**Scenario 3 : Migrating IIB Broker to ACE in CP4I Container**

[**https://ibm.box.com/s/rl2fs396vzcx5vq258ucf5m6atg7d0uu**](https://ibm.box.com/s/rl2fs396vzcx5vq258ucf5m6atg7d0uu)

**Message flow Project exports** : Messageflow\_PI.zip

**BAR file :** IIBV10BAR.bar

**IIB v10 broker backup :** IIBV10\_Broker\_backup.zip

**Step 1: Take IIB Broker backup from IIB command line**

$ mqsibackupbroker IIB1 -d c:\temp

**Step 2: Run ACE Transformation Advisor from ACE v12 command line**

$TADatacollector ace run c:\temp\IIB1\_220220\_103458.zip



**Step 3: Review the TA report will be generated under the directory mentioned on console after step2.**

For example : C:\Users\AMARSHAH\AppData\Local\Temp\TADataCollector

C:\Users\AMARSHAH\AppData\Local\Temp\TADataCollector\output\IIB1\recommendations.html

Example Report will be something similar to :



**Step 4 : Start preparing to refactor your flows according to TA report.**

**Step 5: Group the flows according to business functionality, scalability requirements**



Modernization articles: <https://community.ibm.com/community/user/integration/blogs/kim-clark1/2021/11/23/iib-ace-series>

**List of flows, Its functionality and Refactor advise**

**REST API Flows :**

* **FundTransferAPI**

**Input Message:**

POST <http://localhost:7080/fundtransfer/v1/moneyTransfer>

{

 "accountInfo" : {

 "sourceAccNo" : "1111",

 "targetAccNo" : "2222",

 "amount" : 5000

 }

}

**Response :**

{"transferStatus":{"TransactionID":"1111\_2222","Status":"Transfer of amount 5000 Successful"}}

*Note: A mix of Http, REST API (Swagger) and MQ . So some refactoring needed on MQNodes. A candidate messageflow for exposing via API-Connect in cp4i.*

* **OrderManagementApp**

**Input:** GET <http://localhost:7080/ordermanagementapp/v1/order?orderid=1111>

**Response:** <Order><Status>Order details retrieved successfully</Status></Order>

**Input:** POST <http://localhost:7080/ordermanagementapp/v1/order?orderid=1111>

**Response:** <Order><Status>New order successfully created</Status></Order>

**Input:** DELETE <http://localhost:7080/ordermanagementapp/v1/order?orderid=1111>

**Response:** <Order><Status>Order deleted or canceled</Status></Order>

Note : No refactoring needed.

**MQ Based Flows :**

* MQ\_Client\_App

**Input Message:** <request>current GMT time</request>

**Reply Message :** <Message>The Curernt GMT timestamp is : 2022-15-21 13:02:50</Message>

Note: Refactoring needed . refer to following articles for details on how to do.

[Creating a queue manager in OpenShift from the Command Line](https://community.ibm.com/community/user/integration/blogs/amar-shah1/2021/12/09/creating-a-queue-manager-in-openshift-from-the-com)

[Moving an App Connect flow using MQ onto containers](https://community.ibm.com/community/user/integration/blogs/amar-shah1/2022/01/05/moving-an-integration-that-uses-ibm-mq-onto-contai)

**TCPIPBased Flows:**

* TCPIPClient
* TCPIPServer

Note : Refactoring needed. Contains configurable service. Needs to be converted to Policy when moving to containers. Expose the ports used by TCP flow Service.

mqsicreateconfigurableservice BROKER -c TCPIPServer -o TCPIPTutorialServerCF -n Port -v 7778

mqsicreateconfigurableservice BROKER -c TCPIPClient -o TCPIPTutorialClientCF -n Port,Hostname,MinimumConnections,MaximumConnections -v 7778,localhost,5,10

InputMessage: To drive TCPIPServer flow.



**File Based flows :**

* FileNodeApp (Refactoring needed. Use FTP/SFTP protocol instead of local filesystem)

Input File :



Output File :



Note : Refactoring needed. Use FTP/SFTP instead of local files.

**Http based flows:**

* HttpRequestApp
* HttpResponseApp

Note: No refactoring needed. But can be deployed in separate IntServer if scaling requirements are different.

*Input message* : http://localhost:7080/requestService

*Reply Message*: <Message><Data>Request ServiceA received response from serviceB</Data></Message>

**Global Cache based flows :**

* GlobalCacheGet
* GlobalCachePut

Note : You cannot really migrate global cache based flows to containers. You need to adopt hybrid topology where catalog and containers reside on-prem and message flows in cp4i containers act only as client.

Article describing this refactor topology is here :

|  |  |
| --- | --- |
|  | [**Hybrid Integration Use Case: How to Refactor Global Cache Topology When Adopting App Connect on IBM Cloud Pak for Integration**](https://community.ibm.com/community/user/integration/blogs/amar-shah1/2021/09/07/how-to-refactor-global-cache-topology-app-connect) |

**Auto Scaling flows :**

* AutoScaleApp

Note: No refactoring needed. Follow the article to setup Horizontal pod autoscaler in the OCP

[Load balancing and autoscaling a simple App Connect flow](https://community.ibm.com/community/user/integration/blogs/amar-shah1/2021/11/24/load-balancing-and-autoscaling-a-simple-app-connec)

**Callable Flows :**

* CallableInvokeFlowApp
* CalledApp

Note: Create Switch Server instance from ACE Operator. Deploy each of these app in separate IntServer and connect via switch server

Input Message: http://localhost:7080/callableinvoke

Reply Message : <message><reply>This is the response from callable Input/Reply flow</reply></message>

**Independent Integration Project:**

* **OrderBookingFlow**

Note: Refactoring needed. Independent Integration Flows are not supported in ACE V11/V12. Convert them to Application using Toolkit. Then deploy to containers.

*Input Message :* <http://localhost:7080/createOrder>

*Reply/Output :* <Order><Status>Order successfully created</Status><Number>111111</Number></Order>