

Job Execution Node

Amar A Shah (samar@in.ibm.com)
Shravan K Kudikala(shravankk@in.ibm.com)

IBM India Software Labs

Note : --

Before using this report be sure to read the general information under "Notices".

© Copyright International Business Machines Corporation 2004.

All rights reserved.

Note to US Government Users -- Documentation related to restricted rights-- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

Table of Contents

<i>NOTICES</i>	<i>---</i>	<i>04</i>
<i>OVERVIEW AND CONFIGURATION</i>	<i>---</i>	<i>05</i>
<i>INSTALLING THE PLUG-IN NODE</i>	<i>---</i>	<i>09</i>
<i>EXAMPLE USAGE OF THE NODE</i>	<i>---</i>	<i>11</i>

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS-IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While IBM has reviewed each item for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
IBM
WebSphere MQ
WebSphere Message Broker (WMB)
IBM Integration Bus (IIB)

The following terms are trademarks of other companies:

Windows 2000/XP Microsoft Corporation

Overview and Configuration

This SupportPac is developed using WebSphere Message Broker's Java plugin node API and hence it can be used on any platform supported by the WMB brokers. However problems might be encountered on some platforms as the JRE and its dependant packages might behave differently. The node is tested on the following platforms Solaris, AIX, Windows XP, Linux and found to work normally.

The JobExecutionNode allows users to run operating system commands , windows batch files, unix shell scripts (henceforth referred as job) using WebSphere Message Broker. Jobs can also be scheduled for specific time and interval using Timer nodes.

The JobExecutionNode can be modified to change the following features of it.

- Message out put format.
- Result storage format in messages.
- Log File and OS Logs format.
- Message Input structure.
- Node Properties.

The JobExecutionNode is developed for the following reasons:

- To enable user to execute Operating System commands, windows batch file, U nix shell scripts from within the message flows.
- To enable user to schedule the tasks and execute them from the Message flows .
- A user may have a need to enrich the message by using the results out of the job execution.

What does this support pack provide you?

- JobExecutionNode to schedule and execute the jobs at specific time, interval, frequency with the help of Timer nodes.
- Node properties where user can specify the details about the job , like, job type, job location, result location, source of the job.
- Result of Job Execution can be stored In a file. The output from the job execution can also be appended to the output message tree.
- Sample message flow to illustrate the node usage.
- The Node also provides a method to do a local node tracing to examine any failure during the job execution.

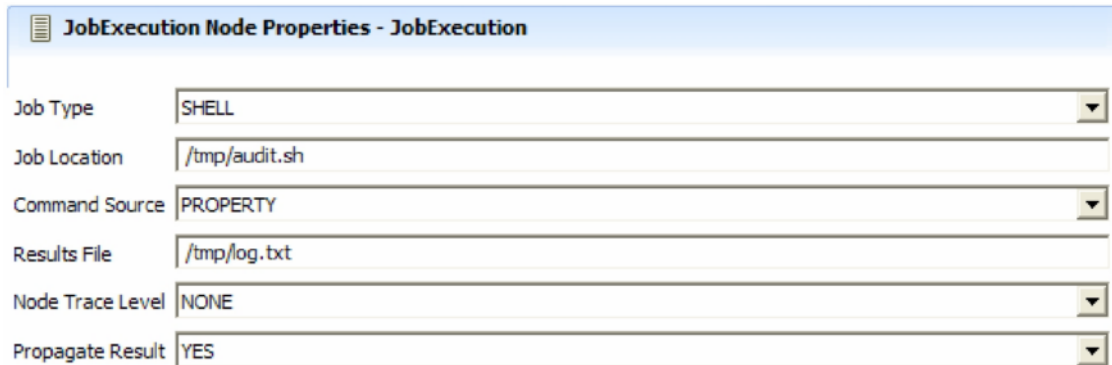
What's planned in the future releases?

- Usage of database instead of storing results in file and propagating a message From the database using the input node.
- Extending the Node to work with JCL (Job Control Language).

The JobExecutionNode node handles messages in the XML format only.

Configuring the Node

The JobExecutionNode has following properties to specify the jobs to be executed. The diagram below shows the properties of Node.



The screenshot shows a configuration window titled "JobExecution Node Properties - JobExecution". It contains several fields with dropdown menus:

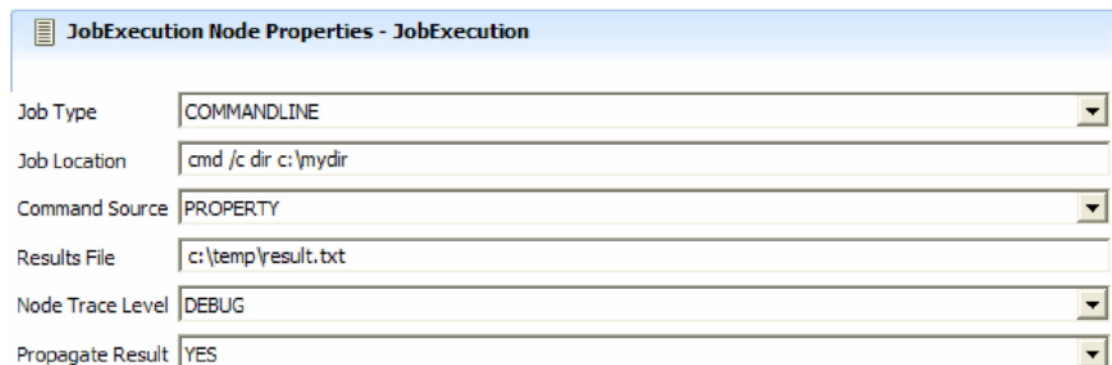
- Job Type: SHELL
- Job Location: /tmp/audit.sh
- Command Source: PROPERTY
- Results File: /tmp/log.txt
- Node Trace Level: NONE
- Propagate Result: YES

Job Type : From the pull down List type of Job can be selected. In this version there are three job types that are supported .

- 1) BATCH - The job to be executed is a windows batch (.bat) file.
- 2) SHELL - The job to be executed is a unix shell (.sh) script.
- 3) COMMANDLINE – The job to be executed is a single command directly to be executed from command prompt (either windows or unix).

Note: -- We have to be careful in deciding job Type. As selecting incorrect job Type will lead to job not getting executed. A shell script can't get executed if we select BATCH as Job Type.

Job Location : This property specifies the Job to be executed. System commands, Unix shell scripts or windows batch files that have to be executed are entered here. If you need to execute Windows Batch Files or Unix Shell scripts, enter the absolute path. If you need to execute more than one command, you can specify them as comma separated.



The screenshot shows a configuration window titled "JobExecution Node Properties - JobExecution". It contains several fields with dropdown menus:

- Job Type: COMMANDLINE
- Job Location: cmd /c dir c:\mydir
- Command Source: PROPERTY
- Results File: c:\temp\result.txt
- Node Trace Level: DEBUG
- Propagate Result: YES

Command Source : This property specifies whether the jobs to be executed would be sourced from the incoming message or from the node property. The two options on the drop down list are –

- 1) MESSAGE - The job will be taken from input message
- 2) PROPERTY - The job will be taken from node property 'Job Location'.

When the command source is selected as MESSAGE, then the inputmessage must have following elements in order in the incoming XML message into the JobExecutionNode.

```
<message><command> "job to be executed"</command></message>
```

Example :

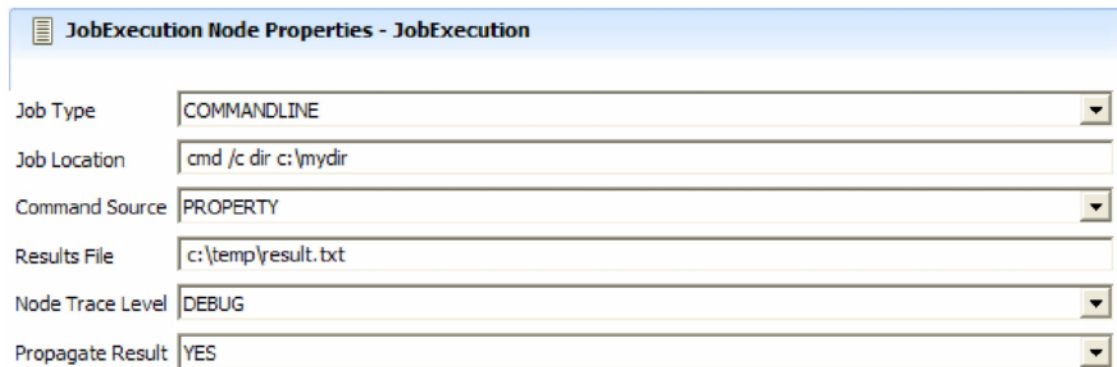
Windows :

```
<message><command>cmd /c dir c:\Work</command></message>
```

Unix :

```
<message><command>ls -l /home</command></message>
```

Results File : User can specify the file where the result of the job to be stored. By default it is blank which means that result will not be stored in the text file on the hard drive.



Job Type	COMMANDLINE
Job Location	cmd /c dir c:\mydir
Command Source	PROPERTY
Results File	c:\temp\result.txt
Node Trace Level	DEBUG
Propagate Result	YES

However the result will be available in the JobExecutionNode if the propagateResult property on the node is set to YES.

If user intends to store the result in a file, an absolute path of the file can be specified here as shown in the above figure.

If the job to be executed is sourced from the message, then the user can also specify different result File for each job in input message as below: -- (This will override the value set on node property).

```
<message><command>cmd /c dir</command><filename>C:\Logs\log.txt</filename></message>
```

Here the results will be stored in filename C:\Logs\log.txt

Node Trace Level : Choose the appropriate level to trace the node. By default this is "NONE". It is encouraged to set NONE on production systems .

The node appends the information into a single log file, irrespective of number of instances of the flows that use this node. The log file "JobeExecutionNodeLogData.txt" can be located in <WMB - install-dir>\bin on windows and on Unix machines it can be located in the /tmp directory. The Node Tracing value set here will have the following significance.

NONE: No logging is done.

ERROR: Log only if an error occurs in the node during message processing.

INFO : Log basic node operations and ERROR(s) if any.

DEBUG : Log very detailed information of the node, including every function it enters and exists along with ERROR(s).

Propagate Result : This property decides whether the output of the job execution will be propagated to the out terminal along with the original message. The default is NO which means job will be fired/scheduled and the message will be propagated to out terminal as it is.

YES : Message will be held in the Job Execution Node and will be propagated after execution of Job is complete and result appended to the message.

NO : Job will be executed and message immediately propagates to out terminal.

Following are the examples with property set to YES.

Example 1:

```
<message><Result command="pwd"> <filename>/tmp/pwd.txt</filename>
<JobType>COMMANDLINE</JobType><output>/opt/IBM/mqsi/6.0/bin</output><error>
</error></Result></message>
```

Example 2: This example shows the error message returned by job execution.

```
<message><Result command="c:\audit.bat"><filename>C:\temp\log.txt</filename>
<JobType>BATCH</JobType><output></output><error>File Not Found. </error>
</Result></message>
```

Warning :

Propagate Result = YES would cause message flow to wait or suspend until the job is over. So use this option with caution if you are scheduling long running job as it may stall the message flow until the job is over.

Connecting the terminals

JobExecutionNode route each message it processes successfully to the out terminal. If this fails, the message is routed to the failure terminal; you can connect nodes to this terminal to handle further processing. If you have not connected the failure terminal, the message is backed out to the input node.

Terminals and properties:

in :-- Input message comes from this terminal

out :-- The output terminal to which the modified message is routed if its processed successfully.

failure:-- The terminal to which the message is routed if an error occurred. If this is not connected then the message is backed out to input node.

Installing the Plug-in Node

SupportPac contents

Download the supportpac and unzip into a temporary directory
The following directories will be present:

\\WMB JobExecutionNode\ia9z.pdf - Current user manual that you are reading

\\WMB JobExecutionNode\6.1Runtime\ - Contains files meant for WMB V6.1 runtime.

\\WMB JobExecutionNode\6.1Workbench\ - Contains files meant for WMB V6.1 Toolkit

\\WMB JobExecutionNode\7.0Runtime\ - Contains files meant for WMB V7.0 runtime.

\\WMB JobExecutionNode\7.0Workbench\ - Contains files meant for WMB V7.0 Toolkit

\\WMB JobExecutionNode\8.0Runtime\ - Contains files meant for WMB V8.0 runtime.

\\WMB JobExecutionNode\8.0Workbench\ - Contains files meant for WMB V8.0 Toolkit

\\IIB JobExecutionNode\9.0Runtime\ - Contains files meant for IIB V9.0 runtime.

\\IIB JobExecutionNode\9.0Workbench\ - Contains files meant for IIB V9.0 Toolkit

Prerequisites

None.

Supported Platforms

This SupportPac has been developed and tested on Microsoft Windows XP, AIX, Linux and Sun Solaris environments. But this plug-in node can be run on any distributed environment that is supported by the WMB Broker V6.1 , v7.0, v8.0 and IIB v9.0

Installing the JobExecution nodes

WebSphere Message Broker v6.1

- Copy '\\WMB JobExecutionNode\6.1Runtime\JobExecutionNode.jar' file to '<WMB-Install-Directory>\jplugin'.
- Copy '\\WMB JobExecutionNode\6.1Workbench\JEN_61.jar' file to '<WMBv6.1Toolkit-Install-Directory>\plugins' directory.
- Restart message broker (use mqsisstart command).
- Restart the toolkit with "--clean" option

WebSphere Message Broker v7.0

- Copy '\\WMB JobExecutionNode\7.0Runtime\JobExecutionNode.jar' file to '<WMB-Install-Directory>\jplugin'.
- Copy '\\WMB JobExecutionNode\7.0Workbench\JEN_70.jar' file to '<WMBv7.0Toolkit-Install-Directory>\plugins' directory.
- Restart message broker (use mqsisstart command).
- Restart the toolkit with "--clean" option

WebSphere Message Broker v8.0

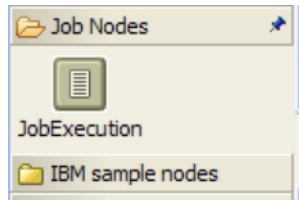
- Copy '\WMB JobExecutionNode \V8.0Runtime\JobExecutionNode.jar' file to '<WMB-Install-Directory>\jplugin'.
- Copy '\WMB JobExecutionNode \V8.0Workbench\JEN_80.jar' file to '<WMBv8.0Toolkit-Install-Directory>\plugins' directory.
- Restart message broker (use mqsisstart command).
- Restart the toolkit with “-clean” option

IBM Integration Bus v9.0

- Copy '\IIB JobExecutionNode \V9.0Runtime\JobExecutionNode.jar' file to '<IIB-Install-Directory>\jplugin'.
- Copy '\IIB JobExecutionNode \V9.0Workbench\JEN_90.jar' file to '<IIBv9.0Toolkit-Install-Directory>\plugins' directory.
- Restart the broker (use mqsisstart command).
- Restart the toolkit with “-clean” option

Sample usage of JobExecutionNode

This section demonstrates the usage of the nodes supplied in this supportpac. Install the plug -in as described above. After the successful installation of the node, you would see the node, JobExecutionNode, in the Message Flow Editor palette, as shown below:



Message Input and Output Format :-

Here we describe two scenarios , one with job sourced from input message and other with job sourced from JobExecution Node properties.

A key thing to note about JobExecution Nodes is the threading model which has been used. If a job is a long running process and another job request comes in, then both are executed parallel. The message flow does not stop accepting new jobs just because the previous jobs are still being executed and pending completion. This threading model helps in scheduling the jobs at specific time or interval.

If a user chooses to append the result (i.e. propagateResult=YES) of the job execution to the outgoing message , then the message will be held in the JobExecutionNode until the Job is executed and results are read.

If the user chooses not to append the result (i.e. propagateResult=NO) to the outgoing message, then the job is just fired and the message is instantly propagated to the out terminal of JobExecutionNode.

i) When the job is sourced from message :

Input messages should be passed with the <message> tag as the root tag. For each job to be executed its corresponding command has to be inside <command> tag which has to be child of <message> tag.

Example:

To execute job 'ls -l' on unix , pass a message in following format.

```
<message><command>ls -l </command></message>
```

The message going out of the node will have following format :

```
<message>
  <Result  command  ="ls -l">
    <output>o/p of ls -l
  </output><error></error> </Result>
</message>
```

ii) When the job is sourced from Node property :

Suppose the job specified is a unix shell script, /tmp/audit.sh

Now suppose we pass the input message as <message>some message</message>

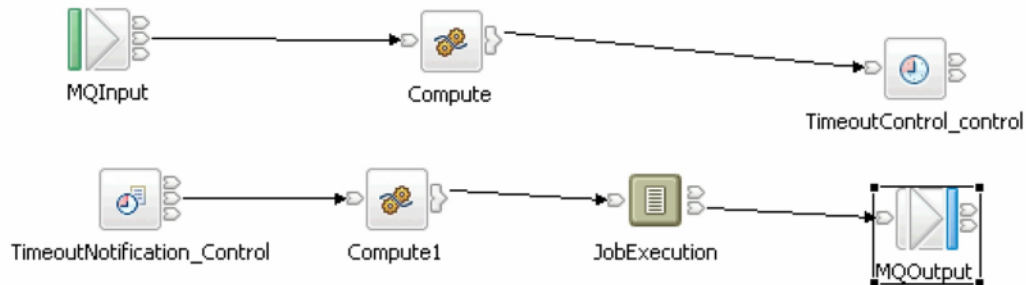
Then the output message will be like :

```
<message>
  <Result  command="/tmp/audit.sh">
    <output>o/p of shell script
  </output><error></error> </Result>
</message>
```

Here we can see that a 'Result' tag gets added for the Job.

A) Using Timer Nodes with JobExecutionNode for scheduling Jobs:

A diagram showing Usage of JobExecutionNode with timer Nodes is as below.



The message to the TimeoutControl node should be of following type

```
<TimeoutRequest>
  <Identifier>id</Identifier>
  <Action>SET</Action>
  <command> ls -l </command>
  <Count>-1</Count>
  <Interval>10</Interval>
</TimeoutRequest>
```

The above fields can be explained as below

- Identifier**: This field has to be unique for each message to be input.
- Command**: This tag will be similar to the command tag defined above.
- Count**: Number of times the commands have to be executed. If this is set to -1. It means infinite times.
- Interval**: Seconds between each execution.

More details regarding above fields and timer Nodes can be found in the following link.

http://www.ibm.com/developerworks/websphere/library/techarticles/0603_schutz/0603_schutz.html