



MQSoftware, Inc.

1660 South Highway 100

Suite 400

Minneapolis, MN 55416

952-345-8720 • 1-888-228-1366

Product Brief

Tracking the Rogue Message with StatWatch

Written by:

Mr. Robert Andresen, Systems Engineer

Tracking the Rogue Message with StatWatch

EXECUTIVE SUMMARY

IT organizations today encompass a wide range of technologies and employ a diverse staff of experts to support them. Our enterprises depend on these business systems to be functioning at peak capacity to remain profitable. Service outages and problems may require Sarbanes-Oxley reporting to avoid fines and legal proceedings.

Clearly, there needs to be a way to monitor and troubleshoot the complete system as a whole. Far too many times each individual technologist is convinced their piece of the corporate infrastructure and applications are working perfectly, even though the entire business system is experiencing an outage or a slow down. In these situations it takes far too long to resolve the problem, all the while the company is hemorrhaging cash.

What is needed is a tool, which transcends technology silos, and tracks corporate application transactions across the enterprise. When a slow down or failure occurs it is critical to quickly identify where the problem is. Too many organizations adopt a war room approach, getting all the technologists together in a room to argue about whose infrastructure is causing the problem. While this may eventually address the problem, it takes time and costs money. A tool that tracks and times transactions across all the applications in a business process speeds this process up considerably. It can identify immediately where the problem is, allowing the right technology group to identify and correct the problem. Assigning a problem to the wrong technology group is no better than not detecting the problem in the first place.

INTRODUCTION:

Many organizations are using IBM's Websphere MQ as the transport layer between applications. These applications run on different platforms using disparate technologies. Each technology group has a set of monitors and tools to measure the health and performance of their application, but organizations typically do not have a way of measuring a business transaction, made up of multiple single purpose applications. If messages are delayed or not processed, the business transaction fails, even though no problems are apparent at any of the component applications.

Many organizations have service level agreements with their clients, but have no way of tracking end-to-end response time. Some organizations run synthetic transactions and monitor their response times. This not only adds non-productive overhead, but is not an accurate measure, as the synthetic transactions generally take the same logic path through every application every time, and read the same database records every time. The real transactions can be experiencing database buffer problems or application logic problems which don't show up for the synthetic transaction.

Other organizations settle for monitoring each component of the business transaction, and assume if each component appears to be working, the entire transaction is working. This approach may not detect misrouted messages. Furthermore, adding the average response time for each component of a business transaction does not accurately predict the total response time for a single business transaction.

Finally, some organizations code their own monitoring logic into their applications. This complicates the application, however, and mixes business logic with infrastructure management logic. Business logic may be in constant flux based on competition and other market conditions. It is better to separate

infrastructure monitoring from business logic so changes become less complicated and easier to test and verify.

THE PROBLEM:

After applications have been running for a while it frequently happens no one in the IT organization has a clear idea of how some MQ applications are structured. Whether because of turnover, poor or lost documentation, or mergers and acquisitions, many shops are not sure how messages flow through their different Queue Managers. This can make supporting those applications difficult. If you don't know how an application is supposed to be working it becomes very difficult to recognize when it is having problems.

Even after the path messages take through different Queue Managers is understood, it now becomes important to measure the time it takes for messages to traverse these paths. You need know how long it takes for messages to travel through Websphere MQ and be serviced by your different applications under normal circumstances, when the systems are performing as expected. Then you can recognize problems when response is slow, whether due to infrastructure problems or workload starting to exceed current capacity.

Being able to replace poorly performing applications is great, but it depends on being able to identify the response for all the applications in a business process. This becomes more difficult and very expensive since each application can be running on a different platform running a different operating system written in a different language. To monitor each application on its native environment would require many different system-monitoring utilities, each of which will work differently and report different metrics. The response numbers may end up comparing apples with oranges. What is needed is a single methodology to measure response from every application in the same way.

THE SOLUTION:

StatWatch, from MQSoftware, Inc. provides the solution. StatWatch will record part or all of Websphere MQ messages as they flow across the enterprise. It then provides analytical reports, which will document where the messages went, and how long they took to get there. This allows us to see the response time of any application by seeing the latency of the amount of time an application took to reply to a request.

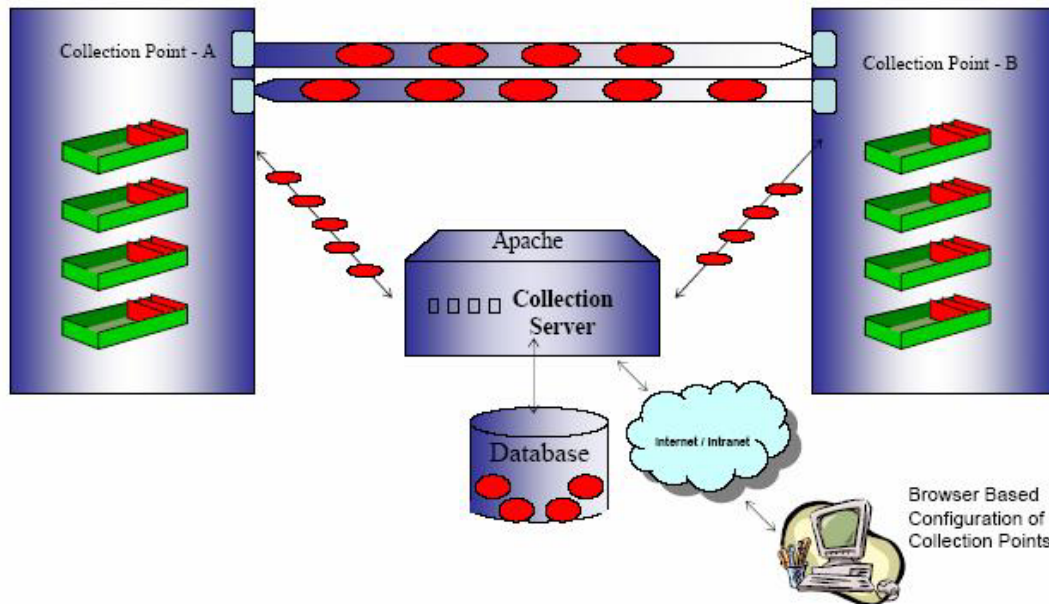
StatWatch makes no changes to the applications and runs with very little overhead. By using it instead of writing Service Level Agreement (SLA) monitoring into the logic of your application, you are able to simplify the code you write, concentrating only on the business logic. Furthermore, StatWatch is monitoring real production transactions, not synthetic transactions. If there is a failure or slowdown anywhere in the monitored transaction, StatWatch will detect it.

StatWatch OVERVIEW:

StatWatch can be used as an investigative tool to discover the paths different messages are taking through all the different Queue Managers and measure the time it takes. StatWatch is installed on a server and is comprised of three services: a collection agent, an Apache server and a Tomcat application server. It connects via MQ client connection or by proxy to the Websphere MQ Queue Managers it monitors. It uses both an API crossing exit as well as a channel exit, depending on the kind of data you wish to collect.

StatWatch runs as a web application with a configurable port. It accesses its database for both administrative and reporting functions. The database can be DB2 or Oracle, and the StatWatch server can run on Windows, AIX or Solaris. It saves message header information and up to 4,000 bytes of the message in its database.

Once StatWatch is configured, you use the same browser interface to run reports against the data it has collected in its database. The reports provide summary, graphical and detailed options. You may also choose to download the report data as a csv file on your workstation for further analysis with other tools, such as a spreadsheet.

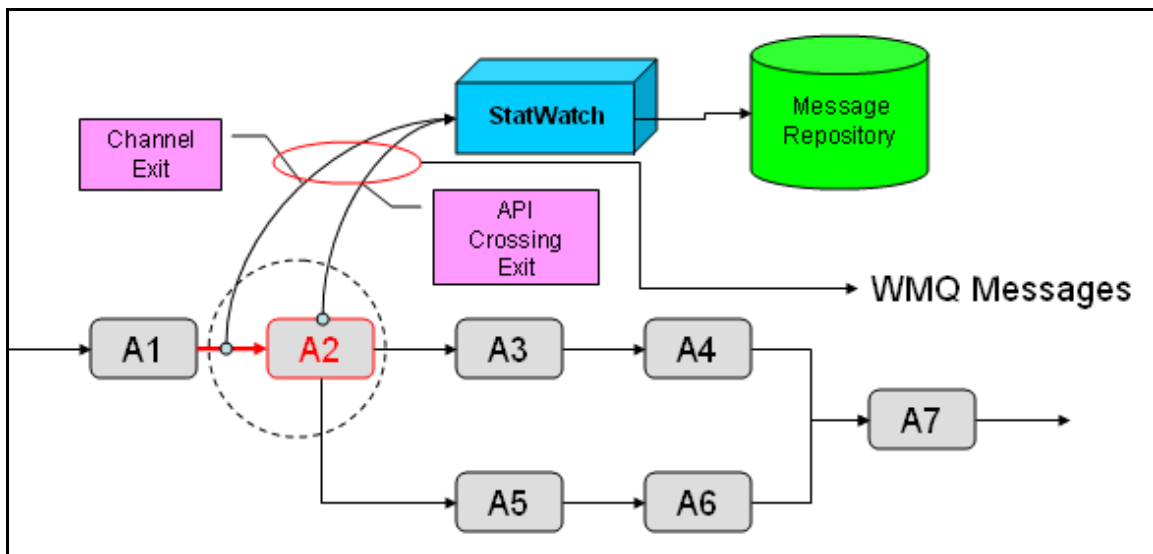


StatWatch SETUP

The first step is to define the Queue Managers to StatWatch where we want to monitor message flows. This is done with a simple GUI panel where we specify how StatWatch will connect to the Queue Manager.

Our choices are either directly or via proxy through another Queue Manager. A direct connection will require Websphere MQ exits be deployed and a server connection channel and listener port be identified. For a proxy connection, we will need to specify the name of the proxy Queue Manager. For either connection we will need to specify the command input and response queues, though system defaults may be used.

Next we will deploy our monitoring exit to the channels we suspect the messages may be flowing over. We pick the channels we want to watch for messages. The important choice for tracking the message flow would be to select the Audit Messages option, which will save a copy of each message coming across the channel in the StatWatch database. For our current purpose, defining message flow, we would only need to select sender channels. If we wanted to measure network transit time we could pick both sender and receiver channels. If we wanted to see message counts and channel starts and stops we could select those options.



Here we see StatWatch is collecting messages on the channel from A1 to A2 using its channel exit, and is collecting messages on a queue on A2 using its API crossing exit. The messages are stored in the StatWatch database for later analysis by the reporting function.

We would repeat these two steps for every Queue Manager and every channel we think our messages may flow through. At this point it will be better to err by picking more channels than we need rather than missing a potentially critical message flow. Next we kick off the message flows, or wait for them to run.

RUNNING StatWatch REPORTS

Once the application we are researching has run at least once we are ready to analyze where the messages flowed. The first challenge is we may have trapped other message flows other than the one we were after. If other applications share the same channels this will almost certainly be the case. This is an easy problem to overcome.

First we will use the “Throughput Report” in StatWatch. This report is a detailed log of all messages that have flowed through our Websphere MQ environment. When we request this or any other report, we are presented with the set of filters.

Our goal here is to find at least one of the messages that made up the application we are researching. If there is some criteria which will narrow down the list of messages to single pot our application, we can enter it here. For example, giving the application or user identifier of the flow will exclude messages belonging to other applications or users.

We are looking to see the actual messages that flowed through the system, so after we enter our filter conditions we would click the “html” button. This will bring up a navigable list of all the messages.

The list of messages is a table we can scroll to the right or left to see other columns, such as message id, message length, message format and application. There are also links to go to other pages of the reports to see other messages. We can drill down on a specific message to see all of its detail.

Once we find one of the messages of the flow we are researching, we will copy its message id into the clipboard. Next we will select the message tracking report from the StatWatch menu. The Message Tracking report organizes all the messages into transactions using message identifier.

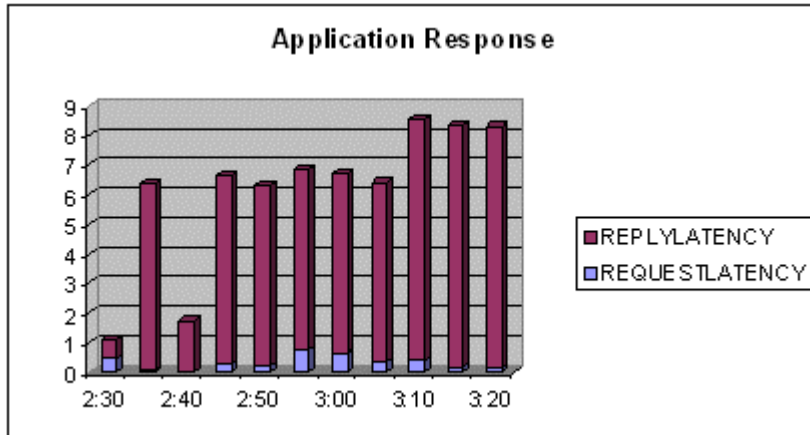
We paste in the message id we saved from the message we just found and once again click the “html” button. The result shows us every message that went through our environment with the same message id, showing every Queue Manager and queue the message went through. The message latency column shows us how long the message took each step along the path it took. We can use the StatWatch database as a performance repository to compare response times of our Websphere MQ applications at different times to recognize bottlenecks and slow downs.

But this is just showing us the message traveling in a single direction. If we pick the Request Reply report we can see all the messages with the original message id and all the replies with the original message id in their correlation id field. This will show us total response time for our application, from the time of the initial request until the time of the response.

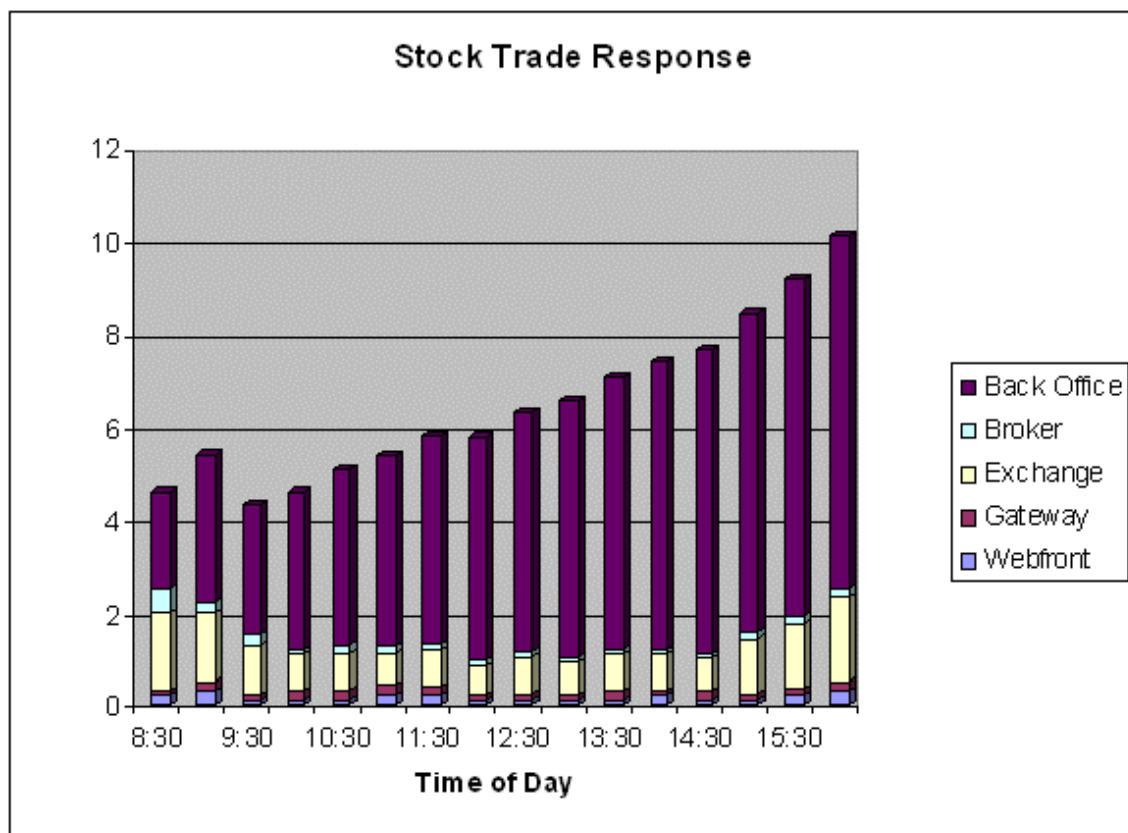
We would pick the Request Reply Report from the StatWatch menu and would be presented with a filter panel. We would once again paste our original message id into the message id field, and click on the “html” button as before.

The report starts out with the fields from the request message on the left. If we use the scroll bar to scroll right we see the reply latency column giving the total application response time from the time of the original request. The correlation id is also in this display. Either of the message id or correlation id fields can be used to drill down to the complete message detail as shown back in the message tracking report.

Perhaps more useful is the option to save the response data in a .csv file. We can use a spreadsheet to open the file and build a graph of the response time data. Here is a graph of request latency and reply latency for an application over time:



Another option is to take the response time data from multiple applications in a business process to build a graph of composite response time of each application in the business process. Here we have a stacked graph of the components of a stock trading system:



We see the critical applications of our stock trade business application (excluding the back office accounting application) run under two seconds, except for an hour

at market open and the half hour before market close. Further analysis shows the Exchange application response time has grown during those instances. Now we know which application to investigate. If we make changes to the Exchange application, we now have a benchmark to compare current performance.

StatWatch BENEFITS

We have taken an unfamiliar Websphere MQ application and mapped out the message flow including the replies to the original messages. We now see the latency of the message at each hop along the path it takes through the system. We have a record of the response time for every application in the business transaction. It is quite a simple task to see which application has the slowest response time. If the business process is not performing as well as it needs to, we know which application needs attention and which technology group should be involved.

It is easy to turn StatWatch collection on and off. It does not require changes to the monitored applications. Since StatWatch is using the same measurements of message response time across the entire enterprise, it is a fair and consistent measure of how well each application performs. StatWatch is not dependant on the internals of disparate systems monitors running on different platforms and different operating systems.

CONCLUSION

StatWatch provides a tool to monitor any production business transaction using Websphere MQ as a backbone. By selecting where we trap messages on their way through business transactions we can isolate the performance of each application that makes up our business transaction as well as map out the route the messages are taking. This tells us if we have an overall response time problem, warning us we are about to fail an SLA while we can still do something about it.

Monitoring the messages as they go from Queue Manager to Queue Manager on their way from application to application will identify the poorest performing application causing the SLA compliance problem, letting you immediately route the problem to the correct group to fix it. As the saying goes; you can only manage what you measure.