Orchestration Basics

Adding process to messaging

Agenda

- What is Orchestration
- Messages in Orchestration
- Understanding Ports
- Sending and receiving messages
- Message Correlation

What is orchestration?

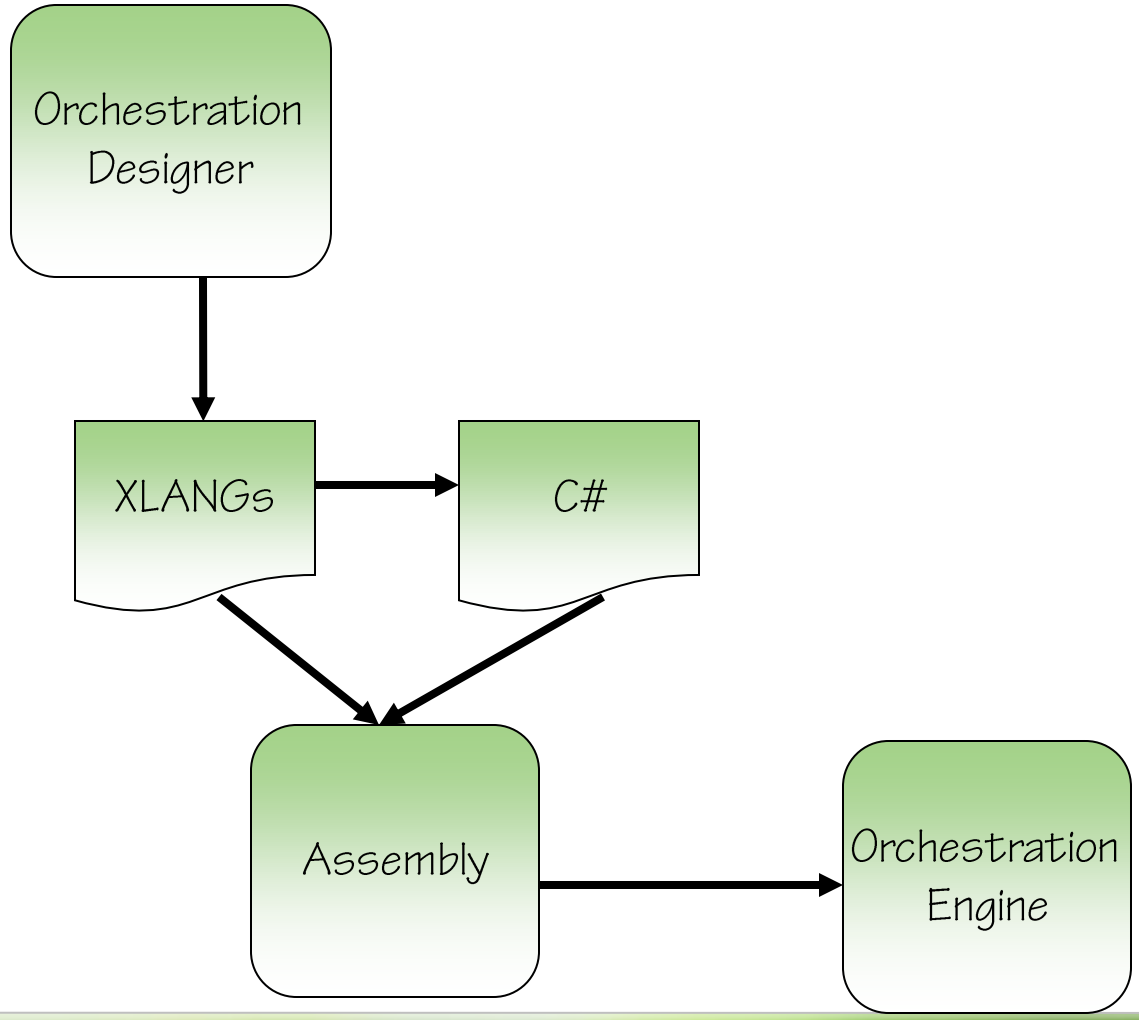
- **Orchestration is a mechanism for defining how messages are exchanged in a business process**
  - Defines interactions between different systems or partners
  - Interactions may be long running (hours, days, months. . .)
  - Provides semantics for modeling exception handling, transactions, synchronization, parallel processing
  - Also known as the workflow engine of BizTalk

Orchestration fundamentals

- **Orchestrations represented internally as XLang documents**
  - XLang is an XML dialect specific to BizTalk Server
  - Orchestration designer helps you define XLang visually
  - Orchestration engine processes XLang orchestrations
- **Orchestration shapes represent actions or steps in the process**

Shape Type	Available Shapes
Messaging	send, receive, construct, transform, port , role link
Process flow	decide (if/else), while, parallel, listen, delay
Coding	expression, call/start orchestration, call rules
Control	throw exception, suspend, terminate, scope

Orchestration compilation



Orchestration activation

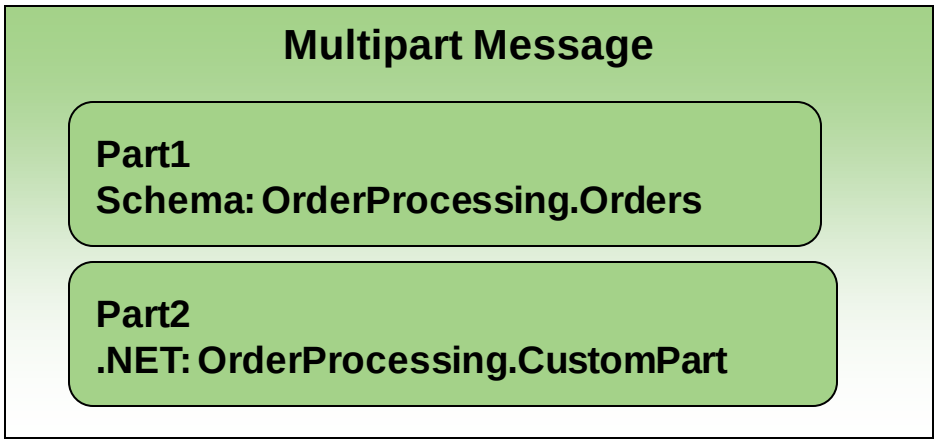
- **Orchestrations can be activated in one of two ways:**
  - By subscribing to messages published to the MB
  - Explicitly called from another orchestration (familiar code model)
- **If an orchestration does not take parameters**
  - The first shape must be a Receive marked with Activating=true
  - MB subscriptions determine when it is instantiated
  - This approach provides looser-coupling

Orchestration execution

- **Orchestrations may be long running**
  - Persistence of orchestration state is required
  - BizTalk provides the engine for managing your process state
  - Objects used in orchestration must be able to be serialized*
- **Orchestration service manages monitoring for messages**
  - Instances dehydrated when not in use
  - Engine rehydrates when messages arrive or delays expire

Orchestration messages

- **All orchestration messages are multipart**
  - Each message part has a specific type
  - Message part types based on XSD or .NET types
- **Many messages only contain a single part**
  - Single part messages need no special multipart type definition



Messages are immutable

- **Orchestration messages are immutable**
  - Once you receive a message, you cannot make changes to it
- **You create new messages when making changes**
  - Create messages from other messages
  - Create new messages from scratch

Creating messages

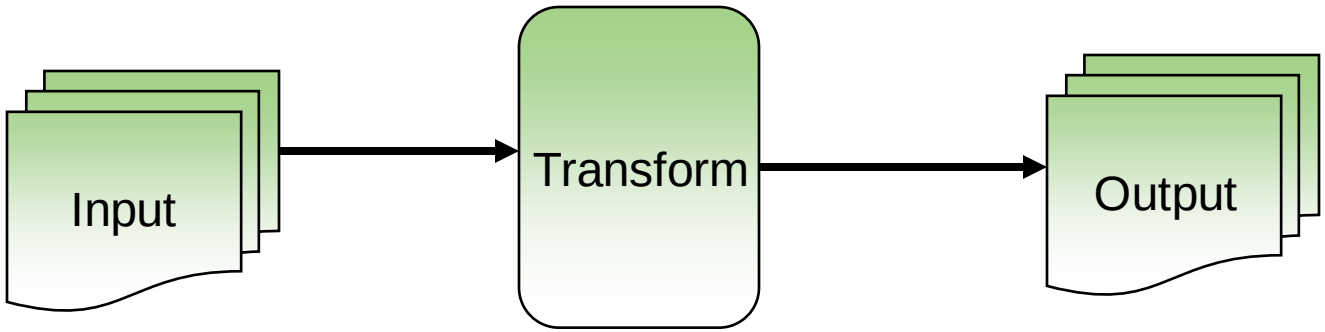
- **Messages must be created in a message construct shape**
- **There are several ways to create messages in an orchestration**
  - Direct message assignment
  - Maps
  - XPath
  - .NET code
- **Message construct can include two specific shapes**
  - Transform – for mapping
  - Message assignment – for all others

Direct message assignment

- **Assign one message variable to another**
  - MessageA = MessageB
  - Works if the message types are the same
- **Assign message parts**
  - MessageA.XmlPart = OrderProcessing.Order
  - Makes a copy of the message or part
- **Copy context properties**
  - MessageA(*) = MessageB(*)
  - MessageA(Ordering.OrderId) = MessageB(Ordering.OrderId);

Maps

- **Use transform shape to map from one message to another**
  - New message(s) constructed for the output
- **Can map from N to N messages**
  - Select multiple source or destination schemas
  - Only supported when creating a new map in orchestration
  - Wrapper schema embedded into the mapper file
  - Message construction shape must define all messages created



XPath

- **Can be used to extract data from a message part**
  - Return node sets, count of elements, specific values, etc.
- **Used to assign values**
  - Entire message part
  - Specific value in the message

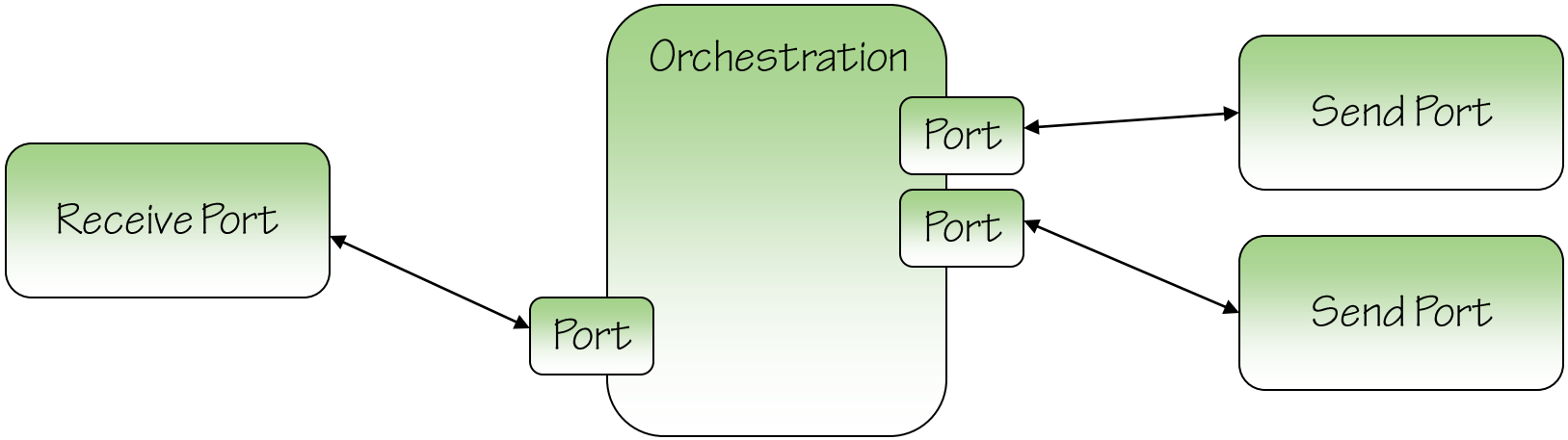
```
//Extra message from another message
ShippingAddressMessage = xpath(
        OrderMessage,
        "/*[local-name()='Order' and
        namespace-uri()='http://Schemas.Order']
        /*[local-name()='ShipTo' and namespace-uri()='']");
```

.NET code

- **Use custom code to create the message or its parts**
- **Methods returning XmlDocument or .NET types**
  - Can be directly assigned to message part
  - Match the message type when loaded or serialized
- **XLANGMessagePart can load itself from**
  - XML Reader
  - XML Document
  - .NET object
  - Stream

Orchestration Ports

- **Logical representations of the endpoints used to send/receive**
  - Orchestration ports are bound to physical ports at deployment
- **Logical ports**
  - Logical message pattern must match the physical pattern
  - Message types must match associated send/receive shapes



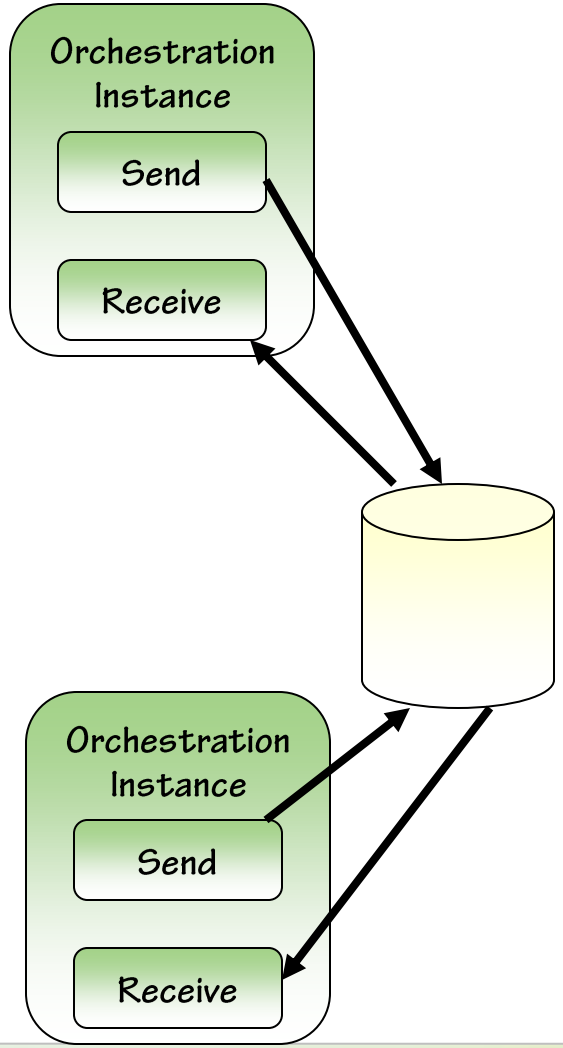
Orchestration Port Types

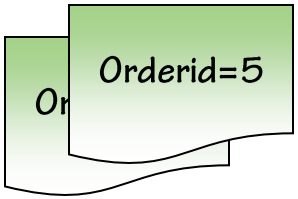
- **Port types defines the structure and operations of ports**
  - One-way or request-response
  - Operations
- **Operations**
  - Define the message types to be sent/received
  - Provide a logical representation of invoking actions
  - Messages sent to port use same binding, regardless of operation
- **Ports are instances of the port type**

Correlation

- **Coordinating messages that are being sent and received?**
  - Define a correlation type in orchestration
  - Uses a collection of context properties to define a subscription
  - Create a correlation set based on the type
  - Initialize the correlation set on a send or receive shape
  - Follow the correlation set for future receives

Correlation Mechanics





Property Name	Value
OrderID	5

Property Name	Value
OrderID	7

Orchestration subscriptions

- **Activation subscriptions**
  - Created for activating receive shapes in an orchestration
  - New instance of the orchestration when a message received
  - Created when an orchestration is enlisted
- **Instance subscriptions (correlation)**
  - Created for receive shapes following a correlation set
  - Includes receives in called orchestrations
  - Created at runtime when a correlation set is initialized
  - Includes an instance ID for the current orchestration instance
  - New messages routed to the existing orchestration instance

Summary

- Orchestrations add process to messaging
- Decoupled from physical messaging configuration
- Messages are immutable
- Messages can be constructed in a variety of ways
- Distinguished properties provide quick access to data
- Correlation helps route messages to the correct instance

References

- **BizTalk developer center orchestration learning**
  - http://msdn.microsoft.com/biztalk/learning/dev/orch/default.aspx
- **BizTalk Server 2004: A Messaging Engine Overview**
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/anch_Biztalk2004.asp