Messaging Architecture

Connecting systems with messages

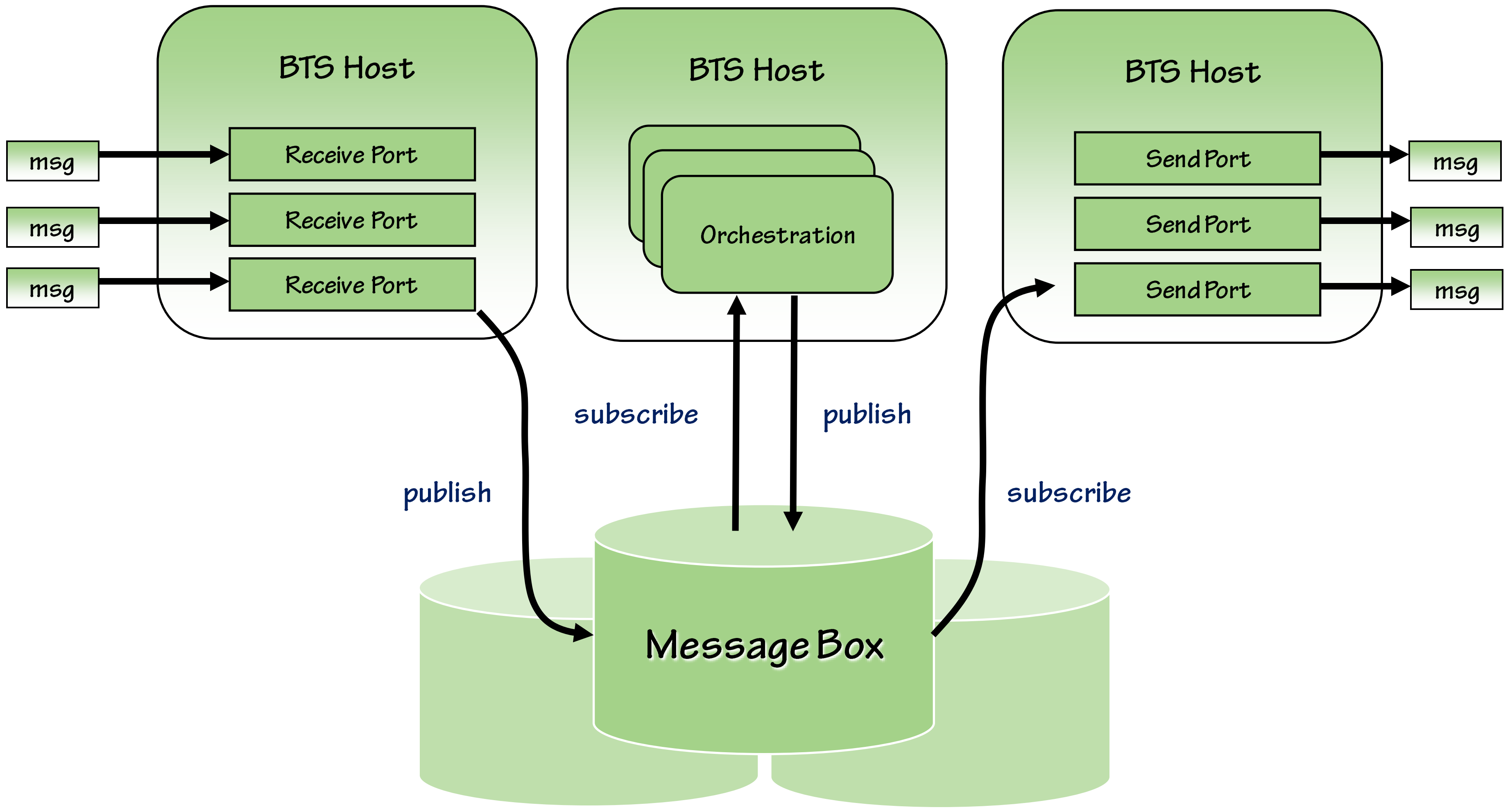
Outline

- Messaging architecture
- Message fundamentals
- Message context and properties
- Publish/subscribe
- Ports and port components

Messaging architecture

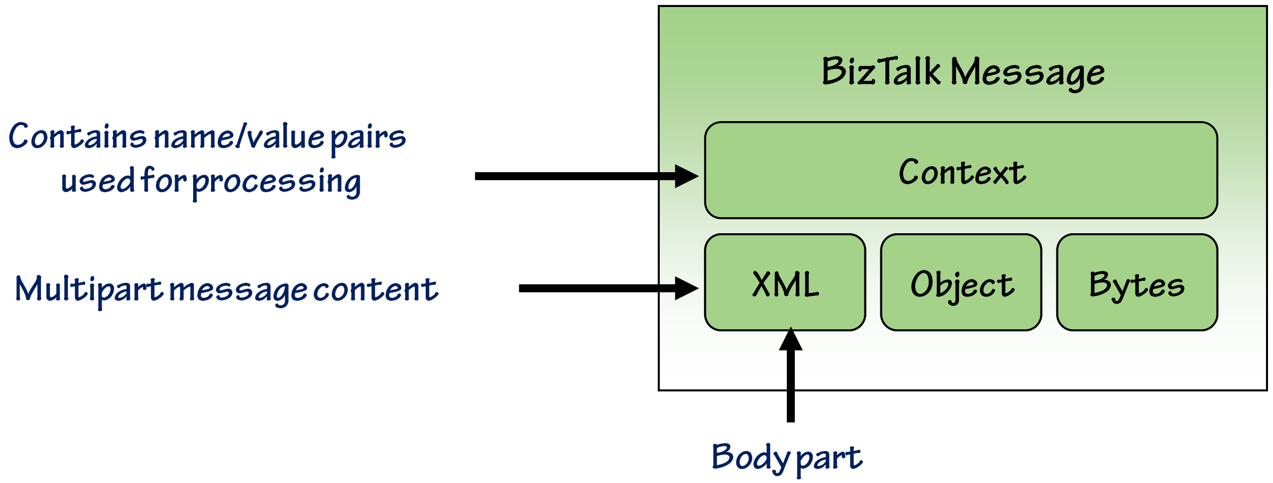
- **BizTalk's *messaging architecture* is built on *Service Orientation***
  - Focused on message processing fundamentals
  - Embraces the concept of explicit boundaries through *ports*
  - Relies on sharing schemas with other applications
  - Provides built-in support for Web services
- **Built on central *pub-sub engine* for flexible message routing**
  - Implemented by the BizTalk *Message Box (MB)*

Messaging architecture overview



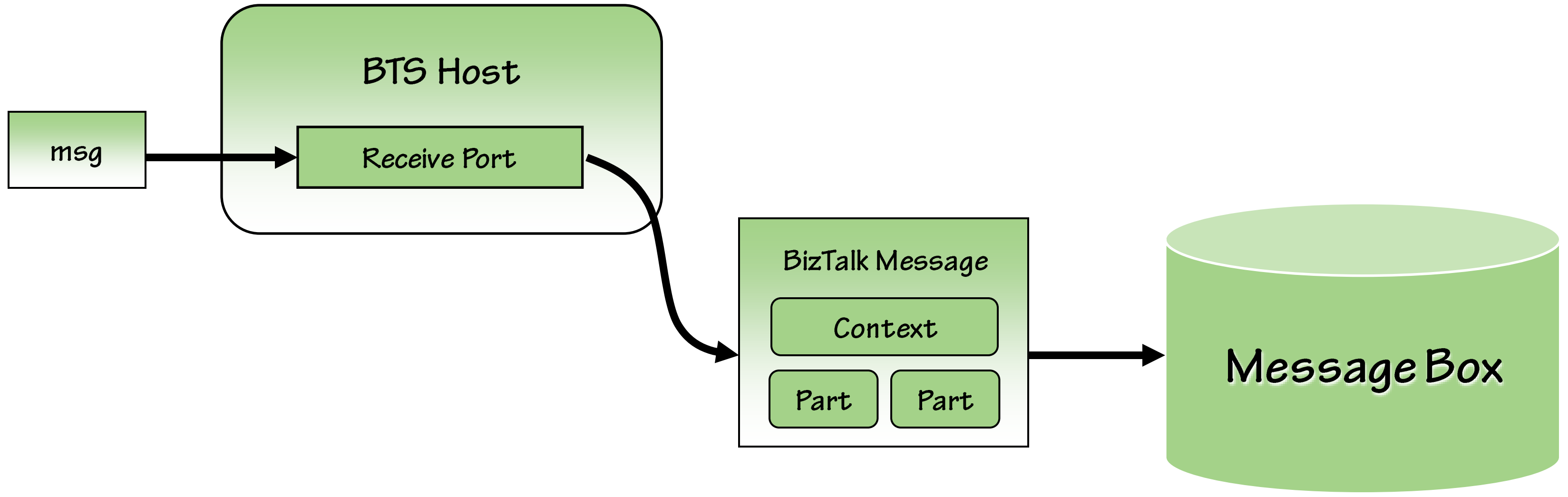
Messaging fundamentals

- **A BizTalk *message* has zero or more *parts* and a *context***
  - Each part consists of data stream, of any type
  - XML document, flat file, serialized .NET object, or raw bytes
  - One part identifies the message body, called the *body part*
  - The body part is used for identification/routing
  - The message context is collection of name/value properties


BizTalk Message
Context
XML
Object
Bytes
Contains name/value pairs used for processing
Multipart message content
Body part

Message processing basics

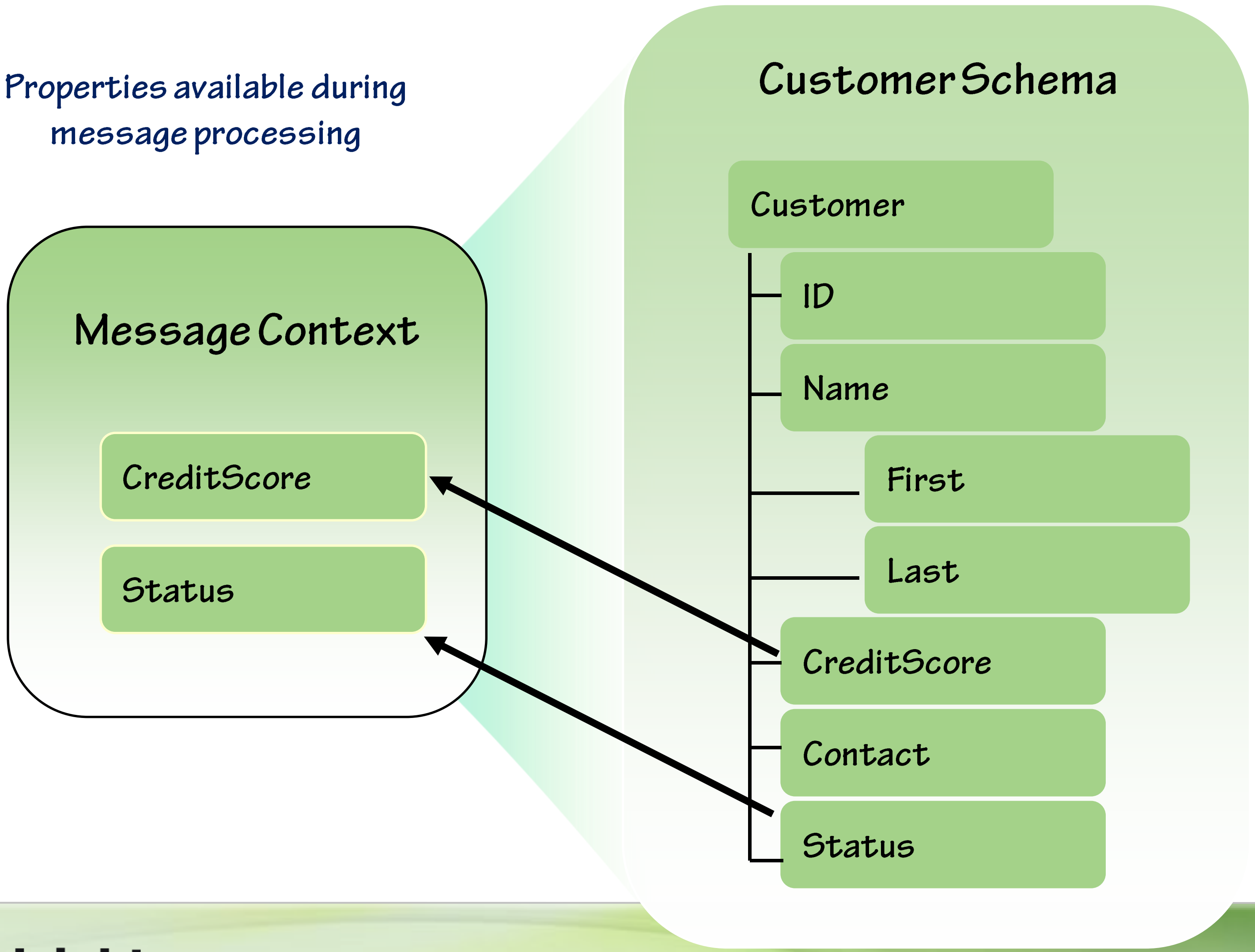
- **Messages are received by ports and published to MB**
  - The *message context* is created before publishing
  - You primarily work with the message context
  - Entire message is only accessed when needed
- **This design accommodates large messages**
  - Entire message is rarely (if ever) held in memory



Message context

- **The message context is a collection of name/value properties**
  - Numerous *system properties* automatically generated
  - Properties can be extracted from the message itself
  - Other properties related to message processing
- **Properties can be *written* or *promoted* to message context**
  - Promoted properties can be used for routing
- **You can identify message properties declaratively in XSD**
  - By defining *distinguished fields* and *promoted properties*

Message context and properties



System-defined message properties

- There are numerous system-defined message properties
  - Populated by BizTalk when message is constructed
  - Below you'll find some examples of useful properties

Property Name	Description
BTS.MessageType	Specified the type of message (schema namespace + root element name)
BTS.ReceivePortID	Specifies the ID of the receive port
BTS.ReceivePortName	Specifies the user-friendly name of the receive port
BTS.SPID	Specifies the ID of the send port
BTS.InterchangeID	Specifies the unique ID that is use to group documents that resulted from the same interchange message
BTS.SourcePartyID	The ID of the BizTalk party

User-defined message properties

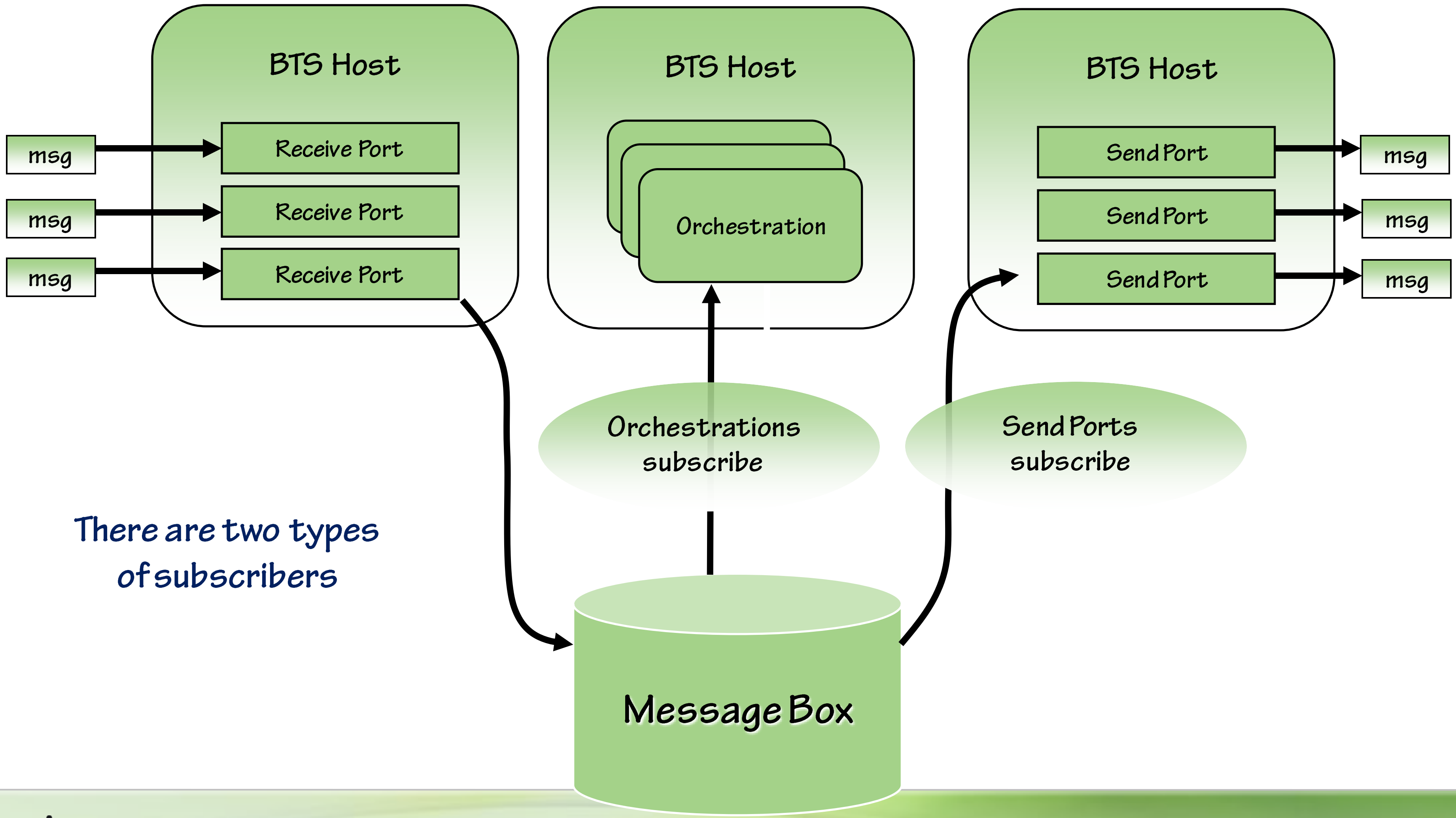
- There are two types of message properties
  - Distinguished fields
  - Promoted properties (requires a property schema)
- There are numerous differences

Characteristic	Distinguished Field	Promoted Property
Available to	Orchestrations only	Messaging components (routing) and orchestrations
Syntax	msg.OrderId	msg(Orders.OrderId)
Visible in HAT	No	Yes
Size limit	Unlimited	Maximum of 256 characters

Understanding publish/subscribe

- **Message routing is based on the MB pub-sub implementation**
  - You define *subscriptions* that inspect the *message context*
  - Also known as message *filters* (throughout the tools)
  - Subscriptions stored in the MB
  - Messages received by ports and published to MB
  - MB routes messages to subscribers
- **Decouples senders from receivers, increases flexibility**

Subscribers



There are two types of subscribers

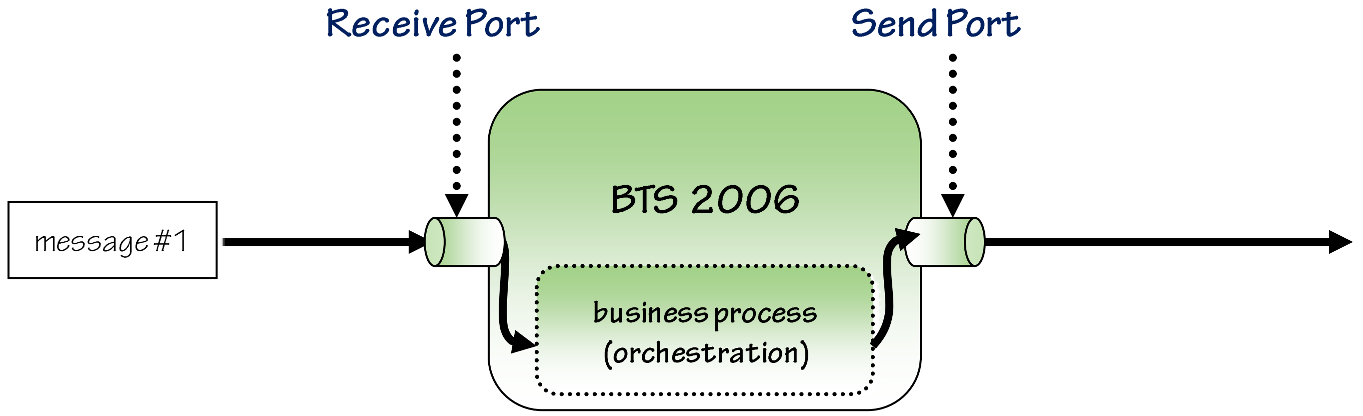
Filters (subscriptions)

- **Filters are predicate expressions that evaluate to true/false**
  - A filter typically evaluates a message property
  - Combine multiple expressions with boolean operators
- **You define filters in send ports and orchestrations**
  - In orchestrations, defined on activate-able Receive shapes
- **Available filter expression operators**
  - `==, !=, <, >, <=, >=, exists`

```
BTS.ReceivePortName == FileReceivePort And
BTS.MessageType == "http://example.org/invoice#Invoice" And
Invoices.OrderId exists
```

Understanding ports

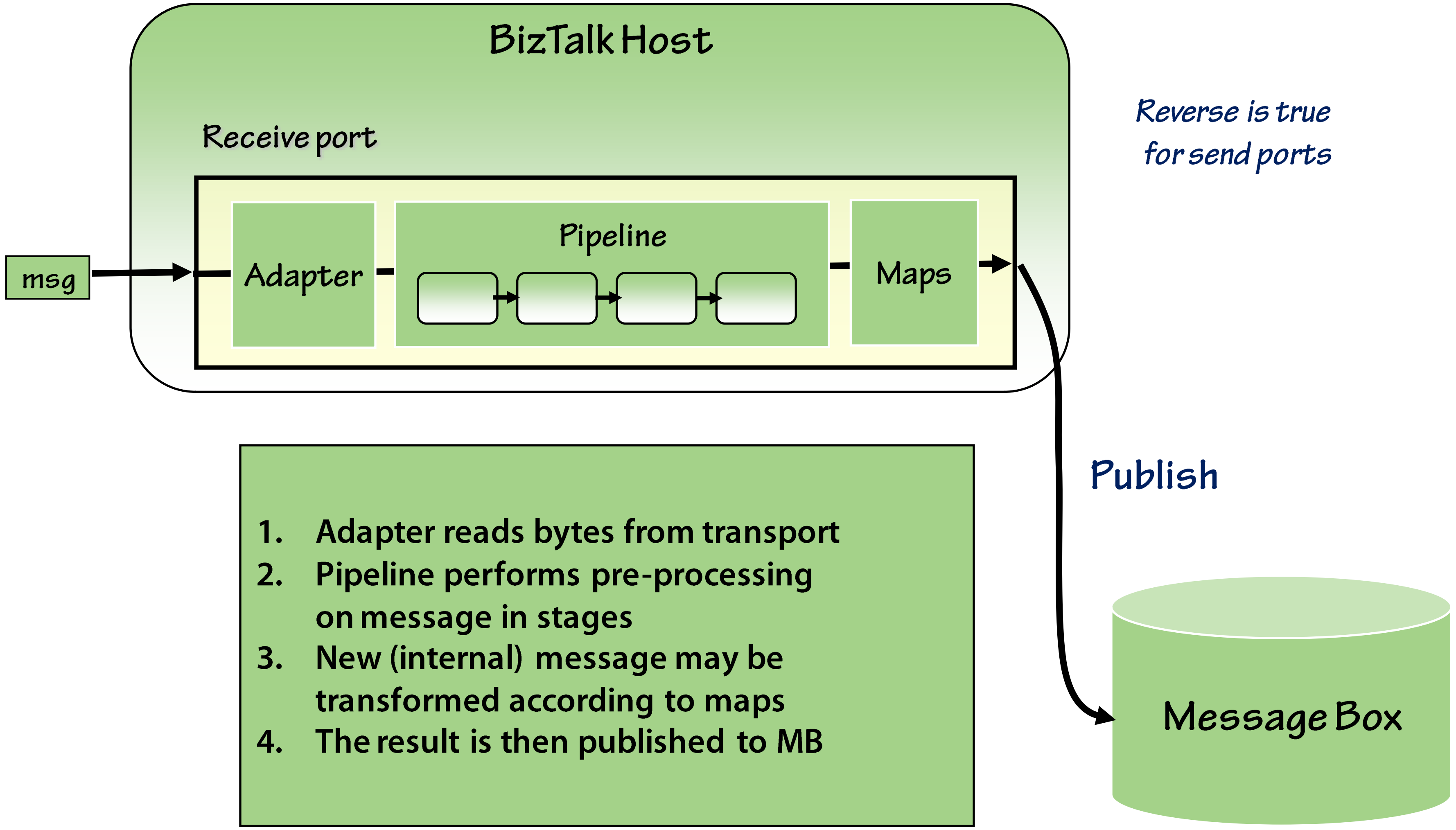
- **BizTalk models boundaries explicitly with ports**
  - *Receive ports* receive messages and publish them to MB
  - *Send ports* send message from MB to a target location
- **Ports define entry/exit points**
  - What happens internally is black-box



Receive ports

- **A *receive port* is a collection of one or more *receive locations***
  - Receive locations define specific entry points
  - You can associate *maps* with a receive port
  - Maps apply to all receive locations
- **A receive location consists of an *adapter* and a *receive pipeline***
  - Adapter is responsible for byte-level transport communications
  - The pipeline prepares the message for publishing to MB
  - Maps execute after the pipeline, before publishing

Port components



1. Adapter reads bytes from transport
2. Pipeline performs pre-processing on message in stages
3. New (internal) message may be transformed according to maps
4. The result is then published to MB

Send ports and groups

- A *send port group* is a collection of *send ports*
  - You can associate filters with send port groups
  - When matched, all send ports fire (like an email distribution list)
- A send port consists of an *adapter*, a *send pipeline*, and *maps*
  - Maps transform BizTalk message into appropriate schema
  - Send pipeline prepares message for transmission
  - Adapter handles transmitting the message bytes

Adapter framework

- **BizTalk supports unlimited transports via adapter framework**
  - An open framework for writing custom adapters
- **Numerous transports ship out of the box**
  - SOAP, File, HTTP, SMTP, FTP, Base EDI, SQL, MSMQ, MSMQ/T
- **New transports shipping with BTS 2006**
  - POP3 receive adapter, WSS adapter
- **A vibrant 3rd party adapter community exists**
  - PeopleSoft, SAP, MQSeries, the list goes on (for a long time)

See BizTalk Server Adapters site for more details

Pipelines

- ***Pipelines*** **define a sequence of message processing steps**
  - Organized into well-defined *stages*
  - Each stage may contain zero or more *pipeline components*
- **A pipeline component defines a processing action**
  - Numerous pipeline components ship out-of-the-box
  - You can write custom pipeline components
- **BizTalk ships several *default pipelines* for your use**

Pipeline Name	Description
XMLReceive	Contains the XML Disassembler (builds the message context) and the Party Resolution components
PassThruReceive	Contains no pipeline components
XMLTransmit	Contains the XML Assembler component
PassThruTransmit	Contains no pipeline components

Maps

- **BizTalk maps are XSLT-based transformations**
  - Defined using the *BizTalk Mapper*
- **Typically used in ports to support multiple external schemas**
  - Maps normalize to/from internal schemas
  - Reduces number of receive ports you need
  - Simplifies internal message processing
- **Maps are configured on port**
  - A port can have multiple maps configured on it
  - But each must have a unique source schema
  - Only one map is used, matched by message type

Is messaging enough?

- **Messaging alone provides valuable integration benefits**
  - Facilitates “connecting the dots”
  - Enough for some integration scenarios
- **Not enough when you need:**
  - Long running processes
  - Message correlation
  - Transaction-based error handling
  - Scheduling and coordination
- **When you need these features, turn to *orchestrations***

Summary

- The BizTalk message architecture revolves around messages
- BizTalk messages consists of multiple parts and a context
- Messages are published to the Message Box
- Messages are considered immutable once created/published
- The MB implements publish/subscribe for flexible routing
- Routing based on subscriptions against message context
- Ports consist of adapters, pipelines, and maps

References

- **BizTalk Server 2004: A Messaging Engine Overview**
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/anch_Biztalk2004.asp
- **Business Process and Integration Developer Center**
  - http://msdn.microsoft.com/bpi/
- **BizTalk Server 2006 Whitepapers**
  - http://www.microsoft.com/biztalk/2006/prodinfo/whitepapers.mspx
- **Pluralsight's BizTalk Wiki**
  - http://pluralsight.com/wiki/default.aspx/Aaron/TheBiztalkWiki.html