Microsoft Dynamics® AX 2012

Using Microsoft BizTalk Server 2010 to exchange documents with Microsoft Dynamics AX

White Paper

This white paper provides information about how to configure Application Integration Framework and Microsoft BizTalk Server 2010 to enable the exchange of service documents.

Date: July 2011

www.microsoft.com/dynamics/ax

Jim Travis, Senior Writer

Send suggestions and comments about this document to adocs@microsoft.com. Please include the title with your feedback.

Microsoft Dynamics®

Table of Contents

Introduction

Microsoft Dynamics® AX 2009 included a special adapter that provided the ability to integrate with Microsoft BizTalk® Server through the Application Integration Framework (AIF). Microsoft Dynamics AX 2012 no longer includes this adapter. Instead, you can connect AIF with BizTalk by using the Windows Communication Framework (WCF)-based adapters that are included with AIF. This document discusses the general issues surrounding this type of integration and describes how to create an integration solution that uses BizTalk and AIF to exchange information.

Audience

This white paper is designed for BizTalk developers and administrators and Microsoft Dynamics AX system administrators who are responsible for installing, configuring, and troubleshooting integration with external systems using AIF.

Prerequisites

To benefit from this white paper, you should have knowledge in the following areas and working installations of the listed software:

- Enterprise application integration (EAI), business-to-business (B2B), and application-to-application (A2A) concepts and technologies.
- Microsoft Dynamics AX 2012 Application Integration Framework (AIF) setup and configuration.
- Microsoft BizTalk Server 2010 administration, including troubleshooting BizTalk applications.
- BizTalk application development.
- Microsoft Visual Studio® 2010.
- Extensible Markup Language (XML) schemas of the documents exchanged.

For information about how to set up and configure AIF, see Services and Application Integration Framework on the TechNet Web site.

System requirements

Your systems must meet the system requirements for Microsoft Dynamics AX with AIF and for BizTalk. For performance reasons, we do not recommend running Microsoft Dynamics AX and BizTalk on the same server in production environments.

To understand the system requirements for Microsoft Dynamics AX 2012, see the Microsoft Dynamics AX 2012 System Requirements white paper. To understand the system requirements for Microsoft BizTalk Server 2010, see the System Requirements on the BizTalk Web site.

Frequently asked questions

This section addresses many common issues about integration between BizTalk and AIF. For some scenarios, this information may be sufficient to help you get up and running. The remaining sections of this white paper provide the same information in a more complete context. If you want to read a complete walkthrough of integration between BizTalk and AIF, you can skip this section.

Upgrading BizTalk integration from Microsoft Dynamics AX2009

What happened to the BizTalk adapter?

Microsoft Dynamics AX 2012 introduces a new design for services and AIF. As a result, the BizTalk adapter was removed. In Microsoft Dynamics AX 2012, you must use various adapters to exchange documents between BizTalk and AIF.

What happened to channels and endpoints?

In Microsoft Dynamics AX 2012, AIF exposes virtual integration ports to simplify the administration of inbound and outbound connections. These virtual ports replace the channels and endpoints used by previous versions of Microsoft Dynamics AX.

Services and AIF

When adding service operations to my AIF port, why don't I see the service operations listed that I expect?

In Microsoft Dynamics AX 2012, AIF requires that you register services by using the **Initialization checklist**. Click **Set up Application Integration Framework**.

Why do I get a permissions error from Microsoft Dynamics AX when attempting to transmit a document from BizTalk to an inbound port?

Make sure that the account that BizTalk runs under is listed as a user account in Microsoft Dynamics AX and has the required roles for the services that BizTalk uses.

Where can I learn about security practices for services and AIF?

To learn about security practices, see the AIF documentation for system administrators on TechNet.

Schemas

How should I format my document before submitting it to AIF?

What format can I expect when AIF sends me a document?

AIF service documents are always constructed from XML that conforms to a service schema definition and is contained in a special wrapper, called an *envelope*. For exchanges that use SOAP, such as exchanges over TCP/IP or HTTP, the envelope is the standard SOAP envelope. For other exchanges, such as through Message Queuing or the file system, AIF provides an envelope schema. The namespace for the AIF envelope schema is:

```
http://schemas.microsoft.com/Microsoft Dynamics/2011/01/documents/Message
```

AIF uses entity key schemas to contain name-value pairs, such as those used to query for a particular item during a read operation or when sending a response to a create operation. The namespaces for entity keys and entity key lists are:

```
http://schemas.microsoft.com/Microsoft Dynamics/2006/02/documents/EntityKey
http://schemas.microsoft.com/Microsoft Dynamics/2006/02/documents/EntityKeyList
```

AIF aggregates common property types in the sharedtypes schema. The namespace for the sharedtypes schema is:

```
http://schemas.microsoft.com/Microsoft Dynamics/2008/01/sharedtypes
```

For message sets, AIF uses the batch schema. The namespace of the batch schema is:

```
http://schemas.microsoft.com/Microsoft Dynamics/2009/06/documents/Batch
```

AIF uses the fault schema to contain response messages about error conditions. The namespace for the fault schema is:

```
http://schemas.microsoft.com/dynamics/2008/01/documents/Fault
```

Where can I get the schema definitions for Microsoft Dynamics AX services?

You can retrieve common schema files from the following directory where you installed Microsoft Dynamics AX:

Program files`\Microsoft Dynamics AX\60\Server\MicrosoftDynamicsAX\bin\Application\Share\Include`

You can save a full or customized version of a service schema by creating a custom data policy during configuration of an AIF integration port. The **Data policies** button (in the **Inbound ports** and **Outbound ports** forms) opens the **Document data policies** form. You can use this form to set rules for the fields that can be used in documents processed by an integration port for a given service operation. However, you cannot disable fields that are required by the service schema. The **View schema** button opens the XML viewer, where you can view and save the schema (and its imported schemas, such as **sharedtypes**) to an XSD file.

You can generate schemas from the published WSDL document of an exposed network-protocol-based service, such as from a port that uses the NetTcp adapter. BizTalk's WCF Consuming Services Wizard can generate the schema documents and binding information for you.

How do I generate an instance of an XML document based on an XSD?

The **Generate Sample XML** feature in Visual Studio generates a sample XML file based on an XSD file. In **Solution Explorer**, right-click the XSD file name and then click **Generate Instance**.

Message exchanges

Does AIF integration require exchanges to be synchronous or asynchronous?

Messages can be exchanged either synchronously or asynchronously, depending on the types of ports and adapters used.

How do I receive documents that originate in Microsoft Dynamics AX?

Documents that originate in Microsoft Dynamics AX are sent through an outbound port. Microsoft Dynamics AX includes outbound port adapters for the file system adapter and the MSMQ adapter. This means that, by default, you can perform only asynchronous exchanges for scenarios that start with Microsoft Dynamics AX. It is possible to write a custom adapter for synchronous, outbound scenarios. This white paper does not cover the creation or use of custom adapters.

How can Microsoft Dynamics AX consume a BizTalk Web service?

Download the white paper Consuming Microsoft Dynamics AX 2012 Web Services.

BizTalk projects

Do I need to use an orchestration to construct an AIF document?

No. You can provide an orchestration if you need to, but no orchestration is required to create an AIF message. Envelope assembly or disassembly happens in a pipeline. You can use the standard XML Receive and XML Transmit pipelines; a custom pipeline is not required. You can configure the default pipelines when you configure the ports, after you have deployed your BizTalk Server application.

What must the project contain?

Add the schemas required for the document exchange (including imported schemas) to the BizTalk project. Usually, these schemas include a source schema, a destination schema, and the AIF message (envelope) schema, if it is required. Whether the AIF document service schema is the source or the destination depends on the direction of the message.

Create a map to translate values from elements in the source schema to elements in the destination schema.

How do I configure the envelope?

For asynchronous exchanges, you must configure the AIF message schema as an envelope in your Visual Studio project as follows:

- Set the **Envelope** property to **Yes**.
- Set **RootReference** to Envelope.
- For the **Envelope** element, point the **Body XPath** property at the **MessageParts** element (not the **Body** element).

For synchronous exchanges, you can configure the SOAP envelope properties in the administration console during port configuration of the send port.

How do I copy values from a source document to the envelope header?

BizTalk's property demotion feature can copy values from a source document to a destination header. This copying happens automatically when the elements have been promoted and they share the same name and data type. You may find this feature useful for copying a message ID between headers, for example.

How should I configure my project for deployment?

In the BizTalk deployment project properties, provide an application name, select **True** for **Restart Host Instances**, and provide a signing key.

How do I enable matching of requests to responses in asynchronous scenarios?

For asynchronous exchanges, always provide a unique **MessageId** value, because you will use this value to correlate requests and responses. Microsoft Dynamics AX requires the message ID to be a GUID. AIF contains the initial message ID in its responses by using the **RequestMessageId** element.

BizTalk administration

Why don't I see my project in the administration console?

Your deployed Visual Studio project will be listed in the **Applications** node in the BizTalk Server Administration Console. If you did not provide an application name before deployment, your application was given a default name, such as "BizTalk Application 1".

How do I configure a BizTalk pipeline to assemble an AIF message?

To assemble an AIF message envelope, provide the **EnvelopeDocSpecNames** and **DocumentSpecNames** values in the XML Transmit pipeline's properties dialog box. These values take the form: "*full schema name*, *full assembly name*". You can view the schema and assembly names in the schema properties dialog box in the BizTalk Server Administration Console. The following text shows an example of specifying the AIF message schema in **EnvelopeDocSpecNames**:

```
MyProj.AIFMessage, MyProj, Version=1.0.0.0, Culture=neutral, PublicKeyToken=MyKeyToken
```

If you need to assemble multiple documents within the envelope, separate the document spec names by using a pipe character ("|").

How do I configure a BizTalk pipeline to disassemble an AIF message?

To disassemble an AIF message, provide the **EnvelopeDocSpecNames** and **DocumentSpecNames** values in the XML Receive pipeline's properties dialog box. See the previous answer for details.

How do I connect the BizTalk ports to each other without using an orchestration?

In BizTalk, you can connect ports to each other (creating a *subscription*) by using a *filter*. For example, you can set the **BTS.ReceivePortName** property in a filter to subscribe a send port to all messages from a particular receive port. You can also subscribe by using other filters, such as message type or port ID. You can subscribe to the receive pipeline of a solicit-response port by using the **BTS.SPName** property.

How do I match a response to its original request?

This process, called *correlation*, requires you to match a value (or set of values) from the request document with corresponding values in the response document. You will typically match the **MessageId** element from the request (a GUID that you generate) to the **RequestMessageId** from the response.

BizTalk Server requires that you create a *correlation set* only if you use an orchestration in a synchronous document exchange. In this case, the correlation set can contain these same ID values. For asynchronous exchanges, and synchronous exchanges that do not use an orchestration, no BizTalk correlation set is required. In this case, you can use whatever method you choose to correlate requests and responses by using the ID values.

Key concepts for integration

This section reviews important concepts, including new concepts for Microsoft Dynamics AX 2012, required to understand the rest of this white paper.

Application Integration Framework

AIF provides an extensible framework that enables you to expose Microsoft Dynamics AX business logic and to exchange data with other applications. In AIF, data is exchanged with external systems through *services*. This model provides the ability to expose any X++ class as a service. A service can be called from X++ or from an external system. As part of this programming model, a set of services is included with Microsoft Dynamics AX that is based on documents such as sales orders or purchase orders.

AIF enables the integration of Microsoft Dynamics AX through Web services, Microsoft Message Queuing (MSMQ), and the file system (using a directory). The following diagram provides an overview of the AIF architecture.

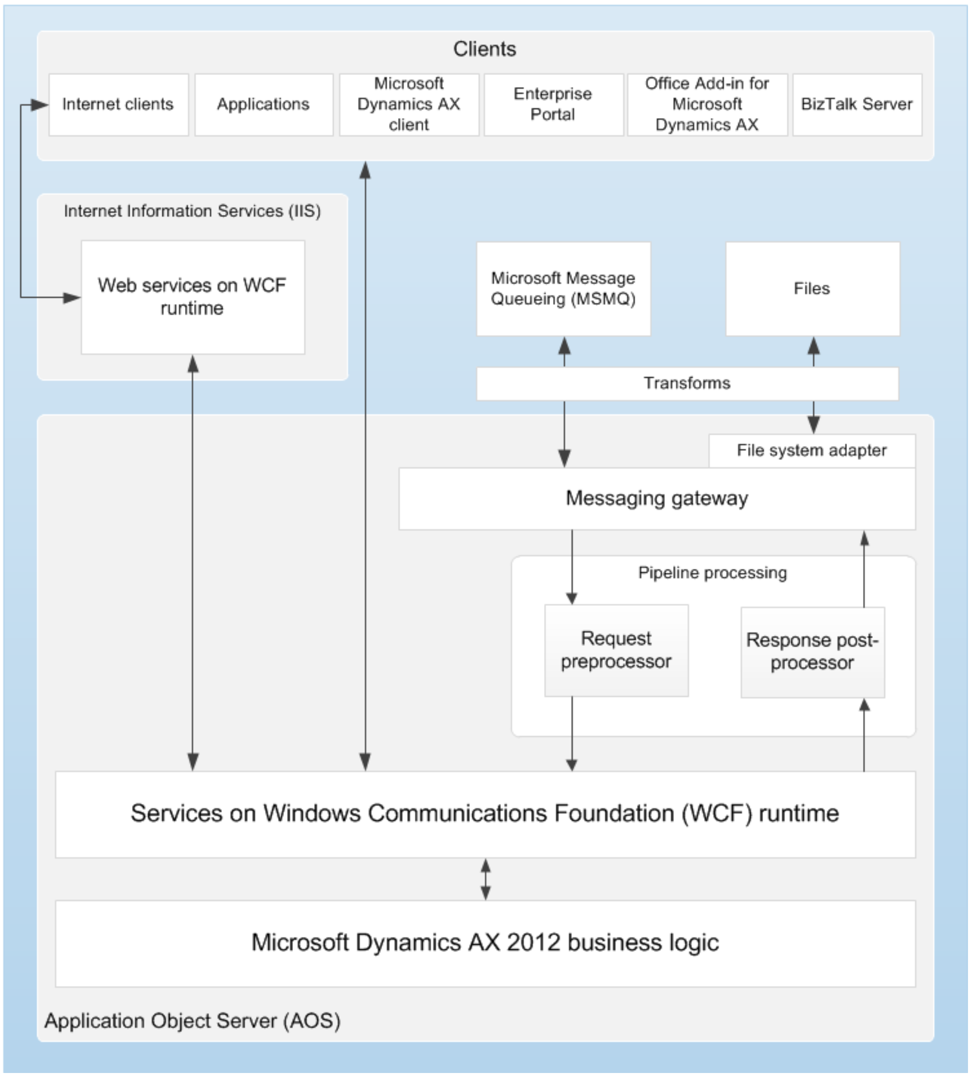

Clients
Internet clients
Applications
Microsoft Dynamics AX client
Enterprise Portal
Office Add-in for Microsoft Dynamics AX
BizTalk Server
Internet Information Services (IIS)
Web services on WCF runtime
Microsoft Message Queueing (MSMQ)
Files
Transforms
File system adapter
Messaging gateway
Pipeline processing
Request preprocessor
Response post-processor
Services on Windows Communications Foundation (WCF) runtime
Microsoft Dynamics AX 2012 business logic
Application Object Server (AOS)

Figure 1: AIF architecture

In Figure 1, you can see that BizTalk is a client application. This means that you do not need to use a specialized adapter between BizTalk and AIF, as you did in Microsoft Dynamics AX 2009. Instead, BizTalk and AIF can communicate through WCF adapters or the file system adapters. Microsoft Dynamics AX can also consume Web services exposed by BizTalk orchestrations, which does not require the use of AIF integration ports. For more information about consuming Web services, see the section Consuming a BizTalk Web service.

Messages can flow in two directions: into AIF and out from AIF. Messages that flow into AIF are called *inbound* messages. Messages that flow out from AIF, including responses to inbound messages, are called *outbound* messages.

Integration ports

Integration ports provide simplified administration of AIF. The concept of integration ports replaces AIF endpoints and related concepts used in previous releases of Microsoft Dynamics AX. Each integration port can expose one or more services and each integration port has a Uniform Resource Identifier (URI) that uniquely identifies the address of the port.

Each integration port also has a direction; that is, it can be either an inbound integration port or an outbound integration port. An inbound integration port is a destination for XML messages originating from outside of Microsoft Dynamics AX. An outbound integration port is a destination for XML messages originating from your Microsoft Dynamics AX system.

Inbound integration ports can be one of two types, either basic or enhanced. Outbound integration ports are always enhanced ports. When compared to a basic integration port, an enhanced integration port enables you to provide a wide variety of customizations, such as choosing the communication protocol, performing pre- and post-processing of documents, and specifying document data policies. In this white paper, you will learn how to configure an enhanced integration port for a simple integration with BizTalk.

XML documents and schemas

Although AIF supports the transfer of data in any format, most information exchanges with AIF services use XML documents. In order to create an XML document that adheres to a standard for a particular exchange of information, AIF requires XML documents to follow an XML style definition (XSD). XSD files (which have an .xsd file name extension) are meta-documents that describe the format, or *schema*, of XML documents that declare the namespace of the XSD. Each schema includes rules about the hierarchical arrangement of XML elements, which elements must be present in the document, and other such requirements.

Service schemas

Each AIF service has an associated XSD that describes the requirements for the service. You can save the XSD file for a service when you create or edit an integration port in Microsoft Dynamics AX. Schemas are available only for those services that have been *registered*. To register all available services, follow these steps:

1. Click System administration > Setup > Checklists > Initialization checklist.
2. Expand the **Initialize system** node.
3. Click Set up Application Integration Framework.

Common schemas

Common schemas are used in conjunction with service schemas. You can retrieve the XSD files for common schemas from the following directory:

```
Program files\Microsoft Dynamics AX\60\Server\MicrosoftDynamicsAX\bin\Application\Share\Include
```

In Microsoft Dynamics AX 2012, all types that are shared between multiple document types have been moved to sharedtypes.xsd, the sharedtypes schema file. All document schemas import this schema. The namespace for the shared-types schema is:

```
http://schemas.microsoft.com/Microsoft Dynamics/2008/01/sharedtypes
```

Network-based exchanges, such as those made through the WCF NetTcp and HTTP adapters, use SOAP conventions. These exchanges require that documents be contained in a SOAP envelope.

For other exchanges, such as those made through the file system or MSMQ adapters, AIF provides its own envelope schema, called the *message schema*, to provide a standard message wrapper for documents used by AIF. In these scenarios, both requests and responses are contained by the AIF message envelope.

The namespace for the AIF message schema is:

```
http://schemas.microsoft.com/Microsoft Dynamics/2011/01/documents/Message
```

For message sets, AIF uses the batch schema. This schema supports including multiple AIF messages in a single document. Each message is contained in an AIF message envelope. A single element, named **Batch**, contains all the envelopes. The namespace of the batch schema is:

```
http://schemas.microsoft.com/Microsoft Dynamics/2009/06/documents/Batch
```

AIF uses the fault schema to contain response messages about error conditions. The namespace for the fault schema is:

```
http://schemas.microsoft.com/dynamics/2008/01/documents/Fault
```

Synchronous and asynchronous exchanges

AIF supports two basic modes of communication. In *synchronous* mode, requests are tightly coupled to responses. This means that the submitter of the request must wait for a response from AIF before proceeding. In this mode, AIF immediately processes the request and then sends a response. In *asynchronous* mode, requests are placed into a queue. AIF processes messages in the queue at a later time and then sends a response when processing is completed. In this mode, responses are delayed, but large volumes of messages can be processed more efficiently, and message processing can be controlled by changing various configuration settings.

The following table provides recommendations for choosing communication modes and protocols when setting up communication between AIF and BizTalk.

Exchange initiated by	Mode	Protocol	Envelope	Notes
Microsoft Dynamics AX	Synchronous	HTTP	SOAP	A BizTalk Server orchestration can expose a Web service.
Microsoft Dynamics AX	Asynchronous	Message Queuing by using the MSMQ adapter, or file exchange by using the file-system adapter	AIF message schema	Recommended for internal exchanges. File system adapter supports all file types. MSMQ supports only XML documents and has a message size limit of 4MB.
BizTalk	Synchronous	HTTP or NetTCP by using the WCF-based adapters.	SOAP	None.
BizTalk	Asynchronous	Message Queuing by using the MSMQ adapter, or file exchange by using the file-system adapter.	AIF message schema	Recommended for internal exchanges. File system adapter supports all file types. File system adapter supports batched message sets. MSMQ supports only XML documents and has a message size limit of 4MB.

The remaining sections of this white paper guide you through the following scenarios:

- Asynchronous scenario: Creating a sales order
- Synchronous scenario: Reading a sales order

Asynchronous scenario: Creating a sales order

This section demonstrates an asynchronous document exchange between BizTalk and AIF. In this scenario, BizTalk builds a request to create a new sales order and then sends the request to AIF for processing. The following parameters apply to this scenario:

- Microsoft Dynamics AX must be preconfigured to accommodate sales orders. For example, the system must contain customers, vendors, and products, and the minimum related accounting requirements must be in place.
- The exchange happens by using the file system adapter.
- The sales order starts out in a custom schema that does not match the AIF sales order schema.
- The standard BizTalk pipelines are used for XML exchanges.
- The communication mode is asynchronous.
- BizTalk initiates the request and AIF responds.

To implement this scenario, you will follow these general steps:

1. Create file system directories.
2. Create and configure the AIF enhanced integration port.
3. Create a new BizTalk project in Visual Studio.
4. Import and configure the schemas.
5. Map the source schema to the destination schemas—the sales order schema and the message schema.
6. Deploy the BizTalk application.
7. Configure the BizTalk Server Administration Console to execute the pipelines, map the schemas, and use the message schema envelope.
8. Configure and start the AIF queues.
9. Submit an XML document that contains a request to create a new sales order.
10. View the XML document that contains the response.
11. Disassemble the response message to remove the AIF envelope.

Note: You might notice that this scenario does not use a BizTalk orchestration. No orchestration is required to create an AIF service document. All of the work happens in the BizTalk pipelines in the BizTalk ports. If you understand how to use BizTalk to create an AIF message and you need to perform additional processing, you can add an orchestration. The key point is that the BizTalk ports (and their pipelines) must be properly configured, and must receive documents that are in an expected format.

Creating the file system directories

Create the following three folders:

- A folder to contain the source document for the request. Name this folder RecIntSO, for Receive Internal Sales Order. This is the folder from which BizTalk will get the original sales order that is written in a custom schema.
- A folder to receive the AIF request. Name this folder AIFIn. This is the folder where BizTalk will place the sales order request that uses the AIF schemas. AIF will retrieve files from this folder when a batch job executes.

- A folder to receive the AIF response. Name this folder RecAIFResp.

Creating and configuring the AIF integration port

To receive the new sales order and provide a response message, you will use an inbound integration port in Microsoft Dynamics AX. Follow these steps to create the port:

1. Open the Inbound ports form. Click System administration > Setup > Services and Application Integration Framework > Inbound ports.
2. Click **New**.
3. Name the new integration port SalesOrderCreate.

Select and configure the file adapter and its address

1. In the **Adapter** list in the **Address** group, select **File system adapter**.
2. In the **URI** list, click the arrow to browse to the folder that you created, named AIFIn.
3. Select the **Response address** check box and repeat steps 1 and 2. This time, in the **URI** list, provide the path for the folder that you created, named RecAIFResp.

Select the service operation

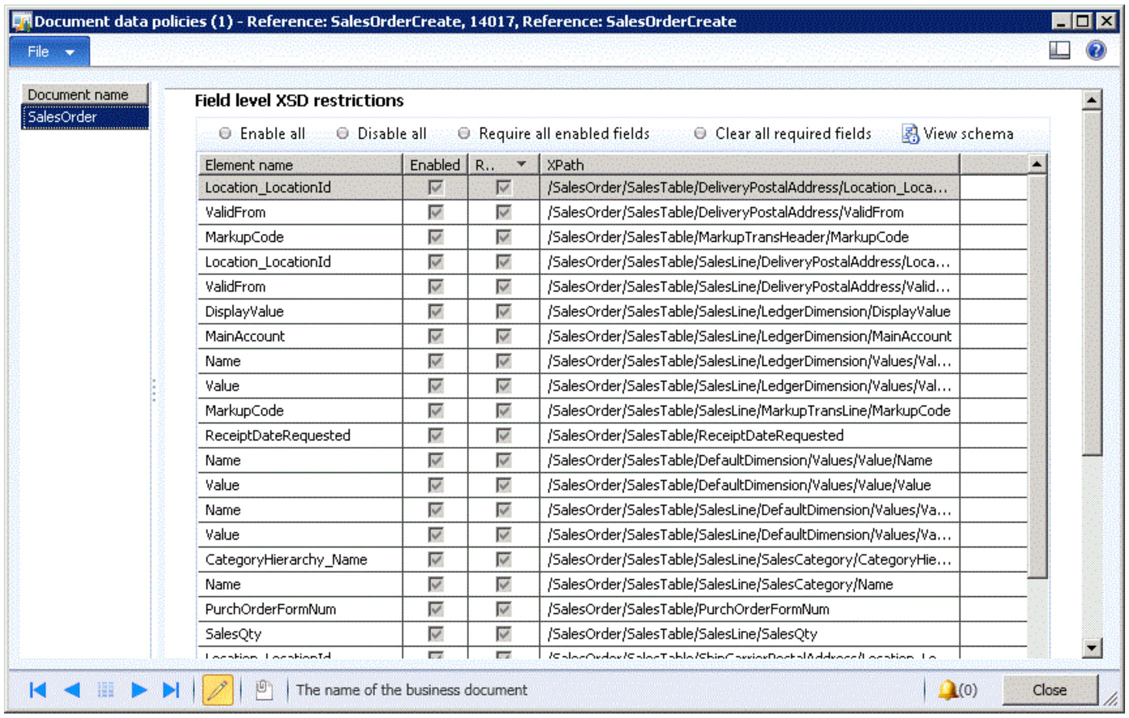
For this example, you will select the **SalesSalesOrderService.create** service operation.

1. On the **Service contract customizations** tab, click **Service operations**.
2. In the **Select service operations** form, select **SalesSalesOrderService.create** in the Remaining service operations list. Then, click the left arrow to move the service operation to the **Selected service operations** list.
3. Close the form.

Configure the data policies

1. The **Document data policies** form enables you to customize the schema for the selected document. You can choose which fields to enable or require. Limiting the number of fields that you expose from an AIF integration port is a security best practice. Doing so can also help you to make the schema more compact by eliminating fields that you do not want to use.
2. This is also a good time to retrieve the sales order schema.
3. On the **Service contract customizations** tab of the **Inbound ports** form, select **Customize documents** and then click **Data policies**. The **Document data policies** form opens.

   If you sort the table on the **Required** field, you can see that the sales order schema requires a number of elements, and that you cannot change the required or enabled settings for these required elements. The following image shows the **Document data policies** form displaying information about the sales order schema.


Document data policies (1) - Reference: SalesOrderCreate, 14017, Reference: SalesOrderCreate
File
Document name
SalesOrder
Field level XSD restrictions
Enable all
Disable all
Require all enabled fields
Clear all required fields
View schema
The name of the business document
(0)
Close

Element name	Enabled	R..	XPath
Location_LocationId	✓	✓	/SalesOrder/SalesTable/DeliveryPostalAddress/Location_Loca…
ValidFrom	✓	✓	/SalesOrder/SalesTable/DeliveryPostalAddress/ValidFrom
MarkupCode	✓	✓	/SalesOrder/SalesTable/MarkupTransHeader/MarkupCode
Location_LocationId	✓	✓	/SalesOrder/SalesTable/SalesLine/DeliveryPostalAddress/Loca…
ValidFrom	✓	✓	/SalesOrder/SalesTable/SalesLine/DeliveryPostalAddress/Valid…
DisplayValue	✓	✓	/SalesOrder/SalesTable/SalesLine/LedgerDimension/DisplayValue
MainAccount	✓	✓	/SalesOrder/SalesTable/SalesLine/LedgerDimension/MainAccount
Name	✓	✓	/SalesOrder/SalesTable/SalesLine/LedgerDimension/Values/Val…
Value	✓	✓	/SalesOrder/SalesTable/SalesLine/LedgerDimension/Values/Val…
MarkupCode	✓	✓	/SalesOrder/SalesTable/SalesLine/MarkupTransLine/MarkupCode
ReceiptDateRequested	✓	✓	/SalesOrder/SalesTable/ReceiptDateRequested
Name	✓	✓	/SalesOrder/SalesTable/DefaultDimension/Values/Value/Name
Value	✓	✓	/SalesOrder/SalesTable/DefaultDimension/Values/Value/Value
Name	✓	✓	/SalesOrder/SalesTable/SalesLine/DefaultDimension/Values/Va…
Value	✓	✓	/SalesOrder/SalesTable/SalesLine/DefaultDimension/Values/Va…
CategoryHierarchy_Name	✓	✓	/SalesOrder/SalesTable/SalesLine/SalesCategory/CategoryHie…
Name	✓	✓	/SalesOrder/SalesTable/SalesLine/SalesCategory/Name
PurchOrderFormNum	✓	✓	/SalesOrder/SalesTable/PurchOrderFormNum
SalesQty	✓	✓	/SalesOrder/SalesTable/SalesLine/SalesQty
Location_LocationId	✓	✓	/SalesOrder/SalesTable/ShipCarrierPostalAddress/Location_Lo…

Figure 2: Document data policies form

   For this scenario, you will need to modify the default policy that is already in the form. Note that some elements use the same name but are located by using different XPaths.

4. Sort the table on the **Element name** column and then select **Enabled** for the following elements:
   - **CustAccount** (/SalesOrder/SalesTable/CustAccount)
   - **ItemId** (/SalesOrder/SalesTable/SalesLine/ItemId)

View and save the XSD

The **Document data policies** form enables you to view the schema XML, including imported schemas, for the configuration that you created. You can also save the schema code from the XML viewer.

1. To view the schema code, click **View schema**.
2. To save the schema code, click **Save as**. Name the file SalesOrder.xsd. When you save a schema by using this technique, the schema file will contain only those elements that have been enabled.
3. In the XML viewer, click **View imported schemas**. Save the shared-types schema as SharedTypes.xsd.
4. Close the open forms, except for the **Inbound ports** form.

Activate the port

1. In the **Inbound ports** form, click **Activate**.
2. Close the form.

Creating the BizTalk project

Use Visual Studio to create the BizTalk project. The BizTalk project that you will create for this scenario contains the necessary schemas and defines a map. The source schema defines some basic XML elements that fulfill the requirements for creating a sales order in Microsoft Dynamics AX. The destination schema is the Microsoft Dynamics AX sales order schema. A third schema, the AIF message schema, must be compiled into the project to enable wrapping the sales order in an AIF envelope when passed through the BizTalk XML Transmit pipeline. The map connects the source schema to the sales order schema.

1. Open Visual Studio 2010 with administrative privileges. (You must run Visual Studio as an administrator to deploy BizTalk solutions.)
2. In Visual Studio, create a new, empty BizTalk Server project. Name the project **SalesOrderProj**.

Create the source schema

For this example, the source schema is a very simple example of an XML schema that represents the *internal* sales order. You can create the internal sales order schema in the BizTalk project by adding a new schema to the project. Alternatively, you can create the schema as an XSD file and then add the existing schema to the project.

The following figure shows the source schema as it appears in the BizTalk project in Visual Studio:

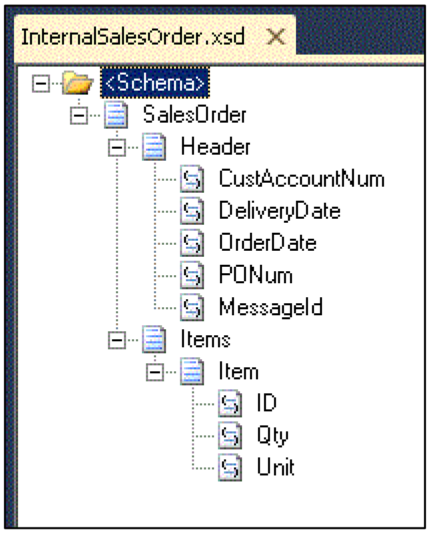


Figure 3: Source schema in BizTalk project in Visual Studio

This internal sales order schema contains a root element, **SalesOrder**, which has two child elements: a **Header** element and an **Items** element. The header contains common information for the sales order, such as the customer's account number. The **Items** element contains one or more **Item** elements; each **Item** element represents a sales line item in the sales order. An item has a product ID, a quantity for the number of items ordered, and a value that describes the unit ordered, such as "dozen." Each child element has an appropriate type, such as **xs:date** for date fields.

The following XML code shows the internal sales order schema:

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema xmlns="http://PropHeader.InternalSalesOrder"
xmlns:b="http://schemas.microsoft.com/BizTalk/2003"
targetNamespace="http://PropHeader.InternalSalesOrder"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="SalesOrder">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Header">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CustAccountNum" type="xs:string" />
              <xs:element name="DeliveryDate" type="xs:date" />
              <xs:element name="OrderDate" type="xs:date" />
              <xs:element name="PONum" type="xs:string" />
              <xs:element name="MessageId" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Items">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Item">
```

```
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ID" type="xs:string" />
                    <xs:element name="Qty" type="xs:nonNegativeInteger" />
                    <xs:element name="Unit" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Name the schema file **InternalSalesOrder.xsd**.

Promotion

After you have created or added the internal sales order schema, you must promote the **MessageId** element. This promotion will enable BizTalk to copy the message ID from the internal sales order to the envelope of the AIF sales order request.

1. Right-click **MessageId**, then point to **Promote**, and then click **Quick promotion**.
2. Visual Studio marks the promoted element with the following icon: .
3. Visual Studio creates a new schema, called the *property schema*, which contains the promoted properties in the BizTalk project.

Add the destination schemas

The destination schema is the Microsoft Dynamics AX sales order schema. This is the file named SalesOrder.xsd that you saved when using the **Document data policies** form when configuring the inbound integration port. You must also remember to include the imported schema that you named SharedTypes.xsd. Add these two files to your BizTalk project in Visual Studio. Then, use the **Properties** pane to verify that the **Imports** collection for the **SalesOrder** schema contains the **SharedTypes** schema.

The sales order schema includes two elements that are not defined in the source schema. Both elements are named **MarkupCode**, though their XPaths differ. These elements are marked as required by the schema, but their parent elements, **MarkupTransHeader** and **MarkupTransLine**, are not. For this example, you can safely delete the **MarkupTransHeader** element and its children from the **SalesTable** node, and the **MarkupTransLine** element and its children from the **SalesLine** node. Use the Visual Studio XML editor to make these changes. If you choose to retain these parent elements as part of the sales order schema, then you must provide values for the **MarkupCode** elements. These instructions do not discuss how to provide values for **MarkupCode**.

Add and configure the envelope schema

Add the file named AIFMessage.xsd to your BizTalk project in Visual Studio:

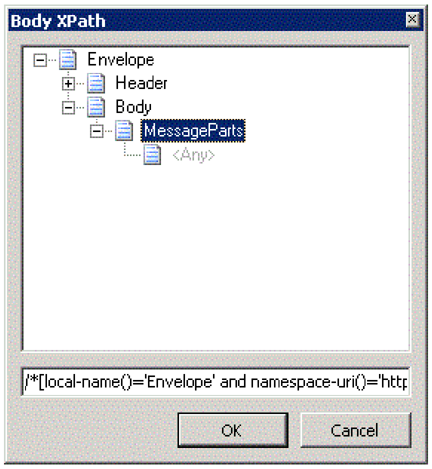
1. Copy the following file to your project folder:

```
Program Files\Microsoft Dynamics AX\60\Server\Microsoft
DynamicsAX\bin\Application\Share\Include\Message.xsd
```

2. Use **Solution Explorer** to add the file to your Visual Studio project.

After adding the AIF message schema to your project, you must configure the schema to enable BizTalk to use it as an envelope. Follow these steps:

1. In the BizTalk project in Visual Studio, open AIFMessage.xsd.
2. In the **Properties** pane, change the **Envelope** property to Yes.
3. For Root Reference, select Envelope.
4. In the schema tree view, select the **Envelope** element.
5. In the **Properties** pane, select **Body XPath** and then click the ellipsis button (**...**).
6. In the **Body XPath** dialog box, select **MessageParts**, as shown in the following image:



/*[local-name()='Envelope' and namespace-uri()='http

Figure 4: Body Xpath dialog box

7. Close the **Body XPath** dialog box.
8. In the schema tree view, expand the **Header** element, and then select **Action.**
9. In the **Properties** pane, enter the **Default Value** for the sales order create operation:
10. http://schemas.microsoft.com/Microsoft Dynamics/2008/01/services/SalesOrderService/create
11. In the schema tree view, promote the **MessageId** element.

In the preceding steps, you marked the AIF message schema as an envelope. This setting is required so that BizTalk can identify the schema as an eligible envelope schema in other configuration settings. You also marked the **MessageParts** element as the body element for the envelope. This means that when BizTalk combines the envelope with a document, the document will be inserted as a child of the **MessageParts** element. This is exactly the organization that AIF requires for service documents.

About the header elements

Specifying a value for **MessageId** for the request enables you, at a later time, to match requests to responses from AIF. Each AIF response message contains a **RequestMessageId** element that contains the same ID as the original message that you sent. If you do not specify a message ID in the request document, then AIF creates one for you. However, because you cannot know the value that AIF creates in advance, you cannot use it to match your request to the correct response from AIF.

For Microsoft Dynamics AX, a message ID must always contain a globally unique identifier (GUID). For the schema, you can use the **xsd:string** data type to contain the GUID.

BizTalk will automatically copy the **MessageId** element's value from the internal sales order's header to the **MessageId** element in the envelope header. This process, called *property demotion*, requires that the element names and types match exactly, that the element has been promoted, and that the value has been set in the source document before submitting the document to the transmission pipeline.

You will view the **RequestMessageId** value, after you receive a response from AIF, in an upcoming section.

In Step 9 of the preceding section, you specified the sales order service **create** operation as the default value for the **Action** element in the envelope XML. When BizTalk creates the envelope portion of the request document, this value will be automatically added to the envelope header. For this scenario, using the default value is a convenient mechanism for specifying a value for the **Action** element. Instead, you could have specified an **Action** element as part of the internal sales order schema and then used property demotion to copy the value to the envelope header, exactly like you did for **MessageId**.

It is likely that your real-world source documents will not contain **Action** or **MessageId** (at least not using the formats or naming conventions required by AIF). In that case, you can think of the internal sales order schema in this example as an intermediate schema between your real source schema and the AIF sales order schema. You can use an orchestration with a transform (a map) and a message assignment to create such values, as appropriate. This white paper does not provide examples of how to use such an orchestration.

Create the map

To translate elements between the source and destination schemas, BizTalk uses a *map*. Add a new map to your BizTalk project in Visual Studio. Accept the default file name, Map1.btm. Configure the map to look like the following screen image:

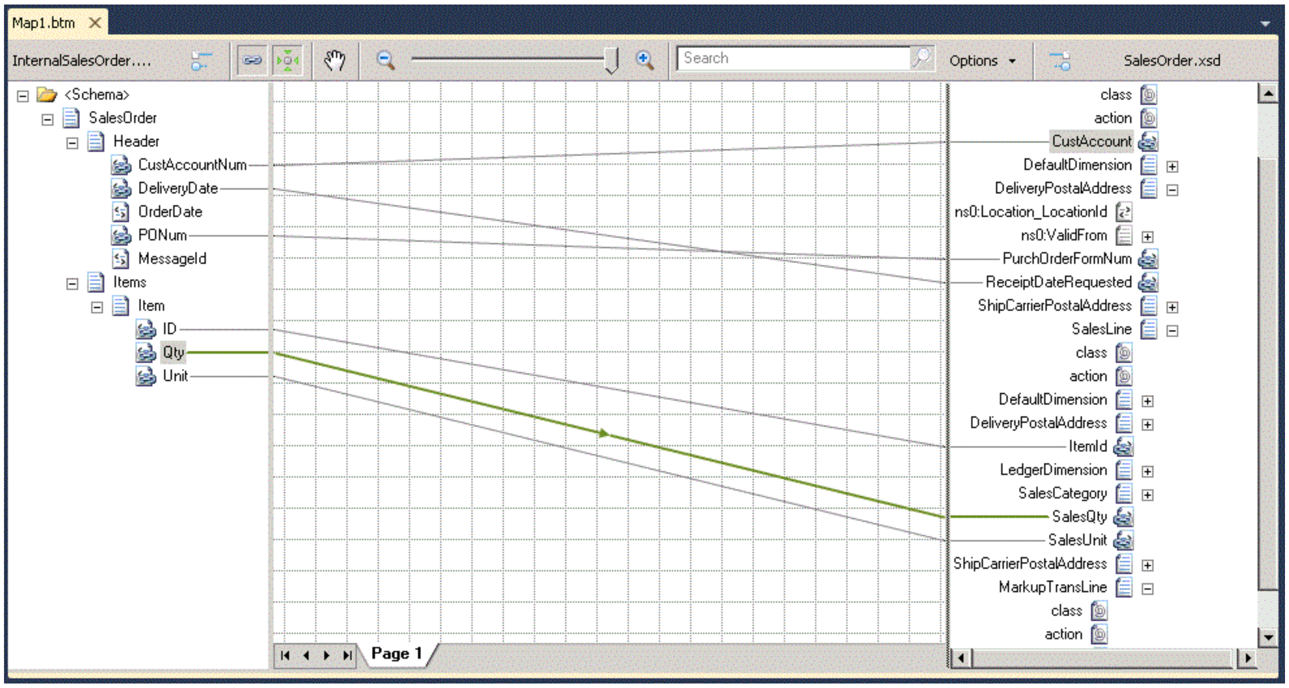


Figure 5: BizTalk map

The following table describes the mapping that is contained in the screen image.

Source element	Destination element
CustAccountNum	CustAccount
DeliveryDate	ReceiptDateRequested
PONum	PurchOrderFormNum
ID	ItemId
Qty	SalesQty
Unit	SalesUnit

Notice that the **OrderDate** element is not mapped. Microsoft Dynamics AX will add the current date and time to the sales order when it is created in the database.

Deploy the BizTalk application

Use Visual Studio to deploy the application. First, make the following changes to the project properties:

1. On the **Deployment** tab, provide a value for the **Application Name** field, such as "CreateSalesOrder". If you do not provide a name for your application, BizTalk will deploy the application by using a default name. It is a best practice to provide a unique name.
2. For Restart Host Instances, select True.
3. It is important to restart the in-process host instances any time that you redeploy the application. Otherwise, cached information can cause unexpected behavior when you run your application.
4. On the **Signing** tab, specify a signing key. Deployed BizTalk assemblies must be strong-name signed.

After deployment configuration is completed, deploy the application. On the **Build** menu, click **Deploy Solution**. Visual Studio builds and then deploys the assembly that contains the schemas and maps.

Configure the BizTalk Server Administration console

The BizTalk Server Administration console is a Microsoft Management Console (MMC) snap-in that provides a user interface that you can use to manage BizTalk Server. For this example, you will use the console to host and run your BizTalk application. To configure the application, open the **BizTalk Server Administration** console and then follow these steps:

First, in the tree-view pane, expand ConsoleRoot > BizTalk Server Administration > BizTalk Group > Applications > CreateSalesOrder.

Create the BizTalk receive port

1. Right-click **Receive ports** and add a new one-way receive port. The **Properties** dialog box opens.
2. Select **Receive Locations** and then click **New**.
3. In the **Type** list, select **FILE**. Then, click **Configure**.
4. In the **File Transport Properties** dialog box, click **Browse** and then browse to the folder that you named "RecIntSO." Close the dialog box.
5. In the Receive Pipeline list, select XML Receive.
6. Close all of the dialog boxes.

Create the schema name strings

BizTalk pipelines identify envelope and document schemas by using fully qualified name strings. A name string contains the fully qualified name of the schema and information about the DLL that contains the schema.

Note that BizTalk attempts to resolve schema namespaces by searching for schemas in all applications. If multiple applications contain schemas with an identical namespace, BizTalk cannot resolve the conflict unless the fully qualified name string is included in the pipeline properties. An unresolved conflict results in a suspended message instance. It is a good idea to always provide fully qualified names for document and envelope specifications in pipelines.

1. Copy the envelope schema name string.

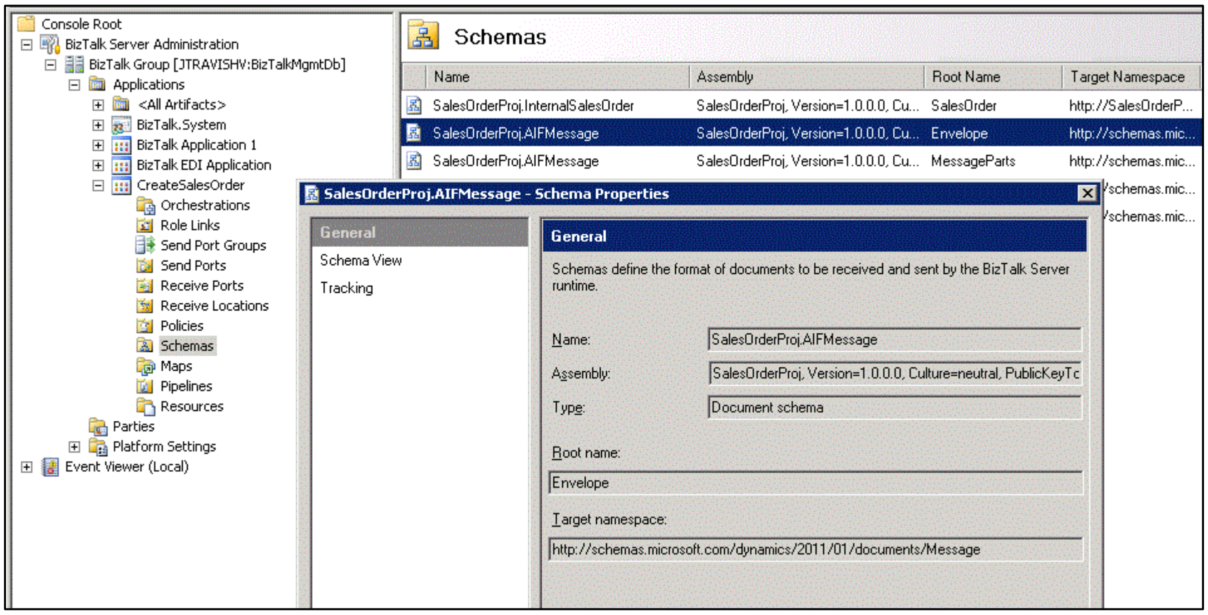
   a. In the tree-view pane, click **Schemas**.

   b. In the **Schemas** pane, double click the envelope schema name: **SalesOrderProj.AIFMessage**.

   c. Open Notepad. Copy the values from the **Name** field and the **Assembly** field and paste them into Notepad. Separate the two values with a comma (**,**). For example:

   ```
   SalesOrderProj.AIFMessage, SalesOrderProj, Version=1.0.0.0, Culture=neutral,
   PublicKeyToken=yourKeyToken
   ```

   You will use this text to configure the send port so that it can create the envelope.



Name	Assembly	Root Name	Target Namespace
SalesOrderProj.InternalSalesOrder	SalesOrderProj, Version=1.0.0.0, Cu...	SalesOrder	http://SalesOrderP...
SalesOrderProj.AIFMessage	SalesOrderProj, Version=1.0.0.0, Cu...	Envelope	http://schemas.mic...
SalesOrderProj.AIFMessage	SalesOrderProj, Version=1.0.0.0, Cu...	MessageParts	http://schemas.mic...

SalesOrderProj.AIFMessage - Schema Properties

General

Schemas define the format of documents to be received and sent by the BizTalk Server runtime.

Name: SalesOrderProj.AIFMessage

Assembly: SalesOrderProj, Version=1.0.0.0, Culture=neutral, PublicKeyTc

Type: Document schema

Root name: Envelope

Target namespace: http://schemas.microsoft.com/dynamics/2011/01/documents/Message

Figure 6: Configure the send port

2. Copy the document schema and assembly names. This process is the same as the one described previously. This time, you need to create a text string that contains the full name for both document schemas: the internal sales order schema and the AIF sales order schema.

   a. Make a copy of the name and assembly for the schema named **SalesOrderProj.InternalSalesOrder**.

   b. Make a copy of the name and assembly for the schema named **SalesOrderProj.SalesOrder**.

   c. Join the assembly names, in a single line of text, by using a pipe character (|). (Recall that the pipe character is the standard symbol for the OR operation.) For example:

```
SalesOrderProj.InternalSalesOrder, SalesOrderProj, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=yourKeyToken | SalesOrderProj.SalesOrder, SalesOrderProj, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=yourKeyToken
```

You will use this text to configure the send port.

Create the BizTalk send port

1. Right-click **Send Ports** and add a new, static, one-way send port. The **Properties** dialog box opens.
2. In the **Type** list, select **FILE**. Then, click **Configure**.
3. In the **File Transport Properties** dialog box, click **Browse** and then browse to the folder that you named AIFIn. Close the dialog box.
4. In the **Send Pipeline** list, select **XML Transmit**. Then, click the ellipsis button (**...**).
5. In the **EnvelopeDocSpecNames** field, enter the full envelope schema name that you created in the previous section.
6. In the **DocumentSpecNames** field, enter the document schema string that you created in the previous section.
7. Click **OK** to close the dialog box.

Note about using a custom BizTalk pipeline

If you are familiar with creating your own pipelines, you might instead choose to create a simple pipeline that contains only the XML assembler component. In this case, instead of configuring the default XML Transmit pipeline as described in the previous section, you can choose values for **EnvelopeDocSpecNames** and **DocumentSpecNames** by using the **Properties** window for the XML assembler, in Visual Studio. Of course, you will need to define the pipeline before deploying the application, and then configure the port to use the custom pipeline instead of the default pipeline.

There is no particular advantage to creating a custom pipeline for this scenario, except perhaps that it may be easier to select schema names from a list, rather than having to copy and paste them from the properties dialog box. If you need to use a custom pipeline for other reasons, then you must add and configure the XML assembler in the custom pipeline instead of in the port configuration.

Add the map to the send port

1. In the **SendPortProperties** dialog box, click **Outbound Maps**.
2. Add a new map. In the **Source Document** list, select **InternalSalesOrder**.
3. In the **Map** list, select **Map1**.
4. In the **Target Document** list, select **SalesOrder**.

Create the subscription

1. Link the ports. In the **SendPortProperties** dialog box, click **Filters**.
2. Add a new filter for the property named **BTS.ReceivePortName**.
3. In the **Operator** field, select **==**.
4. In the **Value** field, select **ReceivePort1** (the name of the receive port that you created in the previous section).
5. Click **OK** to close the dialog box.

Start the application.

In the **BizTalk Server Administration** console, start the application.

Configure and start the AIF batch jobs

For asynchronous document exchanges, such as the one demonstrated in this scenario, you must configure an AIF batch job. This batch job periodically executes business logic through integration ports that are configured to use asynchronous adapters. If you have previously configured an AIF batch job, you can skip this section.

Note: Because you are already familiar with Microsoft Dynamics AX and AIF, this section provides only general guidance for how to create the batch job. For detailed information about how to create batch jobs, see the system administration help on the Microsoft TechNet web site.

To configure and start the AIF queues:

1. Open the **Batch job** form. Click **System Administration** > **Inquiries** > **Batch jobs**.
2. Create a new batch job.
3. Add four tasks that use the following classes to the batch job: **AifGatewayReceiveService**, **AifGatewaySendService**, **AifInboundProcessingService**, **AifOutboundProcessingService**.
4. Set a recurrence interval for the batch job. For testing purposes, it is convenient to set the interval to one minute.
5. To start the batch job, change its status to **Waiting**.

Submit an XML document and receive the response message

Now that your BizTalk application and AIF integration port are running, you can submit a document for processing. First, you must create a source document that complies with the internal sales order schema. Next, you must save the document to the folder named RecIntSO. When the processing is complete, the response from AIF will appear in the folder that you named RecAIFResp.

The following example contains XML code that conforms to the internal sales order schema. Keep in mind that the specific values in this document may not match valid values in your Microsoft Dynamics AX system. Replace such values with valid examples. Be certain to use a new GUID for the message ID in each new request.

```
<?xml version="1.0" encoding="utf-8" ?>
<ns0:SalesOrder xmlns:ns0="http://PropHeader.InternalSalesOrder">
  <Header>
    <CustAccountNum>100002</CustAccountNum>
    <DateReq>2011-05-30</DateReq>
    <DeliveryDate>2011-05-31</DeliveryDate>
    <OrderDate>2011-05-01</OrderDate>
    <PONum>P0</PONum>
    <MessageId>{5603D03A-4380-404D-9F27-738BE0FEA13E}</MessageId>
  </Header>
  <Items>
     <Item>
       <ID>10003</ID>
       <Qty>10</Qty>
       <Unit>Pcs</Unit>
     </Item>
  </Items>
</ns0:SalesOrder>
```

Tip: Keep in mind that you can use Visual Studio to generate an example instance of your XML schema. To do this, right-click the schema file name in **Solution Explorer** and then click **Generate Instance**.

After BizTalk and AIF have processed the sales order, AIF saves the response document in the folder that you named RecAIFResp. The following XML code is an example of an AIF response message:

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="http://schemas.microsoft.com/Microsoft Dynamics/2011/01/documents/Message">
   <Header>
      <MessageId>{849FA371-945D-4A39-A7A8-F0DCF8481044}</MessageId>
      <Action>http://schemas.microsoft.com/Microsoft 
Dynamics/2008/01/services/SalesOrderService/create
      </Action>
      <RequestMessageId>{5603D03A-4380-404D-9F27-738BE0FEA13E}</RequestMessageId>
   </Header>
   <Body>
      <MessageParts xmlns="http://schemas.microsoft.com/Microsoft 
Dynamics/2011/01/documents/Message">
         <EntityKeyList xmlns="http://schemas.microsoft.com/Microsoft 
Dynamics/2006/02/documents/EntityKeyList">
            <EntityKey xmlns="http://schemas.microsoft.com/Microsoft 
Dynamics/2006/02/documents/EntityKey">
               <KeyData>
                  <KeyField>
                     <Field>SalesId</Field>
                     <Value>SO-100015</Value>
                  </KeyField>
              </KeyData>
           </EntityKey>
        </EntityKeyList>
     </MessageParts>
   </Body>
</Envelope>
```

Note that the value for **RequestMessageId** matched the message ID that you submitted in the original message. Recall that you can use these values to match, or *correlate*, requests and responses. How you choose to correlate messages in an asynchronous scenario will depend on your system's requirements. For example, you might choose to maintain a database table that maps identifiers from source documents to GUIDs that you created for AIF documents.

You can process this response much as you processed the original request. That is, you can create your own target schema and then use a map, along with property demotion, to copy the values that you need to your target schema.

Disassemble the response message

In some cases, you might want to remove the AIF envelope from a response so that you can process only the underlying data. For example, you might choose to store the response information to use as part of a future read operation. To remove the envelope, you must use the XML disassembler in the BizTalk receive pipeline. You can think about this process as similar to the assembly process that you used to construct the AIF message. In this case, though, instead of adding the envelope wrapper by using the XML assembler in the BizTalk send pipeline, you use the XML disassembler in the receive pipeline to remove the envelope code. This section builds on what you have done so far, and adds the necessary parts for disassembly.

Add the entity key schemas

The AIF response messages use the following schemas:

```
http://schemas.microsoft.com/Microsoft Dynamics/2006/02/documents/EntityKey
http://schemas.microsoft.com/Microsoft Dynamics/2006/02/documents/EntityKeyList
```

The EntityKeyList schema imports the EntityKey schema. Copy each of these schemas from the common schemas folder to the Visual Studio project folder. Copy the following files:

Program Files\Microsoft Dynamics AX\60\Server\MicrosoftDynamicsAX\bin\Application\Share\Include\EntityKeyList.xsd

Program Files\Microsoft Dynamics AX\60\Server\MicrosoftDynamicsAX\bin\Application\Share\Include\EntityKey.xsd

Use **Solution Explorer** to add these schemas to the Visual Studio project and then redeploy the project.

Create the BizTalk ports

1. Create a new directory to receive the processed response. Name this directory FinalSOResp, for Final Sales Order Response.
2. In the BizTalk administration console, stop the **CreateSalesOrder** application.
3. Create a new receive port, named ReceivePort2, and add a new receive location.
    a. Set the transport type to **File**. Set the directory to RecAIFResp.
    b. Set the receive handler to **BizTalkServerApplication**.
    c. Set the receive pipeline to **XML Receive**. Then, click the ellipsis button (**…**).
    d. In the **EnvelopeDocSpecNames** field, enter the full envelope schema name for the AIF message schema. This is the same name you entered in this field when you created the send pipeline.
    e. In the **DocumentSpecNames** field, enter the full schema name for the EntityKeyList schema.
    f. Click **OK** to close the dialog boxes.
4. Create a new send port, named SendPort2.
    a. Set the transport type to **File**. Set the directory to FinalSOResp.
    b. Set the send handler to **BizTalkServerApplication**.
    c. Set the send pipeline to **Pass Thru Transmit**.
    d. Click **OK** to close the dialog box.

Run the application

In the BizTalk administration console, start the **CreateSalesOrder** application.

BizTalk Server processes the response XML files that are in the directory named RecAIFResp. The application removes the AIF message envelope and then saves a new XML file that contains only the entity key document. BizTalk saves the new file in the directory named FinalSOResp.

Retain this XML file so that you can use it in the next scenario.

Receiving documents that originate in Microsoft Dynamics AX

Documents that originate in Microsoft Dynamics AX are sent through an outbound integration port. Microsoft Dynamics AX includes two asynchronous adapters for use in outbound scenarios: the file-system adapter and the MSMQ adapter. Synchronous outbound adapters are not installed with Microsoft Dynamics AX, though you can create a custom synchronous adapter for outbound port scenarios. This white paper does not discuss custom adapters.

Receiving an outbound document from Microsoft Dynamics AX is just like receiving a response to an asynchronous request, as described in the previous sections. For example, when using the file system, you must create a receive port in BizTalk that uses the BizTalk file adapter to monitor a shared folder. When the AIF outbound port saves a document to the folder, BizTalk will receive the document and then process it by using any associated pipeline components, maps, and orchestrations.

Synchronous scenario: Reading a sales order

This scenario demonstrates a synchronous document exchange between BizTalk and AIF. In this example, BizTalk constructs a request to read a particular sales order and then sends the request to AIF for processing. The following parameters apply to this scenario:

- Microsoft Dynamics AX must be preconfigured to accommodate sales orders. For example, the system must contain customers, vendors, and products, and the minimum related accounting requirements must be in place.
- The exchange happens by using the NetTcp adapter.
- The request starts out as the entity key list that was sent by AIF as a previous response to a create request. This document was the output of the very last step of the previous scenario.
- The standard BizTalk pipelines are used for XML exchanges.
- The communication mode is synchronous.
- BizTalk initiates the request and AIF responds.

To implement this scenario, you will follow these general steps:

1. Create the file system directories.
2. Create and configure an AIF enhanced integration port.
3. Create a new Visual Studio 2010 BizTalk project that references the service's published WSDL document and includes a map to create the service request document.
4. Configure the BizTalk Server Administration Console to add ports to application you deployed in the previous scenario.
5. Submit the XML document that contains a request to read the sales order.
6. View the XML document that contains the requested sales order.

Creating the file system directories

For this scenario, you need to create two directories:

- A directory to supply the initial request document. Name this directory RecSOReq, for Receive Sales Order Request.

- A directory to receive the requested sales order. Name this directory SOResp.

Creating and configuring the AIF integration port

Because this scenario requires a different adapter than the one used in the previous scenario, you must create a second integration port. The steps to create this integration port are the same as you followed in the previous example, with the following modifications:

1. Name the port TCPSalesOrderRead.
2. Select the **NetTcp** adapter. When you save the port configuration, Microsoft Dynamics AX will display the URI for the port.
3. In the Select service operations form, add SalesSalesOrderService.read to the Selected service operations list.
4. You can modify the document data policy to match the fields that you want to return in the response. If you do not select **Customize documents**, then AIF will include all sales order schema elements in the response message. If you select **Customize documents** but do not modify the default data policy, then AIF will include only the default elements in the sales order schema.
5. Activate the integration port.

Creating the BizTalk project

Use Visual Studio 2010 to create the BizTalk project. Remember to open Visual Studio as administrator. In Visual Studio, create a new, empty BizTalk Server project. Name the project TCP_Read_SO.

Run the WCF Service Consuming Wizard

Use the WCF Service Consuming Wizard to generate the artifacts that you need for this scenario, including schema files and binding information files:

1. In **Solution Explorer**, right-click the project name, then point to **Add**, and then click **Add Generated Items**.
2. In the **Add Generated Items** dialog box, double click **Consume WCF Service**. The WCF Service Consuming Wizard opens. Click **Next** until you advance to the **Metadata Endpoint** page.
3. Enter the address of the URI of the WSDL document. This address is displayed in the **Inbound ports** form in Microsoft Dynamics AX. For example, enter:

   ```
   http://MyServer:8101/Microsoft DynamicsAx/Services/TCPSalesOrderRead
   ```

   Note that the default WSDL port is 8101 and the URI ends with the name of the integration port.
4. On the **Import WCF Service Metadata Summary** page, click **Import**. The wizard imports the service information and generates the schema and binding files.
5. Click **Finish** to close the wizard.

The wizard creates an orchestration file, named TCP_Read_SO.odx, which you do not need for this scenario. You can delete this file.

Understanding the schemas

When the Consuming WCF Services Wizard generates schema files, it creates long file names that include the service port name and some of the namespace information. For example:

```
TCPSalesOrderRead_schemas_microsoft_com_Microsoft Dynamics_2008_01_sharedtypes.xsd
```

The remainder of this white paper will not refer to these files by their full names. Instead, files will be referred to by their root schema name. For example, the preceding schema file will be referred to as "sharedtypes.xsd, " or "the shared-types schema."

The services schema (not the framework services schema) contains two possible root nodes:

- **SalesOrderServiceReadRequest**, which is the root element for the request that you send to AIF.
- **SalesOrderServiceReadResponse**, which is the root element in the response that you receive from AIF.

A document that uses this schema will contain only one of these root elements.

Create the map

In order to create the request document, you must merge the entity key list, which you received as a response when you created a sales order, into a document that is based on the services schema. The key goal is to contain the entity key list in the **SalesOrderServiceReadRequest** element, while ensuring that the correct namespace is referenced for the services schema. To merge these documents, you can use a BizTalk map.

Add a new map to your BizTalk project in Visual Studio. Accept the default file name, Map1.btm. Configure the map to look like the following screen image:

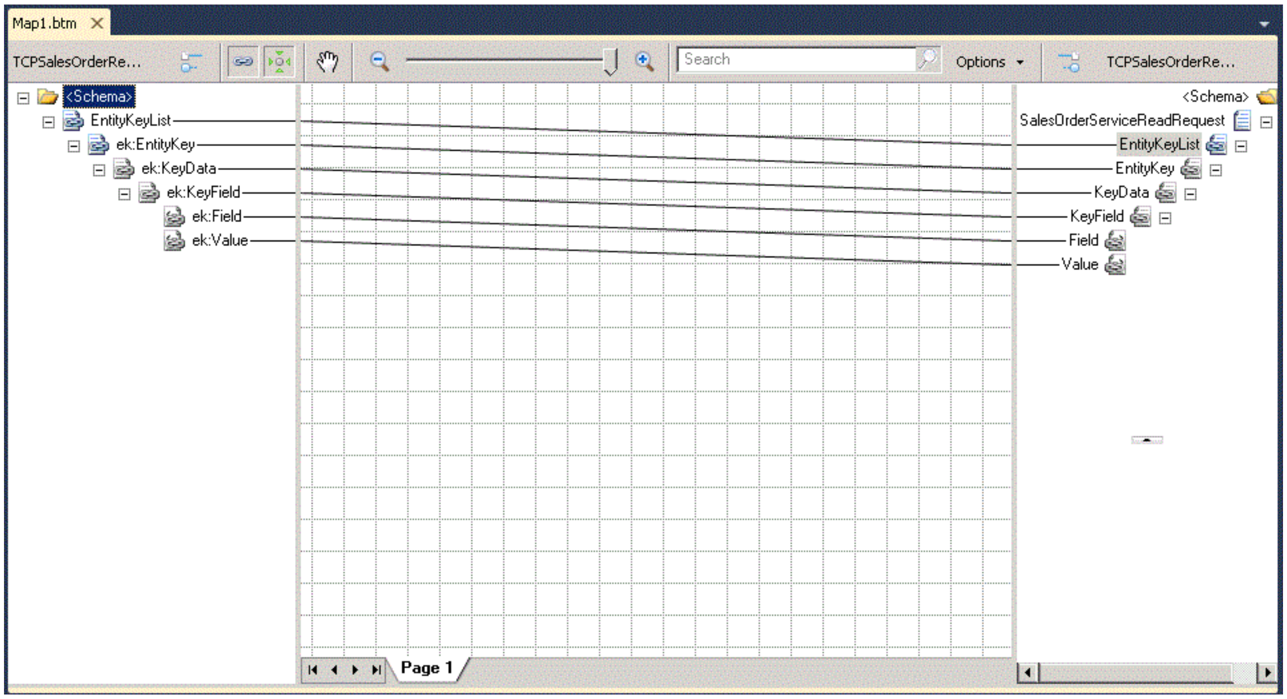


Figure 7: Configure the map

The mapping contained in the screen image provides a one-to-one mapping between the **EntityKeyList** elements in each schema, and their descendants. The schema on the left is the **EntityKeyList** schema. For example:

```
TCPSalesOrderRead_schemas_microsoft_com_Microsoft Dynamics_2006_02_documents_EntityKeyList.xsd
```

The schema on the right is the **services** schema. For example:

```
TCPSalesOrderRead_schemas_microsoft_com_Microsoft Dynamics_2008_01_services.xsd
```

Deploy the BizTalk application

Use Visual Studio to deploy the application. Remember to provide a unique name for the application, select **True** for **Restart Host Instances**, and sign the assembly.

Configure the BizTalk Server Administration console

Use the console to host and run your BizTalk application. To configure the application, open the BizTalk Server Administration console and then follow these steps:

Import the bindings file

1. Right-click the application name, point to **Import**, then click **Bindings**.
2. Browse to the bindings file that the WCF Consuming Services Wizard created. For example, select TCPSalesOrderRead.BindingInfo.xml.
3. Click **Open**.

You can view the imported bindings file information in the administration console, in the **Resources** node for your application.

Create the BizTalk receive port

1. Right-click **Receive ports** and add a new one-way receive port. The **Properties** dialog box opens. You can use the default name or provide a name of your own.
2. Select **Receive Locations** and then click **New**.
3. In the Type list, select **FILE**. Then, click **Configure**.
4. In the **File Transport Properties** dialog box, click **Browse** and then browse to the folder that you named RecSOReq. Close the dialog box.
5. In the **Receive Pipeline** list, select **XML Receive**. Then, configure the pipeline to use the **EntityKeyList** schema from this project for the document specification. (This is an instance where the **EntityKeyList** schema exists in the project from the previous scenario, creating a namespace conflict.)
6. Close all of the dialog boxes.

Create the BizTalk send-receive port

For this synchronous exchange, you must use a single send-receive port. This type of port sends the request to Microsoft Dynamics AX and then waits for the response before proceeding with the rest of the BizTalk document exchange process.

1. Right-click **Send Ports** and add a new, static, solicit-response port. The properties dialog box opens.
2. In the Type list, select **WCF-NetTcp**. Then, click **Configure**.
3. In the **WCF-NetTcp Transport Properties** dialog box, replace the default address with the address for the AIF integration port. For example:

   ```
   net.tcp://Your_AOS_Name:8201/Microsoft DynamicsAx/Services/TCPSalesOrderRead
   ```

   Note: The address displayed in the **Inbound ports** form in Microsoft Dynamics AX is not complete. You must replace the string "AOS_HOST_NAME" with the name of your AOS, followed by a colon, and the port number 8201. This port number differs from the WSDL port number, which is 8101.
4. In the **SOAP Action header** text box, enter the following action string:

   ```
   http://schemas.microsoft.com/Microsoft Dynamics/2008/01/services/SalesOrderService/read
   ```

5. Close the dialog box.
6. In the **Send Pipeline** list, select **XML Transmit**. Then, configure the pipeline to use the AIF message schema.
7. In the Receive Pipeline list, select XML Receive.

Add the map to the send-receive port

1. In the SendPortProperties dialog box, click Outbound Maps.
2. Add a new map. In the **Source Document** list, select the **EntityKeyList** schema.
3. In the **Map** list, select **Map1**.
4. In the **Target Document** list, select the **services** schema.

Create the send-receive port subscription

1. Click **Filters**.
2. For **Property**, select **BTS.ReceivePortName**. Set the property value equal to the name of the receive port that you created.
3. Click **OK** to close the dialog box.

Create the send port

Create a send port to save the response file.

1. Right-click **Send Ports** and add a new, static, one-way send port. The **Properties** dialog box opens.
2. In the **Type** list, select **FILE**. Then, click **Configure**.
3. In the **File Transport Properties** dialog box, click **Browse** and then browse to the folder that you named RecSOResp. Close the dialog box.
4. In the Send Pipeline list, select Pass Thru Transmit.

Create the send port subscription.

1. Click **Filters**.
2. For **Property**, select **BTS.SPName**. Set the property value equal to the name of the solicit-response port that you created.
3. Click **OK** to close the dialog box.

Start the application.

In the **BizTalk Server Administration** console, start the application.

Submit an XML document and receive the response message

The source document for this scenario is the processed response message from the previously described asynchronous scenario. This document contains an entity key list that contains the sales ID for the sales order that you created. The following XML code is an example of a processed response message:

```
<?xml version="1.0" encoding="utf-8"?>
   <EntityKeyList xmlns="http://schemas.microsoft.com/Microsoft
Dynamics/2006/02/documents/EntityKeyList">
      <EntityKey xmlns="http://schemas.microsoft.com/Microsoft
Dynamics/2006/02/documents/EntityKey">
         <KeyData>
            <KeyField>
               <Field>SalesId</Field>
               <Value>SO-100029</Value>
            </KeyField>
         </KeyData>
      </EntityKey>
   </EntityKeyList>
```

Save a copy of this document in the folder that you named **RecSOReq**. When the processing is complete, the sales order document that you requested will appear in the folder that you named RecSOResp.

Consuming a BizTalk Web service

BizTalk Server can expose orchestration functionality as a web service. You can write X++ code to consume a web service in Microsoft Dynamics AX.

For more information about how to expose a web service from BizTalk, see How to Use the BizTalk Web Services Publishing Wizard to Publish an Orchestration as a Web Service in the BizTalk Server 2010 help or on TechNet.

For more information about how to consume a Web service from Microsoft Dynamics AX 2012, download the white paper Consuming Microsoft Dynamics AX 2012 Web Services.

Conclusion

BizTalk Server can integrate with Microsoft Dynamics AX through AIF in a variety of synchronous and asynchronous scenarios. You can connect AIF with BizTalk by using the WCF-based adapters that are included with AIF. Your particular requirements will dictate whether to use an asynchronous or synchronous adapter and which adapter to use