Scheduling jobs/workload from z/OS across the native WSGRID connector to the WXD CG job scheduler on distributed platforms

Antonella Bertoletti
Executive IT Specialist
WebSphere Client Technical Professional Team
IBM Software Group
Table of contents

Introduction ......................................................................................................................................................3
  Purpose of the document ............................................................................................................................3
  Scope of the document ...............................................................................................................................3

Part 1: The WebSphere XD Compute Grid environment ................................................................................4

Part 2: Create the SIB and the MQ link to WebSphere MQ z/OS ...............................................................6
  Create the cluster for the SIB .........................................................................................................................6
  Create the WebSphere SIB ............................................................................................................................7
  Create the foreign bus connection ..............................................................................................................9
  Start the bus cluster ..................................................................................................................................12
  Create the local queue ..............................................................................................................................13

Part 3: WebSphere MQ on z/OS resources definitions .................................................................................14
  Create the transmission queue .................................................................................................................15
  Create the sender channel ..........................................................................................................................15
  Create the receiver channel .........................................................................................................................17
  Create the local queue ..............................................................................................................................17
  Restart the cell ..........................................................................................................................................19

Part 4: Installing the JobSchedulerMDILP application ..................................................................................20
  Install the JobSchedulerMDILP application ...............................................................................................20
  Create JMS resources ................................................................................................................................20
    Connection Factory ...............................................................................................................................20
    Queues ..............................................................................................................................................21
    Activation Specification ..........................................................................................................................21
  Synchronize the changes to the nodes ........................................................................................................22
  Delete the resources created by installWSGridMQClientMode.py ...........................................................22
  Verify that the activation spec is being used .............................................................................................22
  Restart the cluster ClusterSched ..............................................................................................................22

Part 5: Scheduling batch jobs through native WSGRID ................................................................................23
Introduction

WebSphere Extended Deployment Compute Grid V6.1.1 (WXD CG) introduced the WSGRID native connector which allows a workload scheduler such as Tivoli Workload Scheduler for z/OS (TWS) to schedule work into a WXD CG environment more efficiently than the old java WSGRID component. Initially the WSGRID native connector was provided only for scheduling work into WXD CG for z/OS but customers with WXD CG environments on both z/OS and on other platforms wished to use their z/OS-based job scheduler also to schedule work into WXD CG running on distributed platforms. That support was added by APAR PM07417.

The native WSGRID connector makes use of WebSphere MQ for z/OS as a means of communication to/from the z/OS job scheduler into WXD CG running on distributed platforms. Within WXD CG the interface to WebSphere MQ for z/OS could be a client-server connection using the MQ Client or a server-server connection using an MQ Link.

A script was provided with WXD CG 6.1.1.3 to configure the JobSchedulerMDILP application and its related resources. However the script for WXD CG on distributed systems supported only the scenario of an MQ client connection to WebSphere MQ for z/OS and the use of MQ Link was not documented in the Information Center.

Purpose of the document

The purpose of this document is to describe how to schedule batch jobs from z/OS into a WXD CG environment on a distributed platform by means of the WSGRID native connector that is provided as part of the WebSphere Extended Deployment Compute Grid for z/OS.

The configuration described in the pages that follow relies on a WebSphere MQ link configuration between the Service Integration Bus (SIB) of WebSphere Application Server (WAS) running on Red Hat Linux and the WebSphere MQ running on the z/OS platform.

This is a server-to-server configuration where the SIB will treat WebSphere MQ as though it was just another Service Integration Bus and WebSphere MQ will treat the bus as thought it was just another WebSphere MQ Server.

In contrast, if the SIB is configured as a client of the target MQ on z/OS (a client-to-server connection) then it is necessary to install on z/OS the MQ Client Attachment Feature (MQ CAF) which is a separately priced feature.

Scope of the document

This document is limited to describing the environment we set up in order to test the server-to-server connectivity. There are step-by-step instructions to help in re-producing the configuration in your environment.

How to create a WebSphere Application Server cell and how to install and configure WebSphere Extended Deployment Compute Grid on distributed platforms is not explained in this document. Nor are the details of configuring WebSphere XD Compute Grid for z/OS and WebSphere MQ for z/OS. It is assumed that the reader has the knowledge to perform those tasks.
Part 1: The WebSphere XD Compute Grid environment

In order to test native connectivity from the WSGRID native connector on z/OS to WebSphere Extended Deployment Compute Grid on other platforms using an MQ link connection, we created a WebSphere Extended Deployment Compute Grid (WXD CG) cell spread across three Red Hat Linux 64bit machines.

We used a fourth Red Hat Linux 64bit machine on which we installed DB2 v9.7 FP4 for hosting the databases required by the Compute Grid, that is both the LREE (db for the Grid Execution Environment) and the LRSCHED (db for the Job Scheduler).

In the same DB2 instance we created also the database needed by the compute grid installation verification test application, XDCGIVT, that we used as a sample for the tests.

The reason we adopted DB2 instead of the default Derby database is because we wanted to set up our infrastructure like a real production environment with the ability to recover from a crash of any component.

- Operating System Level: Red Hat Enterprise Linux Server release 6.0 (Santiago) 64bit
- WebSphere Application Server Level: WAS Network Deployment v7.0.0.15 64bit
- WebSphere Extended Deployment Compute Grid: WXD Compute Grid v6.1.1.3 64bit

Below is the topology of our cell:

... in which we defined the cluster named 'ClusterSched' that hosts the Compute Grid Job Scheduler component as illustrated below:
... and the cluster named 'IVTCGCluster', made by three Grid Execution Endpoint cluster members, in which we deployed the XDCGIVT sample application.
Part 2: Create the SIB and the MQ link to WebSphere MQ z/OS

This section describes how to create the WebSphere Service Integration Bus (SIB) and the MQ link configuration to WebSphere MQ on z/OS.

Create the cluster for the SIB

First create the cluster that will host the SIB.

Add the cluster members to the cluster definition. For high availability reasons the cluster member number should be at least 2, with the member located on different either physical or virtual machines.

Below are the names chosen for our cluster members and the nodes in which they are configured.

Click Finish to complete the cluster creation and then Save the change to the Master Repository.
Synchronize the nodes by navigating to:
System administration → Save changes to the master repository.
On the next panel, check the box Synchronize changes with Nodes and click Save.

Create the WebSphere SIB

Next create the Service Integration Bus (SIB) definition.

Navigate to Service integration → Buses and then click New in order to create the new SIB.

Name the bus JOBSCHED1, click Next, and then Finish. The new bus is now shown in the list of buses.

Click on the link to the JOBSCHED1 bus and then click Bus members under Topology.

Click Add.
In the next panel select the cluster option and choose the name of the cluster created earlier.

Accept the default for the next two steps and just click **Next**.

In the step **Configuring messaging engines** step, click on the SIB just created (in our case it was called Cluster4Bus1.000-JOBSCHED1) in order to configure the Message Store.

The Message Store can be created as a file store or as data store. A data store implements the Message Store as tables in a database such as DB2. For a high availability production environment you might prefer a data store but for our test we decided to use a file store.

When a cluster is a member of a bus you should configure the permanent store file to be on a file system that is accessible to all members of the cluster.
The specified file store will be created on the system where the Messaging Engines start.

---

Create the foreign bus connection

The next step is to let the bus know about the existence of the MQ network it has to talk to.

The SIB architecture supports notion of a "foreign bus". This is intended to support a gateway either to another SIB (SIBs can be interconnected) or to WebSphere MQ.

When linking to MQ, the SIB will treat the WebSphere MQ system as nothing more than another bus, while MQ will treat the bus as nothing more than another MQ.

In order to define the link between the SIB and the MQ, create a foreign bus connection.
Go back to the SIB definition (in our case **Service integration → Buses → JOBSCHED1** and under Topology click **Foreign bus connections**.

![Foreign bus connections](image)

Click **New**, to start the new foreign bus connection wizard.

Accept the default bus connection type of **Direct connection**.
On the next step choose **WebSphere MQ** as the Foreign Bus type.

Accept the defaults proposed in the next step and click **Next**.
In the next step you define the names of the MQ link and the channels that will be used for connecting the SIB to the MQ on z/OS and vice versa.

We suggest that you give to the Foreign Bus the same name as the target MQ manager, but any name can be used. In our case we chose M600, because this is the name of the MQ manager on our z/OS system.

We set the host name and port of the target MQ manager listener on our z/OS machine. Port 1414 is the default. On your system that port can be different.

We defined two channels:

- The channel JOBSCHED1.TO.M600, which is the sender channel for the SIB and the receiver channel for the z/OS MQ server
- The channel M600.TO.JOBSCHED1 which is the sender channel for the z/OS MQ server and the receiver channel for the SIB.

Click **Next** and then **Finish** to complete the creation.

Then click **Save** in order to save the definitions in the Master repository.

**Start the bus cluster**

Now it's time to start the Cluster hosting the bus.

Go to **Servers → Clusters → WebSphere application server clusters**.

Check the box next to **Cluster4Bus1** and then click **Start**.
Create the local queue

The last thing to do in order to complete the MQ link configuration on the WXD CG distributed environment is the creation of a local queue on the bus.

**Note: Queues that are related to a bus are referred as Destinations.**

One destination is needed that will contain the messages coming from MQ. Since destinations are required for queues managed by some message engine and not for queues belonging to a foreign bus, we don't need to define a second destination queue on the target Queue Manager M600.

Using the WebSphere Administration Console, navigate to:

Service Integration → Buses → JOBSCHED1 → Destinations and click New.

Accept the default destination type of Queue and click Next.

In the next step define the name by which the queue will be known. Since the queue will host the messages coming from MQ, in MQ terms this is referred to as a 'local queue'. Name it LOCALBUS.

Remember that this name has to match that used in the MQ remote queue definition on the mainframe.

Click Next to proceed.

Verify that the assignment to the bus is correct. In our case the bus member was cluster=Cluster4Bus1.

Click Next, then Finish and then Save.
Part 3: WebSphere MQ on z/OS resources definitions

Next define the corresponding definitions on the target queue manager running on the z/OS machine.

As mentioned earlier, on the z/OS we used an existing MQ manager named M600 for which had to define the following resources:

1. A transmission queue JOBSCHED. It will be served by the sender channel that connects the Queue Manager to the target bus on the distributed platform.

2. A sender channel M600.TO.JOBSCHED1. Most of defaults are fine, but we need to choose the transmission queue just created (according to the previous definition that queue is JOBSCHED) and type in the 'Connection Name' field the address by which the host system can point to the Cluster hosting the SIB, plus the port number on which the bus is listening for the MQ traffic. In our case both the members of the cluster listen on port number 5565.

3. A receiver channel JOBSCHED1.TO.M600. This is the receiver channel from the bus.

4. A local queue LOCALMQ. It can be used to test messages sent from the bus to this Queue Manager.

5. A remote queue REMOTEBUS. It will be used to send messages to the bus.

When defining this queue remember to specify the remote queue manager name JOBSCHED1, the remote queue name LOCALBUS and the transmission queue JOBSCHED.

To define resources in MQ you can use the command line interface or the graphical interface provided by MQ Explorer. We decided to use MQ Explorer.

From the MQ Explorer, connect to the queue manager.

In our environment, we connect to M600 on zwasa032.rtp.raleigh.ibm.com(1414).
Create the transmission queue

Define the transmission queue **JOBSCHED** as follows:

![Image of JOBSCHED Properties]

Create the sender channel

Create the sender channel **M600.TO.JOBSCHED1**.

Before proceeding, you need to check the port number on which the Service Integration Bus (SIB) is listening for the MQ traffic. On the WebSphere Administration Console of the WCG cell, navigate to **Servers → Clusters → WebSphere application server clusters** and click on the cluster 'Cluster4Bus1'.

On the next panel click **Cluster members** under **Additional properties** to see the cluster members.
Click on one of the two cluster members (the listening port is the same for all the members in the cluster) and in the next panel, under **Cluster member messaging**, select **WebSphere MQ link inbound transports**.

The InboundBasicMQLink and the InboundSecureMQLink settings will show the listening port to use.

In our 'not secure' environment, the listening port is 5565.

Now go back to the MQ Explorer.

Click on **Channels** and choose **New, Sender Channel**, create the sender channel as follows:
Create the receiver channel

Create the receiver channel JOBSCHED1.TO.M600.

By clicking on Channels and choosing New, Receiver Channel, create the receiver channel as follows:

Create the local queue

Now create the local queue LOCALMQ, by selecting Queues, New, Local Queue.
Remember to set this queue as 'Shareable', because we have two clusters that have to put messages to the same queue. In order to change this setting, move to the Extended tab as shown below.

Create the remote queue

Finally, create the remote queue REMOTEBUS, by selecting Queues, New, Remote Queue Definition.
After the queues definitions have been created you should see them listed as shown below:

![Queue List](image)

### Restart the cell

Now start the cluster members and channels.

Follow the sequence below:

1. On the distributed WXD CG cell, start only the cluster member `sibmem01` of the WebSphere Cluster `Cluster4Bus1`.

2. Start the message engine `Cluster4Bus1.000-JOBSCHED1` related to the bus `JOBSCHED1`.

3. Verify that the M600_link has started, if not, start it.

   (Buses \(\rightarrow\) JOBSCHED1 \(\rightarrow\) Messaging engines \(\rightarrow\) Cluster4Bus1.000-JOBSCHED1 \(\rightarrow\) WebSphere MQ links)

4. Start the MQ link receiver channel `M600.TO.JOBSCHED1`.

   (Buses \(\rightarrow\) JOBSCHED1 \(\rightarrow\) Messaging engines \(\rightarrow\) Cluster4Bus1.000-JOBSCHED1 \(\rightarrow\) WebSphere MQ links \(\rightarrow\) M60_link \(\rightarrow\) Receiver channel)

5. On the WebSphere MQ for z/OS, start the sender channel `M600.TO.JOBSCHED1`.

6. On the distributed WXD CG cell, start the second cluster member `sibmem02` of the WebSphere Cluster `Cluster4Bus1`.

7. Start the MQ link sender channel `JOBSCHED1.TO.M600`.

   (Buses \(\rightarrow\) JOBSCHED1 \(\rightarrow\) Messaging engines \(\rightarrow\) Cluster4Bus1.000-JOBSCHED1 \(\rightarrow\) WebSphere MQ links \(\rightarrow\) M60_link \(\rightarrow\) Sender channel)

8. On the WebSphere MQ for z/OS, start the receiver channel `JOBSCHED1.TO.M600`. 
Part 4: Installing the JobSchedulerMDILP application

An application called JobSchedulerMDILP provides the functions related to the WSGRID native connection and must be installed if you wish to schedule batch jobs using the native WSGRID connector on z/OS to the Compute Grid Job Scheduler running on distributed platforms.

Install the JobSchedulerMDILP application in the same cluster or application server hosting the Compute Grid Job Scheduler and define the resources it will use.

Install the JobSchedulerMDILP application

To install the application, the current level of the product (WXD Compute Grid v6.1.1.3) includes the script named installWSGridMQClientMode.py under the root bin directory. However, this script assumes a client-server configuration between SIB and MQ Server on z/OS where the SIB behaves like a MQ Client towards the target MQ Server running on the mainframe. In a server-to-server configuration the script can be used to install the application but the resources it creates must be deleted after it has run. Then it is necessary to manually create the correct JMS resource definitions for a server-to-server configuration based on the MQ link definition.

Connect to the Linux machine hosting the Deployment Manager, go to the root bin directory (in our environment it was /opt/IBM/WebSphere/AppServer/bin) and find script installWSGridMQClientMode.py.

The script requires all the mandatory inputs shown in the example below:

```
./wsadmin.sh -user xxxxxxxx -password xxxxxxxx -f installWSGridMQClientMode.py -install -cluster Scheduler -qmgr MNGR -qhost myhost.rtp.raleigh.ibm.com -qport 1414 -svrconn WSGRID.SVRCONN -inqueue WASIQ -outqueue WASOQ
```

For our MQ link environment we ran the script using dummy names for all the fields except the cluster, where we used the correct cluster name of the cluster hosting the Job Scheduler.

For example, from the root bin directory run the following command:

```
```

This installs the JobSchedulerMDILP application in the cluster ClusterSched.

Create JMS resources

Now define the correct JMS resources that will be used by the MQ link configuration:

- Connection factory
- Queues
- Activation specification

Connection Factory

From the WebSphere Administration Console, navigate to JMS -> JMS Providers, then select Cluster=ClusterSched as a scope and click Default messaging provider.

Click Connection factories under Additional Properties then click New.

Choose Default messaging provider and enter the following details:

- **Name**: com.ibm.ws.grid.ListenerPortConnectionFactory
- **JNDI Name**: jms/com.ibm.ws.grid.ListenerPortConnectionFactory
- **Bus name**: JOBSCHED1

Click OK and then Save.
Queues

Go back to **Default messaging provider**, click **Queues** under **Additional properties** and then click **New**:

Two queues must be defined, the input queue and the output queue.

**Input queue:**
- **Name**: com.ibm.ws.grid.ListenerPortIncomingQueue
- **JNDI Name**: jms/com.ibm.ws.grid.ListenerPortIncomingQueue
- **Bus name**: JOBSCHED1
- **Queue name**: LOCALBUS

Click **OK** and then **Save**.

**Output queue:**
- **Name**: com.ibm.ws.grid.MQProviderOutputQueue
- **JNDI Name**: jms/com.ibm.ws.grid.MQProviderOutputQueue
- **Bus name**: M600
- **Queue name**: LOCALMQ

Click **OK** and then **Save**.

**Activation Specification**

The last resource is the **Activation specification** definition. Go back to **Default messaging provider**, click **Activation specifications** under **Additional properties**, and then click **New**.

Enter the following details:
- **Name**: com.ibm.ws.grid.ActivationSpec
- **JNDI Name**: eis/com.ibm.ws.grid.ActivationSpec
- **Destination type**: queue
- **Destination JNDI name**: jms/com.ibm.ws.grid.ListenerPortIncomingQueue
- **Bus name**: JOBSCHED1

Click **OK** and then **Save**.
Synchronize the changes to the nodes

Now synchronize the changes to the nodes in the cell by navigating to:
System administration ➔ Save changes to the master repository.

In the next window, check the box Synchronize changes with Nodes and click Save.

Delete the resources created by installWSGridMQClientMode.py

Navigate to: Resources ➔ JMS ➔ JMS Providers, set the scope to ClusterSched, and then click WebSphere MQ messaging provider.

Delete all the resources there which were the ones created by the script for a client-server configuration.

Save the configuration changes and synchronize to the nodes.

Verify that the activation spec is being used

Before starting/restarting the Job Scheduler cluster ClusterSched verify that the application will really use the Activation specification just defined above and not the ListeningPort.

In order to confirm that, move to the machine hosting the Deployment Manager and look under the applications directory in order to look at the JobSchedulerMDILP binding file.

In our environment the path is the following:

```
/opt/IBM/WebSphere/AppServer/profiles/Dmgr01/config/cells/xdblade07b02Cell101/
applications/JobSchedulerMDILP.ear/deployments/JobSchedulerMDILP/
JobSchedulerMDILP.jar/META-INF
```

If you look at the content of the file `ibm-ejb-jar-bnd.xmi` you can check that the activation spec is used.

```
# vi ibm-ejb-jar-bnd.xmi
```

If after the `ejbName` you find ListeningPort instead of activationSpec, change the content as showed in the file above.

If you need to modify the file, synchronize the nodes again and, before starting the cluster, verify that the change has been correctly propagated by checking the `ibm-ejb-jar-bnd.xmi` file on all the nodes on which the Job Scheduler is present.

If not, adjust the content manually.

Restart the cluster ClusterSched

Start or restart the cluster ClusterSched to activate the resources that were defined.
Part 5: Scheduling batch jobs through native WSGRID

Once the Job Scheduler is up and running, launch the Compute Grid Job Management Console (JMC).

In our environment, with a cluster of two cluster members for the Job Scheduler, we could open the JMC on either cluster member one or cluster member two.

To find the hostname:port to use in order to launch the JMC, login to the WebSphere Administration Console and navigate to:

Servers -> Clusters -> WebSphere application server clusters -> ClusterSched

Click Cluster members under Additional properties to see the cluster members as shown below.

![Cluster Members](image)

Click on one of the cluster members and then click Ports. Note the WC_defaulthost port (the port is usually the same for both the cluster members).

The full path is: Servers -> Clusters -> WebSphere application server clusters -> ClusterSched -> Cluster members -> sched02 -> Ports.

In our case the WC_defaulthost port number was the 9081; so in a browser we entered the following URL:

http://xdblade07b03.rtp.raleigh.ibm.com:9081/jmc/console.jsp
If you click on **View jobs** in the Job Management section, you will see the list of the jobs submitted, their current status, the nodes on which they ran or are running, and the application server that served or is serving the request.

Now go to the mainframe machine and login to TSO. On z/OS you must prepare some JCL to run the WSGRID native z/OS program, customized for your environment. Below it’s the JCL we created for our configuration.

According to the MQ Server definitions listed in the previous pages, the scheduler-input-queue is set to REMOTEBUS, whereas the scheduler-output-queue is set to LOCALMQ.

When you submit the job, the execution is visible in the JMC.

The WSGRID native client submits the job and the MDB (provided by the MDILP application) running inside the Job Scheduler schedules the job and sends the information about the JOBID and the execution steps back to the WSGRID client.
In the screen-shot below you can see the job XDCGIVT:00139 which ended correctly. If you click on the job, you access the job log.

From the log you can see the selected grid endpoint and the state of the job.

For example, on our system we received messages like these:

05/02/11 14:40:27:886 EDT] Job XDCGIVT:00139 is queued for execution
CWLRB5586I: [05/02/11 14:40:27:893 EDT] CWLRS6006I: Job class Default, Importance 8, Service Class null, Service Goal Type 0, Application Type j2ee, Submitter UNAUTHENTICATED.
CWLRB5586I: [05/02/11 14:40:27:893 EDT] CWLRS6007I: Job Arrival Time 5/2/11 2:40 PM, Goal Max Completion Time 0, Goal Max Queue Time 0, Breach Time 5/3/11 2:40 PM.
CWLRB5586I: [05/02/11 14:40:27:893 EDT] CWLRS6021I: List of eligible endpoints to execute the job: xdblade07b02Node01/IVTCG01, xdblade07b03Node01/IVTCG02, xdblade07b04Node01/IVTCG03.
CWLRB5586I: [05/02/11 14:40:27:901 EDT] CWLRS6011I: APC is not active. GAP will make the endpoint selection.
CWLRB5586I: [05/02/11 14:40:27:951 EDT] CWLRS6013I: GAP is dispatching job XDCGIVT:00139. Job queue time 0.059 seconds.
CWLRB3090I: [05/02/11 14:40:28:095 EDT] Job XDCGIVT:00139 is dispatched to endpoint xdblade07b04Node01/IVTCG03: result: 0
... and ...

CWLRB5602I: [05/02/11 14:40:29:712 EDT] Job setup manager bean is breaking down job: XDCGIVT:00139
CWLRB5604I: [05/02/11 14:40:29:712 EDT] Freeing IVTStep3 batch data stream: inputStream
CWLRB5604I: [05/02/11 14:40:29:712 EDT] Freeing IVTStep3 batch data stream: generatedOutputInputStream
CWLRB3800I: [05/02/11 14:40:29:731 EDT] Unsubscribing from job cancel or stop subject: BizgridJobCancel_XDCGIVT:00139
CWLRB5594I: [05/02/11 14:40:29:731 EDT] Step IVTStep3 execution is complete: ended normally
CWLRB2250I: [05/02/11 14:40:29:754 EDT] Grid Execution Environment sequential step processing complete: ended
CWLRB2250I: [05/02/11 14:40:29:755 EDT] Job setup manager bean is breaking down job: XDCGIVT:00139
CWLRB5598I: [05/02/11 14:40:29:757 EDT] Removing job abstract resources
CWLRB5600I: [05/02/11 14:40:29:764 EDT] Removing job step status table entries
CWLRB2270I: [05/02/11 14:40:29:778 EDT] Job setup manager bean completed job XDCGIVT:00139 breakdown
CWLRB5764I: [05/02/11 14:40:29:778 EDT] Job XDCGIVT:00139 ended
On the z/OS you can find the corresponding log:

```
session A - [43 x 80]

WebSphere Extended Deployment Compute Grid v6.1.x native WSGRID
WP101783 at ibm.com/support/techdocs

13 June 2011  Page 26 of 26

© 2011, IBM Corporation

We submitted the WSGRID JCL manually, but if you have an Enterprise scheduler such as Tivoli Workload Scheduler (TWS) for z/OS (or any other) you can configure it to submit that JCL and allow it to have complete control of the job’s life cycle.

End of Document