72

MQ update

June 2005

In this issue

A itc PUBLICATION

MQ Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, scripts, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *MQ Update*, comprising twelve monthly issues, costs $380.00 in the USA and Canada; £255.00 in the UK; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the July 2000 issue, are available separately to subscribers for $33.75 (£22.50) each including postage.

Contributions

When Xephon is given copyright, articles published in *MQ Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

MQ Update on-line

Code from *MQ Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at www.xephon.com/mq; you will need to supply a word from the printed issue.

Printed in England.

Exploring MQSeries report options

As we increasingly use assured-delivery, asynchronous, messaging middleware, eg IBM MQSeries, it is vitally important for the applications using the asynchronous protocol to know about the status of their messages/requests. Because of the asynchronous nature of the product, most application programmers, MQ system administrators, and support personnel have had to trace their messages – ie they have spent time finding out where their message is.

MQSeries doesn't leave you blindfolded; it has alternatives in terms of report options, which, when used effectively, allow applications to be kept updated about the status of their message's arrival and delivery, and whether it processed successfully or unsuccessfully, and they can be notified when an error or exception occurs.

Report options are one of the least-exploited valuable MQSeries features. With the aid of various report options, error-sensitive mission-critical applications can be made aware of error situations across the MQSeries network. Application programs can act intelligently in response to the report and feedback received from the interfacing system, thereby unleashing closed loop application systems that can overcome error situations by themselves.

Let us begin by exploring the MQSeries report options using a common analogy – a postal mail system. We know that our postal staff have been doing a good job delivering the mail. But at times when mail delivery becomes critical, the postal agency offers various kinds of acknowledgement services to suit our needs. There are various classes of this service:

1 Acknowledgement sent by the post office when mail reaches the destination city's post office.

2 Acknowledgement sent by the post office when mail is delivered to the intended recipient, with/without the signature of the person it was delivered to.

3 Let us say my mail is a request for $500 from a friend. My friend then decides to transfer the money (an unlikely event!). He may notify me of the good news.

4 More likely, he may notify me with a one liner – goodbye.

5 If the mail can't be delivered, for whatever reason, the post office may notify me with an acknowledgement or the piece of mail itself, stating the reason for the delivery failure.

In MQSeries report terms, an acknowledgement sent by the receiving post office (queue manager) when mail arrives is equivalent to COA (Confirmation On Arrival). Acknowledgement of the delivery is equivalent to COD (Confirmation On Delivery). My friend's (application) notification of good news is PAN (Positive Action Notification). When he couldn't send money, this is equivalent to NAN (Negative Action Notification). Finally, notification received when mail couldn't be delivered is synonymous with the MQSeries Exception and Expiration options.

Let us look at the details of how the report option is specified, used, and identified. MQSeries report options are attributes of a message and are specified in the report field of the message descriptor (MQMD) by the sending application. If set, a message should have a destination address – Reply to queue and Reply to queue manager. When report messages are generated they are specifically decoded with the message format of MQFMT_REPORT to uniquely identify and distinguish them from application messages, and they will be generated either by the queue manager itself (post office) or the servicing application (my friend!).

While requesting report messages, applications are given some flexibility over the data that they would want to see in the report message, which can be either zero bytes of original data (empty message), the first 100 bytes of the original data, or the complete message itself. Also, by default, the normal request/reply protocol is employed to facilitate the correlation

of the generated reports to the request, by copying the message id of the request to the correlation identifier of the report message – this behaviour can also be altered when requesting the report option.

Applications can also seek/request a multiple report option for a single message (for instance COA + Exception). Programmatically this is done by adding (or BIT ORed) the respective constants that denote these report options. And while requesting an Exception report (when the message couldn't be delivered because the remote site queue was full, **put** disabled, etc), the application can specify where the original message should end up – by default the original message will be delivered to a dead letter queue, but it can be changed to discard the message.

Now let's look into the properties of the report message. When the report message is generated, the type of report (COA or COD etc) is given by the feedback field in the MQMD of the message, which can be interrogated by the application to decide on its future course of action. Also, most of the attributes of a message (the MQMD fields) that cause the reports are copied to the report messages. Important things to note are:

- Report messages will have an unlimited expiry irrespective of the original message.
- The identity context of the report message will be set according to the MQPMO_PASS_IDENTITY_CONTEXT option.
- The origin context will have the updated information based on the source and time that the report was generated.

Having said that, here is a small code snippet that will enable an existing application to turn on the report option. The following code requests COA reports to be generated with full data, exceptions with no data, and expirations with 100 bytes of the data.

```
MQMD.Report           = MQRO_COA_WITH_FULL_DATA +
                        MQRO_EXCEPTION +
                        MQRO_EXPIRATION_WITH_DATA;
MQMD.ReplyToQueue     = 'MY.REPORT.QUEUE';
MQMD.ReplyToQmgr      = '';
```

And while processing the generated reported messages from MY.REPORT.QUEUE, an application can determine the course of action by looking at the Feedback field:

```
Switch (MQMD.Feedback)

Case MQFB_COA:
          /* Write a code that update the status of the request sent */

case MQFB_EXPIRATION :
          /* Write a remedy action code to resend the message again */

case MQRC_Q_FULL:
          /* When a report message is generated due to exception, */
          /* feedback code corresponds to mq reason code for */
          /* failure to deliver       */
 /*Write a sleep routine that will retry processing after an interval.*/

case MQRC_PUT_INHIBITED :
case MQRC_NOT_AUTHORIZED :
          /* Write a code that alerts the MQSeries admin about  */
          /* queue set-up and configuration issues */

end switch ; .
```

Notes:

- Report options are generated only if the applications cannot be synchronously notified of problems.
- Although receipt of a report confirms the status of the message, failure to receive a report message doesn't mean that the original message never reached its destination or that it hasn't been delivered to the application. It may be that the report message is being held on one of the queues, for whatever reason.
- COA and COD reports options are honoured only if the message reaches the intended destination queue. If the message is delivered to a dead letter queue, or is on its

way through XMIT queues, neither COA nor COD is generated.

- The user identifier field of the MQMD causing the report message to be produced should have the necessary MQ authorization on the reply-to queue for the report message to make it back to the requestor. Exception reports use the receiving MCA user identifier determined by the PutAuthority channel attribute for authorization, and, in cases where the MCA user identifier is used to **put** the message, COA is generated by the same user identifier.

Common usage:

1 Error-sensitive applications can sense the error effectively and they have the opportunity to work on their recovery or remedy.

For instance, if the remote end queue is **put** disabled or not given the required access, by making use of the EXCEPTION report options in the request message, an application will sense the failure at the first available opportunity.

This more or less enables the request applications to see what problems are happening on the remote system – as if the application itself were running on the other end, sensing the **put** failures, etc.

2 Can be used as an inbuilt, effective debugging technique along with the Feedback field:

- The requestor program, when run in debug mode, will set the Feedback code and report options (for instance whether the entire message has to be captured on arrival) and turn on the COA with full data. That way, irrespective of the responder reading the request message, a copy of the message is still available in the designated reply-to queue that can be made available at the requestor end or reply end with the help of reply-to queue aliasing.

- Likewise, the responder program can run in debug mode on demand, on a message-by-message basis, dynamically when receiving messages with certain Feedback codes. As you can see with the effective use of Feedback fields and Report options between requests and reply programs, we can build a closed-loop system that can respond to the needs of the responder in a dynamic fashion.

3 An application that requires coordination and mutual agreement before starting work can use report options to signal its status.

Consider a situation where a request program is sending a batch of messages that will be inserted into the database at the remote end. When the database is taken down for maintenance (if the remote application is triggered, irrespective of turning off triggering it doesn't prevent the queueing up of messages) we can use report options to signal the status of the application using either of these techniques or others:

- When the remote database is taken down for maintenance, **put** disable the local queue; now the application, prior to sending a batch of messages, will send an exception report message and interrogate the Feedback on the report message. If it's **put** disabled, it will retry the operation after an interval.
- The above approach doesn't involve any coding on the remote application, but where we have permission to modify the code application we can generate a PAN or NAN for a control message issued by the sender prior to the start of batch. A control message will be identified by a special Feedback code on the request message that the remote application should honour.

4 Applications that have message affinity and/or are trying to exploit MQSeries clustering for high availability (on Unix platforms where we don't have the concept of queue

sharing to recover from application failures) can use report options to achieve the same thing – to an extent.

Instead of binding to one of the available cluster queue destinations selected by the queue manager at open time and staying with it, the application can turn on report options on the messages to determine a destination (ReplytoQueue manager field in the report message will have a cluster queue manager that responds to it) that responds with the report message and binds on to it. When that particular destination is not available, sensed by the receipt (remember non-receipt of a report message cannot be assumed as a failure) of a report message, it can be used to switch to the next destinations.

Ideally this enables even the request application sending a message to a cluster queue to behave like a reply application that can use the knowledge of the destination whence the request originated (picked from ReplyToQmgr) to reply just to that destination – bypassing the workload balancing/round-robin algorithm used by clustering.

5 Applications that require acknowledgement from their peers for the work that is being carried out.

All in all, to summarize, report options enable the application to be self tuning, robust, and able to act dynamically based on the feedback they receive. They are very easy to code and are turned on by modifications to message header fields.

Ramasubbu (Ram) Ramanathan
MQSeries Integration Specialist (USA)

Going further beyond event-based MQ monitoring

In the first article, entitled *Going beyond event-based MQ monitoring* (*MQ Update*, Issue 70, April 2005), a new approach to defining, generating, and correlating events across an

integrated business operation was introduced. This new approach was a response to the need for enhanced business value and agility from the often highly-distributed, multi-sourced, integrated IT systems of today. The fundamental aspects of this approach were essentially fourfold:

- Collect appropriate IT metrics and events.
- Correlate complex multi-sourced events across disparate technologies.
- Combine this information with business knowledge to increase its value.
- Dynamically create new events to predict as well as detect problems.

This article follows the subject into more detail, outlining how these factors can be achieved and the resulting benefits in some real-world scenarios. It reinforces the fact that moving to a more business-oriented end-to-end approach turns the acknowledged benefits of monitoring into the more extensive benefits of business operations management, enabling enterprises continually to improve the effectiveness and agility of their operations. The aim is to achieve a solution where end-to-end operational events and real-time metrics can be placed in a business context and delivered to the appropriate teams in the optimal fashion.

DOING THE GROUNDWORK

In order to go beyond traditional event-based monitoring in today's integrated environments, it is necessary to be able to bring together information from a variety of different systems. The first article identified the problem of component-based monitoring tools that are blind beyond the boundaries of the particular component in question. In order to achieve the benefits described above, it is necessary to assemble event information from all the key components in use within the business service and correlate it to gain an overall picture.

This is achieved by offering various ‘plug-in’ capabilities for the central MQ management component. For example, a customer might have a combination of Oracle, WebSphere, and custom-developed applications, integrated through MQ or JMS and combining to deliver a particular business service such as order processing. Although knowledge of the performance and behaviour of MQ will obviously be important in judging the effectiveness of the service operation, events in the other components may also impact the service. Therefore the management tool must provide mechanisms to bring key information in from these components. This can be achieved in various ways. For Java applications such as in the WebSphere environment, JMX instrumentation allows the information to be brought in with a high degree of automation. In the case of Oracle, a plug-in might be provided to interrogate Oracle system information; while for custom-developed applications, an API might be offered. Finally, it will usually be necessary also to offer some sort of SDK to allow users to create their own customized mechanisms for gathering the relevant event information in specialized circumstances.

An important factor to consider here is that, as the sources of information increase, it becomes even more important to provide some level of filtering across the end-to-end operation. Correlating the information from the multiple components needs to take into account the nature of information and its impacts to ensure that the user is not completely swamped by a surfeit of events and alerts. This correlation becomes easier when the events can be understood in a business context (an area that will be considered later).

Once the information is made available to the management tool, it can now reflect a correlated picture of events, alerts, and operational behaviour to the user on an end-to-end basis. An important concept here is that of the business service – that is, the particular business operation being carried out. Defining this business service enables the management tool to know which components form the end-to-end service, and therefore makes it possible to relate operational behaviour at the IT component level to the business operation in question.

DEFINING BUSINESS SERVICES AND KEY BUSINESS INDICATORS

In order to effectively reflect the performance and behaviour of business operations, the business service concept brings together the IT and business aspects of each business operation. It follows a model approach, where the business service identifies the IT components related to the business unit of work, such as applications, technologies, and transactions, and the business rules related to them. From a management point of view, an important part of this definition is the specification of the Key Business Indicators (KBIs). These describe technology aspects that reflect key aspects of business operations for the particular service. Examples of a KBI might be 'Trades executed per minute' or '% of failed funds transfers'. These are the metrics that the management tool will use to determine the overall health and effectiveness of operations.

As well as identifying the IT components and KBIs, the business service also records the business rules that apply to the service. The rule might specify that if errors from a particular sales terminal exceed a pre-defined rate, then that terminal should be placed out of service. Or it might require that a supervisor be informed if end user response time drops below a certain level.

CREATING AN EARLY-WARNING SYSTEM

The combination of being able to gather IT behavioural information from multiple different technologies and environments, understanding how these relate to the business service in terms of its KBIs, and knowing business rules that apply, makes it possible to deliver a much more business-effective monitoring solution. Instead of alerts simply reflecting an IT occurrence, they identify a business impact, and corresponding actions can be dictated by business rather than IT needs. The result is an operational IT infrastructure that is much more closely aligned to business needs.

Using this approach, problems impacting the business can be detected to allow corrective action to be taken. Of course, this corrective action may involve any or all of the components involved in the business service since the event is based on information gleaned from multiple different environments, and the management tool needs to support this. But this is all about problem detection and correction, in other words tolerating the problem by identifying and fixing it as quickly as possible.

In fact, more and more companies are looking to management technology to go a step further. Rather than focusing on detecting and resolving business problems, companies want to avoid the problems in the first place. They are looking for an early-warning system that can see the problem starting to materialize and thereby allow pre-emptive action to be taken. This has led to increasing levels of attention being paid to the area of analytics and how it can be used to support this goal.

Monitoring tools are in an ideal position to take advantage of analytics to deliver an early warning system. By tracking events in both technical and business terms with a business service, the tool can start to build up an historical picture of the behaviour of the operation and its KBIs. Using analytics techniques, it now becomes possible to form predictive conclusions based on a series of events. Whereas static events are usually based on a particular value being exceeded or an occurrence, these predictive judgments are usually based instead on comparative factors and trends. This brings the element of time into the calculation and therefore allows a situation to be extrapolated into the future.

So, experience may show that when KBI 1 increases from the norm by more than 20%, there is a 90% probability that KBI 2 will fall by at least 30%. Therefore, when the increase in KBI 1 reaches 20%, action can be taken to address the effect on KBI 2 before it actually occurs. An example might be that when the volume of order entry exceeds the norm by more than 20%, the error rate might typically climb by 30%. Therefore an

extra supervisory step could be introduced to validate order details before back-end processing.

The mechanism for presenting this information to the user is of critical importance if the information is to be acted on efficiently and effectively. The key is to make the information about the behaviour of a business service and any associated impacts available in such a way that both business and technical users can visualize it in a meaningful fashion, in some sort of 'business dashboard'. The business context is just as important as technical drill-down options. It is within this business dashboard that the user might be able to click on a KBI impact, for example, and be presented with warnings of additional impacts that are likely to result from the initial problem. This allows pre-emptive action to be taken locally, or other teams to be alerted, if appropriate, to warn them of problems building up in their own areas, thus avoiding any new problems materializing.

This ability to start to predict situations that will impact the business based on past performance patterns has immense value in terms of business efficiency and effectiveness. In essence, it enables the system to learn from its mistakes and get smarter.

REAL-LIFE SCENARIO

The points made in this article are illustrated in the following real-life scenario. This particular company in the financial services industry was concerned with the business service provided by an ATM-type application for supporting a network of automated teller machines. The end-to-end flow to be covered in the business service ran from the teller machine through a number of stages to the back-end system and back again. The company was already monitoring MQ events across this flow, but wanted to go further.

The first action was to define the business service model for this particular operation. From an IT perspective this was

straightforward, involving the identification of the applications, transactions, and technologies used as part of the execution flow. Plug-ins were put in place to ensure that information could be drawn together across the whole end-to-end operation, with filtering to focus attention on the most important events and issues.

However, dealing with the business aspects of the service definition required a little more work. Key Business Indicators (KBIs) had to be identified and defined, such as the volume of transactions being processed from the teller machines, how many suffered delays, and for how long. Once the KBIs were selected, rules were put in place, such as generating an alert when more than 1% of transactions take longer than 5 seconds to complete. This alert would contain the relevant business information and be presented to the enterprise systems management frameworks, in this case Tivoli and HP/OpenView.

The business service could now be defined, encompassing the IT components involved, the KBIs, and the rules. This new method of operation had a number of effects. First, alerts and associated rules and procedures were based on business factors and impacts rather than the underlying technical events. This presented a more relevant picture of business performance, and allowed events to be handled with the appropriate priority from a business rather than a technical perspective. Secondly, as new business requirements came in, the technology layer covering the instrumentation and its processing no longer had to be the focus. Rather, the rules within the service became the target for adjustments and modifications, making it much easier to address new requirements and enhancing overall agility.

But beyond this, the company was now able to start watching trends based on historical information collected over a period of general operations. These trends allowed an early warning system to be established based on the rate of change of certain KBIs, for example. Alerts could be raised, for instance, to inform the enterprise console that the overall teller system

was likely to fail in two hours if action was not taken in the interim.

So, this company was able to realize the following benefits by going beyond traditional event-based monitoring:

- End-to-end visibility of business service behaviour and performance.
- Reduced time spent on handling 'unimportant' alerts and events.
- Prioritization of problems causing the biggest business impact.
- Ability to address new business requirements more quickly and easily.
- Enhanced and continually improving service levels.

CONCLUSIONS

As IT systems become more integrated, the need for proper tracking and instrumentation increases dramatically, but the biggest benefits come from moving beyond this traditional, event-based management approach to one that takes full account of the business context of these IT events. By concentrating on business services, it becomes possible to build a far more effective operational system where IT can be more closely aligned to the objectives of the business. The focus can move from monitoring events to managing key business indicators with the appropriate set of business-defined rules, where action can be prioritized to the point of greatest business need. And finally, historic information from IT components can be associated with the related behaviour of these key business indicators to deduce trends, and these can in turn be used to predict and hence prevent disruptions to service, ensuring that operational behaviour is continuously improved. All of this results in helping to deliver operational business systems that are effective, efficient, and agile.

David Mavashev
CEO
Nastel Technologies Inc (USA)

WebSphere MQ security for z/OS

Man does not want to be free. He simply wants to be safe.

– H L Menchen.

WHY WORRY ABOUT SECURITY?

More and more people are on the Internet (the good, the bad, and the ugly). Information is a valuable asset for our national security. In the computer age, information has become the lifeblood of many companies. Failure to safeguard information as you would your home or other assets is ludicrous, and a security breach will cost time, money, data, your customers, your reputation, and your business. Lack of security leads to decreased productivity (denial of service, sabotage) and theft of ideas (research data, proprietary information).

In short, you can't afford not to worry about it. In this article I will discuss security for z/OS WebSphere MQ.

SECURITY IS A PROCESS

Security is a process, a process that involves protection, detection, and action. Enabling Secure Sockets Layer V3 (SSL) on a WebSphere MQ channel is prevention. Having an alert raised when a digital certificate proves that a message has been tampered with is detection. Shutting down the channel and calling in the police is action. Protection by itself is pointless, and although it will deter many attackers, it will probably attract (more competent) others. Without processes in place to detect and take remedial action to respond to such attacks, an enterprise lacks a complete solution.

WHAT YOU NEED TO KNOW ABOUT SECURITY

You need to know the risks and liabilities of your business. The

threats to your company's information are real and can range from reasonably innocent errors and glitches to malicious hacking. Just as you purchase virus-protection software to prevent and solve problems *before* there is damage to your data and system, you need to proactively invest in assuring the security of your data. In a WebSphere MQ environment, several different platforms communicate application commands and your company's critical information through message queue managers. Since messaging is the transmission of technology (data, e-mail, strings, objects) from one system to another across machines or platforms, the 'message' becomes the object that must be secured.

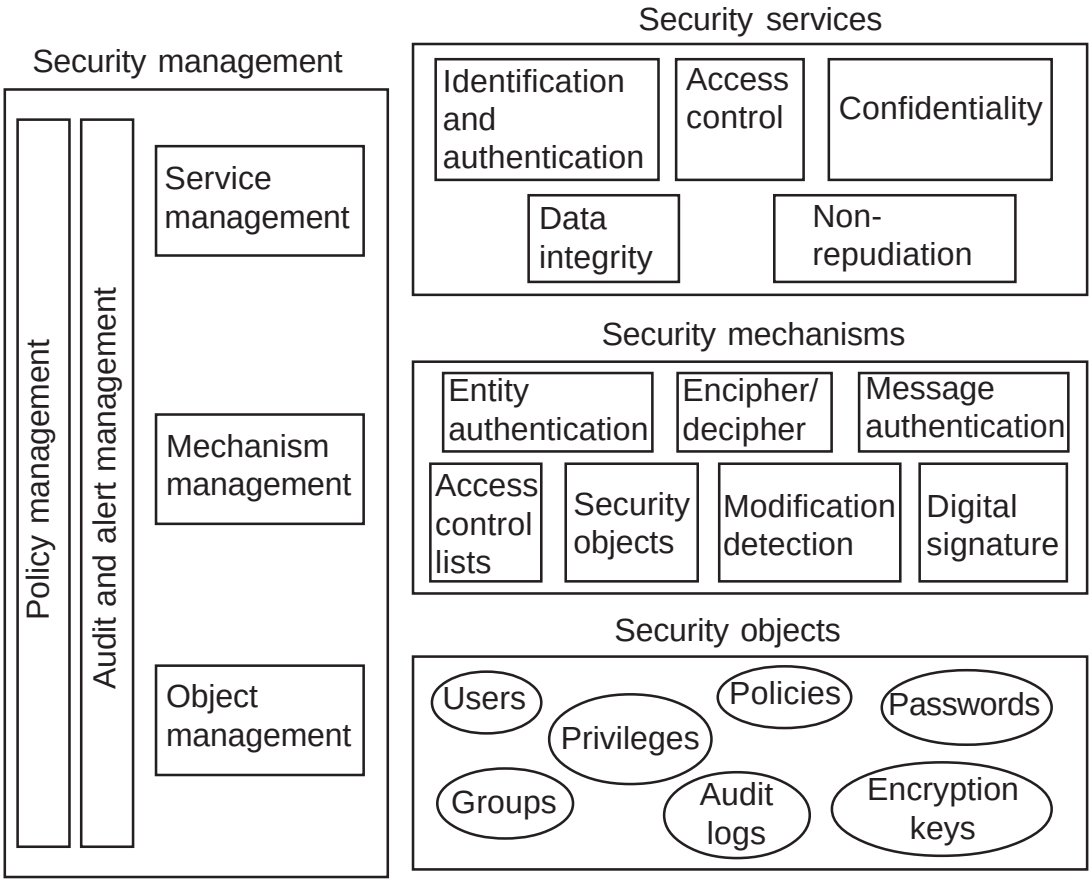

Figure 1: Security facilities available

THE SECURITY ENVIRONMENT

Figure 1 illustrates the set of security facilities that might be available in a secured environment.

Figure 1 also shows the relative positioning of the various security facilities. For example, security services make use of security mechanisms, which, in turn, act on security objects.

SECURITY SERVICES

Security services are the services within a computer system that protect its resources. The five security services that are identified in the IBM Security Architecture are:

- Identification and authentication
- Access control
- Confidentiality
- Data integrity
- Non-repudiation.

Identification and authentication

Identification is being able to identify uniquely a user of a system or an application that is running on the system. Authentication is being able to prove that a user or application is genuinely who or what that person or application claims to be.

Here are some examples of the identification and authentication service in a WebSphere MQ environment:

- Every message can contain message context information. This information is held in the message descriptor and can be generated by the queue manager when a message is put on a queue by an application. Alternatively, the application can supply the information if the user ID associated with the application is authorized to do so.

The context information in a message allows the receiving application to find out about the originator of the message. It contains, for example, the name of the application that put the message and the user ID associated with the application.

- When a message channel starts, it is possible for the Message Channel Agent (MCA) at each end of the channel to authenticate its partner. This is known as mutual authentication. For the sending MCA, this provides assurance that the partner it is about to send messages to is genuine. And, for the receiving MCA, there is a similar assurance that it is about to receive messages from a genuine partner.

Access control

The access control service protects critical resources in a system by limiting access to only authorized users and their applications. It prevents the unauthorized use of a resource or the use of a resource in an unauthorized manner. Here are some examples of the access control service in a WebSphere MQ environment:

- Allowing only an authorized administrator to issue commands to manage WebSphere MQ resources.
- Allowing an application to connect to a queue manager only if the user ID associated with the application is authorized to do so.
- Allowing a user's application to open only those queues that are necessary for its function.
- Allowing a user's application to perform only those operations on a queue that are necessary for its function. For example, an application might need only to browse messages on a particular queue, and not to **put** or **get** messages.

Confidentiality

The confidentiality service protects sensitive information from unauthorized disclosure.

Here are some examples of the confidentiality service that can be implemented in a WebSphere MQ environment:

- After a sending MCA gets a message from a transmission queue, the message is encrypted before it is sent over the network to the receiving MCA. At the other end of the channel, the message is decrypted before the receiving MCA puts it on its destination queue.
- While messages are stored on a local queue, the access control mechanisms provided by WebSphere MQ might be considered sufficient to protect their contents against unauthorized disclosure. However, for a greater level of security, their contents can be encrypted as well.

Data integrity

The data integrity service detects whether there has been unauthorized modification of data. That is all it does; it does not then aim to restore data to its original state. Access control mechanisms can contribute to data integrity insofar as data cannot be modified if access is denied. But, as with confidentiality, access control mechanisms are not effective in a networking environment.

Here are some examples of the data integrity service that can be implemented in a WebSphere MQ environment:

- A data integrity service can be used to detect whether the contents of a message have been deliberately modified while the message was being transmitted over a network.
- While messages are stored on a local queue, the access control mechanisms provided by WebSphere MQ might be considered sufficient to prevent deliberate modification of the contents of the messages. However, for a greater level of security, a data integrity service can be used to

detect whether the contents of a message have been deliberately modified between the time the message was put on the queue and the time it was retrieved from the queue.

Non-repudiation

The non-repudiation service can be viewed as an extension to the identification and authentication service. The overall goal is to be able to prove that a particular message is associated with a particular individual. The non-repudiation service can contain more than one component, where each component provides a different function. If the sender of a message ever denies sending it, the non-repudiation service with proof of origin can provide the receiver with undeniable evidence that the message was sent by that particular individual. If the receiver of a message ever denies receiving it, the non-repudiation service with proof of delivery can provide the sender with undeniable evidence that that particular individual received the message. Non-repudiation is a relevant security service in a WebSphere MQ environment because WebSphere MQ is a means of transmitting data electronically. For example, you might require contemporaneous evidence that a particular message was sent or received by an application associated with a particular individual.

WEBSPHERE MQ SECURITY FOR Z/OS

WebSphere MQ for z/OS uses the z/OS System Authorization Facility (SAF) to provide access control services within the z/OS environment. This is the standard mechanism of providing security in a z/OS environment and has two significant advantages. First, there is a security manager for the entire z/OS environment; secondly, there is a choice of External (to WebSphere MQ) Security Manager (ESM), providing greater flexibility. The main ESMs used by z/OS users are Resource Access Control Facility (RACF), Top Secret, and Access Control Facility (ACF2). The ESM used by WebSphere MQ for z/OS is RACF.

ACLs are implemented as a set of profiles. A user is granted access to a particular profile to allow access to the protected resource. To contain the profiles, WebSphere MQ has a set of RACF classes for its use.

These classes are:

- MQADMIN – contains profiles for administration functions, command resources, context, and alternative user profiles.
- MQCONN – contains profiles to limit connection to the MQ subsystem.
- MQCMDS – contains profiles for command security.
- MQQUEUE – controls queue resources.
- MQPROC – controls process resources.
- MQNLIST – controls namelist resources.

These lists are defined to the RACF system and need to be defined to other security manager products as well.

SWITCH PROFILES

Switch profiles are RACF profiles that control the level of security checking carried out by WebSphere MQ. The switch profiles allow a granular control of security checking within WebSphere MQ for z/OS. These switch profiles are all defined in the MQADMIN class. If the MQADMIN class is not present or not activated (or there is no external security manager) then security checking is deactivated. The use of switch profiles is extended in WebSphere MQ for z/OS V5.2 to support the use of queue sharing groups and shared queues.

The different switches represent the different components of WebSphere MQ to which access control checks may be applied and are split into three areas:

- Connecting to the queue manager

- MQ API
- MQ commands.

Connection security

If connection security is active, you must define profiles in the MQCONN class and permit the necessary user IDs' access to those profiles so they can connect to the queue managers. The profiles needed in the MQCONN class are:

- hlq.BATCH
- hlq.CICS
- hlq.IMS
- hlq.CHIN.

There is one profile per adapter type, which provides granular control for each environment. READ access is required by the WebSphere MQ adapter user ID connecting to the queue manager in each environment:

- Queue manager only checking – if you have the hlq of a queue managername, the profile controls access to the related queue manager for that connection type.
- Queue sharing group only checking – if you have the hlq of a queue sharing group name, the profile controls access to all queue managers within the queue-sharing group for that connection type.

API security

API security is controlled by several profiles in different RACF classes:

- MQQUEUE class
- MQADMIN class
- MQPROC class
- MQNLIST.

Access control checks for the WebSphere MQ API are performed when a queue is opened (MQOPEN or MQPUT1) and (for permanent dynamic queues) when a queue is closed (and deleted). Because of the different ways in which a queue may be opened, there are different access control checks, and different access levels are required. WebSphere MQ provides five different profiles, depending on the object being opened and the access required. There may also be several checks performed if a queue is being opened in a specific way; if a dynamic queue is being created, alternative user IDs are to be used and MQMD context fields are to be accessed.

Command and command resource security

There is a separate profile for each WebSphere MQ command (the verb) and target (the primary keyword), allowing each command to be controlled individually. Thus, a particular user ID may be able to define **qlocals** but not **qremotes** or it may be able to display queues but not define them. It is also possible to control access to the resources accessed by these commands.

WHAT HAS CHANGED IN WEBSPHERE MQ FOR Z/OS V5.2 SECURITY?

Use of DB2

Shared queue managers connecting to DB2 will have connection checking performed by DB2. Use existing security facilities in DB2 to set this up. The shared queue manager address space user ID will be used for this connection check.

Use of Coupling Facility (CF)

In order for a shared queue manager to have access to structures in the CF, each queue manager address space user ID must have ALTER access to the relevant RESOURCE (IXLSTR.structure_name) profiles in the CLASS(FACILITY).

Queue Sharing Groups (QSG)

At shared queue manager start-up, the queue manager will attempt to connect to a Cross Coupling Facility (XCF) group for the queue sharing group. The IXCCREAT to the group is subject to security checks, and these will ensure that the shared queue manager is allowed to join the group.

Use existing security facilities in these areas.

Queue managers

From WebSphere MQ for z/OS V5.2, a queue manager can be known as a local queue manager or a shared queue manager, depending on whether or not it belongs to a queue sharing group.

Shared queue manager

The new features are as follows:

- Queue sharing group high-level qualifiers – queue sharing group profiles will use the queue sharing group ID as the hlq instead of a ssid.
- New switches and new switch profiles:
  - queue manager checks switch, controlled by the profile – this is used to determine whether or not security checks can use profiles with an hlq of ssid.
  - queue sharing group checks switch, controlled by the profile – this is used to determine whether or not security checks can use profiles with an hlq of qsg.
- Improved security switch messages – there is a new format for most security messages to help you see what security is set up and why. Security messages are now written to the log at queue manager start-up.
- Intra Group Queuing (IGQ) – this is a way for less expensive, small, non-persistent messages to be transferred between queue managers within a queue

sharing group without having to define channels between the queue managers. When applications **open** and **put** messages to remote queues, the local queue manager determines whether intra-group queueing should be used for message transfer.

SECURITY COMMANDS

There are four WebSphere MQ for z/OS commands that help you to administer security for your queue manager:

- `DISPLAY SECURITY ALL|INTERVAL|SWITCHES|TIMEOUT`

  This allows you to see what security is active on your queue manager and how frequently the internal clear out of security information held by the queue manager is performed.

- `REFRESH SECURITY(*|MQADMIN|MQQUEUE|MQPROC|MQNLIST)`

  This command allows you to change your queue manager's security set-up without bringing it down. There will be a performance impact while this command is being processed.

- `RVERIFY SECURITY(userid, userid...)`

  This allows you to change a particular user's access level while the system is up and running. Once the change has been made in RACF, you need to issue this command, specifying each user ID that has had its authority changed.

- `ALTER SECURITY INTERVAL() TIMEOUT()`

  This allows you to alter the frequency of the internal clearout and the period of time for which information is allowed to remain unused in the Queue Manager.

SUMMARY

WebSphere MQ is one of the foundation products of the WebSphere family. Over the years, the number and types of

messages flowing through WebSphere MQ networks have grown for most enterprises. As message volumes have grown, the infrastructure supporting these messages has evolved. Security is a key concern. This evolution has led to WebSphere MQ becoming the foundation for an enterprise-wide mission-critical infrastructure. WebSphere MQ is used for a wide variety of applications on the z/OS platform. WebSphere MQ has a reputation within some (ill-informed) environments for having no security. This article has shown the security features WebSphere MQ implements on the z/OS platform.

T S Laxminarayan
System Programmer
Tata Consultancy Services (India)

Getting started with configuration event messages

With WebSphere MQSeries 5.3.0 for z/OS, IBM introduced configuration event messages.

Every time an object is defined, deleted, altered, or refreshed, a configuration event message is written to the SYSTEM.ADMIN.CONFIG.EVENT queue to reflect the change. It can be used, for example, for error trapping or audit reasons. How often have you asked, “Who defined that?”, “Where are my objects?”, or “Who made that change?”? With configuration event messages you are now able to answer these questions because they contain all the information necessary to identify who, how, and what has been changed. Let me give you an example of why these kinds of event message may be useful.

In our shop, the application programmers, developers, and testers have full control over the MQSeries objects that belong to their application (of course only on test and development systems). They are able to alter, delete, and define objects.

The configuration event messages are a good source of information to us in case there is an application that is no longer running, because the messages will tell us if there was a ‘bad’ object change.

All details for configuration event messages are described in the *WebSphere MQ Event Monitoring* manual.

To make your queue manager collect configuration event messages the following two easy steps have to be performed:

1 Check whether the SYSTEM.ADMIN.CONFIG.EVENT queue exists. If not, the easiest way to define it, is to use one of the other event queues as a sample, eg

```
DEFINE QLOCAL(SYSTEM.ADMIN.CONFIG.EVENT)
LIKE(SYSTEM.ADMIN.CHANNEL.EVENT)
```

You may also use the sample definition in the SCSQPROC library.

2 Issue the command:

```
ALTER QMGR CONFIGEV(ENABLED)
```

From now on the queue manager will collect Configuration Event messages.

Some notes and hints:

- The **ALTER QMGR CONFIGEV(ENABLED)** command will be answered by the queue manager with ‘CSQM171I qmgr CSQMAMMS CONFIGURATION EVENTS REFRESH NEEDED’. This is irritating because the refresh is not required to make the queue manager collect the configuration event messages. The refresh will create a configuration event message for every existing object, and in my opinion it is required only if you need a base of configuration event messages to start with.
- If you use DEFPSIST(NO) in the configuration of the SYSTEM.ADMIN.CONFIG.EVENT queue, the created configuration event messages are non-persistent. With DEFPSIST(YES) they are persistent.

- You should set up a procedure that runs regularly to process the messages from the SYSTEM.ADMIN.CONFIG.EVENT queue. Depending on the number of objects changed, the queue may fill up faster than you expect.
- Check the SYSTEM.ADMIN.CONFIG.EVENT's MAXDEPTH. The queue should not be able to fill up your pageset. When the queue is full, the recording of configuration event messages is stopped without harming the command processing itself. It is better to miss some configuration event messages rather than harm other queues by having a full pageset.
- There will be no configuration event messages for temporary dynamic queues. For permanent dynamic queues configuration event messages are created, so you may want to check your model queues and the SYSTEM.COMMAND.REPLY.MODEL queue.
- If you have many channels in retry (eg in a test environment) you will get lots of configuration event messages, because the MCA (Message Channel Agent) alters the transmission queue at every retry interval (GET(ENABLED)/ GET(DISABLED)).
- If you change the SYSTEM.ADMIN.CONFIG.EVENT queue you will get no configuration event messages. It's the same with internal changes of the TRIGGER attribute of a local queue.

A configuration event message looks like any other 'normal' MQSeries message, built from MQMD and message data. The MQMD will show the message format, MQFMT_EVENT, and message type, MQMT_DATAGRAMM. The message data is based on the Programmable Command Format (PCF) and is divided into the event header and the event data.

In some cases, a browse of the queue holding the configuration event messages is sufficient to find out who changed an object, because user ID and object names are in character

format. But if you want to see details (eg what fields have been changed) you need a program to process the messages. Unfortunately, IBM does not supply a program to process the configuration event messages – you have to write your own.

That's why I wrote a COBOL program to process the configuration event messages. It may be used as a sample of how to deal with these kinds of event message, and you may modify it to suit your needs.

I use a few SYSIN statements to be able to do some selection on the input data. In particular I wanted to be able to exclude special user IDs, like the CHIN user ID, because I do not want to see all these transmission queue changes caused by channel retries.

For a description of the SYSIN statements check the comments at the beginning of the program. You will also find sample compile and run JCL there.

A few notes on the program:

- The program uses an internal table to declare all the object attributes that may occur in configuration event messages. If, in future releases of MQSeries, new objects or object attributes occur, they must be added to this table. Anyway, I did not include processing of the byte string parameters structure, because it is used only for the EventAccountingToken field, which I was not interested in.
- If an object is changed, you will get two configuration event messages, one showing the object before it was changed and one showing the object after it was changed. Unfortunately, these two messages are not related by message ID or correlation ID, and there may be other configuration event messages between these two messages. So it is up to the program to find the matching 'before' and 'after' messages. I made a suggestion to IBM to use the correlation ID, and, hopefully, this will be changed some day.

- I also encountered missing configuration event messages in my test environment. Sometimes the ‘before change’ message is missing, sometimes the ‘after change’ message is missing (looks like a timing issue). This record is still open with IBM, and the program will output proper messages if it finds out that configuration event messages are missing.

If you want to write your own program, please note that there are some errors in the *Event Monitoring* manual that will be changed with the next release of MQ:

- Authentication information attributes, page 167:
  - MQIA_AUTH_INFO_DESC should read MQCA_AUTH_INFO_DESC.
  - MQ_AUTH_INFO_CONN_NAME should read MQCA_AUTH_INFO_CONN_NAME.
- CF structure attributes, page 167:
  - MQCA_CF_LEVEL should read MQIA_CF_LEVEL.
- Channel attributes, page 172:
  - ClusterNamelist (MQCFSL) should read ClusterNamelist (MQCFST) – it is a string, not a string list.
  - MQCACH_SSL_CLIENT_AUTH should read MQIACH_SSL_CLIENT_AUTH.
- Namelist attributes, page 173:
  - MQCA_NAMELIST_TYPE should read MQIA_NAMELIST_TYPE.
- Queue manager attributes, page 179:
  - MQIA_IGQ_USER_ID should read MQCA_IGQ_USER_ID.
- Storage class attributes, page 182:

– MQIA_PAGE_SET_ID should read MQIA_PAGESET_ID.

CEVTMSG COBOL PROGRAM

```
CBL  NODYNAM,LIB,OBJECT,RENT,RES,APOST
      * ------------------------------------------------------------ *
       IDENTIFICATION DIVISION.
      * ------------------------------------------------------------ *
       PROGRAM-ID. CEVTMSG.
      *REMARKS
       ****************************************************************
      * SAMPLE TO GET CONFIGURATION EVENT MESSAGES FROM THE           *
      * SYSTEM.ADMIN.CONFIG.EVENT QUEUE AND DO SOME PRINTOUT          *
      * DEPENDING ON EVENT DATA AND SYSIN PARMS                       *
      *                                                               *
      * MARCH 2005 - STEFAN RAABE (stefan.raabe@t-online.de)          *
      * ************************************************************* *
      * SAMPLE JCL AND SYSIN PARMS                                    *
      * //YOUR JOBCARD GOES HERE
      * //CEVTMSG  EXEC PGM=CEVTMSG
      * //STEPLIB  DD DSN=YOUR.STEPLIB,DISP=SHR
      * //SYSPRINT DD SYSOUT=*
      * //SYSIN    DD *
      * * SPECIFY YOUR QMGR NAME
      * QMGR            XXXX
      * * SPECIFY YOUR EVENT QUEUE
      * EVENTQUEUE      SYSTEM.ADMIN.CONFIG.EVENT
      * * SHOW DETAILS FOR DEFINE EVENT MESSAGES (OFF/ON)
      * DETAILS-DEFINE  OFF
      * * SHOW DETAILS FOR ALTER EVENT MESSAGES (OFF/ON)
      * DETAILS-ALTER   OFF
      * * SHOW DETAILS FOR DELETE EVENT MESSAGES (OFF/ON)
      * DETAILS-DELETE  OFF
      * * SHOW DETAILS FOR REFRESH EVENT MESSAGES (OFF/ON)
      * DETAILS-REFRESH OFF
      * * BROWSE QUEUE OR GET MESSAGES (OFF/ON)
      * BROWSE          ON
      * * IGNORE EVENT MESSAGES FROM THESE USERS (UP TO FIVE USERS IN
      * *                                        ONE LINE)
      * SKIP-USER       XXXMSTR XXXCHIN
      * *
      * * IGNORE EVENT MESSAGES FOR THESE OBJECT TYPES / NAMES
      * *     (UP TO 10 LINES CAN BE SPECIFIED, ONE OBJECT EACH)
      * SKIP-OBJ        QUEUE SYSTEM.CSQOREXX*
      * SKIP-OBJ        QUEUE SYSTEM.CSQUTIL*
      * /*
      * //                                                           *
       * ------------------------------------------------------------ *
```

```
 ENVIRONMENT DIVISION.
* ------------------------------------------------------------ *
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT SYSPRINT ASSIGN TO UT-S-SYSPRINT.
     SELECT SYSIN    ASSIGN TO UT-S-SYSIN.
* ------------------------------------------------------------ *
 DATA DIVISION.
* ------------------------------------------------------------ *
 FILE SECTION.
 FD  SYSPRINT
     BLOCK CONTAINS Ø RECORDS
     RECORDING MODE IS F.
 Ø1  SYSPRINT-REC.
     Ø5  CARRIAGE-CONTROL        PIC X.
     Ø5  SYSPRINT-DATA           PIC X(132).
 FD  SYSIN
     RECORD CONTAINS 8Ø CHARACTERS
     RECORDING MODE IS F.
 Ø1  SYSIN-REC.
     Ø5  SYSIN-DATA              PIC X(8Ø).
* ------------------------------------------------------------ *
 WORKING-STORAGE SECTION.
* ------------------------------------------------------------ *
* HEADER LINES OF REPORT
 Ø1  CONSTANTS.
     Ø5  ON-VALUE                PIC X(1)   VALUE '1'.
     Ø5  OFF-VALUE               PIC X(1)   VALUE 'Ø'.
 Ø1  W-HEADER-1.
     Ø5  FILLER                  PIC X(12)  VALUE SPACES.
     Ø5  WH1-DD                  PIC XX.
     Ø5  FILLER                  PIC X      VALUE '.'.
     Ø5  WH1-MM                  PIC XX.
     Ø5  FILLER                  PIC X      VALUE '.'.
     Ø5  WH1-YY                  PIC XX.
     Ø5  FILLER                  PIC XX     VALUE SPACES.
     Ø5  FILLER                  PIC X(35)  VALUE
               'CONFIGURATION EVENT MESSAGES REPORT'.
 Ø1  W-HEADER-2.
     Ø5  FILLER                  PIC X(2Ø)  VALUE
               '--------------------'.
     Ø5  FILLER                  PIC X(37)  VALUE
               '-------------------------------------'.
 Ø1  W-HEADER-3.
     Ø5  FILLER                  PIC X(1Ø)  VALUE SPACES.
     Ø5  FILLER                  PIC X(12)  VALUE '      QMGR: '.
     Ø5  WH3-QMGR                PIC X(48)  VALUE SPACES.
 Ø1  W-HEADER-4.
     Ø5  FILLER                  PIC X(1Ø)  VALUE SPACES.
     Ø5  FILLER                  PIC X(12)  VALUE 'EVENTQUEUE: '.
```

```
     05  WH4-EVENT-QUEUE-NAME    PIC X(48)  VALUE SPACES.
 01  W-HEADER-5C.
     05  FILLER                  PIC X(12)  VALUE ' SHOW CREATE'.
     05  FILLER                  PIC X(10)  VALUE ' DETAILS: '.
     05  WH5-DETAILS-DEF         PIC X(3)   VALUE SPACES.
 01  W-HEADER-5A.
     05  FILLER                  PIC X(12)  VALUE '  SHOW ALTER'.
     05  FILLER                  PIC X(10)  VALUE ' DETAILS: '.
     05  WH5-DETAILS-ALT         PIC X(3)   VALUE SPACES.
 01  W-HEADER-5D.
     05  FILLER                  PIC X(12)  VALUE ' SHOW DELETE'.
     05  FILLER                  PIC X(10)  VALUE ' DETAILS: '.
     05  WH5-DETAILS-DEL         PIC X(3)   VALUE SPACES.
 01  W-HEADER-5R.
     05  FILLER                  PIC X(12)  VALUE 'SHOW REFRESH'.
     05  FILLER                  PIC X(10)  VALUE ' DETAILS: '.
     05  WH5-DETAILS-REF         PIC X(3)   VALUE SPACES.
 01  W-HEADER-6.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(12)  VALUE 'SKIP USERS: '.
     05  WH6-SKIP                PIC X(70)  VALUE SPACES.
 01  W-HEADER-7.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(12)  VALUE 'BROWSE/GET: '.
     05  WH7-BROWSE              PIC X(48)  VALUE SPACES.
 01  W-HEADER-8.
     05  FILLER                  PIC X(10)  VALUE '  SKIP OBJ'.
     05  FILLER                  PIC X(12)  VALUE 'TYPE - OBJ: '.
     05  WH8-SKIPTYPE            PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(3)   VALUE ' - '.
     05  WH8-SKIPNAME            PIC X(48)  VALUE SPACES.
* TRAILER LINES OF REPORT
 01  W-TRAILER-1.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(41)  VALUE
           '     END OF REPORT, TOTEL MESSAGES READ: '.
     05  WT1-MSGCOUNTER-READ     PIC Z(8)9.
 01  W-TRAILER-11.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(41)  VALUE
           '           REFRESH EVENT MESSAGES FOUND: '.
     05  WT11-MSGCOUNTER-REFRESH PIC Z(8)9.
 01  W-TRAILER-12.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(41)  VALUE
           '            CREATE EVENT MESSAGES FOUND: '.
     05  WT12-MSGCOUNTER-CREATE  PIC Z(8)9.
 01  W-TRAILER-13.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(41)  VALUE
```

```
                '            DELETE EVENT MESSAGES FOUND: '.
     05  WT13-MSGCOUNTER-DELETE  PIC Z(8)9.
 01  W-TRAILER-14.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(41)  VALUE
                '             ALTER EVENT MESSAGES FOUND: '.
     05  WT14-MSGCOUNTER-ALTER   PIC Z(8)9.
 01  W-TRAILER-2.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(41)  VALUE
                '               EVENT MESSAGES PROCSSED: '.
     05  WT2-MSGCOUNTER-PROCESSED    PIC Z(8)9.
 01  W-TRAILER-3.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(41)  VALUE
                '              EVENT MESSAGES SKIPPED: '.
     05  WT3-MSGCOUNTER-SKIPPED  PIC Z(8)9.
 01  W-TRAILER-4.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(41)  VALUE
                '             EVENT MESSAGES DISPLAYED: '.
     05  WT4-MSGCOUNTER-DISPLAYED PIC Z(8)9.
* HEADER LINE FOR AN EVENT RECORD
 01  W-EVENT-HEADER-1.
     05  WEH1-DATE.
         10 WEH1-DAY             PIC X(02)  VALUE SPACES.
         10 FILLER               PIC X(01)  VALUE '.'.
         10 WEH1-MONTH           PIC X(02)  VALUE SPACES.
         10 FILLER               PIC X(01)  VALUE '.'.
         10 WEH1-YEAR            PIC X(04)  VALUE SPACES.
     05  FILLER                  PIC X(01)  VALUE SPACE.
     05  WEH1-TIME.
         10 WEH1-HOUR            PIC X(02)  VALUE SPACES.
         10 FILLER               PIC X(01)  VALUE ':'.
         10 WEH1-MINUTE          PIC X(02)  VALUE SPACES.
         10 FILLER               PIC X(01)  VALUE ':'.
         10 WEH1-SECOND          PIC X(02)  VALUE SPACES.
     05  FILLER                  PIC X(01)  VALUE SPACE.
     05  WEH1-QMGR               PIC X(04)  VALUE SPACES.
     05  FILLER                  PIC X(01)  VALUE SPACE.
     05  WEH1-USER               PIC X(12)  VALUE SPACES.
     05  FILLER                  PIC X(01)  VALUE SPACE.
     05  WEH1-ACTION             PIC X(07)  VALUE SPACES.
         88 OBJECT-CHANGE          VALUE 'CHANGE '.
         88 OBJECT-REFRESH         VALUE 'REFRESH'.
         88 OBJECT-DELETE          VALUE 'DELETE '.
         88 OBJECT-CREATE          VALUE 'CREATE '.
     05  FILLER                  PIC X(01)  VALUE SPACE.
     05  WEH1-OBJECTTYPE         PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(01)  VALUE SPACE.
```

```
     05  WEH1-OBJECTNAME         PIC X(48)  VALUE SPACES.
     05  FILLER                  PIC X(01)  VALUE SPACE.
     05  WEH1-DISP               PIC X(06)  VALUE SPACES.
     05  FILLER                  PIC X(01)  VALUE SPACE.
     05  WEH1-HOW                PIC X(20)  VALUE SPACES.
* INTEGER ATTRIBUTE LINES FOR AN EVENT RECORD
 01  W-EVENT-DETAIL-1.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  WED1-DESC               PIC X(120).
 01  W-EVENT-DETAIL-2.
     05  FILLER                  PIC X(10)  VALUE '-> NEW -> '.
     05  WED2-DESC               PIC X(120).
* ERROR MESSAGES
 01  W-ERROR-1.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(27)  VALUE
        '|||        UNKNOWN KEYWORD '.
     05  WE1-KEYWORD             PIC X(20).
     05  FILLER                  PIC X(75)  VALUE
        ' - IGNORED, PROCESSING CONTINUES'.
*
 01  W-ERROR-2.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(122) VALUE
        '|||        NO QUEUEMANAGER NAME PASSED TO PROGRAM.'.
 01  W-ERROR-3.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(122) VALUE
        '|||        NO EVENTQUEUE NAME PASSED TO PROGRAM.'.
 01  W-ERROR-4.
     05  FILLER                  PIC X(13)  VALUE SPACES.
     05  FILLER                  PIC X(32)  VALUE
        '|||        AN ERROR OCCURRED IN '.
     05  WE4-TYPE                PIC X(10).
     05  FILLER                  PIC X(20)  VALUE
        '. COMPLETION CODE = '.
     05  WE4-COMPCODE            PIC Z(8)9.
     05  FILLER                  PIC X(15)  VALUE
        ' REASON CODE ='.
     05  WE4-REASON              PIC Z(8)9.
 01  W-ERROR-5.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(122) VALUE
        '|||        NON EVENT MESSAGE FOUND ON EVENT QUEUE, IGNORE
-       'D'.
 01  W-ERROR-6.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(122) VALUE
        '|||        EVENT HEADER DOES NOT SPECIFY EVENT MESSAGE, I
-       'GNORED'.
```

```
 Ø1  W-ERROR-7.
     Ø5  FILLER                  PIC X(1Ø)  VALUE SPACES.
     Ø5  FILLER                  PIC X(122) VALUE
        '|||        EVENT HEADER DOES NOT SPECIFY CONFIGURATION EV
-       'ENT, IGNORED'.
 Ø1  W-ERROR-8.
     Ø5  FILLER                  PIC X(1Ø)  VALUE SPACES.
     Ø5  FILLER                  PIC X(122) VALUE
        '|||        INTERNAL CHANGE-BEFORE-TABLE FULL, BEFORE VALU
-       'ES ARE PRINTED'.
 Ø1  W-ERROR-9.
     Ø5  FILLER                  PIC X(1Ø)  VALUE SPACES.
     Ø5  FILLER                  PIC X(122) VALUE
        '|||        BEFORE RECORD NOT FOUND IN INTERNAL CHANGE-BEF
-       'ORE-TABLE, AFTER VALUES ARE PRINTED'.
 Ø1  W-ERROR-1Ø.
     Ø5  FILLER                  PIC X(1Ø)  VALUE SPACES.
     Ø5  FILLER                  PIC X(122) VALUE
        '|||        END OF PROCESSING, BUT BEFORE RECORD LEFT IN C
-       'HANGE-BEFORE-TABLE, BEFORE VALUES ARE PRINTED'.
 Ø1  W-ERROR-11.
     Ø5  FILLER                  PIC X(6)  VALUE ' -->:'.
     Ø5  W-ERROR-11-NUM          PIC Z(8)9.
 Ø1  W-ERROR-12.
     Ø5  FILLER                  PIC X(1Ø)  VALUE SPACES.
     Ø5  FILLER                  PIC X(122) VALUE
        '|||        TOO MANY (>1Ø) SKIP-OBJ LINES.'.
* VARIOUS WORK FIELDS
 Ø1  W-RETURN-CODE               PIC S9(Ø4) BINARY VALUE ZERO.
 Ø1  W-RC-OK                     PIC S9(4)  VALUE Ø.
 Ø1  W-RC-WARNING                PIC S9(4)  VALUE 4.
 Ø1  W-RC-ERROR                  PIC S9(4)  VALUE 8.
 Ø1  WORK-FIELDS.
     Ø5 W-SYSPRINT-DATA          PIC X(132).
     Ø5 W-INPUT-KEYWORD          PIC X(3Ø).
     Ø5 W-INPUT-VALUE            PIC X(48).
     Ø5 W-USER-1                 PIC X(12).
     Ø5 W-USER-2                 PIC X(12).
     Ø5 W-USER-3                 PIC X(12).
     Ø5 W-USER-4                 PIC X(12).
     Ø5 W-USER-5                 PIC X(12).
     Ø5 W-OBJSMAX                PIC S9(9)  BINARY VALUE 1Ø.
     Ø5 W-OBJSKIPTAB.
        1Ø W-OBJSKIPTYPE         PIC X(1Ø) VALUE SPACE OCCURS 1Ø.
        1Ø W-OBJSKIPNAME         PIC X(48) VALUE SPACE OCCURS 1Ø.
     Ø5 W-OBJSPTR                PIC S9(9)  BINARY.
     Ø5  W-SKIP-OBJECT           PIC X      VALUE SPACE.
         88 SKIP-OBJECT-TRUE       VALUE '1'.
         88 SKIP-OBJECT-FALSE      VALUE '2'.
     Ø5 W-OBJCOMP-TABLE          OCCURS 1Ø.
```

```
           10 W-OBJCOMP-NAME         PIC X(48).
           10 W-OBJCOMP-TYPE         PIC X(10).
           10 W-OBJCOMP-DELIMITER    PIC X.
           10 W-OBJCOMP-COUNT        PIC S9(9) BINARY.
        05 WORK-DATE.
           10  W-YY                  PIC X(02).
           10  W-MM                  PIC X(02).
           10  W-DD                  PIC X(02).
        05  W-MESSAGE-STORED         PIC X      VALUE SPACE.
            88 MESSAGE-STORED-FALSE    VALUE '0'.
            88 MESSAGE-STORED-TRUE     VALUE '1'.
        05  W-PRINT-ATTRIBUTE        PIC X      VALUE SPACE.
            88 PRINT-ATTRIBUTE-1       VALUE '1'.
            88 PRINT-ATTRIBUTE-2       VALUE '2'.
            88 PRINT-ATTRIBUTE-RESET   VALUE ' '.
        05  W-DETAILS-DEFINE         PIC X      VALUE SPACE.
            88 CRE-DET-ON              VALUE '1'.
        05  W-DETAILS-ALTER          PIC X      VALUE SPACE.
            88 ALT-DET-ON              VALUE '1'.
        05  W-DETAILS-DELETE         PIC X      VALUE SPACE.
            88 DEL-DET-ON              VALUE '1'.
        05  W-DETAILS-REFRESH        PIC X      VALUE SPACE.
            88 REF-DET-ON              VALUE '1'.
        05  W-SKIP-USER              PIC X      VALUE SPACE.
            88 SKIP-USER-ON            VALUE '1'.
        05  W-SKIP-OBJ               PIC X      VALUE SPACE.
            88 SKIP-OBJ-ON             VALUE '1'.
        05  W-BROWSE                 PIC X      VALUE SPACE.
            88 BROWSE-ON               VALUE '1'.
        05  W-CHANGE                 PIC S9(9)  BINARY.
            88 CHANGE-MESSAGE-BEFORE   VALUE 1.
            88 CHANGE-MESSAGE-AFTER    VALUE 2.
        05  WRK-OTHER                PIC X(120).
        05  WRK-NUMBER-RIGHT         PIC ZZZZZZZZZZZZ9.
        05  WRK-NUMBER-ZWO REDEFINES WRK-NUMBER-RIGHT
                                     PIC XXXXXXXXXXXXX.
        05  WRK-NUMBER-TAB.
           10 WRK-NUMBER-X           PIC X      OCCURS 13.
        05  WRK-NUMBER REDEFINES WRK-NUMBER-TAB
                                     PIC XXXXXXXXXXXXX.
        05  YES-VALUE                PIC S9(9)  BINARY VALUE 1.
        05  NO-VALUE                 PIC S9(9)  BINARY VALUE 0.
        05  STRING256                PIC X(256).
    * TABLE INDEX FIELDS
     01  I                           PIC S9(9)  BINARY.
     01  J                           PIC S9(9)  BINARY.
     01  K                           PIC S9(9)  BINARY.
     01  L                           PIC S9(9)  BINARY.
     01  CB                          PIC S9(9)  BINARY.
     01  S1                          PIC S9(9)  BINARY.
```

```
 Ø1  S2                          PIC S9(9)  BINARY.
 Ø1  SC                          PIC S9(9)  BINARY.
 Ø1  COMMIT-COUNTER              PIC S9(9)  BINARY.
* ATTRIBUTE TABLE FOR INTEGER ATTRIBUTES
 Ø1  ATTRIBUTE-INTEGER-TABLE.
     Ø5 AI-NUMBER                PIC S9(9)  BINARY.
     Ø5 AI-TAB  OCCURS 2.
       1Ø AI-REC  OCCURS 1ØØ.
        15 AI-ID                 PIC S9(9)  BINARY VALUE ZERO.
        15 AI-FLAG               PIC X      VALUE SPACE.
           88 AI-NOT-SET            VALUE ' '.
           88 AI-SET                VALUE '1'.
        15 AI-TYPE               PIC X      VALUE SPACE.
        15 AI-DESC               PIC X(2Ø)  VALUE SPACE.
        15 AI-EXP                PIC X(4Ø)  VALUE SPACE.
        15 AI-VALUE              PIC S9(9)  BINARY VALUE ZERO.
* ATTRIBUTE TABLE FOR STRING ATTRIBUTES
 Ø1  ATTRIBUTE-STRING-TABLE.
     Ø5 AS-NUMBER                PIC S9(9)  BINARY.
     Ø5 AS-TAB  OCCURS 2.
       1Ø AS-REC  OCCURS 1ØØ.
        15 AS-ID                 PIC S9(9)  BINARY VALUE ZERO.
        15 AS-FLAG               PIC X      VALUE SPACE.
           88 AS-NOT-SET            VALUE ' '.
           88 AS-SET                VALUE '1'.
        15 AS-DESC               PIC X(2Ø)  VALUE SPACE.
        15 AS-LENGTH             PIC S9(9)  BINARY.
        15 AS-VALUE              PIC X(256) VALUE SPACE.
* ATTRIBUTE TABLE FOR STRINGLIST ATTRIBUTES
 Ø1  ATTRIBUTE-STRLST-TABLE.
     Ø5 ASL-NUMBER               PIC S9(9)  BINARY.
     Ø5 ASL-TAB  OCCURS 2.
       1Ø ASL-REC  OCCURS 1Ø.
        15 ASL-ID                PIC S9(9)  BINARY VALUE ZERO.
        15 ASL-FLAG              PIC X      VALUE SPACE.
           88 ASL-NOT-SET           VALUE ' '.
           88 ASL-SET               VALUE '1'.
        15 ASL-DESC              PIC X(2Ø)  VALUE SPACE.
        15 ASL-LENGTH            PIC S9(9)  BINARY.
        15 ASL-COUNT             PIC S9(9)  BINARY.
        15 ASL-VALUE-DUMMY.
         2Ø ASL-VALUE OCCURS 256 PIC X(256) VALUE SPACE.
        15 ASL-VALUE1 REDEFINES ASL-VALUE-DUMMY PIC X(65536).
* POINTER TO PROCESS EVENT MESSAGES
 Ø1  DATA-PTR                    POINTER.
 Ø1  DATA-PTR-NUM   REDEFINES DATA-PTR   PIC S9(9) COMP.
 Ø1  STRING-PTR                  POINTER.
 Ø1  STRING-PTR-NUM REDEFINES STRING-PTR PIC S9(9) COMP.
* FIELDS REQUIRED FOR MQSERIES API
 Ø1  BUFFER-LENGTH               PIC S9(9)  BINARY VALUE 64ØØØ.
```

```
 Ø1  HCONN                       PIC S9(9)  BINARY.
 Ø1  COMPCODE                    PIC S9(9)  BINARY.
 Ø1  REASON                      PIC S9(9)  BINARY.
 Ø1  OPTIONS                     PIC S9(9)  BINARY.
 Ø1  HOBJ                        PIC S9(9)  BINARY.
 Ø1  DATA-LENGTH                 PIC S9(9)  BINARY.
 Ø1  MESSAGE-DATA                PIC X(64ØØØ) VALUE SPACES.
 Ø1  MESSAGE-DATA2 REDEFINES MESSAGE-DATA.
     COPY CMQCFHL.
* SOME COUNTERS
 Ø1  MSGCOUNTER-REFRESH          PIC S9(9)  BINARY.
 Ø1  MSGCOUNTER-DELETE           PIC S9(9)  BINARY.
 Ø1  MSGCOUNTER-CREATE           PIC S9(9)  BINARY.
 Ø1  MSGCOUNTER-ALTER            PIC S9(9)  BINARY.
 Ø1  MSGCOUNTER-READ             PIC S9(9)  BINARY.
 Ø1  MSGCOUNTER-PROCESSED        PIC S9(9)  BINARY.
 Ø1  MSGCOUNTER-SKIPPED          PIC S9(9)  BINARY.
 Ø1  MSGCOUNTER-DISPLAYED        PIC S9(9)  BINARY.
* TO REMEMBER CHANGE BEFORE RECORDS
 Ø1  CHANGE-BEFORE-TABLE.
     Ø5 CHANGE-BEFORE-TABLE-RECORD OCCURS 5Ø.
        1Ø CBTR-USED             PIC X.
        1Ø CBTR-OBJECT-NAME      PIC X(48).
        1Ø CBTR-OBJECT-TYPE      PIC X(1Ø).
        1Ø CBTR-PUTDATE          PIC X(8).
        1Ø CBTR-PUTTIME          PIC X(8).
        1Ø CBTR-EVENT-MSG        PIC X(64ØØØ).
* SET TO NUMBER OF TABLE ENTRIES
 Ø1  CBTR-ENTRIES                PIC S9(9)  BINARY VALUE 5Ø.
* MQ DATA AREAS
 Ø1  WØ5-MQM-OBJECT-DESCRIPTOR.
     COPY CMQODV.
 Ø1  WØ5-MQM-MESSAGE-DESCRIPTOR.
     COPY CMQMDV.
 Ø1  WØ5-MQM-GET-MESSAGE-OPTIONS.
     COPY CMQGMOV.
*    MQ CONSTANTS, RETURNCODES, ...
 Ø1  WØ5-MQM-CONSTANTS.
     COPY CMQV.
     COPY CMQXV.
     COPY CMQCFV.
* ------------------------------------------------------------- *
 LINKAGE SECTION.
* ------------------------------------------------------------- *
 Ø1  LNK-HFELD                   PIC S9(9)  BINARY.
 Ø1  LNK-STRING                  PIC X(256).
 Ø1  LNK-MQCFST.
     COPY CMQCFSTL.
 Ø1  LNK-MQCFIN.
     COPY CMQCFINL.
```

```
 Ø1  LNK-MQCFSL.
     COPY CMQCFSLL.
 Ø1  LNK-MQCFBS.
     COPY CMQCFBSL.
     EJECT
* ---------------------------------------------------------- *
 PROCEDURE DIVISION.
* ---------------------------------------------------------- *
 MAIN SECTION.
* ---------------------------------------------------------- *
     OPEN    OUTPUT SYSPRINT.
     MOVE    W-RC-OK TO W-RETURN-CODE.
     PERFORM HEADER.
     PERFORM PARM-INPUT.
     PERFORM INIT-TABLES.
     PERFORM RESET-TABLES.
     MOVE ZERO  TO MSGCOUNTER-READ    MSGCOUNTER-PROCESSED
                   MSGCOUNTER-SKIPPED MSGCOUNTER-DISPLAYED
                   MSGCOUNTER-REFRESH MSGCOUNTER-DELETE
                   MSGCOUNTER-CREATE  MSGCOUNTER-ALTER.
     MOVE ZERO TO  COMMIT-COUNTER.
     PERFORM MQ-CONN.
     PERFORM MQ-OPEN.
     PERFORM PROCESS-MESSAGES.
     PERFORM MQ-CLOSE.
 MAIN-DISCONNECT.
     PERFORM MQ-DISC.
 MAIN-END.
     PERFORM TRAILER.
     MOVE    W-RETURN-CODE TO RETURN-CODE.
     CLOSE   SYSPRINT.
     STOP RUN.
* ---------------------------------------------------------- *
* PROCESS THE EVENT MESSAGES (LOOP ON MESSAGES)
* ---------------------------------------------------------- *
 PROCESS-MESSAGES SECTION.
* SET GMO FOR FIRST MQGET CALL
     MOVE MQGMO-NO-WAIT              TO MQGMO-OPTIONS.
     ADD  MQGMO-ACCEPT-TRUNCATED-MSG TO MQGMO-OPTIONS.
     IF BROWSE-ON THEN
       ADD MQGMO-BROWSE-FIRST        TO MQGMO-OPTIONS
     END-IF.
* MAKE THE FIRST GET CALL OUTSIDE THE LOOP BECAUSE THIS CALL
* USES THE BROWSE-FIRST OPTION
     MOVE SPACES TO MESSAGE-DATA.
     ADD 1 TO COMMIT-COUNTER.
     CALL 'MQGET' USING HCONN
                        HOBJ
                        MQMD
                        MQGMO
```

```
                        BUFFER-LENGTH
                        MESSAGE-DATA
                        DATA-LENGTH
                        COMPCODE
                        REASON.
* SET OPTIONS TO BROWSE NEXT FOR THE NEXT MQGET CALLS
     MOVE MQGMO-NO-WAIT              TO MQGMO-OPTIONS.
     ADD  MQGMO-ACCEPT-TRUNCATED-MSG TO MQGMO-OPTIONS.
     IF BROWSE-ON THEN
       ADD MQGMO-BROWSE-NEXT         TO MQGMO-OPTIONS
     END-IF.
* LOOP ON QUEUE, TEST RETURNCODES BEFORE
     PERFORM WITH TEST BEFORE
             UNTIL COMPCODE NOT = MQCC-OK
                 AND NOT (COMPCODE = MQCC-WARNING AND
                          REASON = MQRC-TRUNCATED-MSG-ACCEPTED)
        ADD 1 TO MSGCOUNTER-READ
* CHECK IF EVENT MESSAGE
        IF MQMD-MSGTYPE NOT = MQMT-DATAGRAM OR
           MQMD-FORMAT NOT = MQFMT-EVENT THEN
* NOT EVENT TYPE IN MESSAGE DESCRIPTOR
             MOVE    W-ERROR-5 TO W-SYSPRINT-DATA
             PERFORM PRINT-LINE
        ELSE
          IF MQCFH-TYPE NOT = MQCFT-EVENT THEN
* NOT EVENT IN CFH HEADER
             MOVE    W-ERROR-6 TO W-SYSPRINT-DATA
             PERFORM PRINT-LINE
           ELSE
* NOT CONFIG EVENT IN CFH HEADER
             IF MQCFH-COMMAND NOT = MQCMD-CONFIG-EVENT THEN
                MOVE W-ERROR-7 TO W-SYSPRINT-DATA
                PERFORM PRINT-LINE
              ELSE
* MESSAGE PROCESSING GOES HERE
                PERFORM PROCESS-EVENT-MESSAGE
                ADD 1 TO MSGCOUNTER-PROCESSED
             END-IF
          END-IF
        END-IF
* RESET MSGID CORRELID MSGDATA BEFORE DOING NEXT GET
        MOVE MQMI-NONE TO MQMD-MSGID
        MOVE MQCI-NONE TO MQMD-CORRELID
        MOVE SPACES    TO MESSAGE-DATA
        ADD 1 TO COMMIT-COUNTER
* COMMIT EVERY 1ØØ MESSAGES
        IF COMMIT-COUNTER > 1ØØ THEN
           CALL "MQCMIT" USING HCONN COMPCODE REASON
           MOVE Ø TO COMMIT-COUNTER
        END-IF
```

```
* GET THE NEXT MESSAGE
        CALL 'MQGET' USING HCONN
                           HOBJ
                           MQMD
                           MQGMO
                           BUFFER-LENGTH
                           MESSAGE-DATA
                           DATA-LENGTH
                           COMPCODE
                           REASON
* LOOP ON MESSAGES
     END-PERFORM.
* CHECK WHY WE LEFT THE LOOP
     IF (COMPCODE = MQCC-FAILED) AND
        (REASON = MQRC-NO-MSG-AVAILABLE) THEN
* QUEUE IS AT EOF, SO COMMIT WORK
       CALL "MQCMIT" USING HCONN COMPCODE REASON
* CHECK IF THERE ARE ANY OBJECTS LEFT IN CHANGE-BEFORE TABLE
* WHICH HAVE NOT BEEN PROCESSED
       PERFORM VARYING CB FROM 1 BY 1 UNTIL CB > CBTR-ENTRIES
         IF CBTR-USED (CB) = ON-VALUE THEN
* THIS ENTRY HAS NOT BEEN PROCESSED
           PERFORM RESET-TABLES
           MOVE    CBTR-PUTDATE (CB)   TO MQMD-PUTDATE
           MOVE    CBTR-PUTTIME (CB)   TO MQMD-PUTTIME
           MOVE    CBTR-EVENT-MSG (CB) TO MESSAGE-DATA
           PERFORM EVENT-HEADER
           SET     DATA-PTR            TO ADDRESS OF MESSAGE-DATA
           ADD     MQCFH-STRUCLENGTH   TO DATA-PTR-NUM
           PERFORM EVENT-ATTRIBUTES MQCFH-PARAMETERCOUNT TIMES
* NOW DO THE OUTPUT
           MOVE    W-ERROR-1Ø        TO W-SYSPRINT-DATA
           PERFORM PRINT-LINE
           MOVE    W-EVENT-HEADER-1  TO W-SYSPRINT-DATA
           PERFORM PRINT-LINE
           SET     PRINT-ATTRIBUTE-1 TO TRUE
           PERFORM OBJECT-DETAILS
         END-IF
       END-PERFORM
       MOVE SPACES TO W-SYSPRINT-DATA
* MQSERIES ERROR DURING LOOP ON MESSAGES
     ELSE
        MOVE 'GET'     TO WE4-TYPE
        MOVE COMPCODE  TO WE4-COMPCODE
        MOVE REASON    TO WE4-REASON
        MOVE W-ERROR-4 TO W-SYSPRINT-DATA
        PERFORM PRINT-LINE
     END-IF.
 PROCESS-MESSAGES-END.
     EXIT.
```

```
* ------------------------------------------------------------ *
* PROCESS A SINGLE EVENT MESSAGE
* ------------------------------------------------------------ *
 PROCESS-EVENT-MESSAGE SECTION.
* GO FOR HEADER FIRST
     INITIALIZE W-EVENT-HEADER-1.
     PERFORM EVENT-HEADER.
* NOW GO FOR MESSAGE DATA
     SET DATA-PTR TO ADDRESS OF MESSAGE-DATA.
     ADD MQCFH-STRUCLENGTH TO DATA-PTR-NUM.
* LOOP ON ATTRIBUTES IN MESSAGE DATA
     PERFORM EVENT-ATTRIBUTES MQCFH-PARAMETERCOUNT TIMES.
* CHECK IF THIS EVENT MESSAGE SHOULD BY SKIPPED BY OBJECT NAME
* ASSUME NO, THEN CHECK
     SET SKIP-OBJECT-FALSE TO TRUE
     IF SKIP-OBJ-ON THEN
       PERFORM OBJ-SKIP-CHECK
     END-IF
* CHECK IF THIS EVENT MESSAGE SHOULD BE SKIPPED BY USERID
     IF (SKIP-USER-ON AND ( WEH1-USER = W-USER-1 OR
                            WEH1-USER = W-USER-2 OR
                            WEH1-USER = W-USER-3 OR
                            WEH1-USER = W-USER-4 OR
                            WEH1-USER = W-USER-5 )) OR
* OR BY OBJECT NAME
                            SKIP-OBJECT-TRUE THEN
       ADD 1 TO MSGCOUNTER-SKIPPED
       PERFORM RESET-TABLES
     ELSE
* IF OBJECT IS ALTERED WE GET 2 EVENT RECORDS, WHICH MAY NOT
* BE DIRECTLY IN SEQUENCE, THATS WHY WE HANDLE ALTERATION
* EVENTS SEPARATELY
       IF OBJECT-CREATE OR OBJECT-DELETE OR OBJECT-REFRESH THEN
* PRINT THE EVENT HEADER LINE IN EVERY CASE
         ADD 1 TO MSGCOUNTER-DISPLAYED
         MOVE W-EVENT-HEADER-1 TO W-SYSPRINT-DATA
         PERFORM PRINT-LINE
* CHECK IF DETAILS FOR THESE EVENTS SHOULD BE PRINTED (WE DO NOT
* CARE FOR REFRESH-DETAILS HERE)
         IF (OBJECT-CREATE AND CRE-DET-ON) OR
            (OBJECT-DELETE AND DEL-DET-ON) OR
             OBJECT-REFRESH THEN
* SHOW ALL DETAILS
           PERFORM OBJECT-DETAILS
         END-IF
* RESET TABLES
         PERFORM RESET-TABLES
       ELSE
* THIS IS A CHANGE OBJECT RECORD
         ADD 1 TO MSGCOUNTER-ALTER
```

```
* IS THIS A CHANGE BEFORE RECORD?
         IF CHANGE-MESSAGE-BEFORE THEN
* YES, REMEMBER IT IN THE OBJECT TABLE
           SET MESSAGE-STORED-FALSE TO TRUE
           PERFORM VARYING CB FROM 1 BY 1 UNTIL CB > CBTR-ENTRIES
                                        OR MESSAGE-STORED-TRUE
* CHECK FOR A FREE ENTRY
             IF CBTR-USED (CB) = OFF-VALUE THEN
               MOVE ON-VALUE            TO CBTR-USED (CB)
               MOVE WEH1-OBJECTNAME     TO CBTR-OBJECT-NAME (CB)
               MOVE WEH1-OBJECTTYPE     TO CBTR-OBJECT-TYPE (CB)
               MOVE MESSAGE-DATA        TO CBTR-EVENT-MSG (CB)
               MOVE MQMD-PUTDATE        TO CBTR-PUTDATE (CB)
               MOVE MQMD-PUTTIME        TO CBTR-PUTTIME (CB)
               SET  MESSAGE-STORED-TRUE TO TRUE
             END-IF
           END-PERFORM
* NO SPACE TO STORE MESSAGE, SO PRINT ALL WE GOT
           IF MESSAGE-STORED-FALSE THEN
             MOVE    W-ERROR-8             TO W-SYSPRINT-DATA
             PERFORM PRINT-LINE
             MOVE    W-EVENT-HEADER-1      TO W-SYSPRINT-DATA
             PERFORM PRINT-LINE
             SET     PRINT-ATTRIBUTE-1     TO TRUE
             PERFORM OBJECT-DETAILS
             SET     PRINT-ATTRIBUTE-RESET TO TRUE
           END-IF
           PERFORM RESET-TABLES
         END-IF
* IS THIS THE CHANGE AFTER RECORD?
         IF CHANGE-MESSAGE-AFTER THEN
* YES, CHECK IF BEFORE IMAGE WAS STORED
           SET MESSAGE-STORED-FALSE TO TRUE
           PERFORM VARYING CB FROM 1 BY 1 UNTIL CB > CBTR-ENTRIES
                                        OR MESSAGE-STORED-TRUE
             IF CBTR-USED (CB) = ON-VALUE AND
                CBTR-OBJECT-NAME (CB) = WEH1-OBJECTNAME AND
                CBTR-OBJECT-TYPE (CB) = WEH1-OBJECTTYPE THEN
* BEFORE RECORD FOUND, GET DATA BACK INTO MESSAGE AND FREE TABLE
* ENTRY
               MOVE CBTR-EVENT-MSG (CB) TO MESSAGE-DATA
               SET  MESSAGE-STORED-TRUE TO TRUE
               MOVE OFF-VALUE           TO CBTR-USED (CB)
               MOVE SPACES              TO CBTR-OBJECT-NAME (CB)
               MOVE SPACES              TO CBTR-OBJECT-TYPE (CB)
               MOVE SPACES              TO CBTR-EVENT-MSG   (CB)
               MOVE SPACES              TO CBTR-PUTDATE     (CB)
               MOVE SPACES              TO CBTR-PUTTIME     (CB)
             END-IF
           END-PERFORM
```

```
* BEFORE CHANGE MESSAGE NOT FOUND, SO PRINT ALL WE GOT
           IF MESSAGE-STORED-FALSE THEN
             MOVE    W-ERROR-9             TO W-SYSPRINT-DATA
             PERFORM PRINT-LINE
             MOVE    W-EVENT-HEADER-1      TO W-SYSPRINT-DATA
             PERFORM PRINT-LINE
             SET     PRINT-ATTRIBUTE-2     TO TRUE
             PERFORM OBJECT-DETAILS
             SET     PRINT-ATTRIBUTE-RESET TO TRUE
           ELSE
* THE CHANGE AFTER MESSAGE IS ALREADY IN ATTRIBUTE TABLE ENTRY
* 2, SO WE HAVE TO PERFORM THIS AGAIN TO PUT THE BEFORE CHANGE
* MESSAGE TO TABLE ELEMENT 1.
             PERFORM EVENT-HEADER
             SET     DATA-PTR          TO ADDRESS OF MESSAGE-DATA
             ADD     MQCFH-STRUCLENGTH TO DATA-PTR-NUM
             PERFORM EVENT-ATTRIBUTES MQCFH-PARAMETERCOUNT TIMES
* NOW DO THE OUTPUT
             MOVE    W-EVENT-HEADER-1 TO W-SYSPRINT-DATA
             PERFORM PRINT-LINE
             PERFORM OBJECT-DETAILS
             ADD     1                TO MSGCOUNTER-DISPLAYED
           END-IF
           PERFORM RESET-TABLES
         END-IF
       END-IF
     END-IF.
 PROCESS-EVENT-MESSAGE-END.
     EXIT.
* ------------------------------------------------------------ *
* PROCESS A SINGLE OBJECT ATTRIBUTE
* ------------------------------------------------------------ *
 EVENT-ATTRIBUTES SECTION.
     SET ADDRESS OF LNK-HFELD TO DATA-PTR.
* CHECK ATTRIBUTE TYPE
     EVALUATE LNK-HFELD
       WHEN MQCFT-BYTE-STRING
          PERFORM ATTR-BYTE-STRING
       WHEN MQCFT-INTEGER
          PERFORM ATTR-INTEGER
       WHEN MQCFT-STRING-LIST
          PERFORM ATTR-STRING-LIST
       WHEN MQCFT-STRING
          PERFORM ATTR-STRING
       WHEN OTHER
          MOVE    '|||       UNKNOWN ATTRIBUTE TYPE' TO
                                                 W-SYSPRINT-DATA
          PERFORM PRINT-LINE
          MOVE    LNK-HFELD TO W-SYSPRINT-DATA
          PERFORM PRINT-LINE
```

```
     END-EVALUATE.
 EVENT-ATTRIBUTES-END.
     EXIT.
* ------------------------------------------------------------- *
* BYTE STRING ATTRIBUTE NOT IMPLEMENTED
* ------------------------------------------------------------- *
 ATTR-BYTE-STRING SECTION.
     SET ADDRESS OF LNK-MQCFBS TO DATA-PTR.
* JUST SKIP OVER THE FIELD AND RETURN
     ADD MQCFBS-STRUCLENGTH TO DATA-PTR-NUM.
 ATTR-BYTE-STRING-END.
     EXIT.
     EJECT
```

Editor's note: this article will be continued next month.

Stefan Raabe
Systems Programmer (Germany)

July 2001–June 2005 index

Below is an index of all topics covered in *MQ Update* since Issue 25, July 2001. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon's Web site (www.xephon.com/mq).

MQ news

IBM has released Version 6.0 of WebSphere MQ. It contains 150 technical improvements, including improved support for creating an Enterprise Service Bus (ESB), which is indispensable to an SOA. This ESB will be based on the Eclipse open source Java development platform.

WebSphere MQ 6.0 allows a user to work across multiple computing platforms. The software includes facilities to connect to Oracle, SAP, and Siebel Systems applications. Mainframe systems such as CICS and IMS can be integrated as well. Now, SOAP-based messaging is tightly integrated with the rest of MQ functionality. The size of the queues in which messages can be stored has been increased from 1GB to 4GB.

The new release also includes support for FTP file transfers, meaning WebSphere MQ will check to see whether the recipient is ready to receive a file and obtain confirmation of receipt after it's sent.

IBM also is offering an updated Express version of its WebSphere Business Integration Server for small and mid-size businesses. It includes new adapters for integration with existing applications and makes greater use of wizard-driven business rules for governing processes between applications.

For further information contact your local IBM representative:
URL: www-306.ibm.com/software/integration/wmq/.

* * *

IBM has announced WebSphere Application Server (WAS) Version 6 for IBM eServer zSeries z/OS. It includes new features such as a common code base, mainframe processor optimization, and enhanced software configuration management, which are designed to help speed up application development and deployment. IBM suggests that WAS 6 can cut mainframe application development by up to 25 percent.

The latest release is optimized to take advantage of the zSeries Application Assist Processor (zAAP). Other new capabilities include: self-healing and protection capabilities (the software is designed to allow users to save and process Web-based business transactions); reduced development costs (including a new 'wizards'-based, drag-and-drop environment); improved scaling and migration; enhanced support for Web services and SOA.

For further information contact your local IBM representative:
URL: www-1.ibm.com/servers/eserver/zseries/zos/zos_sods.html.

* * *

Covast has announced Covast Map Accelerator for WebSphere, which automates the migration of legacy EDI systems to the WebSphere platform, allowing for quick integration of EDI-based information with other critical enterprise applications. As a result, enterprises using EDI for business transactions are able to optimize integration benefits inherent in the WebSphere environment, such as Business Activity Monitoring (BAM) and Business Processes Management (BPM).
For further information contact:

URL: www.covast.com/products/map_accelerator_websphere.asp.