# IS11: Interpreting WBI statistics and accounting

June, 2004

Colin Paice

WebSphere MQ Scenarios department
IBM UK Laboratories
Hursley Park
Winchester
Hampshire
SO21 2JN

E-mail PAICE@UK.IBM.COM

## Take Note!

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

**First Edition, June 2004**

This edition name applies to Version 1.0 of the document *IS11: Interpreting WBI statistics and accounting* and to all subsequent versions and modifications until otherwise indicated.

# Notices

This SupportPac provides information on how to understand and use the Statistics and Accounting facility introduced in WBIMB V5.

### 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:

- IBM
- WebSphere MQ Integrator
- WebSphere Business Integration Message Broker
- DB2
- z/OS

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

- Windows
- Excel

# Table of contents

# Facilities available with WBIMB statistics and accounting

### Introduction

WebSphere Business Integration Message Broker (WBIMB) version 5 introduced a facility to collect message flow statistics and accounting information.

This document provides a guide for different personnel (capacity planner, performance analyst, flow debugger) to make effective use of WBIMB statistics and accounting.

If you are not familiar with this facility, please read *Message flow accounting and statistics data* in the Software information centre for WBIMB.

## Typical tasks

There are various tasks that are typically done when monitoring WBIMB.

1. Monitoring long term usage to ensure there is enough CPU resource available to meet current and future needs.
2. Monitoring throughput through business applications (such as Payroll) to ensure expected throughput and actual throughput are similar. Typical information includes
   - Number of messages processed per day or week
   - Number of messages processed during peak hour in a week
   - Amount of CPU used per week
   - Peak hourly CPU used during a week.
   - Overall cost per message.
3. Comparing the cost per message of the current production flows, to the ‘improved’ flows currently being developed, and evaluating if the change will make a significant difference to the production systems.
4. From the throughput of one execution group or instance, calculate how many execution groups or instances will be required to meet your throughput requirements.
5. Identification of flows, and nodes within flows where there are significant delays. For example when doing a DB2 update this will require I/O to the log data sets.
6. Identification of flows, and nodes within flows which have a relatively high cost per message. When looking to reduce the CPU used by a broker, you should start with the nodes which use a lot of CPU.
7. Identifying flow through error paths, for example if messages flow through error processing nodes, you might want to determine why this is occurring.

## Typical roles

Throughout this document various roles will be used. These are described below

1. Capacity planners: These people are responsible for ensuring there is enough CPU to meet the current and future needs. They typically look at the broker or execution group level information
2. Application developers: These people develop the flows and may look at the cost per node to identify where the CPU time, and elapsed time are spent. They may also be interested in which ‘error’ nodes or error terminals were used.
3. Performance analysts: These people are interested in where elapsed time and CPU time are spent, and work with the applications developers to reduce the elapsed time and cost. They also compare the production and the development level of flows to ensure there are no surprises.
4. Application architects: These people have an interest in which business applications are being used, and are interested in the throughput and costs of processing messages

## What data is collected?

WBIMB provides Snapshot and Archive data. Snapshot is for ‘online’ monitoring of the broker, Archive is for long term monitoring. Snapshot data is produced about three times a minute, and Archive is produced typically hourly, or half hourly.

When the statistics or accounting is activated, the data is collected for the next message to be processed. When the statistics or accounting is made inactive, the data is collected for the current message , and the collection is disabled for the next message. This means you get complete data for a message flowing through a flow.

## What format is the data produced in?

The data can be produced in different formats depending on the parameters used to start the collection of the data.

1. User trace. This will typically be used by the application developer.
2. Published via Pub/Sub. The broker publishes statistics and accounting information as XML messages. See *XML Output- Subscribing to the data on* page 11 for examples of the topic subscription. Interested parties subscribe to the relevant topics. You can write your own programs to process the XML, or use the WBIMB flow provided with this SupportPac and use a database update request to write the data directly to a DB2 database. You can then use tools such as spreadsheets or QMF to report on the data from DB2.
3. SMF. This is only available on z/OS. A C program is provided with this SupportPac to process the data, by printing out the record contents.

Data for a flow goes to only one destination, but different flows can go to different destinations.

## Collecting the data

You have to activate the collection of statistics and accounting before any data is collected. The command syntax is slightly different between z/OS and other platforms.

See mqsichangeflowstats command on page 34 for the full command syntax. Some example commands are given below.

For **z/OS** If you want the data to go to SMF then you need **o=SMF** option, if you want the data to go to an WMQ queue where it can be processed by the STATS flow you need the **o=XML** option.
In the examples below the **z/O**S broker is called VCP1BRK, the **distributed** broker is called **BrokerA**

**Note**. It can take up to 20 seconds for a command to take effect. This is because the command is placed on a queue, and the queue is polled regularly.

<u>Collect long term data on all execution groups and flows in the broker</u>

```
F VCP1BRK,CS A=YES,G=YES,J=YES,C=ACTIVE,T=NONE,N=BASIC,O=XML
```

```
mqischangeflowstats BrokerA –a –g -j -c active –t none –n basic –o xml
```

<u>Collect long term data on an execution group called SAMPLE and all flows in this execution group</u>

```
F VCP1BRK,CS A=YES,E=SAMPLE,J=YES,C=ACTIVE,T=NONE,N=BASIC,O=XML
```

```
mqischangeflowstats BrokerA –a –e SAMPLE –j -c active –t none –n basic –o xml
```

<u>Stop collection of long term data on all execution groups and flows in the broker</u>

```
F VCP1BRK,CS A=YES,G=YES,J=YES,C=INACTIVE,T=NONE,N=BASIC,O=XML
```

```
mqischangeflowstats BrokerA –a –e SAMPLE –j -c inactive –t none –n basic –o xml
```

<u>To display the status of the accounting</u>

```
F VCP1BRK,RS A=YES,G=YES,J=YES
```

```
mqisreportflowstats BrokerA –a –g -j
```

To cause the data to be collected at a particular moment, (perhaps you have done a test and want to collect the statistics), you have to change type of the statistics collected.  For example change the node parameter from advanced to basic (to cause the accounting data to be produced), then change the node parameter from basic back to advanced.  You might issue the following commands to change the level of data collected.

```
F VCP1BRK,CS A=YES,G=YES,J=YES,C=ACTIVE,T=NONE,N=BASIC,O=XML
```

```
F VCP1BRK,CS A=YES,G=YES,J=YES,C=ACTIVE,T=NONE,N=ADVANCED,O=XML
```

```
mqischangeflowstats BrokerA –a –j -c active –n basic –o xml
```

```
mqischangeflowstats BrokerA –a–j -c active –n advanced –o xml
```

This will cause the data to be collected with a reason of StatsSettingsModified. Note.  It may take up to 20 seconds for a command to take effect, so you should allow sufficient time between activating or changing a trace and running your test. Similarly, after your test there may be a delay of up to 20 seconds before the statistics and accounting data is available to you.

## *How to change the interval between Archive records*

*See* the product documentation on the mqsichangebroker command.

### *XML Output- Subscribing to the data*

If you want to use the XML data in a WMQ queue, then you will need to subscribe to the relevant topic, specify a queue, and optionally a queue manager, to get the data sent to the queue.

To subscribe to all archive and snapshot statistics you use a subscription like the following. You can use your own program to issue the subscriptions, or use the data in the fields as input to SupportPac IH03, see page 41 for more information.

```
<psc><Command>RegSub</Command>
<Topic>$SYS/Broker/VCP1BRK/StatisticsAccounting/#</Topic>
<QName>STATS</QName><QMgrName>VCP1</QMgrName>
</psc>
```

The data will be sent to queue STATS on queue manager VCP1.

To subscribe only to archive statistics about an execution group SAMPLE use a subscription like

```
<psc><Command>RegSub</Command>
<Topic>$SYS/Broker/VCP1BRK/StatisticsAccounting/Archive/SAMPLE/#<Topic>
<QName>STATS</QName><QMgrName>VCP1</QMgrName>
</psc>
```

To subscribe only to archive statistics about a flow DB2U within an execution group SAMPLE use a subscription like

```
<psc><Command>RegSub</Command>
<Topic>$SYS/Broker/VCP1BRK/StatisticsAccounting/Archive/SAMPLE/DB2U<Topic>
<QName>STATS</QName><QMgrName>VCP1</QMgrName>
</psc>
```

If you no longer want to subscribe to archive or snapshot statistics data then you can issue an unsubscribe command like

```
<psc><Command>DeregSub</Command>
<Topic>$SYS/Broker/VCP1BRK/StatisticsAccounting/#</Topic>
<QName>STATS</QName><QMgrName>VCP1</QMgrName>
</psc>
```

Or to delete **all** subscriptions which go to the queue STATS on queue manager VCP1

```
<psc><Command>DeregSub</Command>
<RegOpt>DeregAll</RegOpt>
<QName>STATS</QName><QMgrName>VCP1</QMgrName>
</psc>
```

Note. The output queue can be defined as a remote queue, so all of your data is sent to a central queue manager for processing.

# What is provided in this SupportPac?

This SupportPac has definitions and tools to help you process the accounting and statistics data you have collected.

- For z/OS a program is provided which prints out the SMF records
- A WBIMB flow which takes the accounting and statistics data and inserts it into a DB2 database.
- The definitions to support the insertion of data into DB2.
- Definitions for flow, node, thread, and terminal tables
- A view for each table where the dates are converted to Week number within year, and data is summarised
- A document showing how to access a DB2 database and display the data in an Excel spreadsheet.

## *DB2 Definitions*

The Data Definition Language for the tables is given in DB2 table definitions on page 38.

Several views are defined to make it easier to display the data. Many fields have the same name in the views as in the base table, for example *flow*, but some fields have been changed to summarise the data. For example the sum of the CPU time in micro seconds could cause overflow, so the value is converted to seconds.

The minimum CPU and elapsed times are calculated when the number of messages or times used is greater than 0.

The changed or additional fields in the views are given below.

### Aw_flowstats – Flow accounting data summarised by week

This displays the flow accounting data with the Startdate rounded down to the previous Monday. This allows data to be displayed by week.

| Field name | Description |
|---|---|
| StartWeek | is the date of the start of the week. It is calculated from *startdate +1 day - dayofweek(startdate) days* |
| TOTCT in Secs | is the sum of the CPU time used processing messages, converted to seconds |
| WAITCT in Secs | is the sum of the CPU time spent waiting for messages, and the time is converted to seconds. |
| SUMCT in Secs | is the sum of the CPU time used processing messages, and the CPU time spent waiting for messages, and the time is converted to seconds. |
| TOTET in Secs | is the elapsed time spent processing for messages, and the time is converted to seconds. |
| WAITET in Secs | is the elapsed time spent waiting for messages, and the time is converted to seconds. |

| | |
|---|---|
| AVGCT in uSecs | is the total CPU time divided by the number of messages |
| AVGET in uSecs | is the total elapsed time processing messages divided by the number of messages. |

## Aw_nodestats – Node accounting data summarised by week

This displays the node accounting data with the Startdate rounded down to the previous Monday. This allows data to be displayed by week.

| Field name | Description |
|---|---|
| StartWeek | is the date of the start of the week. It is calculated from *startdate +1 day - dayofweek(startdate) days* |
| TOTCT in Secs | is the sum of the CPU time used processing messages, converted to seconds |
| TOTET in Secs | is the elapsed time spent processing for messages, and the time is converted to seconds. |
| AVGCT in uSecs | is the total CPU time divided by the number of messages |
| AVGET in uSecs | is the total elapsed time processing messages divided by the number of messages. |

## Ad_flowstats – Flow accounting data summarised by day

This displays the node accounting data with the data summarised collected by Startdate.

| Field name | Description |
|---|---|
| TOTCT in Secs | is the sum of the CPU time used processing messages, converted to seconds |
| WAITCT in Secs | is the sum of the CPU time spent waiting for messages, and the time is converted to seconds. |
| SUMCT in Secs | is the sum of the CPU time used processing messages, and the CPU time spent waiting for messages, and the time is converted to seconds. |
| TOTET in Secs | is the elapsed time spent processing for messages, and the time is converted to seconds. |
| WAITET in Secs | is the elapsed time spent waiting for messages, and the time is converted to seconds. |
| AVGCT in uSecs | is the total CPU time divided by the number of messages |
| AVGET in uSecs | is the total elapsed time processing messages divided by the number of messages. |

## Ad_nodestats – Node accounting data summarised by day

This displays the node accounting data with the data summarised collected by Startdate.

| Field name | Description |
|---|---|
| TOTCT in Secs | is the sum of the CPU time used processing messages, converted to seconds |
| TOTET in Secs | is the elapsed time spent processing for messages, and the time is converted to seconds. |
| AVGCT in uSecs | is the total CPU time divided by the number of messages |
| AVGET in uSecs | is the total elapsed time processing messages divided by the number of messages. |

## Ah_flowstats – Flow accounting data summarised by hour

This displays the flow accounting data with the Starthour rounded down to the previous hour boundary. This allows data to be displayed by hour. This view would usually be used in a query to display data for a particular day. In this case a where-clause like *where startdate = '2003-12-05'* would be used to specify the particular day.

| Field name | Description |
|---|---|
| StartHour | is the hour the start of the perioid. It is calculated from *starttime - minute(starttime) minutes - second(starttime) seconds* |
| TOTCT in Secs | is the sum of the CPU time used processing messages, converted to seconds |
| WAITCT in Secs | is the sum of the CPU time spent waiting for messages, and the time is converted to seconds. |
| SUMCT in Secs | is the sum of the CPU time used processing messages, and the CPU time spent waiting for messages, and the time is converted to seconds. |
| TOTET in Secs | is the elapsed time spent processing for messages, and the time is converted to seconds. |
| WAITET in Secs | is the elapsed time spent waiting for messages, and the time is converted to seconds. |
| AVGCT in uSecs | is the total CPU time divided by the number of messages |
| AVGET in uSecs | is the total elapsed time processing messages divided by the number of messages. |

## Ah_nodestats – Node accounting data summarised by hour

This displays the node accounting data with the Starthour rounded down to the previous hour boundary. This allows data to be displayed by hour. This view would usually be used in a query to display data for a particular day. In this case a where clause like *where startdate = '2003-12-05'* would be used to specify the particular day.

.

| Field name | Description |
|---|---|
| StartHour | this is the hour the start of the period. It is calculated from *starttime - minute(starttime) minutes - second(starttime) seconds* |
| TOTCT in Secs | is the sum of the CPU time used processing messages, converted to seconds |
| TOTET in Secs | is the elapsed time spent processing for messages, and the time is converted to seconds. |
| AVGCT in uSecs | is the total CPU time divided by the number of messages |
| AVGET in uSecs | is the total elapsed time processing messages divided by the number of messages. |

## Typical Tasks

The following examples give example DB2 queries and use the DB2 databases which are loaded from the flow.  These are described in the following pages

- Displaying the CPU used by a broker over a long period
- Display the number of messages processed by a broker over a long period
- Display the average cost or processing a message over a long period
- Displaying the costs per node over a long period
- Displaying the elapsed time processing in a node
- How to tell if there are any unusual paths through the flow
- How to tell if there any errors in the flows
- How applications programmers can check their flows

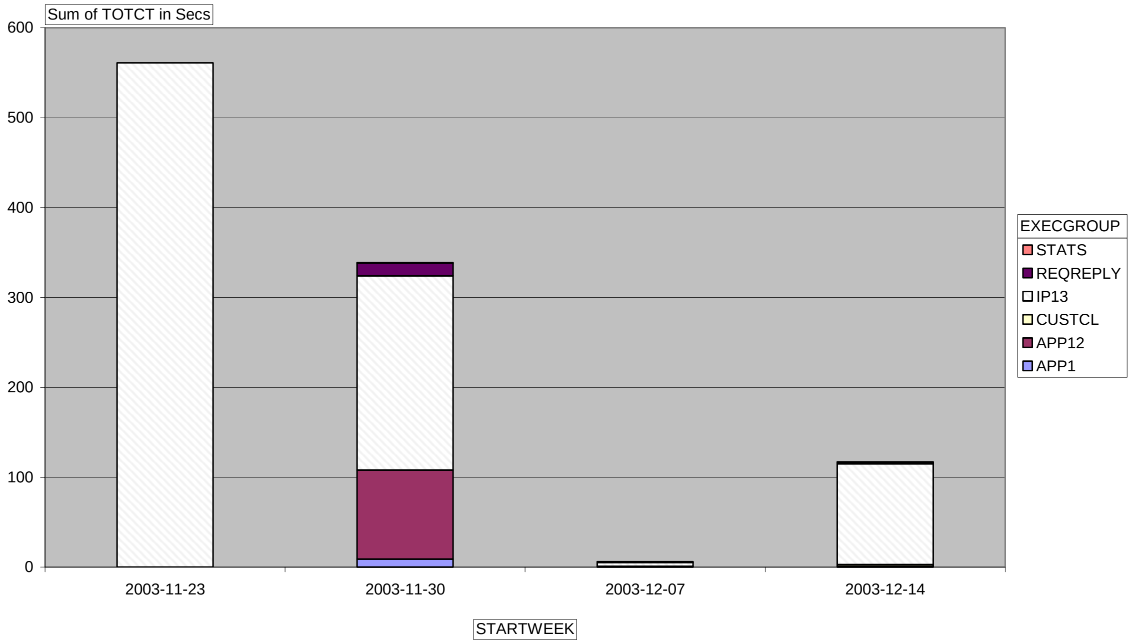
# Displaying the CPU used by a broker over a long period

You can use the spreadsheet aw_flowstats to display the CPU used by week. The example below shows the amount of CPU time used in seconds by execution group.
With this SupportPac is a document describing how to extract data from a DB2 database and display it in Excel.

For example

| Sum of TOTCT in Secs | | | | | | | |
|---|---|---|---|---|---|---|---|
| STARTWEEK | APP1 | APP12 | CUSTCL | IP13 | REQREPLY | STATS | Grand Total |
| 2003-11-23 | | | | 561 | 0 | 0 | 561 |
| 2003-11-30 | 9 | 99 | | 216 | 14 | 1 | 339 |
| 2003-12-07 | | 1 | | 4 | 0 | 1 | 6 |
| 2003-12-14 | | 1 | 2 | 112 | 1 | 1 | 117 |
| Grand Total | 9 | 101 | 2 | 893 | 15 | 3 | 1023 |

Displaying the same data graphically gives



If you have more than one broker on more than one platform, you may want to display the data for each broker individually as the CPU values are not comparable across platforms.

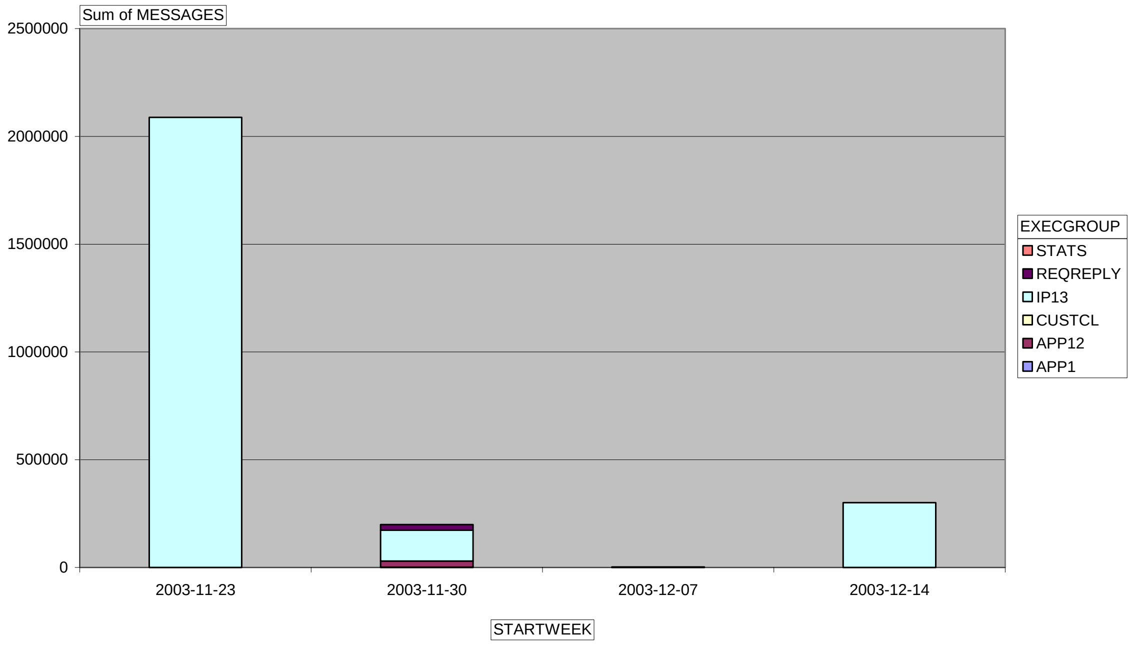
You could use a DB2 query like *Select * from aw_flowstats where broker= 'VCP1BRK'* to display the data for the specified broker.

## Display the number of messages processed by a broker over a long period

You can change the spreadsheet aw_flowstats to display the number of messages processed per week.  See the documentation on using Excel for more information on how to do this.
The example below shows the number of messages processed by execution group.

| Sum of MESSAGES | | | | | | | |
|---|---|---|---|---|---|---|---|
| STARTWEEK | APP1 | APP12 | CUSTCL | IP13 | REQREPLY | STATS | Grand Total |
| 2003-11-23 | | | | 2088003 | 0 | 0 | 2088003 |
| 2003-11-30 | 2000 | 28106 | | 142993 | 25996 | 2 | 199097 |
| 2003-12-07 | | 0 | | 3000 | 0 | 2 | 3002 |
| 2003-12-14 | | 0 | 0 | 301001 | 0 | 1 | 301002 |
| Grand Total | 2000 | 28106 | 0 | 2534997 | 25996 | 5 | 2591104 |

Displaying the same data graphically gives



If you have multiple brokers you could display the data with brokers instead of execution groups.  This would show you how many messages each broker processed.

# Display the average cost or processing a message over a long period

You can change the spreadsheet aw_flowstats to display the average cost to process a message by each execution group.

| Average of AVG COST in us | | | | | | |
|---|---|---|---|---|---|---|
| STARTWEEK | APP1 | APP12 | CUSTCL | IP13 | REQREPLY | STATS |
| 2003-11-23 | | | | 388 | 0 | 0 |
| 2003-11-30 | 1937 | 1754 | | 1132 | 156 | 8490 |
| 2003-12-07 | | 0 | | 47 | 0 | 9023 |
| 2003-12-14 | | 0 | 0 | 3664 | 0 | 11425 |

# Drilling down into the cost of message processing

In *Displaying the CPU used by a broker over a long period* on page 17, the execution group IP13 can be seen to have used most of the CPU time used by the broker (893 seconds out of 1023 seconds). You can break down this cost to idenfify where the CPU time is being spent.

By using the aw_flowstats spreadsheet to display the data in the table you can select the flow data for just execution group IP13. See the documentation on using Excel for more information on how to do this.

You could also have changed the DB2 query to include *where execgroup= 'IP13'* in the search argument.

The table looks like

| Sum of TOTCT in Secs | | | | | | |
|---|---|---|---|---|---|---|
| | IP13 | | | | IP13 Total | Grand Total |
| STARTWEEK | DB2U | FANIN | FANOUT plus original msg | ONE2ONE | | |
| 2003-11-23 | 24 | 0 | 0 | 537 | 561 | 561 |
| 2003-11-30 | 1 | 113 | 65 | 37 | 216 | 216 |
| 2003-12-07 | 0 | 2 | 0 | 2 | 4 | 4 |
| 2003-12-14 | 1 | 4 | 1 | 106 | 112 | 112 |
| Grand Total | 26 | 119 | 66 | 682 | 893 | 893 |

From this you can see that the flows FANIN and ONE2ONE have the highest cost.

The high cost could be due to expensive processing done relatively infrequently or a lot of inexpensive processing. The number of messages processed by each node is

| Sum of MESSAGES | | | | | | |
|---|---|---|---|---|---|---|
| | IP13 | | | | IP13 Total | Grand Total |
| STARTWEEK | DB2U | FANIN | FANOUT plus original msg | ONE2ONE | | |
| 2003-11-23 | 14003 | 0 | 0 | 2074000 | 2088003 | 2088003 |
| 2003-11-30 | 0 | 25995 | 12998 | 104000 | 142993 | 142993 |
| 2003-12-07 | 0 | 0 | 0 | 3000 | 3000 | 3000 |
| 2003-12-14 | 1 | 0 | 0 | 301000 | 301001 | 301001 |
| Grand Total | 14004 | 25995 | 12998 | 2482000 | 2534997 | 2534997 |

This shows that ONE2ONE was used many times, and FANIN was used less often.

You can display the average cost of processing in each node, for example

| Average of AVGCT in uS | | | | | | |
|---|---|---|---|---|---|---|
| | IP13 | | | | IP13 Total | Grand Total |
| STARTWEEK | DB2U | FANIN | FANOUT plus original msg | ONE2ONE | | |
| 2003-11-23 | 1682 | 0 | 0 | 259 | 388 | 388 |
| 2003-11-30 | 0 | 2001 | 2342 | 185 | 1132 | 1132 |
| 2003-12-07 | 0 | 0 | 0 | 188 | 47 | 47 |
| 2003-12-14 | 14464 | 0 | 0 | 191 | 3664 | 3664 |
| Grand Total | 3229 | 800 | 937 | 201 | 1230 | 1230 |

This shows that the cost of ONE2ONE is much less than the cost of FANIN.

## Displaying the costs per node over a long period

With the node statistics DB2 table you can display information about the cost of nodes in a flow. For example you can use the aw_nodestats spread sheet.

The fields of most interest are: StartWeek( or StartDate), Flow, Node, “TotCT in Secs”, “AvgCT in uS”, and TimesUsed

For example from the sample data provided, the flow FANIN has been selected

| Sum of TOTCT in Secs | | | | | | |
|---|---|---|---|---|---|---|
| | FANIN | | | | FANIN Total | Grand Total |
| STARTWEEK | AGG.OUT | Aggregate Reply | REPLY | Reply or Control Msg | | |
| 2003-11-23 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2003-11-30 | 8 | 68 | 23 | 2 | 101 | 101 |
| 2003-12-07 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2003-12-14 | 0 | 0 | 0 | 0 | 0 | 0 |
| Grand Total | 8 | 68 | 23 | 2 | 101 | 101 |

This shows that most of the costs are in the AggregrateReply node. This illustrates the relative cost of using aggregation in a flow. Aggregation has to update DB2 tables which increases the cost of using aggregation.

## Displaying the elapsed time processing in a node

Using  a DB2 query on the aw_nodestats DB2 view you can display the average, maximum and minimum elapsed times within a node.

For example

```
select   startweek,node,nodetype,"AVGET in uSecs","MAXET in uSecs","MINET
in uSecs",timesused   from aw_nodestats
 where  EXECGROUP='IP13'
        and timesused  > 0
```

might give the results below.

| STARTWEEK | NODE | NODETYPE | AVGET in uSecs | MAXET in uSecs | MINET in uSecs | TIMES USED |
|---|---|---|---:|---:|---:|---:|
| 2003-11-30 | AggregateControl | AggregateControlNode | 2360 | 30302 | 2010 | 14584 |
| 2003-11-30 | AggReq_for_ A | AggregateRequestNode | 763 | 155037 | 605 | 14584 |
| 2003-11-30 | AggReq_for_IP_MSG | AggregateRequestNode | 344 | 20359 | 296 | 14584 |
| 2003-11-30 | AGG.OUT | MQOutputNode | 813 | 7326 | 729 | 14628 |
| 2003-11-30 | Build Reply | ComputeNode | 496 | 4598 | 435 | 14628 |
| 2003-11-30 | Build_A | ComputeNode | 418 | 7405 | 367 | 14584 |
| 2003-11-30 | Compute | ComputeNode | 531 | 16458 | 436 | 29257 |
| 2003-11-30 | Compute | ComputeNode | 775 | 5156 | 682 | 14584 |
| 2003-11-23 | DB2U.OUT | MQOutputNode | 189 | 2622 | 148 | 14001 |
| 2003-11-23 | FONE2ONE.IN | MQInputNode | 1964 | 154097 | 78 | 2059338 |
| 2003-11-30 | FONE2ONE.IN | MQInputNode | 93 | 6412 | 74 | 101099 |
| 2003-12-07 | FONE2ONE.IN | MQInputNode | 87 | 971 | 73 | 2000 |
| 2003-12-14 | FONE2ONE.IN | MQInputNode | 85 | 6625 | 75 | 301000 |
| 2003-11-23 | FONE2ONE.OUT | MQOutputNode | 214 | 22460 | 166 | 2059145 |
| 2003-11-30 | FONE2ONE.OUT | MQOutputNode | 188 | 4941 | 164 | 101000 |
| 2003-12-07 | FONE2ONE.OUT | MQOutputNode | 191 | 1130 | 162 | 2000 |
| 2003-12-14 | FONE2ONE.OUT | MQOutputNode | 181 | 9560 | 163 | 301000 |

From this it can be seen that the FONE2ONE.IN node on the week starting 2003-11-23 had a very long maximum response time.  You can then use the other views to more information.

For example displaying the data from the hourly view

```
select  startdate,starthour,node,nodetype,"AVGET in uSecs","MAXET in uSecs"
,"MINET in uSecs",timesused   from ah_nodestats
 where EXECGROUP='IP13'
             and startdate >= '2003-11-23'
             and startdate < '2003-11-30'
             and node = 'FONE2ONE.IN'
            and timesused  > 0
```

This gave the following

| NODE | NODETYPE | AVGET in uSecs | MAXET in uSecs | MINET in uSecs | TIMES USED | STARTDATE | STARTHOUR |
|---|---|---|---|---|---|---|---|
| FONE2ONE.IN | MQInputNode | 87 | 753 | 78 | 1041 | 2003-11-24 | 18:00:00 |
| FONE2ONE.IN | MQInputNode | 2006 | 89189 | 1124 | 599482 | 2003-11-25 | 08:00:00 |
| FONE2ONE.IN | MQInputNode | 1996 | 154097 | 1414 | 398815 | 2003-11-25 | 09:00:00 |
| FONE2ONE.IN | MQInputNode | 2042 | 45191 | 993 | 119000 | 2003-11-25 | 14:00:00 |
| FONE2ONE.IN | MQInputNode | 2078 | 51032 | 1398 | 597600 | 2003-11-25 | 15:00:00 |
| FONE2ONE.IN | MQInputNode | 1962 | 29430 | 1407 | 283400 | 2003-11-25 | 16:00:00 |
| FONE2ONE.IN | MQInputNode | 87 | 1002 | 79 | 10000 | 2003-11-27 | 14:00:00 |
| FONE2ONE.IN | MQInputNode | 85 | 686 | 78 | 10000 | 2003-11-27 | 16:00:00 |
| FONE2ONE.IN | MQInputNode | 88 | 268 | 79 | 20000 | 2003-11-28 | 08:00:00 |
| FONE2ONE.IN | MQInputNode | 89 | 358 | 81 | 10000 | 2003-11-28 | 09:00:00 |
| FONE2ONE.IN | MQInputNode | 88 | 327 | 80 | 10000 | 2003-11-28 | 10:00:00 |

You can use an SQL query to display the average CPU time used and average elapsed time for the above times

```
select   startdate,starthour,node,nodetype,"AVGET in uSecs","AVGCT in uSecs"
,timesused   from ah_nodestats
 where  EXECGROUP='IP13'
              and startdate >= '2003-11-23'
              and startdate < '2003-11-30'
              and node = 'FONE2ONE.IN'
             and timesused  > 0
```

This gave the information below

| STARTDATE | STARTHOUR | NODE | NODETYPE | AVGET in uSecs | AVGCT in uSecs | TIMES USED |
|---|---|---|---|---|---|---|
| 2003-11-24 | 18:00:00 | FONE2ONE.IN | MQInputNode | 87 | 63 | 1041 |
| 2003-11-25 | 08:00:00 | FONE2ONE.IN | MQInputNode | 2006 | 105 | 599482 |
| 2003-11-25 | 09:00:00 | FONE2ONE.IN | MQInputNode | 1996 | 104 | 398815 |
| 2003-11-25 | 14:00:00 | FONE2ONE.IN | MQInputNode | 2042 | 105 | 119000 |
| 2003-11-25 | 15:00:00 | FONE2ONE.IN | MQInputNode | 2078 | 105 | 597600 |
| 2003-11-25 | 16:00:00 | FONE2ONE.IN | MQInputNode | 1962 | 105 | 283400 |
| 2003-11-27 | 14:00:00 | FONE2ONE.IN | MQInputNode | 87 | 63 | 10000 |
| 2003-11-27 | 16:00:00 | FONE2ONE.IN | MQInputNode | 85 | 62 | 10000 |
| 2003-11-28 | 08:00:00 | FONE2ONE.IN | MQInputNode | 88 | 64 | 20000 |
| 2003-11-28 | 09:00:00 | FONE2ONE.IN | MQInputNode | 89 | 65 | 10000 |
| 2003-11-28 | 10:00:00 | FONE2ONE.IN | MQInputNode | 88 | 64 | 10000 |

On the 25th of November the messages were changed to be persistent while we did some investigation. This can be seen from the increase CPU costs which increased from about 64 to 104 micro seconds, and there the elapsed time is now significantly more than the CPU time used. This indicates some I/O operations were performed – and processing persistent messages requires disk I/O.

During this time, the queue manager log filled up and switched to the next log. While the log switch is occurring the time for a commit can increase.  This explains the high 154097 microsecond maximum elapsed time.

### How to tell if there are any unusual paths through the flow

You can tell how many requests went through each terminal. For example a node may have good data flowing through an out terminal, and bad data flowing through an error terminal. You can display how many requests were processed by each terminal

You need to collect the data at the flow, node and terminal level.

```
Select
  TERMLABEL,TERMTYPE, USECOUNT,
  NODELABEL,NODETYPE,
  BROKERLABEL,EGNAME,
  MFLOWNAME,STARTDATEWK,STARTTIME
  from  termSTATSWK
  where USECOUNT > 0;
```

Part of example output

```
TERMLABEL TERMTYPE    USECOUNT     NODELABEL
Error       Output           1     My Error test
In           Input        3129     Build Reply
In           Input        6258     Compute
In           Input        6258     Reply or Control
In           Input        3129     Write Reply
Out         Output        3129     AggregateReply
True        Output        6258     Reply or Control
```

You can see that there was one message from the Error terminal.

### How to tell if there any errors in the flows

You can get different types of errors when processing messages in flows. For example you may receive an error when:

- an MQInputNode received an error getting a message,
- the input message is badly formed,
- there is a problem putting a message to a queue,
- there are processing errors, e.g. an invalid column specified in a DB2 request.

These errors are reported in the statistics.

As an example, the replyto queue for a flow was set to put disabled.  This caused an MQCC =2 and MQRC=2051 when a message was put to the input queue.

```
select
 BROKERLABEL,EGNAME,
 STARTDATE,STARTTIME,
  NUMMQERR,NUMMSGERR,NUMERR, NUMCOMMITS,NUMBACKOUT
    from  flowstats
    where NUMMQERR>0 or NUMMSGERR> 0 or NUMERR > 0;
```

Portion of output

```
EXECGROUP NUMMQERR  NUMMSGERR NUMERR NUMCOMMITS  NUMBACKOUT
SAMPLE           1          0      1          1           1
```

Notes

1. NUMMQERR = 1 shows we had one WMQ error, but as we know it was not an error on the MQGET, in this case it is an error at MQPUT
2. NUMMSGERR = 0 shows we did not have any errors getting messages
3. NUMERR = 1 shows there was one error, which we know was from the MQPUT above
4. For the message there was one backout request and one commit request. When the error was detected, the flow issued **a backout** request.  When the message was got a second time, the MQInputNode checked the message backout count.  As this was greater than zero, the node put this on the queue's backout queue and issued **a commit**.  If the queue's backout queue had not been specified, then the messages would have been put on the system dead letter queue.

The following SQL statement will display the above information in a more usable way

```
 select execgroup,
        nummsgerr as MQGET,
        NUMMQERR - NUMMsgERR  as MQOTHER,
        numerr - nummqerr - nummsgerr as other
        from flowstats
        where numerr > 0
```

This gave

```
EXECGROUP MQGET  MQOTHER OTHER
SAMPLE        0        1     0
```

The reply to queue was reset and a change was made to a flow to cause a DB2 exception and one message was processed.  This caused NUMERR to be set to 1, and NUMMQERR, NUMMSGERR both 0.   This reflects that there was one error detected, and there were no WMQ errors detected.

### *How applications programmers can check their flows*

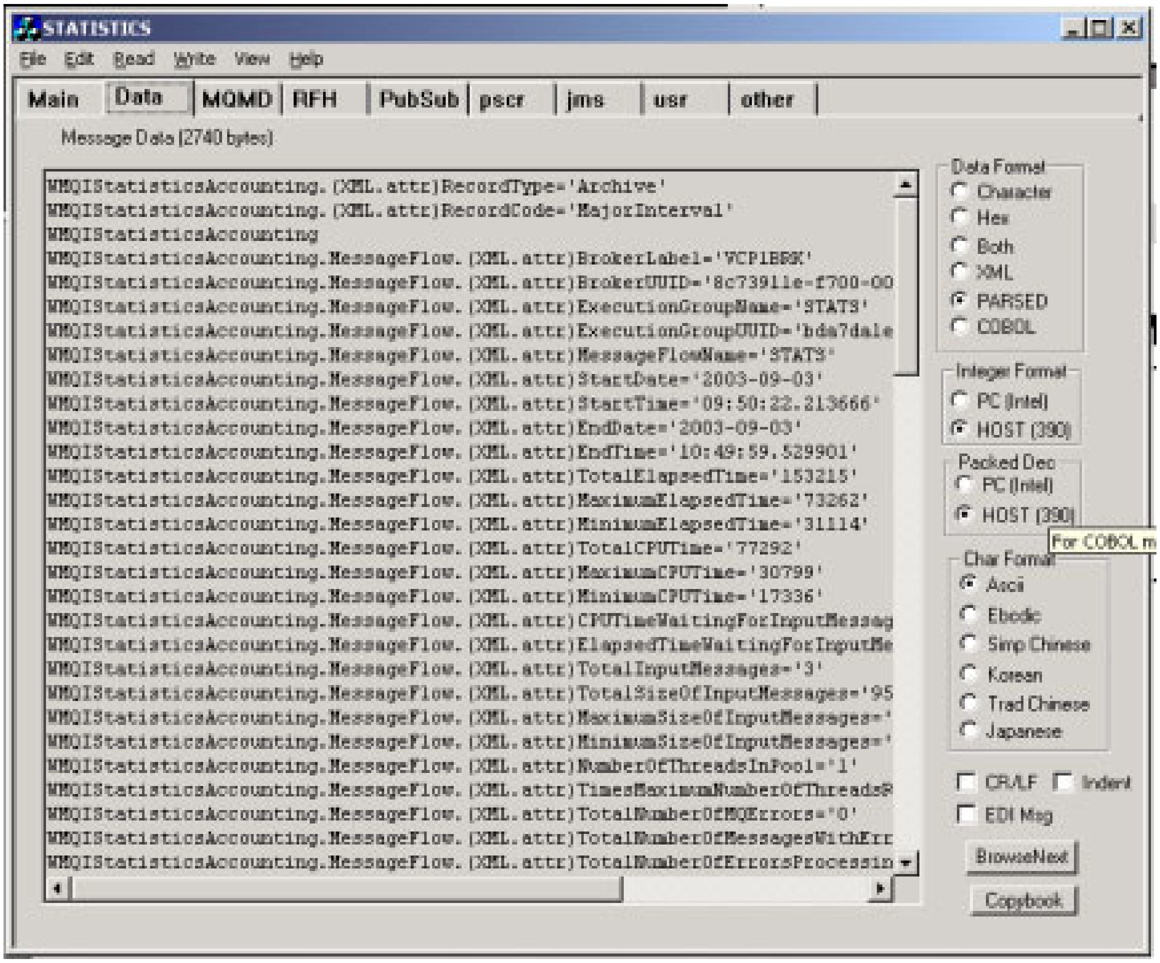
The application developer is usually interested in making changes processing some input messages, testing and evaluating the changes, and making more changes. The statistics and accounting information collected is usually only of interest until the next change and re-measure.

Typical activities include

- Checking to make sure that the average CPU time and elapsed time per message have not significantly changed
- Check the number of messages coming out of different terminals of each node, for example there should be 0 messages coming out of the failure node

The applications programmer will typically use user trace and write the output to a file. This file can then be edited and the data (in XML format) can be viewed.

If PubSub is used, and the data is sent to a queue, you can use your own applications to browse, the queue, or use the RFHUTIL program in SupportPac IH03 which has a WMQ client interface to the queue manager and allows you to browse a queue or save messages to a file.



Message Data (2740 bytes)

```
WMQIStatisticsAccounting.(XML.attr)RecordType='Archive'
WMQIStatisticsAccounting.(XML.attr)RecordCode='MajorInterval'
WMQIStatisticsAccounting
WMQIStatisticsAccounting.MessageFlow.(XML.attr)BrokerLabel='VCP1BRK'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)BrokerUUID='8c73911e-f700-00
WMQIStatisticsAccounting.MessageFlow.(XML.attr)ExecutionGroupName='STATS'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)ExecutionGroupUUID='bda7da1e
WMQIStatisticsAccounting.MessageFlow.(XML.attr)MessageFlowName='STATS'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)StartDate='2003-09-03'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)StartTime='09:50:22.213666'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)EndDate='2003-09-03'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)EndTime='10:49:59.529901'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)TotalElapsedTime='153215'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)MaximumElapsedTime='73262'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)MinimumElapsedTime='31114'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)TotalCPUTime='77292'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)MaximumCPUTime='30799'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)MinimumCPUTime='17336'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)CPUTimeWaitingForInputMessag
WMQIStatisticsAccounting.MessageFlow.(XML.attr)ElapsedTimeWaitingForInputMe
WMQIStatisticsAccounting.MessageFlow.(XML.attr)TotalInputMessages='3'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)TotalSizeOfInputMessages='95
WMQIStatisticsAccounting.MessageFlow.(XML.attr)MaximumSizeOfInputMessages='
WMQIStatisticsAccounting.MessageFlow.(XML.attr)MinimumSizeOfInputMessages='
WMQIStatisticsAccounting.MessageFlow.(XML.attr)NumberOfThreadsInPool='1'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)TimesMaximumNumberOfThreadsR
WMQIStatisticsAccounting.MessageFlow.(XML.attr)TotalNumberOfMQErrors='0'
WMQIStatisticsAccounting.MessageFlow.(XML.attr)TotalNumberOfMessagesWithErr
WMQIStatisticsAccounting.MessageFlow.(XML.attr)TotalNumberOfErrorsProcessin
```

Data Format
○ Character
○ Hex
○ Both
○ XML
● PARSED
○ COBOL

Integer Format
○ PC (Intel)
● HOST (390)

Packed Dec
○ PC (Intel)
● HOST (390)

Char Format
● Ascii
○ Ebcdic
○ Simp Chinese
○ Korean
○ Trad Chinese
○ Japanese

☐ CR/LF ☐ Indent
☐ EDI Msg

## Sample SMF print program for z/OS

The sample SMF program reads data from a sequential file.
To create the sequential file you need a job similar to that given below to copy the data from the SMF datasets

```
//PAICESM3 JOB NOTIFY=PAICE,MSGCLASS=H MSGLEVEL=(0,0)
//SMFDUMP  EXEC PGM=IFASMFDP
//DUMPINA  DD   DSN=SYS1.MV25.MANA,DISP=SHR,AMP=('BUFSP=65536')
//DUMPINB  DD   DSN=SYS1.MV25.MANB,DISP=SHR,AMP=('BUFSP=65536')
//DUMPOUT  DD   DSN=PAICE.SMFA,DISP=(NEW,CATLOG),
//      SPACE=(CYL,(100,100),RLSE)
//SYSPRINT DD   SYSOUT=*
//SYSIN  DD *
  INDD(DUMPINA,OPTIONS(DUMP))
  INDD(DUMPINB,OPTIONS(DUMP))
  OUTDD(DUMPOUT,TYPE(117))
  START(0000)
  END(2359)
/*
/&
```

This creates a data set called PAICE.SMFA.

To run the sample program you need JCL like

```
//PAICESM3 JOB '1',MSGCLASS=H,MSGLEVEL=(2,1),COND=(0,LT)
//S1 EXEC PGM=WBI117,PARM='NOTZERO'
//STEPLIB DD DISP=SHR,DSN=PAICE.MQLOAD
//SYSPRINT DD SYSOUT=*,DCB=(LRECL=132,RECFM=F)
//FLOW     DD SYSOUT=*,DCB=(LRECL=132,RECFM=F)
//NODE     DD SYSOUT=*,DCB=(LRECL=132,RECFM=F)
//TERMINAL DD SYSOUT=*,DCB=(LRECL=132,RECFM=F)
//SUMMARY  DD SYSOUT=*,DCB=(LRECL=133,RECFM=F,BLKSIZE=133)
//OUTPUT   DD SYSOUT=*,DCB=(LRECL=133,RECFM=F,BLKSIZE=133)
//THREAD   DD SYSOUT=*,DCB=(LRECL=133,RECFM=F,BLKSIZE=133)
//SYSDUMP  DD SYSOUT=*,DCB=(LRECL=133,RECFM=F,BLKSIZE=133)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=882)
//SMFIN    DD DISP=SHR,DSN=PAICE.SMFA
```

If the parameter PARM='NOTZERO' is specified, then only those flows which have processed messages are printed.  If this is omitted then all of the information is printed.

## *Explanation of output from the sample program.*

The descriptions of the fields in the SMF record apply to the fields in the XML.

### Heading

Each record has the heading information printed

```
MVS:MV25  2003233 13:16:26.68 VCP1BRK                          SAMPLE

Interval Start  YMD:2003Aug21 Week.dow:34.4 HMS:13:13:23
         End                                    13:14:06
Duration in seconds                          43
Execution group:SAMPLE
```

Where the fields are as follows:

```
MVS:MV25  2003233 13:16:26.68 VCP1BRK                          SAMPLE
```

The MVS system id is MV25

The SMF record was created on day 233 of year 2003 and time 13:16:26.68

The broker is called VCP1BRK

The execution group is called SAMPLE

```
Interval Start  YMD:2003Aug21 Week.dow:34.4 HMS:13:13:23
```

The internal date is given in different formats where Monday is dow=1;

```
End                                    13:14:06
Duration in seconds                          43
```

The duration is calculated from the start and end times.

```
Execution group:SAMPLE
```

This is the first 32 characters of the execution group name.

## Flow records

The interpretation of the fields is explained after the figure

```
MVS:MV25  2003233 13:16:26.68 VCP1BRK                          SAMPLE

Interval Start  YMD:2003Aug21 Week.dow:34.4 HMS:13:13:23
         End                                    13:14:06
Duration in seconds                          43
Execution group:SAMPLE  flow:DB2U
Number of messages                         1000
Message size                      Avg:     1484 max:    1484 min: 1484
Elapsed time procesing msgs in us Avg:    14162 max:  184552 min:
      12818
CPU     time procesing msgs in us Avg:     9764 max:   31480 min: 9540
Number of Commits                          1000
Number of Backouts                            0
Number of MQGET errors                        0
Number of MQ errors                           0
Number of error in flow                       0
Number of Agg node timeouts                   0
Accounting data................................
in hex         0000000000000000000000000000000000000000000000000000000
Elapsed time waiting for messages(sec)       29
CPU time spent waiting in uS             400081
```

Where the field names are as follows

```
Execution group:SAMPLE
```

This is the first 32 characters of the execution group name

```
flow:DB2U
```

This is the first 32 characters name of the flow within the execution group

```
Number of messages                         1000
Message size                      Avg:     1484 max:     1484 min:1484
```

Each message was the same size, so the maximum, minimum and the average as the same size.

```
Elapsed time procesing msgs in us Avg:    14162 max:   184552 min:12818
```

This is the average elapsed time in micro seconds to process a message. The message flow does a DB2 update, so there is I/O to the DB2 log data sets.
As 1000 messages were processed, and on average each message took 14162 microseconds to process, 1000 messages took 14 seconds.

```
CPU     time processing msgs in us Avg:     9764 max:    31480 min:9540
```

This is the average CPU time in micro seconds to process a message. The message flow does a DB2 update, so the CPU cost include the DB2 costs as well as costs within the Broker. The difference between the elapsed time and the CPU time, is the time doing I/O, or 14162 - 9764 or 4408 micro seconds per message.

```
Number of Commits                          1000
Number of Backouts                            0
```

If you get a backout then there will be a commit when the messages is moved to the backout or dead letter queue.

```
Number of MQGET errors                        0
```

This reflects the number of MQGET errors. For example a message has a badly formed message header.

```
Number of MQ errors                           0
```

This is the total number of WMQ errors, for example a queue is put disabled, so a put request fails.

```
Number of error in flow                       0
```

This is the number of errors which caused an exception to be thrown ( for example it displays a message on the console), for example, a request to update a non existence column in a DB2 table throws an exception.

```
Number of Agg node timeouts                   0
```

This is the numbed of times the aggregate node timed out.

```
Accounting data................................
in hex         0000000000000000000000000000000000000000000000000000000
```

This is the first 32bytes of the accounting data, if used.

```
Elapsed time waiting for messages(sec)       29
```

This is the time the execution group was waiting for work to do. The elapsed time spent waiting, plus the elapsed time spent processing messages is approximately the duration of the interval (43 seconds).

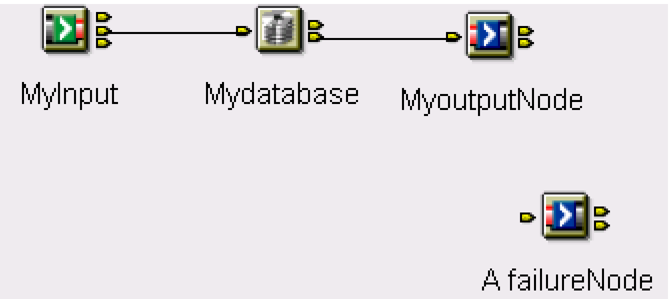
```
CPU time spent waiting in uS             400081
```

This is the amount of CPU time used when there are no messages to process. When a flow is idle, CPU is used as the input node wakes up periodically and checks to see if there are systems requests to process. These include shut down, and redeploy requests.
When the flow has processed one message, and there is another message for it to process, the MQGET request is satisfied immediately. If there is no message immediately available, the thread gets a *no message found* from the MQGET request and waits for the next message. When the next message arrives an

MQGET is issued to retrieve it.  So when there no messages immediately available at an input node there is the cost of an additional MQGET.  This contributes to the *`CPU time spent waiting in uS`* cost.

### Node records

For the execution group called SAMPLE, it has a flow called DB2U  which has four nodes within it: an input node called MyInput, a database node called Mydatabase,  an output node called MyoutputNode and an unused output node called “A failureNode”.



```
MVS:MV25  2003233 18:08:34.85 VCP1BRK
Interval Start  YMD:2003Aug21 Week.dow:34.4 HMS:18:03:19
         End                                    18:05:26
Duration in seconds                         127
Execution group                   SAMPLE
Flow                              DB2U
Node name                         A failureNode
Node Type                         MQOutputNode
Number of messages processed                  0

MVS:MV25  2003233 18:08:34.85 VCP1BRK
Interval Start  YMD:2003Aug21 Week.dow:34.4 HMS:18:03:19
         End                                    18:05:26
Duration in seconds                         127
Execution group                   SAMPLE
Flow                              DB2U
Node name                         MyInput
Node Type                         MQInputNode
Number of messages processed  1000
Elapsed time per msg in us    Avg:     2679     max:  6739  min:  2455
CPU     time per msg in us    Avg:     1839     max:  2744  min:  1812

MVS:MV25  2003233 18:08:34.85 VCP1BRK
Interval Start  YMD:2003Aug21 Week.dow:34.4 HMS:18:03:19
         End                                    18:05:26

Duration in seconds                         127
Execution group                   SAMPLE
Flow                              DB2U
Node name                         Mydatabase
Node Type                         DatabaseNode
Number of messages processed               1000
Elapsed time per msg in us        Avg:     9964 max:  37846 min:  9094
CPU     time per msg in us        Avg:     6890 max:  22938 min:  6789

MVS:MV25  2003233 18:08:34.85 VCP1BRK
Interval Start  YMD:2003Aug21 Week.dow:34.4 HMS:18:03:19
         End                                    18:05:26
Duration in seconds                         127
```

```
Execution group                  SAMPLE
Flow                             DB2U
Node name                        MyoutputNode
Node Type                        MQOutputNode
Number of messages processed              1000
Elapsed time per msg in us       Avg:     1295 max:  3168  min:  1164
CPU     time per msg in us       Avg:      899 max:  1239  min:  872
```

```
Node name                        MyInput
```

The first 32 characters of the node name is MyInput

```
Node Type                        MQInputNode
```

The node type is an MQ Input Node

```
Number of messages processed  1000
Elapsed time per msg in us    Avg:     2679     max:  6739  min:  2455
```

Within this node the average time spent in this node is 2679 micro seconds per message

```
CPU     time per msg in us    Avg:     1839     max:  2744  min:  1812
```

Within this node the average CPU time is 1839 microseconds per message.

```
Flow                             DDB2U
Node name                        Mydatabase
Node Type                        DatabaseNode
Number of messages processed              1000
Elapsed time per msg in us       Avg:     9964 max:  37846 min:  9094
CPU     time per msg in us       Avg:     6890 max:  22938 min:  6789
```

Within this node the average time spent is 9964 microseonds of which 6890 microseconds is spent using CPU.  The difference, 3072 micro seconds, is time doing I/O to the DB2 log data sets.

Note. The first messages through a flow may use more CPU and take longer, as resources have to be set up. For example

- a DB2 statement may not have been used before, so it a cached version does not exist. Subsequent messages can use the cached statement
- DB2 may have to access tables and load data into its buffer.
- ESQL is parsed on the first messages though a node.  Subsequent messages do not have this overhead.

# mqsichangeflowstats command

This section is taken from the product documentation.

### *Supported Platforms*

- Windows 2000, Windows XP
- UNIX platforms
- z/OS

### *Purpose*

Use the mqsichangeflowstats command to:

- Turn on or off accounting and statistics snapshot publication, or archive record output.
- Specify that the command be applied to a specific flow message flow, or all flows in an execution group, or all execution groups belonging to a broker.
- Modify the granularity of the data collected in addition to the standard message flow accounting and statistics. This extra data can include thread related data, node related data, node terminal related data, or a mixture of this data.

The options set using this command remain active until modified by a subsequent mqsichangeflowstats command.

### *Syntax*

### Windows platforms and UNIX platforms

```
>>-mqsichangeflowstats-- brokername --+- -a -+------------------>
                                      '- -s -'

>--+- -e --ExecutionGroupLabel-+--+- -f --MessageFlow-+--------->
   '- -g ---------------------'  '- -j -------------'

>--+------------+--+---------------+--+-------------+-------->
   '- -c control-'  '- -t ThreadData-'  '- -n NodeData-'

>--+------+--+---------------------+--+-----------------+---><
   '- -r -'  '- -b AccountingOrigin-'  '- -o OutputFormat-'
```

### z/OS

**Synonym cs**

Syntax diagram format: ○ Railroad diagram ○ Dotted decimal

Bottom of Form

```
>>-+-changeflowstats-+--+- a=yes-+------------------------------>
   '- cs ------------'  '- s=yes-'

>--+- e=ExecutionGroupLabel-+--+- f=MessageFlow-+--------------->
   '- g=yes-----------------'  '- j=yes---------'

>--+------------+--+---------------+--+-------------+----------->
   '- c=Control-'  '- t=ThreadData-'  '- n=NodeData-'

>--+--------+--+---------------------+--+-----------------+----><
   '- r=yes-'  '- b=AccountingOrigin-'  '- o=OutputFormat-'
```

### *Parameters*

**brokername**

(Required on Windows platforms and UNIX platforms) Specify the label of the broker for which accounting and statistics are to be changed.

**-a**

(Required) Specify that the command modifies archive accounting and statistics collection.

**Note:**

You must specify either **-a** or **-s**. If you do not specify one of these arguments you receive a warning message.

**-s**

(Optional) Specify that the command modifies snapshot accounting and statistics collection.

**Note:**

You must specify either **-a** or **-s**. If you do not specify one of these arguments you receive a warning message.

**-e *ExecutionGroupLabel***

(Required) Specify the label for the execution group, for which accounting and statistics options are to be changed.

**Note:**

You must specify either **-e** or **-g**, or both. If you specify both, **-g** takes precedence.

**-g**

(Required) Specifies that the command applies to **all** execution groups that belong to the broker

**Note:**

You must specify either **-e** or **-g**, or both. If you specify both, **-g** takes precedence.

**-f *MessageFlow***

(Required) Specify the label for the message flow, for which accounting and statistics options are to be changed.

**Note:**

You must specify either **-f** or **-j**, or both. If you specify both, **-j** takes precedence.

**-j**

(Required) Specifies that the command applies to **all** message flows that belong to the execution group.

**Notes:**

1. You must specify either **-f** or **-j**, or both. If you specify both, **-j** takes precedence.
2. If you set the **-g** option for all execution groups, you must use **-j** instead of **-f**.

**-c** ***control***

(Optional) Specify the string value that controls the level of the action to be applied to accounting and statistics collection for snapshot or archiving. Possible values are:

- active - turn on snapshot or archiving
- inactive - turn off snapshot or archiving.

**-t** ***ThreadData***

(Optional) Specify a string value to modify the collection of thread statistics data for a message flow Possible values are:

- none - exclude thread related data from the statistics
- basic - include thread related data in the statistics

**-n** ***NodeData***

(Optional) Specify a string value to modify the collection of node statistics data for a message flow. Possible values are:

- none - exclude node related data in the statistics
- basic - include node related statistics in the statistics
- advanced - include node related and terminal related data in the statistics

**-r**

(Optional) Specify that a reset of archive data is required.

**Note:**

This action is only valid for archive data.

This results in the clearing out of accounting and statistics data accumulated so far for this interval, and restarts collection from this point. All archive data for all flows in the execution group, or groups, is reset.

The archive interval timer is only reset if the **-v** option (*statistics archive interval*) of mqsicreatebroker or mqsichangebroker is non zero. That is , the interval timer is only set if the internal interval notification mechanism of WebSphere MQ Integrator Broker is being used , and not an external method for example, ENF on z/OS.

**-b** ***AccountingOrigin***

(Optional) Specifies that the environment tree path *Broker.Accounting.Origin* is used to partition the collected statistics into distinct outputs. Possible values are:

- none - do not partition statistics according to accounting origin data
- basic - partition statistics according to accounting origin data

**-o** ***OutputFormat***

(Optional) Specify the output destination for the statistics reports. Possible values are:

- usertrace - this is the default and writes "bip" messages to usertrace, which can be post processed in the normal way using the mqsireadlog and mqsiformatlog commands
- xml - the statistics reports are generated as XML documents and published by the broker running the message flow.
- smf - (z/OS only). Statistics reports are output as SMF type 117 records.

### Authorization

The user Id used to issue the command must have mqbrkrs authority.

### Responses

This command returns the following responses:

- BIP2226 Request to change attribute in message flow node ' ': message flow does not exist
- BIP8004 Invalid flags and arguments selected
- BIP8013 Component does not exist
- BIP8020 Unable to access the database
- BIP8029 Broker not configured
- BIP8033 Unable to send XML message
- BIP8038 Unsupported command option
- BIP8039 Execution group not available
- BIP8040 Unable to connect to database

### Examples

Turn on snapshot statistics for the message flow "myFlow1" in all execution groups of BrokerA and specify that the data is to be gathered by accounting origin:

```
mqsichangeflowstats BrokerA -s -g -j -b none
```

Turn off the collection of archive statistics for message flow "MyFlow1" in execution group "EGRP2" for BrokerA, and at the same time modify the granularity of data that is to be collected ( when next activated ) to include thread related data.

```
 mqsichangeflowstats BrokerA -a -e "EGRP2" -f MyFlow1 -c inactive -t basic
```

Turn off snapshot data for all message flows in all execution groups for Broker A.

```
mqsichangeflowstats BrokerA -s -g -j -c inactive
```

# Appendices

## *DB2 table definitions*

### DDL for the Flowstats table

```
CREATE TABLE "FLOWSTATS"  (
                    "RECORDTYPE" CHAR(8) ,
                    "RECORDCODE" CHAR(24) ,
                    "BROKER" VARCHAR(36) ,
                    "EXECGROUP" VARCHAR(36) ,
                    "FLOW" VARCHAR(36) ,
                    "STARTDATE" DATE ,
                    "STARTTIME" TIME ,
                    "ENDDATE" DATE ,
                    "ENDTIME" TIME ,
                    "DURATION" INTEGER ,
                    "MESSAGES" INTEGER ,
                    "TOTET_IN_US" INTEGER ,
                    "MAXET_IN_US" INTEGER ,
                    "MINET_IN_US" INTEGER ,
                    "TOTCT_IN_US" INTEGER ,
                    "MAXCT_IN_US" INTEGER ,
                    "MINCT_IN_US" INTEGER ,
                    "WAITET_IN_S"INTEGER ,
                    "WAITCT_IN_US" INTEGER ,
                    "SUMCT_IN_US" INTEGER ,
                    "TOTMSGSIZE" INTEGER ,
                    "MAXMSGSIZE" INTEGER ,
                    "MINMSGSIZE" INTEGER ,
                    "NUMTHREAD" INTEGER ,
                    "ATMAXTHREAD" INTEGER ,
                    "NUMMQERR" INTEGER ,
                    "NUMMSGERR" INTEGER ,
                    "NUMERR" INTEGER ,
                    "NUMAGGTO" INTEGER ,
                    "NUMCOMMITS" INTEGER ,
                    "NUMBACKOUT" INTEGER ,
                    "ACCOUNTINGORIGIN" VARCHAR(36),
                    "BROKERUUID" VARCHAR(36) ,
                    "EGROUPUUID" VARCHAR(36) )
                   IN "USERSPACE1" ;
```

### DDL for the Nodestats table

```
CREATE TABLE "NODESTATS"  (
                    "RECORDTYPE" CHAR(8) ,
                    "RECORDCODE" CHAR(24) ,
                    "BROKER" VARCHAR(36) ,
                    "EXECGROUP" VARCHAR(36) ,
                    "FLOW" VARCHAR(36) ,
                    "STARTDATE" DATE ,
                    "STARTTIME" TIME ,
                    "ENDDATE" DATE ,
                    "ENDTIME" TIME ,
                    "DURATION" INTEGER ,
                    "NODE" VARCHAR(36) ,
                    "NODETYPE" VARCHAR(36) ,
                    "TOTET_IN_US" INTEGER ,
                    "MAXET_IN_US" INTEGER ,
                    "MINET_IN_US" INTEGER ,
                    "TOTCT_IN_US" INTEGER ,
                    "MAXCT_IN_US" INTEGER ,
                    "MINCT_IN_US" INTEGER ,
                    "TIMESUSED" integer,
                    "NINPUTT" INTEGER ,
                    "NOUTPUTT" INTEGER ,
                    "BROKERUUID" VARCHAR(36) ,
                    "EGROUPUUID" VARCHAR(36) )
                   IN "USERSPACE1" ;
```

## DDL for the Theadstats table

```
CREATE TABLE "THREADSTATS"  (
                     "RECORDTYPE" CHAR(8) ,
                     "RECORDCODE" CHAR(24) ,
                     "BROKER" VARCHAR(36) ,
                     "EXECGROUP" VARCHAR(36) ,
                     "FLOW" VARCHAR(36) ,
                     "STARTDATE" DATE ,
                     "STARTTIME" TIME ,
                     "ENDDATE" DATE ,
                     "ENDTIME" TIME ,
                     "DURATION" INTEGER ,
                     "MESSAGES" INTEGER ,
                     "THREADS" INTEGER ,
                     "THREAD" INTEGER ,
                     "MSGET_IN_US" INTEGER ,
                     "MSGCT_IN_US" INTEGER ,
                     "WAITET_IN_S" INTEGER ,
                     "WAITCT_IN_US" INTEGER ,
                     "SUMCT_IN_US" INTEGER ,
                     "TOTALBYTES" DOUBLE ,
                     "MAXMSGSIZE" INTEGER ,
                     "MINMSGSIZE" INTEGER ,
                     "BROKERUUID" VARCHAR(36) ,
                     "EGROUPUUID" VARCHAR(36) )
                    IN "USERSPACE1" ;
```

## DDL for the Termstats table

```
CREATE TABLE "TERMSTATS"  (
                     "RECORDTYPE" CHAR(8) ,
                     "RECORDCODE" CHAR(24) ,
                     "BROKER" VARCHAR(36) ,
                     "EXECGROUP" VARCHAR(36) ,
                     "FLOW" VARCHAR(36) ,
                     "STARTDATE" DATE ,
                     "STARTTIME" TIME ,
                     "ENDDATE" DATE ,
                     "ENDTIME" TIME ,
                     "DURATION" INTEGER ,
                     "NODE" VARCHAR(36) ,
                     "NODETYPE" VARCHAR(36) ,
                     "TERMLABEL" VARCHAR(36),
                     "TERMTYPE" CHAR(6) ,
                     "TIMESUSED" INTEGER ,
                     "BROKERUUID" VARCHAR(36) ,
                     "EGROUPUUID" VARCHAR(36) )
                    IN "USERSPACE1" ;
```

# DB2 definitions for the aw_flowstats view

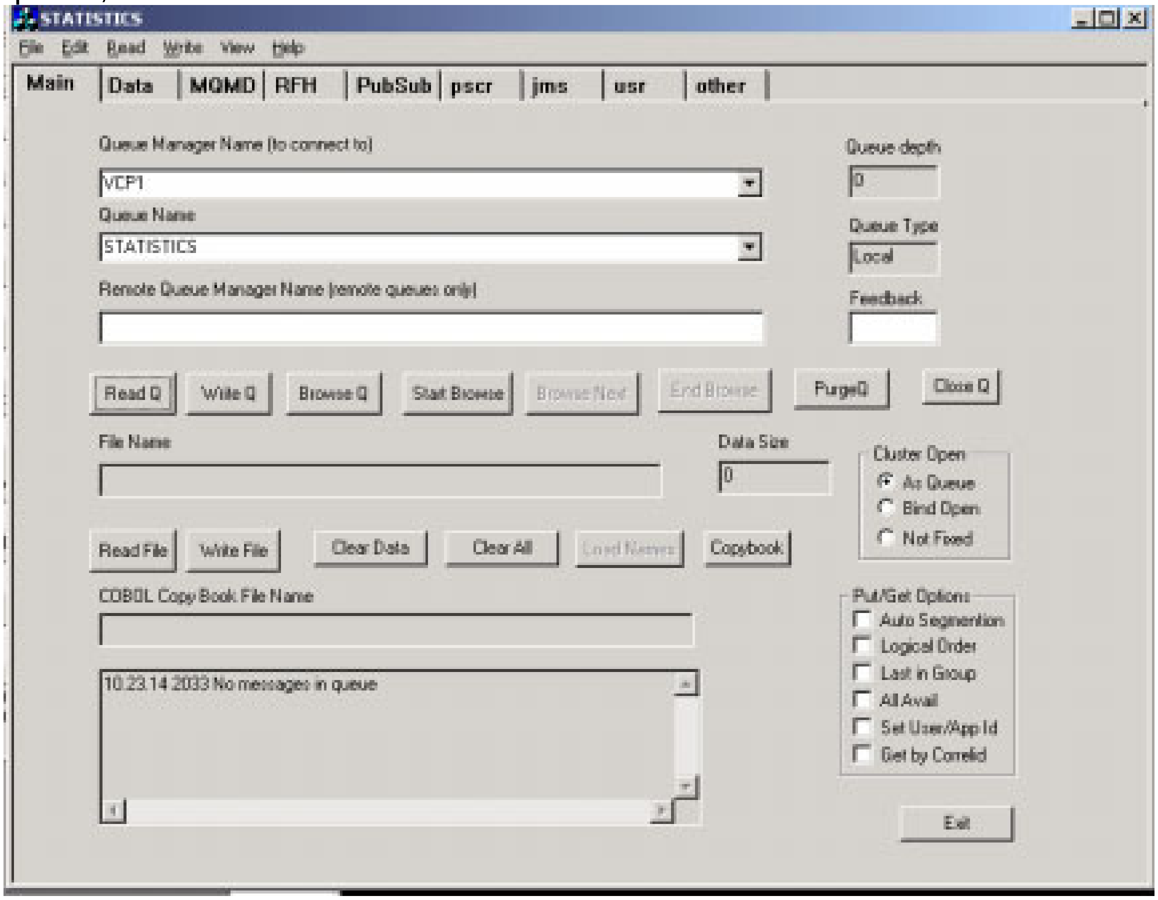
```
CREATE VIEW AW_FLOWSTATS(
   "BROKER"
 , "EXECGROUP"
 , "FLOW"
 , "STARTWEEK"
 , "DURATION"
 , "TOTET in Secs"
 , "MAXET in uSecs"
 , "MINET in uSecs"
 , "AVGET in uSecs"
 , "SUMCT in Secs"
 , "AVGCT in uSecs"
 , "TOTCT in Secs"
 , "MAXCT in uSecs"
 , "MINCT in uSecs"
 , "WAITET in Secs"
 , "WAITCT in Secs"
 , "MESSAGES"
 , "EGROUPUUID")
as
Select
   "BROKER"
 , "EXECGROUP"
 , "FLOW"
 , startdate +1 day  - dayofweek(startdate) days
 , SUM("DURATION")
 , round(sum(float(totet_in_us))/1000000,0)
 , MAX(MAXET_IN_US)
 , MIN(case when messages > 0 then minet_in_us else null end)
 , case when sum(messages) = 0 then 0 else round(sum(float(totet_in_us))/sum(messages),0) end
 , round(sum(float(sumct_in_us))/1000000,0)
 , case when sum(messages) = 0 then 0 else round(sum(float(totct_in_us))/sum(messages),0) end
 , round(sum(float(sumct_in_us))/1000000,0)
 , MAX(MAXCT_in_us)
 , MIN(case when messages > 0 then minct_In_us else null end)
 , sum(waitet_in_s)
 , round( sum(float(WAITCT_IN_US)) /1000000,0)
 , SUM(MESSAGES)
 , EGROUPUUID

  from flowSTATS
  where RECORDTYPE = 'Archive'
  group by Broker,EXECGROUP,FLOW,EGROUPUUID,startdate +1 day  - dayofweek(startdate) days ;
```

See the provided files for the other views.

## How to subscribe to a statistics and accounting data

One of the easiest ways of subscribing to statistics and accounting data is to use the RFHUTIL program in the SupportPac IH03. This provides a GUI interface which allows you to issue subscriptions, to read from queues, and to write to queues, through the WMQ client interface.

1. Define a queue on the host queue manager to receive the statistics, for example STATISTICS
2. Install the SupportPac and go to the directory
3. Define the client connection, for example using the MQSERVER environment variable SET MQSERVER=SYSTEM.DEF.SVRCONN/TCP/9.20.129.2(2181)
4. Run the RFHUTILC program.
5. Put the host queue manager name into the queue manager name field
6. Put the name of the host queue into the Queue name field
7. Press the Read Queue Button
8. In a window at the bottom it should display a message 'no messages in queue', as show below



Queue Manager Name (to connect to): VCP1

Queue Name: STATISTICS

Remote Queue Manager Name (remote queues only):

Queue depth: 0

Queue Type: Local

Feedback:

File Name:

Data Size: 0

Cluster Open

● As Queue

○ Bind Open

○ Not Fixed

COBOL Copy Book File Name:

10.23.14.2033 No messages in queue

Put/Get Options

□ Auto Segmention

□ Logical Order

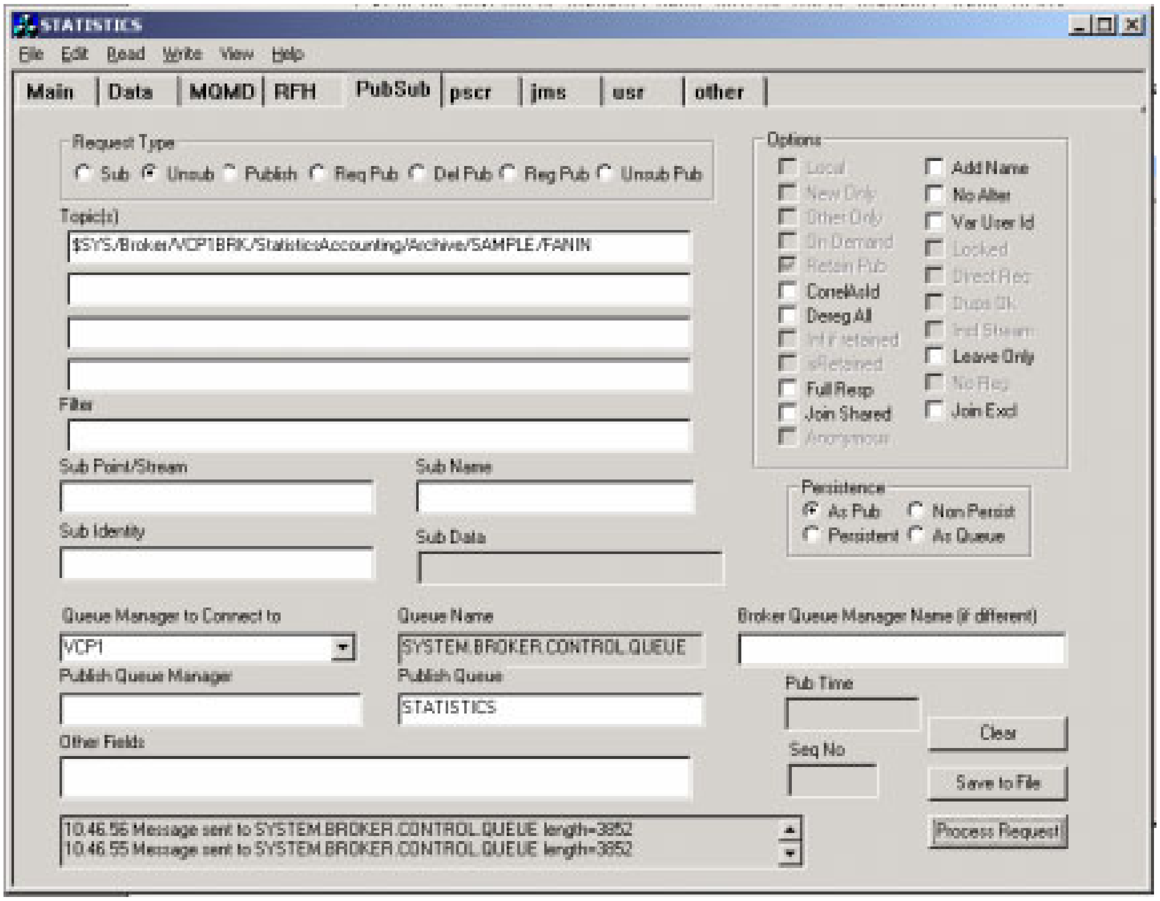
□ Last in Group

□ All Avail

□ Set User/App Id

□ Get by Correlid

9. |Click on the PubSub tab, fill in the details and press the process request button



Request Type: ○ Sub ● Unsub ○ Publish ○ Req Pub ○ Del Pub ○ Reg Pub ○ Unsub Pub

Topic(s): $SYS/Broker/VCP1BRK/StatisticsAccounting/Archive/SAMPLE/FANIN

Options:
☐ Local
☐ New Only
☐ Other Only
☐ On Demand
☑ Retain Pub
☐ CorrelAsId
☐ Dereg All
☐ Inf if retained
☐ isRetained
☐ Full Resp
☐ Join Shared
☐ Anonymous
☐ Add Name
☐ No Alter
☐ Var User Id
☐ Locked
☐ Direct Req
☐ Dups Ok
☐ Incl Stream
☐ Leave Only
☐ No Reg
☐ Join Excl

Persistence: ● As Pub ○ Non Persist ○ Persistent ○ As Queue

Queue Manager to Connect to: VCP1

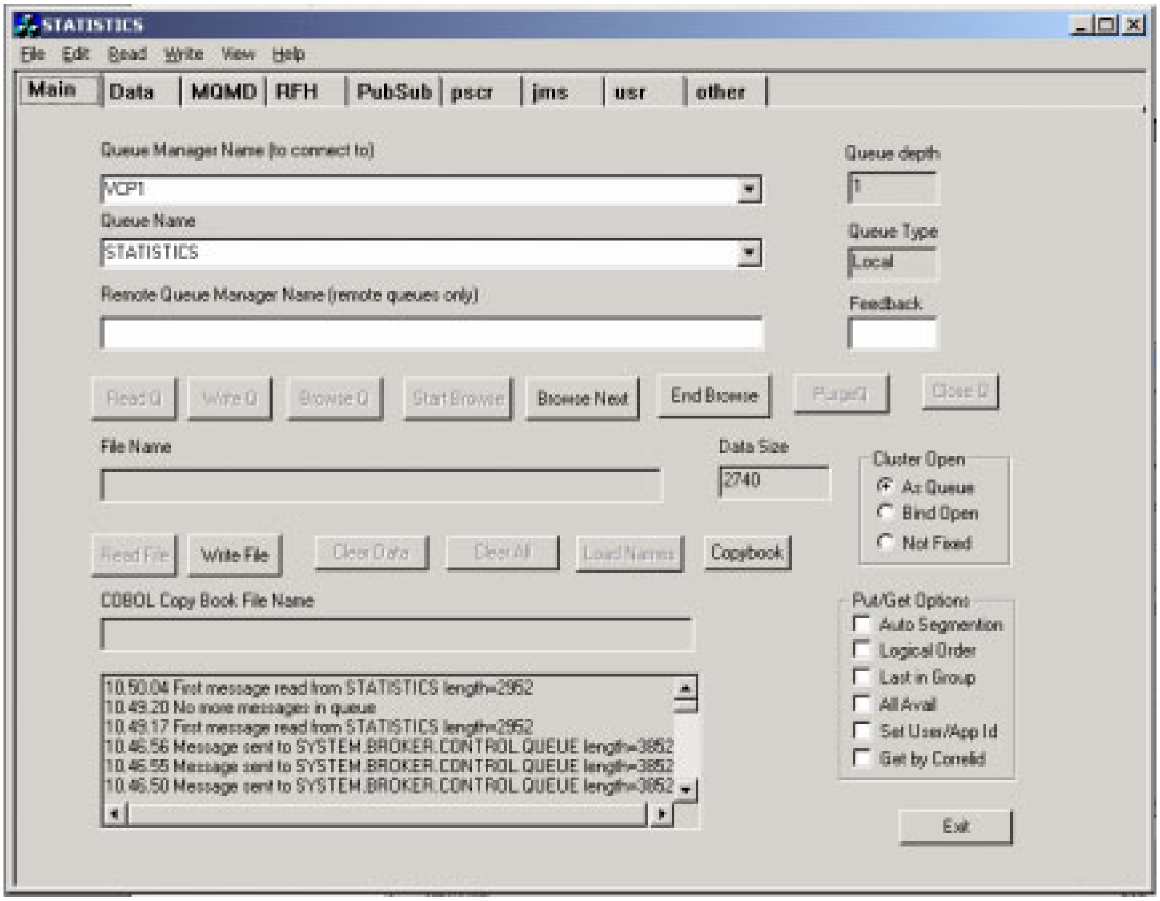
Queue Name: SYSTEM.BROKER.CONTROL.QUEUE

Publish Queue: STATISTICS

```
10.46.56 Message sent to SYSTEM.BROKER.CONTROL.QUEUE length=3852
10.46.55 Message sent to SYSTEM.BROKER.CONTROL.QUEUE length=3852
```

10. Click the main tab and enter your statistics queue in the queue name.

11. Press the Start Browse button, if you have data returned, press the data tab to see the data.



Queue Manager Name (to connect to): VCP1

Queue Name: STATISTICS

Remote Queue Manager Name (remote queues only):

Queue depth: 1

Queue Type: Local

Feedback:

File Name:

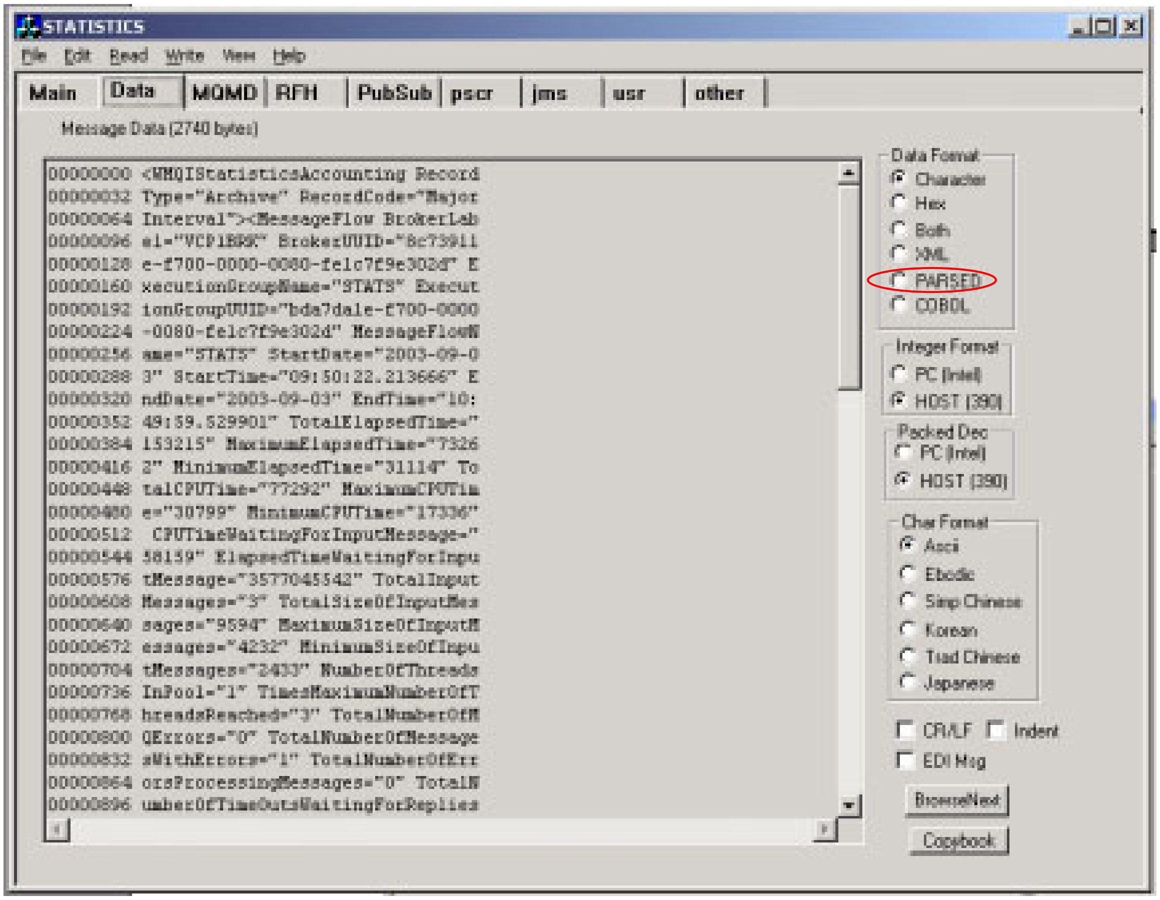
Data Size: 2740

Cluster Open
● As Queue
○ Bind Open
○ Not Fixed

COBOL Copy Book File Name:

Put/Get Options
□ Auto Segmention
□ Logical Order
□ Last in Group
□ All Avail
□ Set User/App Id
□ Get by Correlid

```
10.50.04 First message read from STATISTICS length=2952
10.49.20 No more messages in queue
10.49.17 First message read from STATISTICS length=2952
10.46.56 Message sent to SYSTEM.BROKER.CONTROL.QUEUE length=3852
10.46.55 Message sent to SYSTEM.BROKER.CONTROL.QUEUE length=3852
10.46.50 Message sent to SYSTEM.BROKER.CONTROL.QUEUE length=3852
```



Message Data (2740 bytes)

```
00000000 <WMQIStatisticsAccounting Record
00000032 Type="Archive" RecordCode="Major
00000064 Interval"><MessageFlow BrokerLab
00000096 el="VCP1BRK" BrokerUUID="8c73911
00000128 e-f700-0000-0080-fe1c7f9e302d" E
00000160 xecutionGroupName="STATS" Execut
00000192 ionGroupUUID="bda7da1e-f700-0000
00000224 -0080-fe1c7f9e302d" MessageFlowN
00000256 ame="STATS" StartDate="2003-09-0
00000288 3" StartTime="09:50:22.213666" E
00000320 ndDate="2003-09-03" EndTime="10:
00000352 49:59.529901" TotalElapsedTime="
00000384 153215" MaximumElapsedTime="7326
00000416 2" MinimumElapsedTime="31114" To
00000448 talCPUTime="77292" MaximumCPUTim
00000480 e="30799" MinimumCPUTime="17336"
00000512  CPUTimeWaitingForInputMessage="
00000544 58159" ElapsedTimeWaitingForInpu
00000576 tMessage="3577045542" TotalInput
00000608 Messages="3" TotalSizeOfInputMes
00000640 sages="9594" MaximumSizeOfInputM
00000672 essages="4232" MinimumSizeOfInpu
00000704 tMessages="2433" NumberOfThreads
00000736 InPool="1" TimesMaximumNumberOfT
00000768 hreadsReached="3" TotalNumberOfM
00000800 QErrors="0" TotalNumberOfMessage
00000832 sWithErrors="1" TotalNumberOfErr
00000864 orsProcessingMessages="0" TotalN
00000896 umberOfTimeOutsWaitingForReplies
```

Data Format
● Character
○ Hex
○ Both
○ XML
○ PARSED
○ COBOL

Integer Format
○ PC (Intel)
● HOST (390)

Packed Dec
○ PC (Intel)
● HOST (390)

Char Format
● Ascii
○ Ebcdic
○ Simp Chinese
○ Korean
○ Trad Chinese
○ Japanese

□ CR/LF □ Indent
□ EDI Msg

.

Click the PARSED button to have the XML formatted, as shown on page 28.

(end of document)