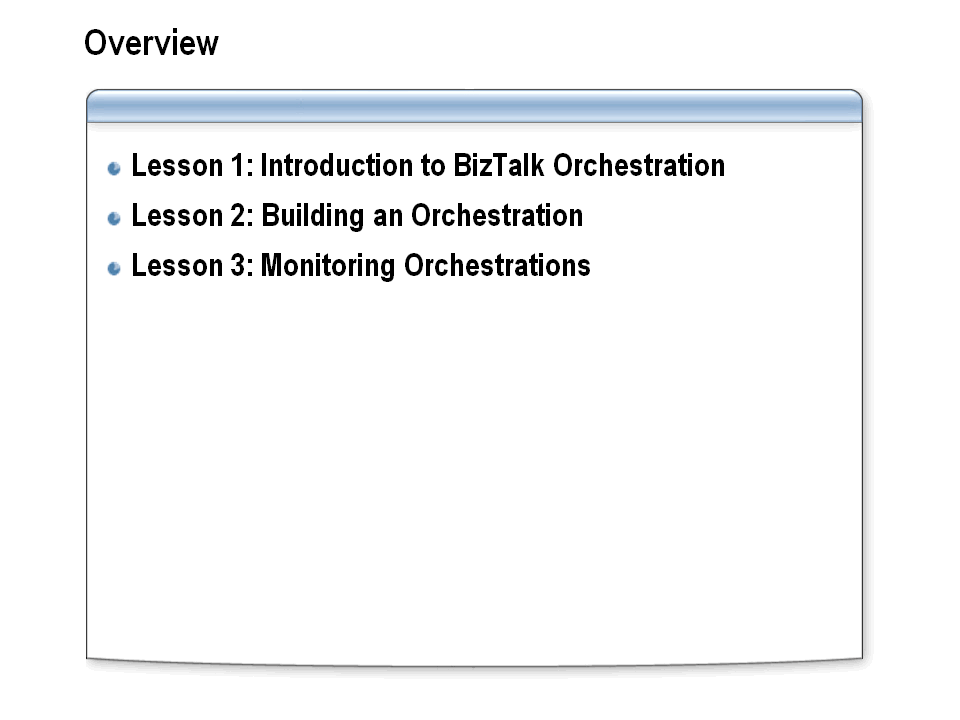
Module 8: Creating a BizTalk Orchestration

##### Time estimated: 105 minutes



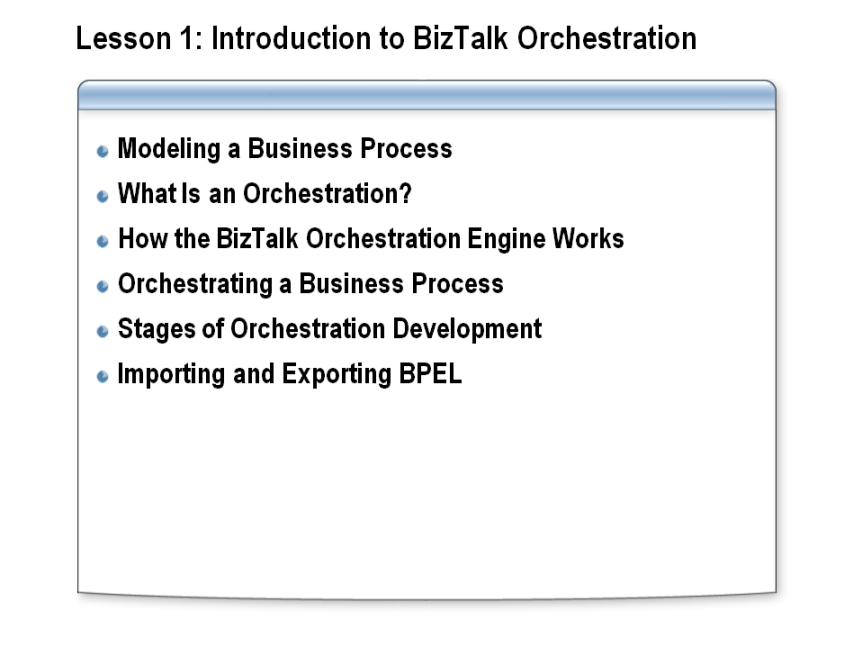
Module objective:

In this module, you will learn how BizTalk orchestration services work, how to create a BizTalk orchestration, and how to use debugging tools to monitor a running orchestration.

##### Overview

By using Microsoft BizTalk Server, you can orchestrate dynamic business processes both within and between organizations. By using BizTalk Server, developers, information workers, and IT professionals can work together to rapidly design, create, and deploy integration solutions that work across applications, platforms, and organizations.

Lesson 1: Introduction to BizTalk Orchestration



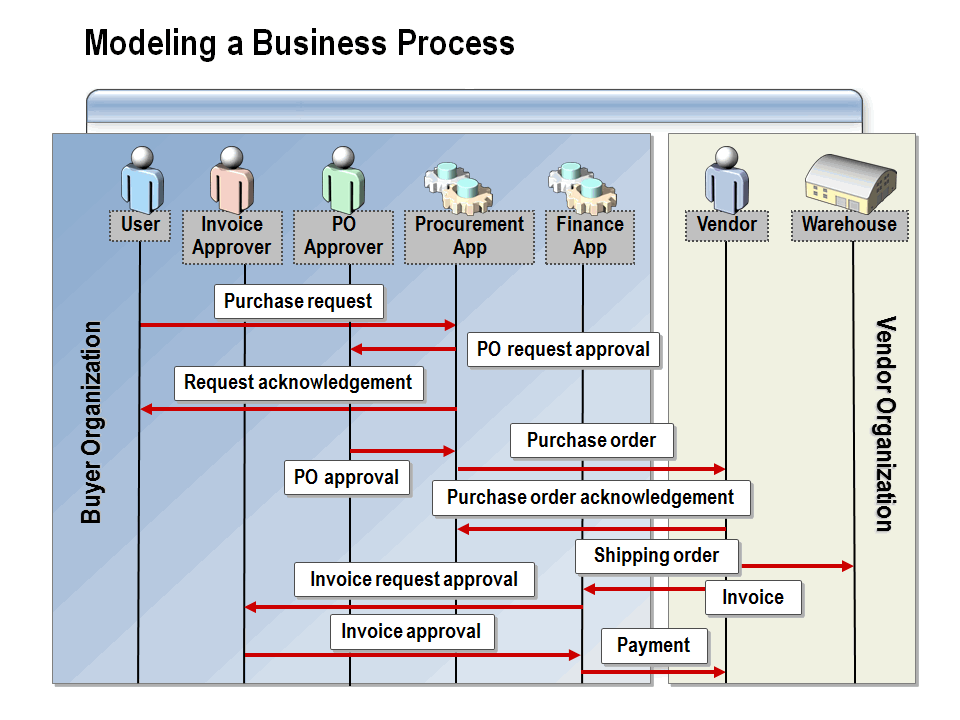
Lesson objective:

Explain how BizTalk processes orchestrations, and identify the steps required to create an orchestration.

##### Overview

A business process is a set of actions that, taken together, meet a business need. You can use the Orchestration Designer to define these actions graphically. Rather than requiring you to express the various steps in a programming language, this tool makes it possible to create an orchestration by connecting a series of graphical shapes in a logical way.

Modeling a Business Process



Define the term *business process* and describe the concept and benefits of business-process modeling.

##### What Is a Business Process?

Something to consider when building business processes is that a business process is seldom a single, monolithic process. Rather, a business process is made up of multiple components that will grow and change over time with the business. It is a good idea, then, to break up a business process into stages. This will help later for versioning and for isolating disruptions, which can happen throughout the business process. One good way to break down a complex business process is to use general criteria to determine the stages of the business process, including:

* The logical groupings of steps in the business process.
* The scopes of operations within an orchestration.
* The elements of the process that might need to be versioned at the same time.

A well-designed business process solution enables you to add or remove components from the process without having to reconstruct the entire application.

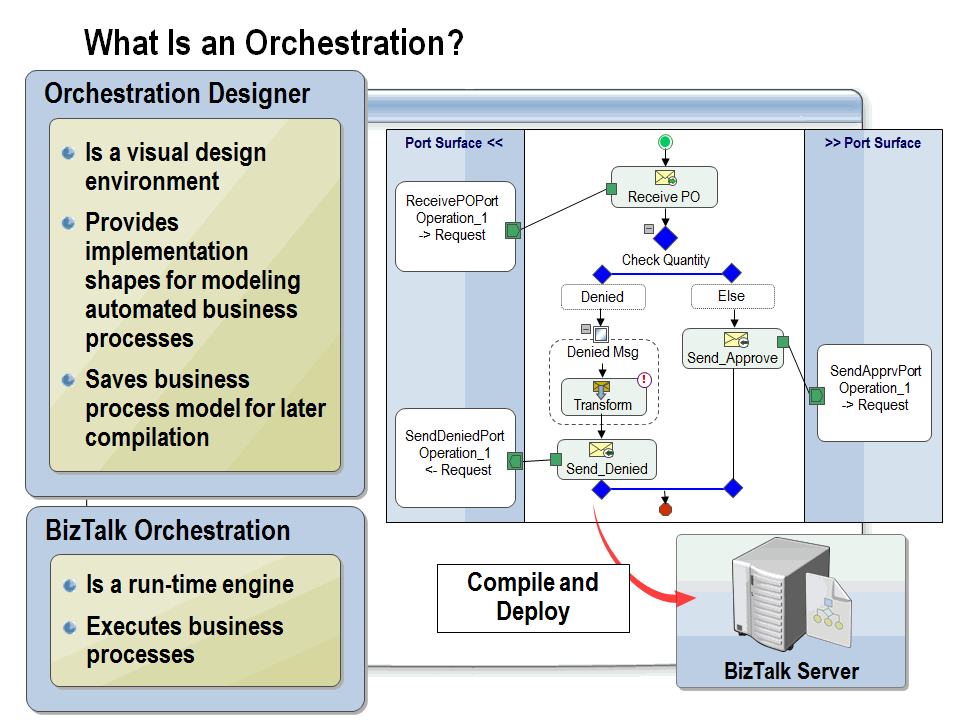
##### Business-Process Modeling

Business-process modeling is a technique that you can use to describe and optimize business processes in an organization. It provides a set of understandable, repeatable structures for defining business processes. By using a highly structured approach to process automation, you can increase throughput, reduce cycle times, improve efficiency, and maximize customer satisfaction.

Business-process modeling improves the interaction and coordination between people and systems throughout the organization by automating the flow of each business process, and it provides tools for managing day-to-day business processes. At any given time, there may be many instances of a given process. Because every instance follows the same flow and therefore has an identical structure, you can easily automate process management to derive answers to important questions, such as:

* What are the bottlenecks?
* What resources are required within the process?
* What commitments are not being met?
* How can processes be optimized?

What Is an Orchestration?



Define BizTalk orchestration and describe how the BizTalk run-time engine processes orchestrations.

##### Orchestration Designer

BizTalk Orchestration Designer is a tool that is hosted within Microsoft Visual Studio®. The Orchestration Designer is used to create visual representations of your business processes. The actual underlying code that carries out the business process gets built into an executable module that runs on a BizTalk Server computer.

The Orchestration Designer design environment provides a versatile drawing surface and a comprehensive set of implementation tools. These tools include shapes (such as Send, Receive, Transform, and Construct) that you can use to coordinate all of your business-to-business orchestrations on the screen using a graphical interface. The addition of each shape generates code that becomes part of the eventual business process.

Orchestrations designed using Visual Studio 2010 are saved with an .odx extension and are built as part of a Microsoft .NET assembly DLL. This DLL is then deployed into the run-time environment to be accessible to BizTalk.

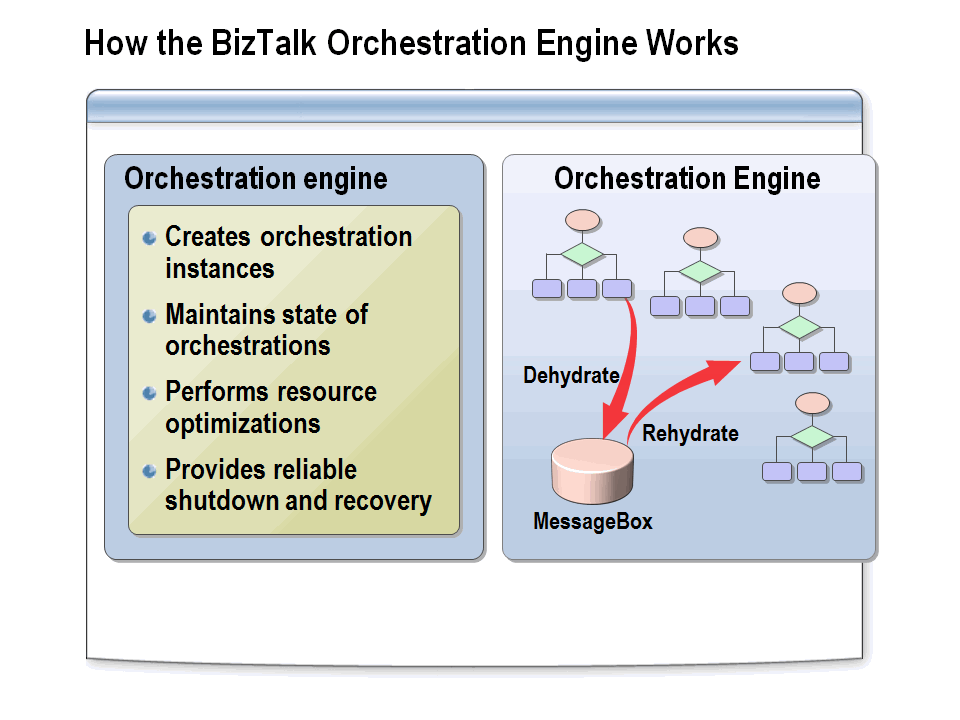
##### BizTalk Orchestration

BizTalk orchestrations provide the services that make it possible for you to design, execute, and manage business processes. An orchestration represents the underlying logic for processing messages similar to traditional procedural programming languages. An orchestration is a graphical rendering of the processing logic, providing many of the normal programming constructs as well as additional features needed to address requirements such as long-running transactions, parallel actions and decisions. An orchestration can be viewed as an event-driven, finite state machine in which the state-transition events are the arrivals of messages or ticks of the clock.

##### Run-Time Engine

The BizTalk orchestration run-time engine executes the orchestration by creating individual instances of the business process. The run-time engine coordinates multiple instances of a business process, ensuring, for example, that a response message gets routed to the correct orchestration instance, which it does by using a specialized routing pattern called correlation. For example, you may create a process involved with hiring new employees. For each employee, there are a number of messages that need to be sent and tasks that need to be performed. BizTalk Server can manage several instances of the orchestration for each new employee and correlate each incoming message to the correct orchestration instance.

How the BizTalk Orchestration Engine Works



Explain how the BizTalk orchestration engine works and describe the concept of dehydration and rehydration.

##### Introduction

The BizTalk orchestration engine is a highly optimized service that monitors and executes orchestrations. The orchestration management tasks the orchestration engine is responsible for include:

* Creating instances of and executing orchestrations.
* Maintaining the state of a running orchestration instance so that it can be restored to memory when required.
* Performing optimizations of running orchestrations to maximize scalability, throughput, and efficient use of resources.
* Providing a reliable shutdown-and-recovery system.

##### Dehydration and Rehydration

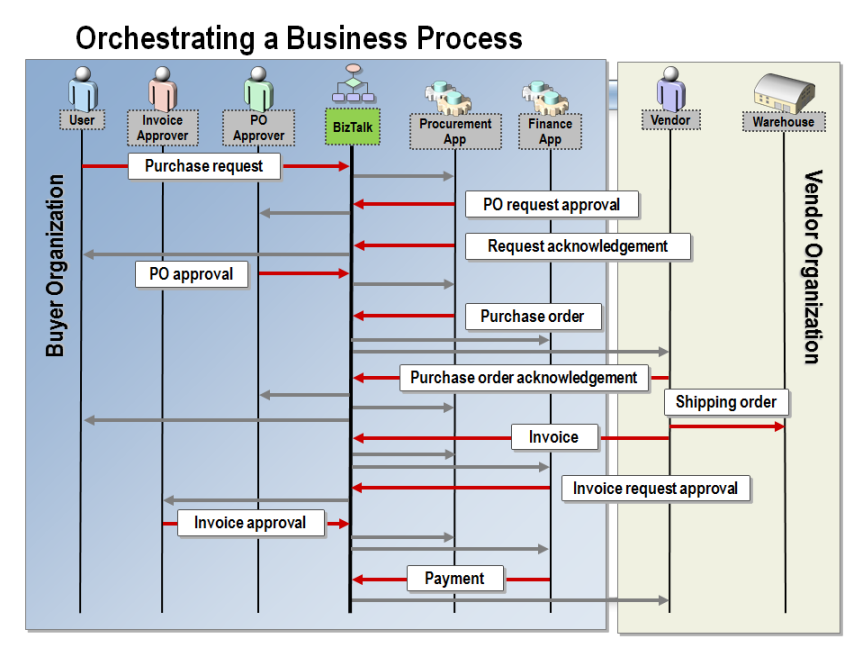
When many business processes are running at the same time, memory and performance can be compromised. The orchestration engine solves these issues by performing the following actions on orchestration instances:

**Dehydration**. Dehydration is the process of saving the current state of an orchestration instance to disk and then removing that instance from memory. When the run-time engine determines that an orchestration instance has been idle for a period of time, it may dehydrate the instance, thereby freeing up the resources that the instance had been using. By dehydrating orchestration instances, the engine makes it possible for a large number of business processes to run concurrently on the same computer. Dehydration is the responsibility of the orchestration engine. However, the developer can control the dehydration threshold by changing BizTalk Server configuration properties. Dehydration may take place at any time except during the execution of an atomic transaction.

Note: Atomic transactions will be covered in more detail in Module 10, “Creating Transactional Business Processes.”

**Rehydration**. Rehydration is the process of restoring the saved state of an orchestration instance from disk back to memory. When a message is received, or when a time out expires, the orchestration engine will automatically rehydrate the orchestration instance that the message belongs to or for which the expiration has occurred. The engine loads the saved orchestration instance into memory, restores its state, and runs it from the point where it left off.

Orchestrating a Business Process



Explain how a BizTalk orchestration could facilitate a business process.

##### Introduction

BizTalk orchestration is a flexible and powerful capability that provides various services and tools to enable you to design, automate, and manage business processes. Similar to traditional procedural programming languages, an orchestration represents the underlying logic for processing messages.

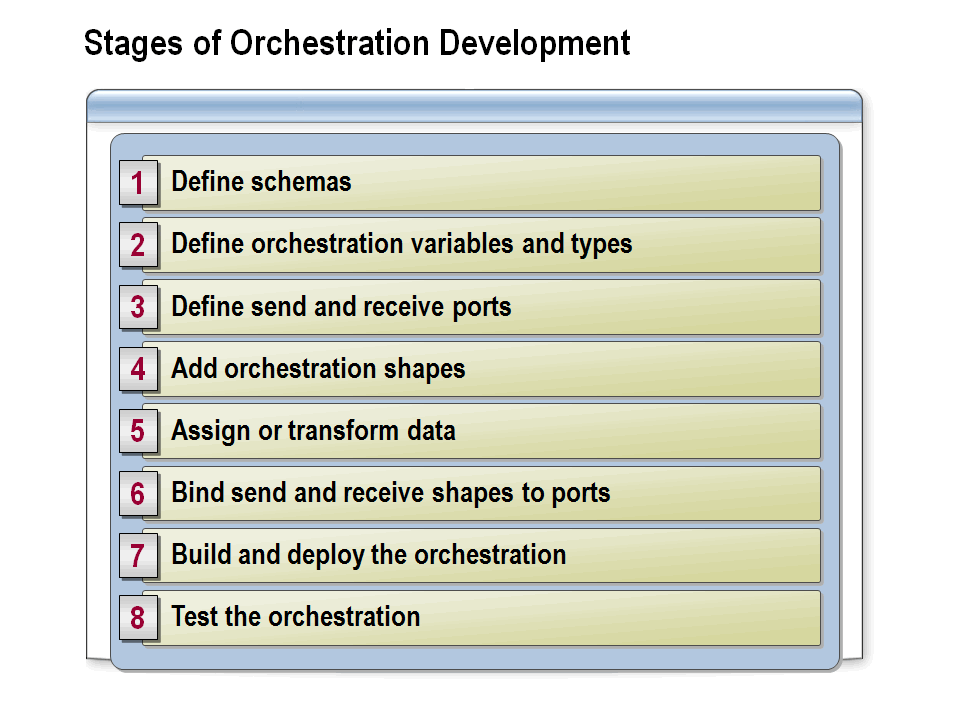
BizTalk orchestration provides a transactional programming model that includes support for exception handling and recovery from failed transactions. You can define two types of transactions when creating an orchestration:

*Atomic transaction*. Enables a transaction to automatically role back to a previous state in case the transaction does not successfully complete.

*Long running transaction*. Can span days, weeks, and longer time durations, contain nested transactions, and use custom exception handling to recover from error scenarios.

As a result, orchestrations provide you the flexibility to define the course of action in the event of a business process failure.

Stages of Orchestration Development



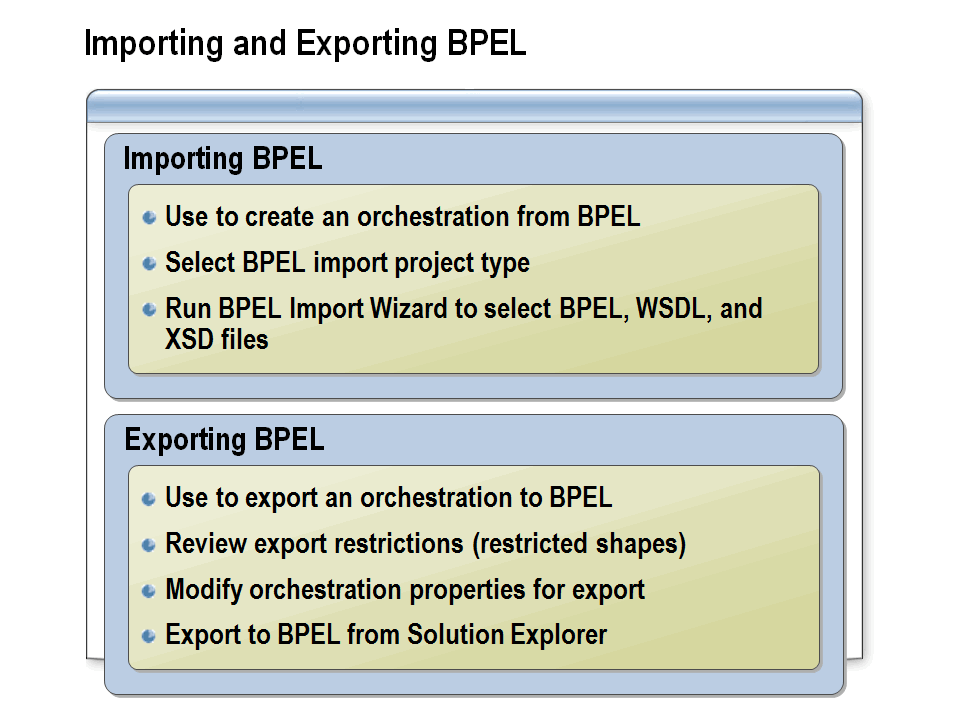
Identify and describe each of the steps involved in developing a BizTalk orchestration.

##### Orchestration Steps

To develop an orchestration, you will typically perform the following basic steps:

1. Define schemas to describe the format of those messages that the orchestration will process. Because BizTalk Server is message based, all actions that are to be performed will relate to messages.
2. Define and assign orchestration variables to declare and manage the data that is used in your running orchestration.
3. Define ports to specify how and where messages are sent and received. Ports are the mechanism whereby BizTalk communicates with external systems and processes.
4. Add shapes to represent the various actions that are required to define the business process. Many shapes require configuration to identify the messages that they will interact with.
5. Construct new message instances within the orchestration. BizTalk Server provides transformation and message assignment shapes to create new messages within an orchestration. Because messages are immutable within BizTalk orchestrations, if a message must be modified, it is necessary to construct a new instance.
6. Bind the Send and Receive shapes to ports and specify the physical ports that they will use.
7. Build and deploy the orchestration.
8. Test the orchestration for any errors.

Importing and Exporting BPEL



Define BPEL and explain how BPEL files can be imported into an orchestration or exported from an orchestration.

##### What Is BPEL?

Microsoft and IBM, working together with other companies, have created the Business Process Execution Language (BPEL) as an industry standard to support the interaction of business processes between heterogeneous systems. A business process defined by using the BizTalk Orchestration Designer can be exported to BPEL, and similarly, the BizTalk Orchestration Designer can import processes defined in BPEL. This interactivity makes it easy to import designs created using other BPEL-compliant tools into BizTalk. BPEL is built entirely on Web services, whereas BizTalk Server 2010 supports a broader set of services. For example, BizTalk Server 2010 supports mapping between differing XML schemas, calling methods in local objects, broad support for transactions, and other features that are not available in BPEL.

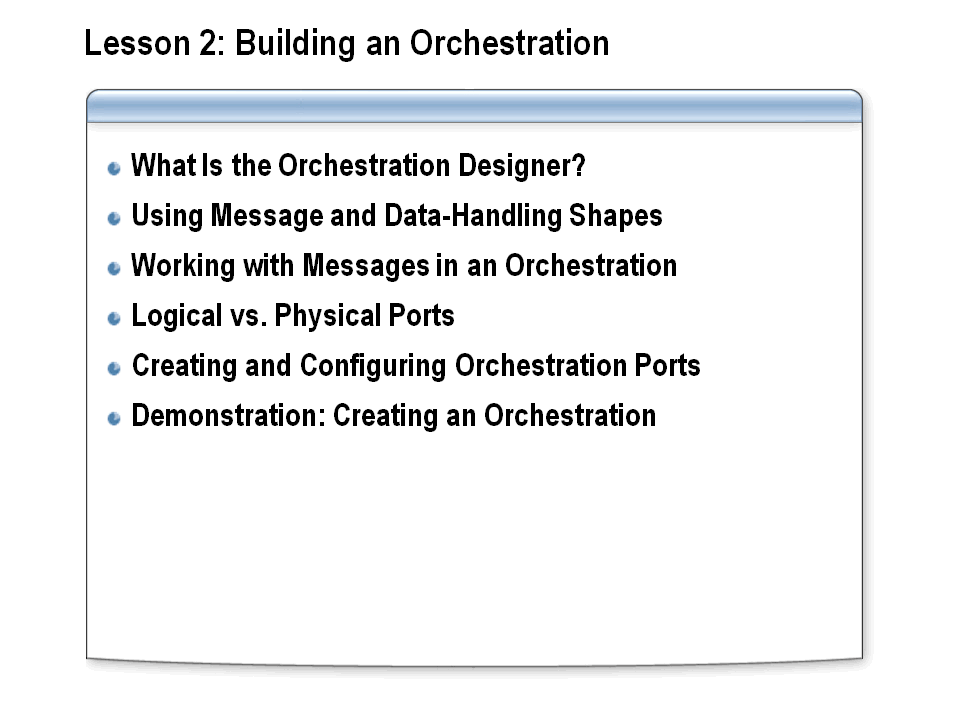
##### Importing BPEL

You can use the BizTalk Server BPEL Import Wizard to import BPEL, Web Service Description Language (WSDL), or XML Schema Definition (XSD) language files into a BizTalk project. To import BPEL into an orchestration, create a new BizTalk project choosing the BizTalk Server BPEL Import Project template, which starts the BPEL Import Wizard.

##### Exporting BPEL

If you are exporting to BPEL, the orchestrations can contain only features that are common between BizTalk Server and BPEL or features that can be translated into BPEL without affecting behavior. To export BPEL from an orchestration, first modify the orchestration properties to make the orchestration exportable, right-click the orchestration in Solution Explorer, and then click Export To BPEL.

Lesson 2: Building an Orchestration



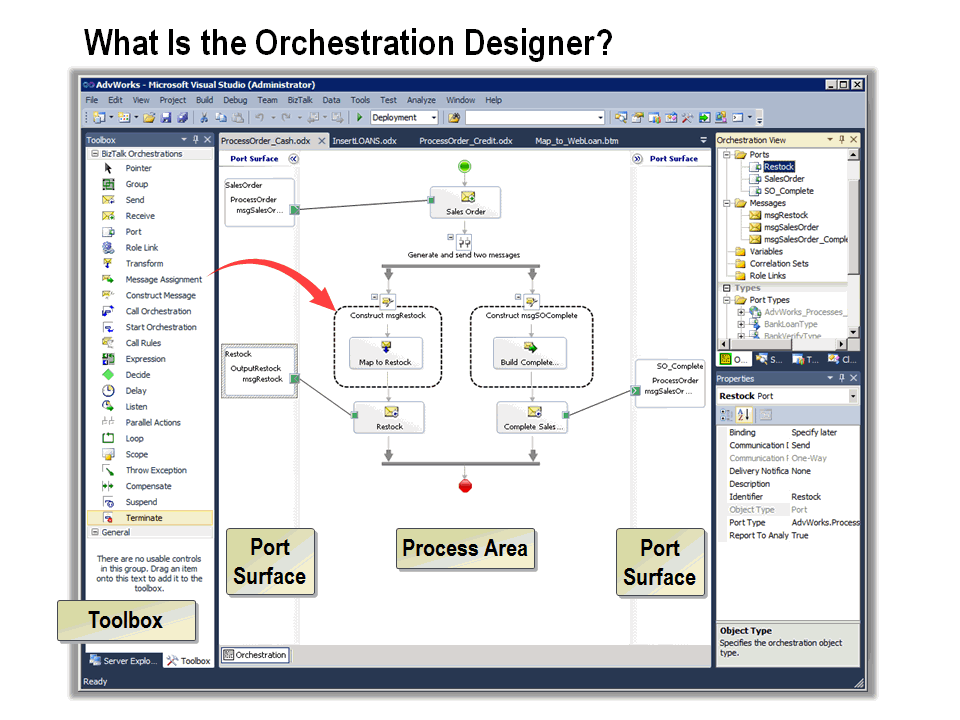
Lesson objective:

Create message variables and types and use the Orchestration Designer to create a basic orchestration.

##### Introduction

The BizTalk Orchestration Designer provides a graphical method of designing, deploying, and maintaining distributed business processes. In this lesson, you will learn how you can use BizTalk Orchestration Designer to design business processes for managing the logic that is needed for processing business documents.

What Is the Orchestration Designer?



Explain the functions and capabilities of the Orchestration Designer.

##### Introduction

BizTalk Orchestration Designer provides a visual drawing surface that you use to represent a business process. The design surface is a working canvas to which you drag shapes from the Toolbox. In most cases, you must configure options for each shape that you use when you build the orchestration.

##### The Design Surface

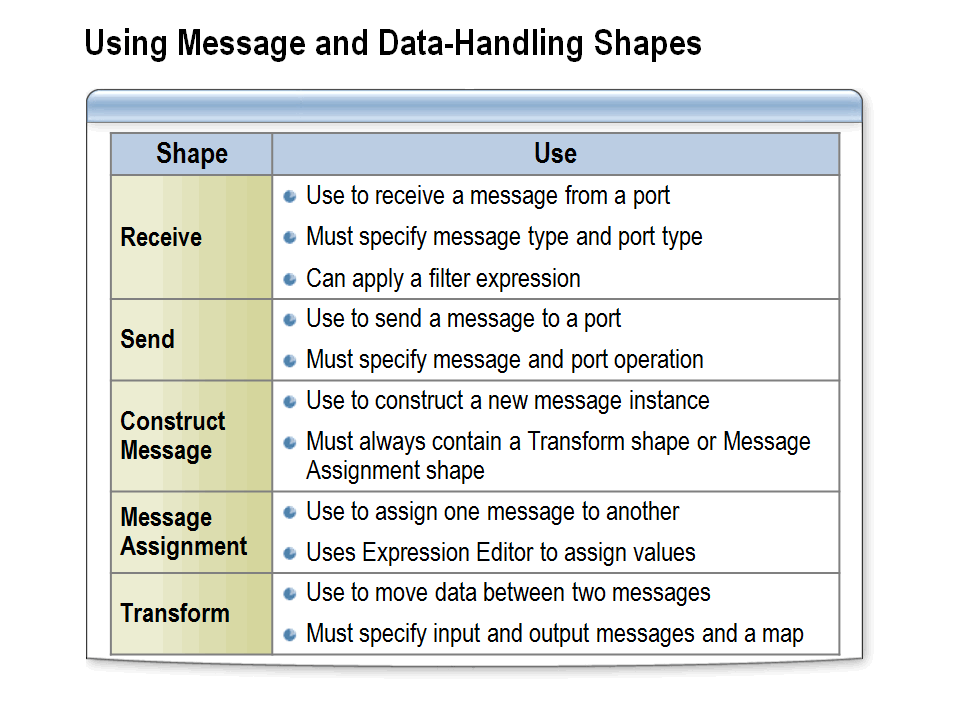
The design surface itself is divided into three areas: the Process Area and two Port Surfaces. The Process Area contains shapes that describe the actual process flow of the orchestration. It is flanked on both sides by Port Surfaces, which contain only Port and Role Link shapes that interact with the send and receive shapes in the Process Area.

It makes no difference which port Surface you use for a send or receive port (the left side or the right side). With two port surfaces to place your new ports on, you can create orchestrations that have fewer crisscrossing connectors, making your orchestrations easier to read.

##### Zoom Support

BizTalk Server 2010 provides the ability to zoom the Orchestration Designer window. You can zoom by right-clicking a clear area of the Orchestration Designer surface and then clicking Zoom. You can then select the percentage of magnification that you would like to apply. Alternatively, you can press the CTRL key while using the scrolling function of your pointing device to zoom in and out.

Using Message and Data-Handling Shapes



Create and configure message and data-handling shapes for an orchestration.

##### Message and Data-Handling Shapes

Orchestration Designer provides you with the following shapes to use for processing messages and data in your orchestrations:

**Receive**. This shape must be connected to a receive port on one of the port surfaces. You specify the type of message that will be received and the orchestration port over which it will be received. Optionally, you can specify a filter expression that tests properties on the message to determine whether an orchestration should process the message. You can also configure information that enables the message to be correlated with an existing orchestration instance. Correlation is covered in Module 9, “Automating Business Processes.”

**Send***.* This shape must be connected to a send port on one of the port surfaces. You specify the message to send and the port operation that will send it. If you expect to receive an indirect response—that is, a response that does not use a request-response port—you must provide information to correlate the message with its associated orchestration instance.

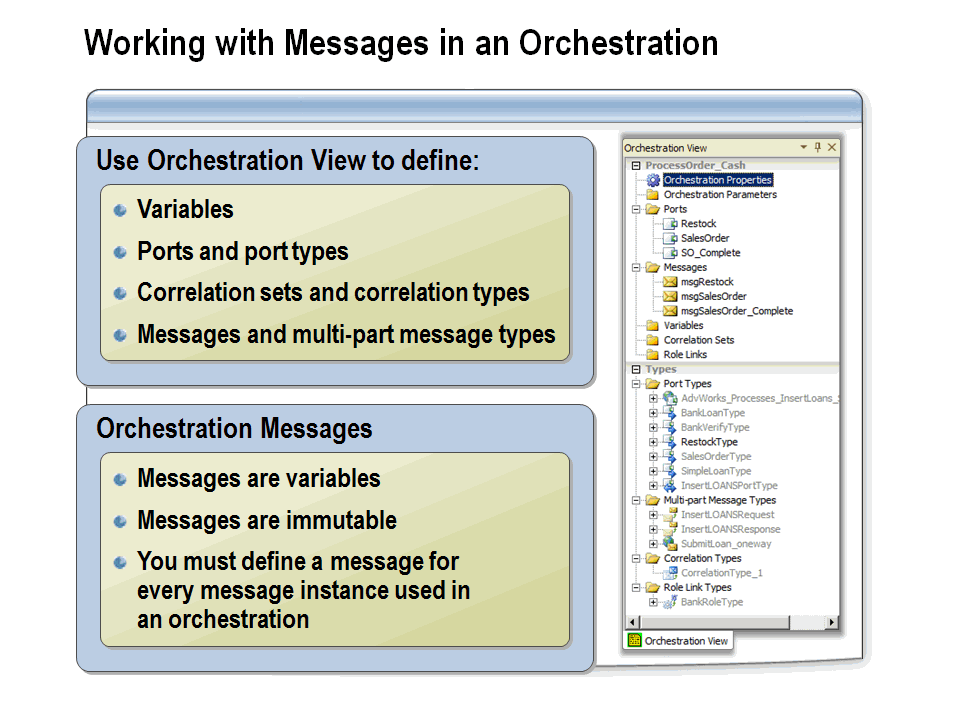
**Orchestration Port**. This shape is a logical representation of a port and must be bound to a physical port to enable the orchestration to communicate outside this orchestration to other orchestrations or through the MessageBox to the outside world.

**Construct Message***.* Constructs a new instance of a message in your orchestration. The construct message shape can contain only message assignment or transform shapes. You specify the messages that you want to construct and then make assignments either to the message or to its parts. A single construct message shape may contain a combination of message assignment and transform shapes. Once processing has moved out of the construct message shape, the constructed messages are immutable.

**Message Assignment**. Constructs messages by assigning one message to another, by assigning individual message nodes, or by calling a .NET class to construct the message. The message assignment shape must be contained within a construct message shape.

**Transform**. Applies a BizTalk map to create a new message. You specify one or more input messages, one or more output messages, and an Extensible Stylesheet Language Transformations (XSLT) map for the transform. The map assigns message parts from the input messages to message parts in the output messages.

Important: Many Orchestration Designer tasks, including those outlined here, require you to select various items such as schemas or orchestrations. If these items are not in the current project, you must add a reference in your project to the assembly that contains the item that you want to select.Working with Messages in an Orchestration



Define message variables, ports, correlation sets, and multi-part messages.

##### Message Variables

Before working with messages in an orchestration, a message variable must be defined using the Orchestration View in Visual Studio. A message variable represents an instance of an incoming message to be processed by the orchestration. This is true whether the message will be received through a port or constructed within the orchestration itself.

In many cases, multiple message variables may need to be defined. For example, if an orchestration receives a purchase order from which an acknowledgement message and shipping advice message need to be generated, three different message variables need to be defined.

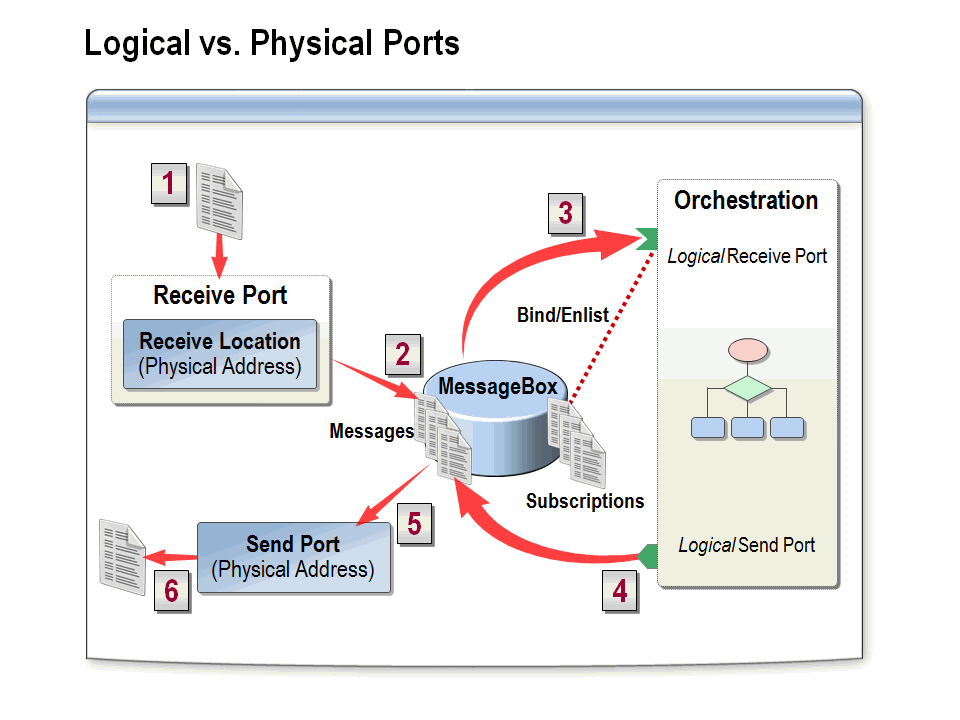
##### Messages Are Immutable

Messages are treated as immutable by the BizTalk messaging and orchestration engines once they arrive at the MessageBox. If it is necessary to change a message in any way within an orchestration, it is necessary to create a copy of that message by using the construct message shape.

##### Orchestration View

Visual Studio provides the Orchestration View window while you are actively working on an orchestration. To display the Orchestration View window if it is not already displayed, from the View menu, click Other Windows. This view allows developers to define variables, correlation sets, and ports for use in the orchestration. The bottom section displays the types defined anywhere within the current project or from any referenced project. Types defined in conjunction with the currently selected orchestration are modifiable, whereas those defined elsewhere are read-only and unavailable. The Orchestration View window allows specifying user-defined multi-part message types; however, the simple types upon which you may want to create messages are not shown in the Types section. This is because these messages will be based on deployed schemas that have no properties that are configurable within the orchestration context.

Logical vs. Physical Ports



Describe the differences between physical and logical orchestration ports.

##### Introduction

BizTalk Server uses two different types of ports: physical ports (used for messaging) and orchestration or logical ports.

##### Physical Ports vs. Logical Ports

Physical ports are used to receive and send messages between BizTalk Server and the outside world. Physical ports are configured and managed by using BizTalk Explorer and the BizTalk Administration console. They can also be created via scripts by using Microsoft Windows® Management Instrumentation (WMI). The messaging engine processes messages received through physical receive ports through pipelines and into the MessageBox database, whereas physical send ports are configured to deliver messages to external processes and applications.

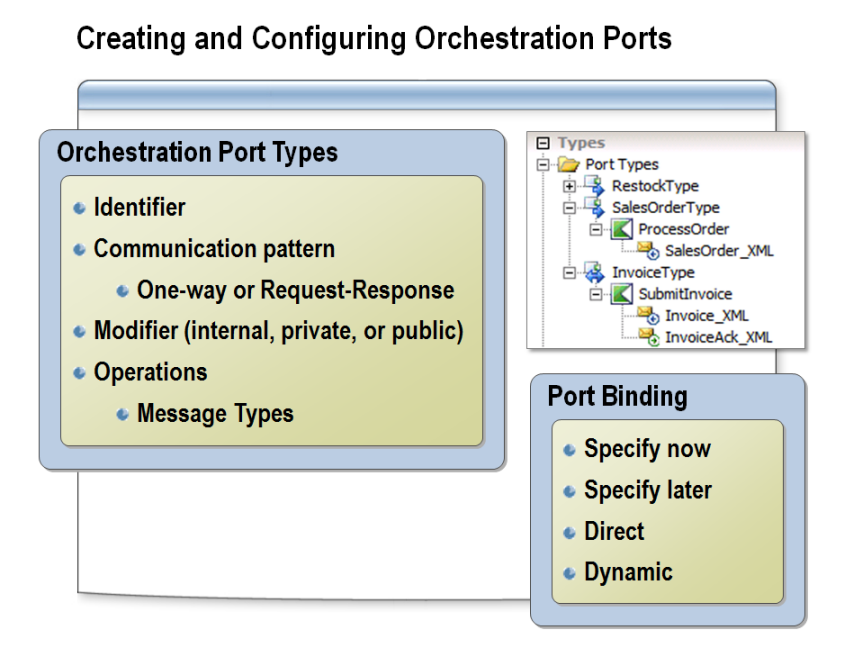
Logical ports are used to send and receive messages within the BizTalk Server environment. Logical ports are configured by using Orchestration Designer, and they appear on the Port Surface areas. Logical ports can be bound to logical ports in other orchestrations to enable direct communication or can be bound to physical ports to enable an orchestration to send and receive messages externally.

##### Example

The following example shows how messages are passed through BizTalk messaging and orchestration services.

1. A message is received by a physical receive port through an associated receive location.
2. The message is passed through the receive location (and any associated pipelines) and then saved in the MessageBox database.
3. The message is picked up by a subscribing orchestration and then passed to the logical receive port in the orchestration. This subscription is generated by binding the logical port to the physical port that received the message.
4. The message is processed by the orchestration, and a resulting message is passed back to the MessageBox database through the logical send port.
5. A subscribing physical send port picks up the message and then passes it through an associated pipeline.
6. The message is sent to the target destination.

Creating and Configuring Orchestration Ports



Create and configure orchestration ports.

##### Introduction

Each orchestration (logical) port added to a BizTalk orchestration must be based on a port type. Port types and ports are defined within the Orchestration View window in Visual Studio 2010. Ports and types have properties that must be configured before linking the ports to send and receive shapes.

##### Orchestration Port Types

Types have properties for the identifier (which must be unique to the project), a description, the communication pattern (one-way or two-way), and the modifier (public, private, or internal), which identifies the accessibility of ports based in this type.

If it is public, it is visible to anyone interacting with the orchestration. If it is private, it is visible to other orchestrations within the same project and namespace. If it is internal, the port type is visible only within the project. Because a port type definition includes message types, the scope of the message type must encompass that of any port type that uses it.

##### Orchestration Ports

Orchestration ports have Identifier and Description properties, as well as the type with which they are associated. Additionally, ports identify the direction of communication, which for one-way ports will be either sending or receiving. For two-way ports, the direction of the first communication is identified; for instance, a port can receive a request from an outside process and return a response (request-response), or it may solicit an external process to perform some action and wait for the receipt of a response (solicit-response).

##### Orchestration Port Bindings

Finally, ports define the binding, which identifies how the port will communicate with the outside world. Available binding settings are as follows:

**Specify now**. When a port’s binding is set for specify now (early-bound), the configuration of the connection to the physical port will be defined at design time. This will automatically create a dedicated physical port when the orchestration assembly is deployed. Ports that are early-bound can be modified after they have been deployed.

**Specify later**. When a port’s binding is set for specify later (late-bound), physical ports will need to be manually created and bound before the orchestration can be started. This option is good if the orchestration will bind to already existing ports and provides for better reusability between multiple send ports and orchestrations.

**Direct***.* Direct binding allows orchestrations to communicate directly with other orchestrations without the message passing through the MessageBox database as occurs with early- and late-bound ports.

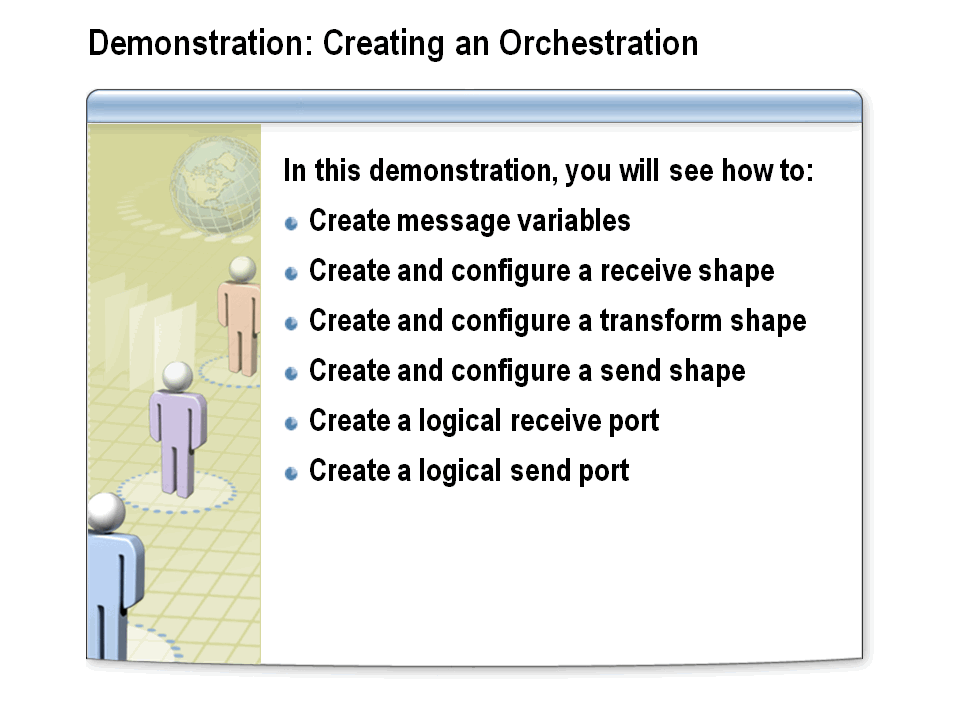
**Dynamic**. Dynamically bound ports are not configured at design time or at run time, but rather a decision is made within the orchestration that will determine the protocol and address to be used. Dynamic ports are frequently used for sending e-mail messages at run time or when communicating with trading partners for which different port configurations are required. This allows reading an e-mail address from a message within the orchestration and then specifying this as the destination address.

Not all port configurations are supported by all adapters—for example, two-way ports cannot be used in conjunction with FILE receive adapters—but rather are used primarily for communicating with Web services, SQL servers, and other request/response systems.

##### Creating Orchestration Ports and Port Types

Although it is possible to create ports in the Orchestration View window, it is common to create them by using the Port Configuration Wizard. Launch the wizard by right-clicking either (right or left) port surface and then clicking New Configured Port.

Demonstration: Creating an Orchestration



Learn how to create message variables and configure a receive shape, a transform shape, and a send shape. You will then see how to create logical send and receive shapes.

*Before starting this demonstration, delete the existing Demos application in the BizTalk Server Administration Console. The Demos application might need to be stopped before it can be deleted.*

##### Create a New Orchestration

1. In Windows Explorer, navigate to **C:\AllFiles\DemoCode\Module8\Demo**, and then double-click **Demo.sln**.
2. In Solution Explorer, right-click **Messaging**, and then click **Build**.
3. In Solution Explorer, right-click **Processes**, point to **Add**, and then click **New Item**.
4. In the **Add New Item** dialog box, click **BizTalk Orchestration**, in the **Name** box, type **ProcessOrder\_Cash.odx**, and then click **Add**.

##### Create a Message Variable

1. In Orchestration View, right-click **Messages**, and then click **New Message**.
2. In the Properties window, in the **Identifier** box, type **msgSalesOrder**.
3. In the **Message Type** list, expand **Schemas**, and then click **<Select from referenced assembly>**.
4. In the **Select Artifact Type** dialog box, click **Demo.Messaging**, click **SalesOrder**, and then click **OK**.
5. In Orchestration View, right-click **Messages**, and then click **New Message**.
6. In the Properties window, in the **Identifier** box, type **msgRestock**.
7. In the **Message Type** list, under **Schemas**, click **<Select from referenced assembly>**.
8. In the **Select Artifact Type** dialog box, click **Demo.Messaging**, click **Restock**, and then click **OK**.

##### Add a Receive Shape

1. Drag a **Receive** shape from the **Toolbox** to the area below the green circle (**Start**) in the orchestration.
2. In the Properties window, in the **Name** box, type **Sales Order**, and then set the **Activate** property to **True**.
3. In the Properties window, in the **Message** list, click **msgSalesOrder**.

##### Add a Transform Shape

1. Drag a **Transform** shape from the **Toolbox** to below the **Sales Order** receive shape in the orchestration.

Notice that when you dropped the Transform shape in the orchestration, the Construct Message shape was also added.

1. Click the **ConstructMessage\_1** shape, and then in the Properties window, in the **Name** box, type **Construct msgRestock**.
2. In the **Messages Constructed** list, click **msgRestock**.
3. Click the **Transform\_1** shape, and then in the Properties window, in the **Name** box, type **Map to Restock**.
4. In the Properties window, click **Map Name**, and then click the **ellipsis** (**…**) button.
5. In the **Transform Configuration** dialog box, click **Existing Map**, and then in the **Fully Qualified Map Name** list, click **<Select from referenced assembly>**.
6. In the **Select Artifact Type** dialog box, click **Demo.Messaging**, click **SalesOrder\_to\_Restock**, and then click **OK**.
7. In the **Transform Configuration** dialog box, click **Source**, and then in the **Variable Name** list, click **msgSalesOrder**.
8. Click **Destination**, in the **Variable Name** list, click **msgRestock**, and then click **OK**.

##### Add a Send Shape

1. Right-click the arrow immediately below the **Construct msgRestock** shape, point to **Insert Shape**, and then click **Send**.
2. In the Properties window, in the **Name** box, type **Restock**, and then in the **Message** list, click **msgRestock**.

##### Add a Logical Receive Port

1. Right-click the left **Port Surface**, and then click **New Configured Port**.
2. On the **Welcome to the Port Configuration Wizard** page of the **Port Configuration Wizard**, click **Next**.
3. On the **Port Properties** page, in the **Name** box, type **SalesOrder**, and then click **Next**.
4. On the **Select a Port Type** page, in the **Port Type Name** box, type **SalesOrderType**, and then click **Next**.
5. In the **Port direction of communication** list, click **I’ll always be receiving messages on this port**, and then in the **Port binding** list, click **Specify now**.
6. In the **URI** box, type **C:\AllFiles\DemoCode\Module8\SalesOrderIN\\*.xml**, in the **Transport** list, click **FILE**, and then click **Next**.
7. On the **Completing the Port Configuration Wizard** page, click **Finish**.
8. In the **SalesOrder** receive port shape, click **Operation\_1**, and then, in the Properties window, in the **Identifier** box, type **ProcessOrder**.
9. In the **SalesOrder** receive port shape, click **Request**, and then in the Properties window, in the **Name** box, type **SalesOrder\_XML**.
10. Drag the green arrow from **SalesOrder** receive port to the **Sales Order** receive shape.

##### Add a Logical Send Port

1. Right-click the left **Port Surface**, and then click **New Configured Port**.
2. On the **Welcome to the Port Configuration Wizard** page of the **Port Configuration Wizard**, click **Next**.
3. On the **Port Properties** page, in the **Name** box, type **Restock**, and then click **Next**.
4. On the **Select a Port Type** page, in the **Port Type Name** box, type **RestockType**, and then click **Next**.
5. In the **Port direction of communication** list, click **I’ll always be sending messages on this port**, and then on the **Port binding** list, click **Specify now**.
6. In the **URI** box, type **C:\AllFiles\DemoCode\Module8\OUT\Restock%MessageID%.xml**, in the **Transport** list, click **FILE**, and then click **Next**.
7. On the **Completing the Port Configuration Wizard** page, click **Finish**.

The port shapes can be repositioned by dragging the shape up or down the Port Surface.

1. In the **Restock** send port, click **Operation\_1**, and then in the Properties window, in the **Identifier** box, type **OutputRestock**.
2. In the **Restock** send port, click **Request**, and then in the Properties window, in the **Name** box, type **Restock\_XML**.
3. Drag the green arrow from **Restock** send shape to the **Restock** send port.

##### Deploy and Configure the Orchestration

1. In Solution Explorer, right-click the **Messaging** project, and then click **Deploy**.
2. In Solution Explorer, right-click the **Processes** project, and then click **Deploy**.
3. In the BizTalk Server Administration Console, right-click **Applications,** and then click **Refresh**.
4. Right-click **AdventureWorks**, and then click **Configure**.
5. In the **Host** list, click **BizTalkServerApplication**, and the click **OK**.
6. Right-click **AdventureWorks**, and then click **Start** twice.
7. Pause the **bt10d-demos** virtual machine.

Lesson 3: Monitoring Orchestrations



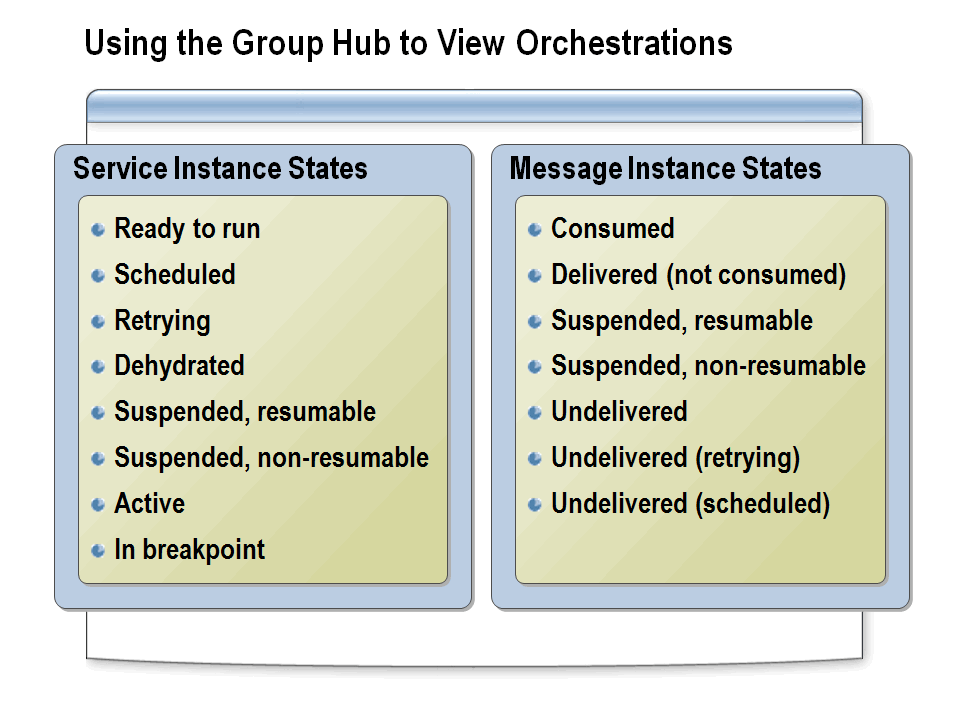
Lesson objective:

Use BizTalk tools to monitor and debug an orchestration.

##### Introduction

BizTalk Server 2010 provides tools for monitoring business processes at various stages. The BizTalk Group Hub provides features for reporting, analyzing, and debugging both live and archived data and messages.

Using the Group Hub to View Orchestrations



Use the Group Hub to monitor and view orchestration processing.

##### Group Hub

The Group Hub page (the Hub) in the BizTalk Administration console is used to view current service instances. The Hub displays links arranged in categories such as Work in Progress, or Suspended, or grouped by other characteristics such as Application, Service Name, or Error Code. Clicking any link initiates a query that returns all instances presently in the selected state or condition.

The BizTalk Group Hub also provides features for searching, reporting, analyzing, and debugging data and messages that are archived in the BizTalk Tracking databases. You can use the Group Hub to locate an orchestration based on the message that initiated the orchestration.

##### Service Instance States

These instances can be in a number of different states depending upon various conditions, including the load on the server, availability of resources, and the design of the orchestration that is being processed. The service instance states include:

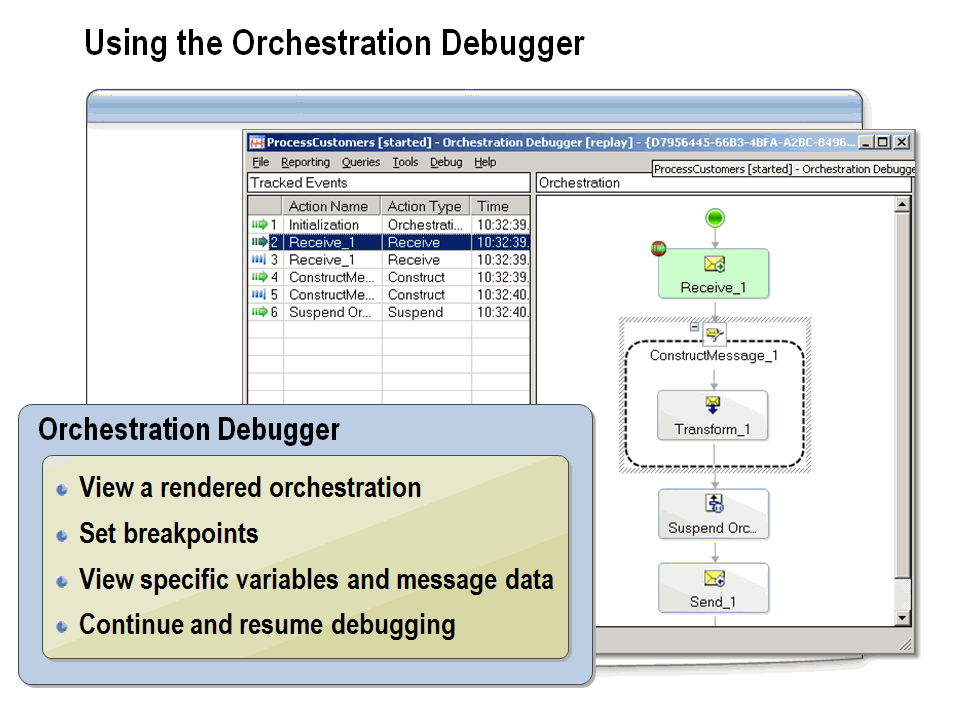
* *Ready to run*. A service instance has received an activation message but hasn’t started yet. When the number of ready-to-run instances continually increases, the resources to process the workload may be insufficient or unavailable.
* *Scheduled*. Scheduled is a ready-to-run sub-state in which a service is ready to be processed but will commence processing only within a specified window of time (service window). The service window can be specified by the user in the **Port Properties** dialog box for the port. Outside that service window, the service is shown as *scheduled*.
* *Retrying*. When a send port instance encounters a failed message transmission, typically because a resource is unavailable, it periodically tries to resend the message.
* *Dehydrated*. The orchestration instance is idle and not in memory. Dehydrated is essentially the same state as retrying, but it relates to an orchestration instead of to a message port. A dehydrated orchestration typically is reactivated when it receives a message.
* *Suspended*, *resumable*. The service instance is suspended. You may be able to resume the service instance by means of an API call or an administrative action.
* *Suspended*, *nonresumable*. The service instance is suspended and cannot be resumed.
* *Active*. The service instance is currently in memory.
* *In breakpoint*. The service instance has stopped execution at a preset breakpoint. You can resume the execution of the service instance through the Orchestration Debugger or by right-clicking the service instance and then clicking Resume.

##### Message Instance States

As has been noted, all processing within BizTalk begins with the receipt of a message. A single service instance can have a number of messages with which it is involved, each of which will have some state associated with that message. Valid states for messages are:

* *Consumed*. The message has been processed by a service instance. The service that processed the instance retains a reference so that it can access the message later. The message is considered delivered. For example, BizTalk Message Queuing Adapter MSMQ messages are in the consumed state during batched resending of messages. MSMQ can be blocked while it waits for an acknowledgement, and the messages will be flagged as consumed until the acknowledgement arrives, at which time MSMQ will restart sending messages.
* *Delivered (not consumed).*The message has been delivered to the engine, is being processed, and is in memory. It is considered delivered.
* *Suspended, resumable*. The service instance associated with the message is suspended and can be resumed.
* *Suspended, non-resumable*. The service instance associated with the message is suspended and cannot be resumed.
* *Undelivered.* There may be no services available to process the message, or there may be no services running. For example, in an ordered delivery scenario, a message is undelivered when another message that precedes it is being retried by the ordered delivery send port.
* *Undelivered (retrying).*The message is associated with a send port that is attempting to resend it because the destination resource is unavailable. (See the previous definition for the “retrying” service instance.)
* *Undelivered (scheduled).*The message is waiting to be sent by a send port that has a service window set.

Using the Orchestration Debugger



Use the Orchestration Debugger to track orchestration activity and diagnose problems with orchestration processing.

##### The Orchestration Debugger

The Orchestration Debugger enables you to track the activity of a single orchestration instance on a shape-by-shape basis. The Orchestration Debugger displays a rendered view of the orchestration that was previously created in Orchestration Designer.

##### Accessing the Orchestration Debugger

You access the Orchestration Debugger by right-clicking any service or message instance associated with an orchestration type and then clicking Orchestration Debugger. This can be done from the Hub. You can switch back and forth between the Orchestration Debugger and the Message Flow views.

##### Orchestration Debugger Functions

Using the Orchestration Debugger, you can:

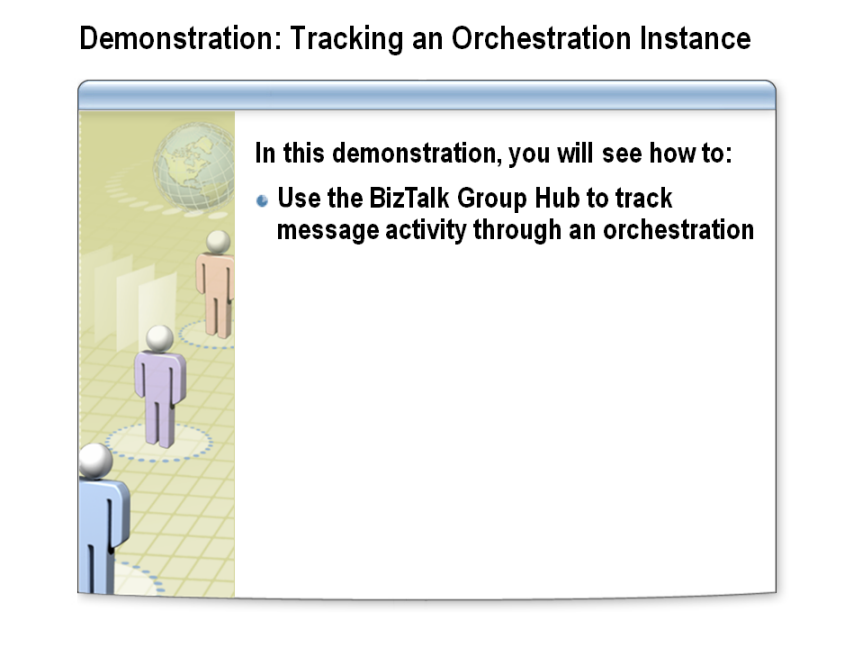
Display a rendered view of the orchestration in which you can replay each processing step for that particular orchestration.

Set breakpoints before any orchestration shape, and then continue execution. Setting a breakpoint sets it on the class, not the instance, so each instance will break at this same point. For this reason, you would not want to set breakpoints on orchestrations in a production environment, and you will need to clear the breakpoints on the class when no longer needed.

View specific variables and message data while in debugging mode.

Continue and resume in debug mode, and then terminate the particular orchestration instance.

Demonstration: Tracking an Orchestration Instance



Learn how to use the BizTalk Group Hub to track message and orchestration activity.

##### Tracking an Orchestration Instance

1. Resume the **bt10d-demos** virtual machine.
2. In Windows Explorer, navigate to **C:\AllFiles\DemoCode\Module8**, and then copy **CashSalesOrder1Info.xml** to the **SalesOrderIN** folder.
3. In the BizTalk Administration Console, click on **BizTalk Group**, and then in the center pane, click on the **New Query** tab.
4. Select **Tracked Service Instances** from the **Value** drop-down list, and click **Run Query**.
5. In the Query results, right-click the most recent instance of **Microsoft.BizTalk.DefaultPipelines.XMLReceive**, and then click **Message Flow**.
6. Maximize the Message Flow window.

The top section of the Message Flow window displays helpful troubleshooting information about this service instance, such as Host, Type, Start Time, and End Time. The lower section displays the flow of the message instance. This message was received as an Unparsed Interchange through the specified port (the randomly generated port name is a result of early binding the orchestration ports). It was then sent to the Demo.Processes.ProcessOrder\_Cash orchestration.

1. Click the **Demo.Processes.ProcessOrder\_Cash** link at the bottom of the page.

Notice that the Type of the service is Orchestration and the State is Completed. In the lower section, notice that the message was received with the SalesOrder port, sent using the Restock port.

1. Click the **Microsoft.BizTalk.DefaultPipelines.XMLTransmit** link in the **Restock** port section.

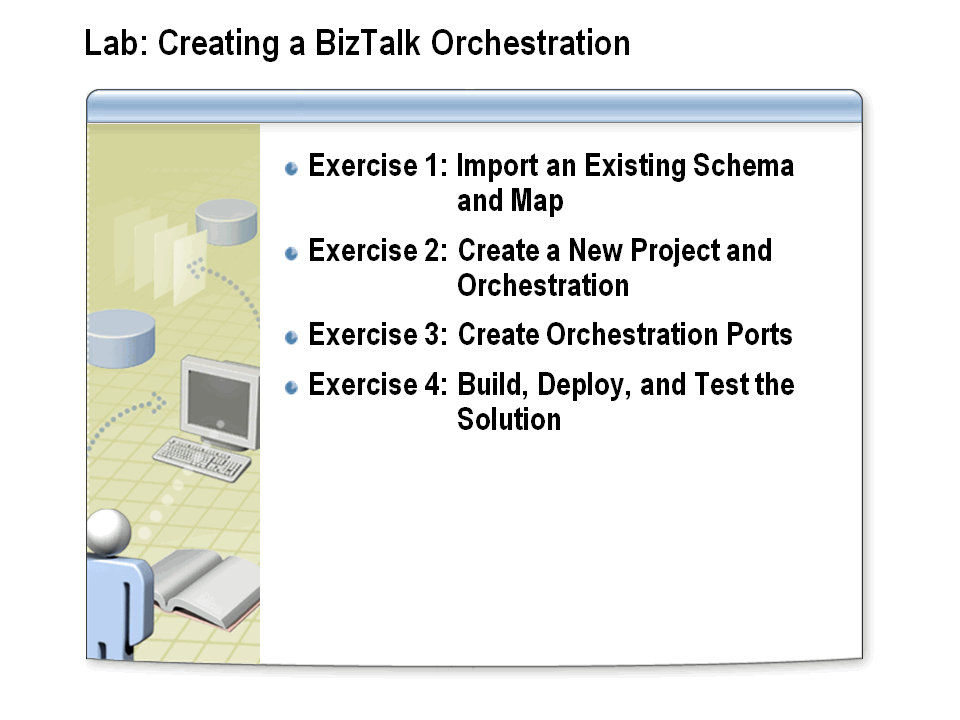
Information about the service instance appears.

1. Click the **Demo.Processes.ProcessOrder\_Cash** link.
2. At the top of the **Message Flow** window, click the **Orchestration Debugger** link.

The orchestration debugger shows the orchestration as seen in the design environment, and the Tracked Events pane shows the start and completion of each shape as the message passes through the orchestration.

1. Close all windows, and shut down the **bt10d-demos** virtual machine.

Lab: Creating a BizTalk Orchestration



##### Time estimated: 45 minutes

##### Scenario

BizTalk messaging can be used for basic message transformation and routing. Processing that requires decisions or multiple actions generally requires the use of orchestrations. The IT manager of Adventure Works has asked you to create orchestrations for processing both cash and credit sales orders. In this lab, you will create the orchestration for processing cash sales orders. In subsequent labs, you will create the orchestration responsible for processing credit sales orders.

##### 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-08** 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 **startBtServices.cmd**.
3. When prompted, press any key to close the command-line window.

Exercise 1: Import an Existing Schema and Map

##### Overview

The orchestration you are going to build requires the generation of a restock message and a map to convert from the SalesOrder format to the Restock format. In this exercise, you will add the Restock schema and the SalesOrder\_To\_Restock map to the Messaging project.

##### Add and Examine Existing Artifacts

Procedure List

1. In Windows Explorer, navigate to **C:\AllFiles\LabFiles\Lab8\AdvWorks**, and then open **AdvWorks.sln**.
2. In Solution Explorer, right-click **Messaging**, point to **Add**, and then click **Existing Item**.
3. In the **Add Existing Item** dialog box, browse to **C:\AllFiles\LabFiles\Lab8\Artifacts**.
4. While holding the CTRL key, click both **Restock.xsd** and **SalesOrder\_to\_Restock.btm**, and then click **Add**.
5. In Solution Explorer, double-click **Restock.xsd**.

Examine the structure of the Restock schema.

1. In Solution Explorer, double-click **SalesOrder\_to\_Restock.btm**.

Examine the links in the SalesOrder\_to\_Restock map.

1. Close the **Restock** schema and the **SalesOrder\_to\_Restock** map.
2. In Solution Explorer, right-click **Messaging**, and then click **Build**.

Exercise 2: Create a New Project and Orchestration

##### Overview

The Adventure Works business process specifies that restock messages for the inventory system and a copy of the completed sales order be generated when a cash sales transaction is completed. In this exercise, you will create an orchestration that processes cash sales orders sent by the Adventure Works stores.

##### Create a New Project

Procedure List

1. In Visual Studio, on the **File** menu, point to **Add**, and then click **New Project**.
2. In the **Add New Project** dialog box, in the left pane, click **BizTalk Projects**, then in the center pane click **Empty BizTalk Server Project**. In the **Name** box, type **Processes**, and then click **OK**.
3. On the **File** menu, click **Save All**.
4. In Solution Explorer, right-click **Processes**, and then click **Properties**.
5. In the **Process** property pages window, on the **Application** tab, in the **Assembly Name** and **Default Namespace** boxes, type **AdvWorks.Processes**.
6. In the **Process** property pages window, click the **Signing** tab, and then check the **Sign the assembly** check box.
7. Choose **<Browse…>** from the drop-down list, and navigate to **C:\AllFiles\LabFiles\Common**, click **AdvWorks.snk**, and then click **Open**.
8. On the **File** menu, click **Save All**.
9. On the **Window** menu, click **Close All Documents**.
10. In Solution Explorer, right-click **Processes**, and then click **Add Reference**.

Since the Processes project will contain orchestrations that use the schemas in the Messaging project, you will need a reference to the Messaging project.

1. In the **Add Reference** dialog box, on the **Projects** tab, click **Messaging**, and then click **OK**.

##### Create an Orchestration

Procedure List

1. In Solution Explorer, right-click **Processes**, point to **Add**, and then click **New Item**.
2. In the **Add New Item** dialog box, click **Orchestration Files**, click **BizTalk Orchestration**, in the **Name** box, type **ProcessOrder\_Cash.odx**, and then click **Add**.

The ProcessOrder\_Cash orchestration opens automatically.

##### Create Messages

Procedure List

1. In Orchestration View, right-click **Messages**, and then click **New Message**.
2. In the **Message\_1** Properties window, in the **Identifier** box, type **msgSalesOrder**.
3. In the **Message Type** list, under **Schemas**, click **<Select from referenced assembly>**.
4. In the **Select Artifact Type** dialog box, in the left pane, click **AdvWorks.Messaging**, in the right pane, click **SalesOrder**, and then click **OK**.

This is the message variable for the incoming sales order.

1. In Orchestration View, right-click **Messages**, and then click **New Message**.
2. In the **Message\_1** Properties window, in the **Identifier** box, type **msgSalesOrder\_Complete**.
3. In the **Message Type** list, under **Schemas**, click **<Select from referenced assembly>**.
4. In the **Select Artifact Type** dialog box, click in the left pane, **AdvWorks.Messaging**, in the right pane, click **SalesOrder**, and then click **OK**.

This is the message variable for the completed processing acknowledgement message.

1. In Orchestration View, right-click **Messages**, and then click **New Message**.
2. In the **Message\_1** Properties window, in the **Identifier** box, type **msgRestock**.
3. In the **Message Type** list, under **Schemas**, click **<Select from referenced assembly>**.
4. In the **Select Artifact Type** dialog box, in the left pane, click **AdvWorks.Messaging**, in the right pane, click **Restock**, and then click **OK**.

This is the message variable for the restock message sent to the inventory system.

##### Add Shapes to the Orchestration

Procedure List

1. Drag a **Receive** shape from the **Toolbox** to the area below the green circle (**Start**) in the orchestration.

If the Toolbox is not docked on the left side of the window, on the View menu, click Toolbox.

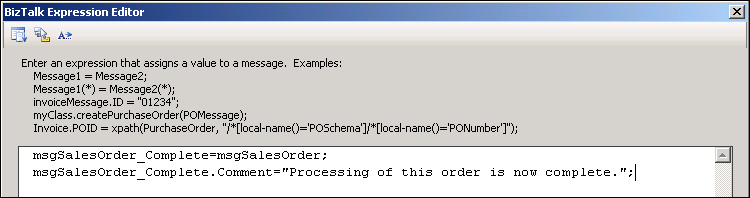
1. In the Properties window, in the **Name** box, type **Sales Order**.
2. In the Properties window, in the **Message** list, click **msgSalesOrder**.
3. Drag a **Transform** shape from the **Toolbox** to the area below the Receive shape in the orchestration.

Notice that when you dropped the Transform shape in the orchestration, the Construct Message shape was also added.

1. Click the **ConstructMessage\_1** shape, and then in the Properties window, in the **Name** box, type **Construct msgRestock**.
2. In the **Messages Constructed** list, click **msgRestock**.
3. Click the **Transform\_1** shape, and then in the Properties window, in the **Name** box, type **Map to Restock**.
4. In the Properties window, click **Map Name**, and then click the **ellipsis** (**…**) button.
5. In the **Transform Configuration** dialog box, click **Existing Map**, and then in the **Fully Qualified Map Name** list, click **<Select from referenced assembly>**.
6. In the **Select Artifact Type** dialog box, in the left pane, click **AdvWorks.Messaging**, in the right pane, click **SalesOrder\_to\_Restock**, and then click **OK**.
7. In the **Transform Configuration** dialog box, click **Source**, and then in the **Variable Name** list, click **msgSalesOrder**.
8. Click **Destination**, and then in the **Variable Name** list, click **msgRestock**, and then click **OK**.
9. Drag a **Message Assignment** shape from the **Toolbox** to the area beneath the Construct Message shape in the orchestration.

Notice that when you dropped the Message Assignment shape in the orchestration, a new Construct Message shape was also added.

1. Click the **ConstructMessage\_1** shape, and then in the Properties window, in the **Name** box, type **Construct msgSalesOrder\_Complete**.
2. In the **Messages Constructed** list, click **msgSalesOrder\_Complete**.
3. Click the **MessageAssignment\_1** shape, and then in the Properties window, in the **Name** box, type **Build Complete SO**.
4. In the **Build Complete SO** Properties window, click **Expression**, and then click the **ellipsis** (**…**) button.
5. In the BizTalk Expression Editor, in the text box, type the text shown in the text box in the following screen shot, and then click **OK**.



1. Right-click the arrow immediately below the **Construct msgRestock** shape, point to **Insert Shape**, and then click **Send**.
2. Configure the **Send\_1** shape with the properties shown in the following table.

|  |  |
| --- | --- |
| **Property** | **Setting** |
| Name | Restock |
| Message | msgRestock |

1. Right-click the arrow immediately below the **Construct msgSalesOrder\_Complete** shape, point to **Insert Shape**, and then click **Send**.
2. Configure **Send\_1** with the properties shown in the following table.

|  |  |
| --- | --- |
| **Property** | **Setting** |
| Name | Complete Sales Order |
| Message | msgSalesOrder\_Complete |

Exercise 3: Create Orchestration Ports

##### Overview

Orchestration ports (logical ports) provide the connections between an orchestration’s send and receive shapes and the BizTalk MessageBox database. In this exercise, you will create a receive port and two send ports. These ports will be connected to send and receive shapes in the orchestration.

##### Create an Orchestration Receive Port

Procedure List

1. Right-click the left **Port Surface**, and then click **New Configured Port**.
2. On the **Welcome to the Port Configuration Wizard** page of the Port Configuration Wizard, click **Next**.
3. On the **Port Properties** page, in the **Name** box, type **SalesOrder**, and then click **Next**.
4. On the **Select a Port Type** page, in the **Port Type Name** box, type **SalesOrderType**, and then click **Next**.
5. In the **Port direction of communication** list, click **I’ll always be receiving messages on this port**, and then in the **Port binding** list, click **Specify now**.
6. In the **URI** box, type **C:\AllFiles\LabFiles\Lab8\SalesOrderIN\\*.xml**, in the **Transport** list, click **FILE**, and then click **Next**.
7. On the **Completing the Port Wizard** page, click **Finish**.
8. In the **SalesOrder** receive port shape, click **Operation\_1**, and then in the **Identifier** box of the Properties window, type **ProcessOrder**.
9. In the **SalesOrder** receive port shape, click **Request**, and then in the **Name** box of the Properties window, type **SalesOrder\_XML**.
10. Drag the green arrow from **SalesOrder** receive port to the **Sales Order** receive shape.

##### Create Orchestration Send Ports

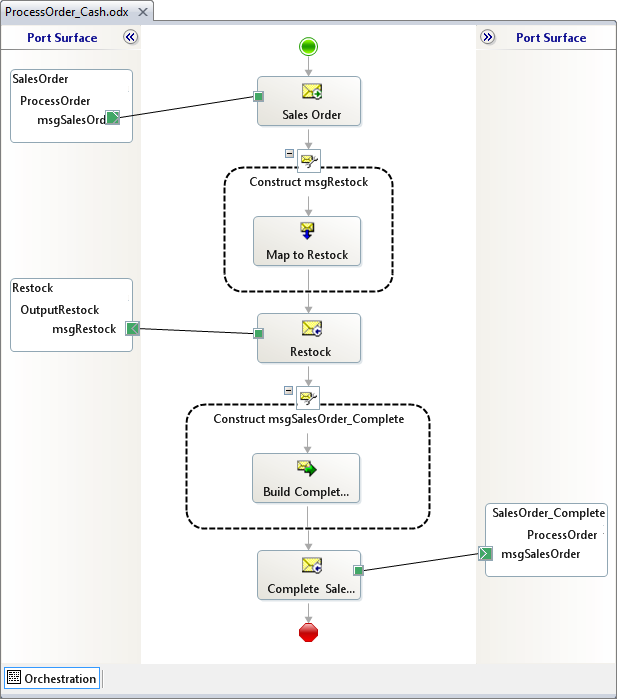
Procedure List

1. Right-click the left **Port Surface**, and then click **New Configured Port**.
2. On the **Welcome to the Port Configuration Wizard** page of the **Port Configuration Wizard**, click **Next**.
3. On the **Port Properties** page, in the **Name** box, type **Restock**, and then click **Next**.
4. On the **Select a Port Type** page, in the **Port Type Name** box, type **RestockType**, and then click **Next**.
5. In the **Port direction of communication** list, click **I’ll always be sending messages on this port**, and then on the **Port binding** list, click **Specify now**.
6. In the **URI** box, type **C:\AllFiles\LabFiles\Lab8\OUT\Restock%MessageID%.xml**, in the **Transport** list, click **FILE**, and then click **Next**.
7. On the **Completing the Port Wizard** page, click **Finish**.

The port shapes can be repositioned by dragging the shape up or down the Port Surface.

1. In the **Restock** send port shape, click **Operation\_1**, and then in the **Identifier** box of the Properties window, type **SendRestock**.
2. In the **Restock** send port shape, click **Request**, and then in the **Name** box of the Properties window, type **Restock\_XML**.
3. Drag the green arrow from **Restock** send shape to the **Restock** send port.
4. Right-click the right **Port Surface**, and then click **New Configured Port**.
5. On the **Welcome to the Port Configuration Wizard** page of the **Port Configuration Wizard**, click **Next**.
6. On the **Port Properties** page, in the **Name** box, type **SalesOrder\_Complete**, and then click **Next**.
7. On the **Select a Port Type** page, click **Use an existing Port Type**, click **AdvWorks.Processes.SalesOrderType**, and then click **Next**.
8. In the **Port direction of communication** list, click **I’ll always be sending messages on this port**, and then in the **Port binding** list, click **Specify now**.
9. In the **URI** box, type **C:\AllFiles\LabFiles\Lab8\OUT\CompleteSO%MessageID%.xml**, in the **Transport** list, click **FILE**, and then click **Next**.
10. On the **Completing the Port Wizard** page, click **Finish**.
11. Drag the green arrow from **Complete Sales Order** send shape to the **SalesOrder\_Complete** send port.

Your orchestration should look like the following figure:



Exercise 4: Build, Deploy, and Test the Solution

##### Overview

In this exercise, you will deploy and test the ProcessOrder\_Cash orchestration. The early bound orchestration ports that you defined in this lab will cause new messaging ports (physical ports) to be created and will cause these ports to be bound to the orchestration ports (logical ports). You will submit a test message to the newly created receive port. The orchestration will generate two new messages and send them to the file system using the send ports.

##### Test the Orchestration

Procedure List

1. In Solution Explorer, right-click **Processes**, and then click **Build**.
2. You will receive the following error:

**“You must specify at least one already-initialized correlation set for a non-activation receive that is on a non-self correlating port.”**

This is one of the most common errors committed when creating a new orchestration. The receive shape for the orchestration is not configured as an activation receive.

1. Click the **Sales Order** receive shape, and then in the Properties window, in the **Activate** list, click **True**.
2. In Solution Explorer, right-click **Processes**, and then click **Build**.
3. After the build has completed, right-click **Processes**, and then click **Deploy**.
4. On the **Start** menu, point to **All Programs**, point to **Microsoft BizTalk Server 2010**, and then click **BizTalk Server Administration**.
5. In the BizTalk Server Administration Console, expand **BizTalk Server Administration**, **BizTalk Group**, **Applications**, **BizTalk Application 1**, and then click **Resources**.

Notice the AdvWorks.Messaging and AdvWorks.Processes resources.

1. Right-click **BizTalk Application 1**, and then click **Start**.
2. In the **Configure Application** message box, read the message, and then click **Yes**.

The host instance used by the orchestration has not yet been configured. A host instance must be set before the orchestration, and therefore the application, can be started.

1. In the **Configure Application** dialog box, click **ProcessOrder\_Cash**.

Notice that the physical receive port and send ports have been created and bound to the orchestration. This is because you configured the port bindings when configuring the orchestration ports. This is known as early binding.

1. In the **Host** list, click **BizTalkServerApplication**, and then click **OK**.
2. Right-click **BizTalk Application 1**, and then click **Start**.
3. In the **Start ‘BizTalk Application 1’ Application** dialog box, click **Options**, ensure that all check boxes are selected, and then click **Start**.
4. In Windows Explorer, navigate to **C:\AllFiles\LabFiles\Lab8**, and then copy **CashSalesOrder1Info.xml** to the **SalesOrderIN** folder.
5. Navigate to the **SalesOrderIN** folder.

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

1. Navigate to the **OUT** folder.
2. Open **CompleteSO{GUID}.xml**.

The message is displayed using a Microsoft InfoPath® form.

1. Close InfoPath.
2. In Windows Explorer, open **Restock{GUID}.xml**.

The message is displayed in XML format.

1. Close all open windows.