Microsoft

Implementando transações de negócio seguras entre organizações

Implementando transações de negócio seguras entre organizações

Publicado em: outubro de 2003
Aplica-se a: Microsoft® BizTalk® Server 2002 e Web Services Enhancements para Microsoft .NET

Autores e desenvolvedores: Adam Zilinskas, Morris Brown, Brian Loomis
Editor: Mary Browning
Desenho gráfico: Steven Fulgham, Matthew Pinkerton
Produção: Scott Dines, Dave Swift, Sigrid Elenga, Bryan Franz
Revisores técnicos: equipe de produto do BizTalk Server, equipe do XML Messaging

Sumário

VISÃO GERAL

Introdução

Para quem tem experiência com o Microsoft® BizTalk® Server 2002, a arquitetura do Microsoft .NET Framework e o Microsoft Visual Studio® .NET, já conhece o Microsoft BizTalk Server Toolkit para Microsoft .NET. As ferramentas e exemplos de aplicações incluídos nesse importante toolkit demonstram como:

- O BizTalk Server funciona em conjunto com os Web services.
- Os recursos do BizTalk Server complementam os Web services.
- O Visual Studio .NET se comunica com o BizTalk Server e o estende.

Entretanto, o toolkit não define nem recomenda padrões ou práticas de segurança. Isso não é problema, pois agora contamos com um importante componente na estratégia .NET para o BizTalk Server: Web Services Enhancements para Microsoft .NET (WSE).

O WSE oferece uma implementação da arquitetura de Web services para desenvolvedores que criam esses serviços usando ASP.NET e aplicações cliente com o Microsoft .NET Framework. Além de permitir aos desenvolvedores incorporar recursos avançados aos Web services como: segurança, roteamento e arquivos anexos, a implementação das especificações de Web services no WSE permitem incluir três recursos importantes para aplicações distribuídas que usam Web services:

- Aplicações seguras entre plataformas e domínios de confiança.
- A rota percorrida por uma mensagem SOAP até um Web service pode ser delegada de forma transparente entre servidores web;
- A comunicação entre Web services pode conter anexos que não são serializados em XML.

O WSE oferece aprimoramentos no suporte a Web service já encontrado no .NET Framework.

Envie feedback sobre este documento para arquitetodesolucoes@microsoft.com.br.

O que é WSE?

O WSE é uma extensão ao Visual Studio .NET e ao.NET Framework, que oferece implementações de recursos avançados de Web services de forma antecipada, com ênfase em segurança. O WSE oferece implementações das especificações de WS*, em constante evolução, que oferecem comunicação segura, confiável e transacionada com Web services. O WSE possui um ciclo de liberação de funcionalidades mais curto do que o Visual Studio .NET, oferecendo aos desenvolvedores implementações antecipadas das especificações à medida que estas evoluem e são padronizadas.

O WSE contém bibliotecas e componentes para utilização com as classes do .NET Framework, estendendo o SOAP para incluir assinaturas digitais de mensagens e criptografia de documentos entre Web services (WS-Security), capacidade de estabelecer relações de confiança e conversações seguras (WS-Trust, WS-SecureConverstation), roteamento e referenciação entre Web services, suporte a anexos, e um modelo de objeto de mensagens SOAP. O WSE também suporta criptografia de mensagens tanto simétrica (“segredo compartilhado) quanto assimétrica (infra-estrutura de chave pública ou PKI,).

Segurança e Roteamento de Web Services com WSE

De que forma o WSE se relaciona com o BizTalk Server, Web services e, principalmente, este guia? Este guia aborda exatamente isto. O guia trata da correlação entre estas três tecnologias de BizTalk Server, Web services e WSE e de como o BizTalk Server pode, efetivamente, utilizar os recursos do WSE para criar Web services altamente seguros e adaptáveis viabilizando a implementação de processos de negócios entre organizações.

Como o WSE oferece segurança aos Web services?

O WSE oferece os seguintes recursos que contribuem para a transmissão segura de mensagens SOAP:

- Credenciais seguras permitem a segurança escalável de Web services através de toda a rota de uma mensagem SOAP. Difere de um transporte seguro, como SSL (Secure Sockets Layer), que protege apenas entre um ponto e outro.
- A assinatura digital permite que um destinatário de mensagem SOAP confirme criptograficamente que uma mensagem SOAP não foi alterada desde que assinada. Também confirma quem assinou a mensagem, oferecendo efetivamente um mecanismo para autenticação.
- A criptografia torna altamente provável que apenas o destinatário a quem realmente se destina a mensagem possa ler seu conteúdo.
- Um serviço confiável de emissão de certificados pode ser estabelecido para recuperar e validar identificadores de segurança.
- Uma conversação segura também pode ser estabelecida diretamente entre partes de forma que a autenticação e a autorização de chamadas dentro de uma sessão possa ocorrer mais rapidamente do que operações criptográficas mais complexas.

Como o WSE roteia os Web services?

Uma aplicação distribuída usando WSE pode ser projetada de forma que a topologia da rede que implementa a aplicação seja transparente aos clientes. Para definir uma topologia de rede transparente, configure um computador intermediário para executar o roteador WSE.

> ***Nota***
> *Na nossa aplicação exemplo, não executamos o roteador em um computador intermediário.*

Os clientes enviam as mensagens SOAP para o roteador WSE em vez de fazê-lo diretamente para o Web service. O roteador WSE delega a mensagem SOAP para um computador que hospeda o Web service, que pode ser alterado por meio de um arquivo de configuração presente no roteador WSE.

Uma das vantagens de usar o roteador WSE com uma aplicação distribuída é que um computador que hospede um Web service pode ser desligado para manutenção sem modificar o código do cliente nem sua configuração. Um administrador do computador que hospeda o roteador WSE pode fazer todas as alterações necessárias para redirecionar as mensagens SOAP para outro computador. Para fazer isso, um administrador prepara um computador backup para ser colocado on-line e hospedar o Web service enquanto o roteador continua a delegar mensagens SOAP para o computador primário. Em seguida, o administrador prepara um arquivo web.config que especifica o arquivo contendo o cache de referência e um novo cache de referência contendo o URL correspondente ao computador de backup. (Um cache de referência é um arquivo XML que contém o destino final dos URLs recebidos por um roteador.) Quando o computador primário é desligado, o arquivo web.config e o cache de referência são colocados no computador que hospeda o roteador. Mensagens SOAP subseqüentes são, então, dinamicamente roteadas para um computador de backup processo que é transparente para a aplicação cliente, que continua enviando mensagens SOAP para o roteador.

O WSE também suporta o encaminhamento de mensagens SOAP a seus destinatários baseado em informações que indicam uma rota alternativa para um Web service. Um destinatário de mensagem SOAP pode, então, rotear mensagens dinamicamente de acordo com a referência sem a intervenção do administrador. O WSE suporta apenas o encaminhamento de mensagens SOAP não destinadas a um computador com o roteador WSE.

Visão geral sobre segurança de Web services no contexto B2B com uso de WSE

Para demonstrar essa conjunção de tecnologias, criamos uma aplicação exemplo B2B (business-to-business) que você poderá usar como referência para o desenvolvimento de sua própria solução usando o BizTalk Server, a plataforma .NET e o WSE. Essa aplicação exemplo demonstra como usar o WSE para criar Web services que realizam autorização e criptografia de dados. A aplicação exemplo também mostra como o BizTalk Server interage com os web services para a troca de mensagens seguras.

Nossa aplicação exemplo funciona dentro de um cenário genérico de compras eletrônicas (e-procurement) que envolve um fabricante que manufatura um produto acabado com peças fornecidas por um ou mais fornecedores. Embora o fabricante use o BizTalk Server para coordenar seus processos de negócios, o fornecedor não o faz, e integra suas interações com o cliente usando Web services. A Figura 0.1 mostra o processo de alto nível entre esses dois parceiros de negócio.

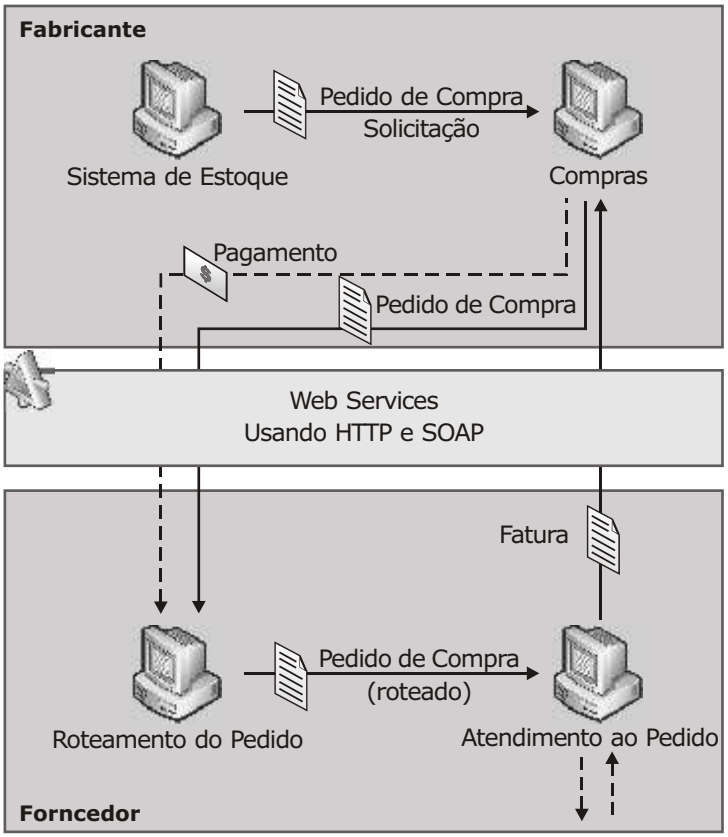

Figura 0.1 Visão geral do processo de negócios

Ainda que a Figura 0.1 mostre o fluxo de dados, vamos esclarecer possíveis dúvidas analisando o processo de negócios, a começar pelo fabricante.

O que acontece no fabricante

Primeiro, o fabricante determina que o estoque para determinada peça está baixo. Para manter as linhas de montagem em funcionamento, o fabricante automaticamente pede uma nova remessa de peças ao fornecedor.

Para realizar esses pedidos de forma automática, o fabricante possui um sistema de monitoração de produção em um servidor na retaguarda da produção que envia um pedido de compra para um sistema de compras baseado no BizTalk Server. Nesse momento, o BizTalk Server ativa um XLANG schedule e submete um pedido de compra ao Web service do fornecedor.

> ***Importante***
> *Durante essa transação, o fabricante assina digitalmente a mensagem com sua chave privada (para que o fornecedor saiba que o pedido partiu dele) e criptografa o pedido. Isso impede que os concorrentes tenham acesso ao pedido de compra ou, ainda pior, que criem seus próprios pedidos e mandem a fatura para outro lugar.*

O que acontece no fornecedor

O fornecedor tem um Web service que aceita pedidos de compra e os roteia aos Web services apropriados internamente para processamento. Um Web service de roteamento existe porque pode haver um Web service diferente para cada tipo de pedido de compra recebido ou o fornecedor pode subcontratar a peça de outro fornecedor e precisa passar o pedido de compra adiante.

Portanto, o fornecedor aceita o pedido de compra com um Web service e processa o pedido em um segundo Web service. Depois de processar o pedido de compra, o fornecedor envia uma fatura ao fabricante. A fatura é recebida por um Web service no fabricante e associada ao pedido de compra original para reconciliação. O WSE fornece segurança para os documentos envolvidos na transação de negócio.

O que você aprenderá neste guia?

Este guia se concentra no BizTalk Server, na plataforma .NET e no WSE, explicando em detalhes como implementar um sistema genérico de compras eletrônicas através da integração das tecnologias do Windows Server System™. Para conseguir isso, este guia:

- Demonstra a integração do WSE em um cenário B2B comum.
- Explica e demonstra práticas de segurança consistentes.
- Exibe importantes recursos dos produtos, como correlação, manipulação de formatos de dados díspares, implementação de processos de negócios e Web services.
- Explica como criar Web services seguros.
- Fornece uma aplicação exemplo genérico que pode ajudar o desenvolvedor a conhecer melhor os produtos, recursos, personalizações e serviços do Windows Server System.

A chave para atingir esses objetivos é a aplicação exemplo que acompanha este guia. Essa aplicação exemplo não apenas demonstra esses objetivos, mas também incorpora os seguintes processos e tecnologias de negócios:

- Fluxo de pedidos e gerenciamento de pedidos
- Troca segura de documentos de transações de negócios
- Processos B2B
- Correlação de XLANG schedules
- Web services e Microsoft Visual Studio .NET
- Integração com o Windows Server System.

Embora o enfoque seja no WSE e segurança, os desenvolvedores em BizTalk Server também poderão aprender muito com este guia. Você verá como integrar Web services com BizTalk Server e obterá idéias sobre como pode usar o BizTalk Server e .NET para melhorar os seus projetos pessoais.

Quem deve ler este guia?

Este guia é direcionado para desenvolvedores corporativos, arquitetos de sistema e profissionais de tecnologia da informação (TI) que já trabalham ou pensam em trabalhar com o Windows Server System.

Para obter os maiores benefícios deste guia, é preciso estar familiarizado com HTML, ASP.NET, scripts no cliente com Microsoft Visual Basic® Scripting Edition (VBScript), Microsoft Visual C#® .NET, Microsoft Visual Studio .NET e Microsoft SQL Server™. Este guia também supõe que você tenha alguma experiência em programação em XML.

Embora qualquer pessoa com formação técnica possa se beneficiar com as informações apresentadas neste guia, ele foi criado para o máximo benefício dos seguintes tipos de profissionais:

- **Desenvolvedores corporativos.** Aqueles indivíduos, em uma organização, que são responsáveis pelo projeto e a implementação de soluções no âmbito corporativo e entre organizações.
- **Arquitetos de sistema.** Aqueles indivíduos, em uma organização, responsáveis pelo planejamento e a criação de estratégias e soluções de negócios em geral.
- **Profissionais de TI.** Os indivíduos responsáveis pela instalação, a manutenção e a administração de software de uma empresa. Incluem-se nesta categoria: gerentes, webmasters, engenheiros de sistemas e administradores de bases de dados.

É importante notar, porém, que a maioria das informações apresentadas neste guia será mais útil aos desenvolvedores corporativos.

Que tecnologias são abordadas neste guia?

Durante a leitura deste guia, você vai aprender sobre as seguintes tecnologias no contexto da aplicação exemplo:

Web Services Enhancements para Microsoft .NET

O Web Services Enhancements para Microsoft .NET (WSE) fornece as implementações de segurança necessárias para assegurar a transferência segura de dados entre parceiros de negócios.

Para baixar o WSE, visite a página Web Services Enhancements for Microsoft .NET (http://go.microsoft.com/fwlink/?LinkId=17829).

BizTalk Server 2002

O BizTalk Server 2002 oferece as ferramentas e os serviços que controlam os processos de negócios e a troca de dados entre os parceiros de negócios e as aplicações. Para obter a documentação do BizTalk Server 2002, visite a página BizTalk Server Product Documentation (http://go.microsoft.com/fwlink/?LinkId=9779).

SQL Server 2000

O SQL Server 2000 oferece a base de dados que mantém o estoque da linha de montagem, informações sobre os parceiros de negócios, informações de associação entre documentos, etc. Para obter a documentação do SQL Server 2000, visite a página SQL Server Product Documentation (http://go.microsoft.com/fwlink/?LinkID=6976).

Visual Studio .NET

O Visual Studio .NET é opcional mas, para visualizar, compilar manualmente ou executar o código usado neste guia, você precisará ter o Visual Studio .NET instalado em seu computador.

Todos os componentes e Web services documentados e usados neste guia foram criados no Visual Studio .NET. Para obter a documentação do Visual Studio .NET, visite a página Visual Studio Product Documentation (http://go.microsoft.com/fwlink/?LinkId=9780).

BizTalk Server Toolkit para Microsoft .NET

Embora não usemos os assemblies incluídos no BizTalk Server Toolkit para Microsoft .NET, é altamente recomendável que você instale o toolkit e explore as diferentes formas pelas quais o BizTalk Server pode chamar e acessar Web services.

O toolkit provê a estrutura para integrar o BizTalk Server 2002 com a arquitetura Microsoft .NET Framework e com o Visual Studio .NET. Para baixar o BizTalk Server Toolkit para Microsoft .NET, visite o MSDN® (Microsoft Developer Network) em (http://go.microsoft.com/fwlink/?LinkId=20489).

Web services

Os Web services habilitam a troca de documentos entre o fabricante e o fornecedor. Para mais informações sobre Web services, visite o MSDN (http://go.microsoft.com/fwlink/?LinkId=9782).

Como este guia está estruturado?

Este guia possui sete capítulos. Para melhores resultados, leia-os em seqüência, pois cada capítulo complementa os conceitos dos anteriores e, em grande parte, segue um fluxo lógico de desenvolvimento de componentes na aplicação exemplo.

Capítulo 1, “Instalação”

Lista a aplicação e os requisitos de sistema para instalar, executar, monitorar e desinstalar a aplicação exemplo.

Capítulo 2, “Web services Seguros”

Detalha o fluxo de dados da aplicação exemplo, apresenta os conceitos por trás da criptografia e da assinatura digital e ilustra, em alto nível, como os benefícios do WSE se integram na aplicação exemplo.

Capítulo 3, “Usando o WSE no fornecedor"

Explica como as ferramentas e tecnologias do WSE são implementadas nos Web services do fornecedor.

Capítulo 4, “Usando o WSE no fabricante"

Explica como as ferramentas e tecnologias do WSE usadas para implementação dos Web services no fabricante.

Capítulo 5, "Configurando a troca de mensagens e orquestrando o processo de negócios"

Explica como usar o modelo de objetos BizTalk Messaging Configuration e o BizTalk Orchestration Designer para projetar, configurar e automatizar o processo de negócios.

Capítulo 6, "Associando a transação"

Oferece informações conceituais e códigos que detalham a associação de documentos de negócios na aplicação exemplo.

Capítulo 7, "Lições aprendidas e considerações finais"

Resume as limitações de arquitetura que se teve de aceitar e fala sobre como a aplicação exemplo pode ser aplicada em um ambiente de produção.

Segurança

Neste guia, trocamos documentos entre parceiros de negócios usando Web services seguros no mesmo servidor. É importante notar que, em um verdadeiro ambiente de produção, o fabricante e o fornecedor estariam em servidores diferentes.

Na aplicação exemplo, o fabricante possui o certificado que contém sua própria assinatura e chave de criptografia, e tem uma cópia da assinatura do fornecedor sem a chave privada deste, e vice-versa. Isso é, obviamente, uma prática de segurança sensata, e é feita de forma que apenas o produtor possua a chave privada para documentos enviados ao fornecedor e que apenas o fornecedor possa descriptografar os documentos.

Recursos de segurança do BizTalk Server

Embora o enfoque deste guia seja o WSE, você gostará de saber que o BizTalk Server possui uma série de funções de segurança incorporadas, oferecendo a estrutura para as empresas trocarem dados de forma segura com seus parceiros de negócio. Por exemplo, o BizTalk Server tira proveito dos seguintes recursos de segurança baseados na segurança do Microsoft Windows® 2000:

- Segurança de conta e diretivas locais do Windows
- Segurança de logon integrada Windows e SQL Server
- Segurança baseada em papéis do COM+
- Infra-estrutura de chave pública (PKI, Public Key Infrastructure)
- CryptoAPI
- Smart Cards
- Protocolo Kerberos

O BizTalk Server controla o acesso ao SQL Server com base na segurança integrada desse produto. Com essa segurança integrada, o SQL Server confia na autenticação do Windows para conceder acesso aos recursos do SQL Server. Por padrão, todas as contas que necessitam de acesso a uma das bases de dados SQL Server exigidas pelo BizTalk Server precisam ter logon integrado do SQL Server e ter direito de acesso à base de dados.

O BizTalk Server controla a segurança entre parceiros de negócio através de PKI e de S/MIME (Secure Multipurpose Internet Mail Extensions). Trocando certificados de chave pública e privada, os parceiros de negócio podem autenticar um ao outro e criptografar a comunicação através de mensagens X.509 assinadas digitalmente e S/MIME (ou criptografia personalizada usando seus próprios componentes).

Como o BizTalk Server tira proveito do SSL (Secure Sockets Layer) do Windows 2000, os parceiros de negócio podem criar e usar páginas web para trocar dados de forma segura na Internet. O SSL, que é implementado no Internet Information Services (IIS), é um protocolo que provê privacidade entre um cliente web e um servidor web. O protocolo começa com uma fase de handshake que negocia um algoritmo de criptografia, verifica as chaves (pública e privada) e autentica o servidor para o cliente. Depois de completado o handshake e iniciada a transmissão de dados da aplicação, todos os dados são criptografados usando as chaves de sessão negociadas durante o handshake. O suporte a padrões PKI abertos e protocolos seguros, como IPSec, L2TP, SSL/TLS e S/MIME, permite que uma rede seja estendida aos fornecedores e parceiros rapidamente, enquanto protege contra impostores, roubo de dados e hackers.

Para mais informações sobre o BizTalk Server e a segurança de seu ambiente, consulte a Ajuda do BizTalk Server 2002.

Continue pensando em segurança

Você deve suplementar qualquer informação fornecida em nossos guias com outras, não apenas sobre a segurança de Web services, mas também sobre segurança em geral. Como ponto de partida, sugerimos que você leia Writing Secure Code, de Michael Howard e David LeBlanc, e visite o site Microsoft Security (http://go.microsoft.com/fwlink/?LinkId=9791).

CAPÍTULO 1

Instalação

O assistente de instalação do “Implementing Secure Business Transactions Across Organizations” requer que você possua direitos de administrador e seja membro do grupo Administradores do BizTalk Server no computador em que pretende executar o assistente. Além disso, antes de executar o assistente, deve instalar as ferramentas e tecnologias listadas abaixo. É altamente recomendável instalar essas ferramentas na ordem apresentada.

Revisando os requisitos de hardware

Para executar o componente fabricante ou fornecedor da aplicação exemplo, o computador deve atender aos seguintes requisitos de hardware:

- Processador compatível com Pentium de 400 MHz (megahertz) ou superior
- 256 MB (megabytes) de RAM
- Disco rígido de 6 GB (gigabytes)
- Unidade de CD-ROM
- Adaptador de rede
- Monitor VGA ou Super VGA
- Microsoft® Mouse ou dispositivo apontador compatível

Revisando os requisitos do assistente de instalação

Esta seção descreve os pré-requisitos de software para a instalação da aplicação exemplo, os procedimentos técnicos para a instalação dessa aplicação e como executá-la para ilustrar o fabricante enviando um pedido de compra ao fornecedor e este retornando uma fatura.

Se o seu computador atender a esses requisitos, você estará pronto para executar o assistente de instalação.

> ***Alerta***
> *Caso você não tenha os softwares necessários instalados antes de executar o assistente de instalação, o processo de instalação não terá êxito e a aplicação exemplo não funcionará..*

Windows XP Professional ou Windows 2000 Service Pack 3 com NTFS

Para ajudar a garantir a segurança de seu computador, instale as mais recentes atualizações do Microsoft Windows® visitando o site Windows Update (http://go.microsoft.com/fwlink/?linkid=9609) e seguindo as instruções de instalação on-line.

Microsoft Internet Information Services 5.0 ou posterior

Para ajudar a minimizar a vulnerabilidade de seu computador, instale o IIS de forma segura (principalmente quando o instalar em computadores com Windows 2000). Para ajudá-lo nessa tarefa, fornecemos os seguintes procedimentos que descrevem como instalar o IIS 5.0 (ou posterior) em computadores com Microsoft Windows XP ou Windows 2000.

> ***Nota***
> *Ao instalar o IIS em um computador com Windows 2000, instale o Windows 2000 SP3 antes do IIS.*

Para executar o IIS 5.0 (ou posterior) de forma segura em um computador com Windows XP ou Windows 2000 SP3

1. Abra o Painel de Controle e clique duas vezes em **Add or Remove Programs.**
2. Clique em **Add/Remove Windows Components.**
3. No Windows Components Wizard, na lista de componentes, selecione a caixa **Internet Information Services (IIS)** e clique em **Next.** Siga as instruções apresentadas na tela para concluir o processo de instalação.
4. No Microsoft Internet Explorer, visite o site Microsoft Windows Update (http://go.microsoft.com/fwlink/?linkid=9609) e siga as instruções on-line para instalação.

Message Queuing

Para executar a aplicação exemplo, você precisa ter o Message Queuing (também chamado MSMQ) instalado no computador.

Para instalar o Message Queuing

1. Abra o Painel de Controle e clique duas vezes em **Add or Remove Programs.**
2. Clique em **Add/Remove Windows Components.**
3. No Windows Components Wizard, selecione a caixa **Message Queuing** e clique em **Next.** Siga as instruções apresentadas na tela para concluir o processo de instalação.

SQL Server 2000 com SP3

É necessário o Microsoft SQL Server™ 2000 com Service Pack 3 (SP3). Instale a Developer Edition nos computadores com Windows XP e a Enterprise Edition nos computadores com Windows 2000 SP3. Para mais informações sobre o SQL Server 2000, visite o site do produto (http://go.microsoft.com/fwlink/?LinkId=6791).

Microsoft .NET Framework e Microsoft Visual Studio .NET

É necessário o Microsoft .NET Framework 1.0. O Visual Studio .NET é opcional mas, para visualizar, compilar manual ou automaticamente, ou executar o código usado neste guia, você precisará ter o Visual Studio .NET instalado em seu computador. Para mais informações sobre o Visual Studio .NET, visite o site do produto (http://go.microsoft.com/fwlink/?LinkId=12161).

Microsoft Visio 2002

Para usar o BizTalk Server Orchestration Designer, você precisará ter o Microsoft Visio® 2002 instalado em seu computador. Ao executar a instalação, aceite os padrões. Para mais informações sobre o Visio, visite o site do produto (http://go.microsoft.com/fwlink/?linkid=9785).

Microsoft BizTalk Server 2002

É necessário o Microsoft BizTalk® Server 2002. Para mais informações sobre o BizTalk Server, visite o site do produto (http://go.microsoft.com/fwlink/?linkid=6959).

Para obter a versão de avaliação do BizTalk Server, visite o site de avaliação do BizTalk Server (http://go.microsoft.com/fwlink/?linkid=9788).

> ***Importante***
> *Antes de instalar o BizTalk Server, é altamente recomendável ler o item "Understanding Security" na documentação do BizTalk Server 2002.*

Para executar a aplicação exemplo, você precisará ter o BizTalk Server instalado no computador. Para instruções de instalação completas, consulte a Ajuda do BizTalk Server 2002. Ao executar a instalação, aceite os padrões.

> ***Nota***
> *É recomendável executar o assistente de instalação do "Implementing Secure Business Transactions Across Organizations" em uma instalação limpa do BizTalk Server em um ambiente que não seja de produção.*

Microsoft BizTalk Server Toolkit para Microsoft .NET

Conforme mencionado na Introdução, nossa aplicação exemplo não usa os assemblies incluídos no BizTalk Server Toolkit para Microsoft .NET; porém, o toolkit é recomendado para estender e aprimorar a aplicação exemplo.

O toolkit fornece os assemblies e o código fonte para os *runtime callable wrapper*s (RCWs, ou código executável responsável pela interface entre componentes) que você precisará para executar Web services com o componente de integração de aplicações (AIC, application integration component) do BizTalk Server com o .NET Framework. Para baixar o BizTalk Server Toolkit para Microsoft .NET, visite o MSDN® (Microsoft Developer Network) em (http://go.microsoft.com/fwlink/?LinkId=20489).

Para usar o toolkit, você precisa adicionar os arquivos RCW e os arquivos PIA (primary interop assembly, ou conjunto de módulos de interoperabilidade primária) ao cache de assembly global do Microsoft .NET.

Para adicionar os arquivos RCW e PIA ao cache de assembly global

1. Entre na pasta **C:\Program Files\Microsoft BizTalk Server\NET Toolkit.**
2. Execute **setup_toolkit.cmd.**

Web Services Enhancements para Microsoft .NET versão 1.0 SP1

O Web Services Enhancements para Microsoft .NET (WSE) é um software development kit (SDK) que permite ao cliente alavancar seu investimento em Web services, fornecendo suporte para diversos cenários centrais de Web services, incluindo:

- Segurança baseada em mensagens (conforme definido em WS-Security)
- Roteamento e independência de topologia da rede
- Arquivos anexos em mensagens SOAP

Para instalar o WSE

- Visite o página do Web Services Enhancements for Microsoft .NET no MSDN (http://go.microsoft.com/fwlink/?LinkId=17829) e siga as instruções.

Executando o assistente de instalação

Depois de confirmar que seu computador atende aos requisitos de hardware e software, você está pronto para executar o assistente de instalação do “Implementing Secure Business Transactions Across Organizations”.

Para executar o assistente de instalação do “Implementing Secure Business Transactions Across Organizations”

- Na página de resumo do guia *Implementing Secure Business Transactions Across Organizations*, clique no link **Sample Code** e siga as instruções apresentadas na tela para fazer o download do código fonte da aplicação exemplo associado a este guia.

Revisando as ações do assistente de instalação

- O assistente de instalação do *"Implementing Secure Business Transactions Across Organizations"*:
- Instala o código fonte da aplicação exemplo usado neste guia.
- Atualiza o cache de assembly global com um assembly de interoperabilidade primária.
- Anexa as de bases de dados ao SQL Server 2000.
- Cria os diretórios virtuais necessários no IIS para executar a aplicação exemplo.
- Configura o Message Queuing.
- Copia arquivos de exemplo do BizTalk Server, como esquemas e XLANG schedules.
- Configura o BizTalk Messaging Services.

> ***Nota***
> *Por razões tanto de segurança quanto de desempenho, um ambiente de produção verdadeiro requer SQL Server e IIS rodando em computadores diferentes. Para mais informações sobre como instalar o SQL Server 2000, visite a página SQL Server 2000 Books Online (http://go.microsoft.com/fwlink/?LinkID=6965). Para informações detalhadas sobre como instalar e configurar a segurança do IIS, visite o site do produto (http://go.microsoft.com/fwlink/?LinkID=6967).*

As seções a seguir listam os diretórios que o assistente de instalação instala no sistema.

Código fonte e arquivos de exemplo

O assistente de instalação instala o código fonte, os arquivos exemplo do BizTalk Server e os arquivos executáveis na unidade de instalação do BizTalk Server, o diretório C:\Program Files\Microsoft WSS Books\eBusiness\Secure Business Transactions\Development, por padrão. Os arquivos são instalados em subdiretórios conforme mostram as tabelas a seguir.

Tabela 1.1 Arquivos no subdiretório \Certificate Manager

Diretório	Descrição dos arquivos
\Certificate Manager	Aplicação usada para registrar certificados

Tabela 1.2 Arquivos no subdiretório \Databases

Diretório	Descrição dos arquivos
\Databases	Bases de dados usadas tanto pelo fabricante quanto pelo fornecedor para gerenciar certificados e associar documentos

Tabela 1.3 Diretórios e arquivos no subdiretório \Manufacturer

Diretório	Descrição dos arquivos
\COM Component	Especificações de documentos usados tanto pelo fabricante quanto pelo fornecedor
\Document Specifications	Especificações de documentos usados pelo fabricante
\XLANG Schedule	XLANG schedule para o fabricante
\XML Web Service	Classes usadas para implementar o Web service e o WSE no fabricante

Tabela 1.4 Arquivos no subdiretório \Setup

Diretório	Descrição dos arquivos
\Setup	Scripts e uma aplicação para configurar a aplicação exemplo

Tabela 1.5 Diretórios e arquivos no subdiretório \Supplier\XML Web Services

Diretório	Descrição dos arquivos
\ProcessPurchaseOrder	Classes usadas para implementar o Web service de processamento de pedidos de compra e o WSE no fornecedor
\PurchaseOrderRouter	Classes usadas para implementar o Web service de roteamento de pedidos de compra e o WSE no fornecedor

Diretórios exemplo

Além disso, o programa de instalação instala os diretórios exemplo listados na tabela a seguir.

Tabela 1.6 Descrição dos arquivos nos exemplos de diretório

Diretório	Descrição do diretório
\Sample Data\Invoice Receive	Coleta faturas após a conclusão da transação
\Sample Data\PORequest Submit	Coleta solicitações de pedidos de compra a serem submetidas fornecedor
\Sample Data\WSE Trace	Coleta dados de rastreamento

Documentos copiados para WebDAV

O assistente de instalação copia as seguintes especificações de documentos para o Web Distributed Authoring and Versioning (WebDAV):

- Invoice.xml
- PORequest.xml

Objetos de troca de mensagens

Esta seção lista os objetos de troca de mensagens que o assistente de instalação instala no sistema. Definimos e explicamos esses objetos no Capítulo 5, “Configurando o troca de mensagens e organizando o processo de negócios”.

Tabela 1.7 Objetos de troca de mensagens e objetos associados

Tipo de objeto de troca de mensagens	Nome dos objetos associados
Fila privada de troca de mensagens	porequest
Organizações	Manufacturer Supplier
Definições de documento	Invoice PORequest
Envelope	PORequest_Envelope
Portas de troca de mensagens	PORequest_Port SaveInvoice Port
Canais	PORequest_Channel SaveInvoice_Channel
Função de recepção	PORequest Receive Function

Bases de dados

O assistente de instalação anexa as seguintes bases de dados ao SQL Server:

- dbManufacturerCertificates
- dbSupplierCertificates
- dbPurchaseOrders

Assemblies

O assistente de instalação instala e registra o seguinte assembly no cache de assembly global:

- Man_SendPO.dll

Diretórios virtuais

O assistente de instalação instala os seguintes diretórios virtuais no IIS:

- ProcessPurchaseOrder
- PurchaseOrderRouter
- ReceiveInvoice

Instalando e configurando certificados

Depois de executar o assistente de instalação do *"Implementing Secure Business Transactions Across Organizations"*, você precisará registrar os certificados. As seções seguintes explicam passo a passo como instalar e configurar os certificados com êxito.

> ***Nota***
> *O Windows XP não pode ser usado como autoridade de certificação.*

Configurando a autoridade de certificação

Primeiro, você precisa instalar os serviços de certificado no computador da autoridade de certificação seguindo as instruções abaixo:

Para instalar a autoridade de certificação

1. Abra o Painel de Controle e clique duas vezes em **Add or Remove Programs.**
2. Clique em **Add/Remove Windows Components.**
3. No Windows Components Wizard, selecione a caixa **Certificate Services.**
4. Selecione **Standalone Root CA** como tipo e digite o nome da autoridade de certificação, a organização, a unidade organizacional, a cidade, o estado, etc.
5. Aceite o local padrão para a base de dados de certificados e a pasta de configuração, e conclua o assistente de instalação.

> ***Importante***
> *Para evitar erros durante a instalação dos certificados no computador da autoridade de certificação, substitua o arquivo denominado certsgcl.inc. O arquivo substituto (certsgcl.inc) está nas pastas C:\Program Files\Microsoft WSS Books\eBusiness\Secure Business Transactions\bin na unidade de instalação da aplicação exemplo. Copie o arquivo para a pasta WindowsNT\System32\CertServ no Microsoft Windows 2000 Server para sobrescrever o arquivo existente.*

Adquirindo os certificados

Agora que você já configurou o computador da autoridade de certificação, é hora de estabelecer os certificados X.509 necessários para a assinatura digital e a criptografia dos pedidos de compra entre o fabricante e o fornecedor.

> ***Nota*** *Nesta aplicação exemplo, é possível usar o mesmo certificado para o fabricante e o fornecedor tanto para criptografar quanto para assinar os dados. Nas seções a seguir, supomos que você esteja usando certificados separados. Também é importante notar que o uso do mesmo certificado para o fabricante e o fornecedor não é recomendado em ambientes de produção.*

A assinatura digital garante ao fornecedor que o pedido de compra é do fabricante e de ninguém mais. A criptografia do pedido de compra garante que nenhuma outra parte que não os dois parceiros de negócio possa determinar a real mensagem enviada. Assim, antes de executar a aplicação exemplo, você precisa de certificados que tenham tanto recursos de assinatura quanto de criptografia.

Você precisará ser o administrador do computador no qual estiver criando os certificados.

Para obter os certificados

1. Execute a página de requisisão **http://localhost/certsrv.** Note que o nome do computador da autoridade de certificação pode ser localhost ou computador externo.

> ***Importante***
> *Se for outro computador, consulte a seção seguinte, "Se a autoridade de certificação for um computador externo".*

2. Solicite um certificado e clique em **Next.**

3. Selecione **Advanced Request** e clique em **Next.**

4. Selecione **Submit a certificate request to this CA using a form** e clique em **Next.**

5. No formulário, digite um nome e um endereço eletrônico (email). Além disso, verifique se a opção **Microsoft Enhanced Cryptography Provider v1.0** está selecionada nas caixa de listagem de opções de chave.

6. Selecione a opção **Use local machine store.**

Este valor é importante para garantir que todo o certificado com a chave privada seja acessável pelo Web service.

7. Clique em **Submit.**

> ***Nota***
> *Se você estiver usando um computador externo como autoridade de certificação, consulte a seção seguinte, "Se a autoridade de certificação for um computador externo" e faça os ajustes necessários..*

O computador da autoridade de certificação pode ser definido para emitir certificados imediatamente (modo imediato) ou exigir aprovação (o padrão). Se você estabelecer o modo imediato, a página web vai diretamente para a instalação do certificado depois que se clica em **Submit.** Caso contrário, deve-se ir até a ferramenta de gerenciamento do computador da autoridade de certificação e aprovar a solicitação.

8. Agora volte ao computador em que pretende executar a aplicação exemplo, execute a página **http://server/certsrv** e selecione **Check on a pending certificate** para instalar o certificado.

Repita as etapas anteriores para adquirir o segundo certificado. Porém, conforme mencionado acima, a aplicação exemplo funciona com apenas um certificado.

> ***Importante***
> *É interessante notar a hora em que os certificados foram instalados. Desta forma, você pode facilmente localizá-los para corrigir suas permissões posteriormente, durante a instalação.*

Se a autoridade de certificação for um computador externo

Se a autoridade de certificação estiver no computador local, você não precisará lidar com a questão da confiança. Entretanto, se a autoridade de certificação for um computador externo, você precisará seguir as etapas abaixo para estabelecer a autoridade de certificação de forma segura como uma raiz de segurança confiável.

Você precisará ser o administrador do computador no qual estiver criando os certificados.

Para estabelecer um computador externo como raiz de segurança confiável

1. Execute a página de requisição **http://*nomedoservidor*/certsrv.** Note que o valor de ***nomedoservidor*** é o nome do computador externo da CA.

2. Obtenha o certificado da autoridade de certificação e clique em **Next.**

3. Selecione a opção **Download the CA certificate** e salve o arquivo (certnew.cer) em um diretório conhecido.

Você acabou de obter o certificado que precisa usar quando do estabelecimento de um console de gerenciamento de certificados na seção seguinte, "Exibindo certificados".

Exibindo certificados

É útil criar um snap-in do MMC (Microsoft Management Console) para exibir, gerenciar e solicitar certificados.

Para criar um console de gerenciamento de certificados

1. Abra uma janela de comando (Run) e digite **mmc.**

É exibido um console de gerenciamento.

2. No menu **Console,** clique em **Add/Remove Snap-in.**

3. Na caixa de diálogo **Add/Remove Snap-in**, clique em **Add** e clique duas vezes em **Certificates** na lista de Add-ins disponíveis.

4. Selecione **My user account** como tipo de certificado a gerenciar e clique em **Finish**.

Agora, inclua um segundo snap-in, idêntico ao primeiro, mas selecione **Computer account** como tipo de certificado a gerenciar. O MMC final deve ser semelhante à Figura 1.1.

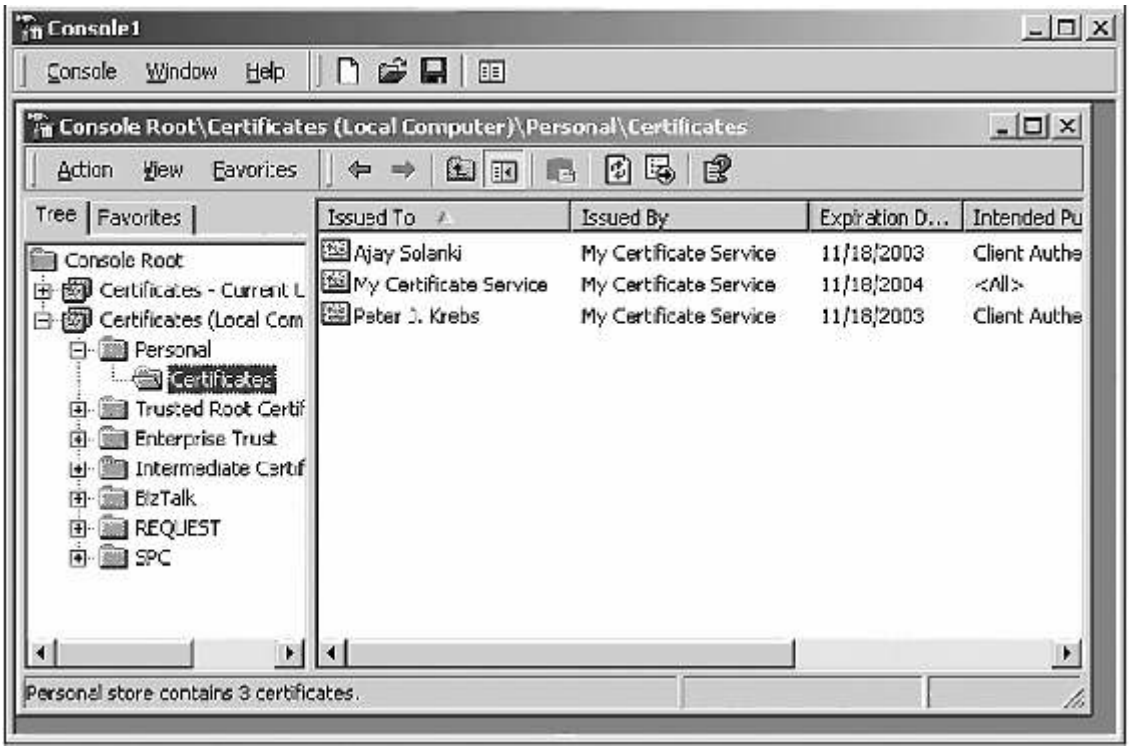

Figura 1.1 Console de gerenciamento de certificados

O console de gerenciamento de certificados recém-criado gerencia os repositórios de certificados X.509 conhecidos pelo WSE como CurrentUser Store, que é o conjunto de certificados que pertencem ao usuário conectado, para o LocalMachine store e para o computador físico.

> ***Nota***
> *Os certificados que instalamos estão no LocalMachine store (computador local), sob a pasta Personal/Certificates. Se você expandir a árvore sob o conjunto de certificados do usuário atual, talvez não encontre nenhum certificado no painel direito do MMC.*

Definindo a raiz confiável

Se você estiver usando uma autoridade de certificação que esteja em um computador externo, precisará definir uma raiz confiável. Caso contrário, não precisa completar esta etapa e pode seguir direto para "Definindo permissões".

Para definir a raiz confiável

1. Abra o console de gerenciamento de certificados.
2. Clique com o botão direito na pasta **Trusted Root Certification Authorities;** selecione, então, **All Tasks e Import.**

É exibido o assistente para importação de certificados.

3. Clique em **Next** e vá até o certificado (**certnew.cer**) que foi obtido anteriormente do computador externo da autoridade de certificação. Selecione o certificado e clique em **Next.**
4. Coloque o certificado no local padrão e clique em **Next.**
5. Clique em **Finish** para fechar o assistente para importação de certificados.

Definindo permissões

Finalmente, você precisa adicionar as permissões apropriadas para os certificados de modo que o WSE possa acessar a chave de descriptografia.

Para definir permissões

1. Abra o Windows Explorer e vá até a pasta C**:\Documents and Settings\All Users\Application Data\Microsoft\Crypto\RSA\MachineKeys**
2. Selecione os arquivos que contêm as chaves que o WSE precisa recuperar.

> ***Nota***
> *Pode ser difícil determinar que arquivo de chaves na pasta MachineKeys está associado a um certificado específico. Um método para facilitar essa identificação é notar a data e a hora de criação de um novo certificado. Ao exibir os arquivos na pasta MachineKeys, clique no campo Date Modified para ver a data e a hora correspondentes.*

3. No menu **File**, clique em **Properties**.
4. Na guia **Security**, adicione a conta **ASPNET** e selecione as opções apropriadas.

> ***Nota***
> *Selecione a opção Read para assinatura digital e as opções Read e Write para criptografia digital.*

Testando a aplicação exemplo

Nesta seção, você vai realizar as seguintes tarefas:

- Registrar os certificados.
- Submeter uma requisição de pedido de compra ao BizTalk Server para iniciar a transação.
- Exibir os dados de rastreamento do WSE para ver as mensagens criptografadas.
- Executar o XLANG Event Monitor para determinar quando o XLANG schedule começa e termina no fabricante

A conclusão dessas tarefas simples permite executar e monitorar a aplicação exemplo. Agora vejamos como fazer isso.

Registrando certificados

Você já criou os certificados, mas antes de executar a aplicação exemplo, precisa conceder permissões executando nossa aplicação exemplo denominada Certificate Manager. Você executa essa operação para selecionar, ou registrar, os certificados que deseja usar na aplicação exemplo.

Para registrar certificados

1. Na pasta **C:\Program Files\Microsoft WSS Books\eBusiness\Secure Business Transactions\Development\Certificate Manager\bin\debug,** localize e execute **Certificate Manager.exe.**

2. Na caixa de grupo C**ertificates in LocalMachine store**, na caixa de listagem, selecione um dos certificados que você criou e instalou.

3. Na caixa de grupo **Manufacturer**, clique em **Load**.

Agora a aplicação exemplo está configurada para a assinatura do fabricante e a criptografia do fornecedor. As letras exibidas nas caixas de texto são as identificações de certificado.

4. Selecione o outro certificado e, na caixa de grupo **Supplier**, clique em **Load**.

Agora a aplicação exemplo está configurada para a assinatura do fornecedor e a criptografia do fabricante. As letras exibidas nas caixas de texto são as identificações de certificado.

5. Clique em **Commit CertificateIDs** para armazenar os certificados nas bases de dados usadas pela aplicação exemplo.

A Figura 1.2 mostra a tela do Certificate Manager em execução.

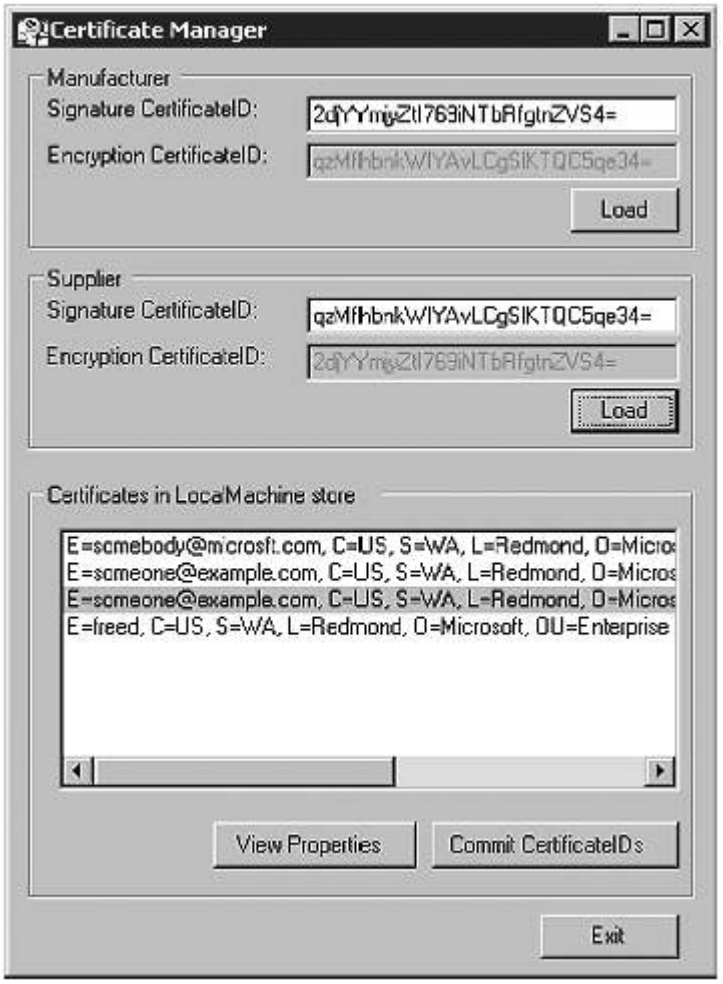

Figura 1.2 Certificate Manager gerenciador de certificados

Você também pode ir até a caixa de grupo **Certificates in LocalMachine store**, selecionar um dos certificados na caixa de listagem e clicar em **View Properties** para exibir as propriedades daquele certificado.

Executando a aplicação exemplo

Esta seção discute como executar a aplicação exemplo, exibir os dados de rastreamento do WSE e monitorar o XLANG schedule.

Para executar a aplicação exemplo

- Copie o arquivo **PORequest.txt** da pasta **C:\Program Files\Microsoft WSS Books\eBusiness\Secure Business Transactions\Sample Data** e cole-o no diretório **File receive function** que se localiza na pasta **PORequest Submit**.

> ***Nota***
> *Você pode usar também o atalho que fornecemos no menu Programs para executar a aplicação exemplo.*

A operação de cópia manual da requisição de pedido de compra de alguns itens para o diretório monitorado pela função de recepção de arquivos simula uma interface pobre com o sistema de manufatura na retaguarda da produção que determina reduções de estoque. Usando as ferramentas de monitoração, você pode ver a solicitação entrar no XLANG schedule, recuperar informações de associação na base de dados e ser enviada como pedido de compra ao fornecedor.

Então, o Web service de roteamento do fornecedor (PurchaseOrderRouter) encaminha o pedido de compra para seu Web service de processamento (ProcessPurchaseOrder), que recebe o pedido de compra, cria uma fatura e envia essa fatura ao Web service do fabricante (ReceiveInvoice).

O Web service do fabricante consulta as informações de associação na base de dados, insere a mensagem na fila apropriada do Message Queuing para recepção e assinala o fim do processo de negócio.

Exibindo os dados de rastreamento do WSE

Depois de concluída uma transação, você poderá exibir os arquivos de dados de rastreamento do WSE localizados na pasta **C:\Program Files\Microsoft WSS Books\eBusiness\Secure Business Transactions\Sample Data\WSE Trace**.

A transação cria os seguintes arquivos:

- supplier_input.xml
- supplier_output.xml
- manufacturer_input.xml

> ***Nota***
> *Por padrão, esses arquivos podem ser exibidos por qualquer um que disponha de um navegador; portanto, esses arquivos devem ser protegidos.*

Monitorando a aplicação exemplo

Pode-se usar a ferramenta XLANG Event Monitor para exibir informações de status em tempo real durante o processamento de pedidos.

> ***Nota***
> *A execução do XLANG Event Monitor é opcional.*

Para executar o XLANG Event Monitor

1. Abra o Windows Explorer e vá até a pasta **C:\Program Files\Microsoft BizTalk Server\SDK\XLANG Tools**.

2. Clique duas vezes em **XLANGMon.exe**.

Depois que a função de recepção de arquivos obtém uma requisição de pedido de compra, Schedule do fabricante aparece na pasta de schedules correntes. No final, o Schedule do fabricante se encerra e se move para a pasta completada.

Para obter informações adicionais sobre o estado atual do XLANG schedule, como que ação está sendo executada no momento, clique duas vezes no schedule.

Para mais informações sobre como usar o XLANG Event Monitor, leia o arquivo Readme.htm associado à ferramenta. Tanto o XLANG Event Monitor (XLANGMon.exe) quanto o arquivo Readme instalado pelo assistente de instalação do Microsoft BizTalk Server 2002 localizam-se na pasta C:\Program Files\Microsoft BizTalk Server\SDK\XLANG Tools.

Desinstalando a aplicação exemplo

Esta seção explica como desinstalar a aplicação exemplo.

Para desinstalar a aplicação exemplo

1. Abra o Painel de Controle e clique duas vezes em **Add/Remove Programs**.
2. Selecione **Secure Business Transaction Sample** e clique em **Remove**.

CAPÍTULO 2

Web services Seguros

Agora vamos começar a falar sobre segurança. Neste capítulo, vamos discutir, em alto nível, os Web services envolvidos na aplicação exemplo e apresentar os conceitos de assinatura e criptografia digital.

Troca de documentos com Web services

Do ponto de vista técnico, vamos implementar os sistemas do fabricante e do fornecedor usando o Microsoft® BizTalk® Server 2002 e Web Services Enhancements para Microsoft .NET (WSE). A Figura 2.1 mostra os sistemas.

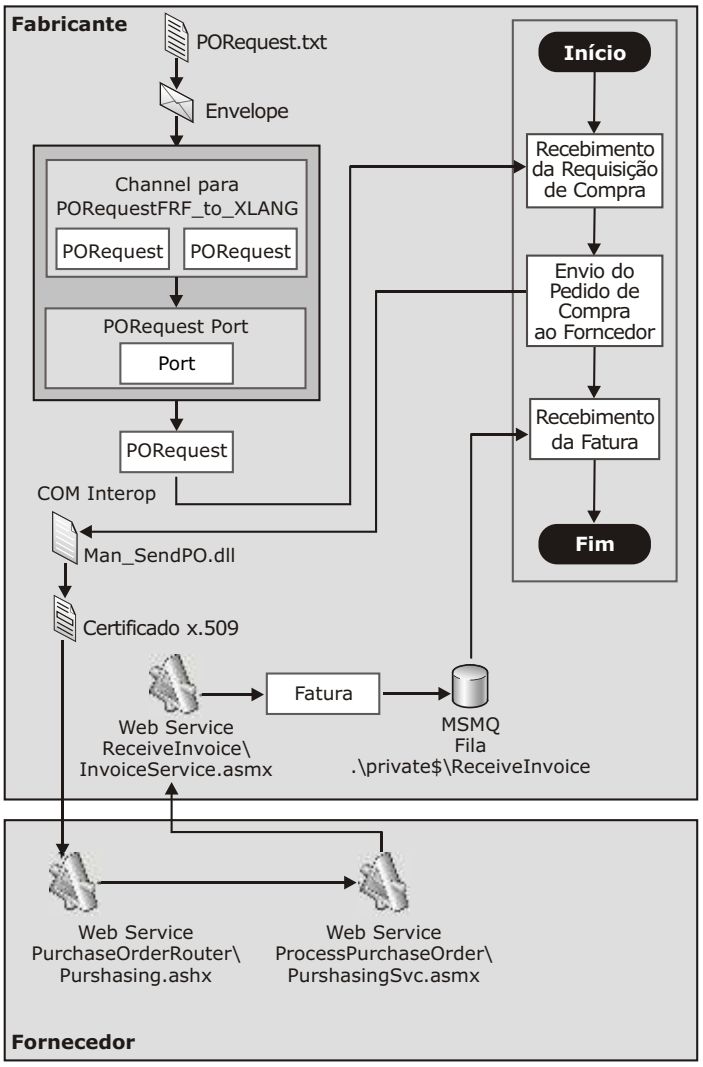

Figura 2.1 Sistemas do fabricante e do fornecedor

As seções a seguir discutem o processo no fabricante e no fornecedor.

O que acontece no fabricante?

O processo de negócios no fabricante é implementado com o BizTalk Messaging Services e o BizTalk Orchestration Services. Para enviar pedidos de compra ao fornecedor, implementamos um componente de interoperabilidade COM para poder postar a requisição para o Web service do fornecedor. Para receber a fatura do fornecedor, implementamos um Web service público no fabricante que irá associar a fatura com o pedido no XLANG schedule do processo de negócios.

O que acontece no fornecedor?

Para o fornecedor, implementamos dois Web services:

- O primeiro Web service recebe o pedido de compra e o roteia apropriadamente.
- O segundo Web service processa o pedido de compra e envia a fatura para o fabricante.

> ***Nota***
> *Nem o BizTalk Messaging Services nem o BizTalk Orchestration Services são implementados pelo fornecedor.*

Tudo bem, mas onde se encaixa o WSE?

Nossa aplicação exemplo usa alguns recursos comuns de segurança e roteamento do WSE, mas não tenta ilustrar todos os recursos disponíveis no WSE. Vamos mostrar os seguintes recursos:

- Como assinar e criptografar digitalmente a mensagem SOAP que representa o pedido de compra trocado entre os dois parceiros de negócio.
- Como tornar a rede mais "transparente" para que o fabricante não precise saber exatamente onde o destino final para o pedido de compra se localiza, mas apenas o Web service de roteamento do fornecedor.

Implicitamente, vamos abordar uma das mais importantes vantagens de usar o WSE e, realmente, de usar Web services com SOAP ou seja, desenvolvimento em plataformas heterogêneas. É bem provável que um ou mais fornecedores que atendam ao fabricante estejam rodando Web services em uma plataforma que não inclua o BizTalk Server.

O uso de mensagens dentro do padrão (incluindo Web services, SOAP e implementação das especificações de WS*) significa que o remetente e o destinatário podem estar em uma plataforma heterogênea e assim mesmo participar nos processos de negócios. Nem o fabricante nem o fornecedor precisam saber qual é a plataforma um do outro simplesmente funciona de forma segura!

Por que criptografar e assinar digitalmente?

Os certificados são usados para autenticar e proteger as trocas de informação em redes não-seguras, como a Internet. É possível gerenciar certificados para um usuário, computador ou serviço. A assinatura e/ou a criptografia digital da mensagem SOAP pode proteger ainda mais os Web services. A assinatura digital de uma mensagem SOAP protege um Web service habilitando um destinatário de mensagem SOAP, como um Web service, a verificar criptograficamente que uma mensagem SOAP não foi alterada desde que assinada. A criptografia de uma mensagem SOAP ajuda a proteger um Web service tornando altamente provável que apenas o Web service alvo leia o conteúdo da mensagem.

Nas próximas seções vamos detalhar os certificados X.509 e discutir como eles são usados em nossa aplicação exemplo.

Introdução aos certificados X.509

O servidor de certificados de chave pública baseado em X.509, incorporado no Microsoft Windows® 2000 Server, permite às organizações emitir certificados de chave pública para a autenticação de seus usuários sem depender dos serviços de uma autoridade de certificação comercial. Os certificados também contêm assinaturas digitais, que podem ser aplicadas a documentos e verificadas em documentos recebidos. Os certificados facilitam a criptografia, a descriptografia e a assinatura digital dos dados. A tecnologia de criptografia de chave pública é suportada para todos os documentos transmitidos por meio dos serviços de transporte do BizTalk Server.

> ***Importante***
> *O certificado não deve expirar e deve suportar assinatura digital e criptografia digital.*

O destinatário da mensagem assinada e criptografada digitalmente também deve ter acesso aos certificados relevantes (no caso de criptografia assimétrica, o destinatário precisa de uma chave privada e o remetente usa a chave pública do destinatário para criptografar; no caso de criptografia simétrica, tanto o remetente quanto o destinatário compartilham a chave secreta.

> ***Nota***
> *O manipulador do WSE executa a criptografia e a descriptografia propriamente ditas do corpo da mensagem SOAP.*

Acessando certificados X.509

Vamos agora dar uma olhada na classe (código) de apoio a segurança usada para gerenciar os certificados X.509 dentro dos Web services e do componente de interoperabilidade COM da aplicação exemplo.

Há dois métodos principais:

- **GetSecurityTokenForEncryption**
- **GetSecurityTokenForSigning**

O código para esses métodos, os Web services e o componente de interoperabilidade COM são mostrados nos capítulos seguintes. Esses métodos são idênticos, exceto pela checagem lógica para ver se o identificador (token) suporta assinatura digital (mudado para uma checagem para verificar se o identificador suporta criptografia de dados no segundo método). O componente de interoperabilidade COM chama cada método para obter um identificador de segurança do armazenamento do computador local.

Da teoria à implementação

Nos capítulos seguintes, vamos mostrar como usar as informações aprendidas neste capítulo para criar Web services que se integram com o WSE. As seções seguintes explicam como criar Web services que usam certificados X.509 tanto para criptografia digital quanto para assinatura digital.

CAPÍTULO 3

Usando WSE no fornecedor

Este capítulo descreve e fornece uma análise detalhada do código dos Web services executados no fornecedor. Esses Web services são:

- **PurchaseOrderRouter.** Roteia o pedido de compra enviado pelo fabricante.
- **ProcessPurchaseOrder.** Processa o pedido de compra enviado pelo fabricante.

O Web Services Enhancements para Microsoft .NET (WSE) oferece acesso a recursos especificados na arquitetura de Web services (também conhecida como especificações de Web services) melhorando o modelo de programação dos Web services criados com ASP.NET. Isto é, o WSE estende o modelo de programação introduzindo uma classe **SoapContext** e o roteador WSE. Os desenvolvedores usam a classe **SoapContext** como a principal forma de acessar a implementação das especificações de Web services DIME e WS-Security disponíveis no WSE.

O WSE permite a um cliente de Web service enviar mensagens SOAP a um roteador WSE. Com base nas informações contidas em seu cache de redirecionamento, o roteador WSE pode redirecionar de forma transparente uma mensagem SOAP para outro Web service, que vai processar a mensagem.

Primeiro, vamos falar sobre o Web service de roteamento, pois está na seqüência do fluxo de dados na aplicação exemplo. Em grande parte, este capítulo serve como análise detalhada da nossa aplicação exemplo; porém, vamos fornecer algumas informações básicas sobre como modificamos os arquivos fonte e sobre os tipos de projeto que você deverá criar no Microsoft® Visual Studio® .NET para replicar um cenário similar por si próprio.

Web service de roteamento

Na nossa aplicação exemplo, temos um serviço de roteamento para o fabricante (e outros clientes potenciais) para referenciação como local e destino para o envio de pedidos de compra. Esse manipulador é responsável por redirecionar solicitações SOAP específicas para outro servidor em que o Web service de processamento está localizado.

Caso o computador no qual o Web service de processamento roda precise ser colocado off-line e um novo computador seja configurado para fornecer o serviço, o fornecedor precisa apenas atualizar o manipulador de roteamento para redirecionar para um novo computador. Para criar esse mecanismo de referenciação, usamos o manipulador do WSE:

```
Microsoft.Web.Services.Routing.RoutingHandler
```

Além disso, usamos um arquivo de configuração simples que pode ser alterado durante a execução do serviço.

Configurando o serviço de roteamento

Esta seção explica como configuramos o serviço de roteamento.

Como configuramos o serviço de roteamento

1. Primeiro, criamos um projeto web vazio denominado PurchaseOrderRouter no Visual Studio .NET.

2. Em seguida, adicionamos um arquivo XML ao projeto, chamando-o de referralCache.config, e copiamos a definição (tags) abaixo:

```
<r:referrals xmlns:r="http://schemas.xmlsoap.org/ws/2001/10/referral">
 <r:ref>
  <r:for>
   <r:exact>http://localhost/PurchaseOrderRouter/Purchasing.ashx
    </r:exact>
  </r:for>
  <r:if />
  <r:go>
   <r:via>http://localhost/ProcessPurchaseOrder/PurchasingSvc.asmx
     </r:via>

  </r:go>
  <r:refId>uuid:ce91ab2b-8ff1-44cd-97d3-bcee71de29f5</r:refId>
 </r:ref>
</r:referrals>
```

Agora este arquivo XML contém a especificação de redirecionamento correta para o Web service de processamento. A sintaxe do arquivo XML possui seções para a condição da referência e a conseqüência. Basicamente, para qualquer solicitação SOAP direcionado para:

```
http://localhost/PurchaseOrderRouter/Purchasing.ashx
```

a solicitação SOAP será redirecionada para:

```
http://localhost/ProcessPurchaseOrder/PurchasingSvc.asmx
```

Um arquivo de referência pode ter condições múltiplas bem como inexatas (por exemplo, se a mensagem corresponde a um endereço de destino com prefixo http://localhost/Supplier) e uma ação para cada caso. As ações são especificadas na marca <via> como outro Web service de processamento.

> ***Nota***
> *No fim de cada ação, um identificador de roteador opcional (GUID) pode ser adicionado ao cabeçalho SOAP para que o Web service referenciado possa determinar a seção do roteador invocado. Isso pode ser útil na depuração de roteadores e Web services interconectados.*

O único outro arquivo necessário em nosso projeto era o web.config para a declaração do manipulador que usamos para implementar o roteamento das solicitações para o Web service principal do fornecedor.

Como criamos e editamos o web.config

- Adicionamos um arquivo XML ao projeto PurchaseOrderRouter, chamando-o de web.config, e copiamos as definições abaixo:

```
<configuration>
 <configSections>
  <section name="microsoft.web.services"   type="Microsoft.Web.Services.Configuration.WebServicesConfigura
  tion, Microsoft.Web.Services, Version=1.0.0.0, Culture=neutral,   PublicKeyToken=31bf3856ad364e35" />
 </configSections>

<system.web>
```

```
 <httpHandlers>
  <add verb="*" path="Purchasing.ashx"   type="Microsoft.Web.Services.Routing.RoutingHandler,
Microsoft.Web.Services, Version=1.0.0.0, Culture=neutral,   PublicKeyToken=31bf3856ad364e35" />
 </httpHandlers>
</system.web>
 <microsoft.web.services>
  <diagnostics>
   <trace enabled="false" />
  </diagnostics>

  <referral>
   <cache name="referralCache.config" />
  </referral>
 </microsoft.web.services>
</configuration>
```

Neste ponto, este arquivo XML contém a declaração do manipulador de roteamento e a referência para o arquivo referralCache.config. A primeira referência inclui o assembly do WSE para a implementação do manipulador HTTP.

Declaramos que todas as solicitações para Purchasing.ashx deverão ser manipuladas pela seguinte classe:

```
Microsoft.Web.Services.Routing.RoutingHandler
```

Poderíamos ter especificado que as solicitações a um ou mais destinos fossem manipuladas por manipuladores específicos. Finalmente, declaramos que as regras de roteamento para esse manipulador estão definidas no arquivo referralCache.config.

Web service de processamento

Agora que você viu como criamos e configuramos o Web service de roteamento, vamos falar sobre o código por trás do Web service de processamento. Esta seção explica e discute o Web service de processamento, que recebe uma mensagem SOAP contendo o pedido de compra do fabricante. Este Web service usa o WSE para fornecer uma camada de segurança sobre a mensagem SOAP incluindo tanto assinatura digital para o documento como criptografia digital, usando a chave pública do certificado X.509 do fornecedor.

Agora vamos mostrar como criamos a funcionalidade básica do Web service de processamento e, em seguida, nas seções seguintes, como adicionamos os componentes do WSE.

Criando ProcessPurchaseOrder

Para implementar o Web service do fornecedor para receber pedidos de compra, desenvolvemos um serviço para receber a mensagem SOAP a classe serializada que representa o pedido de compra do fabricante. Este Web service processa o pedido de compra e envia uma fatura ao Web service do fabricante.

Como criamos um Web service em C# para processar pedidos de compra

1. Primeiro, criamos um projeto de Web service ASP.NET em C# no Visual Studio .NET e o denominamos ProcessPurchaseOrder.

2. Em seguida, clicamos com o botão direito no título do projeto, clicamos em **Add** e selecionamos **Add Web Service**, chamando o novo Web service de **PurchasingSvc**.

   Este procedimento criou o novo arquivo .asmx e adicionou o código ao projeto. Dentro do novo arquivo, implementamos nossa lógica de negócios em um método de Web services definido dentro da classe de Web service denominada PurchasingSvc.

3. Finalmente, clicamos com o botão direito em **PurchasingSvc.asmx** e selecionamos **Set As StartUp Page**.

> ***Nota***
> *Para limpar o projeto, clicamos com o botão direito em Service1.asmx na árvore do projeto e depois clicamos em Delete.*

Depois de criar o Web service, adicionamos o código necessário para processar os pedidos de compra.

Criando um método no Web service denominado Process

Em seguida, criamos um método no Web service denominado **Process** em PurchasingSvc.asmx.cs que retorna vazio (void). Desta forma, o método no Web service Process é assíncrono. Para fazer isso, marcamos o método como OneWay, indicando que o método de Web service não deveria esperar um valor de retorno antes de continuar com seu processamento.

```
[WebMethod]
[SoapDocumentMethod(OneWay=true,
ParameterStyle=SoapParameterStyle.Bare)]
```

> ***Nota***
> *A propriedade SoapDocumentMethod informa ao cliente para não esperar pelo método de Web services para finalizar; caso contrário, o cliente esperará pelo retorno do método de Web service. Você pode definir esta propriedade com segurança, pois o usuário do Web service não tem de esperar que a descriptografia aconteça.*

Solicitando o contexto SOAP (o documento e os cabeçalhos da mensagem SOAP), o manipulador WSE declarado no arquivo web.config realiza a descriptografia e a validação da assinatura digital. O certificado é incorporado à mensagem SOAP para que o manipulador possa localizá-lo no armazenamento do computador. Se quiséssemos, poderíamos ter enumerado todos esses identificadores de segurança (como para a assinatura digital) usando a coleção **Security.Tokens** do contexto.

O WSE descriptografa o pedido de compra recebido e verifica a assinatura digital. Então, o método no Web service **Process** usa um argumento da classe **PurchaseOrder** que contém o pedido de compra descriptografado e assinado para o fornecedor e cria uma fatura que é enviada ao Web service do fabricante.

```
public void Process(PurchaseOrder purchaseOrder)
{
   SoapContext receiveContext = (SoapContext)
   this.Context.Items["RequestSoapContext"];
```

Nota
Para recuperar o contexto SOAP, poderíamos ter usado a seguinte linha de código:

```
SoapContext receiveContext = HttpSoapContext.RequestContext;
```

O código a seguir assegura que existe um contexto SOAP. Esta etapa confirma que o WSE está instalado e irá identificar uma mensagem não-SOAP.

```
   if(receiveContext == null)
   {
      throw new ApplicationException("Web Method requires
      Encryption");
   }
```

Note que também precisamos da assinatura do remetente. O método de Web services Process extrai a assinatura com o seguinte código:

```
   X509SecurityToken sender_signature =      ExtractSignature(receiveContext);

   if ( sender_signature == null)
   {
      throw new ApplicationException("Web Method requires a sender
      signature");
   }
```

Nota
Neste ponto, você deve verificar a assinatura para determinar o que foi digitalmente assinado.

Agora vamos analisar o corpo do método no Web service Process que cria a fatura a partir do conteúdo do pedido de compra. O código abaixo preenche a fatura com base no pedido de compra:

```
processpurchaseorder.localhost.Invoice invoice = new
processpurchaseorder.localhost.Invoice();
invoice.LineItems = new
processpurchaseorder.localhost.LineItem[purchaseOrder.LineItems.Count()];
invoice.PurchaseOrderId = purchaseOrder.PurchaseOrderId;

int index;
IEnumerator enumerator;
for (index = 0, enumerator =
   purchaseOrder.LineItems.GetEnumerator();
   index < purchaseOrder.LineItems.Count(); index++)
{
   purchaseOrder.LineItems.MoveNext();
   CommonLib.BusinessObjects.LineItem businessobject_lineitem =
  (CommonLib.BusinessObjects.LineItem)(enumerator.Current);
   processpurchaseorder.localhost.LineItem invoice_lineItem = new
   processpurchaseorder.localhost.LineItem();

   invoice_lineItem.Description =
   businessobject_lineitem.Description;
   invoice_lineItem.ExtendedPrice =
   businessobject_lineitem.ExtendedPrice;
   invoice_lineItem.ExtendedPriceSpecified = true;
   invoice_lineItem.LineNumber =
   businessobject_lineitem.LineNumber;
   invoice_lineItem.LineNumberSpecified = true;
   invoice_lineItem.ProductId =  businessobject_lineitem.ProductId;
   invoice_lineItem.QuantityOrdered =
   businessobject_lineitem.QuantityOrdered;
```

```
   invoice_lineItem.QuantityOrderedSpecified = true;
   invoice.LineItems.SetValue(invoice_lineItem,index);
}
```

Adicionamos o seguinte código para executar o Web service do fabricante, definindo um manipulador de proxy, que faz a chamada do método no Web service do fabricante, Process:

```
   InvoiceService invoiceSvc = new InvoiceService();
   invoiceSvc.Process(invoice);
}
```

Nota
Vamos adicionar código de criptografia ao método de Web services Process adiante nesta seção. Aqui, queremos apenas mostrar a você como construir o método de Web services.

Adicionando classes auxiliares a ProcessPurchaseOrder

Para o método de Web services **Process** ser compilado, precisamos adicionar a classe auxiliar **PurchaseOrder** ao projeto de Web service de processamento.

Como adicionamos classes auxiliares

1. Primeiro abrimos o Windows® Explorer e copiamos a pasta **Program Files\Microsoft WSS Books\eBusiness\Secure Business Transactions\Development\Supplier\XML Web Services\ProcessPurchaseOrder\BusinessObjects** para a pasta de projeto.

2. Em seguida, usando o Solution Explorer do Visual Studio .NET, selecionamos **Show All Files**, clicamos com o botão direito na pasta **BusinessObjects** e clicamos em **Include in Project**.

> ***Nota***
> *Se analisarmos esses objetos de negócios incluídos (classes), haverá dois grupos: os arquivos que especificam os objetos serializados (faturas, pedidos de compra, itens de linha, etc.) e a classe de segurança (para o gerenciamento de certificados).*

Neste ponto, escrevemos o Web service de processamento e precisamos criar um cliente para esse serviço.

Criando um cliente para ProcessPurchaseOrder

Esta seção discute como criamos o componente de interoperabilidade COM usado como cliente para o Web service de processamento. Depois de criar o componente, nós o conectamos ao XLANG schedule do fabricante.

Como criamos o cliente

- Criamos um projeto de biblioteca de classes C# no Visual Studio .NET e o denominamos MAN_SendPO.

Primeiro, definimos a interface ISendPO. Depois, asseguramos que a classe SendPO implementasse essa interface.

> ***Nota***
> *A implementação de uma interface facilita a ligação do componente de interoperabilidade COM com o XLANG schedule.*

```
public interface ISendPO
{
   void ProcessInventory(string strDocument, string instance,
   string queueid);
}
```

Depois de definir a interface ISendPO, definimos a classe SendPO, que implementa ISendPO:

```
   public class SendPO : ISendPO
   {
      public SendPO(){}
      public void ProcessInventory(string strDocument, string
       instance, string queueid)
```

> ***Nota***
> *Aqui, omitimos o corpo do código para o método ProcessInventory, mas discutiremos os detalhes do código nas seções seguintes.*

Adicionando uma referência web

Agora precisamos adicionar uma referência web diretamente ao Web service de processamento no fornecedor.

Como adicionamos uma referência web

1. Primeiro clicamos com o botão direito na pasta **References,** no Solution Explorer, e clicamos em **Add Web Reference.**

2. Na caixa de diálogo A**dd Web Reference**, digitamos o URL correto para o Web service de processamento:

```
http://localhost/ProcessPurchaseOrder/PurchasingSvc.asmx
```

3. Então, clicamos em **Add Reference** para gerar o código no nosso projeto.

O próximo passo é adicionar uma declaração "using" a man_sendPO.cs no projeto de biblioteca de classes MAN_SendPO para que pudéssemos referenciar o Web service de processamento no fornecedor.

```
using Man_SendPO.localhost;
```

Habilitando a interoperabilidade COM

Em seguida, precisamos habilitar a interoperabilidade COM para permitir ao Microsoft BizTalk® Server acessar a classse **SendPO.** Para fazer isso, adicionamos a seguinte declaração "using", que nos permitiu atribuir um GUID à classe **SendPO.**

```
using System.Runtime.InteropServices;
```

Agora pudemos adicionar um GUID exclusivo à interface ISendPO:

```
[Guid("BF399900-5A46-45c4-AB19-83A4B47C2260")]
```

e pudemos adicionar um GUID exclusivo à classe SendPO:

```
  [Guid("104B9773-1070-4cd9-9B41-5D26D941CDEA")]
```

Finalmente, registramos o componente para interoperabilidade COM.

Como habilitamos a interoperabilidade COM

1. Primeiro clicamos com o botão direito no projeto, no Solution Explorer, e clicamos em **Properties**.

2. Na caixa de diálogo **Property Pages**, clicamos duas vezes nas pasta **Configuration Properties** e definimos **Register for COM interop** como **True**.

Adicionando uma chave de nome forte (strong name key)

Além disso, sabemos que a classe necessita de um nome forte para poder ser carregada no cache de assembly global. O atributo de nome forte (strong named attribute) é necessário no componente para que este seja registrado junto aos serviços COM. Assim, modificamos as seguintes linhas em AssemblyInfo.cs:

```
[assembly: AssemblyKeyFile(@"..\..\sendpo.snk")]

[assembly: AssemblyKeyName("")]
```

Como precisamos de um arquivo de chave de nome forte, abrimos um prompt de comando em .NET, fomos até a pasta de projeto e criamos uma chave assinada para o assembly digitando:

```
sn k sendpo.snk
```

Adicionando código para executar ProcessPurchaseOrder a partir do cliente

Adicionamos o código a seguir ao arquivo man_sendPO.cs do componente de interoperabilidade COM, no projeto de biblioteca de classes MAN_SendPO. O código executa o Web service de processamento e gera um pedido de compra baseado no parâmetro strDocument passado para o método **ProcessInventory**.

Primeiro criamos uma ocorrência da classe **PurchaseOrder**, conforme mostra o código a seguir:

```
PurchaseOrder po = new PurchaseOrder();
```

Depois, criamos um novo pedido de compra:

```
po.PurchaseOrderId = new Guid(instance);
```

Em seguida, adicionamos um item de linha ao pedido de compra:

```
LineItem lineItem = new LineItem();

lineItem.Description = "xxxx";

lineItem.QuantityOrdered = 2;

lineItem.QuantityOrderedSpecified = true;

lineItem.ProductId = "yyyy";

lineItem.ExtendedPrice = 12.59m;

lineItem.ExtendedPriceSpecified = true;

lineItem.LineNumber = 1;

lineItem.LineNumberSpecified = true;

po.LineItems = new LineItem[1];

po.LineItems.SetValue(lineItem,0);
```

Finalmente, compilamos o componente.

Como compilamos o componente

- No Visual Studio .NET, clicamos em **Build** e em **Build Solution**.

A esta altura, concluímos a implementação e o registro do componente de interoperabilidade COM do fabricante para utilização no envio de pedidos de compra do XLANG schedule do fabricante para o Web service de roteamento no fornecedor. Agora revisamos nosso código e adicionamos componentes WSE no fornecedor.

Habilitando criptografia digital e assinatura digital

Esta seção ilustra como adicionamos a criptografia e a assinatura ao Web service de processamento e a seu componente de interoperabilidade COM cliente. Para fazer isso, adicionamos código tanto ao Web service de processamento quanto ao componente COM.

Adicionando descriptografia e assinatura a ProcessPurchaseOrder

Para fazê-lo, adicionamos a funcionalidade de descriptografia ao projeto de Web service ProcessPurchaseOrder.

Editando web.config

Primeiro, fazemos algumas modificações no arquivo web.config. Ou seja, adicionamos as seguintes informações de configuração:

```
<configuration>
  <configSections>
    <section name="microsoft.web.services"
    type="Microsoft.Web.Services.Configuration.WebServicesConfig
    uration, Microsoft.Web.Services, Version=1.0.0.0,
    Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
  </configSections>
```

Em seguida vem o código que ativa o WSE, que então manipula a criptografia. Para usar o WSE em um Web service criado com ASP.NET, o arquivo web.config precisa ser modificado.

Primeiro incluímos a referência ao manipulador WSE do HTTP:

```
<system.web>
...
<webServices>
  <soapExtensionTypes>
    <add type="Microsoft.Web.Services.WebServicesExtension,
     Microsoft.Web.Services, Version=1.0.0.0, Culture=neutral,
     PublicKeyToken=31bf3856ad364e35" priority="1" group="0"/>
  </soapExtensionTypes>
</webServices>
<system.web>
```

Em seguida, adicionamos a lógica específica para habilitar o rastreamento. Basicamente, definimos o atributo enabled como true tanto para rastreamento de entrada quanto de saída.

```
<microsoft.web.services>
  <diagnostics>
    <trace enabled="true" input="<path inserted here by .msi>"
    output="<path inserted here by .msi>" />
  </diagnostics>
</microsoft
```

> ***Nota***
> *O pacote de instalação msi irá ajustar os nomes de caminhos nas propriedades de entrada e de saída.*

Adicionando referência e declarações "using"

Em seguida, no projeto de Web service ProcessPurchaseOrder, adicionamos uma referência a Microsoft.Web.Services.dll.

Em seguida, no arquivo PurchasingSvc.asmx.cs, adicionamos as seguintes declarações "using":

```
using Microsoft.Web.Services;
using Microsoft.Web.Services.Security;
using Microsoft.Web.Services.Security.X509;
using Microsoft.Web.Services.Timestamp;
```

Adicionando o método ExtractSignature

Depois de adicionar as declarações "using", adicionamos código em PurchasingSvc.asmx.cs para extrair a assinatura digital do contexto SOAP. O seguinte método de suporte, **ExtractSignature**, verifica a validade das informações de segurança fornecidas pelo cliente.

A mensagem deve conter uma assinatura que assina o corpo e usa um identificador de segurança que vamos aceitar.

```
private X509SecurityToken ExtractSignature(SoapContext context)
{
```

Se não forem encontrados identificadores, rejeitamos a mensagem, conforme mostra o código a seguir:

```
    if ( context.Security.Tokens.Count == 0 )
        return null;
```

Se forem encontrados identificadores, percorremos todos os identificadores de segurança procurando a assinatura digital que assinou a mensagem, conforme mostra o código abaixo:

```
X509SecurityToken found_signature = null;

for ( int i = 0; found_signature == null && i < context.Security.Elements.Count; i++ )
{
    Signature signature = context.Security.Elements[i] as Signature;
    if ( signature != null && (signature.SignatureOptions &     SignatureOptions.IncludeSoapBody) != 0 )
    {
        found_signature = signature.SecurityToken as        X509SecurityToken;
    }
}

return found_signature;
}
```

Em seguida, adicionamos o seguinte código ao método do Web service **Process**, no arquivo PurchasingSvc.asmx.cs, para confirmar que o WSE está instalado e irá identificar uma mensagem não-SOAP.

```
if(receiveContext == null)
{
throw new ApplicationException("Web Method requires Encryption");
}
```

Além disso, recuperamos a assinatura do remetente, embora esse valor não seja usado na aplicação exemplo:

```
X509SecurityToken sender_signature =
ExtractSignature(receiveContext);
if ( sender_signature == null)
{
    throw new ApplicationException("Web Method requires a sender
    signature");
}
```

Importante
Na aplicação exemplo, não verificamos quem assinou o documento de negócios. Em um ambiente de produção, você acrescentaria um código que confirmasse o certificado assinado, determinando se ele contém o certificado correto, ou não.

Adicionando criptografia e assinatura ao cliente

Esta seção discute como adicionar criptografia e assinatura ao componente de interoperabilidade COM implementado no projeto da biblioteca de classe MAN_SendPO.

Editando Reference.cs

No projeto de biblioteca de classes MAN_SendPO, adicionamos uma referência a Microsoft.Web.Services.dll.

Em seguida, no arquivo Reference.cs, adicionamos uma declaração que faz referência a WSE:

```
using Microsoft.Web.Services;
```

> ***Nota***
> *Este arquivo (Reference.cs) localiza-se no projeto de biblioteca de classes MAN_SendPO, no Solution Explorer, sob Web References\localhost\Reference.map.*

Por padrão, o proxy do Web service é derivado de **SoapHttpClientProtocol.** Mas para usar o WSE em uma aplicação cliente de Web service, a classe de proxy precisa herdar de **WebServicesClientProtocol**; portanto, precisamos modificar o proxy do Web service para **WebServicesClientProtocol**.

A versão original do código em Reference.cs aparece desta forma:

```
public class PurchasingSvc :
System.Web.Services.Protocols.SoapHttpClientProtocol
```

Modificamos o código de modo que fique assim:

```
public class PurchasingSvc: WebServicesClientProtocol
```

Criando a classe de segurança

A esta altura, precisamos criar nossa classe de segurança no projeto de biblioteca de classes MAN_SendPO. Usamos a classe de segurança para definir nossa camada de segurança, incluindo os seguintes métodos:

- **GetSecurityTokenForEncryption**
- **GetSecurityTokenForSigning**

Vamos ver como criamos a classe de segurança no cliente:

Como criamos a classe de segurança

1. Primeiro clicamos com o botão direito no nome do projeto, no Solution Explorer, clicamos em **Add** e depois em **Add Class**.
2. Na caixa de diálogo **Add New Item**, selecionamos **Class** nos modelos disponíveis, chamamos a classe de **Security.cs** e clicamos em **Open**.

Agora temos uma classe **Security**, definida em Security.cs, no projeto de biblioteca de classes MAN_SendPO, que vamos usar para recuperar certificados do armazenamento LocalMachine.

Primeiro, adicionamos as seguintes declarações “using” a Security.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
...
using Microsoft.Web.Services;
using Microsoft.Web.Services.Security;
using Microsoft.Web.Services.Security.X509;
```

O construtor de **Security** conecta-se a uma base de dados, conforme mostra o código a seguir:

```
public class Security
{
   X509CertificateStore store;
   SqlConnection connection;

   public Security()
   {
      store =
      X509CertificateStore.LocalMachineStore(X509CertificateStore.
      MyStore);
      bool open = store.OpenRead();
      string Server = "localhost";
      string Database = "dbManufacturerCertificates";
      string _ConnectString = "Initial Catalog = " + Database +
      ";Data Source = " + Server + ";Integrated Security=SSPI;";
      connection = new SqlConnection(_ConnectString);
      try
```

```
      {
         connection.Open();
      }
      catch (Exception e)
      {
         string detail = e.Message;
         throw new ApplicationException(string.Format("Connection
         error, inner message {0}", detail),e);
      }
   }
```

A função de apoio **get_certificate_KeyIdentifier** usa uma declaração SQL para recuperar a identificação do certificado (CertificateID) de uma base de dados:

```
private string get_certificate_KeyIdentifier(string sql_command,
string type)
{
   SqlCommand myCommand = new SqlCommand(sql_command, connection);
   SqlDataReader dr = myCommand.ExecuteReader();

   string retval;
   if (dr.Read())
   {
      retval = dr.GetString(0);
   }
   else
   {
      throw new ApplicationException(string.Format("Certificate ID
      not found for {0}", type));
   }

   dr.Close();
   return retval;
}
```

O método abaixo, **GetSecurityTokenForSigning**, recupera o identificador de assinatura, conforme mostra o código a seguir:

```
public X509SecurityToken GetSecurityTokenForSigning()
{
   X509SecurityToken securityToken = null;
   string certKeyID = get_certificate_KeyIdentifier("select * from
   signing_id", "Signing");

   byte[] keyId = Convert.FromBase64String(certKeyID);
   X509CertificateCollection matchingCerts =
   store.FindCertificateByKeyIdentifier(keyId);

   if (matchingCerts.Count == 0)
   {
      throw new ApplicationException(string.Format("No matching
      certificates were found for the key ID provided for {0}.",
      "Signing"));
   }
   else
   {
      X509Certificate certificate = matchingCerts[0];
       if (certificate.SupportsDigitalSignature == false)
      {
         throw new ApplicationException("Signature certificate
         does not support DigitalSignatures");
      }
      securityToken = new X509SecurityToken(certificate);
   }

   return securityToken;
}
```

O método **GetSecurityTokenForEncryption** recupera o identificador de segurança:

```
public X509SecurityToken GetSecurityTokenForEncryption()
{
   X509SecurityToken securityToken = null;
   string certKeyID = get_certificate_KeyIdentifier("select * from
   encrypt_id", "Encryption");
   byte[] keyId = Convert.FromBase64String(certKeyID);
   X509CertificateCollection matchingCerts =
   store.FindCertificateByKeyIdentifier(keyId);

   if (matchingCerts.Count == 0)
   {
      throw new ApplicationException(string.Format("No matching
      certificates were found for the key ID provided for {0}.",
      "Encryption"));
   }
   else
   {
      X509Certificate certificate = matchingCerts[0];
      if (certificate.SupportsDataEncryption == false)
      {
         throw new ApplicationException("Encryption certificate
         does not support Encryption");
      }
      securityToken = new X509SecurityToken(certificate);
   }

   return securityToken;
}
```

> ***Nota***
> *Em um ambiente de produção, teríamos múltiplos certificados de criptografia, um para cada parceiro. Na aplicação exemplo, como só temos um parceiro, só estamos preocupados com o primeiro elemento.*

Usando a classe de segurança

Em seguida, retornamos ao arquivo mansend_po.cs no projeto de biblioteca de classes MAN_SendPO e adicionamos o código para anexar o certificado ao contexto SOAP.

> ***Nota***
> *Deve-se adicionar um certificado para criptografia e outro para assinatura.*

Primeiro, adicionamos as declarações "using" adequadas no início do arquivo mansend_po.cs:

```
using Microsoft.Web.Services;
using Microsofot.Web.Services.Security;
using Microsoft.Web.Services.Security.X509
```

Em seguida, adicionamos o seguinte código para definir as configurações de segurança do WSE no método **ProcessInventory**:

```
Security securityHelper = new Security();
X509SecurityToken signatureToken =
securityHelper.GetSecurityTokenForSigning();
X509SecurityToken encryptionToken =
securityHelper.GetSecurityTokenForEncryption();

requestContext.Security.Tokens.Add(signatureToken);
requestContext.Security.Tokens.Add(encryptionToken);
```

O código abaixo especifica o identificador de segurança com o qual assinamos a mensagem:

```
requestContext.Security.Elements.Add(new Signature(signatureToken));
```

O código abaixo especifica a criptografia da mensagem:

```
requestContext.Security.Elements.Add(new
EncryptedData(encryptionToken));
```

Então, compilamos o componente.

Como compilamos o componente

- No Visual Studio .NET, clicamos em Build e em Build Solution.

Neste ponto, nosso componente de interoperabilidade COM cliente estava se comunicando diretamente com o Web service de processamento no fornecedor e assinando e criptografando a mensagem. Agora, precisamos ativar o Web service de roteamento.

Ativando o Web service de roteamento

Para ativar o Web service de roteamento, editamos da seguinte forma o arquivo Reference.cs no projeto de biblioteca de classes MAN_SendPO:

```
public PurchasingSvc() {

// this is a direct connection to the web service

   // this.Url =     "http://localhost/processpurchaseorder/PurchasingSvc.asmx";

   // this is a connection through a router

   this.Url =     "http://localhost/PurchaseOrderRouter/Purchasing.ashx";

}
```

Comentamos nossa linha de código ativando a conexão direta e adicionamos a linha de código para ativar o Web service de roteamento.

Do fornecedor para o fabricante

A esta altura, o componente de interoperabilidade COM está criptografando e assinando sua mensagem e o Web service de processamento está descriptografando e extraindo a assinatura. Além disso, o serviço de roteamento foi ativado. A próxima etapa é fazer com que o Web service de processamento no fornecedor atue como um cliente para o Web service da fatura que o recebe, no fabricante.

CAPÍTULO 4

Usando WSE no fabricante

Este capítulo descreve e fornece uma análise detalhada do Web service de recepção executado no fabricante. As informações aqui contidas são muito semelhantes às informações apresentadas no capítulo anterior, onde adicionamos criptografia e assinatura digital ao componente de interoperabilidade COM MAN_SendPO e ao Web service de processamento.

Tendo isso em mente, vamos tratar de como construir o Web service de recepção.

Web service de recepção

Na nossa aplicação exemplo, o fabricante disponibiliza um Web service de recepção para receber faturas enviadas pelo fornecedor. Este serviço web usa o Web Services Enhancements para Microsoft .NET (WSE) para fornecer uma camada de segurança sobre a mensagem SOAP incluindo tanto assinatura digital para a fonte do documento como criptografia digital, usando a chave pública do certificado X.509 do fornecedor.

Agora vamos mostrar como criamos a funcionalidade básica do Web service de recepção e, em seguida, nas seções seguintes, como adicionamos os componentes do WSE.

Criando ReceiveInvoice

Para implementar o Web service do fabricante para receber faturas, desenvolvemos um serviço para receber faturas vindas do fornecedor.

Como criamos um Web service C# para receber faturas

1. Primeiro, criamos um projeto de Web service ASP.NET em C# no Microsoft® Visual Studio® .NET e o denominamos ReceiveInvoice.

2. Em seguida, clicamos com o botão direito no título do projeto, clicamos em **Add** e selecionamos **Add Web Service**, chamando o novo Web service de **InvoiceService**.

   Este procedimento cria o novo arquivo .asmx e adiciona o código ao projeto. Dentro do novo arquivo, implementamos nossa lógica de negócios em um método de Web services definido dentro da classe de Web service denominada I**nvoiceService**.

3. Finalmente, clicamos com o botão direito em **InvoiceService.asmx** e selecionamos **Set As StartUp Page**.

> ***Nota***
> *Para limpar o projeto, clicamos com o botão direito em Service1.asmx na árvore do projeto e depois clicamos em Delete.*

Depois de criar o Web service, adicionamos o código necessário para receber faturas.

Criando um método no Web service denominado Process

Em seguida, criamos um método no Web service denominado **Process** em InvoiceService.asmx.cs que retorna vazio (void). Desta forma, o método **Process** é assíncrono. Para fazer isso, marcamos o método como OneWay.

```
[WebMethod]
[SoapDocumentMethod(OneWay=true,
 ParameterStyle=SoapParameterStyle.Bare)]
public void Process(Invoice invoice)
{
```

O método **Process** toma um objeto de fatura serializado e, em seguida, determina a que fila de mensagem deve submetê-lo como um documento XML. O XLANG schedule apropriado, então, identifica a mensagem e continua o processo de negócios.

Adicionando classes auxiliares a ReceiveInvoice

Para o método de Web services **Process** ser compilado, precisamos adicionar a classe auxiliar **Invoice** ao projeto de Web service de recepção.

Como adicionamos classes auxiliares

1. Primeiro abrimos o Windows® Explorer e copiamos a pasta **Program Files\Microsoft WSS Books\eBusiness\Secure Business Transactions\Development\Manufacturer\XML Web Service\ReceiveInvoice\BusinessObjects** para a pasta de projeto.

2. Em seguida, usando o Solution Explorer do Visual Studio .NET, selecionamos **Show All Files**, clicamos com o botão direito na pasta **BusinessObjects** e clicamos em **Include in Project**.

> ***Nota***
> *Se analisarmos esses objetos de negócios incluídos (classes), haverá três grupos: os arquivos que especificam os objetos serializados (faturas, pedidos de compra, itens de linha, etc.), as bases de dados (para associação) e a classe de segurança (para o gerenciamento de certificados).*

Neste ponto, escrevemos o Web service de recepção e precisamos criar um cliente para esse serviço.

Criando um cliente para ReceiveInvoice

Estas seções discutem as modificações feitas no Web service de processamento no fornecedor para torná-lo um cliente para o Web service de recepção no fabricante.

Adicionando uma referência web

Para fazer isso, precisamos adicionar uma referência web ao projeto de Web service ProcessPurchaseOrder.

Como adicionamos uma referência web

1. Primeiro clicamos com o botão direito na pasta **References**, no Solution Explorer, e clicamos em **Add Web Reference**.

2. Na caixa de diálogo **Add Web Reference**, digitamos o URL correto para o Web service do fabricante:

```
http://localhost/ReceiveInvoice/InvoiceService.asmx
```

3. Então, clicamos em Add Reference para gerar o código no nosso projeto.

O próximo passo é adicionar a seguinte declaração “using” a PurchasingSvc.asmx, no projeto de Web service ProcessPurchaseOrder:

```
using processpurchaseorder.localhost;
```

Editando ProcessPurchaseOrder

No arquivo PurchasingSvc.asmx, fazemos a chamada ao Web service de recepção no fabricante implementando um manipulador de proxy e invocando o método de Web services do fabricante, **Process**, com o seguinte código:

```
    InvoiceService invoiceSvc = new InvoiceService();
    invoiceSvc.Process(invoice);
}
```

> ***Nota***
> *Vamos adicionar código de criptografia ao método de Web services Process adiante nesta seção. Aqui, queremos apenas mostrar a você como construir o método de Web services.*

A esta altura, o Web service de processamento no fornecedor já se comunica com o Web service de recepção no fabricante, mas não há criptografia. Nas seções seguintes, vamos mostrar como adicionamos criptografia e assinatura digital.

Habilitando criptografia digital e assinatura digital

Esta seção ilustra como adicionar a criptografia e a assinatura ao Web service de recepção e a seu cliente, o Web service de processamento, no fornecedor. Para fazer isso, adicionamos código tanto ao Web service de recepção quanto ao cliente.

Adicionando descriptografia e assinatura a ReceiveInvoice

Adicionando funcionalidade de descriptografia ao projeto de Web service ReceiveInvoice.

Editando web.config

Primeiro, fazemos algumas modificações no arquivo web.config. Ou seja, adicionamos as seguintes informações de configuração:

```
<configuration>
  <configSections>
    <section name="microsoft.web.services"
    type="Microsoft.Web.Services.Configuration.WebServicesConfig
    uration, Microsoft.Web.Services, Version=1.0.0.0,
    Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
  </configSections>
```

Em seguida vem o código que ativa o WSE, que então manipula a criptografia. Primeiro incluímos a referência ao manipulador HTTP do WSE:

```
<system.web>
...
<webServices>
  <soapExtensionTypes>
    <add type="Microsoft.Web.Services.WebServicesExtension,
    Microsoft.Web.Services, Version=1.0.0.0, Culture=neutral,
    PublicKeyToken=31bf3856ad364e35" priority="1" group="0"/>
  </soapExtensionTypes>
</webServices>
<system.web>
```

Em seguida adicionamos a lógica específica para habilitar o rastreamento. Basicamente, definimos o atributo **enabled** como **true** tanto para rastreamento de entrada quanto de saída.

```
<microsoft.web.services>
  <diagnostics>
    <trace enabled="true" input="<path inserted here by .msi>"
    output="<path inserted here by .msi>" />
  </diagnostics>
</microsoft.web.services>
```

Adicionando referência e declarações "using"

Em seguida, no projeto de Web service ReceiveInvoice, adicionamos uma referência a Microsoft.Web.Services.dll.

Então, no arquivo InvoiceService.asmx.cs, adicionamos as seguintes declarações "using":

```
using Microsoft.Web.Services;
using Microsofot.Web.Services.Security;
using Microsoft.Web.Services.Security.X509
```

Adicionando o método ExtractSignature

Depois de incluir as declarações "using", adicionamos código em InvoiceService.asmx.cs para extrair a assinatura digital do contexto SOAP. O seguinte método de suporte, **ExtractSignature**, verifica a validade das informações de segurança fornecidas pelo cliente.

A mensagem deve conter uma assinatura que assina o corpo e usa um identificador de segurança que vamos aceitar:

```
private X509SecurityToken ExtractSignature(SoapContext context)
{
```

Se não forem encontrados identificadores, rejeitamos a mensagem, conforme mostra o código a seguir:

```
  if ( context.Security.Tokens.Count == 0 )
    return null;
```

Se forem encontrados identificadores, percorremos todos os identificadores de segurança procurando a assinatura digital que assinou a mensagem, conforme mostra o código abaixo:

```
X509SecurityToken found_signature = null;

for ( int i = 0; found_signature == null && i < context.Security.Elements.Count; i++ )
{
  Signature signature = context.Security.Elements[i] as Signature;
  if ( signature != null && (signature.SignatureOptions &
  SignatureOptions.IncludeSoapBody) != 0 )
  {
    found_signature = signature.SecurityToken as
    X509SecurityToken;
  }
}

return found_signature;
}
```

Em seguida, adicionamos o seguinte código ao método no Web services **Process**, no arquivo InvoiceService.asmx.cs, para confirmar que o WSE está instalado e irá identificar a mensagem não-SOAP.

```
if(receiveContext == null)
{
throw new ApplicationException("Web Method requires Encryption");
}
```

Além disso, recuperamos a assinatura do remetente, embora esse valor não seja usado na aplicação exemplo:

```
X509SecurityToken sender_signature = ExtractSignature(receiveContext);
if ( sender_signature == null)
{
   throw new ApplicationException("Web Method requires a sender
   signature");
}
```

Importante
Na aplicação exemplo, não verificamos quem assinou o documento de negócios. Em um ambiente de produção, você acrescentaria um código que confirmasse o certificado assinado, determinando se ele contém o certificado correto, ou não.

Adicionando criptografia e assinatura ao cliente

Esta seção discute como adicionar criptografia e assinatura ao projeto de Web service ProcessPurchaseOrder.

Editando Reference.cs

No projeto de Web service ProcessPurchaseOrder, adicionamos uma declaração "using" em Reference.cs, que faz referência ao WSE:

```
using Microsoft.Web.Services;
```

Nota
Esse arquivo (Reference.cs) localiza-se no Solution Explorer, sob Web References\localhost\Reference.map.

Por padrão, o proxy do Web service é derivado de **SoapHttpClientProtocol.** Mas para usar o WSE em uma aplicação cliente de Web service, a classe de proxy precisa herdar de **WebServicesClientProtocol;** portanto, precisamos modificar o proxy do Web service para **WebServicesClientProtocol.**

A versão original do código em Reference.cs aparece desta forma:

```
public class InvoiceService :
System.Web.Services.Protocols.SoapHttpClientProtocol
```

Modificamos o código de modo que fique assim:

```
public class InvoiceService: WebServicesClientProtocol
```

Criando a classe de segurança

A esta altura, precisamos criar nossa classe de segurança no projeto de Web service ProcessPurchaseOrder, que usamos para definir nossa camada de segurança, incluindo os seguintes métodos:

- **GetSecurityTokenForEncryption**
- **GetSecurityTokenForSigning**

Vamos ver como criamos a classe de segurança no cliente:

Como criamos a classe de segurança

1. Clicamos com o botão direito no nome do projeto, no Solution Explorer, clicamos em **Add** e depois em **Add Class**.

2. Na caixa de diálogo A**dd New Item**, selecionamos **Class** nos modelos disponíveis, chamamos a classe de Security.cs e clicamos em Open.

Agora temos uma classe Security, definida em **Security.cs** no projeto de Web service ProcessPurchaseOrder, que vamos usar para recuperar certificados do armazenamento LocalMachine.

Primeiro, adicionamos as seguintes declarações "using" a Security.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
...
using Microsoft.Web.Services;
using Microsoft.Web.Services.Security;
using Microsoft.Web.Services.Security.X509;
```

O construtor de **Security** conecta-se a uma base de dados, conforme mostra o código a seguir:

```
public class Security
{
   X509CertificateStore store;
   SqlConnection connection;

   public Security()
   {
      store =       X509CertificateStore.LocalMachineStore(X509CertificateStore.
      MyStore);
```

```
        bool open = store.OpenRead();
        string Server = "localhost";
        string Database = "dbSupplierCertificates";
        string _ConnectString = "Initial Catalog = " + Database +
        ";Data Source = " + Server + ";Integrated Security=SSPI;";
        connection = new SqlConnection(_ConnectString);
        try
        {
            connection.Open();
        }
        catch (Exception e)
        {
            string detail = e.Message;
            throw new ApplicationException(string.Format("Connection error, inner message {0}", detail),e);
        }
    }
```

A função de apoio **get_certificate_KeyIdentifier** usa uma declaração SQL para recuperar a identificação do certificado (CertificateID) de uma base de dados:

```
    private string get_certificate_KeyIdentifier(string sql_command, string type)
    {
        SqlCommand myCommand = new SqlCommand(sql_command, connection);
        SqlDataReader dr = myCommand.ExecuteReader();

        string retval;
        if (dr.Read())
        {
            retval = dr.GetString(0);
        }
        else
        {
            throw new ApplicationException(string.Format("Certificate ID

            not found for {0}", type));
        }

        dr.Close();
        return retval;
    }
```

O método abaixo, **GetSecurityTokenForSigning,** recupera o identificador de assinatura, conforme mostra o código a seguir:

```
public X509SecurityToken GetSecurityTokenForSigning()
{
   X509SecurityToken securityToken = null;
   string certKeyID = get_certificate_KeyIdentifier("select * from
   signing_id", "Signing");

   byte[] keyId = Convert.FromBase64String(certKeyID);
   X509CertificateCollection matchingCerts =
   store.FindCertificateByKeyIdentifier(keyId);

   if (matchingCerts.Count == 0)
   {
      throw new ApplicationException(string.Format("No matching
      certificates were found for the key ID provided for {0}.",
      "Signing"));
   }
   else
   {
      X509Certificate certificate = matchingCerts[0];
       if (certificate.SupportsDigitalSignature == false)
      {
         throw new ApplicationException("Signature certificate
         does not support DigitalSignatures");
      }
      securityToken = new X509SecurityToken(certificate);
   }

   return securityToken;
}
```

O método **GetSecurityTokenForEncryption** recupera o identificador de segurança:

```
public X509SecurityToken GetSecurityTokenForEncryption()
{
   X509SecurityToken securityToken = null;
   string certKeyID = get_certificate_KeyIdentifier("select * from
   encrypt_id", "Encryption");
   byte[] keyId = Convert.FromBase64String(certKeyID);
   X509CertificateCollection matchingCerts =
   store.FindCertificateByKeyIdentifier(keyId);
```

```
    if (matchingCerts.Count == 0)
    {
        throw new ApplicationException(string.Format("No matching
        certificates were found for the key ID provided for {0}.",
        "Encryption"));
    }
    else
    {
        X509Certificate certificate = matchingCerts[0];
        if (certificate.SupportsDataEncryption == false)
        {
            throw new ApplicationException("Encryption certificate
            does not support Encryption");
        }
        securityToken = new X509SecurityToken(certificate);
    }

    return securityToken;
}
```

Nota
Em um ambiente de produção, teríamos múltiplos certificados de criptografia, um para cada parceiro. Nesta aplicação exemplo, como só temos um parceiro, só estamos preocupados com o primeiro elemento.

Usando a classe de segurança

Em seguida, retornamos ao arquivo PurchasingSvc.asmx.cs no projeto de Web service ProcessPurchaseOrder e adicionamos o código para anexar o certificado ao contexto SOAP.

Nota
Deve-se adicionar um certificado para criptografia e outro para assinatura.

Em seguida, adicionamos o seguinte código para definir as configurações de segurança de WSE no método de Web services **Process:**

```
SoapContext requestContext = invoiceSvc.RequestSoapContext;
Security securityHelper = new Security();
X509SecurityToken signatureToken =
securityHelper.GetSecurityTokenForSigning();
X509SecurityToken encryptionToken = sender_signature;
requestContext.Security.Tokens.Add(signatureToken);

requestContext.Security.Tokens.Add(encryptionToken);
```

O código abaixo especifica o identificador de segurança com o qual assinamos a mensagem:

```
requestContext.Security.Elements.Add(new Signature(signatureToken));
```

O código abaixo especifica a criptografia da mensagem:

```
requestContext.Security.Elements.Add(new
EncryptedData(encryptionToken));
```

Finalmente, compilamos o código.

Como compilar o código

- No Visual Studio .NET, clicamos em **Build** e em **Build Solution**.

Da proteção da comunicação à configuração dos parceiros de negócio

Neste ponto, todos os Web services e os clientes são criptografados, descriptografados e assinados corretamente. Agora vamos prosseguir para a configuração do fabricante e do fornecedor usados na aplicação exemplo.

CAPÍTULO 5

Configurando a troca de mensagens e organizando o processo de negócios

Neste capítulo, mostramos como construir esquemas e configurar os serviços de troca de mensagens do BizTalk para assegurar que as mensagens XML sejam roteadas corretamente através da aplicação exemplo e entre o fabricante e o fornecedor. Além disso, mostramos como usar o BizTalk Orchestration Services para coordenar os processos de negócios e salvar os documentos de negócios no fabricante.

> ***Nota***
> *Não há XLANG schedule no fornecedor..*

Configurando a troca de mensagens

Esta seção mostra como usamos o modelo de objeto BizTalk Messaging Configuration para criar programaticamente as configurações de troca de mensagens no sistema de pedidos. Os seguintes tipos de objetos de troca de mensagens estão presentes no sistema de pedidos:

- Filas privadas de troca de mensagens
- Organizações
- Definições de documento
- Envelopes
- Portas de troca de mensagens
- Canais
- Funções de recepção

Agora vamos examinar mais profundamente os objetos e o código que escrevemos para criá-los e configurá-los.

Criando os objetos de troca de mensagens

Estabelecemos a configuração de troca de mensagens usando o Microsoft® Visual Basic® Scripting Edition (VBScript).

Para exibir o arquivo .vbs

- No Windows® Explorer, vá até a pasta **\Program Files\Microsoft WSS Books\eBusiness\Secure Business Transactions\Development\Setup** e exiba o arquivo vbconfig.vbs.

Fila privada de troca de mensagens

As filas privadas de troca de mensagens são serviços usados para enviar e receber mensagens.

O código abaixo mostra a criação e a configuração da fila privada de troca de mensagens porequest, que ativa o XLANG schedule no fabricante:

```
Set objmsmqinfo = CreateObject("MSMQ.MSMQQueueInfo")


objmsmqinfo.PathName = ".\private$\porequest"
objmsmqinfo.Label = "Secure Business Transactions"
objmsmqinfo.Create True, True
```

> ***Nota***
> *Além disso, como o Microsoft BizTalk® Server cria uma fila de troca de mensagens por ocorrência para receber as faturas, uma fila privada de troca de mensagens é criada para cada ocorrência de XLANG schedule ao executar a aplicação exemplo.*

Organizações

As organizações representam unidades de negócios ou parceiros de negócio, servindo como origens e destinos para documentos.

O código abaixo mostra a criação da organização Manufacturer, que representa o fabricante:

```
Set org = nothing
...
org.clear
org.name = "Manufacturer"
org.Comments = "For WSE example"
lOrg110009 = org.Create
...
```

Além disso, criamos e configuramos a seguinte organização:

Tabela 5.1 Descrição da organização

Organização	Descrição
Supplier	Representa o fornecedor

> ***Importante***
> *Em um ambiente de produção, as duas organizações estariam em computadores separados e, como resultado, haveria duas organizações em cada computador ou cluster. Por exemplo, a rede do fornecedor teria organizações tanto para o fornecedor quanto para o fabricante.*

Definições de documento

As definições de documento representam tipos específicos de documentos processados pelo BizTalk Server.

O código abaixo mostra a criação e a configuração da definição de documento Invoice, que faz referência à especificação Invoice.xml:

```
Set doc = nothing
...
doc.clear
doc.name = " Invoice"
doc.reference = "http://" & gServerName  &
   "/BizTalkServerRepository/DocSpecs/WSS Books/eBusiness/Secure
   Business Transactions/Invoice.xml"
lDoc140016 = doc.Create
...
```

Além disso, criamos e configuramos a seguinte definição de documento:

Tabela 5.2 Descrição de definição de documento

Definição de documento	Descrição
PORequest	Faz referência à especificação PORequest.xml.

> ***Nota***
> *Na aplicação exemplo, o pedido de compra é construído programaticamente.*

Envelope

Os envelopes encapsulam dados eletrônicos de negócios para o transporte.

O código abaixo mostra a criação do envelope PORequest_Envelope, que faz referência à especificação PORequest.xml:

```
Set env = nothing
...
env.name = "PORequest_Envelope"
env.reference = "http://" & gServerName  &
   "/BizTalkServerRepository/DocSpecs/WSS Books/eBusiness/Secure
   Business Transactions/PORequest.xml"
env.format = "FLATFILE"
lEnv150002 = env.Create
...
```

É importante notar que é necessário um envelope se o tipo de documento de entrada for FLATFILE, pois então o envelope contém informações sobre como converter o documento em XML e que código (parser) deverá processar o documento. Neste caso, criamos um envelope para converter o formato de arquivo flat (da aplicação legada do fabricante) em uma solicitação de pedido de compra em XML.

Portas de troca de mensagens

As portas de troca de mensagens representam um conjunto de propriedades que você pode usar para configurar o BizTalk Messaging Services para transportar documentos até um destino específico usando um serviço de transporte especificado.

O código abaixo mostra a criação e a configuração da porta de troca de mensagens PORequest_Port, que entrega a solicitação de pedido de compra ao XLANG schedule no fabricante:

```
Set port = nothing
...
port.name = " PORequest_Port"
```

Depois de definir o nome da porta, definimos o objeto de destino:

```
port.DestinationEndPoint.Organization = lOrg110001
port.DestinationEndPoint.Alias = lAlias120001
port.DestinationEndPoint.Openness = _
   BIZTALK_OPENNESS_TYPE_EX_TOWORKFLOW
```

Então, definimos as informações de transporte para o transporte primário:

```
port.PrimaryTransport.Address = gProjectRoot & _
  "Development\Manufacturer\XLANG Schedule\MAN_Purchasing.skx"
port.PrimaryTransport.Parameter = "PORequest"
port.PrimaryTransport.Type = _
  BIZTALK_TRANSPORT_TYPE_ORCHESTRATIONACTIVATION
lPort160023 = port.Create
...
```

Além disso, criamos e configuramos a seguinte porta de troca de mensagens:

Tabela 5.3 Descrição da porta de troca de mensagens

Porta de troca de mensagens	Descrição
SaveInvoice_Port	Salva o arquivo da fatura

Canais

Os canais representam um conjunto de propriedades que podem ser usadas para configurar o BizTalk Messaging Services para processar um documento recebido.

O código abaixo mostra a criação e a configuração do canal PORequest_Channel, que processa e roteia a solicitação de pedido de compra:

```
Set channel = nothing
...
channel.name = "PORequest_Channel"
```

Depois de definir o nome do canal, definimos o objeto de destino:

```
channel.SourceEndPoint.Organization = lOrg110009
channel.SourceEndPoint.Alias = lAlias120009
channel.InputDocument = lDoc140021
channel.OutputDocument = lDoc140018
```

Então, definimos a configuração de porta no objeto dictionary:

```
Set dict = CreateObject("Commerce.Dictionary")

dict.SkedFile = gProjectRoot & "Development\Manufacturer\XLANG" & _
"Schedule\MAN_Purchasing.skx"
dict.Port = "PORequest"
```

Além disso, criamos e configuramos o seguinte canal:

Tabela 5.4 Descrição do canal

Canal	Descrição
SaveInvoice Channel	Processa e roteia a fatura

Função de recepção

Quando você precisa receber documentos de um ponto de recepção e submetê-los ao BizTalk Server, pode configurar uma função de recepção para processar os dados.

O código abaixo mostra a criação da função de recepção PORequest Receive Function, que recebe e processa a solicitação de ordem de compra do arquivo flat:

```
...
With oReceiveFunctionInstance
    .Name = "PORequest Receive Function"
    .ProtocolType = 1
    .GroupName = "BizTalk Server Group"    .FileNameMask = "*.txt"
    .PollingLocation = gProjectRoot & "Sample Data\PORequest Submit"
    .OpennessFlag = 1
    .IsPassThrough = False
    .EnvelopeName = "PORequest_Envelope"
    .ChannelName = "POReqeust Channel"
    .DisableReceiveFunction = False
    .HttpReturnCorrelationToken = False
    .ProcessingServer=gHostName
```

Em seguida, especificamos o sinalizador wbemChangeFlagCreateOnly:

```
    .Put_ (2)
End With
...
```

Agora que mostramos como criar e configurar o BizTalk Messaging Services, vamos ver como usamos esses serviços para automatizar o processo de negócios no fabricante.

Orquestrando o processo de negócios

O BizTalk Orchestration Designer é uma ferramenta baseada no Microsoft Visio® 2002 que permite criar diagramas de processos de negócios que podem ser compilados e executados como XLANG schedules. XLANG é uma linguagem baseada em XML que descreve o processo de negócio e a ligação desse processo com serviços de aplicações.

Para mais informações sobre o BizTalk Orchestration Designer, consulte a Ajuda do BizTalk Server 2002.

Processo de negócios no fabricante

Esta seção discute o processo de negócios no fabricante. A Figura 5.1 mostra o XLANG schedule MAN_Purchasing.skv.

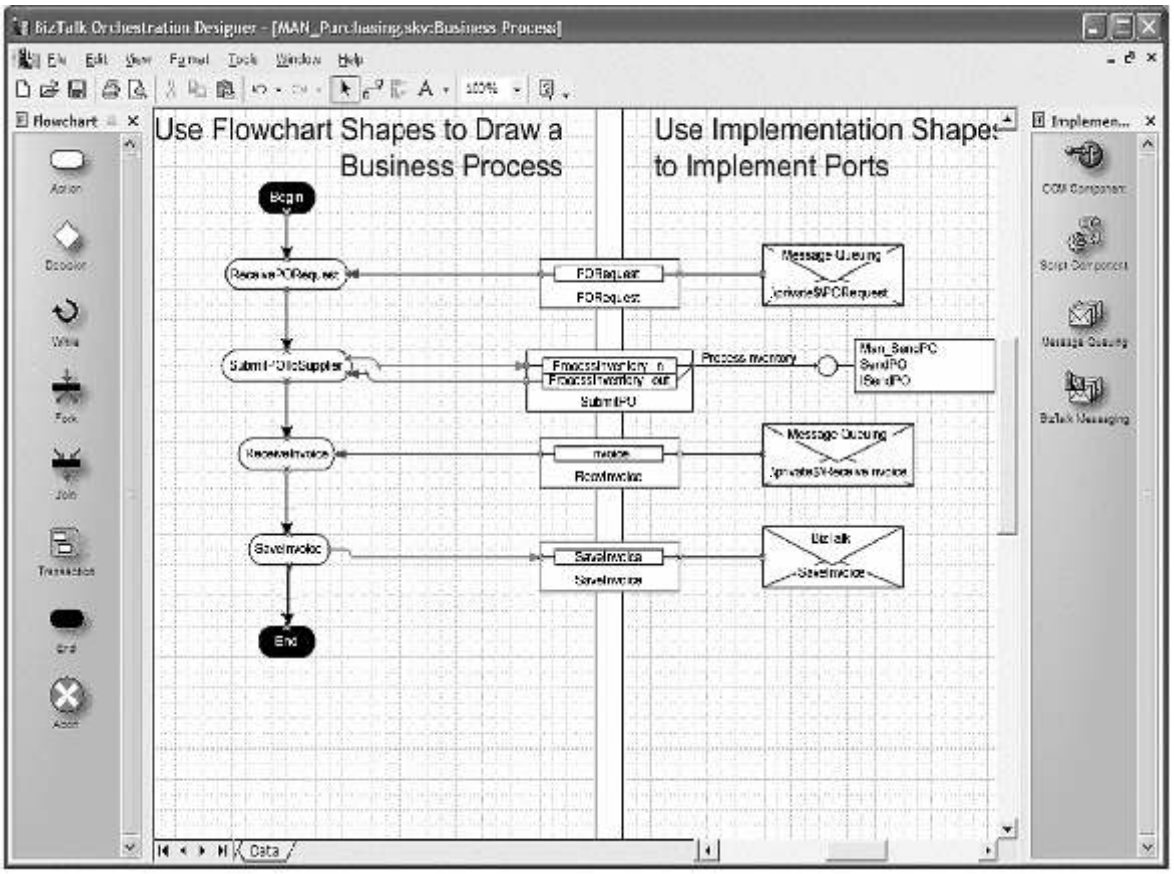

Figura 5.1 XLANG schedule no fabricante

Para exibir o processo de negócios no fabricante

1. Clique em **Start**, aponte para **Programs**, aponte para **Microsoft BizTalk Server 2002** e clique em **BizTalk Orchestration Designer**.
2. No menu **File**, clique em **Open.**
3. Vá até a pasta **Program Files\Microsoft WSS Books\eBusiness\Secure Business Transactions\Development\Manufacturer\XLANG Schedule** e clique duas vezes em **MAN_Purchasing.skv**.

As seções a seguir discutem o processo de negócios no fabricante, detalhando a seqüência de ações definidas pelo XLANG schedule do fabricante.

Ativando o XLANG schedule

Visualizando o PORequest_Port no BizTalk Messaging Manager, você vai notar que uma porta de troca de mensagens do BizTalk Server ativa o XLANG schedule do fabricante. Porém, a ação inicial definida no agenda do fabricante é ligada a um identificador ou figura representando o Message Queuing, que referencia uma fila privada estática ao invés de um identificador ou figura representando uma Mensagem do BizTalk, de forma que as filas de mensagens por ocorrência não são criadas. Fizemos isso para otimizar o desempenho do schedule, impedindo a criação, e a subseqüente eliminação, de uma fila de mensagens usada para iniciar o schedule.

Enviando a solicitação de pedido de compra

Depois que a solicitação de pedido de compra entra no XLANG schedule, este envia o pedido de compra ao fornecedor instanciando um componente de interoperabilidade COM, MAN_SendPO, que criptografa e assina os dados e, em seguida, envia a solicitação de pedido de compra ao fornecedor.

Recebendo a fatura

O XLANG schedule pára na ação ReceiveInvoice, esperando um documento de fatura de entrada. Então, o fornecedor envia a fatura e o fabricante recebe a mensagem XML da fila ReceiveInvoice.

Salvando a fatura

Quando o fabricante recebe a fatura, o XLANG schedule precisa passar os dados para a próxima ação no Schedule que é necessária para atuar sobre os dados. Nesse caso, a ação SaveInvoice obtém o documento e o salva para o fabricante.

Da configuração e organização a associação

Até agora, mostramos como proteger os Web services e como configurar o fabricante e o fornecedor para receber e enviar documentos de negócios. No capítulo seguinte, mostraremos como associar os documentos de negócios trocados entre os parceiros de negócio.

CAPÍTULO 6

Associando a transação

Agora que discutimos e detalhamos o Web Services Enhancements para Microsoft .NET (WSE) e a configuração do Microsoft® BizTalk® Server, vamos falar um pouco sobre como associamos documentos de negócios na aplicação exemplo. Quando o fabricante envia um pedido de compra ao fornecedor, este devolve uma fatura referente a esse pedido de compra específico. É possível que o fabricante envie múltiplos pedidos de compra a um ou mais fornecedores e que as faturas devolvidas cheguem em uma ordem aleatória. Nesse caso, a fatura correta deve ser combinada com o pedido de compra correto. Esse processo é chamado de associação.

Como se poderia imaginar, há diversas maneiras de fazer a associação. Nesta aplicação exemplo, o BizTalk Server cria uma fila de mensagens exclusiva para cada pedido de compra emitido. Então, o Web service de recepção (ReceiveInvoice) pega a fatura que recebe e a coloca na fila de mensagens correta. Parece bem simples, não?

Agora que você tem uma visão geral de por que fazemos a associação e como a fazemos na aplicação exemplo, vamos aos detalhes.

Entendendo o mecanismo de associação

Vamos revisar o processo de associação mas, desta vez, vamos discutí-lo um pouco mais detalhadamente. Primeiro, toda ocorrência de XLANG schedule do fabricante cria uma fila exclusiva de troca de mensagens. Quando o XLANG schedule cria um pedido de compra de saída, dois valores exclusivos são salvos em uma tabela da base de dados:

- O InstanceID do XLANG schedule.
- O nome da fila de troca de mensagens que pertence àquele XLANG schedule.

Agora o XLANG schedule espera que uma mensagem apareça em sua fila de troca de mensagens.

Quando uma fatura de entrada chega ao Web service, a identificação de ocorrência (InstanceID) é extraída do documento e a tabela da base de dados é lida para obter o nome da fila de mensagens para aquele Instance ID específico. O documento é, então, armazenado na fila de troca de mensagens correspondente. Neste ponto, a ocorrência em execução do XLANG schedule recupera o documento de sua fila de mensagens e continua o processo de negócios.

O fornecimento desses detalhes ajuda a esclarecer o mecanismo de associação, mas vamos dar uma olhada no código para entender o processo totalmente.

Codificando o mecanismo de associação

Em grande parte, o mecanismo de associação envolve dois métodos públicos:

- **CreateCorrelationInfo**
- **RetrieveCorrelationInfo**

Esses métodos são definidos no arquivo Database.cs na pasta BusinessObjects. Você pode encontrar essa pasta e os arquivos incluídos nos seguintes projetos exemplo do Microsoft Visual Studio® .NET:

- **ReceiveInvoce.** O Web service do fabricante que vai receber as faturas.
- **MAN_SendPO.** O cliente (no fabricante) para os Web services do fornecedor que roteia e processa os pedidos de compra.

Vamos analisar mais de perto esses métodos na seção a seguir.

Criando as informações de associação

Primeiro, vamos analisar o método **CreateCorrelationInfo** que escreve o identificador da fila no caminho de saída:

```
public void CreateCorrelationInfo(string skedid, string queueid, string purchaseorderid)
{
   SqlCommand command = new     SqlCommand("usp_CreateCorrelationInfo");
   command.Connection = this.Connect();
   command.CommandType = CommandType.StoredProcedure;
   command.Parameters.Add(new SqlParameter("@SkedId", skedid));
   command.Parameters.Add(new SqlParameter("@QueueId", queueid));

   command.Parameters.Add(new SqlParameter("@PurchaseOrderId",
  purchaseorderid));
   command.ExecuteNonQuery();
}
```

Como você vê, este método chama uma stored procedure do SQL que insere as informações de associação na tabela da base de dados.

Recuperando as informações de associação

Em seguida, vamos analisar o método **RetrieveCorrelationInfo**:

```
public string RetrieveCorrelationInfo(string correlation_key)
{
   string SQL_command = string.Format("select QueueId From
   tblCorrelationInfo Where PurchaseOrderId = '{0}'",
   correlation_key);
   SqlCommand myCommand = new SqlCommand(SQL_command,Connect());

   SqlDataReader dr = myCommand.ExecuteReader();

   string retval;
   if (dr.Read())
   {
      retval = dr.GetString(0);
   }
   else
   {
      retval = "";
   }
return retval;
}
```

Este método recupera o nome da fila de mensagens e especifica que será anexado ao nome completo da fila para que o documento de fatura possa ser retornado à ocorrência correta do XLANG schedule.

Implementando o mecanismo de associação

As seções seguintes explicam como os métodos abaixo são usados para associar documentos de saída e de entrada.

- **CreateCorrelationInfo**
- **RetrieveCorrelationInfo**

Para começar, vamos falar sobre a associação de saída de documentos, pois este processo inicializa os valores de associação na base de dados.

Documento de saída

O código abaixo coleta e salva as informações de associação do componente de interoperabilidade COM no fabricante (MAN_SendPO):

```
Correlation correlate = new Correlation();
correlate.CreateCorrelationInfo(instance, queueid,
po.PurchaseOrderId.ToString());
supplierProxy.Process(po);
```

Documento de entrada

O código a seguir no método do Web service ReceiveInvoice no fabricante é usado para ler a base de dados de associação para recuperar o nome da fila de mensagens:

```
CommonLib.Database.Correlation db = new
CommonLib.Database.Correlation();

string queueName =
db.RetrieveCorrelationInfo(invoice.PurchaseOrderId.ToString());
```

Então a fatura é entregue à fila de mensagens correta:

```
MessageQueue queue = new MessageQueue();

if (MessageQueue.Exists(queueName))
{
   queue.Path = queueName;
   Message msg = new Message();
```

```
        ActiveXMessageFormatter format = new
        ActiveXMessageFormatter();
        format.Write(msg, this.SerializeXml(invoice));

        if (queue.Transactional)
        {
           MessageQueueTransaction tx = new
           MessageQueueTransaction();
           tx.Begin();
           queue.Send(msg,"Invoice", tx);
           tx.Commit();
        }
        else

        {
        queue.Send(msg);
        }
     }
return;
}
```

Armazenando as informações de associação

A tabela a seguir, pertencente à base de dados dbPurchaseOrders, é usada para manter as informações de associação:

- tblCorrelationInfo

É uma tabela simples com colunas para o identificador do pedido de compra, o identificador da ocorrência do XLANG schedule e o nome da fila de mensagens. O identificador do pedido de compra é um GUID que representa a ocorrência do documento do cabeçalho de mensagem do BizTalk Server. O identificador de Schedule também é um GUID, mas não está sendo usado. O identificador da fila é o nome da fila para a qual o documento deve ser postado quando se reentra nos BizTalk Orchestration Services.

Da associação a lições aprendidas

Quase terminamos. Vimos como gerar uma solicitação de pedido de compra, configurar os serviços BizTalk Messaging Services e BizTalk Orchestration Services, construir os Web services e os componentes COM, usar o WSE e associar os documentos de negócios.

Agora leia o próximo capitulo para ver o que aprendemos durante a construção da aplicação exemplo e do trabalho com o WSE.

CAPÍTULO 7

Lições aprendidas e considerações finais

Agora que você viu como o Microsoft® BizTalk® Server é configurado, como escrever os Web services usando a funcionalidade do Web Services Enhancements para Microsoft .NET (WSE), e percorreu o código para associar documentos, vamos refletir sobre algumas decisões que tomamos e o que podemos aprender com isto.

Nossa aplicação exemplo demonstra como integrar o BizTalk Server e a funcionalidade fornecida no WSE. Porém, como a aplicação exemplo tem escopo bastante restrito, não usamos todos os recursos do WSE e também não implementamos recursos exigidos em um sistema de qualidade de produção. Esta seção descreve as limitações implícitas na aplicação exemplo e explora os pontos a considerar quando se usa esse enfoque de integração para projetar soluções técnicas.

Lições aprendidas

Esta seção discute as lições a serem aprendidas com a leitura deste guia e da análise da aplicação exemplo.

Cenário de dois computadores

Outra simplificação da aplicação exemplo envolve o fato de que implementamos nossa aplicação exemplo em um único computador, não representando de forma realista a infra-estrutura de hardware de dois parceiros de negócio distintos. Além da presença de firewalls e roteadores (hardware) em tal cenário, a aplicação exemplo teria de ser distribuído em dois servidores web separados da seguinte forma:

- Os serviços de pedido de compra residiriam na organização do fornecedor.
- O BizTalk Server e o Web service de faturamento residiria na organização do fabricante.

Do ponto de vista da segurança, o uso de assinaturas e criptografia implica que as chaves públicas e os certificados para as duas organizações sejam compartilhados. Normalmente, o ambiente de produção usa certificados de uma autoridade confiável como VeriSign e não autoridades de certificação autônomas. Nesse caso, os certificados são baixados da autoridade confiável e instalados nos servidores web e não são necessárias modificações de código para os Web services, exceto para confirmar que o certificado VeriSign é para o seu parceiro de negócio

Segurança do BizTalk Server

Deliberadamente, também não usamos nenhum dos recursos de segurança prontos do BizTalk Server para podermos ilustrar os novos recursos do WSE. Cenários B2B (business-to-business) típicos também podem usar a segurança HTTPS para proteger o canal entre as duas organizações, além da segurança mais refinada oferecida pelo WSE.

> ***Importante***
> *Esteja ciente de que a ferramenta de rastreamento SOAP não funciona com uma sessão HTTP criptografada; portanto, lembre-se de testar as soluções em um ambiente de texto puro antes de migrar para protocolos seguros.*

BizTalk Server Toolkit para Microsoft .NET

Nossa aplicação exemplo não usa os assemblies incluídos com o BizTalk Server 2002 Toolkit para Microsoft .NET. Porém, isso não deve desencorajá-lo de baixar o toolkit e trabalhar com os exemplos de aplicação fornecidos. Em vez do componente de interoperabilidade COM, poderíamos ter escrito um componente de integração de aplicações (AIC, application integration component) em .NET no BizTalk Server.

Caso você não esteja familiarizado com o toolkit, eis aqui algumas informações básicas. O BizTalk Server foi o primeiro dos produtos do Microsoft Windows Server System™ a suportar plenamente o Microsoft Visual Studio® .NET e o Microsoft .NET Framework. Os desenvolvedores podem usar o toolkit para estender os serviços do BizTalk Server bem como a funcionalidade de automação de processos de negócios deste com o Visual Studio .NET, os Web services e o .NET Framework.

O toolkit ajuda os desenvolvedores a alavancar a nova geração de padrões como SOAP e Web Services Description Language (WSDL) com as mais recentes linguagens de programação como C# e Microsoft Visual Basic® .NET, além do .NET Framework, para criar integrações sofisticadas entre aplicações e o BizTalk Server. O toolkit fornece as bibliotecas necessárias para criar AICs para o BizTalk Server a partir do Visual Studio .NET e integrar os serviços BizTalk Messaging Services e BizTalk Orchestration Services diretamente aos Web services.

Considerações finais

Conforme mencionado, não ilustramos todo o conjunto de recursos do WSE com a nossa aplicação exemplo. Determinados cenários podem pedir recursos como o uso de anexos binários em mensagens ou assinatura de nomes de usuário.

Porém, na nossa aplicação exemplo, usamos um cenário de integração comum de B2B para demonstrar a amplificação de funcionalidades do BizTalk Server quando combinado com a plataforma .NET e os Web services. À medida que os Web services crescem com a adoção de novos padrões de segurança, roteamento, referência, transações e coordenação, cresce o escopo dos problemas de negócios que podem ser atendidos de forma bem sucedida.

A nova geração de padrões de Web services integrados no BizTalk Server e na plataforma .NET oferecem um sofisticado conjunto de ferramentas para parceiros de negócio para integrar processos de negócios entre organizações de forma segura, confiável e escalável.

Microsoft®