Return Path Problems

1 The Return Path Problem

If the same service can be requested by two different applications then there is a problem with knowing how to route a message back to the correct application. This is illustrated by the diagram below:

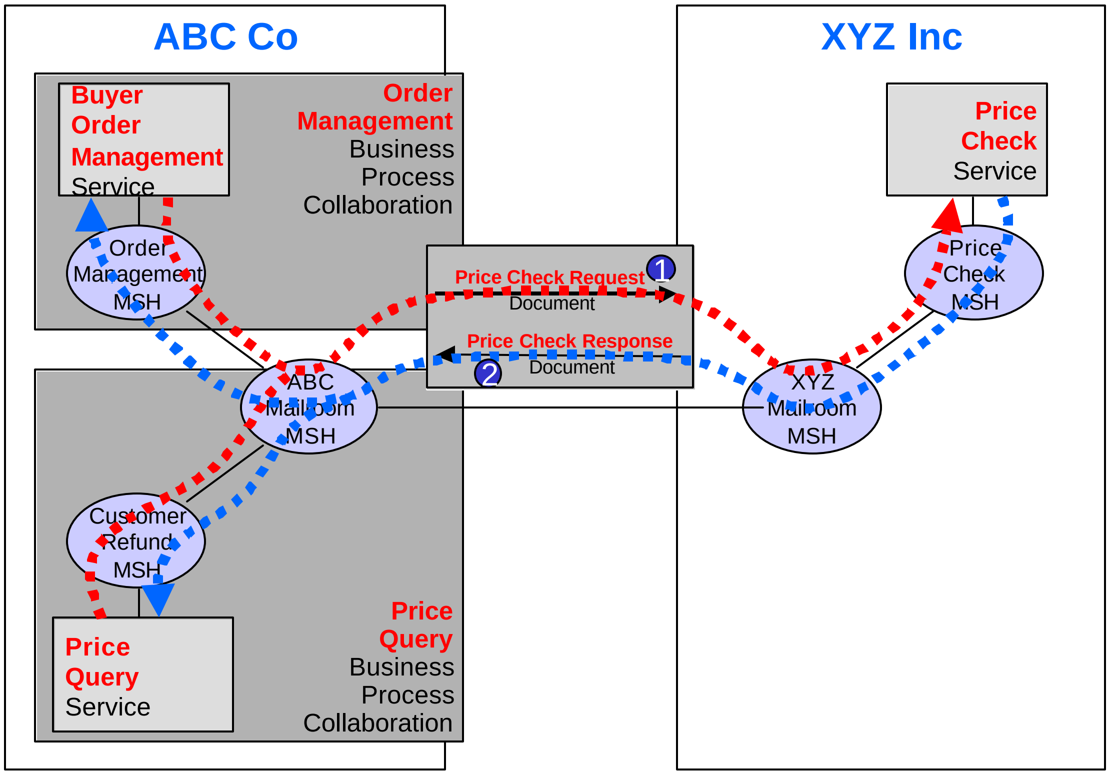


Figure 1-1 The Return Path Problem

In the diagram above, the outbound message (dotted red line) is sent to the Price Check Service operated by XYZ Inc. This results in a Price Check Response being sent back to ABC Co. To keep agreements (and implementations) simple, ABC Co has implemented a mailroom MSH that is used to accept all external messages. This mailroom MSH needs to determine which Business Process or Service to send the message to. How does it do this? Currently the ebXML Messaging Spec would require the Price Check Response message to look something like this:

```
<MessageHeader>
  <From><PartyId>XYZinc</PartyId></From>
  <To><PartyId>ABCco</PartyId></To>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>PriceCheck</Service>
  <Action>PriceCheckResponse</Action>
  <MessageData>
    <MessageId>56723</MessageId>
    <RefToMessageId>79465</RefToMessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

There is no information in this message that indicates which business process collaboration or service should receive the response.

One way to fix this problem is to include the ProcessType, for example:

```
<MessageHeader>
  <From><PartyId>XYZinc</PartyId></From>
  <To><PartyId>ABCco</PartyId></To>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ProcessType>FixPrice</ProcessType>
  <ConversationId>5678</ConversationId
  <Service>PriceCheck</Service>
  <Action>PriceCheckResponse</Action>
  <MessageData>
    <MessageId>56723</MessageId>
    <RefToMessageId>79465</RefToMessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

The ProcessType could then be mapped to application/service in order to determine where to send a message to, but would require a look up.

The other alternative is to include the From Service and From Action in the message. For example:

- Message 1:

```
<MessageHeader>
  <From><PartyId>ABCco</PartyId></From>
  <FromService>BuyerOrderManagement</FromService>
  <FromAction>FixPriceResponse</FromAction>
  <To><PartyId>XYZinc</PartyId></To>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>PriceCheck</Service>
  <Action>PriceCheckRequest</Action>
  <MessageData>
    <MessageId>79465</MessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

- Message 2:

```
<MessageHeader>
  <From><PartyId>XYZinc</PartyId></From>
  <To><PartyId>ABCco</PartyId></To>
  <FromService>PriceCheck</FromService>
  <FromAction>PriceCheckResponse</FromAction>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>BuyerOrderManagement</Service>
  <Action>FixPriceResponse</Action>
  <MessageData>
    <MessageId>56723</MessageId>
    <RefToMessageId>79465</RefToMessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

In this case the FromService and FromAction are included in Message 1. These are then used to populate the Service and Action elements in the response. This means that the Mailroom MSH at ABC Co can use this information to determine which Application/Service should receive the message.

2 The Delivery Receipt, ErrorMessage, etc. Problem

This problem is associated with routing delivery receipts, error messages, message statues Requests & Responses, and MSH Pings/Pongs etc, back to the correct destination.

Consider the previous example where you have two different applications both making a Price Check request on a supplier, only in this instance, there a delivery receipt is requested and so the Receiving MSH at XYZ Inc needs to return a Delivery Receipt (in addition to any other message).

This is illustrated by the following diagram:

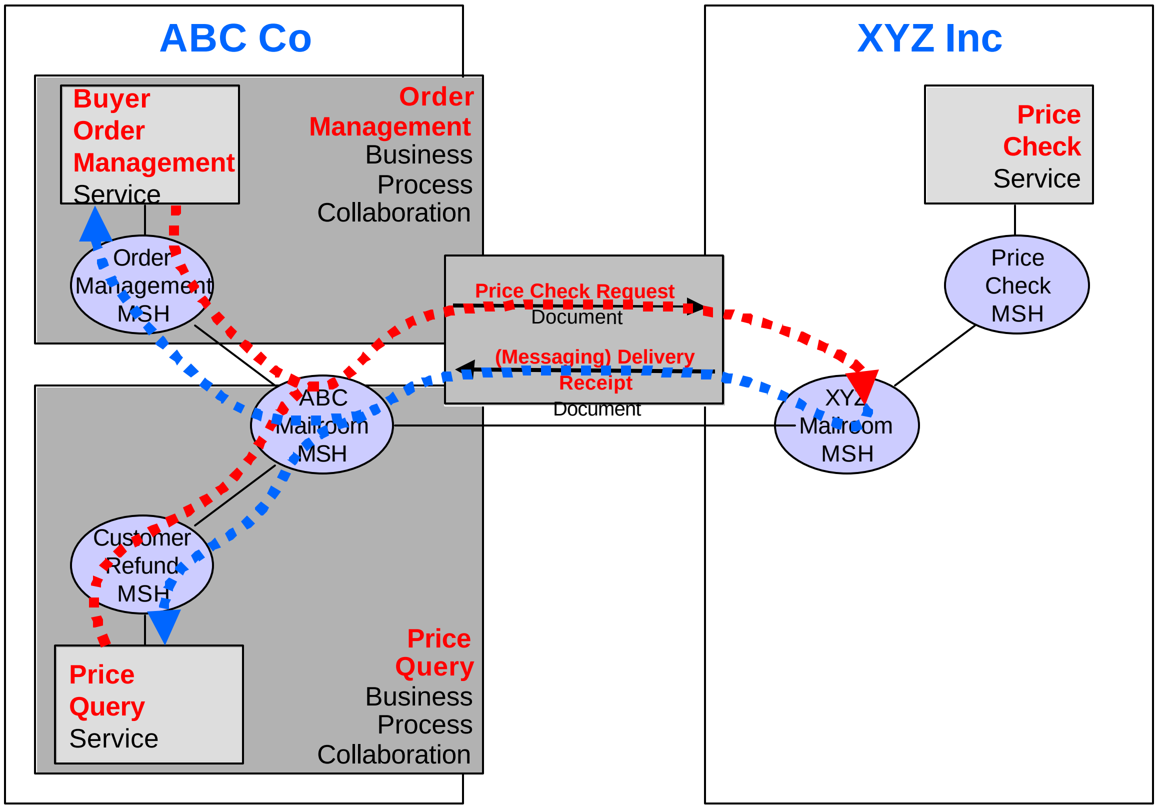

ABC Co
XYZ Inc
Buyer Order Management Service
Order Management Business Process Collaboration
Order Management MSH
ABC Mailroom MSH
Customer Refund MSH
Price Query Service
Price Query Business Process Collaboration
Price Check Request Document
(Messaging) Delivery Receipt Document
Price Check Service
Price Check MSH
XYZ Mailroom MSH

Figure 2-1 The Delivery Receipt Problem

In this case, the ABC Mailroom MSH would probably need to notify the correct Business Process Collaboration or Service that the message was received. However, it can't easily do this. Consider the following example that describes what the current messaging specification would require that the Delivery Receipt looked like:

- Message 1:

```
<MessageHeader>
  <From><PartyId>ABCco</PartyId></From>
  <To><PartyId>XYZinc</PartyId></To>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>OrderManagement</Service>
  <Action>PriceCheck</Action>
  <MessageData>
    <MessageId>79465</MessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

- Message 2:

```
<MessageHeader>
  <From><PartyId>XYZinc</PartyId></From>
  <To><PartyId>ABCco</PartyId></To>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>uri:www.ebxml.org/messageService/</Service>
  <Action>DeliveryReceipt</Action>
  <MessageData>
    <MessageId>56723</MessageId>
    <RefToMessageId>79465</RefToMessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

There is no information in message 2 that can be used directly by the ABC MSH to determine which application/service should receive the message.

The only way that the author can think of that would work using the current specification, is for the ABC MSH, when it sends the message, to note which service sent the message and save the MessageId. Then when the error message comes back, it could correlate the RefToMessageId in the Delivery Receipt with the original message to work out which service/application to notify.

The problem with this is that the sending MSH will have to remember the MessageId and sending application of EVERY message sent, even if the message is being sent unreliably.

Another approach is to adopt a variation of the approach described in section 1 as illustrated in the following examples:

- Message 1:

```
<MessageHeader>
  <From><PartyId>ABCco</PartyId></From>
  <FromService>BuyerOrderManagement</FromService>
  <FromAction>FixPriceResponse</FromAction>
  <To><PartyId>XYZinc</PartyId></To>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>PriceCheck</Service>
  <Action>PriceCheckRequest</Action>
  <MessageData>
    <MessageId>79465</MessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

- Message 2:

```
<MessageHeader>
  <From><PartyId>XYZinc</PartyId></From>
  <To><PartyId>ABCco</PartyId></To>
  <FromService>PriceCheck</FromService>
  <FromAction>PriceCheckResponse</FromAction>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>BuyerOrderManagement</Service>
  <Action>uri:www.ebxml.org/messageService/DeliverReceipt</Action>
  <MessageData>
    <MessageId>56723</MessageId>
    <RefToMessageId>79465</RefToMessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

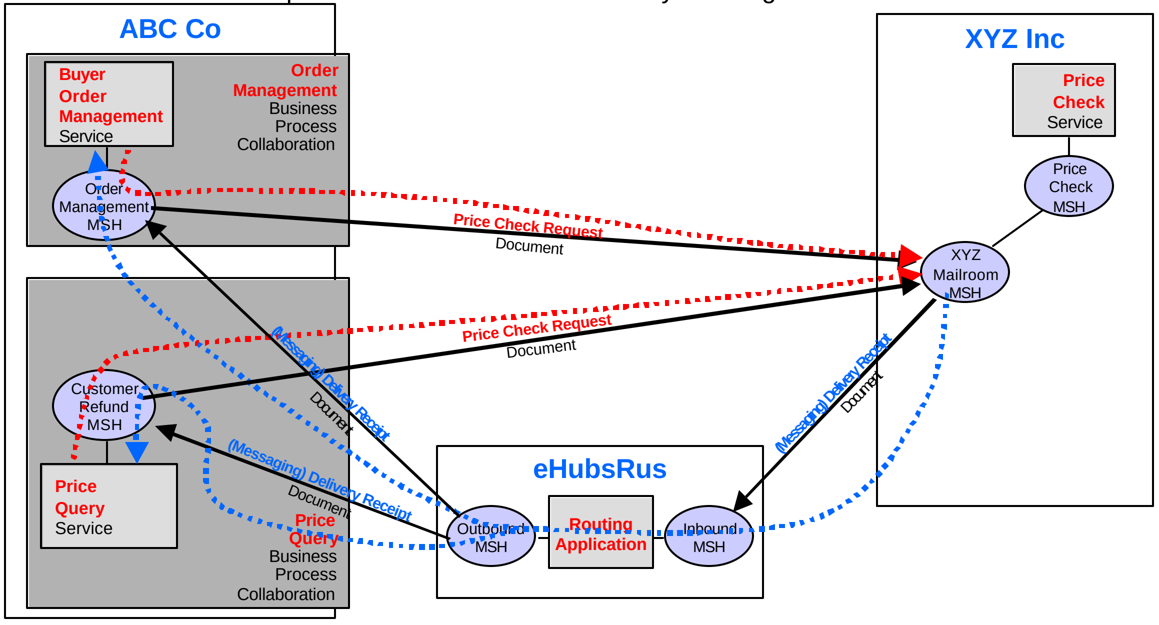
In this example, the From Service is used to identify the Service that needs to be notified and the Action alone identifies that it is a DeliveryReceipt. This would make it easy for the ABC Co Mailroom MSH to work out which Service or Application to notify that the message had been delivered.

This solution should also solve the similar problems for error messages, Message Status Responses and MSH Pings/Pongs.

3 The Return Path using an Intermediary Problem

A variation on the last problem is where the return path involves an intermediary. In this case, ABC Co has two separate MSHs and no mailroom. Each Application can send a message directly to another TP and can receive, via different URLs, the message being returned.

However when XYZ Inc send a reply back they always send it via an intermediary as eHubsRUs keep up-to-date directories of the URL used by services for different companies removing the need for XYZ Inc to keep these lists. This is illustrated by the diagram below.



Following the current rules, the Delivery Receipt Message that eHubsRUs would receive would look something like this ...

```
<MessageHeader>
  <From><PartyId>XYZinc</PartyId></From>
  <To><PartyId>ABCco</PartyId></To>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>uri:www.ebxml.org/messageService/</Service>
  <Action>DeliveryReceipt</Action>
  <MessageData>
    <MessageId>56723</MessageId>
    <RefToMessageId>79465</RefToMessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

The problem is that eHubsRUs cannot determine, from information in the Delivery Receipt, which MSH to send the Delivery Receipt to as:

- It never received the outbound message and therefore cannot retrieve any data associated with the Message Id in the RefToMessageId field
- The destination is only identified by the To Party Id which is ambiguous as, in this instance there are two URLs that the message could be sent to, one for the Buyer Order Management Service and one for the Price Query Service.

This problem can also be solved in the Service contains the real destination service as this can then be used for routing purposes, for example:

- Message 1:

```
<MessageHeader>
  <From><PartyId>ABCco</PartyId></From>
  <FromService>BuyerOrderManagement</FromService>
  <FromAction>FixPriceResponse</FromAction>
  <To><PartyId>XYZinc</PartyId></To>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>PriceCheck</Service>
  <Action>PriceCheckRequest</Action>
  <MessageData>
    <MessageId>79465</MessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

- Message 2:

```
<MessageHeader>
  <From><PartyId>XYZinc</PartyId></From>
  <To><PartyId>ABCco</PartyId></To>
  <FromService>PriceCheck</FromService>
  <FromAction>PriceCheckResponse</FromAction>
  <CPAId>ABC-XYZ-CPA</CPAId>
  <ConversationId>5678</ConversationId
  <Service>BuyerOrderManagement</Service>
  <Action>uri:www.ebxml.org/messageService/DeliverReceipt</Action>
  <MessageData>
    <MessageId>56723</MessageId>
    <RefToMessageId>79465</RefToMessageId>
    ...
  </MessageData>
  ...
</MessageHeader>
```

In this case, both the To Party Id "ABCCo" and the Service "BuyerOrderManagement" can be used for routing.

David Burdett

Solution Strategy, Commerce One
4400 Rosewood Drive, Pleasanton, CA 94588, USA
Tel/VMail: +1 (925) 520 4422; Cell: +1 (925) 216 7704
mailto:david.burdett@commerceone.com ; Web: http://www.commerceone.com