IIB and MQ HA on Pure Applications

IIB is Client connected via CCDT

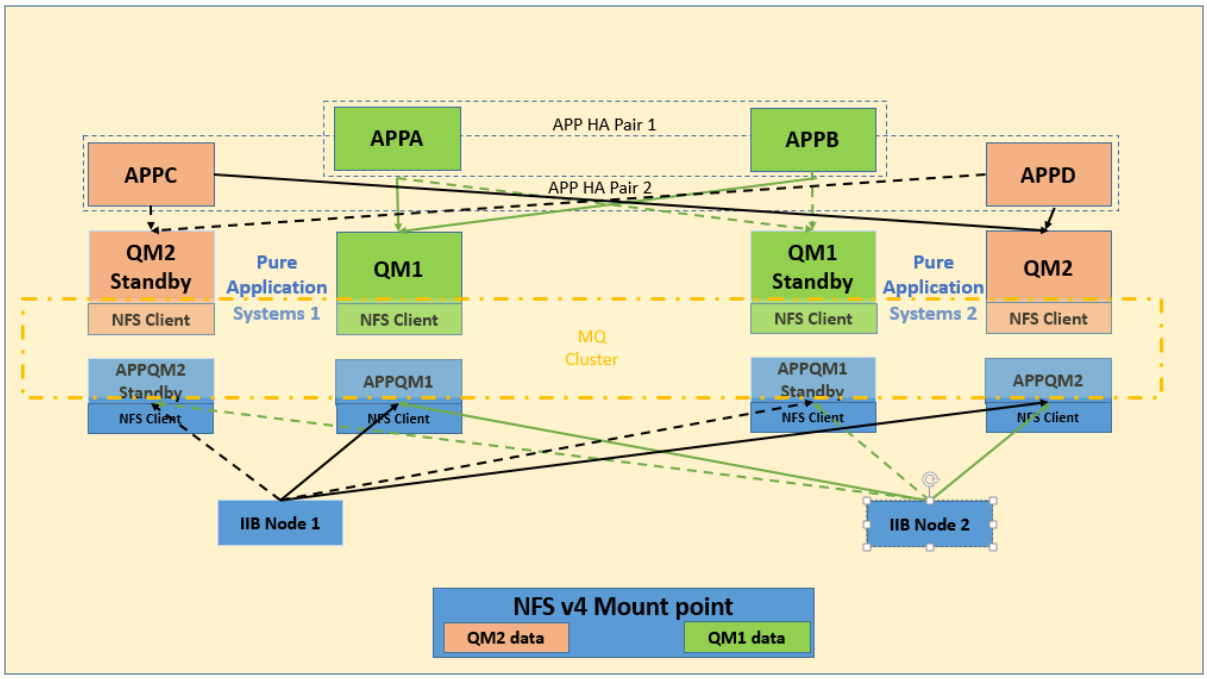
MQ and IIB logs and data on an NFS Server in the pattern

Pattern Documentation

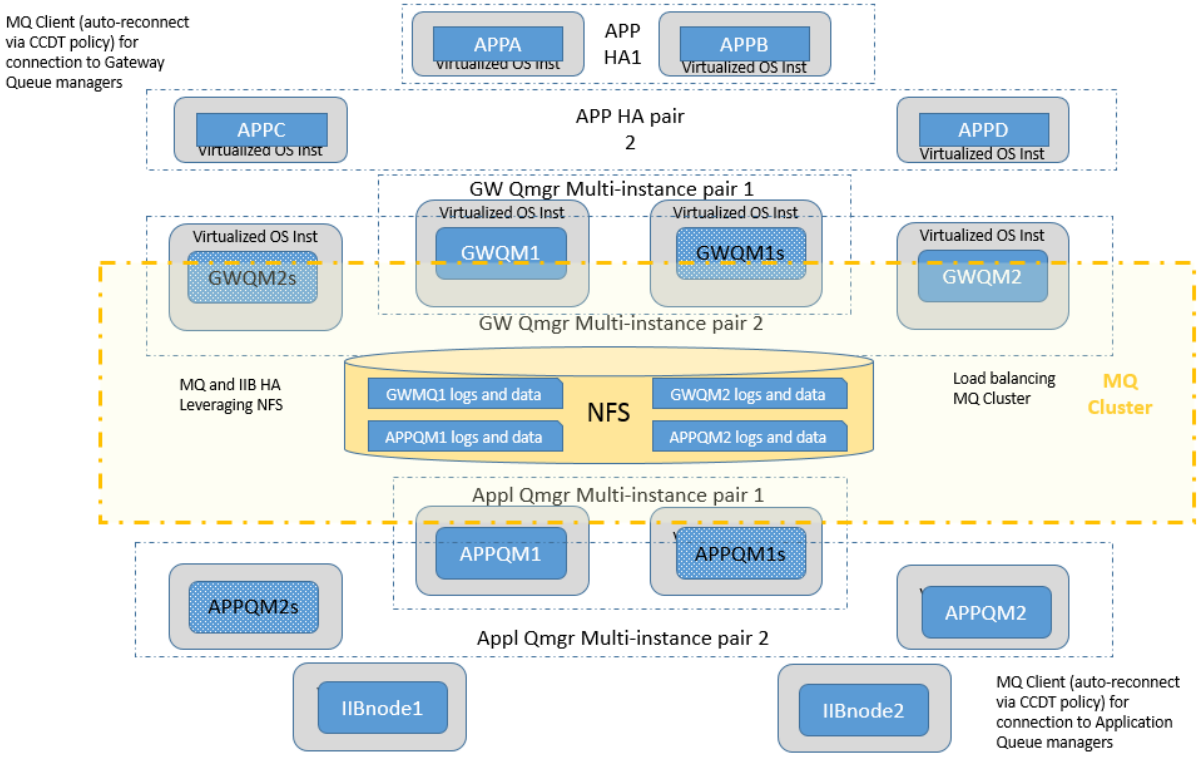
Table of Contents

Context diagrams

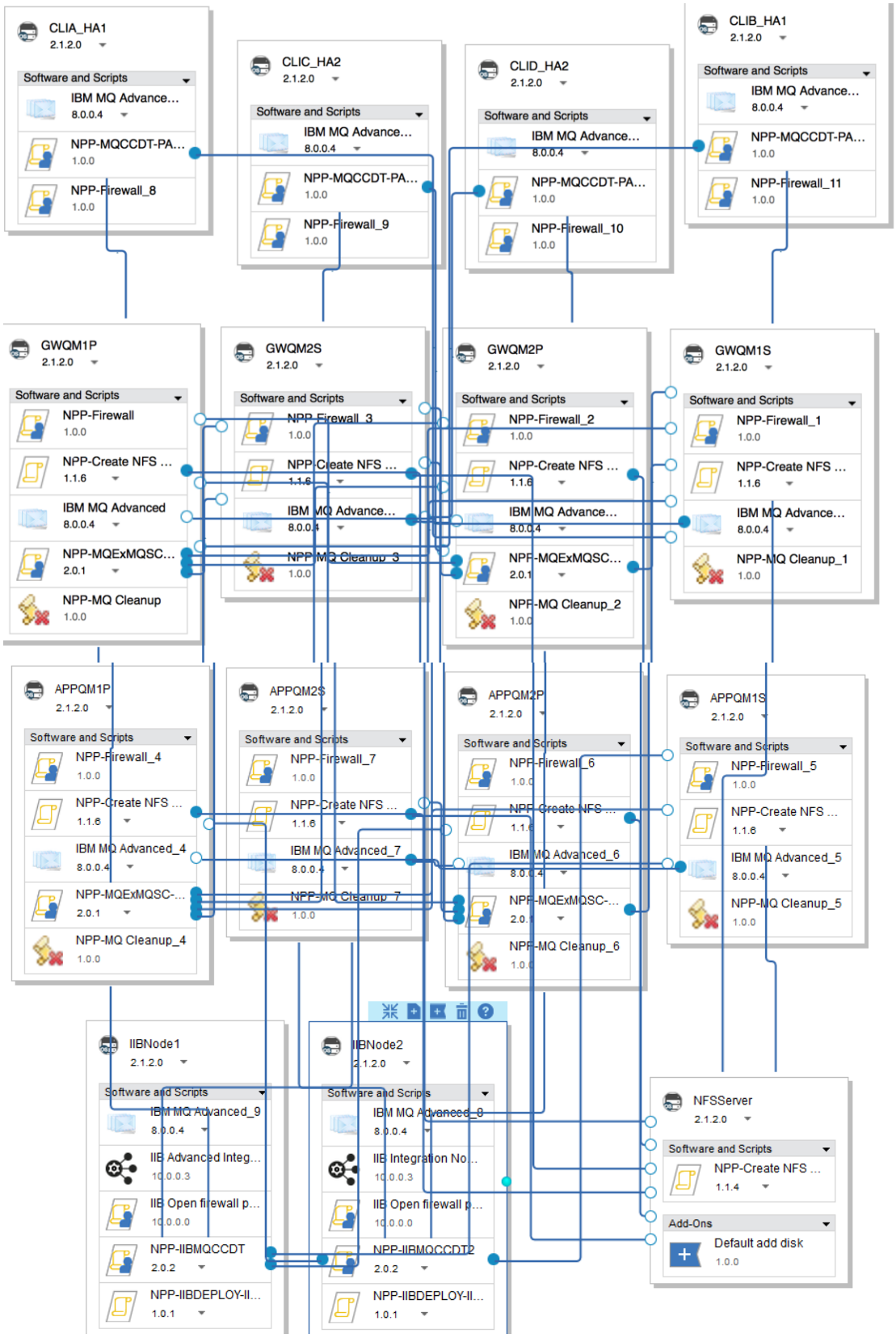
High level physical view



High level logical MQ/IIB view



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

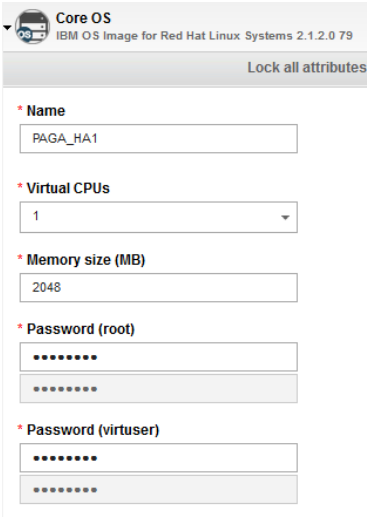
MQ Client Application Gateway nodes

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



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

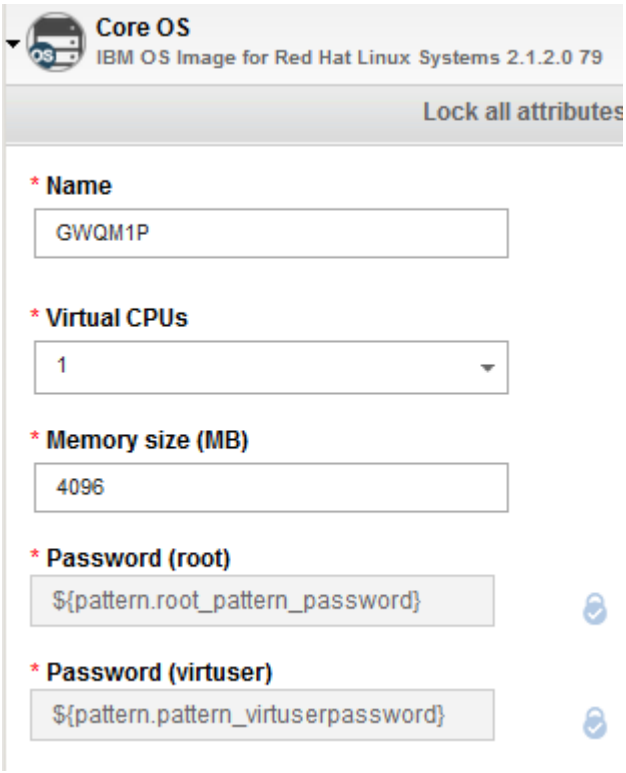
Twin MQ Gateway queue managers each with a multi-instance partner working in an MQ cluster with the MQ Application queue managers. Gateway queue managers host a full cluster repository each.

EQs are local and shared in the cluster.

MQ Security is disabled.

MQ Gateway Parameters

GWQM1P – primary node 1



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

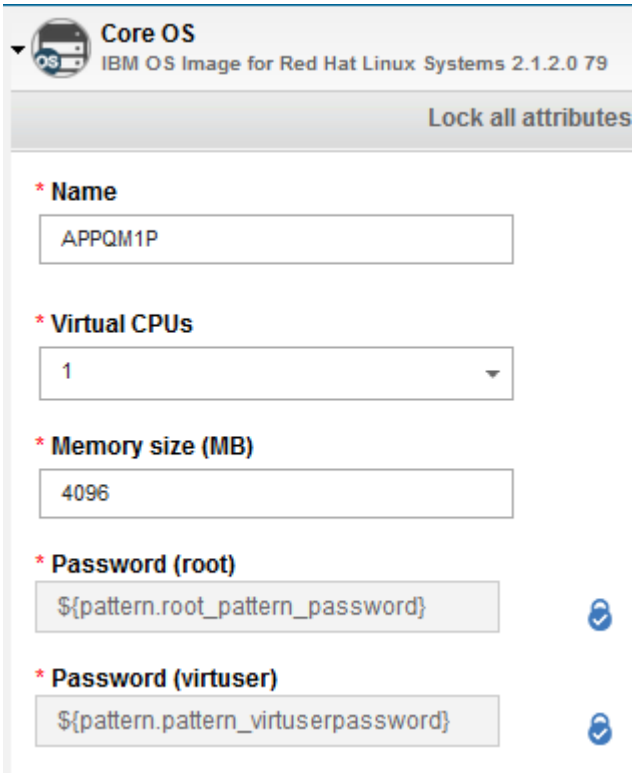
IIB has “loop back” flows deployed serving RQs and putting to EQs

RQs are local and shared in the MQ cluster.

MQ Security is disabled.

MQ Application Parameters

APPQM1P – primary node 1



APPQM1S – standby node 1

As above for APPQM1P

APPQM2P – primary node 2

As above for APPQM1P

APPQM2S – standby node 2

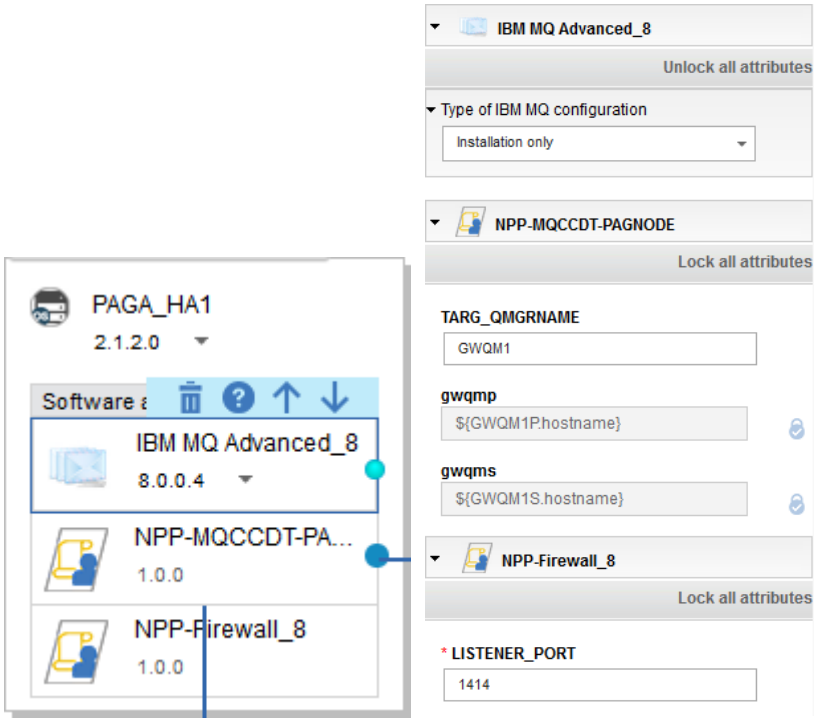
As above for APPQM1P

Pattern Software Parts

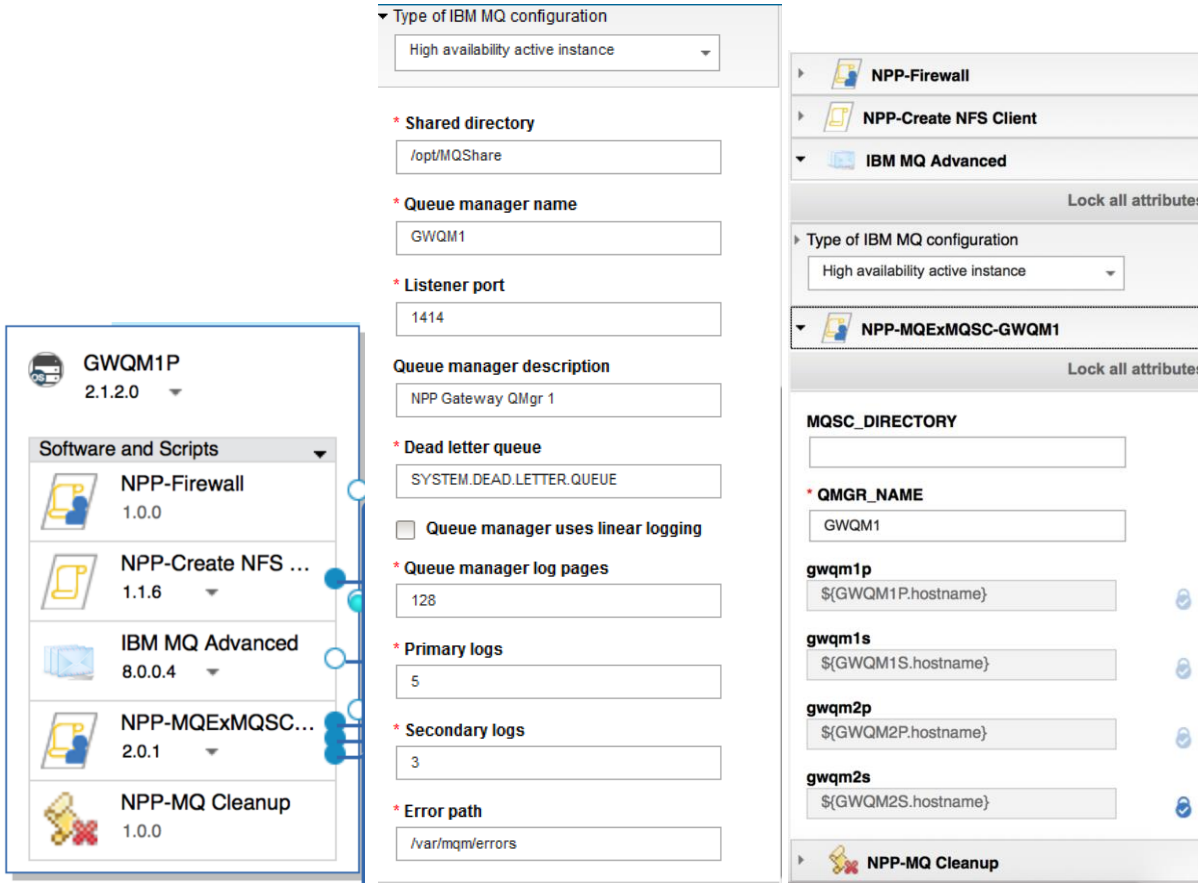
IBM MQ v8.0.0.4

IBM MQ version 8 fix pack 4

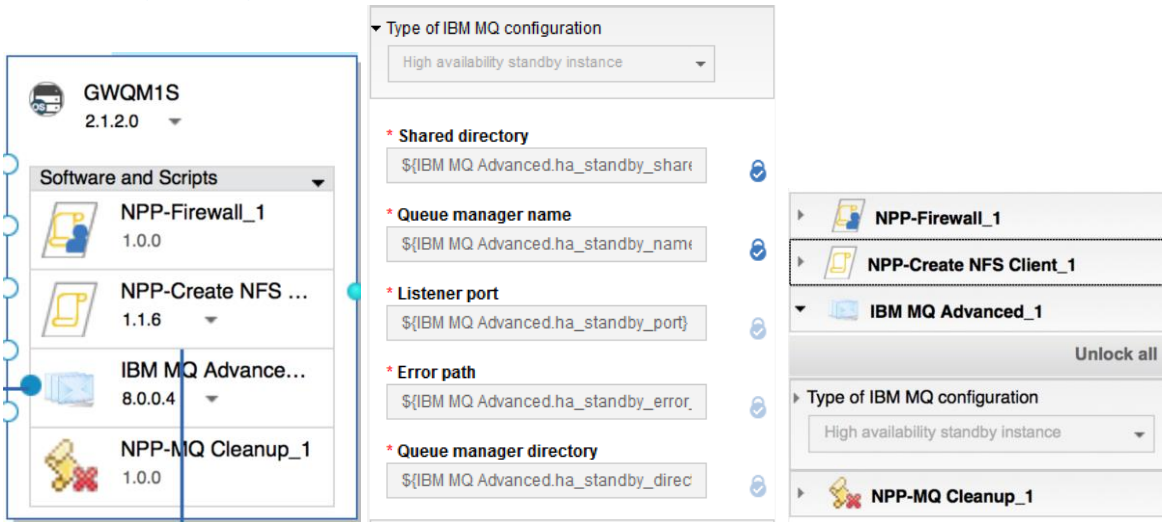
MQ Client Application Gateway Parameters



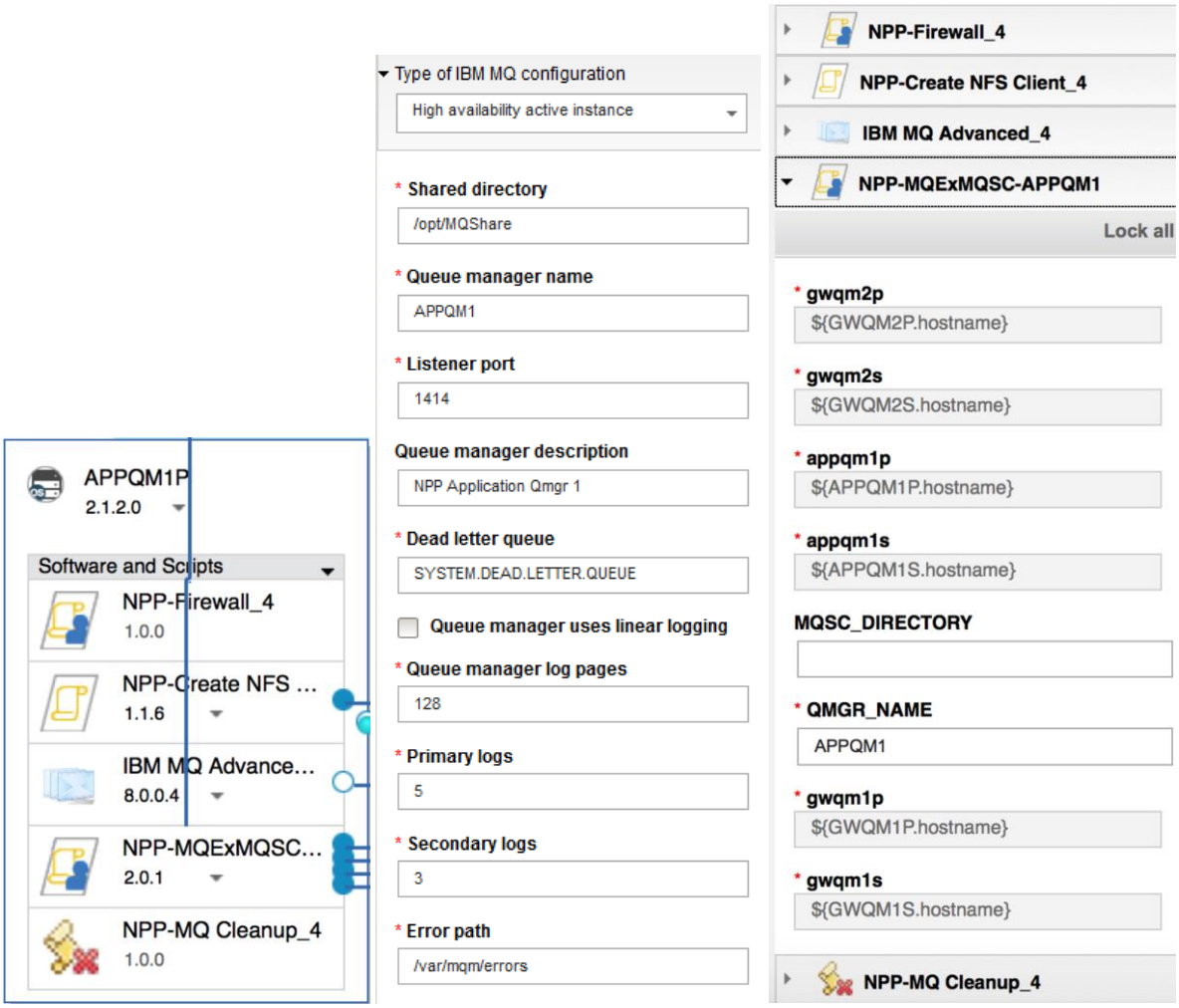
MQ Gateway Primary Parameters



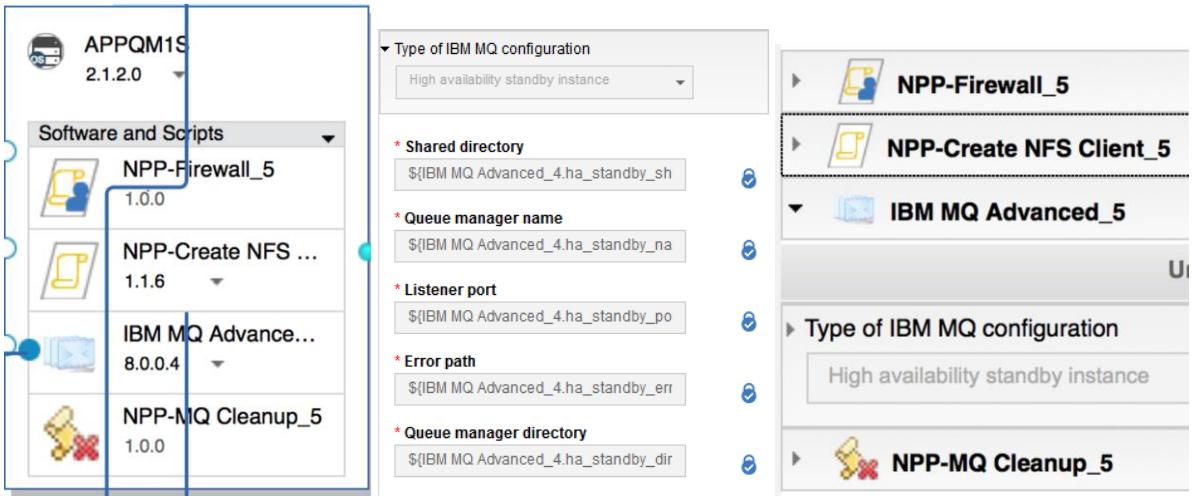
MQ Gateway Standby Parameters



MQ Application Primary Parameters



MQ Application Standby Parameters



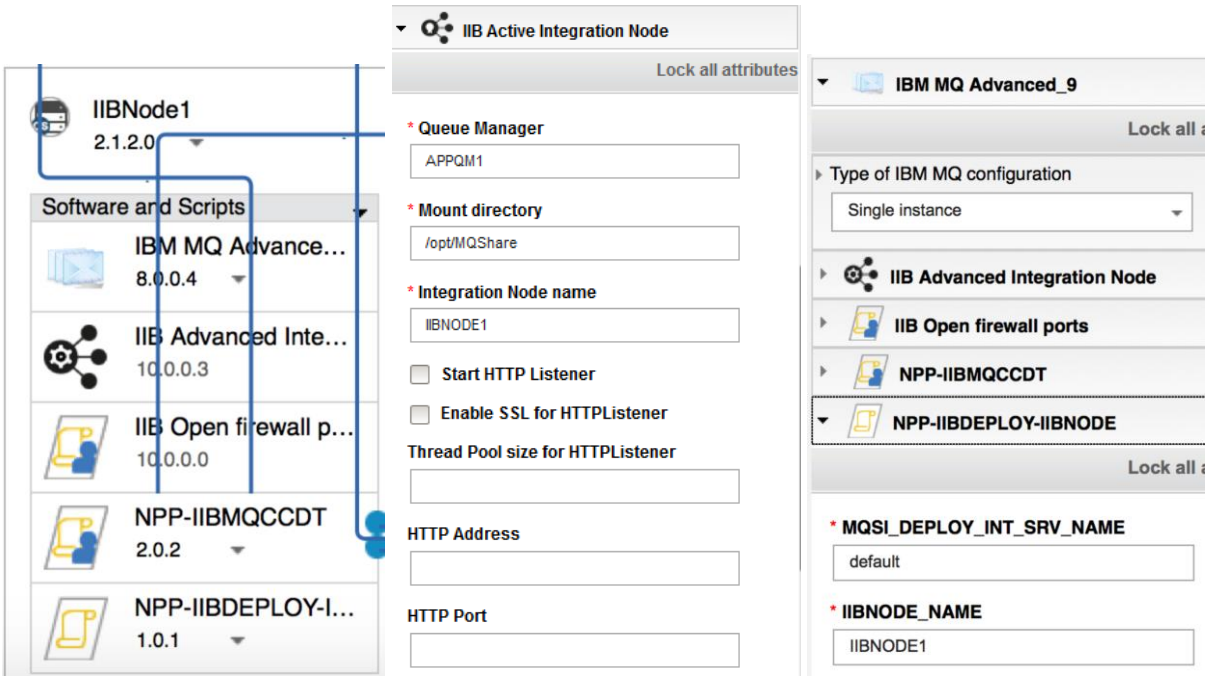
IBM Integration Bus V10.0.0.n

The pattern was built against IBM Integration Bus 10.0.0.3.

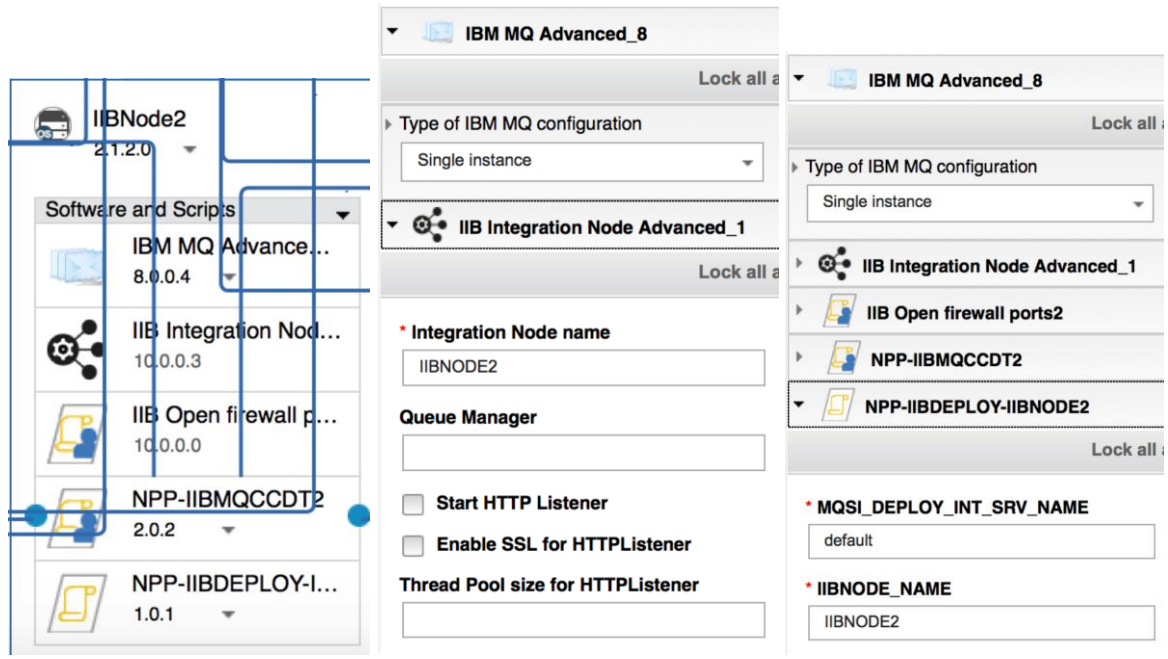
The NPP-firewall and IIB Open firewall ports script packages are doing the same job in the IIB/Application nodes. Assuming no additional ports for IIB specifically are required the one or other script package could be removed in these nodes.

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



IIB Node 2 Parameters

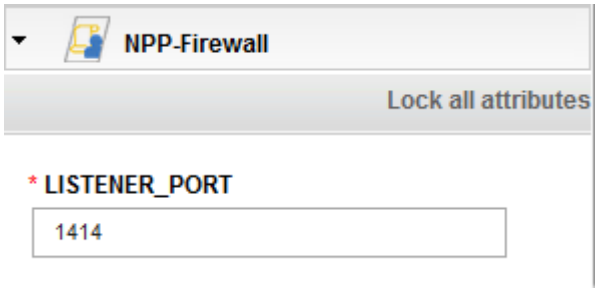


Script package descriptions

NPP-firewall

Open a MQ Listener port in the firewall.

Parameters



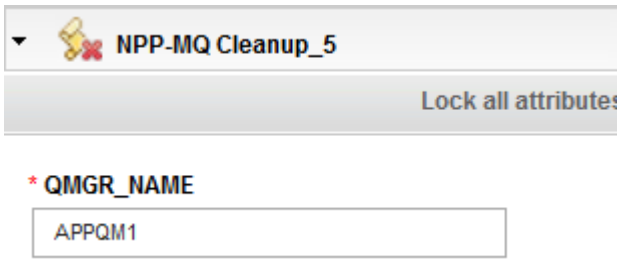
NPP-MQ Cleanup

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"

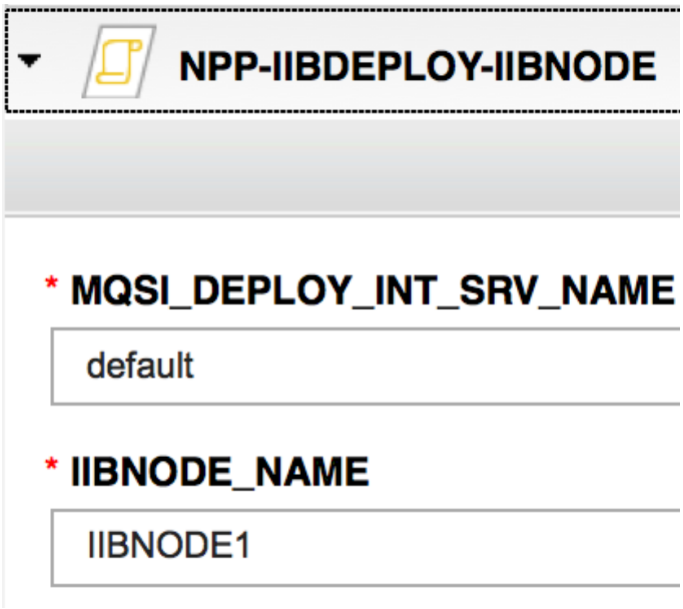
NPP-IIBDEPOY-IIBNODE

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 II 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



Source code

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

NPP-MQCCDT-PAGNODE

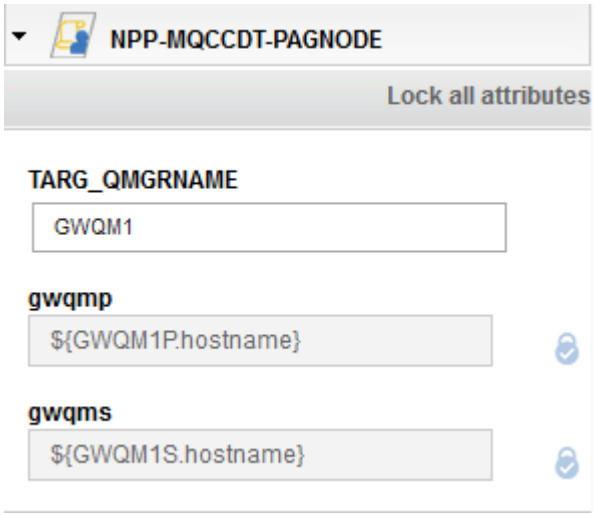
For use with MQ Client application wishing to resolve more than one MQ queue manager. In this example we have 2 queue managers that are a multi-instance pair (A primary and a standby). The script package could be extended to be greater than 2 queue managers if required.

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



MQSC File

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

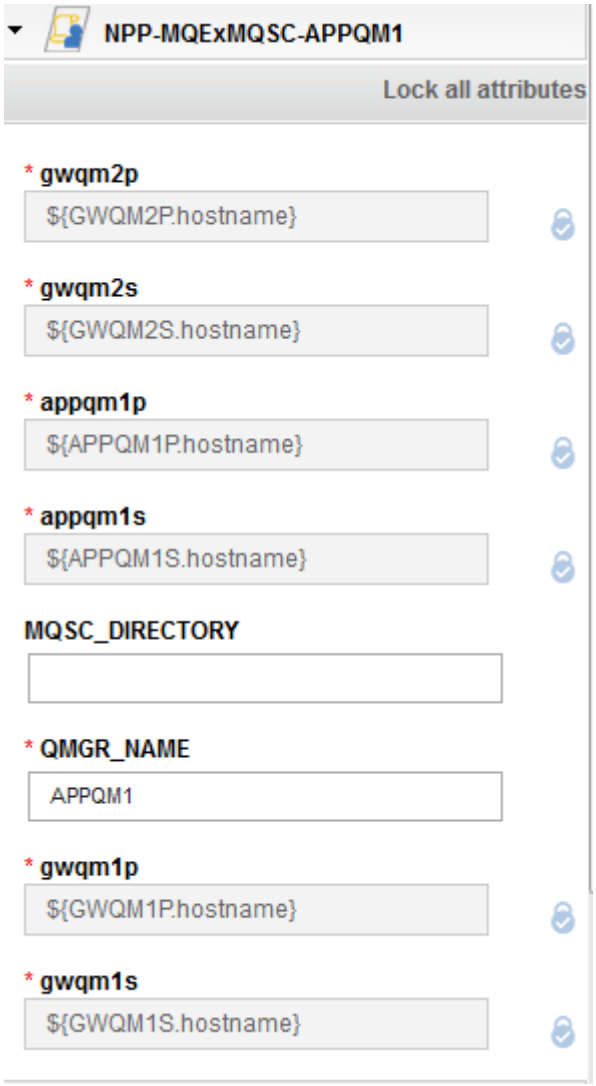
Source code

```
echo "modifiying ipaddress for gwqm1p"
sed -i s/gwqm1p/$gwqm1p/g PAGCLCHL.mqsc

echo "modifiying 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



MQSC File

```
ALTER QMGR +
  CCSID(850) +
  CLWLUSEQ(LOCAL) +
  DEADQ('SYSTEM.DEAD.LETTER.QUEUE') +
  CHLAUTH(DISABLED) +
  FORCE

DEFINE QLOCAL('RQ1') +
 CLUSTER('NPPCLUSTER') +
 DEFPSIST(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 "modifiying ipaddress for gwqm1p"
sed -i s/gwqm1p/$gwqm1p/g APPQM1noSYS.mqsc
```

```
echo "modifiying ipaddress for gwqm1s"
sed -i s/gwqm1s/$gwqm1s/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 appqm1p"
sed -i s/appqm1p/$appqm1p/g APPQM1noSYS.mqsc

echo "modifiying ipaddress for appqm1s"
sed -i s/appqm1s/$appqm1s/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

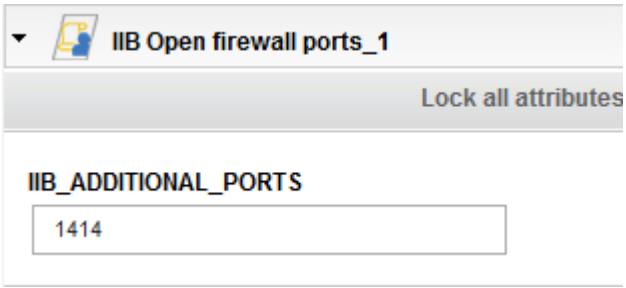
NPP-MQExMQSC-GWQM2-201

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



NPP-IIBMQCCDT-IIBNODE-2.0.2

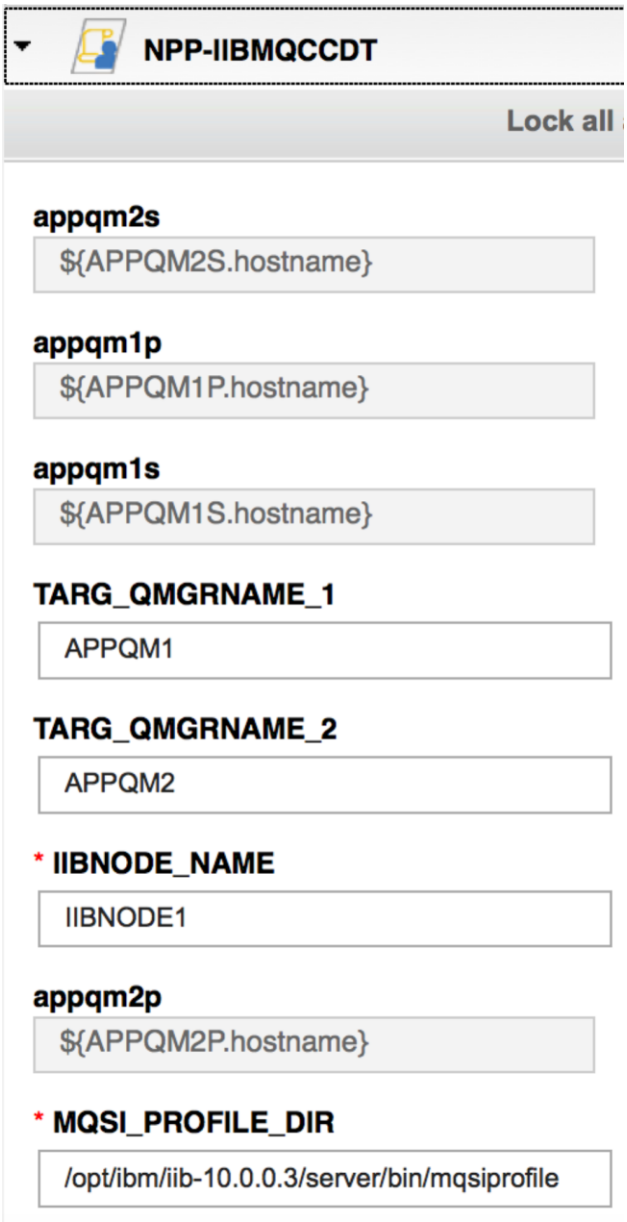
This script package creates as MQ CCDT file of the default name in the default directory i.e. `/var/mqm/AMQCLCHL.TAB`

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



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) +

CONNAME('appqm2p(1414),appqm2s(1414)') +

QMNAME(TARG_QMGRNAME_2) +

REPLACE

Source code

```
echo "modifiying ipaddress for appqm2p"
sed -i s/appqm2p/$appqm2p/g IIBCLCHL.MQSC

echo "modifiying ipaddress for appqm2s"
sed -i s/appqm2s/$appqm2s/g IIBCLCHL.MQSC

echo "modifiying ipaddress for appqm1p"
sed -i s/appqm1p/$appqm1p/g IIBCLCHL.MQSC

echo "modifiying ipaddress for appqm1s"
sed -i s/appqm1s/$appqm1s/g IIBCLCHL.MQSC

echo "modifiying Qmgr Name for First Qmgr"
sed -i s/TARG_QMGRNAME_1/$TARG_QMGRNAME_1/g IIBCLCHL.MQSC

echo "modifiying 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

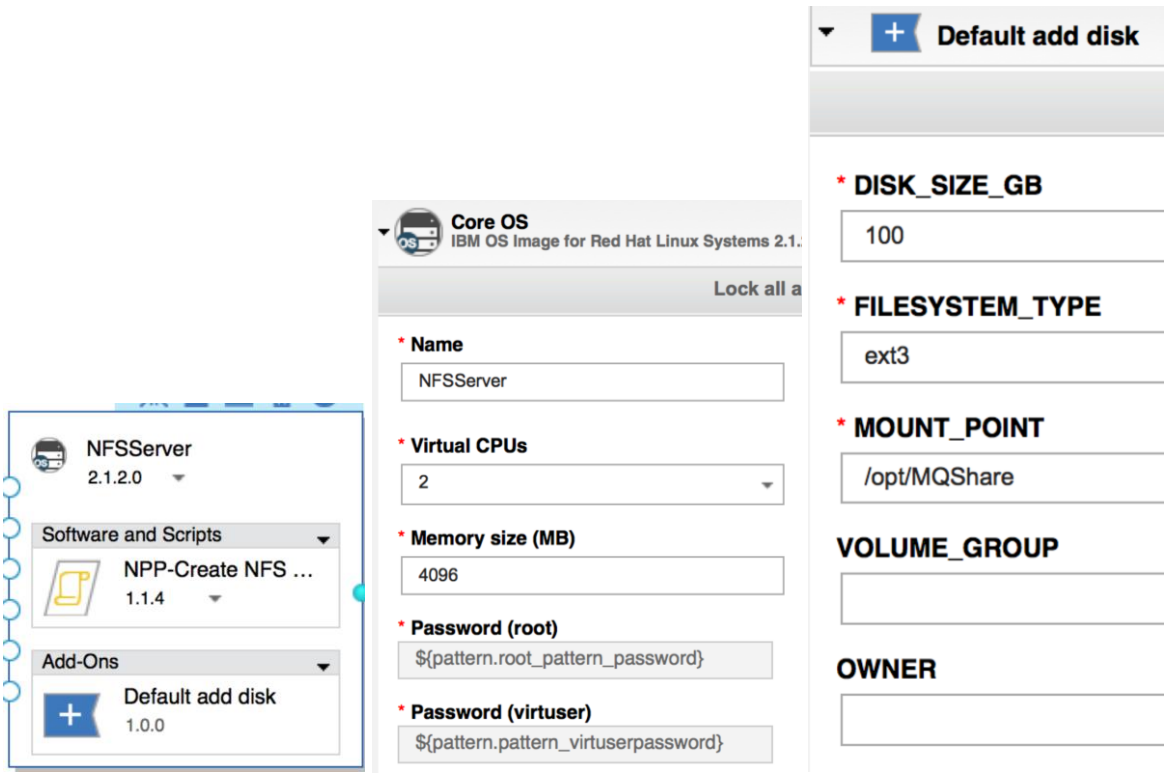
See NFS client configuration section below

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.

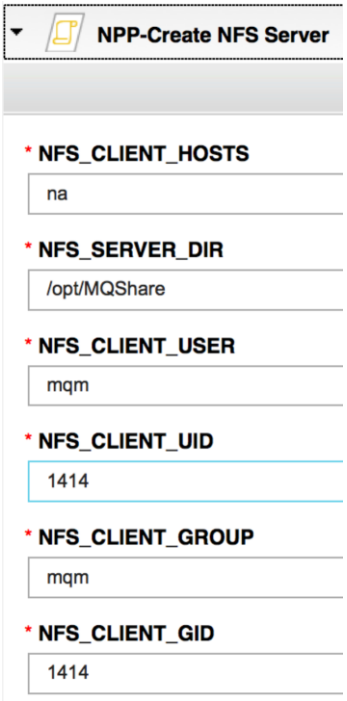
NFS v4 Server node set up



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



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_UDPPORT=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
```

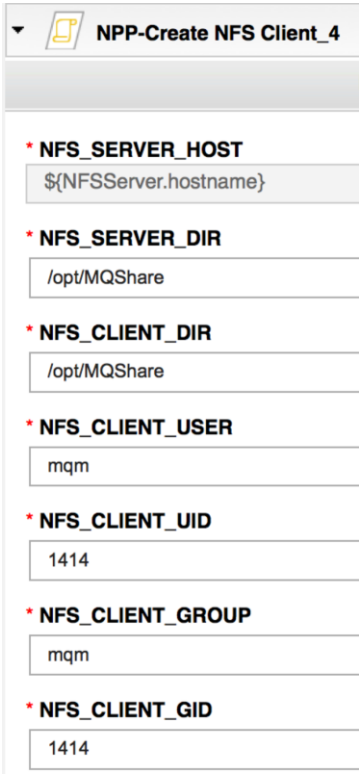
```
echo 'enabling nfs and rpcbind'
chkconfig rpcbind on
chkconfig --list rpcbind
chkconfig nfs on
chkconfig --list nfs

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



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.

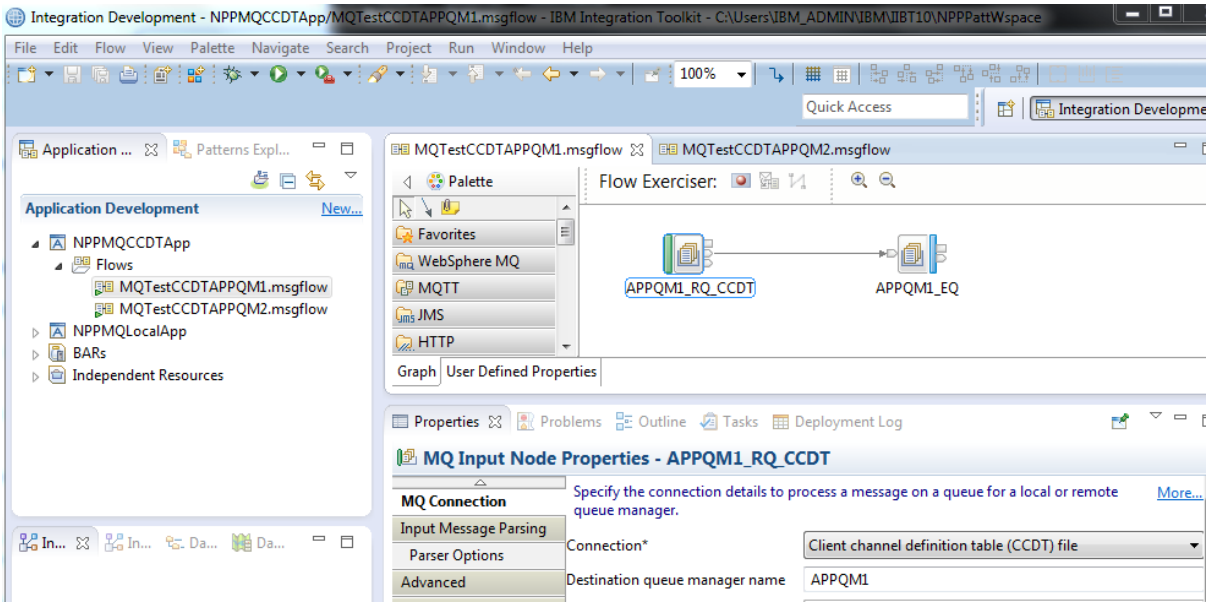
IBM Integration Bus Message Flows

There is a Project Interchange supplied in the materials called IIBCCDTAppProjInt.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