove these layers and incorporate them into a message bus instead.

Choosing a messaging pattern is not as clear cut as choosing one set of standards. When designing this part of the solution we had to take a step back and look at the business and legacy aspects of each individual transaction.

Some transactions such as receiving credit a credit report were easier decisions. However, other transactions such as pulling an APS report required the ability to contain attachments.

Here are some aspects to consider when designing your messaging:

- **Understand the business process** – It is critical to understand how the business uses these messages. For example, securing data. If the data being sent is not sensitive, you do not need to take extensive security precautions for the message.

- **Understand how the transactions are consumed from the service providers** – This can be both internal and external. Many times when relying on service providers there are technical limitations. This can range from standards support to hours of operation.

- **Security** – This is often overlooked. Protocol level security such as SSL/TLS often is sufficient, but not always. Make sure you evaluate the sensitivity of the data and review the message paths to determine how many end points there are before the ultimate consumer.

- **Be realistic and pragmatic** – When designing these services do not go overboard trying to use every standard. Do not force a standard into a message if it does not belong. This will only introduce unneeded complexity.
What is the Value?

We talked quite a bit about the Microsoft platform and development technologies by walking through the scenario. We also highlighted architecture decisions. But what we did not do is highlight the features of these Microsoft technologies.

The following are the core benefits using Microsoft technologies in the insurance industry:

- **Business process automation** - Business processes are complex and specific to each carrier. With the orchestration tools provided in BizTalk, orchestrations
can be developed by business analysts, removing the developer from this process and enabling the business.

- **Reduction of integration code** – With the custom adapters in BizTalk and the unified programming model of WCF, the code required to integrate systems is drastically reduced.
- **Alignment with standards** – WCF and BizTalk are based on Open XML standards out of the box. No more custom coding to incorporate Web services standards.
- **Productivity** – With an integrated Visual Studio IDE and .Net 3.0 technologies, both the tools and the development language provide substantial productivity gains over other languages.

**Conclusion**

As demonstrated above using protocol level standards alone is not enough; capturing the business side of the messaging transactions is key to making interoperability work for your business. This is true across all industries, not just insurance. We have Web services standards but that is not enough. There is still a level of due diligence that is required to make the optimal technology decisions for your organization. With this specific reference implementation, we go through a real-world scenario and determine the optimum messaging with this scenarios business forces. This can serve as a guide to help you choose message exchange patterns in your enterprise. With all architectures, there are trade offs when choosing specific standards. It is important to understand these tradeoffs and be willing to take on the resulting implications.

Microsoft is committed to making the job of architecting and developing service-oriented solutions easier for its customers. As it is shown here that Microsoft has removed many of the industry barriers complexities that daunt the customers today. These range from providing thought leadership in the industry standards to automating and building out-of-the box Web services support.

**Resources**

Microsoft Financial Services Architecture Center  
http://msdn.microsoft.com/architecture/industry/finserv

BizTalk  
http://msdn.microsoft.com/biztalk/

.Net 3.0 Framework  
http://msdn.microsoft.com/netframework/

Web Services Development Center  
http://msdn.microsoft.com/webservices

MTOM 1.0  
http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/
SOAP 1.2  
http://www.w3.org/TR/2003/REC-soap12-part1-20030624/  

WS-Addressing  
http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/  

WSS 1.1  

Username Token Profile 1.1  

X.509 Token Profile 1.1  

WS-Secure Conversation  

WS-Trust  

WS-Reliable Messaging  