

# IIB and MQ HA on Pure Applications

IIB is Client connected via  
CCDT

MQ and IIB logs and data on  
an NFS Server in the pattern

Pattern Documentation

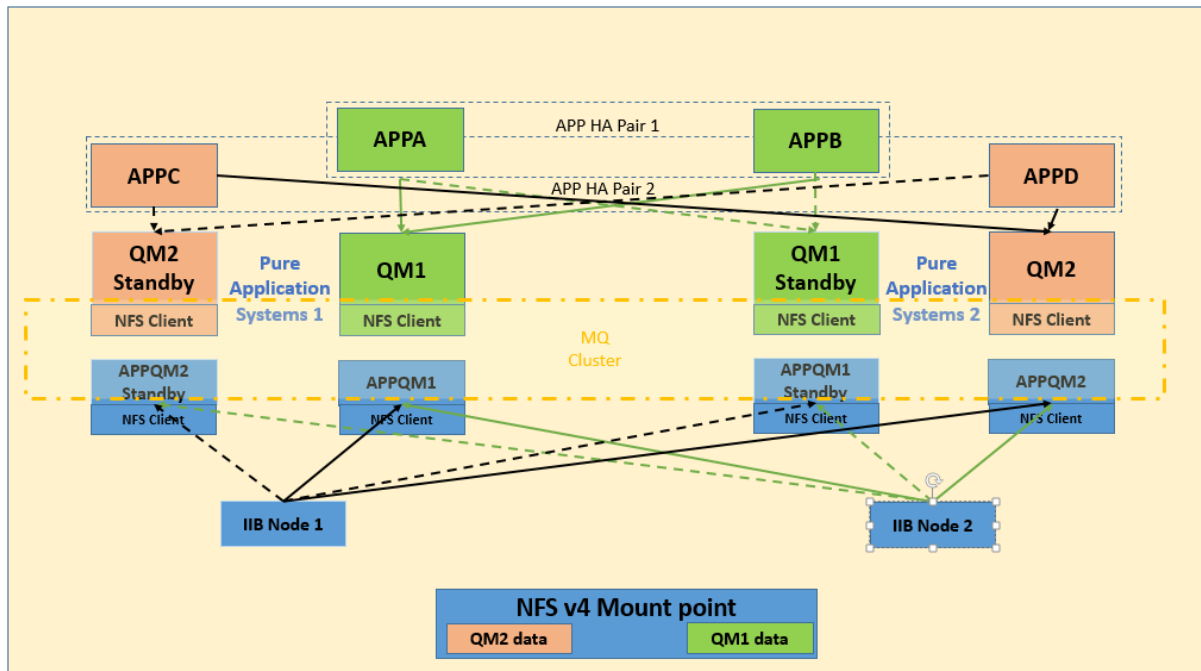
## Table of Contents

Context diagrams .....	4
High level physical view .....	4
High level logical MQ/IIB view .....	4
Pattern Diagram- DA1 - MQ and IIB client HA on NFS new (FULL) v2.4 .....	5
Pattern Definition .....	6
Overview Description.....	6
MQ Cluster .....	6
MQ multi-instance HA pairs.....	6
IBM Integration Bus .....	6
MQ Client Application Gateway nodes.....	6
Pattern Parameters.....	6
Passwords .....	6
Pattern Nodes .....	8
MQ Client Application Gateway nodes.....	8
Description.....	8
MQ Client Application Gateway Parameters .....	8
MQ Gateway nodes .....	8
Description.....	8
MQ Gateway Parameters.....	9
MQ Application nodes .....	9
Description.....	9
MQ Application Parameters.....	10
Pattern Software Parts.....	11
IBM MQ v8.0.0.4 .....	11
IBM Integration Bus V10.0.0.n.....	13
Script package descriptions .....	15
NPP-firewall .....	15
NPP-MQ Cleanup .....	15
NPP-IIBDEPOY-IIBNODE .....	15
NPP-MQCCDT-PAGNODE .....	16
NPP-MQExMQSC-APPQM1-201.....	17
NPP-MQExMQSC-APPQM2-201.....	21
NPP-MQExMQSC-GWQM1-201 .....	21
NPP-MQExMQSC-GWQM2-201 .....	21
OpenfirewallPorts-10.0.0.0.....	21

NPP-IIBMQCCDT-IIBNODE-2.0.2 .....	21
NPP-Create NFS Server-1.1.4 .....	23
NPP-Create NFS client-1.1.6.....	23
NFS v4 Shared file system .....	24
NFS v4 Server node set up .....	24
Script package NPP-Create NFS Server-1.1.4.....	24
NFS client configuration.....	26
Script package NPP-Create NFS client-1.1.6 .....	26
Ordering .....	27
IBM Integration Bus Message Flows .....	28
Message Flow servicing Application Queue Manager 1 .....	28
Message Flow servicing Application Queue Manager 2 .....	28

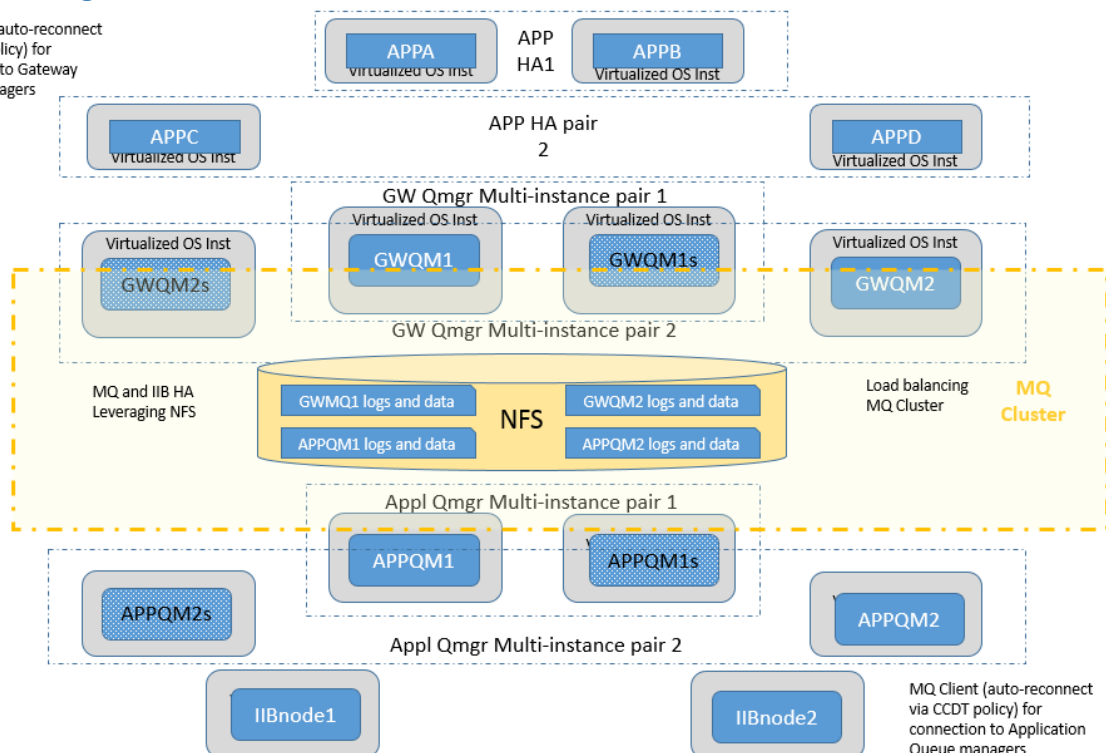
## Context diagrams

### High level physical view

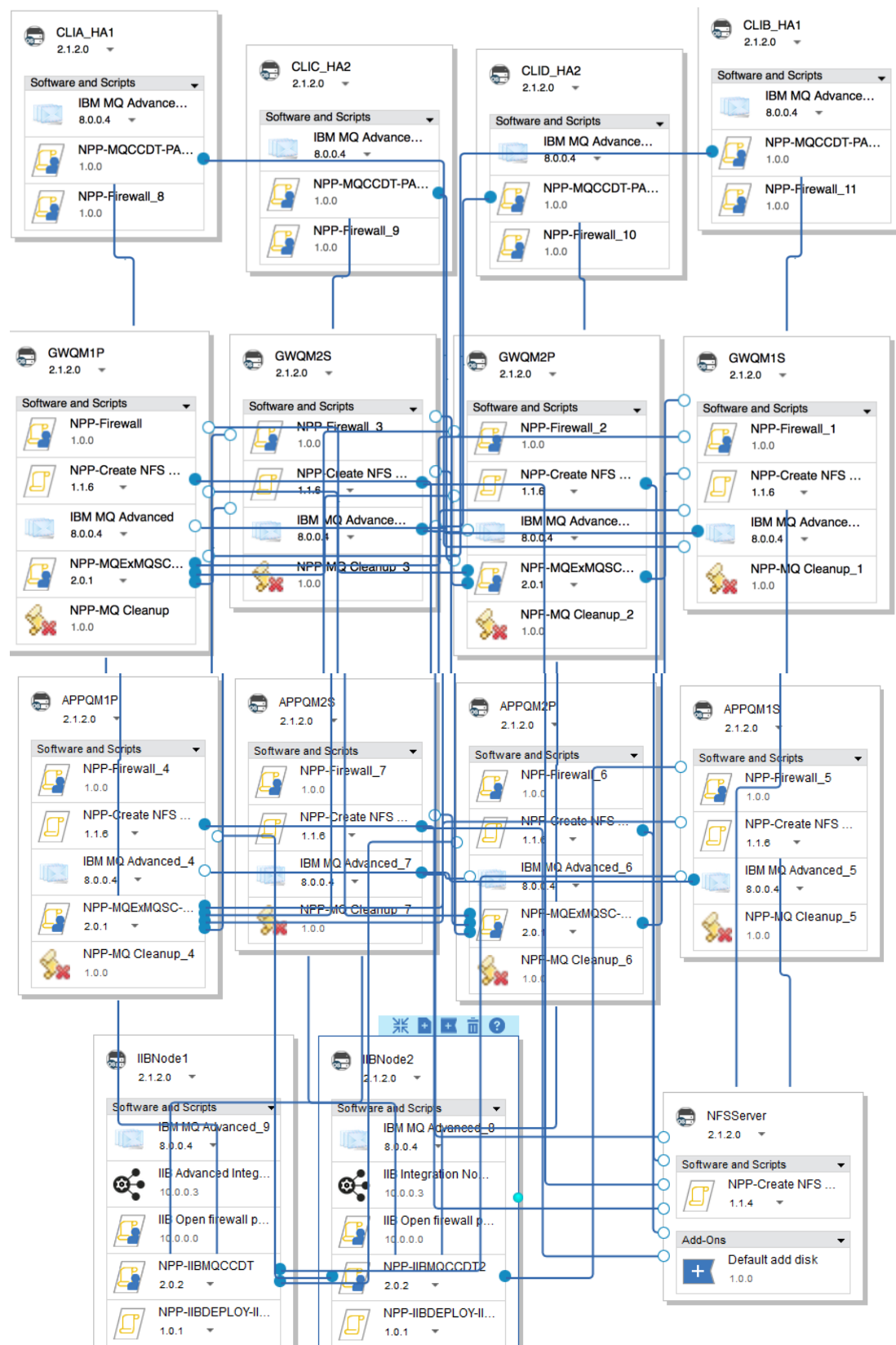


### High level logical MQ/IIB view

MQ Client (auto-reconnect via CCDT policy) for connection to Gateway Queue managers



## Pattern Diagram- DA1 - MQ and IIB client HA on NFS new (FULL) v2.4



## Pattern Definition

### Overview Description

The “DA1 – MQ and IIB Client HA on NFS” v2.4 features

#### MQ Cluster

- 2 nodes wide
  - 2 Gateway queue managers
  - 2 Application queue managers
- Gateway and Application queue managers
- Non-dedicated full cluster repositories on Gateway queue managers
- Gateway to Application queue manager inbound load balancing via RQ (receipt queues)
- Application to Gateway queue manager outbound load balancing via EQ (emission queues)
- Cluster queues defined with DEFBIND(NOTFIXED)

#### MQ multi-instance HA pairs

- Each queue manager has a single Multi-instance standby partner

#### IBM Integration Bus

- 2 IIB Nodes
- Each IIB Node is standalone but client connected to both Application Queue Managers
  - CCDT set up to resolve both Appl Qmgrs and their Standby idle partners
- There are no IIB Multi-instance standby nodes
- “Loop back” message flows are deployed to
  - MQGet from RQs
  - MQPut to EQs

#### MQ Client Application Gateway nodes

- 4 MQ Client nodes
- MQ Installation only no queue managers
- MQ CCDT used to resolve a Gateway queue manager per pair
  - MQ client HA1 pair resolve GWQM1 primary and standby
  - MQ client HA2 pair resolve GWQM2 primary and standby

### Pattern Parameters

#### Passwords

Virtuser = passw0rd

Root = passw0rd

**\* Name**

DA - NPP MQ and IIB client HA on NFS new

**\* Version**

2.4

**Description**

NPP MQ and IIB HA on NFS  
v2.4 leveraging the latest in the  
MQ v8.0.0.4 and IIB releases to  
simplify the implementation. 2 \*

**\* Type**



Pattern



Pattern Template

**Lock option for plug-in usage**



Unlock plug-ins



Lock all plug-ins except Foundation  
plug-ins



Lock all plug-ins

▼ Pattern-level Parameters



Add new parameter

**\* Password (root)**

.....

.....

**\* Password (virtuser)**

.....

.....

## Pattern Nodes

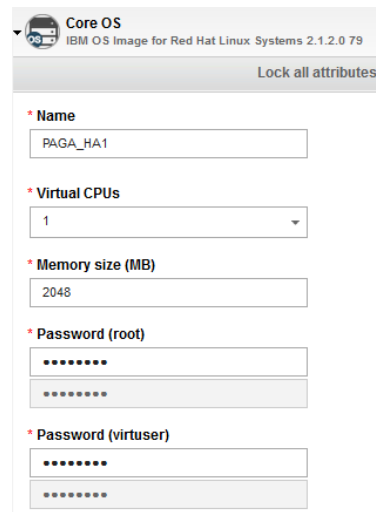
### MQ Client Application Gateway nodes

#### Description

Place holder virtual machine set up to house MQ Client connected applications. In this pattern the MQ Client application will work in HA pairs. Each pair resolving connectivity to a Gateway Queue Manager Multi-instance HA pair via the Client Channel Definition Table (CCDT)

### MQ Client Application Gateway Parameters

#### HA1-A



The screenshot shows the configuration interface for a Core OS virtual machine. At the top, it says 'Core OS' and 'IBM OS Image for Red Hat Linux Systems 2.1.2.0 79'. Below this is a 'Lock all attributes' button. The configuration fields are as follows:

- Name:** A text field containing 'PAGA\_HA1'.
- Virtual CPUs:** A dropdown menu set to '1'.
- Memory size (MB):** A text field containing '2048'.
- Password (root):** Two password fields, both masked with dots.
- Password (virtuser):** Two password fields, both masked with dots.

#### HA1-B

As above for HA1-A

#### HA2-C

As above for HA1-A

#### HA2-D

As above for HA1-A

### MQ Gateway nodes

#### Description

Twin MQ Gateway queue managers each with a multi-instance partner working in an MQ cluster with the MQ Application queue managers. Gateway queue managers host a full cluster repository each.


EQs are local and shared in the cluster.

MQ Security is disabled.



## MQ Gateway Parameters

### *GWQM1P – primary node 1*

**Core OS**  
IBM OS Image for Red Hat Linux Systems 2.1.2.0 79


Lock all attributes

**\* Name**


**\* Virtual CPUs**

**\* Memory size (MB)**

**\* Password (root)**

**\* Password (virtuser)**

### *GWQM1S – standby node 1*

As above for GWQM1P

### *GWQM2P – primary node 2*

As above for GWQM1P

### *GWQM2S – standby node 2*

As above for GWQM1P

## MQ Application nodes

### Description

Twin MQ Application queue managers each with a multi-instance partner working in an MQ cluster with the MQ Gateway queue managers.

Primary Application queue managers are connected to be 2 IIB nodes via CCDT

Standby Application queue managers can be connected to be 2 IIB nodes via CCDT


IIB has “loop back” flows deployed serving RQs and putting to EQs

RQs are local and shared in the MQ cluster.

MQ Security is disabled.

## MQ Application Parameters

*APPQM1P – primary node 1*

 **Core OS**  
IBM OS Image for Red Hat Linux Systems 2.1.2.0 79


Lock all attributes

**\* Name**


**\* Virtual CPUs**

**\* Memory size (MB)**

**\* Password (root)**



**\* Password (virtuser)**



*APPQM1S – standby node 1*

As above for APPQM1P

*APPQM2P – primary node 2*

As above for APPQM1P

*APPQM2S – standby node 2*

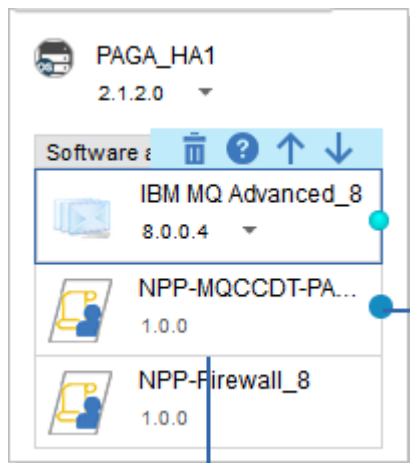
As above for APPQM1P

## Pattern Software Parts

IBM MQ v8.0.0.4

IBM MQ version 8 fix pack 4

### MQ Client Application Gateway Parameters



▼ IBM MQ Advanced\_8

Unlock all attributes

▼ Type of IBM MQ configuration

Installation only

▼ NPP-MQCCDT-PAGNODE

Lock all attributes

TARG\_QMGRNAME

GWQM1

gwqmp

\$(GWQM1P.hostname)

gwqms

\$(GWQM1S.hostname)

▼ NPP-Firewall\_8

Lock all attributes

\* LISTENER\_PORT

1414

## MQ Gateway Primary Parameters

GWQM1P  
2.1.2.0

Software and Scripts

NPP-Firewall  
1.0.0

NPP-Create NFS ...  
1.1.6

IBM MQ Advanced  
8.0.0.4

NPP-MQExMQSC...  
2.0.1

NPP-MQ Cleanup  
1.0.0

Type of IBM MQ configuration  
High availability active instance

\* Shared directory  
/opt/MQShare

\* Queue manager name  
GWQM1

\* Listener port  
1414

Queue manager description  
NPP Gateway QMgr 1

\* Dead letter queue  
SYSTEM.DEAD.LETTER.QUEUE

☐ Queue manager uses linear logging

\* Queue manager log pages  
128

\* Primary logs  
5

\* Secondary logs  
3

\* Error path  
/var/mqm/errors

NPP-Firewall

NPP-Create NFS Client

IBM MQ Advanced

Type of IBM MQ configuration  
High availability active instance

NPP-MQExMQSC-GWQM1

MQSC\_DIRECTORY

\* QMGR\_NAME  
GWQM1

gwm1p  
\${GWQM1P.hostname}

gwm1s  
\${GWQM1S.hostname}

gwm2p  
\${GWQM2P.hostname}

gwm2s  
\${GWQM2S.hostname}

NPP-MQ Cleanup

## MQ Gateway Standby Parameters

GWQM1S  
2.1.2.0

Software and Scripts

NPP-Firewall\_1  
1.0.0

NPP-Create NFS ...  
1.1.6

IBM MQ Advance...  
8.0.0.4

NPP-MQ Cleanup\_1  
1.0.0

Type of IBM MQ configuration  
High availability standby instance

\* Shared directory  
\${IBM MQ Advanced.ha\_standby\_share}

\* Queue manager name  
\${IBM MQ Advanced.ha\_standby\_name}

\* Listener port  
\${IBM MQ Advanced.ha\_standby\_port}

\* Error path  
\${IBM MQ Advanced.ha\_standby\_error}

\* Queue manager directory  
\${IBM MQ Advanced.ha\_standby\_dir}

NPP-Firewall\_1

NPP-Create NFS Client\_1

IBM MQ Advanced\_1

Type of IBM MQ configuration  
High availability standby instance

NPP-MQ Cleanup\_1

## MQ Application Primary Parameters

APPQM1P  
2.1.2.0

Software and Scripts

NPP-Firewall\_4  
1.0.0

NPP-Create NFS ...  
1.1.6

IBM MQ Advance...  
8.0.0.4

NPP-MQExMQSC...  
2.0.1

NPP-MQ Cleanup\_4  
1.0.0

Type of IBM MQ configuration  
High availability active instance

\* Shared directory  
/opt/MQShare

\* Queue manager name  
APPQM1

\* Listener port  
1414

Queue manager description  
NPP Application Qmgr 1

\* Dead letter queue  
SYSTEM.DEAD.LETTER.QUEUE

☐ Queue manager uses linear logging

\* Queue manager log pages  
128

\* Primary logs  
5

\* Secondary logs  
3

\* Error path  
/var/mqm/errors

NPP-Firewall\_4

NPP-Create NFS Client\_4

IBM MQ Advanced\_4

NPP-MQExMQSC-APPQM1

Lock all

\* gwqm2p  
\${GWQM2P.hostname}

\* gwqm2s  
\${GWQM2S.hostname}

\* appqm1p  
\${APPQM1P.hostname}

\* appqm1s  
\${APPQM1S.hostname}

MQSC\_DIRECTORY

\* QMGR\_NAME  
APPQM1

\* gwqm1p  
\${GWQM1P.hostname}

\* gwqm1s  
\${GWQM1S.hostname}

NPP-MQ Cleanup\_4

## MQ Application Standby Parameters

APPQM1S  
2.1.2.0

Software and Scripts

NPP-Firewall\_5  
1.0.0

NPP-Create NFS ...  
1.1.6

IBM MQ Advance...  
8.0.0.4

NPP-MQ Cleanup\_5  
1.0.0

Type of IBM MQ configuration  
High availability standby instance

\* Shared directory  
\${IBM MQ Advanced\_4.ha\_standby\_sh}

\* Queue manager name  
\${IBM MQ Advanced\_4.ha\_standby\_na}

\* Listener port  
\${IBM MQ Advanced\_4.ha\_standby\_po}

\* Error path  
\${IBM MQ Advanced\_4.ha\_standby\_err}

\* Queue manager directory  
\${IBM MQ Advanced\_4.ha\_standby\_dir}

NPP-Firewall\_5

NPP-Create NFS Client\_5

IBM MQ Advanced\_5

U

Type of IBM MQ configuration  
High availability standby instance

NPP-MQ Cleanup\_5

IBM Integration Bus V10.0.0.n

The pattern was built against IBM Integration Bus 10.0.0.3.

The NPP-firewall and IIB Open firewall ports script packages are doing the same job in the IIB/Application nodes. Assuming no additional ports for IIB specifically are required the one or other script package could be removed in these nodes.

The MQ Part is included and a single instance queue manager is created but it is not actually used. It is simply there in case message flows are used that contain “stateful” nodes such as sequence/resequence as these nodes require a local queue manager.

#### *IIB Node 1 Parameters*

**IIBNode1**  
2.1.2.0

**Software and Scripts**

- IBM MQ Advance... 8.0.0.4
- IIB Advanced Inte... 10.0.0.3
- IIB Open firewall p... 10.0.0.0
- NPP-IIBMQCCDT 2.0.2
- NPP-IIBDEPLOY-I... 1.0.1

**IIB Active Integration Node**

Lock all attributes

- \* Queue Manager: APPQM1
- \* Mount directory: /opt/MQShare
- \* Integration Node name: IIBNODE1
- ☐ Start HTTP Listener
- ☐ Enable SSL for HTTPListener
- Thread Pool size for HTTPListener:
- HTTP Address:
- HTTP Port:

**IBM MQ Advanced\_9**

Lock all attributes

- Type of IBM MQ configuration: Single instance
- IIB Advanced Integration Node
- IIB Open firewall ports
- NPP-IIBMQCCDT
- NPP-IIBDEPLOY-IIBNODE
- Lock all attributes
- \* MQSI\_DEPLOY\_INT\_SRV\_NAME: default
- \* IIBNODE\_NAME: IIBNODE1

#### *IIB Node 2 Parameters*

**IIBNode2**  
2.1.2.0

**Software and Scripts**

- IBM MQ Advance... 8.0.0.4
- IIB Integration Nod... 10.0.0.3
- IIB Open firewall p... 10.0.0.0
- NPP-IIBMQCCDT2 2.0.2
- NPP-IIBDEPLOY-I... 1.0.1

**IIB Integration Node Advanced\_1**

Lock all attributes

- \* Integration Node name: IIBNODE2
- Queue Manager:
- ☐ Start HTTP Listener
- ☐ Enable SSL for HTTPListener
- Thread Pool size for HTTPListener:

**IBM MQ Advanced\_8**

Lock all attributes

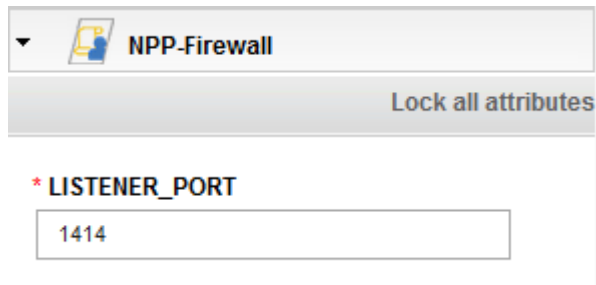
- Type of IBM MQ configuration: Single instance
- IIB Integration Node Advanced\_1
- IIB Open firewall ports2
- NPP-IIBMQCCDT2
- NPP-IIBDEPLOY-IIBNODE2
- Lock all attributes
- \* MQSI\_DEPLOY\_INT\_SRV\_NAME: default
- \* IIBNODE\_NAME: IIBNODE2

## Script package descriptions

### NPP-firewall

Open a MQ Listener port in the firewall.

#### Parameters



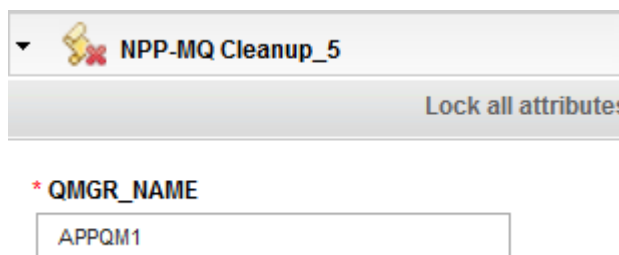
### NPP-MQ Cleanup

This script package was written to run when “tearing down” a pattern instance. The intention is to delete the queue managers and clean up their data log files on shared storage.

When working with the NFS version of the pattern it is not really needed. Given the NFS is provided via a single NFS Server node (virtual machine), when the pattern is stopped and deleted the NFS Server is deleted taking the shared storage with it.

I have retained this script package with this variant of the pattern as it may be of use in cleaning up queue managers at times other than pattern deletion.

#### Parameters



#### Source code

Not updated for the introduction of the pattern instance ID

```
./execute.sh "sudo su - mqm -c \"dspmq\""  
./execute.sh "sudo su - mqm -c \"endmqm -i $QMGR_NAME\""  
./execute.sh "sudo su - mqm -c \"dltmqm -z $QMGR_NAME\""  
./execute.sh "sudo su - mqm -c \"dspmq\""  
echo "Deleting GPFS folder."  
./execute.sh "rm -rf /opt/MQShare/$QMGR_NAME"
```


### NPP-IIBDEPOY-IIBNODE

The BAR file contained in this script package contains a two messages flows. This BAR file is deployed to both IIB Nodes.

- 1) MQGet from the RQ of the application queue manager 1 and MQPut to the EQ of the same application queue manager 1. Connecting by CCDT to application queue manager 1 and it's standby.
- 2) MQGet from the RQ of the application queue manager 2 and MQPut to the EQ of the same application queue manager 2. Connecting by CCDT to application queue manager 2 and it's standby.

Working in this way means that regardless of whether an IIB node is lost both application queue managers will continue to be serviced by the remaining IIB Node. If an application queue manager is lost the CCDT set up will allow the IIB Nodes to reconnect automatically when its standby comes up.

#### Parameters

**NPP-IIBDEPLOY-IIBNODE**

**\* MQSI\_DEPLOY\_INT\_SRV\_NAME**

**\* IIBNODE\_NAME**

#### Source code

```
./execute.sh "mqsideploy $IIBNODE_NAME -e
```

#### NPP-MQCCDT-PAGNODE

For use with MQ Client application wishing to resolve more than one MQ queue manager. In this example we have 2 queue managers that are a multi-instance pair (A primary and a standby). The script package could be extended to be greater than 2 queue managers if required.


The pure app pattern wiring is used to obtain the IP addresses of the nodes hosting the two queue managers. These are then substituted for placeholders in the script package runmqsc input file such that the CCDT is created in the node with real IP addresses. The CCDT is created in the default location with the default name such that the MQ client code picks it up.

The client code checks the MQclient.ini file, MQSERVER environment variable and MQCHLTAB environment variable none of which are set before it goes after a default location CCDT.

I have supplied but not used a script package called NPP-SET-MQENV-1.0.0.1 that can be used to set up environment variables such as MQSERVER or MQCHLTAB





### Parameters

 **NPP-MQCCDT-PAGNODE**

Lock all attributes

**TARG\_QMGRNAME**

**gwqmp**  
 

**gwqms**  
 

### MQSC File

```
DEFINE CHANNEL(TARG_QMGRNAME) +  
    CHLTYPE(CLNTCONN) +  
    TRPTYPE(TCP) +  
    CONNAME('gwqmp(1414),gwqms(1414)') +  
    QMNAME(TARG_QMGRNAME) +  
    REPLACE
```


### Source code

```
echo "modifying ipaddress for gwqm1p"  
sed -i s/gwqm1p/{gwqm1p}/g PAGCLCHL.mqsc  
  
echo "modifying ipaddress for gwqm1s"  
sed -i s/gwqm1s/{gwqm1s}/g PAGCLCHL.mqsc  
  
./execute.sh "runmqsc -n < /tmp/mq/mqsc/PAGCLCHL.MQSC"
```

### NPP-MQExMQSC-APPQM1-201

The IP addresses of connect nodes in the pattern are capture via the pattern wiring and fed into the script package. The placeholders appqm1s, appqm1p,gwqm1s,gwqm1p are replaced with the actual IP addresses in the MQSC file used to configure the queue manager by running the runmqsc command. In this way all MQ channels resolve to the correct IP addresses for all primary and standby MQ nodes in the pattern.


### Parameters


**NPP-MQExMQSC-APPQM1**

Lock all attributes


\* gwqm2p

{GWQM2P.hostname}




\* gwqm2s

{GWQM2S.hostname}




\* appqm1p

{APPQM1P.hostname}



\* appqm1s

{APPQM1S.hostname}




MQSC\_DIRECTORY

\* QMGR\_NAME

APPQM1


\* gwqm1p

{GWQM1P.hostname}



\* gwqm1s

{GWQM1S.hostname}



## MQSC File

ALTER QMGR +

CCSID(850) +

$$\text{CLWLUSEQ}(\text{LOCAL}) +$$

DEADQ('SYSTEM.DEAD.LETTER.QUEUE') +

CHLAUTH(DISABLED) +

FORCE

```
DEFINE QLOCAL('RQ1') +
```

CLUSTER('NPPCLUSTER') +

 $\text{DEFPSIST}(\text{YES}) +$ 

DEFBIND(NOTFIXED) +

DISTL(NO) +

MAXDEPTH(5000) +

REPLACE

DEFINE QLOCAL('SRA') +

CLUSTER('NPPCLUSTER') +

DEFBIND(NOTFIXED) +

DISTL(NO) +

MAXDEPTH(5000) +

REPLACE

DEFINE QLOCAL('SRB') +

CLUSTER('NPPCLUSTER') +

DEFBIND(NOTFIXED) +

DISTL(NO) +

MAXDEPTH(5000) +

REPLACE

DEFINE CHANNEL('TO.APPQM1') +

CHLTYPE(CLUSRCVR) +

CLUSTER('NPPCLUSTER') +

CONNAME('appqm1p(1414),appqm1s(1414)') +

DISCINT(6000) +

MCATYPE(THREAD) +

TRPTYPE(TCP) +

REPLACE

DEFINE CHANNEL('TO.GWQM1') +

CHLTYPE(CLUSSDR) +

CLUSTER('NPPCLUSTER') +

CONNAME('gwqm1p(1414),gwqm1s(1414)') +

DISCINT(6000) +

MCATYPE(THREAD) +

TRPTYPE(TCP) +

REPLACE

```

DEFINE CHANNEL('TO.GWQM2') +
    CHLTYPE(CLUSSDR) +
    CLUSTER('NPPCLUSTER') +
    CONNAME('gwqm2p(1414),gwqm2s(1414)') +
    DISCINT(6000) +
    MCATYPE(THREAD) +
    TRPTYPE(TCP) +
    REPLACE
DEFINE LISTENER('LISTENER.TCP') +
    TRPTYPE(TCP) +
    CONTROL(QMGR) +
    PORT(1414) +
    REPLACE
DEFINE CHANNEL(APPQM1) +
    CHLTYPE(SVRCONN) +
    TRPTYPE(TCP) +
    REPLACE

DEFINE CHANNEL(APPQM1) +
    CHLTYPE(CLNTCONN) +
    TRPTYPE(TCP) +
    CONNAME('appqm1p(1414),appqm1s(1414)') +
    QMNAME(APPQM1) +
    REPLACE
ALTER CHL(APPQM1) CHLTYPE(SVRCONN) MCAUSER('mqm')
ALTER CHL(SYSTEM.DEF.SVRCONN) CHLTYPE(SVRCONN) MCAUSER('mqm')
ALTER QMGR CHLAUTH(DISABLED)
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)
REFRESH SECURITY

```

[Source code](#)

```

echo "modifying ipaddress for gwqmlp"
sed -i s/gwqmlp/$gwqmlp/g APPQM1noSYS.mqsc

```

```

echo "modifiying ipaddress for gwqmls"
sed -i s/gwqmls/$gwqmls/g APPQM1noSYS.mqsc

echo "modifiying ipaddress for gwqm2p"
sed -i s/gwqm2p/$gwqm2p/g APPQM1noSYS.mqsc

echo "modifiying ipaddress for gwqm2s"
sed -i s/gwqm2s/$gwqm2s/g APPQM1noSYS.mqsc

echo "modifiying ipaddress for appqmlp"
sed -i s/appqmlp/$appqmlp/g APPQM1noSYS.mqsc

echo "modifiying ipaddress for appqmls"
sed -i s/appqmls/$appqmls/g APPQM1noSYS.mqsc

./execute.sh "runmqsc $QMGR_NAME < \"$f\""
```

#### NPP-MQExMQSC-APPQM2-201

See NPP-MQExMQSC-APPQM1-201 above. APPQM2 has its own unique .mqsc file

#### NPP-MQExMQSC-GWQM1-201

See NPP-MQExMQSC-APPQM1-201 above. GWQM1 has its own unique .mqsc file

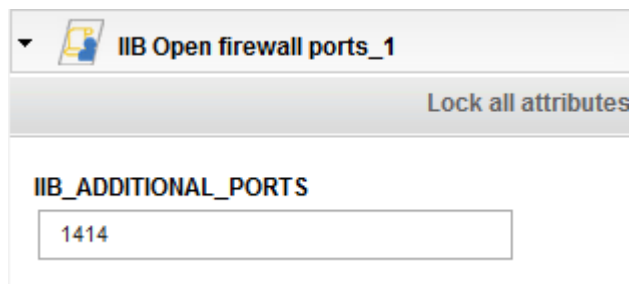
#### NPP-MQExMQSC-GWQM2-201

See NPP-MQExMQSC-APPQM1-201 above. GWQM2 has its own unique .mqsc file

#### OpenfirewallPorts-10.0.0.0

Open any additional fire wall ports. May not be required as NPP-Firewall was used to open 1414 for these IIB nodes. Use it if you have other IIB ports you need to open.

##### Parameters



The screenshot shows a configuration window with a title bar that says "IIB Open firewall ports\_1". Below the title bar is a button labeled "Lock all attributes". Underneath that is a label "IIB\_ADDITIONAL\_PORTS" followed by a text input field containing the number "1414".

#### NPP-IIBMQCCDT-IIBNODE-2.0.2

This script package creates as MQ CCDT file of the default name in the default directory i.e. `/var/mqm/AMQCLCHL.TAB`

It does this by capturing the IP addresses of the application queue managers (and their standby nodes) and modifying the MQSC file in the package to reflect those actual ip address.

It uses the `runmqsc -n` with the MQSC file to create the CCDT.

Next it configures the IIB Node via the mqsichangeproperties command to use that CCDT file. The IIB node is then stopped and restarted.

#### Parameters

NPP-IIBMQCCDT	
Lock all	
<b>appqm2s</b>	
<input type="text" value="\${APPQM2S.hostname}"/>	
<b>appqm1p</b>	
<input type="text" value="\${APPQM1P.hostname}"/>	
<b>appqm1s</b>	
<input type="text" value="\${APPQM1S.hostname}"/>	
<b>TARG_QMGRNAME_1</b>	
<input type="text" value="APPQM1"/>	
<b>TARG_QMGRNAME_2</b>	
<input type="text" value="APPQM2"/>	
<b>* IIBNODE_NAME</b>	
<input type="text" value="IIBNODE1"/>	
<b>appqm2p</b>	
<input type="text" value="\${APPQM2P.hostname}"/>	
<b>* MQSI_PROFILE_DIR</b>	
<input type="text" value="/opt/ibm/iib-10.0.0.3/server/bin/mqsiprofile"/>	

#### MQSC File

```
DEFINE CHANNEL(TARG_QMGRNAME_1) +  
    CHLTYPE(CLNTCONN) +  
    TRPTYPE(TCP) +  
    CONNAME('appqm1p(1414),appqm1s(1414)') +  
    QMNAME(TARG_QMGRNAME_1) +  
    REPLACE  
  
DEFINE CHANNEL(TARG_QMGRNAME_2) +
```

```
CHLTYPE(CLNTCONN) +  
TRPTYPE(TCP) +  
CONNNAME('appqm2p(1414),appqm2s(1414)') +  
QMNAME(TARG_QMGRNAME_2) +  
REPLACE
```

#### *Source code*

```
echo "modifying ipaddress for appqm2p"  
sed -i s/appqm2p/$appqm2p/g IIBCLCHL.MQSC  
  
echo "modifying ipaddress for appqm2s"  
sed -i s/appqm2s/$appqm2s/g IIBCLCHL.MQSC  
  
echo "modifying ipaddress for appqm1p"  
sed -i s/appqm1p/$appqm1p/g IIBCLCHL.MQSC  
  
echo "modifying ipaddress for appqm1s"  
sed -i s/appqm1s/$appqm1s/g IIBCLCHL.MQSC  
  
echo "modifying Qmgr Name for First Qmgr"  
sed -i s/TARG_QMGRNAME_1/$TARG_QMGRNAME_1/g IIBCLCHL.MQSC  
  
echo "modifying Qmgr Name for Second Qmgr"  
sed -i s/TARG_QMGRNAME_2/$TARG_QMGRNAME_2/g IIBCLCHL.MQSC  
  
echo "run the runmqsc command to create the CCDT called AMQCLCHL.TAB  
in /var/mqm directory"  
./execute.sh "runmqsc -n < /tmp/mq/mqsc/IIBCLCHL.MQSC"  
  
echo "execute mqsi profile"  
#./opt/ibm/iib-10.0.0.3/server/bin/mqsiprofile  
. $MQSI_PROFILE_DIR  
  
echo "Configuring CCDT file: AMQCLCHL.TAB for IIB Node:  
$IIBNODE_NAME"  
mqsichangeproperties $IIBNODE_NAME -o BrokerRegistry -n mqCCDT -v  
/var/mqm/AMQCLCHL.TAB  
  
echo "stopping IIB Node: $IIBNODE_NAME"  
mqsistop $IIBNODE_NAME  
  
echo "starting IIB Node: $IIBNODE_NAME"  
mqsistart $IIBNODE_NAME
```

#### NPP-Create NFS Server-1.1.4

See [NFS v4 Server node set up](#) section below

#### NPP-Create NFS client-1.1.6

See [NFS client configuration](#) section below

## NFS v4 Shared file system

The pattern has been tested with in single rack against an NFSv4 Server. The NFS v4 Server is delivered as a single virtual machine node that offers a location /opt/MQShare for NFS v4 Client MQ nodes to place their log and data files. This NFS server is not highly available and as such is a single point of failure. This approach is not recommend to production set ups.

It is however, very convenient for sandpit and demo environments and makes the whole pattern very easy to package and deploy into Pure App System, Software or Service on Softlayer without requiring a shared service or GPFS administrator.

### NFS v4 Server node set up

The screenshot displays the configuration interface for an NFS v4 Server node. On the left, a sidebar shows the node's name 'NFSServer' and version '2.1.2.0'. Below this, there are sections for 'Software and Scripts' (listing 'NPP-Create NFS ...' version 1.1.4) and 'Add-Ons' (listing 'Default add disk' version 1.0.0). The main configuration area is titled 'Core OS' and 'IBM OS Image for Red Hat Linux Systems 2.1.'. It includes a 'Lock all a' button and several input fields: 'Name' (NFSServer), 'Virtual CPUs' (2), 'Memory size (MB)' (4096), 'Password (root)' (a placeholder), and 'Password (virtuser)' (a placeholder). On the right, a 'Default add disk' section contains fields for 'DISK\_SIZE\_GB' (100), 'FILESYSTEM\_TYPE' (ext3), 'MOUNT\_POINT' (/opt/MQShare), 'VOLUME\_GROUP' (empty), and 'OWNER' (empty).


Field	Value
Name	NFSServer
Virtual CPUs	2
Memory size (MB)	4096
Password (root)	\$(pattern.root_pattern_password)
Password (virtuser)	\$(pattern.pattern_virtuserpassword)
DISK_SIZE_GB	100
FILESYSTEM_TYPE	ext3
MOUNT_POINT	/opt/MQShare
VOLUME_GROUP	
OWNER	

### Script package NPP-Create NFS Server-1.1.4

The MQ Part creates the mqm userid and mqm group with UID and GID of 1414. This needs to tie up between the NFS clients and servers.



## Parameters

 **NPP-Create NFS Server**

**\* NFS\_CLIENT\_HOSTS**

**\* NFS\_SERVER\_DIR**

**\* NFS\_CLIENT\_USER**

**\* NFS\_CLIENT\_UID**

**\* NFS\_CLIENT\_GROUP**

**\* NFS\_CLIENT\_GID**

## Source code

```
groupadd ${NFS_CLIENT_GROUP} -g ${NFS_CLIENT_GID}
useradd -u ${NFS_CLIENT_UID} -g ${NFS_CLIENT_USER} -m -d /home/${NFS_CLIENT_USER}
${NFS_CLIENT_GROUP}
```

```
mkdir ${NFS_SERVER_DIR}
chown -R ${NFS_CLIENT_USER}:${NFS_CLIENT_GROUP} ${NFS_SERVER_DIR}
chmod -R 777 ${NFS_SERVER_DIR}
```

```
echo "${NFS_SERVER_DIR}
*(rw,sync,no_root_squash,anonuid=${NFS_CLIENT_UID},anongid=${NFS_CLIENT_GID})" >>
/etc/exports
```

```
echo "RQUOTAD_PORT=875" >> /etc/sysconfig/nfs
echo "LOCKD_TCPPORT=32803" >> /etc/sysconfig/nfs
echo "LOCKD_UDPSPORT=32769" >> /etc/sysconfig/nfs
echo "MOUNTD_PORT=892" >> /etc/sysconfig/nfs
echo "STATD_PORT=662" >> /etc/sysconfig/nfs
```

```
service iptables start
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 111 -j ACCEPT
iptables -A INPUT -m state --state NEW -m udp -p udp --dport 111 -j ACCEPT
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 32803 -j ACCEPT
iptables -A INPUT -m state --state NEW -m udp -p udp --dport 32769 -j ACCEPT
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 892 -j ACCEPT
iptables -A INPUT -m state --state NEW -m udp -p udp --dport 892 -j ACCEPT
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 875 -j ACCEPT
iptables -A INPUT -m state --state NEW -m udp -p udp --dport 875 -j ACCEPT
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 662 -j ACCEPT
iptables -A INPUT -m state --state NEW -m udp -p udp --dport 662 -j ACCEPT
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 2049 -j ACCEPT
iptables -A INPUT -m state --state NEW -m udp -p udp --dport 2049 -j ACCEPT
```

```
chkconfig iptables on
service iptables save
service iptables stop
# service iptables start
# service iptables status
```

```
echo 'enabling nfs and rpcbind'
chkconfig rpcbind on
chkconfig --list rpcbind
chkconfig nfs on
chkconfig --list nfs

echo 'starting nfs and rpcbind services'
service rpcbind start
service rpcbind status
service nfs start
service nfs status


exportfs
```

## NFS client configuration

### Script package NPP-Create NFS client-1.1.6

Note the UID and GID for mqm user and mqm group as created by the MQ Part in nodes that will leverage the NFS client is 1414. The script package will synchronize this with the NFS server such that the queue managers can write to the NFS server /opt/MQShare directory and files.

#### Parameters

 **NPP-Create NFS Client\_4**

**\* NFS\_SERVER\_HOST**

**\* NFS\_SERVER\_DIR**

**\* NFS\_CLIENT\_DIR**

**\* NFS\_CLIENT\_USER**

**\* NFS\_CLIENT\_UID**

**\* NFS\_CLIENT\_GROUP**

**\* NFS\_CLIENT\_GID**

#### Source code

See createNFSclient.sh in the NPP-Create NFS Client-1.1.6 script package.

## Ordering

Ordering is important such that IP addresses etc can be resolved when script packages to configure MQ channels are run. Refer to the ordering tab in the IBM Pattern builder for the pattern.



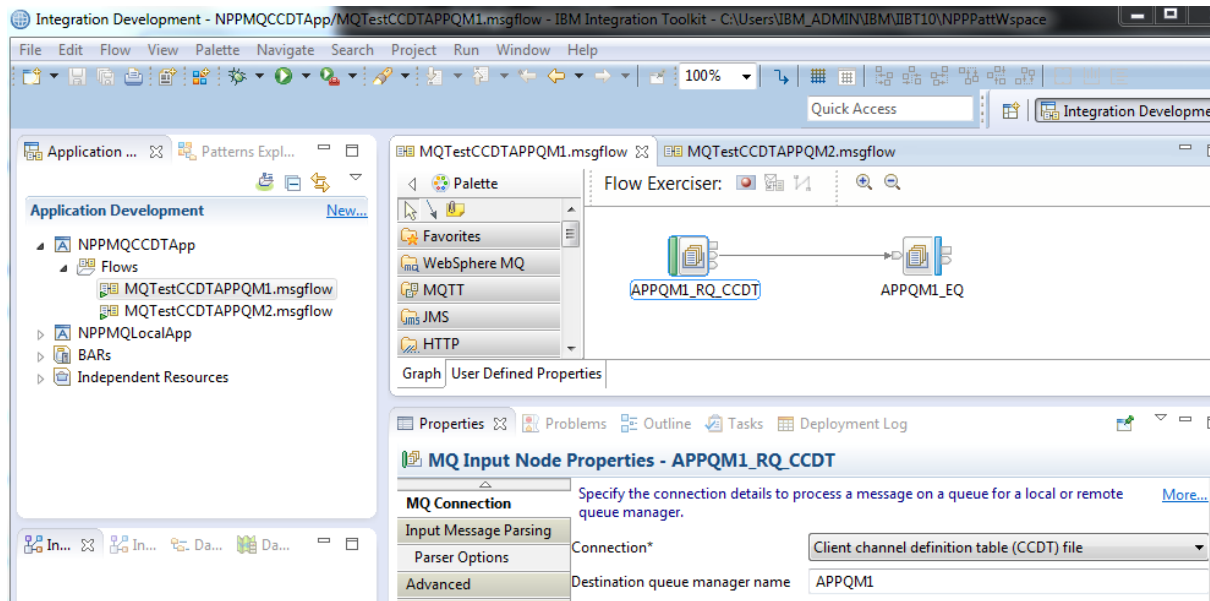
## IBM Integration Bus Message Flows

There is a Project Interchange supplied in the materials called IIBCCDAppProjInt.zip

The IIB Application contains 2 message flows. One for servicing Application queue manager 1 and its standby partner (in the case of a failover) and a second for serving Application queue manager 2 and its standby partner (in the case of a failover).

The message flows are simple “loop back” flows.

### Message Flow servicing Application Queue Manager 1



### Message Flow servicing Application Queue Manager 2

