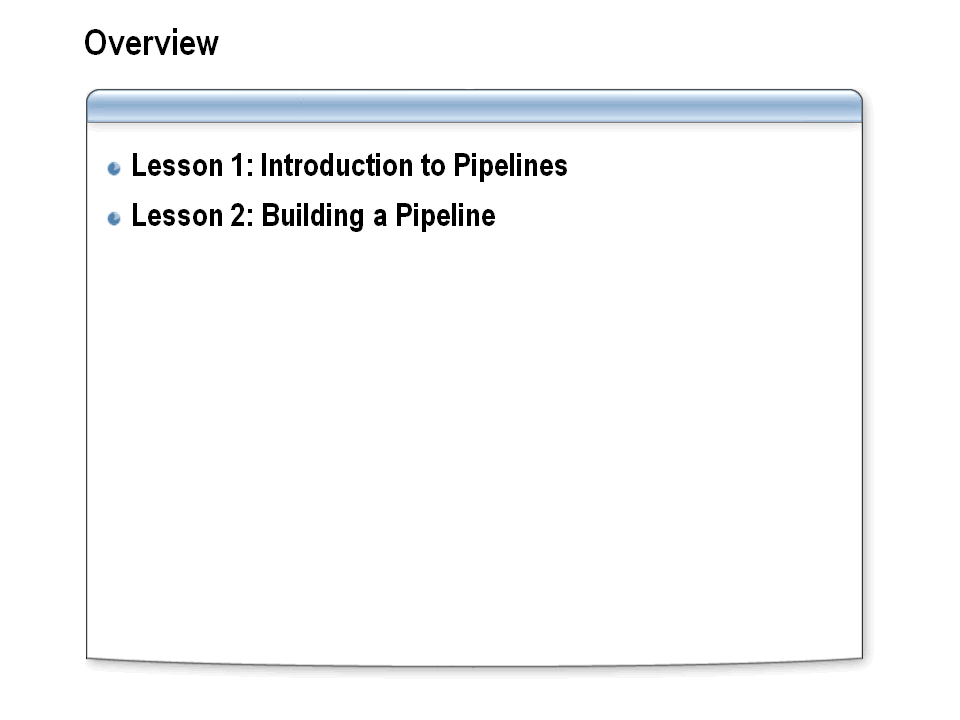
Module 6: Creating Pipelines

##### Time estimated: 120 minutes



Module objective:

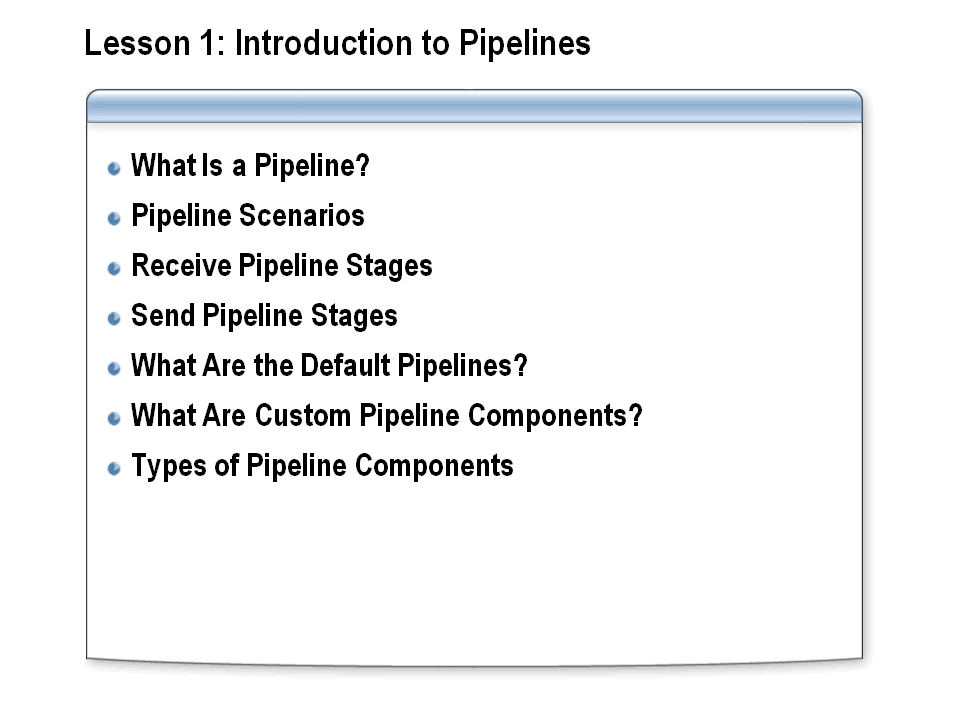
In this module, you will learn how to create send pipelines and receive pipelines for specific processing scenarios.

##### Overview

Pipelines enable the pre-processing of messages as they enter or leave the MessageBox database. Pipelines can also be called from within an orchestration. Pipelines are used to provide additional processing to messages, such as encoding and decoding, encrypting and decrypting, and other processing that might be required to work with messages in Microsoft BizTalk Server.

BizTalk Server supports the ability to modify pipeline properties on a per-use basis, reducing the need to modify pipelines frequently, and the ability to call pipeline components from within an orchestration.

Lesson 1: Introduction to Pipelines



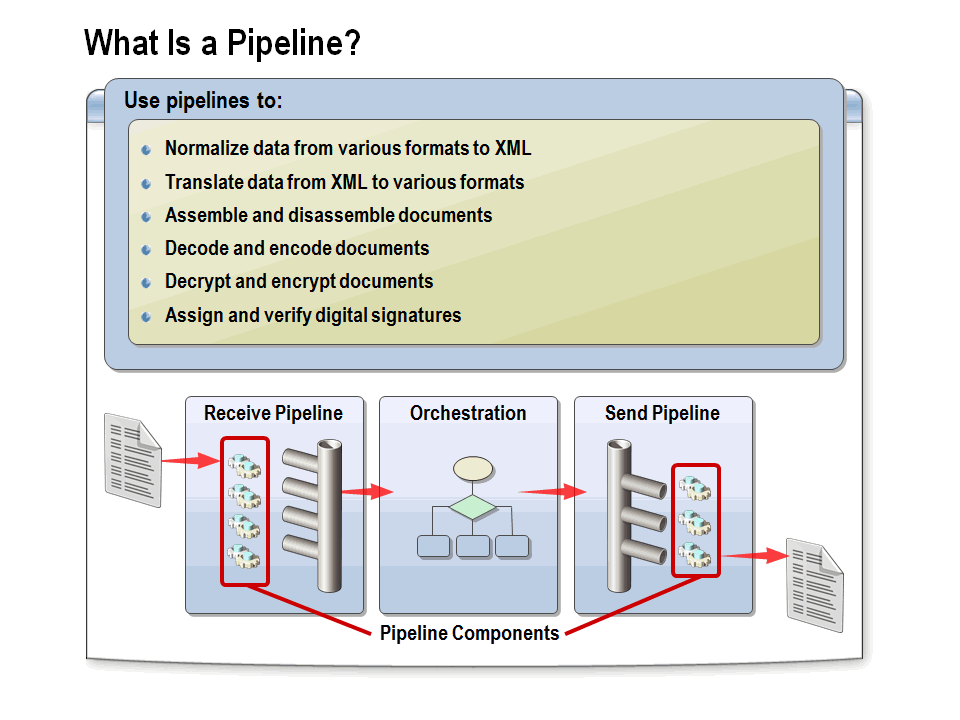
Lesson objective:

Describe both send and receive pipelines and explain how BizTalk Server uses pipelines to process messages.

##### Overview

A pipeline in BizTalk Server is a software infrastructure that defines and links together one or more stages of a business process. Stages define logical work groups, determine which components can belong to a stage, and specify how the pipeline components in the stage are run. Pipelines are executed in sequence to complete a specific task, such as encrypting a document or validating a document against a schema.

What Is a Pipeline?



Describe how pipelines can be used to process messages.

##### Pipelines

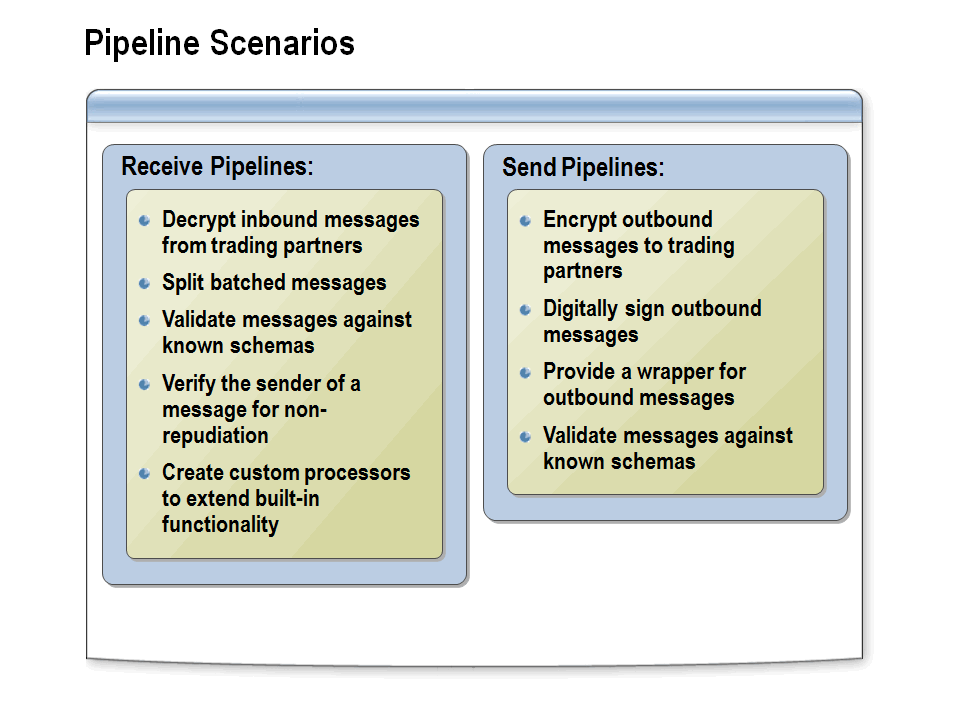
*Pipelines* are software components that can process messages, either as the messages are received or just before they are sent out through a send port. A pipeline divides processing into categories of work called *processing* *stages* and specifies the sequence in which each stage of work is performed. Each stage of a pipeline contains one or more pipeline components (Microsoft .NET objects or COM objects) that can be configured to work with the specific requirements of the messaging solution or orchestrated business process.

##### Processing Stages

Pipeline processing stages can include functions such as decoding or encoding, disassembling or assembling, and decrypting or encrypting. Processing stages are implemented in a prescribed order that cannot be modified.

The processing stages for a pipeline depend upon its intended use. BizTalk Server provides two types of pipelines: receive and send. These two types of pipelines require separate categories of work, such as the encoding versus the decoding of a message. The pipeline also governs the process sequence by the use of policy files that specify the order in which each stage is to be executed. For instance, an incoming message must usually be decoded before it can be disassembled.

Pipeline Scenarios



Describe receive and send pipeline usage scenarios.

##### Overview

There are a number of reasons why you may need to implement pipelines in your BizTalk solution. By using pipelines, it is possible to modify the message in many different ways as it is being passed into and out of the MessageBox database.

##### Receive Pipelines

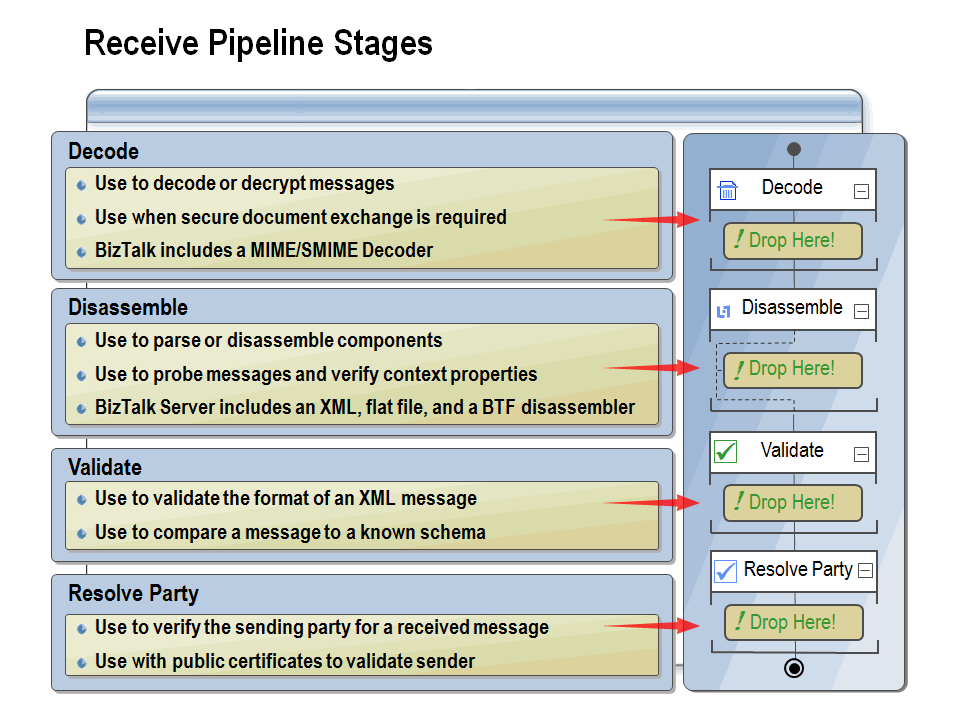
Receive pipelines can be used to process messages as they are received by BizTalk Server. For example, you can use receive pipelines to decode and/or decrypt messages (by using a private key) as well as verify the sender of messages as they are being received. This is important because messages exchanged over the Internet must frequently be encrypted, and it is necessary to confirm the identity of the sender of the message.

Other tasks that can be performed in receive pipelines include extracting individual messages from a message interchange and validating messages against a schema to ensure that they meet the requirements of your processes. Validation can be performed in both a receive pipeline and a send pipeline.

##### Send Pipelines

Send pipelines are used to process messages as they are being sent by BizTalk Server. For example, you can use send pipelines to encrypt messages (using a public key), or digitally sign outbound messages (using a private key) as proof of who the sender is. Before a message is sent, you can also use the validate component of a send pipeline to ensure that a message is valid against a known schema.

Receive Pipeline Stages



Explain how each of the receive pipeline stages works to process messages.

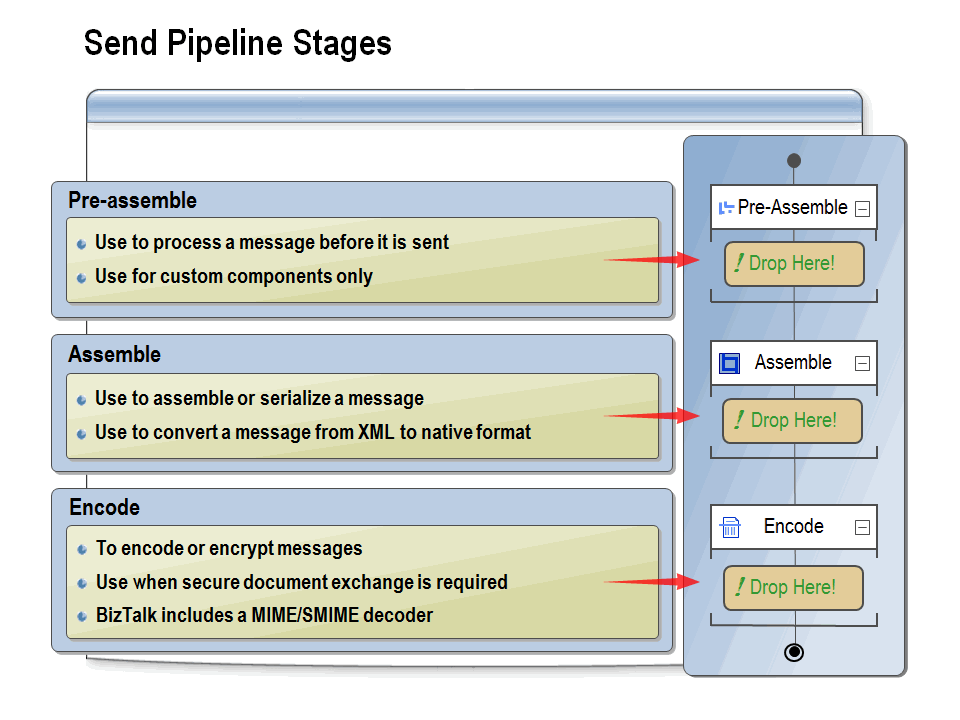
##### Receive Pipeline Stages

Receive pipelines are associated with receive ports (which are discussed in Module 5, “Routing BizTalk Messages”). When a port receives a message via an adapter (or is called from within an orchestration), the message is passed to the pipeline for processing. The receive pipeline parses the initial message, with each component tentatively processing the message. The result of the pipeline process will be zero, one, or more messages that will make their way to the MessageBox and on to various subscribers.

Each of the four stages in a receive pipeline performs a specific function and can contain only components specified for use in that stage. Each receive pipeline stage can contain up to 255 components, which will all be executed in order with the exception of the disassemble stage, in which only one component will execute. The four stages are as follows:

* **Decode**. This stage is used for components that decode or decrypt messages. For example, there is a built-in MIME/SMIME decoder pipeline component that can be used to decode MIME-encoded messages. Custom components for this stage could include a component to decode a compressed (zipped) file before further processing.
* **Disassemble**. Use this stage if you need to parse or disassemble the inbound message. The components within this stage probe the message to see if the message format is recognized, and then, if the message format is recognized, one of the components disassembles the message. Tasks performed in this stage include conversions of flat-file messages to XML format and splitting of messages. In order for property promotion to occur, an appropriate (flat-file or XML) disassembler must be specified in this stage.
* **Validate**. In this stage, messages are validated against a collection of schemas. Pipelines process only messages that conform to the schemas specified in this component, if present. If a message is received by the pipeline whose schema is not associated with any component in the pipeline, the message is not processed. Depending on the adapter, the message is either suspended or an error is issued to the sender. This stage runs once per message created by the Disassemble stage. The built-in validate component can be used in this stage as well as in other stages.
* **Resolve Party**. In this stage, the certificate associated with the sender’s security identifier (SID) is mapped to the corresponding configured BizTalk Server party. If the message was digitally signed, the component uses the signature to look up a Microsoft Windows® identity in the BizTalk Server 2010 Configuration database. If the message carries the authenticated SID of a Windows user, this identity is used. If neither mechanism succeeds, the sender is assigned a default anonymous identity. Party resolution is an important feature for managing trading partner relationships. Not all adapters support party resolution.

Send Pipeline Stages



Explain how each of the send pipeline stages works to process messages.

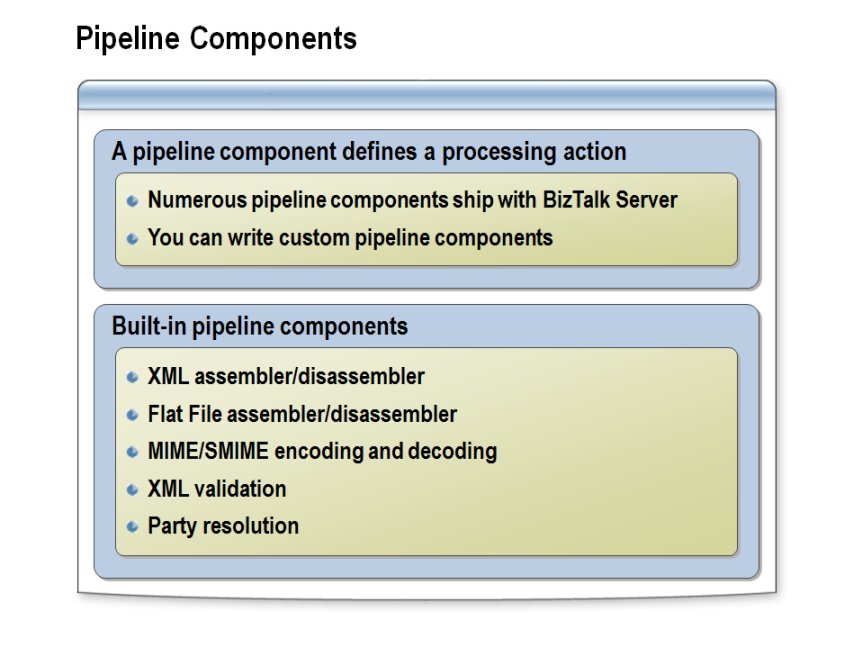
##### Send Pipeline Stages

A send pipeline is responsible for processing documents before they are sent to their final destinations. The send pipeline accepts one message and produces one message for sending. You can create a new send pipeline, which contains empty stages by default, or you can use one of the two default send pipelines included in BizTalk Server: the *Pass-Through Send Pipeline* or the *XML Send Pipeline*.

By default, the send pipeline consists of three empty stages:

* **Pre-Assemble***.* This stage is a placeholder for custom components that should perform some action on the message before the message is serialized. This stage is run once per message and can contain between 0 and 255 components. All components in this stage are run.
* **Assemble***.* In this stage, components are responsible for assembling or serializing the message and converting it to or from XML. This stage accepts either 0 or 1 component, which if present will be executed once for each message. Possible uses for this stage include converting to a flat-file formatted message and placing an envelope wrapper around a message.
* **Encode**. This stage is used for components that encrypt or encode the message, and it runs once per message. This stage can contain between 0 and 255 components, and all components in this stage are executed. If message signing is required, for example, place the MIME/SMIME Encoder component or a custom-encoding component in this stage. You may create a custom component to generate a message as a PDF file or to convert a message to compressed Zip format before sending it.

Pipeline Components



Explain how each of the send pipeline stages works to process messages.

##### Built-in Pipeline Components

The Visual Studio Toolbox is populated with several standard BizTalk Server components that you can use to create a pipeline.

The **XML Disassembler** pipeline component combines XML parsing and disassembling into one component. Its primary functions are: removing envelopes, disassembling the interchange, and promoting the content properties from interchange and individual document levels on to the message context. The XML Assembler component performs the reverse operations.

The **Flat File Disassembler** component parses delimited and positional flat file format messages and converts them into an XML representation. The Flat File Disassembler also removes the header and trailer structures from the flat file message, and breaks the interchange within the message into individual documents. It also promotes properties from the documents and headers. The Flat File Assembler performs the reverse operations.

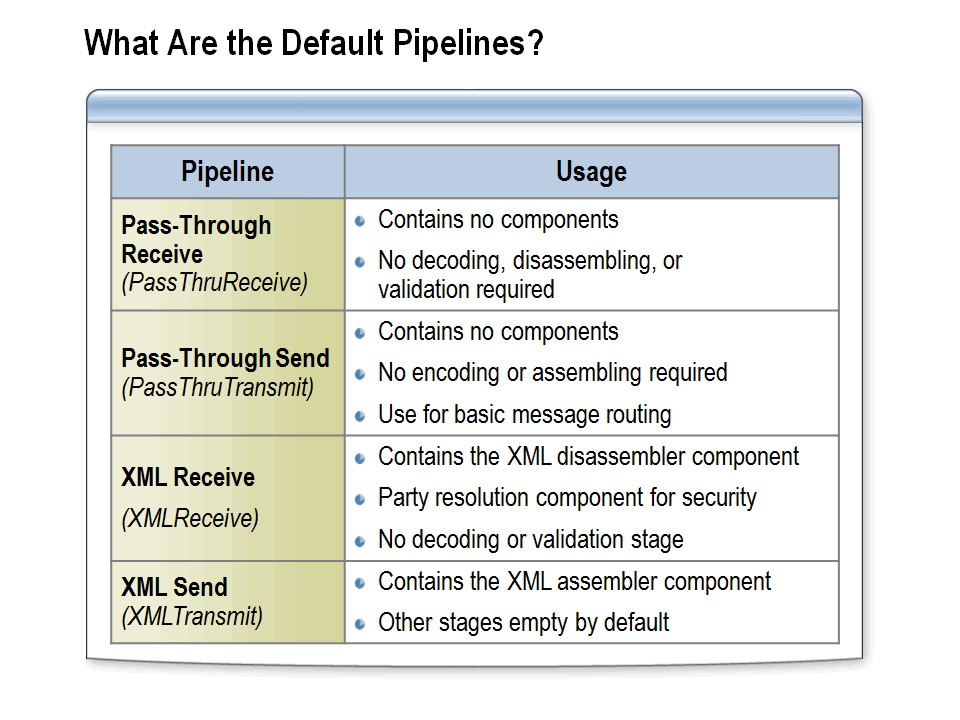
The **MIME/SMIME Decoder** component provides MIME decoding functionality for messages. This pipeline component can be placed into the Decode stage of a receive pipeline, and it supports 7bit, 8bit, binary, quoted-printable, UUEncode, and base64 decoding. Localized data character set changes will not affect the decoding.

The MIME/SMIME Decoder component can decrypt and sign-validate an incoming message. Decryption certificates are used from the personal certificate store of the current user under which the service is running. Sign-validation certificates are used from the Address Book store of the local computer or from the message itself.

The **XML Validator** pipeline component can be used in both send and receive pipelines in any stage except for Disassemble or Assemble. The XML Validator component validates the message against the specified schema or schemas, and if the message does not conform to these schemas, the component raises an error and Messaging Engine places the message in the suspended queue.

The responsibility of the **Party Resolution** pipeline component is to map the sender certificate or the sender security identifier (SID) to the corresponding configured BizTalk Server party.

What Are the Default Pipelines?



Identify each of the default pipelines and describe how they work to process messages.

##### Default Pipelines

When you create a new application, a reference to the Microsoft.BizTalk.DefaultPipelines assembly is automatically added to the project. This assembly contains the default pipelines provided with BizTalk Server. The default pipelines cannot be modified using the Pipeline Designer.

##### XML Receive Pipeline

The XML receive pipeline consists of all four stages; however, the decode and validate stages are empty. The disassemble stage contains the XML Disassembler component, and the ResolveParty stage runs the Party Resolution component, which resolves the certificated subject or the source security ID to the party ID.

##### XML Send Pipelines

The XML send pipeline consists of all three stages, with the assemble stage having an XML assembler and the other two stages being empty. The assemble stage ensures that the outbound XML is well-formed.

##### Pass-Through Receive Pipelines

The Pass-Through receive pipeline contains no components. It is used for simple pass-through scenarios in which no message processing is required. This pipeline is generally used when both the source and the destination of the file are known, and the message requires no validation, decoding, or disassembling. This pipeline is commonly used in conjunction with the Pass-Through send pipeline. Because this pipeline does not have a disassembler component, it cannot be used to route messages to orchestrations or to promote properties from within messages. Property promotion is discussed in detail in Module 5, “Routing BizTalk Messages.”

Pass-Through pipelines are typically used to process non-message data, such as graphics files, Microsoft Word and PDF documents, or files that are sent as attachments. The Pass-Through pipelines treat the message as a blob (raw message data) and simply place the content directly in the MessageBox database.

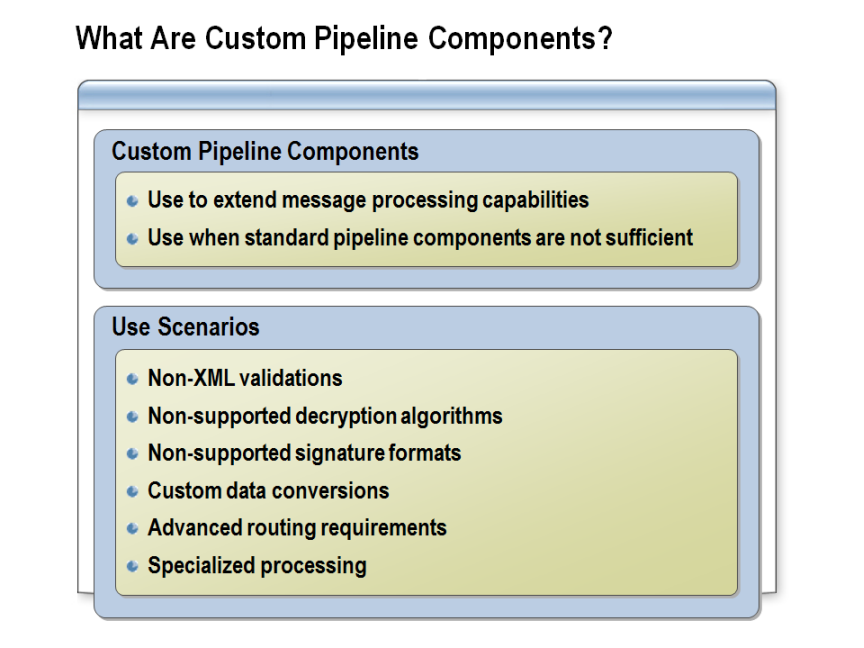
##### Pass-Through Send Pipelines

The Pass-Through Send pipeline contains no components. It is used for simple pass-through scenarios in which no message processing is necessary. This pipeline is the default pipeline that is used when you create a new send port. Because there are no components and no processing takes place, this pipeline is very efficient and should be used whenever practical. This pipeline is frequently used in conjunction with the Pass-Through receive pipeline when the message is not being processed by BizTalk but is simply being routed through BizTalk Server.

##### Flat-File Pipelines

There are no default flat-file pipelines. Any time flat-file messages need to be processed, it will be necessary to create a custom pipeline as outlined in the following topics. When any flat-file message is to be processed through BizTalk Server, the pipeline will have at least one disassembler in which the document property will identify the flat-file message format that the disassembler can process. Each flat-file disassembler can specify only one document schema; therefore, if the pipeline can process multiple flat-file message formats, multiple disassemblers will be required.

What Are Custom Pipeline Components?



Explain when to use a custom pipeline and describe usage scenarios for custom pipelines.

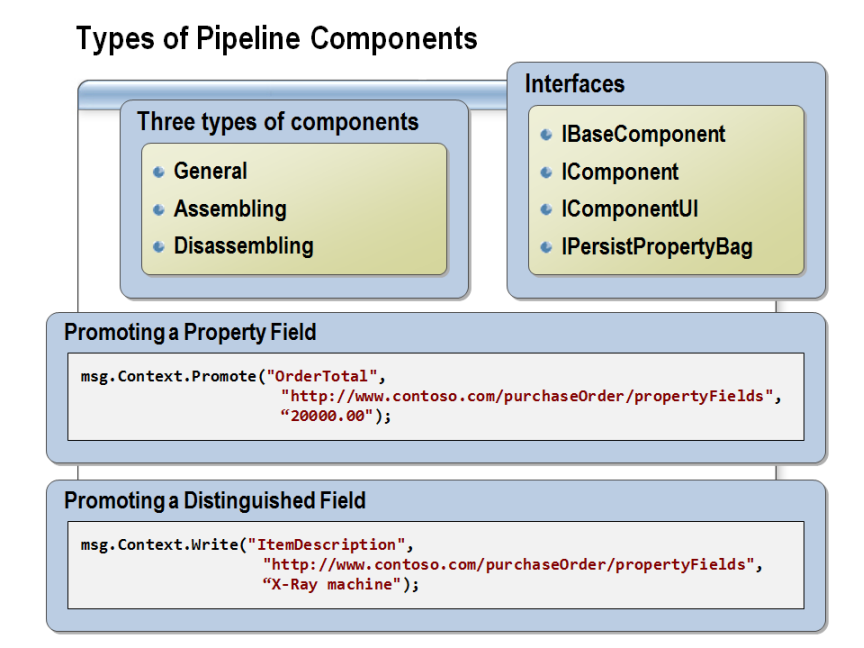
##### Custom Pipeline Components

BizTalk Server pipelines allows you to customize the processing of documents received or sent via the various adapters. Custom pipeline components extend the behavior of the default pipelines to enable the processing of virtually any data format. Custom pipelines can be a powerful solution if you support legacy systems that require integration with other products but do not follow standards. For example, your data may contain carriage returns in odd places or records that span multiple lines of text.

Examples of when to consider using a custom pipeline include:

* Validations that cannot be expressed in an XML schema.
* Decryption algorithms not supported by BizTalk Server out of the box.
* Checks on signature formats that BizTalk Server does not yet recognize.
* Conversions that might not be possible by using the BizTalk Mapper.
* A custom disassemble algorithm is required to split up your messages.
* Context must be added to a received message to support advanced routing and correlation scenarios.
* Built-in components can’t process a message, as is the case with converting to or from a PDF or Zip file format.

Types of Pipeline Components



Explain the different types of pipeline components and how they are used.

##### Overview

You can create three types of pipeline components: *general*, *assembling*, and *disassembling*. Each of the three types can additionally implement probing functionality. Each type of pipeline component has an associated interface that must be implemented for the component to be plugged into the BizTalk Messaging Engine. In order to develop a general pipeline component, you must implement the following interfaces:

* **IBaseComponent**. This interface contains three functions, one each to retrieve the computer name, description, and version.
* **IComponent**. This interface contains the single core method for implementing the pipeline process that allows you to give read and write access to both the message’s data parts and context.
* **IComponentUI**. The two methods in this component enable you to validate the component’s configuration and provide a design-time environment icon.
* **IPersistPropertyBag**. This interface enables a component to both retrieve and store configuration information for the component by using a property bag (used to persistently save properties).
* **IProbeMessage***.* This interface is required for disassembler components. The pipeline manager will call the disassembler through this interface, allowing the component to examine a message to determine whether or not it can recognize and process the message format.

The built-in pipeline components are streaming components which means that they get one pass at the data as it flows through the pipeline. For performance reasons, and to conserve memory, you should create streaming pipeline components whenever feasible. Non-streaming components will consume more memory and can degrade performance.

##### Implementing a Custom Pipeline Component

To create a custom pipeline component, create a Class Library project in Visual Studio, and then add a new class to it. Mark the class with the ComponentCategory attribute to indicate that this class is a pipeline component. This attribute also indicates which stages of execution are appropriate for this component. Then, you will need to write the code that implements the interfaces described above. The actual message processing will be performed in your IComponent.Execute method.

It is possible to modify messages in a pipeline component, but you must handle messages with care in pipeline components. Keep in mind that messages are generally considered immutable. You must actually clone the message before you can make any changes to it. You can promote properties on the original message, however, without being required to perform the cloning.

When cloning a message, you will need to copy the message, parts, and context. BizTalk provides a few helper classes to simplify this process. The PipelineUtil class provides methods for cloning. The PipelineContext class provides other utility methods and properties.

The PipelineContext is passed to the IComponent.Execute method. It provides details about component location in the pipeline, and it provides methods to access schemas based on type or name. It also provides access to the message factory. You can use the message factory to create new messages, message parts, context and property bags.

PipelineContext also provides handle to current transaction. Your pipeline executes in in the context of the transaction, and the component’s work is committed as part of persisting to the message box.

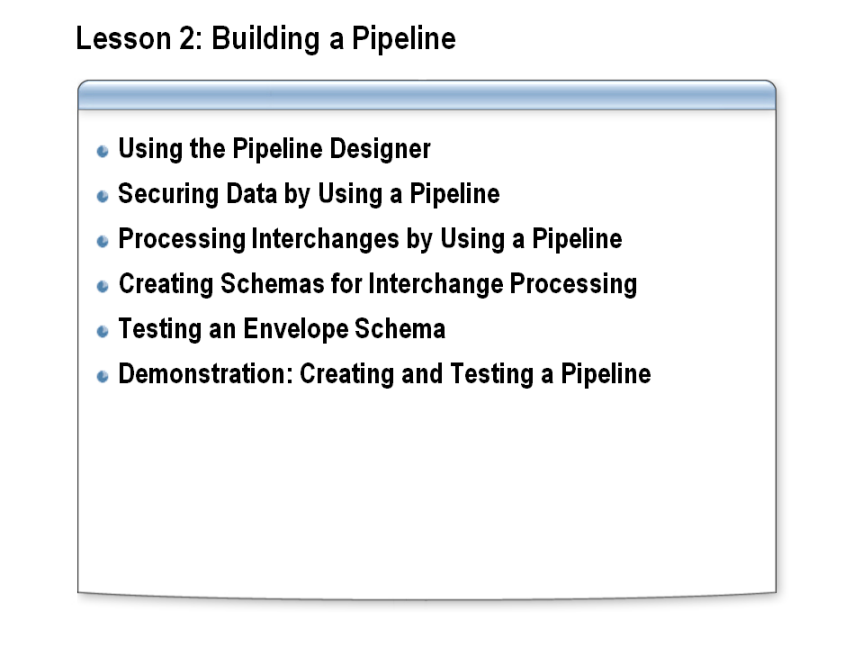
Pipeline components can be deployed in two locations:

• Directory: [BTSINSTALL\_DIR]\Pipeline Components

• Global Assembly Cache (GAC)

BTSInstall\_Dir is primarily used for development, and this is where Visual Studio will look for assemblies used in the pipeline designer. A pipeline component must be deployed to the GAC if it will be used within an orchestration. The recommended practice is to deploy all custom components to GAC.

Lesson 2: Building a Pipeline



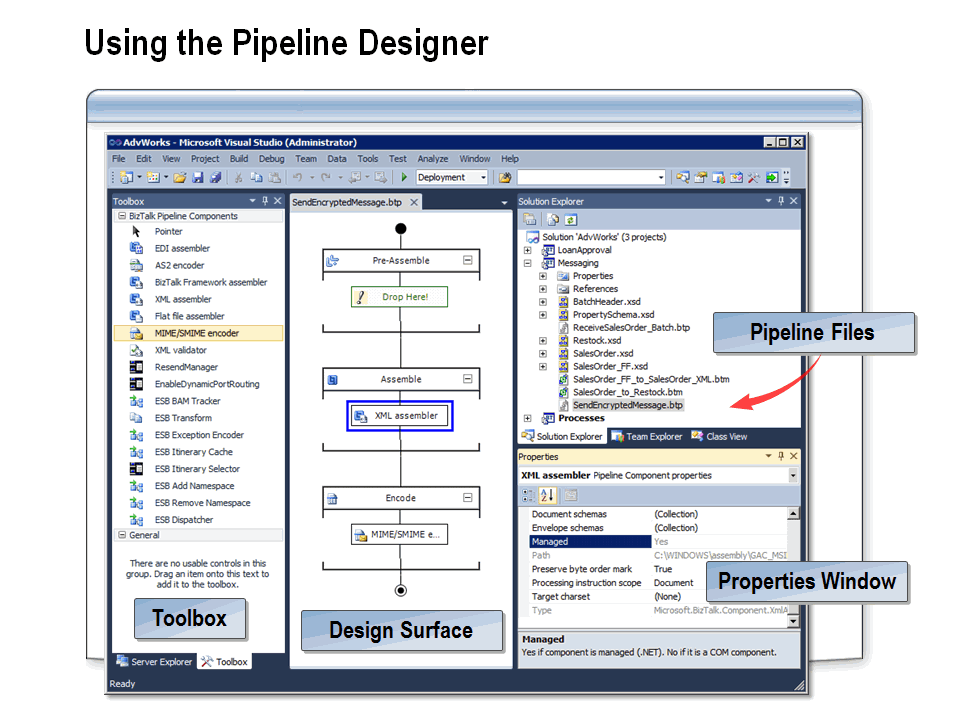
Lesson objective:

Describe and demonstrate how to design and build send and receive pipelines.

##### Overview

The send and receive pipelines that you create can be a very important part of your BizTalk deployment. In this lesson you will learn how to use Visual Studio to design and build pipelines.

Using the Pipeline Designer



Use the Pipeline Designer to create receive pipelines and send pipelines.

##### Pipeline Designer

The BizTalk Pipeline Designer is a feature in Microsoft Visual Studio® 2010. The Pipeline Designer provides a graphical representation of a pipeline and enables you to construct or edit send or receive pipelines for a BizTalk project. You can navigate between pipelines by clicking the tabs at the top of the design surface. The file extension for both receive pipelines and send pipelines is .btp.

You create a new pipeline by selecting a pipeline template for a project. Separate templates are provided for send and receive pipelines. When you name a new pipeline, you should get in the habit of indicating the primary purpose of the pipeline along with the pipeline direction (send or receive), for example, SendEncryptedMessages.

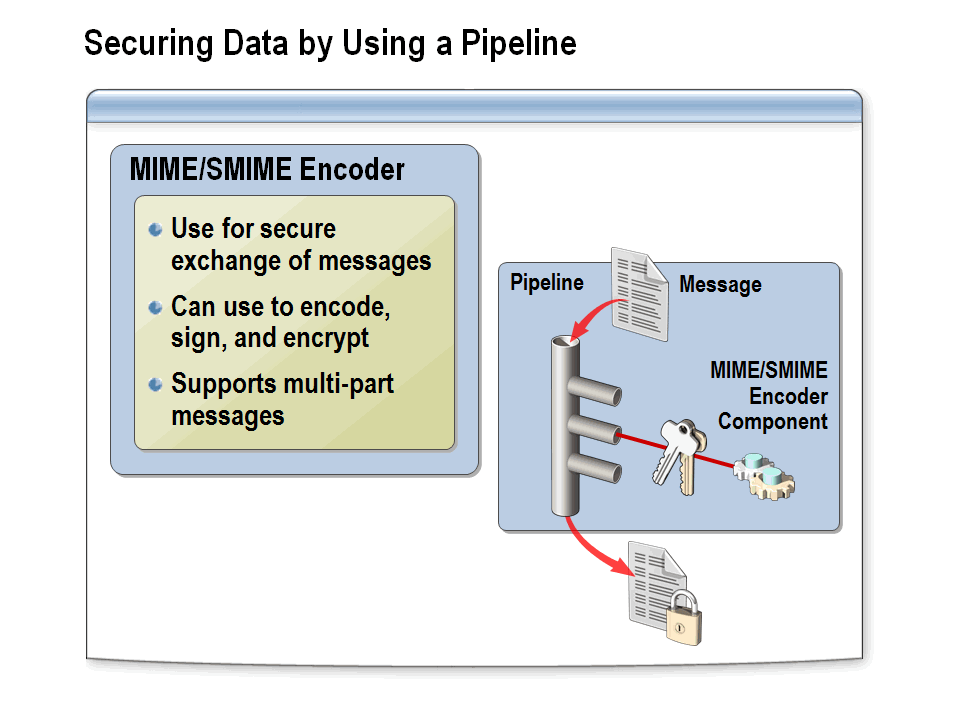
##### Pipeline Stages

You create or modify pipelines by dragging and dropping components to the different pipeline stages. If there are no components in any one stage, a placeholder indicates that shapes can be inserted there from the Toolbox. When the first shape is inserted into a stage, the placeholder text disappears.

The design surface shows the pipeline vertically, running from the start of the pipeline (at the top) to the end of the pipeline. Note that with the exception of the disassemble stage (in the receive pipeline), the execution path goes through each stage indicating that each component in the stage will be executed in order. In the case of the disassemble stage, the path is represented as a dotted line that extends around the components in the stage, indicating that the message will be probed and that the first component that matches will be executed.

Only components that are valid for the type of pipeline that you are creating are displayed. For example, the assembler components can be used only in send pipelines so that they are not visible when creating a receive pipeline. Additionally, the design interface will allow you to add only those components that are valid for a given stage to that stage. For example, you can add only a disassembler to the disassemble stage.

Securing Data by Using a Pipeline



Configure pipelines to enable the secure processing of messages.

##### MIME/SMIME Encoder Pipeline Component

BizTalk Server includes the MIME/SMIME Encoder pipeline component. This component is used to encode messages before they are passed to other business processes outside BizTalk Server. This is useful, for example, when you require the secure exchange of documents between external processes and business partners. The MIME/SMIME pipeline component can be used to either MIME encode, sign, or encrypt outgoing messages with encryption and signing certificates. It also enables the sending of multi-part messages outside of BizTalk Server.

Note: All of the possible properties that you can configure for the MIME/SMIME Encoder pipeline component are listed in the BizTalk Server 2010 documentation under “Configuring the MIME/SMIME Encode Pipeline Component.”

##### Configuring Encoding

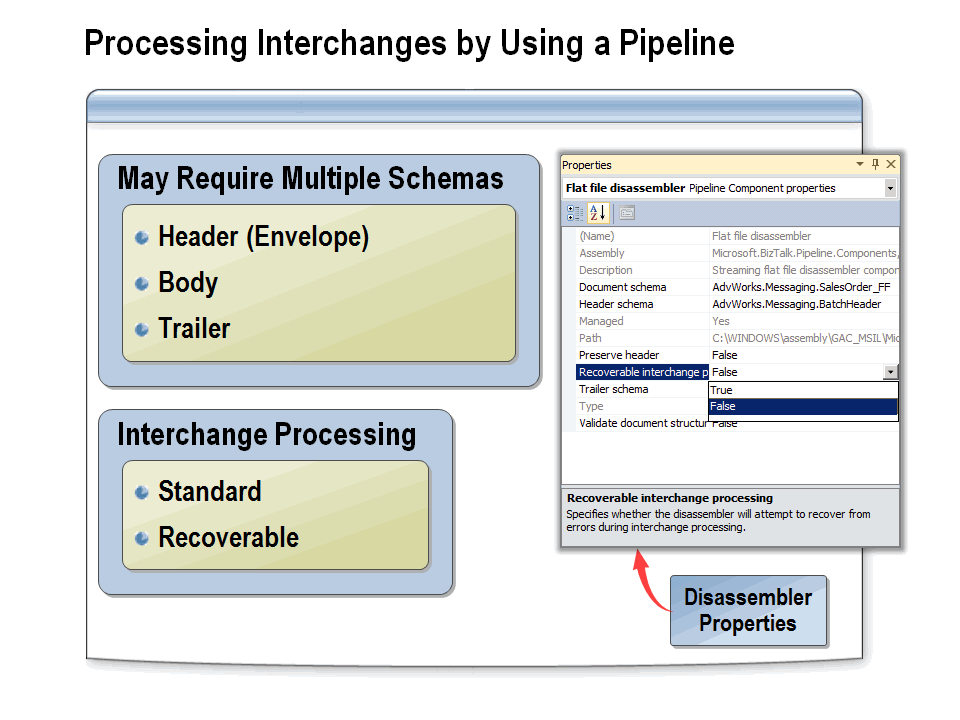
If an encoding component in the send pipeline is configured to sign messages, and the BizTalk group that includes the host running the pipeline is not configured with a signing certificate, the outgoing message will be suspended (with the appropriate error being displayed) to the suspended queue of the host running the pipeline. If the signing certificate cannot be found in the personal certificate store of the current user under which the service is running, the message will be routed to the suspended queue of the host running the pipeline.

If there is an encoding component in the outbound pipeline configured to encrypt outbound messages, and the certificate cannot be found in the certificate store, the message will be suspended to the suspended queue of the host that is running the pipeline.

Also, if you have a request-response port that is receiving signed and encrypted messages, and you want to perform response encryption on this port, you must add a custom pipeline component to the pipeline before the MIME/SMIME Encoder pipeline component that must identify the thumbprint corresponding to the encryption certificate. You will also need to set the BTS.EncryptionCert property on the message context.

Note: In BizTalk 2010, the MIME/SMIME encoder pipeline component will not have native 64-bit support. This means that this component must be run in a 32-bit emulation mode process (WOW64), which implies that the host instance in which this encoder component (or the send pipeline of which it is a part) runs must be running in 32-bit emulation mode. Be aware of the performance (and other) implications of this restriction for other elements of BizTalk Server running in this same host instance.

Processing Interchanges by Using a Pipeline



Configure pipelines to enable the processing of message interchanges.

##### What Is an Interchange?

An *interchange* is a group of messages that are contained within one larger message. Depending upon the design of the messaging or orchestration solution, BizTalk Server can often process many small messages faster than it can process a single large message. For example, you can think of an interchange as a manila envelope that contains ten individual paper purchase order (PO) documents. Once you open the envelope, you will want to process each PO separately. Interchange processing in BizTalk Server is used to extract these individual messages so that they can be processed separately.

BizTalk Server 2010 supports a feature called *recoverable interchange processing*.

##### Standard Interchange Processing

By default, when an interchange arrives at a receive location, the configured pipeline will break down the interchange into one or more messages. Messages are then individually validated by the pipeline and then collected within the EPM (End-Point Manager) inside BizTalk Server. If, at any time, any message within the collection fails, the entire interchange is suspended. The suspended message will appear as the complete original interchange, not the separate parts.

Consider a scenario that contains 10 messages, all of which the disassembler successfully extracts from the interchange:

1. The first five messages propagate through the pipeline and are ready to be published.
2. The sixth message fails to process at the disassembling stage in the pipeline, which causes all of the other messages that have already been processed to roll back and the original interchange message to be suspended as resumable.
3. Nothing is published. The original interchange (in this case, all ten messages) is suspended because in standard interchange processing, any extracted message that fails at any point during or after interchange disassembly results in all extracted messages being discarded and the original interchange being placed on the suspend queue as resumable. Suspended messages can be viewed by using the BizTalk Group Hub, and notification of the offending message can be sent by using Microsoft Systems Center Operations Manager (SCOM). By default, failed messages cannot be subscribed to by end points such as an orchestration or a send port.

Note: Standard interchange processing is the default for XML receive pipelines in BizTalk Server 2010.

##### Recoverable Interchange Processing

Recoverable interchange processing is a feature of BizTalk Server that supports additional flexibility in how multi-message interchanges are processed. When a new interchange arrives, and the Recoverable Interchange property is set to true, the message is broken down into individual messages and passed through the pipeline for disassembly and validation. Unlike standard interchange processing, messages are individually validated and individually sent by the EPM to the MessageBox. If any message has an error, it is individually suspended in the original format. This means that if a message comes in as a flat file, it will be suspended as a flat file, not parsed as XML.

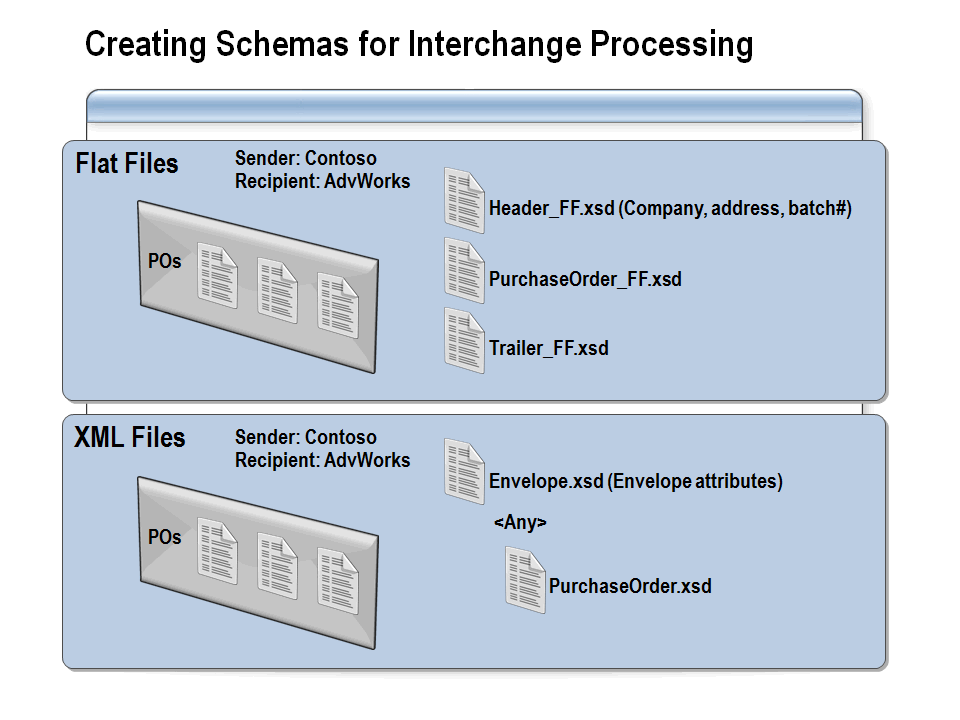
All messages that pass validation are not affected by any failed messages. Failed messages will show up as individual suspended messages within the BizTalk Administration console. As with standard message processing; inbound suspended messages can be resumed if the suspending error is corrected.

Consider the same example that was used previously with the exception that the disassembly stage is configured to perform *recoverable interchange* processing. The result is that individual extracted messages are all processed, and the original interchange is discarded. The individual messages are processed as follows:

1. The first five messages propagate through the pipeline and are ready to be published.
2. The sixth message fails disassembly processing and is suspended.
3. The seventh message and all other messages in the interchange propagate through pipeline and are ready to be published.
4. After all messages are extracted from the interchange, messages 1, 2, 3, 4, 5, 7, 8, 9, and 10 are published into the MessageBox, and message 6 is placed on the suspended queue.

Note: BizTalk Server 2010 supports another feature that is helpful in this case as well: the feature allows not only the suspension of the failed message, it supports subscription to these failed messages as well.

Creating Schemas for Interchange Processing



Configure schemas to enable interchange processing of flat files and XML files through a pipeline.

##### Overview

A message interchange can be a single XML document that contains other messages, or it may be a collection of flat-file messages contained within a single flat-file message. BizTalk Server has different requirements for processing flat-file interchanges than for XML interchanges. As does all flat-file processing, flat-file interchanges require a custom pipeline, whereas XML interchanges can use the default XML pipeline. Additionally, you can create a custom XML pipeline to process interchanges.

##### Flat-File Interchanges

Processing of flat-file messages always requires a custom pipeline containing the flat-file disassembler. A pipeline for flat-file interchanges requires you to configure a flat-file disassembler that specifies the schemas that make up the entire interchange message. This will necessarily include a document schema and may optionally include header and/or trailer schemas.

The document schema needs to represent the individual messages (which may be one or more records) to be extracted by the disassembler. The Header schema must define any records that occur before individual documents in the interchange. The trailer defines any records in the interchange that appear after the end of the last document message. After adding the flat-file disassembler component to a custom receive pipeline, set the Document schema, Header schema, and Trailer schema properties of the flat-file disassembler pipeline component to schemas you have created that match the header, body, and trailer of the interchange that will be processed. Each flat-file disassembler must be associated with only a single message type, so it is necessary to create a separate receive pipeline for each type of flat file that your BizTalk Server system will be processing.

Once the interchange is processed, each resulting message will be sent on through the remaining stages (validation and party resolution) individually.

##### XML Interchanges

Since BizTalk Server comes with a default XML receive pipeline, it is typically not necessary to create custom pipelines to process XML documents. BizTalk Server evaluates each received XML message to determine its document type (this is the combination of the namespace and root node) and then locates any deployed schema that matches the message instance. In this way, the standard pipeline can be used to process any type of XML message. If a custom pipeline is defined with a collection of XML schemas specified in the XML disassembler, that pipeline can be used with only the specified message types. All other messages will be suspended. This is useful if you want to limit the types of messages processed through a given pipeline.

When a message that represents an interchange is received, BizTalk Server evaluates the document type of the message and then locates the appropriate schema (known as the *envelope* schema). Envelope schemas have two special BizTalk Server properties that specify that these messages will be split into individual messages to be processed and saved to the MessageBox. The first property is the Envelope property, which you will find in the reference section of the Properties window when the Schema node is selected. The Root node of the schema has a Body XPath property in the Parse section, which is used only when the schema is an envelope. This expression should be set to identify the immediate parent node of the body messages.

For example, assume that you must process the following message, which contains multiple updates. You want to split up the batch so that each update is processed separately by an orchestration that can process only a single update at a time.

<Batch Department=“Sales">

<Updates>

<CustomerUpdate CustID=“12345" Address=“123 My Street" />

<CustomerUpdate CustID=“12346" Address=“246 Your Street" />

<CustomerUpdate CustID=“98765" Address=“564 Any Street" />

</Updates>

</Batch>

You must first create a CustomerUpdate.xsd schema. This schema has no special properties and could in fact be the same schema that you are processing individual updates with already:

<CustomerUpdate CustID=“12345" Address=“12345 My Street" />

Next, you must create an envelope schema that represents the rest of the message with an Any Element element in place of the <CustomerUpdate> section. The Body XPath expression would identify the <Updates> section.

<Batch Department=“Sales">

<Updates>

<ANY Element>

</Updates>

</Batch>

When these two schemas are deployed and an interchange message is received, BizTalk Server will identify the envelope schema, and when it encounters the Body XPath node, it will emit one message for each child within that node.

If this same envelope is used to process other types of updates, vendor updates for example, only the vendor update schema needs to be created. A single update message can now contain both the vendor and customer updates.

Testing an Envelope Schema



Using BizTalk Server tools to test envelope schemas.

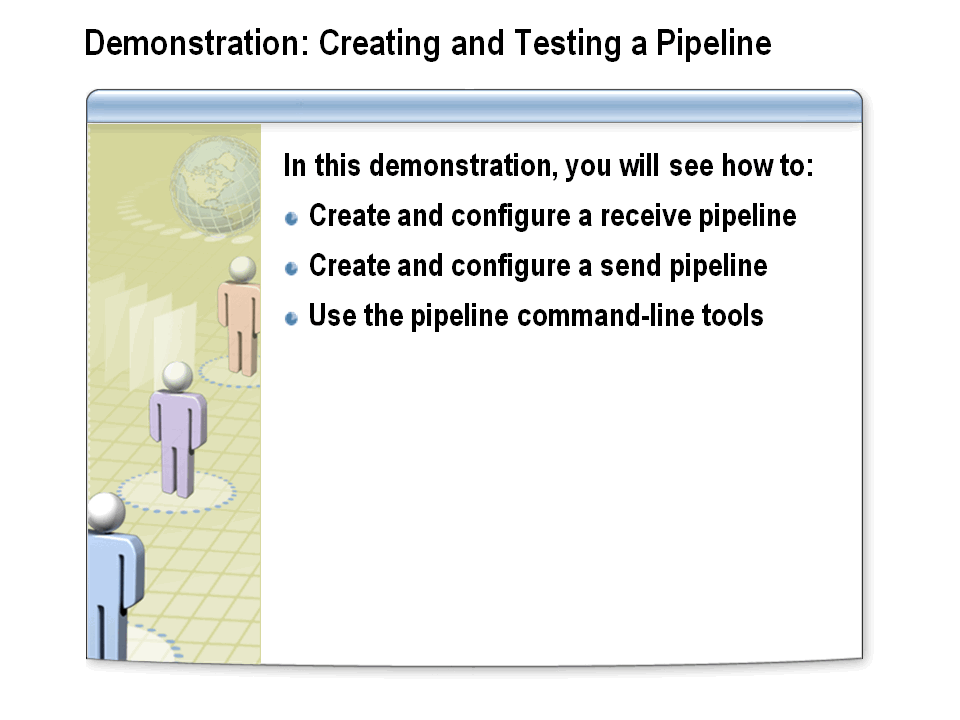
##### Overview

*Pipeline Tools* is a set of tools that are used for execution, debugging, and profiling of pipelines and pipeline components, namely flat-file and XML assembler and disassembler components. The individual tools are as follows:

* **Pipeline.exe**. This tool executes a specific send or receive pipeline. It accepts one or more input documents and their parts, XSD schemas and their related information, and after a pipeline execution, it produces an output document.
* **FFAsm.exe**. This tool executes the flat-file assembler component, directly invoking it, and then it emulates a send pipeline, thereby enabling the user to see how the send pipeline processes (serializes/assembles) the user’s XML document into a flat-file document.
* **FFDasm.exe**. This tool executes the flat-file disassembler component, directly invoking it, and then emulates a receive pipeline, thereby enabling the user to see how the receive pipeline processes (parses/disassembles) the user’s flat-file document into one or more XML documents.
* **XMLAsm.exe**. This tool executes the XML assembler component, directly invoking it, and then emulates a send pipeline, thereby enabling the user to see how the send pipeline processes (serializes/assembles/envelopes) the user’s XML document into an output XML document.
* **XMLDasm.exe**. This tool executes the XML disassembler component, directly invoking it, and then emulates a receive pipeline, thereby enabling the user to see how the receive pipeline processes (parses/disassembles/un-envelopes) the user’s XML document into one or more XML documents.

Note: You can find these tools in the <Installation Path>\SDK\Utilities\PipelineTools folder.

Demonstration: Creating and Testing a Pipeline



Learn how to create a receive pipeline and a send pipeline and use the command-line tool to test a pipeline.

##### Create a Receive Pipeline

1. In Windows Explorer, navigate to **C:\AllFiles\DemoCode\Module6\Demo**, and then double-click **Demo.sln**.

The following demonstration is not dependent upon completion of the previous demonstrations. This solution provides artifacts and file paths that differ from those used in the previous demonstrations.

1. In Solution Explorer, right-click the **Messaging** project, point to **Add**, and then click **New Item**.
2. In the **Add New Item** dialog box, click **Receive Pipeline**, rename the pipeline **ReceivePurchaseOrders.btp**, and then click **Add**.
3. Click the **Decode** stage.

The Decode stage can contain up to 255 individual components, all of which will be executed in sequence.

1. Drag a **MIME/SMIME decoder** component from the **Toolbox** to the **Decode** stage of the pipeline.

The MIME/SMIME Decoder component can decrypt and/or sign-validate an incoming message.

1. Click the **MIME/SMIME decoder** component.
2. In the Properties window, change the **Allow non MIME message** property to **True**.

This property indicates whether to allow non-MIME messages to be passed to the decoder.

1. Click the **Disassemble** stage.

The Disassemble stage can contain up to 255 individual components, of which the first matching component will be the only one executed.

1. Drag an **XML disassembler** component and a **Flat file disassembler** component from the **Toolbox** to the **Disassemble** stage of the pipeline.

The XML disassembler component can be used to remove message envelopes, disassemble interchanges, and promote message content to message context.

The flat file disassembler component is used to parse raw data into one or more documents and optionally remove headers and/or trailers.

Only one of the components will run in the Disassemble stage. If the XML disassembler reports that it cannot process the message, the message will be passed on to the Flat file disassembler.

1. Click the **Flat file disassembler** pipeline component.
2. In the Properties window, from the **Document schemas** drop-down list, select **Demo.Messaging.SalesOrder\_FF**.

This schema represents the message format of the flat file.

1. Click the **Validate** stage.

The Validate stage can contain up to 255 individual components, all of which will be executed in sequence.

1. Drag an **XML validator** component from the **Toolbox** to the **Validate** stage of the pipeline.

Use the XML validator pipeline component to verify the message against the specified schema or schemas. If the message does not conform to these schemas, the component raises an error, and the message is placed in the suspended queue.

1. Click the **XML validator** component.
2. In the Properties window, click the **ellipsis** (**…**) button next to **Document schemas**.
3. In the **Schema Collection Property Editor** dialog box, click **Demo.Messaging.SalesOrder\_FF**, click **Add**, and then click **OK**.

The disassembler components do not validate the message during processing. Add the XML validator component to ensure that the message conforms to the document schema.

Setting the XML dissassembler’s Validate Document Structure property to true will only instruct it to check for well-formed XML, it will not enable validation against a schema.

1. Click the **ResolveParty** stage.

The ResolveParty stage can contain up to 255 individual components, all of which will be executed in sequence.

1. Drag a **Party resolution** component from the **Toolbox** to the **ResolveParty** stage of the pipeline.

The Party Resolution pipeline component maps the sender’s security identifier to the corresponding BizTalk Server party.

##### Create a Send Pipeline

1. In Solution Explorer, right-click the **Messaging** project, point to **Add**, and then click **New Item**.
2. In the **Add New Item** dialog box, click **Send Pipeline**, rename the pipeline **SendProcessedOrders.btp**, and then click **Add**.
3. Click the **Pre-Assemble** stage.

The Pre-Assemble stage can contain up to 255 individual components, all of which will be executed in sequence. There are no standard pipeline components for the Pre-Assemble stage. If you require processing in the Pre-Assemble stage, you will need to create a custom pipeline component to do so.

1. Click the **Assemble** stage.

The Assemble stage can contain up to 255 individual components, all of which will be executed in sequence.

1. Drag an **XML assembler** component from the **Toolbox** to the **Assemble** stage of the pipeline.

The XML assembler component inserts the values of promoted properties in to the message body. It can also be used to insert processing instructions, and to create interchange messages.

1. Click the **Encode** stage.

The Encode stage can contain up to 255 individual components, all of which will be executed in sequence.

1. Drag a **MIME/SMIME encoder** component from the **Toolbox** to the **Encode** stage of the pipeline.

The MIME/SMIME encoder component can be used to encode, sign, or encrypt an outgoing message with encryption and signing certificates.

1. On the **File** menu, click **Save All**, and then close the two pipelines.

##### Execute the Assembly/Disassembly Components with Command-line Tools

1. In Solution Explorer, double-click **InterchangeMessage.xml**.

Notice that the message has an ItemsList section, which contains sender and batch number information. The message also has an Items node, which contains four Item nodes.

1. In Solution Explorer, double-click **InterchangeEnvelope.xsd**.

Notice that the InterchangeEnvelope schema has an ItemsList record, which contains the ItemHeader and Items nodes. The Items node contains an Any Element.

1. In Solution Explorer, double-click **InterchangeBody.xsd**.

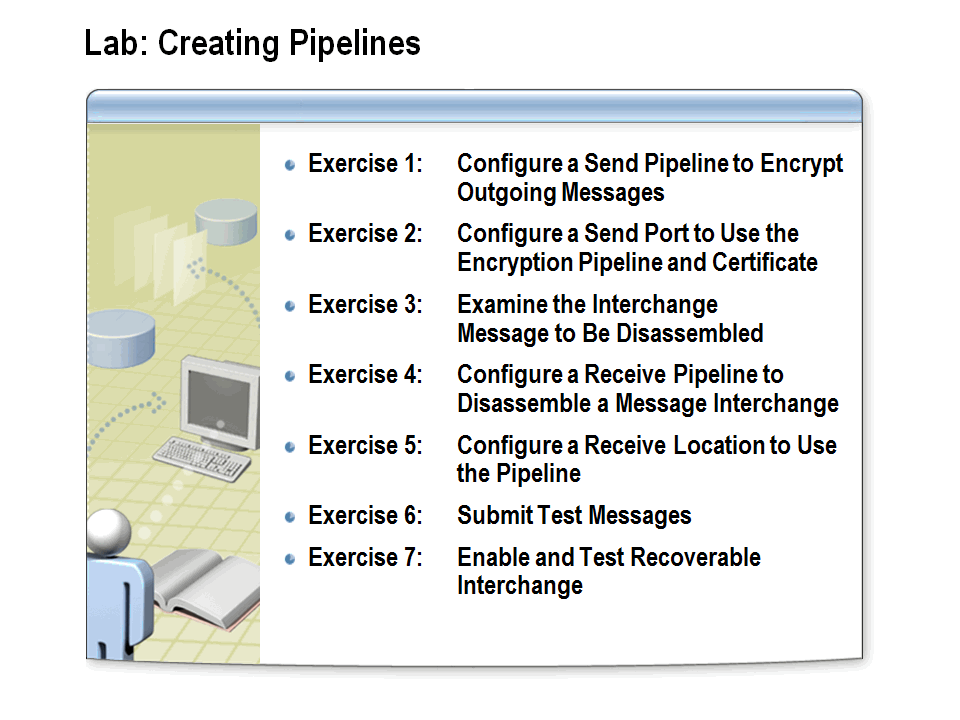
Notice that the InterchangeBody schema contains the Item record, which represents the individual messages of the interchange.

1. On the **Start** menu, click **Run**.
2. In the **Open** box, type **cmd**, and then click **OK**.
3. In the command prompt window, type **path = %path%;”C:\Program Files (x86)\Microsoft BizTalk Server 2010\SDK\Utilities\PipelineTools”**, and then press ENTER.
4. In the command prompt window, type **cd** **C:\AllFiles\DemoCode\Module6\Demo\Messaging**, and then press ENTER.
5. In the command prompt window, type **xmldasm InterchangeMessage.xml -ds InterchangeBody.xsd -es InterchangeEnvelope.xsd –c**, and then press ENTER.

The preceding command uses the xmldasm utility; xmldasm is used to simulate the use of an XML disassembler component in a pipeline. The parameters used are: message (InterchangeMessage.xml), -ds (document schema = InterchangeBody.xsd), -es (envelope schema = InterchangeEnvelope.xsd) and –c (display results in the console).

1. Four messages representing the four items in the list are displayed.
2. Close the Visual Studio solution and all other open windows.
3. Shut down the **bt10d-demos** virtual machine.

Lab: Creating Pipelines



##### Time estimated: 60 minutes

##### Scenario

Pipelines allow you to customize the processing of messages within send or receive ports. In this lab, you will create and test a custom send pipeline that encrypts communications between Adventure Works and Woodgrove Bank. Next, you will configure a receive pipeline that splits a batch of messages (an interchange) to be processed as individual messages. Finally, you will enable recoverable interchange processing to address issues that arise from malformed messages within the batch.

##### Start the Virtual Machine

Procedure List

1. If the **Server Manager** window is not already open, click on the **Server Manager** icon located in the task bar next to the **Start** button.
2. Expand **Roles, Hyper-V, Hyper-V Manager.** The last node to appear displays the machine name. Click on it to see the list of virtual machines available.
3. Double-click the virtual machine **bt10d-06** to open a **Virtual Machine Connection** window.
4. Click on the **Action** menu in the Virtual Machine Connection window and choose **Start.**
5. Once the virtual machine starts, press **CTRL+ALT+END**.
6. Log on using the user name **Administrator** and the password **pass@word1**.
7. At the Windows Activation prompt, click **Ask Me Later**, then click **OK**.

##### Ensure that the BizTalk Services are started

Procedure List

1. In Windows Explorer, navigate to **C:\AllFiles**.
2. Double click on **startBtServices.cmd**.
3. When prompted, press any key to close the command-line window.

Exercise 1: Configure a Send Pipeline to Encrypt Outgoing Messages

##### Overview

Communicating with trading partners often requires that messages be encrypted. In this exercise, you will create a custom pipeline that will be used to encrypt messages being exchanged with Woodgrove Bank. You will then deploy the project with the newly added pipeline.

##### Add a New Send Pipeline to the Messaging Project

Procedure List

1. In Windows Explorer, navigate to **C:\AllFiles\LabFiles\Lab6\AdvWorks**, and then open the **AdvWorks.sln** file.
2. In Solution Explorer, right-click the **Messaging** project, point to **Add**, and then click **New Item**.
3. In the **Add New Item – Messaging** dialog box, in the left pane, click **Pipeline Files**, and then in the center pane, click **Send Pipeline**.
4. In the **Name** box, type **EncryptionPipeline.btp**, and then click **Add**.

The newly added pipeline opens in the Pipeline Designer.

##### Add an XML Assembler Pipeline Component to the Pipeline

Procedure List

1. Drag the **XML assembler** from the Toolbox to the **Drop Here** box in the **Assemble** stage of the pipeline.

##### Add and Configure a MIME/SMIME Pipeline Component to the Pipeline

Procedure List

1. Drag the **MIME/SMIME encoder** from the **Toolbox** to the **Drop Here** box in the **Encode** stage of the pipeline.
2. Right-click the **MIME/SMIME encoder**, and select **Properties**.
3. In the MIME/SMIME encoder Properties window, in the **Check revocation list** box, click **False**.
4. In the **Enable encryption** box, click **True**.
5. In the **Send body part as attachment** box, click **True**.
6. On the **File** menu, click **Save** **All**.

##### Deploy the Messaging Project

Procedure List

1. In Solution Explorer, right-click **Messaging**,and then click **Deploy**.

Exercise 2: Configure a Send Port with the Encryption Pipeline and Certificate

##### Overview

In order to process encrypted messages, a certificate must be installed on the computer running BizTalk Server that will process the message. In this exercise, you will install the certificate used to encrypt messages, configure the send port to use the Encryption Pipeline, and specify the encryption certificate.

##### Attempt to Assign an Encryption Certificate

Procedure List

1. On the **Start** menu, click **All Programs**, then click **Microsoft BizTalk Server 2010**, and then click **BizTalk Server Administration**.
2. In the BizTalk Server Administration Console, expand **BizTalk Server Administration**, **BizTalk Group**, **Applications**, and **AdventureWorks**, and then click **Send Ports**.
3. In the right-pane, double-click **CreditOrdersFILE**.
4. In the **Send Port Properties** dialog box, in the **Send pipeline** list, click **EncryptionPipeline**, and then click **Apply**.
5. In the left pane, click **Certificate**, and then in the right-pane, click **Browse**.

Notice that there are no certificates that can be used to encrypt.

1. Click **Cancel**.

##### Install the WoodGrove Encryption Certificate

Procedure List

1. On the **Start** menu, click **Run**.
2. In the **Run** box, type **mmc**, and then press ENTER.
3. In the Console1 window, on the **File** menu, click **Add/Remove Snap-in**.
4. In the **Add or Remove Snap-ins** dialog box, in the **Available snap-ins** list, double-click **Certificates**.
5. In the **Certificates snap-in** dialog box, click **Computer account**, then click **Next**, and then click **Finish**.
6. In the **Add or Remove Snap-ins** dialog box, click **OK**.
7. In the Console1 window, expand the **Certificates (Local Computer)** node.
8. Right-click **Other People**, point to **All Tasks**, and then click **Import**.
9. On the **Welcome to the Certificate Import Wizard** page of the **Certificate Import Wizard**, click **Next**.
10. On the **File to Import** page, click **Browse**.
11. In the **Open** dialog box, navigate to **C:\AllFiles\LabFiles\Lab6\AdvWorks\WoodGroveCert**, click **WoodGrove.cer**, and then click **Open**.
12. On the **File to Import** page, click **Next**.
13. On the **Certificate Store** page, click **Next**,and then click **Finish** to close the **Certificate Import Wizard**.
14. In the **Certificate Import Wizard** message box, click **OK**.
15. In the Console1 window, expand the **Other People** node, and then click **Certificates**.
16. In the **Issued To** column, double-click **Woodgrove**.

Notice that Microsoft Windows does not have enough information to verify this certificate.

1. Click the **Certification Path** tab.

Notice the bottom pane reads: “The issuer of the certificate could not be found.” This indicates that the issuer is not in the list of trusted certificate authorities.

1. Click **OK** to close the Certificate window.

##### Install the WoodGrove Encryption Certificate Chain

Procedure List

1. In the left pane of the Console1 window, click **Trusted Root Certification Authorities**.
2. In the right pane, right-click **Certificates**, point to **All Tasks**, and then click **Import**.
3. On the **Welcome to the Certificate Import Wizard** page, click **Next**.
4. On the **File to Import** page, click **Browse**.
5. In the **Open** dialog box, navigate to **C:\AllFiles\LabFiles\Lab6\AdvWorks\WoodGroveCert**, click **WoodGroveCACertChain.cer**, and then click **Open**.
6. Click **Next** twice to accept the default configurations, and then click **Finish**.
7. In the **Certificate Import Wizard** message box, click **OK**.
8. In left pane of the Console1 window, expand **Trusted Root Certification Authorities**, and then click **Certificates**.
9. In the right pane, double-click **WoodGroveCA** to examine the certificate properties.
10. Click **OK**.
11. In the left pane of the Console1 window, under **Other People**, click **Certificates**.
12. Double-click **Administrator** to examine the certificate properties.

Notice that this certificate is now valid.

1. Click **OK** to close the Certificate window.
2. Close the Console1 window, and then click **No** when asked to save changes.

##### Assign the WoodGrove Encryption Certificate to the Send Port

Procedure List

1. In the **CreditOrdersFILE - Send Port Properties** dialog box, on the **Certificate** tab, click **Browse**.

Notice that the certificate you just installed now appears.

1. Click **OK** to select the certificate, and then click **OK** to close the **Send Port Properties** window.

##### Test the Encryption Pipeline

Procedure List

1. In the BizTalk Server Administration Console, right-click **AdventureWorks**, and then click **Start**.
2. In the **Start ‘AdventureWorks’ Application** dialog box, click **Start**.
3. In Windows Explorer, navigate to **C:\AllFiles\LabFiles\Lab6**.
4. Copy **CredSalesOrder1.xml** and **CredSalesOrder2.xml** to the **SalesOrderIN** folder.
5. Open the **SalesOrderIN** folder to verify that the messages have been processed.
6. In Windows Explorer, navigate to the **CreditOrders** folder.
7. Right-click one of the XML files, and then click **Edit** to open the file in Notepad.

Notice that the message has been encrypted.

1. Close Notepad.
2. In Windows Explorer, navigate to the **AllOrders** folder.
3. Open one of the XML files in Notepad.

Notice that this file is not encrypted.

1. Close Notepad.
2. In Windows Explorer, delete all XML files from the **AllOrders** and **CreditOrders** folders.

Exercise 3: Examine the Interchange Message to Be Disassembled

##### Overview

An interchange, also known as a message batch, is a single message that contains multiple nested messages. A pipeline can be used to split an interchange into smaller messages. In this exercise, you will examine the sample interchange message that the pipeline you create will split.

##### Examine the Interchange Message

Procedure List

1. In Windows Explorer, navigate to and open **C:\AllFiles\LabFiles\Lab6\** **SalesOrder\_FF\_Interchange.txt** in Notepad.

Notice that this interchange contains a cash (Cash) transaction message and two credit (Cred) transaction messages. Also notice that the first line contains the batch information. This first line is known as the batch header.

1. Close Notepad.

Exercise 4: Configure a Receive Pipeline to Disassemble a Message Interchange

Overview

Disassembler components in custom flat file pipelines can be configured to split interchanges by defining separate header and body schemas. The header schema represents the introductory portion of a batch. The header typically contains information common to all messages in the batch. In this exercise, you will import a header schema and create a pipeline to process the batch of flat file sales order messages submitted by the retail stores. A second disassembler in the pipeline will allow processing of individual sales orders.

##### Add the Header Schema to the Messaging Project

Procedure List

1. In Solution Explorer, right-click **Messaging**, point to **Add**, and then click **Existing Item**.
2. Navigate to **C:\AllFiles\LabFiles\Lab6\AdvWorks\Artifacts**, click **BatchHeader.xsd**, and then click **Add**.
3. In Solution Explorer, double-click **BatchHeader.xsd**.
4. In the BizTalk Editor, expand **Batch**, expand **BatchDetail**, and then click **ManagerName**.

Notice the ManagerName and EmailAddress nodes in the schema. These fields represent the header information for the interchange.

1. Close the **BatchHeader.xsd** schema. If prompted to save your changes, click **Yes**.

##### Create a Receive Pipeline

Procedure List

1. In Solution Explorer, right-click **Messaging**, point to **Add**, and then click **New Item**.
2. In the left pane, click **Pipeline Files**, in the center pane, click **Receive Pipeline**, in the **Name** box, type **ReceiveSalesOrderInterchange.btp**, and then click **Add**.

##### Add a Flat File Disassembler Pipeline Component to the Pipeline

Procedure List

1. Drag the **Flat file disassembler** from the **Toolbox** to the **Disassemble** stage of the pipeline.
2. Drag a second **Flat file disassembler** to the pipeline, below the one you just added.
3. Click the first **Flat file disassembler**, and then in the Properties window, in the **Document schema** box, click **AdvWorks.Messaging.SalesOrder\_FF**.
4. In the **Header** **schema** box, click **AdvWorks.Messaging.BatchHeader**.
5. Click the second **Flat file disassembler**, and then in the Properties window, in the **Document schema** box, click **AdvWorks.Messaging.SalesOrder\_FF**.

Only the first Flat file disassembler component that matches the incoming message will be used on the message. This means that this pipeline will be able to process messages in the SalesOrder\_FF format, both as single messages and as part of an interchange.

##### Deploy the Messaging Project

Procedure List

1. In Solution Explorer, right-click the Messaging project, and then click Deploy.

You may receive warnings regarding the SalesOrder\_To\_LoanApp map; these warnings can safely be ignored. Also notice the warning that states: “If any of the assemblies were previously loaded by a Host Instance, it may be necessary to restart the Host Instance for changes to take effect.” If the Error List pane is not visible at the bottom of the screen, click Error List on the View menu.

Exercise 5: Configure a Receive Location to Use the Pipeline

Overview

The pipeline used to process inbound messages is configured on the receive location. In this exercise, you will create a new receive location that is associated with the existing SalesOrder port and configure it to use the ReceiveSalesOrderInterchange pipeline. This configuration will allow receiving and processing of XML orders, flat file orders, or flat file interchange messages.

Create a New Receive Location for Flat File Message Receipt

Procedure List

1. In the BizTalk Server Administration Console, expand **BizTalk Server Administration**, **BizTalk Group**, and **Platform Settings**, and then click **Host Instances**.
2. Right-click **BizTalkServerApplication**, and then click **Restart**.

The BizTalk host instance associated with an assembly needs to be restarted whenever the assembly is updated. Otherwise, the old copy of the assembly will remain in memory.

1. Under **Applications**, right-click **AdventureWorks**, and then click **Refresh**.
2. In the BizTalk Server Administration Console, click **AdventureWorks**.
3. Right-click **Receive Locations**, point to **New**, and then click **One-way Receive Location**.
4. In the **Select a Receive Port** dialog box, select **SalesOrder**, and then click **OK**.
5. In the **Receive Location Properties** dialog box, in the **Name** box, type **FFSalesOrderFILE**.
6. In the **Type** list, click **FILE**, and then click **Configure**.
7. In the **FILE Transport Properties** dialog box, click **Browse**.
8. Expand **C:\AllFiles\LabFiles\Lab6**, click **SalesOrderIN**, and then click **OK**.
9. In the **FILE Transport Properties** dialog box, in the **File mask** box, type **\*.txt**,and then click **OK**.
10. In the **Receive Location Properties** dialog box, in the **Receive pipeline** list, click **ReceiveSalesOrderInterchange**, and then click **OK**.
11. In BizTalk Server Administration Console, in the left pane click **Receive Locations**, in the right pane, right-click **FFSalesOrderFILE**, and then click **Enable**.

Exercise 6: Submit Test Messages

##### Overview

A successfully processed interchange message will result in multiple individual messages. By default, any invalid interchange messages will be suspended. In this exercise, you will submit two test messages: an interchange message containing valid data and an interchange message containing invalid data.

##### Submit a Valid Interchange Message

Procedure List

1. In Windows Explorer, navigate to **C:\AllFiles\LabFiles\Lab6**.
2. Copy **SalesOrder\_FF\_Interchange.txt** to the **SalesOrderIN** folder.
3. Open the **SalesOrderIN** folder.

Notice that the message has been processed and moved from this folder.

1. In Windows Explorer, navigate to the **AllOrders** folder.

Notice that all three documents from the batch appear in this folder.

1. Double-click any one of the documents.

Notice that the message has been changed to the SalesOrder format.

1. Close Microsoft Internet Explorer®.
2. In Windows Explorer, navigate to the **CreditOrders** folder.
3. Right-click the document of your choice, and then click **Edit**.

Notice that the file is encrypted and the message content cannot be read.

1. Close the document.
2. In Windows Explorer, navigate to the **CashOrders** folder.

Notice that the CASH order was processed successfully.

1. In Windows Explorer, delete all XML files from the **AllOrders**, **CashOrders**, and **CreditOrders** folders.

##### Submit an Invalid Interchange Message

Procedure List

1. In Windows Explorer, navigate to the **Lab6** folder, and then open **SalesOrder\_FF\_InterchangeBADDATA.txt**.

Notice that the batch contains a normal Cash message, a normal Cred message, and a Cred message that contains {BADDATA}.

1. Close the document.
2. Copy **SalesOrder\_FF\_InterchangeBADDATA.txt** to the **SalesOrderIN** folder.
3. Open the **SalesOrderIN** folder.

Notice that the message has been processed and moved from this folder.

1. In Windows Explorer, navigate to the **CreditOrders** folder.

Notice that the messages do not appear in this folder.

1. In Windows Explorer, navigate to the **AllOrders** folder.

Notice that the messages do not appear in this folder.

1. In the BizTalk Server Administration Console, right-click **BizTalk Group**, and then click **Refresh**.
2. In the center pane, in the **Group Hub** tab, Under **Suspended Items**, click **Resumable**.

Notice that the Query results pane shows one item that is suspended.

1. In the Preview pane, double-click the suspended **SalesOrder** message.
2. In the **Service Details** dialog box, click the **Error Information** tab.

Notice the error description.

1. Click the **Messages** tab, and then double-click the suspended message.
2. In the **Message Details** dialog box, click **body**.

This dialog box can be used to examine the message body to locate the invalid information.

1. Close the **Message Details** dialog box, and then click **OK** to close the **Service Details** dialog box.
2. In the Query results pane, right-click the suspended message, and then click **Terminate Instance**.
3. In the **Confirm Terminate Operation** dialog box, read the message, and then click **Yes**.
4. In the BizTalk Server Administration message box, click **OK**.

Exercise 7: Enable and Test Recoverable Interchange

##### Overview

Enabling recoverable interchange processing allows the valid messages of an interchange to be successfully processed, while any bad messages are individually suspended. In this exercise, you will enable a recoverable interchange for a pipeline, and then test the pipeline by submitting two messages.

##### Enable Recoverable Interchange Processing

Procedure List

1. In the BizTalk Server Administration Console, under **AdventureWorks**, click **Receive Locations**, and then double-click **FFSalesOrderFILE**.
2. In the **Receive Location Properties** dialog box, next to the **Receive pipeline** list, click the **ellipsis (…)** button.
3. In the **Configure Pipeline** dialog box, under **Disassemble – Component(1)**,in the **RecoverableInterchangeProcessing** list, click **True**, and then click **OK**.
4. In the **Receive Location Properties** dialog box, click **OK**.

##### Submit an Interchange Message that contains bad data

Procedure List

1. In Windows Explorer, navigate to **C:\AllFiles\LabFiles\Lab6**.
2. Copy **SalesOrder\_FF\_InterchangeBADDATA.txt** to the **SalesOrderIN** folder.
3. Open the **SalesOrderIN** folder.

Notice that the messages have been processed and moved from this folder.

1. Navigate to the **AllOrders** folder.

Notice that two messages have been added to this folder.

1. Navigate to the **CashOrders** folder.

Notice that the Cash message from the batch file has been added to this folder.

1. Navigate to the **CreditOrders** folder.

Notice that only one Cred message has been added to this folder. The Cred message that contains bad data has been suspended.

1. Close all open windows.