Managed Fault Tolerance and Load Balancing capability in BW 6.x

The assumption is the user reading this document is aware of the basic concepts of BW6.x and EMS. Concepts and configuration on Fault Tolerance in BW 6.x are explained at https://docs.tibco.com/pub/activematrix_businessworks/6.3.1/doc/html/GUID-8361DA81-4AA8-45EA-A98A-1624FEC6BF54.html

Product versions used for the demo:

- BW6.3.1 | EMS 8.2 | Postgres Database | Windows Server 2012

About the demo - Managed Fault Tolerance and Load Balancing.

- we rely on the transport (EMS in this case) to redeliver the message on the specified queue if a message has NOT been confirmed by the BW application.
- **Only Fault Tolerance and NO Load Balancing:**
  - *Single Appnode Activation* - BW application is deployed on 03 appnodes but is ACTIVE on only one appnode (say appnode01), other two appnodes (say appnode02 and appnode03) have the same BW application in PASSIVE mode.
  - When appnode01 goes down without confirming the message, the message is redelivered on the EMS queue and BW application either on appnode02 or appnode03 becomes ACTIVE (automatically) and starts processing the messages from the queue.
  - After the appnode01 crash, BW application only on either appnode02 or appnode03 is ready to process the messages from the EMS queue.
- **Both Fault Tolerance and Load Balancing:**
  - *Multiple Appnodes Activation* - BW application is deployed on 03 appnodes and is ACTIVE on all appnodes (say appnode01, appnode02 and appnode03).
  - When appnode01 goes down without confirming the message, the message is redelivered on the EMS queue and BW application either on appnode02 or appnode03 start processing the messages from the queue.
  - After the appnode01 crash, BW application on both appnodes 02 and 03 are ready to process the messages from the EMS queue.

Details on the project design:

- covers the Managed Fault Tolerance capability in the “group” persistence mode. (no checkpoints!)
- Receiver BW application (JMS Receiver) uses Client Acknowledge Mode and an explicit Confirm activity to confirm the message.
- Message is received on a queue → logs → invokes SOAP (over JMS) service(s) → writes to a file → confirms the message.
- `getCityInfo` SOAP service have a sleep for 30 seconds for the delay.
- Sender BW application sends 05 messages in a 10 second interval on the same queue.

**Deployment Setup:**

- Receiver BW application is deployed within a single domain, single appspace and 03 appnodes.
- Sender application is deployed in a separate domain with single appspace and single appnode.
- Postgres DB is used to configure the engine for a “managed” fault tolerance capability.
**Single Appnode Activation - only Fault Tolerance and NO Load Balancing:**

- **Steps to setup the project on local environment:**
  1. Start the EMS server.
  2. Configure the database for the engine:
     a. run the bundled scripts - create.sql and create-dcp.sql available under BW_HOME/config/dbscripts/engine.
  3. Configure the necessary DB parameters in the config.ini template.
     a. config.ini provided in the MFTDemo.zip folder.
  4. Edit the bat scripts in the singleactivation folder under MFTDemo.zip and provide the necessary variables. (like BW_HOME, project_folder, etcetera)
  5. Execute the bat scripts:
     a. create receiver application by running MFTJMSDemo_SingleAppnodeActivation.bat - this will create a domain, appspace, 03 appnodes, configure the engine to use the config.ini with the DB details from step # 2, deploy and start the receiver BW application.
     b. create sender application by running MFTJMSDemo_JMSSender.bat - this will create a domain, appspace, 01 appnode, deploy and start the sender BW application.

- **Test setup:**

  Ensure both the receiver and sender applications have started properly by verifying the appnode logs.

  1. One of the three appnodes starts processing the messages as shown in the appnode log because as the process starter is only ACTIVE on one appnode. (Single Appnode activation mode)

  2. The appnode sleeps for 30 seconds as configured in the BW application and continues processing the message.
3. This particular appnode is killed now via the Task Manager before it confirmed the messages.

4. The messages are redelivered by EMS and one of the remaining 02 appnodes become ACTIVE now and start processing the messages, note: the “redelivered flag” value. This appnode processes all the 05 messages. (03 redelivered and 02 new messages)

---

Multiple Appnode Activation - both Fault Tolerance and Load Balancing:

- Steps to setup the project on local environment:

  1. Follow steps #1 to #3 from the Single Appnode Activation section setup steps.
  2. Edit the bat scripts in the multipleactivation folder under MFTDemo.zip and provide the necessary variables. (like BW_HOME, project_folder, etcetera)
  3. Execute the bat scripts:

     a. create receiver application by running

```
MFTJMSDemo_MultipleAppnodeActivitation.bat - this will create a domain, appspace, 03 appnodes, configure the engine to use the config.ini with the DB details from step # 2, deploy and start the receiver BW application.
```
b. create sender application by running MFTJMSDemo_JMSSender - this will create a domain, appspace, 01 appnode, deploy and start the sender BW application.

- Test setup:

Ensure both the receiver and sender applications have started properly by verifying the appnode logs.

1. Appnode01 started processing the messages as shown in the appnode log.

```
   1. Received Message Message Body with timestamp 1441:08 PM from Queue.. Going to Sleep now. Redelivered boolean 0
   1. Received Message Message Body with timestamp 1441:30 PM from Queue.. Going to Sleep now. Redelivered boolean 0
   1. Received Message Message Body with timestamp 1441:48 PM from Queue.. Going to Sleep now. Redelivered boolean 0
```

2. Appnode01 sleeps for 30 seconds as configured in the BW application and continues processing the message.

3. Appnode01 is killed now via the Task Manager before it confirmed the messages.

4. The messages are redelivered by EMS and the remaining 02 appnodes start processing the messages, note: the “redelivered flag” value. This appnode processes all the 05 messages. (02 redelivered and 03 new messages)

```
   1. Received Message Message Body with timestamp 1441:08 PM from Queue.. Going to Sleep now. Redelivered boolean 0
   1. Received Message Message Body with timestamp 1441:30 PM from Queue.. Going to Sleep now. Redelivered boolean 0
   1. Received Message Message Body with timestamp 1441:48 PM from Queue.. Going to Sleep now. Redelivered boolean 0
   1. Output of getCityInfo : 61801 Urbana Illinois Urbana, Illinois, United States 40.11 88.207
   1. Illinois Illinois Urbana, Illinois, United States Champaign, Illinois, United States
   1. Message confirmed for Message Body with timestamp 1441:08 PM
   1. Log message after Sleep
   1. Output of getCityInfo : 61801 Urbana Illinois Urbana, Illinois, United States 40.11 88.207
   1. Illinois Illinois Urbana, Illinois, United States Champaign, Illinois, United States
   1. Message confirmed for Message Body with timestamp 1441:08 PM
   1. Log message after Sleep
   1. Output of getCityInfo : 61801 Urbana Illinois Urbana, Illinois, United States 40.11 88.207
   1. Illinois Illinois Urbana, Illinois, United States Champaign, Illinois, United States
   1. Message confirmed for Message Body with timestamp 1441:08 PM
   1. Log message after Sleep
   1. Output of getCityInfo : 61801 Urbana Illinois Urbana, Illinois, United States 40.11 88.207
   1. Illinois Illinois Urbana, Illinois, United States Champaign, Illinois, United States
   1. Message confirmed for Message Body with timestamp 1441:08 PM
   1. Log message after Sleep
```

Notes:

- **“Non-managed” Fault Tolerance and Load Balancing pattern:**
  - The scenario mentioned in Multiple Appnode Activation can also be achieved with engine persistence as “in-memory” (instead of group).
  - This will avoid the need of a DB (steps #2, #3) in the setup but the appnodes within the same appspace will not be aware of each other’s existence.