<https://vkbiztalk.com/2018/05/16/biztalk-sequential-convoy/>

BIZTALK SEQUENTIAL CONVOY

[May 16, 2018](https://vkbiztalk.com/2018/05/16/biztalk-sequential-convoy/) [Bala & Vinay](https://vkbiztalk.com/author/luckvinay/) [Orchestration](https://vkbiztalk.com/category/orchestration/) [2 comments](https://vkbiztalk.com/2018/05/16/biztalk-sequential-convoy/#comments)

This blog is to explain about BizTalk Sequential Convoy.

**What is Sequential Convoy?**

Sequential Convoy is one type of BizTalk design pattern to achieve particular type of requirements. This can be defined as “A sequential convoy enables multiple single messages to join together to achieve a required result. A sequential convoy is a set of related messages that have a predefined order. Although the messages do not have to be exactly the same, BizTalk Server must receive them in a sequential order.”

Sequential Convoy also called as singleton pattern convoy.

**Sample Scenario for Sequential Convoy:**

My Orchestration has to receive the messages from source location and create a batch for first three messages and send to destination system.

In order to achieve this requirement, we need to make our orchestration as singleton, that means my orchestration has to receive all 3 messages using single instance (By default, Orchestration creates dedicated instance for each message)

Once my orchestration receives 3 messages, instance will be completed and new instance will be created for next 3 messages. Likewise, my orchestration has to run for ever.

Note: In order to correlate all 3 messages, we need to have common element in all messages (Example: CustomerId/OrderId..)

In real time, usually this type of scenario comes for internet banking transactions or retail orders process.

**Sample Input message:**

This message has to be dropped 3 times in receive location with different data.

<ns0:Billing xmlns:ns0=”http://BTSamples1.BillingAddress”&gt;

<CustId>123</CustId>

<Address1>ABC Apt</Address1>

<Address2>XXX</Address2>

<ZipCode>XXXX</ZipCode>

</ns0:Billing>

**Sample Output message:**

This message is a batch of 3 input messages.

<BillingDetails>

<ns0:Billing xmlns:ns0=”http://BTSamples1.BillingAddress”&gt;

<CustId>123</CustId>

<Address1>ABC Apt</Address1>

<Address2>XXX</Address2>

<ZipCode>XXXX</ZipCode>

</ns0:Billing>

<ns0:Billing xmlns:ns0=”http://BTSamples1.BillingAddress”&gt;

<CustId>123</CustId>

<Address1>BAC Apt</Address1>

<Address2>XXX</Address2>

<ZipCode>XXXX</ZipCode>

</ns0:Billing>

<ns0:Billing xmlns:ns0=”http://BTSamples1.BillingAddress”&gt;

<CustId>123</CustId>

<Address1>CDE Apt</Address1>

<Address2>XXX</Address2>

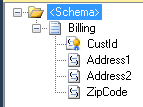
</ns0:Billing>

</BillingDetails>

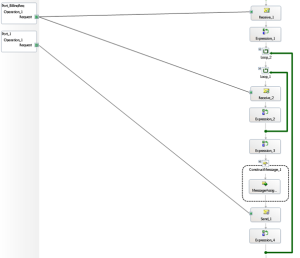
Note: In all messages we need to have a common element “CustId”.

**Implementation:**

1. Need to create a schema for incoming message and promote the common element as property promotion. Also create [correlation type and correlation set](https://vkbiztalk.com/2018/05/14/biztalk-correlations/) on CustId.



2. Design the orchestration like below.



“Receive\_1” shape receives very first message and initialize the correlation set.

“Receive\_2 shape follow the correlation and receives remaining messages using the same instance created while receiving first message.

“Loop\_2” shape make sure the orchestrations run for ever.

“Loop\_1” shape make sure to receive maximum 3 messages for each instance and batch it.

Once max messages (3) are received, “Loop\_1” will break and batched message will be sent to Destination system using “Send\_1” shape.

**3. Create below orchestration variables.**

//Count is a variable of type “System.Int32

//Continue is a variable of type “System.Boolean”

//XmlDocVar is a variable of type “System.Xml.XmlDocument”

//StrFinal is a variable of type “System.String”

//BillingStrVar is a variable of type “System.String”

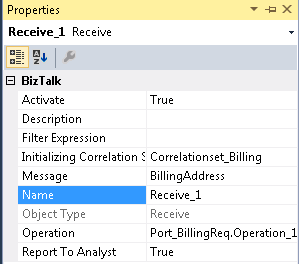
**4. Create below Orchestration Messages**

//XmlDocMsg is a message of type “System.Xml.XmlDocument

//BillingAddress is a message of type BillingAddress Schema

**5. Keep below code in all orchestration shapes.**

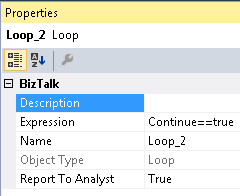
* **Receive\_1**



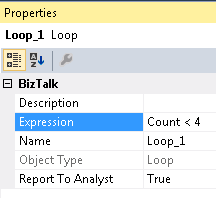
* **Expression\_1**



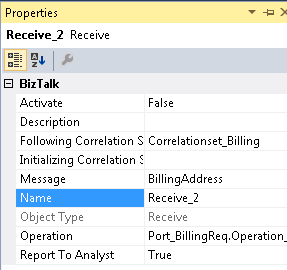
* **Loop\_2**



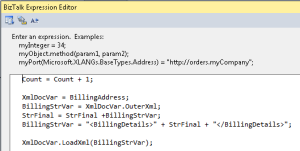
* **Loop\_1**



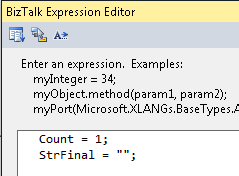
* **Receive\_2:**



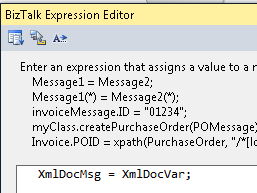
* **Expression\_2**



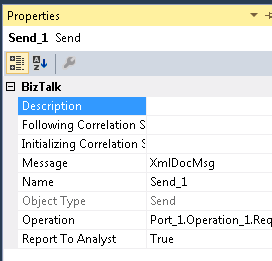
* **Expression\_3**



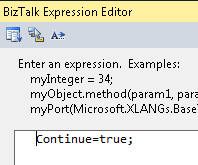
* **ConstructMessage\_1**



* **Send\_1**



* **Expression\_3**



6. Build and deploy the application

7. Configure the ports and start the application

8. **Unit Testing:** Drop couple of sample message copies in input folder with same customer id then for every 3 messages one batch file will be batched and dropped in a destination system. For example, if you drop 6 input messages, you will see 2 output messages in destination folder.