74

MQ update

August 2005

In this issue

A xephon 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.

How and why to do asynchronous messaging on WebSphere MQ

Application integration is a major focus for enterprises having numerous ‘siloed’ applications in a distributed IT environment that were never meant to interoperate with each other. To power this integration effort, approximately 60–70% of all larger organizations use WebSphere MQ, the most dominant message-oriented middleware for achieving integration.

“A large majority of these organizations are using WebSphere MQ for synchronous communications”, said Mark Cresswell, CEO of NEON Systems, a systems integrator and middleware supplier. “These organizations are building request/reply applications using MQ as a transport mechanism. There are many more efficient and cost-effective ways to synchronously link applications than WebSphere MQ. The reality is, organizations can achieve greater advantage in integration and application processing by adopting a strategy of using asynchronous messaging with WebSphere MQ.”

This article discusses the advantages of using asynchronous messaging for more effective system integration and application performance in the WebSphere MQ environment. It illustrates how asynchronous messaging works in applications integration, reviewing set-up considerations for asynchronous messaging and presenting several application examples that demonstrate how asynchronous messaging provides healthy investment returns for both system integration and application processing. The article concludes with a series of best practices for implementing asynchronous messaging.

SYNCHRONOUS VERSUS ASYNCHRONOUS MESSAGING

When enterprises employ synchronous messaging in their application environments, system architects, performance tuners, and capacity planners all have to design for expected

peak loads. The software architect has to build internal data structures that can grow to support the maximum expected peak transaction arrival rate. These structures are typically memory resident. They are susceptible to growth, which can lead to virtual storage constraints and possible system failure because of storage unavailability. In the capacity planning area, hardware resources have to be sufficient to not cause significant queueing under peak transaction volumes. Performance tuning professionals have to be concerned with applications performance, since there is the inherent risk that some applications will 'sit and wait' while others are processing or handling error exceptions.

Asynchronous messaging takes the load off the application when it comes to reading, processing, and writing messages, because it utilizes WebSphere MQ queues for those messages that are able to operate independently of the applications using them. With asynchronous transactional messaging, you are building a log with 'units of work'. A unit of work fits within one memory cycle. It comprises the message read, the processing of the message, and the message write.

"Asynchronous messaging has a significant inherent benefit to performance/capacity optimization", said Gregg Willhoit, Chief Software Architect for NEON Systems. "With asynchronous messaging, the complexity of system design is reduced – along with the innate vicissitudes of trying to size one's hardware based upon the constantly changing target of peak transaction volume. The queue itself, combined with asynchronous messaging, allows the system design to be built for a mean transaction arrival rate. If the arrival rate of the transactions and messages to the queue manager is greater than the capacity of the application pulling the messages off the queue, the messages are simply stored in the queue until the application server is able to 'catch up'. In this example, the queues can be thought of as 'buffers' in the historical sense, in that they provide the capability for devices which operate at different internal speeds to communicate with each other reliably."

Since messages can be read from, processed, and written to queues, and then subsequently stored in these queues until needed, significant burden is removed from the applications using these queues. This is because portions of the internal message handling logic no longer need to be resident in the applications themselves. Relief from the tasks of message monitoring and status handling removes the application developer from 'deep-level' integration that can slow projects, introduce error, and complicate application maintenance.

"By way of contrasting synchronous and asynchronous integration, e-mail makes for a useful analogy", says Cresswell. "If you send someone an e-mail, when the recipient looks at the message, he will typically leave the message in the inbox as a reminder until the work has been completed, since it may not be possible to satisfy the work request immediately. The e-mail software will typically provide several layers of notification back to the sender, such as read-receipt, that indicate some degree of progress in the process. In the event the request cannot be satisfied, the recipient will communicate that fact back to the sender, or perhaps to another individual. You can be certain also that the sender is not sitting waiting for a response. If the request had such synchronous requirements, the sender would have used the telephone instead."

Cresswell goes even further. "There are no circumstances where MQ should be used for synchronous communications. Synchronous communications are time-dependent. The calling application will have been designed to wait. As in the e-mail analogy, there are many superior alternatives designed specifically for synchronous communications."

With asynchronous messaging, each given message can also be assigned a service level for processing. If a message is not processed in a certain timeframe, the message can be deleted or a 'failure processing' application notified. Since MQ loosely couples applications together, this can be very valuable because the application logic doesn't have to be concerned about what is happening with the message. If you don't use

this technique, the responsibility of message tracking becomes that of the application programmer instead of the software infrastructure, and service level management considerations have to be included with the business logic.

PREPARING TO USE ASYNCHRONOUS MESSAGING ON WMQ

Organizations using asynchronous messaging over WebSphere MQ can simply put a message on a queue and forget about it. This is possible since asynchronous messaging systems understand that downstream application responses are also handled asynchronously. In asynchronous messaging systems, responses are reported back to a different application on the originating system. This mechanism more closely mirrors human integration of disparate work processes.

Conversion to asynchronous message processing on WebSphere MQ requires planning, system environment revisions, and adaptation of the application development methodology. This can initially discourage sites from making the move to asynchronous messaging. However, IBM and others have, over the past five years, invested R&D into supporting technologies that facilitate asynchronous message processing. This makes the transition easier.

“Asynchronous messaging requires that an infrastructure layer exists that will read a message off a queue, process that message, and write a message back to the same or a different queue, all as part of the same unit of work”, said Mark Cresswell. “On mainframes, the ability to do this emerged when in 2000 IBM started enabling WebSphere MQ to work with the z/OS Resource Recovery Services (RRS), which was first implemented in 1999. Today, CICS, IMS, Adabas, DB2, and IDMS have all been adapted to support RRS. RRS can be configured to take full responsibility for transactional integrity on the mainframe, and IBM has gone to great lengths to support asynchronous message processing with WebSphere MQ through its integration with RRS.”

Cresswell acknowledges that few sites are truly exploiting asynchronous messaging, because asynchronous messaging can be tricky to conceptualize from a development standpoint. Furthermore, the available tooling and infrastructure software is limited.

“In addition to WebSphere MQ, the various subsystems on the mainframe, which host the applications to be synchronously integrated, all need to be configured to use RRS – which requires fairly extensive planning”, said Cresswell. “In many environments, CICS is the primary transaction manager, and all transactions are journalled by CICS itself. If you have Version 1.3 of CICS TS or later, you can allow RRS to manage that journalling. This is a big change, and some sites are reluctant to take it on. Even with the subsystems correctly configured, only half the story has been addressed. Without the right infrastructure layer, applications programmers are required to code directly to the appropriate RRS APIs as part of the CICS or IMS MQ bridge layer in order to link the whole process together transactionally, in addition to the existing bridge burden of mapping and data type translation.”

APPLICATIONS THAT BENEFIT FROM ASYNCHRONOUS MESSAGING ON WEBSPHERE MQ

Many applications can benefit from applying the principles of asynchronous messaging on WebSphere MQ, but those serving mobile communications environments, highly complex transaction processing, and Web services environments are among the best examples.

“A real world example of asynchronous messaging in a highly-complex transaction environment is a financial services company”, said Gregg Willhoit. “Customers have complex portfolios, which have hundreds, if not thousands, of different instruments like bonds, stocks, options/futures, etc. An asynchronous messaging application can initiate a request for all of the instruments to be analysed. This request can then

spawn thousands of concurrent messages to many different platforms. Each message is processed separately and returned to the application requestor for final aggregation. Once aggregation is complete, the result is then returned to the customer. The customer's request is asynchronous, and their resources are free to work on other tasks while the complex data analysis is being done concurrently on a variety of platforms."

The independence from resource utilization and timing in the asynchronous messaging technique also brings stability and reliability to the world of mobile communications.

Asynchronous messaging with guaranteed delivery is the only platform delivering a 100% reliable interface for mobile computing. The nature of mobile applications is such that the longer a connection must be maintained, the greater the probability of failure. Asynchronous messaging eliminates the need for a connection of significant duration. Mobile applications also benefit from the publish/subscribe capability inherent with asynchronous messaging. Since the publish/subscribe model is topic based rather than recipient based, mobile users can be continuously updated based on the topics that they've subscribed to.

A third application area that greatly benefits from asynchronous messaging is Web services applications. WS applications are superior to HTML-based applications because they offer a loosely-coupled interface, which offers greater interoperability. This combines with a loosely-coupled message architecture that is based on asynchronous messaging, which offers a 'best of breed' solution. "Web services provide discoverable, self-describing, platform-independent messaging for the ultimate in interoperability", said Gregg Willhoit. "This, combined with the loosely-coupled architecture of asynchronous messaging, provides guaranteed delivery, increased scalability, and platform independence, since applications communicate with queues and not with other applications."

A fourth application area that benefits from asynchronous messaging is workflow applications. These applications have to be able to communicate over heterogeneous networks, which connect systems that may or may not be available. They also require a loosely-coupled architecture with asynchronous messaging.

BEST PRACTICES FOR ASYNCHRONOUS MESSAGING

For sites making the move to asynchronous message processing using WebSphere MQ, there is a series of 'best practice' steps that can be taken that will ease the transition.

Important strides can be taken if administrative and regulatory policies and procedures are defined and instituted in advance for systems and applications. NEON's Gregg Willhoit recommends the following to sites:

- Use naming conventions and categories.
- Reflect corporate procedures as message queueing workflows.
- Establish auditing points in workflows to ensure adherence to corporate and regulatory requirements.
- Use the MQSTR format string to allow messages to move easily from one platform to another.
- Use **Resolve** with the backout option for *Sender channel in doubt*.
- Standardize message descriptor use.
- Establish maximum message length standards.
- Use uniform queue definition parameters.
- Standardize dead letter queue handling and definitions.
- Standardize the retransmission of messages via an external messaging API, which is made available to all customers.

CONCLUSION

From the standpoint of application interoperability, time independence for highly-complex and mobile transactions, ease of integration and maintenance in application development, and IBM technology support, asynchronous messaging in the WebSphere MQ environment is a robust but currently under-exploited technique in most enterprise IT environments. There is understandable trepidation in migrating to an asynchronous messaging approach, since any movement to async entails extensive planning and revision to mainframe subsystems, the use of the correct APIs, and changes to the application development process itself. However, infrastructure software technology is now appearing that eliminates many of the barriers to asynchronous messaging.

At the same time, IT managers are being continually pressed by their upper management to deliver interoperability for siloed applications that were never designed to work with other business software. “Wiring these applications together with synchronous messaging is not a good idea since the approach requires deep, costly, labour-intensive integration that results in a fragile infrastructure, which must be over specified from a capacity standpoint”, said NEON’s Mark Cresswell. “Why not use asynchronous messaging over WebSphere MQ to provide a more robust, reliable, and cost-effective integration environment throughout the enterprise?”

Mary E Shacklett
President
Transworld Data (USA)

JMS: a high-level guide to what it is and is not

This article is aimed at those professionals who have heard of JMS and want to know more, at a high level. It will also be useful for those contemplating introducing a JMS application into a proprietary IBM MQSeries/WebSphere MQ (referred to

as MQ throughout this document) environment. I suspect that those with an operational or architectural interest would benefit most.

I will cover three broad areas. The first is around a common, yet subtle, misconception about JMS interoperability. Then I talk about the JMS standard itself, and end with a discussion of messaging bridges.

In a later article I will cover the subject of JNDI, another Java API, closely associated with JMS. I suggest that configuring a JNDI environment can be a significant barrier to JMS adoption or experimentation.

JMS INTEROPERABILITY

During my work with JMS, I have encountered the same pre-conceived idea repeatedly. I must also admit that I held the same conception myself initially. The idea is that JMS messages are interoperable, out of the box, so to speak. People are surprised to learn that JMS, say on a WebLogic Application Server, cannot exchange messages with a JMS application on a non-WebLogic Server. Indeed, 'bridging' between messaging providers is not a new concept, certainly for BEA users who have the MQ Bridge for translating between Weblogic JMS and MQ messages. But surely a homogenous JMS environment, regardless of provider, doesn't need bridges too? After all, it is the JMS standard on all machines. Wrong!

As with any standard you can use the 'pure' standard, adhering to strict common classes and methods, or a provider's adulterated 'enhancements'. However, JMS, even if coded in a pure fashion, does not interoperate across providers.

No, unfortunately, the standard does not touch upon interoperability from the perspective we are looking at, which is from the Message Oriented Middleware (MOM) level. The perspective of the standard is towards the application.

One term that we need to understand, going forward in this discussion, is 'provider'. The JMS standard talks about the

JMS provider. Quoting from the standard (V1.1), the provider is described as follows:

1.2.3.1 JMS Provider

As noted earlier, a JMS provider is the entity that implements JMS for a messaging product.

Ideally, JMS providers will be written in 100% Pure Java so they can run in applets; simplify installation; and, work across architectures and OS's.

An important goal of JMS is to minimize the work needed to implement a provider.

MQ, via its JMS libraries, is able to act as a JMS provider.

THE JMS STANDARD

To help understand this apparent anomaly of lack of message interoperability we need to look at the motivation for the JMS standard. JMS sprung from the Java standard, clearly. As such it is written with J2EE programmers (Java 2 Enterprise Edition) in mind, rather than MQ or other MOM professionals, even though many MOM providers were involved in the specification process. When we realize this starting point, it becomes clear that the promise of JMS is not one of 'provider' interoperability, but rather one of application interoperability. In this sense it is a very useful standard. To quote a well-known TV commercial in the UK, "It does what it says on the tin". Here is the relevant section of the standard:

1.2.3 JMS Objectives

JMS defines a common set of enterprise messaging concepts and facilities. It attempts to minimize the set of concepts a Java language programmer must learn to use enterprise messaging products. It strives to maximize the portability of messaging applications.

The JMS API is an API for accessing enterprise messaging systems from Java programs.

With respect to messaging interoperability between JMS providers, however, the standard is very thin in detail. Hence the need for bridges becomes apparent. The requirement to write your own bridge is tedious, if not problematic, and not often what an organization would choose to do if there was an off-the-shelf, third-party product instead. I am not aware of any such product, but believe there is a market for such a thing.

JMS is nothing more than an abstraction layer – it is similar to an abstract class and an interface class in Java from the perspective of the application. The standard is a specification only. It defines 'what', not 'how'. In the Java language context, it is a contract, not an obligation. Therefore, to refer to a JMS application is accurate, but referring to a JMS MOM system is perhaps misleading. It is more accurate to refer to the provider's JMS implementation – such as BEA WebLogic JMS, IBM WAS JMS, JBoss JMS, WebSphere MQ JMS, rather than simply JMS.

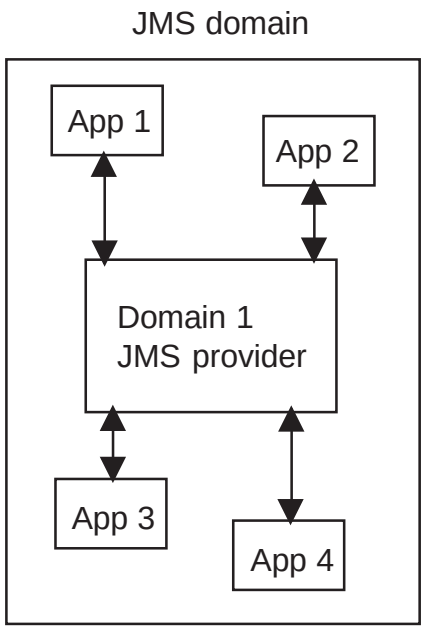

Figure 1: A single JMS provider domain

Therefore an application written in JMS is portable across all JMS providers, provided the pure standard has been followed. The application does not know, or care, what or who the provider is. So the provider can change without any changes to the code at all. However, across the enterprise, all JMS applications must use the same provider if they want messages to interoperate, or else provide a bridge(s).

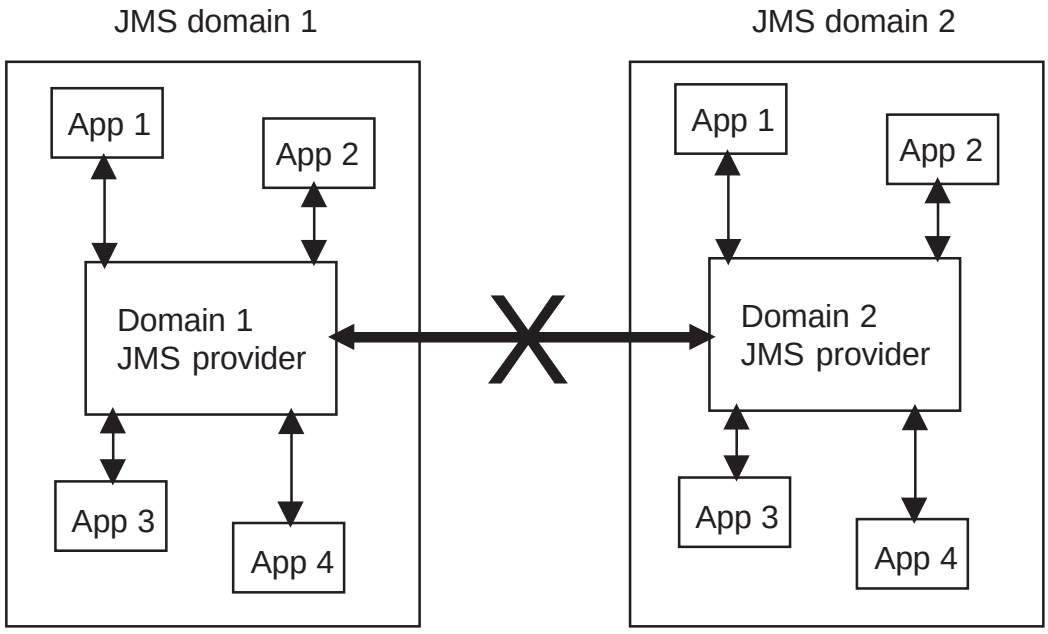

Figure 2: The problem with multiple JMS providers

Let us give an example. A fictional retailer has a back-office HR system that runs within a JBoss application server. The HR application uses JMS to communicate with another JBoss-hosted application for purchasing. Now, let us say that a new system is required for CRM and is to be hosted on a WebSphere Application Server environment. The CRM system will not be able to exchange messages with the JBoss applications without bridging software.

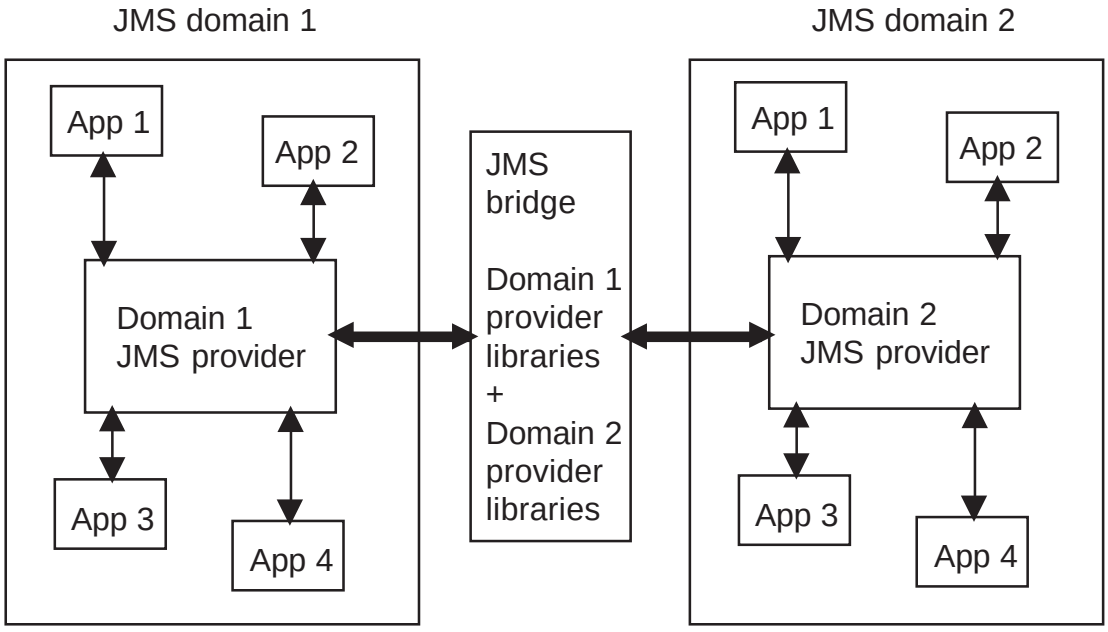

Figure 3: The solution to the multiple JMS providers problem

MESSAGE BRIDGES

A bridge is a software solution to the JMS provider interoperability issue. In a nutshell this code will connect to the two (or more) providers and exchange messages using a common medium, such as the bridge program's memory, to store the intermediate messages. Therefore, the memory representation of the JMS messages becomes the true 'common' producer and consumer messaging architecture.

Let us look at a few Figures to help illustrate the point.

In Figure 1 we can see a number of applications, all of which operate in the same JMS provider domain. For example Domain 1 could be BEA WebLogic Application Server.

Let us now look at the interoperability space.

Figure 2 illustrates the problem with interoperation. It is not possible to translate between the two domains directly. A bridge is required to map between the two environments.

To move JMS messages between different provider domains, a software bridge is needed. The bridge is nothing more than code that utilizes each domain provider's JMS libraries, enabling the messages to be read from one domain, into memory usually, and then written out to another domain. It is an arbitrary decision as to where the bridge sits in the system topology. It is as valid to have the bridge invoked by Message Driven Beans (MDBs) within each domain, or have a central system waiting on specific topics and queues. As with most things, the best solution is dependent on the goals of the overall architecture. In the absence of a common implementation between providers, it would also be feasible for organizations to implement a bridge backbone that can manage any-to-any interoperation. In such a scenario, each application's provider could bridge to the single backbone provider (many-to-one) and the bridge backbone would provide the one-to-many mappings.

SUMMARY

JMS is a very useful standard and one that should be in all MQ professionals' toolkits.

Remember, though, that it is first a Java specification, then a MOM specification – not the other way around. The providers or implementers of the JMS specification do not need to interoperate. So if you find you need to interoperate, you will require a software bridge.

Neil Taylor
Senior Consultant
CommerceQuest (UK)

Getting started with configuration event messages – part 3

This month we conclude the code to capture configuration event messages.

```
*    PRINT THE HEADER LINES
     MOVE    W-HEADER-3 TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-HEADER-4 TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-HEADER-5C TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-HEADER-5A TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-HEADER-5D TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-HEADER-5R TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE "<NONE>" TO WH6-SKIP.
     IF SKIP-USER-ON THEN
       STRING W-USER-1 DELIMITED BY SPACE
              " "      DELIMITED BY SIZE
              W-USER-2 DELIMITED BY SPACE
              " "      DELIMITED BY SIZE
              W-USER-3 DELIMITED BY SPACE
              " "      DELIMITED BY SIZE
              W-USER-4 DELIMITED BY SPACE
              " "      DELIMITED BY SIZE
              W-USER-5 DELIMITED BY SPACE
       INTO WH6-SKIP
     END-IF.
     MOVE    W-HEADER-6 TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     IF SKIP-OBJ-ON THEN
       PERFORM VARYING I FROM 1 BY 1 UNTIL I > W-OBJSPTR
         MOVE    W-OBJSKIPTYPE(I) TO WH8-SKIPTYPE
         MOVE    W-OBJSKIPNAME(I) TO WH8-SKIPNAME
         MOVE    W-HEADER-8 TO W-SYSPRINT-DATA
         PERFORM PRINT-LINE
* BUILD TABLE FOR COMPARE (NEED TO KNOW WHETHER AN OBJECT
* WAS SPECIFIED GENERIC OR NOT FOR CORRECT COMPARE)
         UNSTRING W-OBJSKIPNAME(I) DELIMITED BY " " OR "*"
           INTO W-OBJCOMP-NAME(I)
           DELIMITER IN W-OBJCOMP-DELIMITER(I)
           COUNT IN W-OBJCOMP-COUNT(I)
```

```
            MOVE W-OBJSKIPTYPE(I) TO W-OBJCOMP-TYPE(I)
          END-PERFORM
        ELSE
          MOVE    "<NONE>"   TO WH8-SKIPTYPE
          MOVE    "<NONE>"   TO WH8-SKIPNAME
          MOVE    W-HEADER-8 TO W-SYSPRINT-DATA
          PERFORM PRINT-LINE
        END-IF.
        MOVE    W-HEADER-7 TO W-SYSPRINT-DATA
        PERFORM PRINT-LINE.
        MOVE    SPACES     TO W-SYSPRINT-DATA
        PERFORM PRINT-LINE.
        MOVE    W-HEADER-2 TO W-SYSPRINT-DATA.
        PERFORM PRINT-LINE.
        MOVE    SPACES     TO W-SYSPRINT-DATA
        PERFORM PRINT-LINE.
      * THATS IT FOR INPUT PARMS
       PARM-INPUT-END.
        EXIT.
        EJECT
      * INITIALIZE ATTRIBUTE TABLES
       INIT-TABLES SECTION.
      * INTEGER ATTRIBUTES
        PERFORM VARYING I FROM 1 BY 1 UNTIL I > 2
        MOVE MQIA-AUTH-INFO-TYPE      TO         AI-ID(I, 1)
        MOVE 'AUTHTYPE            '   TO       AI-DESC(I, 1)
        MOVE MQIA-CF-LEVEL            TO         AI-ID(I, 2)
        MOVE 'CVLEVEL             '   TO       AI-DESC(I, 2)
        MOVE MQIA-CF-RECOVER          TO         AI-ID(I, 3)
        MOVE 'RECOVERY            '   TO       AI-DESC(I, 3)
        MOVE MQIACH-CHANNEL-TYPE      TO         AI-ID(I, 4)
        MOVE 'CHLTYPE             '   TO       AI-DESC(I, 4)
        MOVE MQIACH-XMIT-PROTOCOL-TYPE TO        AI-ID(I, 5)
        MOVE 'TRPTYPE             '    TO      AI-DESC(I, 5)
        MOVE MQIACH-BATCH-SIZE        TO         AI-ID(I, 6)
        MOVE 'BATCHSZ             '   TO       AI-DESC(I, 6)
        MOVE MQIACH-DISC-INTERVAL     TO         AI-ID(I, 7)
        MOVE 'DISCINT             '   TO       AI-DESC(I, 7)
        MOVE MQIACH-SHORT-RETRY       TO         AI-ID(I, 8)
        MOVE 'SHORTRTY            '   TO       AI-DESC(I, 8)
        MOVE MQIACH-SHORT-TIMER       TO         AI-ID(I, 9)
        MOVE 'SHORTTMR            '   TO       AI-DESC(I, 9)
        MOVE MQIACH-LONG-RETRY        TO         AI-ID(I, 10)
        MOVE 'LONGRTY             '   TO       AI-DESC(I, 10)
        MOVE MQIACH-LONG-TIMER        TO         AI-ID(I, 11)
        MOVE 'LONGTMR             '   TO       AI-DESC(I, 11)
        MOVE MQIACH-DATA-CONVERSION   TO         AI-ID(I, 12)
        MOVE 'CONVERT             '   TO       AI-DESC(I, 12)
        MOVE MQIACH-PUT-AUTHORITY     TO         AI-ID(I, 13)
        MOVE 'PUTAUT              '   TO       AI-DESC(I, 13)
```

```
MOVE MQIACH-SEQUENCE-NUMBER-WRAP TO      AI-ID(I, 14)
MOVE 'SEQWRAP              '      TO    AI-DESC(I, 14)
MOVE MQIACH-MAX-MSG-LENGTH    TO        AI-ID(I, 15)
MOVE 'MAXMSGL              '   TO       AI-DESC(I, 15)
MOVE MQIACH-MCA-TYPE          TO        AI-ID(I, 16)
MOVE 'MCATYPE              '   TO       AI-DESC(I, 16)
MOVE MQIACH-BATCH-INTERVAL    TO        AI-ID(I, 17)
MOVE 'BATCHINT             '   TO       AI-DESC(I, 17)
MOVE MQIACH-HB-INTERVAL       TO        AI-ID(I, 18)
MOVE 'HBINT                '   TO       AI-DESC(I, 18)
MOVE MQIACH-NPM-SPEED         TO        AI-ID(I, 19)
MOVE 'NPMSPEED             '   TO       AI-DESC(I, 19)
MOVE MQIACH-NETWORK-PRIORITY  TO        AI-ID(I, 20)
MOVE 'NETPRTY              '   TO       AI-DESC(I, 20)
MOVE MQIACH-BATCH-HB          TO        AI-ID(I, 21)
MOVE 'BATCHB               '   TO       AI-DESC(I, 21)
MOVE MQIACH-KEEP-ALIVE-INTERVAL TO      AI-ID(I, 22)
MOVE 'KAINT                '     TO     AI-DESC(I, 22)
MOVE MQIACH-SSL-CLIENT-AUTH   TO        AI-ID(I, 23)
MOVE 'SSLCAUTH             '   TO       AI-DESC(I, 23)
MOVE MQIA-NAMELIST-TYPE       TO        AI-ID(I, 24)
MOVE 'NLTYPE               '   TO       AI-DESC(I, 24)
MOVE MQIA-NAME-COUNT          TO        AI-ID(I, 25)
MOVE '*NAMECOUNT           '   TO       AI-DESC(I, 25)
MOVE MQIA-APPL-TYPE           TO        AI-ID(I, 26)
MOVE 'APPLTYPE             '   TO       AI-DESC(I, 26)
MOVE MQIA-Q-TYPE              TO        AI-ID(I, 27)
MOVE '*QTYPE               '   TO       AI-DESC(I, 27)
MOVE MQIA-INHIBIT-GET         TO        AI-ID(I, 28)
MOVE 'GET                  '   TO       AI-DESC(I, 28)
MOVE MQIA-INHIBIT-PUT         TO        AI-ID(I, 29)
MOVE 'PUT                  '   TO       AI-DESC(I, 29)
MOVE MQIA-DEF-PRIORITY        TO        AI-ID(I, 30)
MOVE 'DEFPRTY              '   TO       AI-DESC(I, 30)
MOVE MQIA-DEF-PERSISTENCE     TO        AI-ID(I, 31)
MOVE 'DEFPSIST             '   TO       AI-DESC(I, 31)
MOVE MQIA-MAX-Q-DEPTH         TO        AI-ID(I, 32)
MOVE 'MAXDEPTH             '   TO       AI-DESC(I, 32)
MOVE MQIA-MAX-MSG-LENGTH      TO        AI-ID(I, 33)
MOVE 'MAXMSGL              '   TO       AI-DESC(I, 33)
MOVE MQIA-BACKOUT-THRESHOLD   TO        AI-ID(I, 34)
MOVE 'BOTHRESH             '   TO       AI-DESC(I, 34)
MOVE MQIA-SHAREABILITY        TO        AI-ID(I, 35)
MOVE 'SHARE                '   TO       AI-DESC(I, 35)
MOVE MQIA-DEF-INPUT-OPEN-OPTION TO      AI-ID(I, 36)
MOVE 'DEFSOPT              '     TO     AI-DESC(I, 36)
MOVE MQIA-HARDEN-GET-BACKOUT  TO        AI-ID(I, 37)
MOVE 'HARDENBO NOHARDENBO '   TO       AI-DESC(I, 37)
MOVE MQIA-MSG-DELIVERY-SEQUENCE TO      AI-ID(I, 38)
MOVE 'MSGDLVSQ             '     TO     AI-DESC(I, 38)
```

```
MOVE MQIA-RETENTION-INTERVAL  TO        AI-ID(I, 39)
MOVE 'RETINTVL            '   TO      AI-DESC(I, 39)
MOVE MQIA-DEFINITION-TYPE     TO        AI-ID(I, 40)
MOVE '*DEFINITIONTYPE     '   TO      AI-DESC(I, 40)
MOVE MQIA-USAGE               TO        AI-ID(I, 41)
MOVE 'USAGE               '   TO      AI-DESC(I, 41)
MOVE MQIA-TRIGGER-CONTROL     TO        AI-ID(I, 42)
MOVE 'TRIGGERCONTROL      '   TO      AI-DESC(I, 42)
MOVE MQIA-TRIGGER-TYPE        TO        AI-ID(I, 43)
MOVE 'TRIGTYPE            '   TO      AI-DESC(I, 43)
MOVE MQIA-TRIGGER-MSG-PRIORITY TO       AI-ID(I, 44)
MOVE 'TRIGMPRI            '    TO     AI-DESC(I, 44)
MOVE MQIA-TRIGGER-DEPTH       TO        AI-ID(I, 45)
MOVE 'TRIGDPTH            '   TO      AI-DESC(I, 45)
MOVE MQIA-Q-DEPTH-HIGH-LIMIT  TO        AI-ID(I, 46)
MOVE 'QDEPTHHI            '   TO      AI-DESC(I, 46)
MOVE MQIA-Q-DEPTH-LOW-LIMIT   TO        AI-ID(I, 47)
MOVE 'QDEPTHLO            '   TO      AI-DESC(I, 47)
MOVE MQIA-Q-SERVICE-INTERVAL  TO        AI-ID(I, 48)
MOVE 'QSVCINT             '   TO      AI-DESC(I, 48)
MOVE MQIA-DEF-BIND            TO        AI-ID(I, 49)
MOVE 'DEFBIND             '   TO      AI-DESC(I, 49)
MOVE MQIA-INDEX-TYPE          TO        AI-ID(I, 50)
MOVE 'INDXTYPE            '   TO      AI-DESC(I, 50)
MOVE MQIA-PLATFORM            TO        AI-ID(I, 51)
MOVE '*PLATFORM           '   TO      AI-DESC(I, 51)
MOVE MQIA-COMMAND-LEVEL       TO        AI-ID(I, 52)
MOVE '*COMMANDLEVEL       '   TO      AI-DESC(I, 52)
MOVE MQIA-CPI-LEVEL           TO        AI-ID(I, 53)
MOVE '*CPILEVEL           '   TO      AI-DESC(I, 53)
MOVE MQIA-IGQ-PUT-AUTHORITY   TO        AI-ID(I, 54)
MOVE 'IGQAUT              '   TO      AI-DESC(I, 54)
MOVE MQIA-INTRA-GROUP-QUEUING TO        AI-ID(I, 55)
MOVE 'IGQ                 '   TO      AI-DESC(I, 55)
MOVE MQIA-EXPIRY-INTERVAL     TO        AI-ID(I, 56)
MOVE 'EXPRYINT            '   TO      AI-DESC(I, 56)
MOVE MQIA-SSL-TASKS           TO        AI-ID(I, 57)
MOVE 'SSLTASKS            '   TO      AI-DESC(I, 57)
MOVE MQIA-CONFIGURATION-EVENT TO        AI-ID(I, 58)
MOVE 'CONFIGEV            '   TO      AI-DESC(I, 58)
MOVE MQIA-TRIGGER-INTERVAL    TO        AI-ID(I, 59)
MOVE 'TRIGINT             '   TO      AI-DESC(I, 59)
MOVE MQIA-MAX-PRIORITY        TO        AI-ID(I, 60)
MOVE 'MAXPRIORITY         '   TO      AI-DESC(I, 60)
MOVE MQIA-CODED-CHAR-SET-ID   TO        AI-ID(I, 61)
MOVE '*CCSID              '   TO      AI-DESC(I, 61)
MOVE MQIA-MAX-HANDLES         TO        AI-ID(I, 62)
MOVE 'MAXHANDS            '   TO      AI-DESC(I, 62)
MOVE MQIA-MAX-UNCOMMITTED-MSGS TO       AI-ID(I, 63)
MOVE 'MAXUMSGS            '    TO     AI-DESC(I, 63)
```

```
           MOVE MQIA-SYNCPOINT           TO         AI-ID(I, 64)
           MOVE '*SYNCPOINT          '   TO       AI-DESC(I, 64)
           MOVE MQIA-AUTHORITY-EVENT     TO         AI-ID(I, 65)
           MOVE 'AUTHOREV            '   TO       AI-DESC(I, 65)
           MOVE MQIA-INHIBIT-EVENT       TO         AI-ID(I, 66)
           MOVE 'INHIBTEV            '   TO       AI-DESC(I, 66)
           MOVE MQIA-LOCAL-EVENT         TO         AI-ID(I, 67)
           MOVE 'LOCALEV             '   TO       AI-DESC(I, 67)
           MOVE MQIA-REMOTE-EVENT        TO         AI-ID(I, 68)
           MOVE 'REMOTEEV            '   TO       AI-DESC(I, 68)
           MOVE MQIA-START-STOP-EVENT    TO         AI-ID(I, 69)
           MOVE 'STRSTPEV            '   TO       AI-DESC(I, 69)
           MOVE MQIA-PERFORMANCE-EVENT   TO         AI-ID(I, 7Ø)
           MOVE 'PERFMEV             '   TO       AI-DESC(I, 7Ø)
           MOVE MQIA-CLUSTER-WORKLOAD-LENGTH TO     AI-ID(I, 71)
           MOVE 'CLWLLEN             '       TO   AI-DESC(I, 71)
           MOVE MQIA-PAGESET-ID          TO         AI-ID(I, 72)
           MOVE 'PSID                '   TO       AI-DESC(I, 72)
           MOVE MQIACF-EVENT-ORIGIN      TO         AI-ID(I, 73)
           MOVE '*EVENTORIGIN        '   TO       AI-DESC(I, 73)
           MOVE MQIACF-EVENT-APPL-TYPE   TO         AI-ID(I, 74)
           MOVE '*EVENTAPPLTYPE      '   TO       AI-DESC(I, 74)
           MOVE MQIACF-OBJECT-TYPE       TO         AI-ID(I, 75)
           MOVE '*OBJECTTYPE         '   TO       AI-DESC(I, 75)
           MOVE MQIA-QSG-DISP            TO         AI-ID(I, 76)
           MOVE 'QSGDISP             '   TO       AI-DESC(I, 76)
           END-PERFORM.
       * NUMBER OF INTEGER ATTRIBUTES
           MOVE 76 TO AI-NUMBER.
       * STRING ATTRIBUTES
           PERFORM VARYING I FROM 1 BY 1 UNTIL I > 2
           MOVE MQCA-AUTH-INFO-DESC              TO     AS-ID(I,  1)
           MOVE 'DESCR               '           TO   AS-DESC(I,  1)
           MOVE MQCA-AUTH-INFO-CONN-NAME         TO     AS-ID(I,  2)
           MOVE 'CONNAME             '           TO   AS-DESC(I,  2)
           MOVE MQCA-LDAP-USER-NAME              TO     AS-ID(I,  3)
           MOVE 'LDAPUSER            '           TO   AS-DESC(I,  3)
           MOVE MQCA-LDAP-PASSWORD               TO     AS-ID(I,  4)
           MOVE 'LDAPPWD             '           TO   AS-DESC(I,  4)
           MOVE MQCA-ALTERATION-DATE             TO     AS-ID(I,  5)
           MOVE '*ALTERATION-DATE    '           TO   AS-DESC(I,  5)
           MOVE MQCA-ALTERATION-TIME             TO     AS-ID(I,  6)
           MOVE '*ALTERATION-TIME    '           TO   AS-DESC(I,  6)
           MOVE MQCA-CF-STRUC-DESC               TO     AS-ID(I,  7)
           MOVE 'DESCR               '           TO   AS-DESC(I,  7)
           MOVE MQCACH-CHANNEL-NAME              TO     AS-ID(I,  8)
           MOVE '*CHANNEL-NAME       '           TO   AS-DESC(I,  8)
           MOVE MQCACH-MODE-NAME                 TO     AS-ID(I,  9)
           MOVE 'MODENAME            '           TO   AS-DESC(I,  9)
           MOVE MQCACH-TP-NAME                   TO     AS-ID(I, 1Ø)
```

```
MOVE 'TPNAME              '          TO   AS-DESC(I, 1Ø)
MOVE MQCA-Q-MGR-NAME                 TO     AS-ID(I, 11)
MOVE 'QMGRNAME            '          TO   AS-DESC(I, 11)
MOVE MQCACH-XMIT-Q-NAME              TO     AS-ID(I, 12)
MOVE 'XMITQ               '          TO   AS-DESC(I, 12)
MOVE MQCACH-CONNECTION-NAME          TO     AS-ID(I, 13)
MOVE 'CONNAME             '          TO   AS-DESC(I, 13)
MOVE MQCACH-MCA-NAME                 TO     AS-ID(I, 14)
MOVE 'MCANAME             '          TO   AS-DESC(I, 14)
MOVE MQCACH-DESC                     TO     AS-ID(I, 15)
MOVE 'DESCR               '          TO   AS-DESC(I, 15)
MOVE  MQCACH-SEC-EXIT-NAME           TO     AS-ID(I, 16)
MOVE 'SCYEXIT             '          TO   AS-DESC(I, 16)
MOVE MQCACH-SEC-EXIT-USER-DATA       TO     AS-ID(I, 17)
MOVE 'SCYDATA             '          TO   AS-DESC(I, 17)
MOVE MQCACH-MCA-USER-ID              TO     AS-ID(I, 18)
MOVE 'MCAUSER             '          TO   AS-DESC(I, 18)
MOVE MQCACH-USER-ID                  TO     AS-ID(I, 19)
MOVE 'USERID              '          TO   AS-DESC(I, 19)
MOVE MQCACH-PASSWORD                 TO     AS-ID(I, 2Ø)
MOVE 'PASSWORD            '          TO   AS-DESC(I, 2Ø)
MOVE MQCA-CLUSTER-NAME               TO     AS-ID(I, 21)
MOVE 'CLUSTER             '          TO   AS-DESC(I, 21)
MOVE MQCA-CLUSTER-NAMELIST           TO     AS-ID(I, 22)
MOVE 'CLUSTNL             '          TO   AS-DESC(I, 22)
MOVE MQCACH-LOCAL-ADDRESS            TO     AS-ID(I, 23)
MOVE 'LOCALADDR           '          TO   AS-DESC(I, 23)
MOVE MQCACH-SSL-CIPHER-SPEC          TO     AS-ID(I, 24)
MOVE 'SSLCIPH             '          TO   AS-DESC(I, 24)
MOVE MQCACH-SSL-PEER-NAME            TO     AS-ID(I, 25)
MOVE 'SSLPEER             '          TO   AS-DESC(I, 25)
MOVE MQCA-NAMELIST-NAME              TO     AS-ID(I, 26)
MOVE '*NAMELISTNAME       '          TO   AS-DESC(I, 26)
MOVE MQCA-NAMELIST-DESC              TO     AS-ID(I, 27)
MOVE 'DESCR               '          TO   AS-DESC(I, 27)
MOVE MQCA-PROCESS-NAME               TO     AS-ID(I, 28)
MOVE '*PROCESSNAME        '          TO   AS-DESC(I, 28)
MOVE MQCA-PROCESS-DESC               TO     AS-ID(I, 29)
MOVE 'DESCR               '          TO   AS-DESC(I, 29)
MOVE MQCA-APPL-ID                    TO     AS-ID(I, 3Ø)
MOVE 'APPLICID            '          TO   AS-DESC(I, 3Ø)
MOVE MQCA-ENV-DATA                   TO     AS-ID(I, 31)
MOVE 'ENVVDATA            '          TO   AS-DESC(I, 31)
MOVE MQCA-USER-DATA                  TO     AS-ID(I, 32)
MOVE 'USERDATA            '          TO   AS-DESC(I, 32)
MOVE MQCA-Q-NAME                     TO     AS-ID(I, 33)
MOVE '*QNAME              '          TO   AS-DESC(I, 33)
MOVE MQCA-Q-DESC                     TO     AS-ID(I, 34)
MOVE 'DESCR               '          TO   AS-DESC(I, 34)
MOVE MQCA-BACKOUT-REQ-Q-NAME         TO     AS-ID(I, 35)
```

```
MOVE 'BOQNAME             '          TO   AS-DESC(I, 35)
MOVE MQCA-CREATION-DATE              TO     AS-ID(I, 36)
MOVE '*CREATION-DATE      '          TO   AS-DESC(I, 36)
MOVE MQCA-CREATION-TIME              TO     AS-ID(I, 37)
MOVE '*CREATION-TIME      '          TO   AS-DESC(I, 37)
MOVE MQCA-INITIATION-Q-NAME          TO     AS-ID(I, 38)
MOVE 'INITQ               '          TO   AS-DESC(I, 38)
MOVE MQCA-TRIGGER-DATA               TO     AS-ID(I, 39)
MOVE 'TRIGDATA            '          TO   AS-DESC(I, 39)
MOVE MQCA-BASE-Q-NAME                TO     AS-ID(I, 40)
MOVE 'TARGQ               '          TO   AS-DESC(I, 40)
MOVE MQCA-REMOTE-Q-NAME              TO     AS-ID(I, 41)
MOVE 'RNAME               '          TO   AS-DESC(I, 41)
MOVE MQCA-REMOTE-Q-MGR-NAME          TO     AS-ID(I, 42)
MOVE 'RQMNAME             '          TO   AS-DESC(I, 42)
MOVE MQCA-XMIT-Q-NAME                TO     AS-ID(I, 43)
MOVE 'XMITQ               '          TO   AS-DESC(I, 43)
MOVE MQCA-STORAGE-CLASS              TO     AS-ID(I, 44)
MOVE 'STGCLASS            '          TO   AS-DESC(I, 44)
MOVE MQCA-CF-STRUC-NAME              TO     AS-ID(I, 45)
MOVE 'CFSTRUCT            '          TO   AS-DESC(I, 45)
MOVE MQCA-Q-MGR-DESC                 TO     AS-ID(I, 46)
MOVE 'DESCR               '          TO   AS-DESC(I, 46)
MOVE MQCA-IGQ-USER-ID                TO     AS-ID(I, 47)
MOVE 'IGQUSER             '          TO   AS-DESC(I, 47)
MOVE MQCA-SSL-KEY-REPOSITORY         TO     AS-ID(I, 48)
MOVE 'SSLKEYR             '          TO   AS-DESC(I, 48)
MOVE MQCA-SSL-CRL-NAMELIST           TO     AS-ID(I, 49)
MOVE 'SSLCRLNL            '          TO   AS-DESC(I, 49)
MOVE MQCA-QSG-NAME                   TO     AS-ID(I, 50)
MOVE '*QSGNAME            '          TO   AS-DESC(I, 50)
MOVE MQCA-DEAD-LETTER-Q-NAME         TO     AS-ID(I, 51)
MOVE 'DEADQ               '          TO   AS-DESC(I, 51)
MOVE MQCA-COMMAND-INPUT-Q-NAME       TO     AS-ID(I, 52)
MOVE '*COMMANDINPUTQUEUE  '          TO   AS-DESC(I, 52)
MOVE MQCA-DEF-XMIT-Q-NAME            TO     AS-ID(I, 53)
MOVE 'DEFXMITQ            '          TO   AS-DESC(I, 53)
MOVE MQCA-CHANNEL-AUTO-DEF-EXIT      TO     AS-ID(I, 54)
MOVE 'CHADEXIT            '          TO   AS-DESC(I, 54)
MOVE MQCA-CLUSTER-WORKLOAD-EXIT      TO     AS-ID(I, 55)
MOVE 'CLWLEXIT            '          TO   AS-DESC(I, 55)
MOVE MQCA-CLUSTER-WORKLOAD-DATA      TO     AS-ID(I, 56)
MOVE 'CLWLDATA            '          TO   AS-DESC(I, 56)
MOVE MQCA-Q-MGR-IDENTIFIER           TO     AS-ID(I, 57)
MOVE '*QMGRIDENTIFIER     '          TO   AS-DESC(I, 57)
MOVE MQCA-REPOSITORY-NAME            TO     AS-ID(I, 58)
MOVE 'REPOS               '          TO   AS-DESC(I, 58)
MOVE MQCA-REPOSITORY-NAMELIST        TO     AS-ID(I, 59)
MOVE 'REPOSNL             '          TO   AS-DESC(I, 59)
MOVE MQCA-STORAGE-CLASS-DESC         TO     AS-ID(I, 60)
```

```
     MOVE 'DESCR               '           TO   AS-DESC(I, 6Ø)
     MOVE MQCA-XCF-GROUP-NAME              TO     AS-ID(I, 61)
     MOVE 'XCFGNAME            '           TO   AS-DESC(I, 61)
     MOVE MQCA-XCF-MEMBER-NAME             TO     AS-ID(I, 62)
     MOVE 'XCFMNAME            '           TO   AS-DESC(I, 62)
     MOVE MQCACF-EVENT-USER-ID             TO     AS-ID(I, 63)
     MOVE '*EVENTUSERID        '           TO   AS-DESC(I, 63)
     MOVE MQCACF-EVENT-Q-MGR               TO     AS-ID(I, 64)
     MOVE '*EVENTQMGR          '           TO   AS-DESC(I, 64)
     MOVE MQCACF-EVENT-APPL-IDENTITY       TO     AS-ID(I, 65)
     MOVE '*EVENTAPPLIDENTITY  '           TO   AS-DESC(I, 65)
     MOVE MQCACF-EVENT-APPL-NAME           TO     AS-ID(I, 66)
     MOVE '*EVENTAPPLNAME      '           TO   AS-DESC(I, 66)
     MOVE MQCACF-EVENT-APPL-ORIGIN         TO     AS-ID(I, 67)
     MOVE '*EVENTAPPLORIGIN    '           TO   AS-DESC(I, 67)
     MOVE MQCA-AUTH-INFO-NAME              TO     AS-ID(I, 68)
     MOVE '*AUTHINFONAME       '           TO   AS-DESC(I, 68)
     END-PERFORM.
* NUMBER OF STRING ATTRIBUTES
     MOVE 68 TO AS-NUMBER.
* STRING-LIST ATTRIBUTES
     PERFORM VARYING I FROM 1 BY 1 UNTIL I > 2
     MOVE MQCACH-MSG-EXIT-NAME             TO     ASL-ID(I,  1)
     MOVE 'MSGEXIT             '           TO   ASL-DESC(I,  1)
     MOVE MQCACH-SEND-EXIT-NAME            TO     ASL-ID(I,  2)
     MOVE 'SENDEXIT            '           TO   ASL-DESC(I,  2)
     MOVE MQCACH-RCV-EXIT-NAME             TO     ASL-ID(I,  3)
     MOVE 'RCVEXIT             '           TO   ASL-DESC(I,  3)
     MOVE MQCACH-MSG-EXIT-USER-DATA        TO     ASL-ID(I,  4)
     MOVE 'MSGDATA             '           TO   ASL-DESC(I,  4)
     MOVE MQCACH-SEND-EXIT-USER-DATA       TO     ASL-ID(I,  5)
     MOVE 'SENDDATA            '           TO   ASL-DESC(I,  5)
     MOVE MQCACH-RCV-EXIT-USER-DATA        TO     ASL-ID(I,  6)
     MOVE 'RCVDATA             '           TO   ASL-DESC(I,  6)
     MOVE MQCA-NAMES                       TO     ASL-ID(I,  7)
     MOVE 'NAMES               '           TO   ASL-DESC(I,  7)
     END-PERFORM.
* NUMBER OF STRING LIST ATTRIBUTES
     MOVE  7 TO ASL-NUMBER.
* THIS IS FOR TEST ONLY, CHECK THAT THERE ARE NO DUPLICATE
* ENTRIES IN THE ATTRIBUTE TABLES WHICH MAY BE CAUSED BY TYPING
* ERRORS. REMOVE IF YOU LIKE.
     PERFORM CHECK-TABLES.
* INITIALIZE TABLE TO REMEMBER BEFORE-CHANGE EVENT RECORDS
     PERFORM VARYING CB FROM 1 BY 1 UNTIL CB > CBTR-ENTRIES
        MOVE OFF-VALUE TO CBTR-USED (CB)
        MOVE SPACES    TO CBTR-OBJECT-NAME (CB)
                          CBTR-OBJECT-TYPE (CB)
                          CBTR-EVENT-MSG  (CB)
     END-PERFORM.
```

```
 INIT-TABLES-END.
     EXIT.
     EJECT
*  CHECK FOR DUPLICATE ENTRIES IN TABLES IN THE ATTRIBUTE TABLES
*  WHICH MAY BE CAUSED BY TYPING ERRORS. REMOVE IF YOU LIKE
 CHECK-TABLES SECTION.
     MOVE 1 TO I.
* CALL CHECK INTEGER TABLE
     PERFORM CHECK-TABLE-INTEGER
             VARYING J FROM 1 BY 1 UNTIL J > AI-NUMBER
              AFTER  K FROM 1 BY 1 UNTIL K > AI-NUMBER.
* CALL CHECK STRING TABLE
     PERFORM CHECK-TABLE-STRING
             VARYING J FROM 1 BY 1 UNTIL J > AS-NUMBER
              AFTER  K FROM 1 BY 1 UNTIL K > AS-NUMBER.
* CALL CHECK STRINGLIST TABLE
     PERFORM CHECK-TABLE-STRINGLIST
             VARYING J FROM 1 BY 1 UNTIL J > ASL-NUMBER
              AFTER  K FROM 1 BY 1 UNTIL K > ASL-NUMBER.
 CHECK-TABLES-END.
     EXIT.
* CHECK FOR DUPLICATE INTEGER ENTRIES
 CHECK-TABLE-INTEGER SECTION.
        IF J = K THEN
          NEXT SENTENCE
        ELSE
          IF AI-ID (I, J) = AI-ID (I, K) THEN
            MOVE 'DUBLICATE INTEGER TABLE ENTRY ' TO
                                      W-SYSPRINT-DATA
            PERFORM PRINT-LINE
            MOVE    AI-DESC (I, J) TO W-SYSPRINT-DATA
            PERFORM PRINT-LINE
            MOVE    AI-DESC (I, K) TO W-SYSPRINT-DATA
            PERFORM PRINT-LINE
          END-IF
        END-IF.
 CHECK-TABLE-INTEGER-END.
     EXIT.
* CHECK FOR DUPLICATE STRING ENTRIES
 CHECK-TABLE-STRING SECTION.
        IF J = K THEN
           NEXT SENTENCE
        ELSE
           IF AS-ID (I, J) = AS-ID (I, K) THEN
              MOVE    'DUBLICATE STRING TABLE ENTRY ' TO
                                        W-SYSPRINT-DATA
              PERFORM PRINT-LINE
              MOVE    AS-DESC (I, J) TO W-SYSPRINT-DATA
              PERFORM PRINT-LINE
              MOVE    AS-DESC (I, K) TO W-SYSPRINT-DATA
```

```
              PERFORM PRINT-LINE
           END-IF
        END-IF.
 CHECK-TABLE-STRING-END.
     EXIT.
* CHECK FOR DUPLICATE STRINGLIST ENTRIES
 CHECK-TABLE-STRINGLIST SECTION.
        IF J = K THEN
           NEXT SENTENCE
        ELSE
           IF ASL-ID (I, J) = ASL-ID (I, K) THEN
              MOVE    'DUBLICATE STRINGLIST TABLE ENTRY ' TO
                                          W-SYSPRINT-DATA
              PERFORM PRINT-LINE
              MOVE    ASL-DESC (I, J) TO W-SYSPRINT-DATA
              PERFORM PRINT-LINE
              MOVE    ASL-DESC (I, K) TO W-SYSPRINT-DATA
              PERFORM PRINT-LINE
           END-IF
        END-IF.
 CHECK-TABLE-STRINGLIST-END.
     EXIT.
* CLEAR ATTRIBUTES IN OBJECT TABLES
 RESET-TABLES SECTION.
* CALL TO RESET INTEGER TABLE
      PERFORM RESET-TABLE-INTEGER
              VARYING I FROM 1 BY 1 UNTIL I > 2
               AFTER  J FROM 1 BY 1 UNTIL J > AI-NUMBER.
* CALL TO RESET STRING TABLE
      PERFORM RESET-TABLE-STRING
              VARYING I FROM 1 BY 1 UNTIL I > 2
               AFTER  J FROM 1 BY 1 UNTIL J > AS-NUMBER.
* CALL TO RESET STRINGLIST TABLE
      PERFORM RESET-TABLE-STRINGLIST
              VARYING I FROM 1 BY 1 UNTIL I > 2
               AFTER  J FROM 1 BY 1 UNTIL J > ASL-NUMBER.
 RESET-TABLES-END.
     EXIT.
* RESET INTEGER TABLE ELEMENT
 RESET-TABLE-INTEGER SECTION.
        SET AI-NOT-SET (I, J) TO TRUE.
        MOVE ZERO   TO AI-VALUE (I, J).
 RESET-TABLE-INTEGER-END.
     EXIT.
* RESET STRING TABLE ELEMENT
 RESET-TABLE-STRING SECTION.
        SET AS-NOT-SET (I, J) TO TRUE.
        MOVE SPACES TO AS-VALUE (I, J).
        MOVE ZERO   TO AS-LENGTH (I, J).
 RESET-TABLE-STRING-END.
```

```
     EXIT.
* RESET STRINGLIST TABLE ELEMENT
 RESET-TABLE-STRINGLIST SECTION.
        SET ASL-NOT-SET (I, J) TO TRUE.
        MOVE SPACES TO ASL-VALUE1 (I, J).
        MOVE ZERO   TO ASL-LENGTH (I, J).
        MOVE ZERO   TO ASL-COUNT  (I, J).
 RESET-TABLE-STRINGLIST-END.
     EXIT.
     EJECT
* CONNECT TO THE SPECIFIED QUEUE MANAGER.
 MQ-CONN SECTION.
     CALL 'MQCONN' USING WH3-QMGR
                         HCONN
                         COMPCODE
                         REASON.
     IF (COMPCODE NOT = MQCC-OK) THEN
        MOVE    'CONNECT'  TO WE4-TYPE
        MOVE    COMPCODE   TO WE4-COMPCODE
        MOVE    REASON     TO WE4-REASON
        MOVE    W-ERROR-4  TO W-SYSPRINT-DATA
        PERFORM PRINT-LINE
        MOVE    W-RC-ERROR TO W-RETURN-CODE
        GO TO   MAIN-END
     END-IF.
 MQ-CONN-END.
     EXIT.
     EJECT
* OPEN EVENT QUEUE
 MQ-OPEN SECTION.
     MOVE MQOT-Q               TO MQOD-OBJECTTYPE.
     MOVE WH4-EVENT-QUEUE-NAME TO MQOD-OBJECTNAME.
* BROWSE OR GET ?
     IF BROWSE-ON THEN
       MOVE MQOO-BROWSE       TO OPTIONS
     ELSE
       MOVE MQOO-INPUT-SHARED TO OPTIONS
     END-IF.
* OPEN THE QUEUE.
     CALL 'MQOPEN' USING HCONN
                         MQOD
                         OPTIONS
                         HOBJ
                         COMPCODE
                         REASON.
     IF (COMPCODE NOT = MQCC-OK) THEN
        MOVE    'OPEN'     TO WE4-TYPE
        MOVE    COMPCODE   TO WE4-COMPCODE
        MOVE    REASON     TO WE4-REASON
        MOVE    W-ERROR-4  TO W-SYSPRINT-DATA
```

```
           PERFORM PRINT-LINE
           MOVE    W-RC-ERROR TO W-RETURN-CODE
           GO TO   MAIN-DISCONNECT
        END-IF.
 MQ-OPEN-END.
        EXIT.
        EJECT
* CLOSE EVENT QUEUE
 MQ-CLOSE SECTION.
        MOVE MQCO-NONE TO OPTIONS.
        CALL 'MQCLOSE' USING HCONN
                             HOBJ
                             OPTIONS
                             COMPCODE
                             REASON.
        IF (COMPCODE NOT = MQCC-OK) THEN
           MOVE    'CLOSE'    TO WE4-TYPE
           MOVE    COMPCODE   TO WE4-COMPCODE
           MOVE    REASON     TO WE4-REASON
           MOVE    W-ERROR-4  TO W-SYSPRINT-DATA
           PERFORM PRINT-LINE
           MOVE    W-RC-ERROR TO W-RETURN-CODE
        END-IF.
 MQ-CLOSE-END.
        EXIT.
        EJECT
* DISCONNECT FROM THE QUEUE MANAGER
 MQ-DISC SECTION.
        CALL 'MQDISC' USING HCONN
                            COMPCODE
                            REASON.
        IF (COMPCODE NOT = MQCC-OK) THEN
           MOVE    'DISCONNECT' TO WE4-TYPE
           MOVE    COMPCODE     TO WE4-COMPCODE
           MOVE    REASON       TO WE4-REASON
           MOVE    W-ERROR-4    TO W-SYSPRINT-DATA
           MOVE    W-RC-ERROR   TO W-RETURN-CODE
           PERFORM PRINT-LINE
        END-IF.
 MQ-DISC-END.
        EXIT.
        EJECT
* VARIOUS PRINT SUBROUTINES
* PRINT A LINE
 PRINT-LINE SECTION.
        MOVE W-SYSPRINT-DATA TO SYSPRINT-DATA.
        WRITE SYSPRINT-REC   AFTER ADVANCING 1.
 PRINT-LINE-END.
        EXIT.
* WORK ON HEADER
```

```
 HEADER SECTION.
     MOVE    SPACES     TO W-SYSPRINT-DATA.
     PERFORM PRINT-LINE.
     ACCEPT  WORK-DATE  FROM DATE.
     MOVE    W-MM       TO WH1-MM.
     MOVE    W-DD       TO WH1-DD.
     MOVE    W-YY       TO WH1-YY.
     MOVE    W-HEADER-1 TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-HEADER-2 TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    SPACES     TO W-SYSPRINT-DATA.
     PERFORM PRINT-LINE.
 HEADER-END.
     EXIT.
* WORK ON TRAILER
 TRAILER SECTION.
     MOVE    MSGCOUNTER-READ      TO WT1-MSGCOUNTER-READ
     MOVE    MSGCOUNTER-REFRESH   TO WT11-MSGCOUNTER-REFRESH
     MOVE    MSGCOUNTER-CREATE    TO WT12-MSGCOUNTER-CREATE
     MOVE    MSGCOUNTER-DELETE    TO WT13-MSGCOUNTER-DELETE
     MOVE    MSGCOUNTER-ALTER     TO WT14-MSGCOUNTER-ALTER
     MOVE    MSGCOUNTER-PROCESSED TO WT2-MSGCOUNTER-PROCESSED
     MOVE    W-TRAILER-1          TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-TRAILER-11         TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-TRAILER-12         TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-TRAILER-13         TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-TRAILER-14         TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    SPACES               TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    MSGCOUNTER-DISPLAYED TO WT4-MSGCOUNTER-DISPLAYED
     MOVE    W-TRAILER-4          TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    MSGCOUNTER-SKIPPED   TO WT3-MSGCOUNTER-SKIPPED
     MOVE    W-TRAILER-3          TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
     MOVE    W-TRAILER-2          TO W-SYSPRINT-DATA
     PERFORM PRINT-LINE.
 TRAILER-END.
     EXIT.
* FILL THE EVENT HEADER
 EVENT-HEADER SECTION.
     MOVE MQMD-PUTDATE(1:4)  TO WEH1-YEAR.
     MOVE MQMD-PUTDATE(5:2)  TO WEH1-MONTH.
     MOVE MQMD-PUTDATE(7:2)  TO WEH1-DAY.
```

```
      MOVE MQMD-PUTTIME(1:2)  TO WEH1-HOUR.
      MOVE MQMD-PUTTIME(3:2)  TO WEH1-MINUTE.
      MOVE MQMD-PUTTIME(5:2)  TO WEH1-SECOND.
      MOVE MQCFH-MSGSEQNUMBER TO I W-CHANGE
      EVALUATE MQCFH-REASON
        WHEN MQRC-CONFIG-CHANGE-OBJECT
           SET  OBJECT-CHANGE  TO TRUE
        WHEN MQRC-CONFIG-CREATE-OBJECT
           SET  OBJECT-CREATE  TO TRUE
           ADD  1              TO MSGCOUNTER-CREATE
        WHEN MQRC-CONFIG-DELETE-OBJECT
           SET  OBJECT-DELETE  TO TRUE
           ADD  1              TO MSGCOUNTER-DELETE
        WHEN MQRC-CONFIG-REFRESH-OBJECT
           SET  OBJECT-REFRESH TO TRUE
           ADD  1              TO MSGCOUNTER-REFRESH
         WHEN OTHER
           MOVE '?UNKNOWN'     TO WEH1-ACTION
      END-EVALUATE.
* CHECK THE IMPOSSIBLE (MSG 1 BUT LAST CHANGE RECORD)
      IF OBJECT-CHANGE THEN
        IF MQCFH-MSGSEQNUMBER = 1 AND MQCFH-CONTROL =
                                                 MQCFC-LAST THEN
          MOVE    " ||||| THIS SHOULD NOT HAPPEN " TO
                                                 W-SYSPRINT-DATA
          PERFORM PRINT-LINE
        END-IF
      END-IF.
 EVENT-HEADER-END.
      EXIT.
      EJECT
* CHECK IF A RECORD SHOULD BE SKIPPED BECAUSE OF OBJECT NAME
 OBJ-SKIP-CHECK SECTION.
      PERFORM VARYING SC FROM 1 BY 1 UNTIL SC > W-OBJSPTR
                                     OR    SKIP-OBJECT-TRUE
        IF W-OBJCOMP-DELIMITER(SC) = "*" THEN
          IF W-OBJCOMP-TYPE(SC) = WEH1-OBJECTTYPE AND
                 W-OBJCOMP-NAME(SC) (1:W-OBJCOMP-COUNT(SC)) =
                 WEH1-OBJECTNAME (1:W-OBJCOMP-COUNT(SC)) THEN
             SET SKIP-OBJECT-TRUE  TO TRUE
             ADD 1                 TO MSGCOUNTER-SKIPPED
          END-IF
        ELSE
          IF W-OBJCOMP-TYPE(SC) = WEH1-OBJECTTYPE AND
                 W-OBJCOMP-NAME(SC) = WEH1-OBJECTNAME THEN
             SET SKIP-OBJECT-TRUE  TO TRUE
             ADD 1                 TO MSGCOUNTER-SKIPPED
          END-IF
        END-IF
      END-PERFORM.
```

```
OBJ-SKIP-CHECK-END.
    EXIT.
```


WebSphere Portal installation on z/Linux

This document describes the multi-tier installation of IBM WebSphere Portal Server with various bundled components on a z/Linux platform. The hardware used here is an IBM mainframe (z/800) that is logically partitioned and consists of multiple z/Linux machines acting as guests. Each guest is installed with the SuSE Linux v8 operating system with service pack 3.

We create three Linux guests with identical operating systems and service packs, identified as:

- WPSNODE, which will be loaded with the IBM WebSphere Portal Server (includes WebSphere Application Server). This machine will also be loaded with the client components of DB2 UDB and Tivoli Directory Server.
- DB2NODE, which will be loaded with the DB2 UDB Server and the Tivoli Directory Server.
- IHSNODE, which will be loaded with the IBM HTTP Server and WebSphere Plug-in for HTTP Server.

These z/Linux guest(s) are not accessible directly and have to be accessed by a remote machine (preferably Linux). This remote machine, installed with Red Hat Linux, also acts as an NFS server sharing its CD-ROM drive to install the software. In addition to the three Linux guests, we would have a remote stand-alone Intel-based node with the Linux operating system. This is called NFSNODE, and it is where we will copy all the software to be installed on the z/Linux. We will also use this

remote node to connect to each of the z/Linux machines.

It is assumed that the hardware, network, and operating systems are installed and configured prior to the installation of WebSphere Portal Server.

Tip: it is recommended to add the IP addresses and corresponding hostnames in the */etc/hosts* file of all the nodes.

To facilitate the installation process, it is better to copy the contents of the following CDs to the NFS server and install from the NFS server:

- Create a set-up directory */opt/setup*:

  ```
  # mkdir /opt/setup
  ```

- Insert and copy the contents of the following CDs in the */opt/setup* directory from the WebSphere Portal for Multiplatforms Version 5.0.2.2 pack:
  - Portal Install, Portal Info Center.
  - WebSphere Application Server for Linux for zSeries.
  - WebSphere Application Server Fix Pack for Linux for zSeries.
  - Portal Server and WebSphere Portal Content Publishing.
  - IBM Tivoli Directory Server.
  - DB2 Universal Database Enterprise Edition for Linux on zSeries.
  - DB2 Enterprise Edition Fix Pack for Windows, Linux, and Linux on zSeries.
- Change permissions for the *install.sh* file to make it executable:

  ```
  # chmod 755 /opt/setup/install.sh
  ```

- To share the */opt/setup* directory as the NFS mount point

add the following line in the */etc/exports* file and save the file:

```
/opt/setup       *(rw,sync)
```

- Restart the NFS server:

```
# /etc/init.d/nfs stop
# /etc/init.d/nfs start
```

INSTALL DB2 UDB SERVER

Install DB2 on the DB2NODE using the command line installation procedure.

To install the DB2 UDB Server perform these steps:

1 Log in on SuSE host DB2NODE:

```
# ssh -X DB2NODE
```

2 **Mount** the shared NFS disk on the SuSE Linux:

```
# mount -t nfs NFSNODE:/opt/setup /mnt
```

3 Change to the following directory and execute the install command:

```
# cd /mnt/db2/linux390
# ./db2_install
```

4 When prompted for the product to be installed, type:

```
DB2.ESE
```

5 DB2 will be installed and you can see the file copying process. You will be returned to the command prompt when the installation has completed.

6 (Optionally) check the status of the installation in the log file */tmp/db2_install_log.636*.

7 Change to the following directory and execute the command:

```
# cd /mnt/db2fp/zlinux
# ./installFixPak
```

8 DB2 fixpack will be installed and you can see the file copying process. You will be returned to the command prompt when the installation is completed.

9 (Optionally) you can check the status of the installation in the log file */tmp/installFixPak.tmp1*.

10 Create the following groups and users:

- # groupadd -g 600 db2grp1
- # groupadd -g 601 db2fadm1
- # groupadd -g 602 db2asgrp
- # useradd -u 600 -g db2grp1 -G db2grp1 -d /home/db2admin -m db2admin -p <password>
- # useradd -u 601 -g db2fadm1 -G db2fadm1 -d /home/db2fenc1 -m db2fenc1 -p <password>
- # useradd -u 602 -g db2asgrp -G db2asgrp -d /home/db2as -m db2as -p <password>.

11 Type the following command to create the Administrative instance:

```
# ./opt/IBM/db2/V8.1/instance/dascrt -u db2as
```

Look for the message:

```
SQL4406W The DB2 Administration Server was started successfully.
DBI1070I Program dascrt completed successfully
```

12 To check that the DB2 administration instance was created, type the following command and locate the created instance:

```
# ./opt/IBM/db2/V8.1/instance/daslist
db2as
```

13 Create the DB2 instance by typing the following command:

```
# ./opt/IBM/db2/V8.1/instance/db2icrt -u db2fenc1 db2admin
```

Look for the message:

```
DBI1070I Program db2icrt completed successfully.
```

14 To check that the DB2 instance was created, type the following command and locate the created instance:

```
# ./opt/IBM/db2/V8.1/instance/db2ilist
db2admin
```

15 Update the database manager configuration for the instance created:

```
$ cat /etc/services | grep db2admin
```

You should see output similar to the following at each node:

```
DB2_db2admin    60004/tcp
DB2_db2admin_1  60005/tcp
DB2_db2admin_2  60006/tcp
DB2_db2admin_END        60007/tcp
$ db2 update dbm cfg using SVCENAME 60005
```

16 Add the following line at the start of the */home/db2admin/sqllib/db2profile* file:

```
DB2COMM=TCPIP
```

17 Switch to the DB2 user and install the licence:

```
# su - db2admin
$ db2licm -a /mnt/db2/license/db2ese.lic
```

You should see the 'License Added Successfully' message.

18 **Umount** the mount point on SuSE Linux:

```
# umount /mnt
```

TIVOLI DIRECTORY SERVER INSTALLATION

Since the Tivoli Directory Server also requires DB2 UDB, we prefer to install the Tivoli Directory Server on the DB2NODE and create a separate instance for the LDAP data. Prepare the DB2 node with the following prerequisites before installing the IBM Tivoli Directory Server:

1 Ensure that the IBM Java is installed in the directory */opt/IBMJava2-s390.131*.

2 Since the Tivoli Directory Server looks for the Java in the */usr/lib* directory, create a soft link as follows:

```
# cd /usr/lib
# ln -sf /opt/IBMJava2-s390.131 /usr/lib/IBMJava2-s390.131
```

3 Add the following line at the bottom of the */root/.profile*:

```
PATH=/opt/IBMJava2-s390.131:/usr/lib/IBMJava2-s390.131:$PATH
```

4 Configure JCE extensions for the IBM GSKit:

- Copy the following files from */usr/local/ibm/gsk7/classes/jre/lib/ext* to */usr/lib/IBMJava2-1.3.1/jre/lib/ext*:
  - /usr/lib/IBMJava2-1.3.1/jre/lib/ext/ibmjceprovider.jar
  - /usr/lib/IBMJava2-1.3.1/jre/lib/ext/ibmpkcs.jar
  - /usr/lib/IBMJava2-1.3.1/jre/lib/ext/ibmjcefw.jar
  - /usr/lib/IBMJava2-1.3.1/jre/lib/ext/local_policy.jar
  - /usr/lib/IBMJava2-1.3.1/jre/lib/ext/US_export_policy.jar
  - /usr/lib/IBMJava2-1.3.1/jre/lib/ext/ibmpkcs11.jar.
- Change to the following directory:

  ```
  # cd /usr/lib/IBMJava2-1.3.1/jre/lib/security
  ```

- Back up the *java.security* file:

  ```
  # cp java.security java.security.bak
  ```

- Modify the *java.security* file. Replace the current providers with the following three providers:
  - security.provider.1=sun.security.provider.Sun
  - security.provider.2=com.ibm.spi.IBMCMSProvider
  - security.provider.3=com.ibm.crypto.provider.IBMJCE.

5 To avoid configuration conflicts between the openldap client installed by SUSE LINUX and the Tivoli Directory Server, move some of the openldap-client files to a new directory prior to installing the Tivoli Directory Server. Do not uninstall the openldap client, since shared libraries that depend on the openldap client being installed are required by many other Linux packages.

- Create a directory to back up the openldap-client files. Enter:

```
# mkdir -p /usr/bin/openldapclient
```

- Change to the */usr/bin* directory.
- Move the following openldap-client files found in the */usr/bin* directory to the */usr/bin/openldapclient* directory:

```
# mv ldapdelete ldapmodrdn ldapsearch ldapmodify
                                ldapadd /usr/bin/openldapclient/
```

Install IBM Tivoli Directory Server V5.2

To install the Tivoli Directory Server V5.2 follow these steps:

1 Log in on SuSE host DB2NODE:

```
# ssh -X DB2NODE
```

2 **Mount** the shared NFS disk on the SuSE Linux:

```
# mount -t nfs NFSNODE:/opt/setup /mnt
```

3 Install the packages one by one. Issue the following commands in the listed order:

```
rpm -ivh ldap-clientd-5-2-1.s390.rpm
rpm -ivh ldap-serverd-5-2-1.s390.rpm
rpm -ivh ldap-msg-en_US-5-2-1.s390.rpm
rpm -ivh ldap-html-en_US-5-2-1.s390.rpm
rpm -ivh ldap-webadmind-5-2-1.s390.rpm
```

4 (Optionally) unpack and install the required fixpacks FP520L-01 and FP520L-02:

```
# tar -xvf FP520L-01.tar
# /tmp/tds52.fp1/FP520L-01/install_update.sh
```

```
# tar -xvf FP520L-02.tar
# /tmp/tds52.fp1/FP520L-02/install_update.sh
```

5 To verify that the fixpack(s) are installed successfully, confirm that the following files exist:

```
/usr/ldap/bin/FP520-01.txt
/usr/ldap/bin/FP520-02.txt
```

Note: the *install_update.sh* script saves a back-up of the previous version of the Tivoli Directory Server in the */usr/ldap/data* directory. Do not remove this directory if you intend to uninstall the update.

1 Type the following commands to create the LDAP instance owner user:

```
# useradd -u 600 -g db2grp1 -G db2grp1 -d
                            /home/ldapdb2 -m ldapdb2 -p <password>
```

2 Create the instance by typing the following command:

```
# ./opt/IBM/db2/V8.1/instance/db2icrt -u db2fenc1 ldapdb2
```

Look for the message:

```
DBI1070I Program db2icrt completed successfully.
```

3 To check the DB2 instance created above, type the following command and locate the created instance:

```
# ./opt/IBM/db2/V8.1/instance/db2ilist
ldapdb2
db2admin
```

4 **Umount** the mount point on SuSE Linux:

```
# umount /mnt
```

5 Ensure that the DB2 server is started.

6 Start the configuration tool by entering the following command:

```
# ldapxcfg
```

7 Set the administrator distinguished name (DN) as cn=root and password <password> using the Configuration Tool.

8 Create and configure the Tivoli Directory Server database from the Configuration Tool:

- Under **Choose a task**, select **Configure database**.
- Select **Create a new database** and then click **Next**.
- On the **Configure Database - user ID** page, enter the User ID ldapdb2 and password <password>.
- On the **Configure database - database name** page, enter the database name LDAPDB2, and click **Next**.
- On the **Configure database - database code** page, select **Create a universal DB2 database (UTF-8/ UCS-2)** and click **Next**.
- On the **Configure database - database location** page, browse to the ldapdb2 home directory (*/home/ ldapdb2*) and click **Next**.
- On the **Configuration Summary** page, review the selections and click **Finish**.
- During the configuration, the configuration status is displayed. Notice that a DB2 instance, LDAPDB2, is created with the name of the user designated as the DB2 owner.
- You should see the following messages if the configuration is successful:

```
Configured IBM Tivoli Directory Server Database.
IBM Tivoli Directory Server Configuration complete.
```

- When the configuration is complete, click **Close**.

Preparing LDAP organization structure for WebSphere Portal

After the Tivoli Directory Server is installed and the base configuration is completed, we will need to prepare our organization structure. Perform the following steps:

1 Ensure that the ibmslapd service is not running:

```
# ps -ef | grep ibmslapd
```

This should not return any processes. If it returns a list of ibmslapd processes, kill them by typing this command:

```
# killall -9 ibmslapd
```

2 Create an LDAP suffix in the IBM Tivoli Directory Server:

```
# ldapcfg -s "o=company, o=com"
```

Where *o= company, o=com* can be replaced by the suffix you choose for your LDAP directory structure.

3 The portal installation CD provides a sample ldif file, *PortalUsers.ldif*, to import groups and users required by WebSphere Portal. Copy this file in the */tmp* directory of the LDAP server. Modify this file and import Portal groups and users.

Optionally you may also create a new file, *PortalUsers.ldif*, in */tmp* with this content:

```
# NOTE: you must edit this file before importing it and replace all
# occurrences of the default suffix "dc=yourco,dc=com" with the suffix
# that your LDAP server is configured for.

dn: o=company, o=com
objectclass: domain
objectclass: top
# Add lines according to this scheme that corresponds to your suffix
dc: o=company, o=com

dn: cn=users, o=company, o=com
objectclass: container
objectclass: top
cn: users

dn: cn=groups, o=company, o=com
objectclass: top
objectclass: container
cn: groups

dn: uid=wpsadmin,cn=users, o=company, o=com
objectclass: organizationalPerson
objectclass: person
objectclass: top
objectclass: inetOrgPerson
uid: wpsadmin
```

```
userpassword: wpsadmin
sn: wpsadmin
givenName: wpsadmin
cn: wpsadmin

dn: uid=wpsbind,cn=users, o=company, o=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: wpsbind
userpassword: wpsbind
sn: wpsbind
givenName: wpsbind
cn: wpsbind

dn: cn=wpsadmins,cn=groups, o=company, o=com
objectclass: groupOfUniqueNames
objectclass: top
uniquemember: uid=wpsadmin,cn=users, o=company, o=com
cn: wpsadmins
```

4 Save this file as */tmp/PortalUsers.ldif*.

5 Start the Tivoli Directory Server Configuration Tool:

```
# ldapxcfg
```

6 Select **Import LDIF data** and on the **Import LDIF Data** page, complete the following tasks:

- For path and LDIF file name, type:

  ```
  /tmp/PortalUsers.ldif
  ```

- Select **Standard impor**t.
- Click **Import** at the bottom of the page.

7 After the import finishes successfully, a message is displayed that reports how many entries have been imported into the directory server. You should see six entries imported successfully.

8 When the import is complete, click **Clos**e and restart LDAP service:

```
# ibmslapd
```

Notice the following start-up processes:

```
Server starting.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type EXTENDEDOP is successfully loaded from libldaprepl.so.
Plugin of type PREOPERATION is successfully loaded from libDSP.so.
Plugin of type PREOPERATION is successfully loaded from libDigest.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type AUDIT is successfully loaded from /lib/libldapaudit.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type DATABASE is successfully loaded from /lib/libback-
rdbm.so.
Plugin of type REPLICATION is successfully loaded from /lib/
libldaprepl.so.
Plugin of type EXTENDEDOP is successfully loaded from /lib/libback-
rdbm.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type DATABASE is successfully loaded from /lib/libback-
config.so.
Anonymous binds will be allowed.
Server configured for secure connections.
Configuration read securePort 636.
Plugin of type EXTENDEDOP is successfully loaded from libloga.so.
Configuration file successfully read.
Non-SSL port initialized to 389.
SSL port initialized to 636.
```

WEBSPHERE PORTAL SERVER INSTALLATION

Installing the IBM WebSphere Portal Server V5.0.2.2 on the WPSNODE will also install the following underlying software components:

- IBM HTTP Server.
- IBM WebSphere Application Server Enterprise (with underlying base software).
- Cumulative fixes for WebSphere Application Server – base and enterprise.
- IBM WebSphere Portal Content Publisher.

Before you start the installation, verify the following

prerequisites to ease the installation process:

- At least 2.5GB of free disk space is available for installation.
- Groups *mqm* and *mqbrkrs* are created.
- User *mqm* is created and is added to the group *mqm*.
- User root is added to the groups *mqm* and *mqbrkrs*.

Installation procedure

Once these prerequisites are verified, you may start the installation on the WPSNODE machine. Do the following to install the software:

1 Log in on SuSE host WPSNODE:

```
# ssh -X WPSNODE
```

2 **Mount** the shared NFS disk on the SuSE Linux:

```
# mount -t nfs NFSNODE:/opt/setup /mnt
```

3 Change to the following directory and execute the install command:

```
# cd /mnt
# ./install.sh
```

4 You will get the following and will be returned to the command prompt:

```
InstallShield Wizard
Initializing InstallShield Wizard...

Searching for Java(tm) Virtual Machine ...
```

5 You will be returned to the command prompt. Wait until you get the Installation Wizard to appear at this time. (You may have to wait for 5–10 minutes, depending on the machine's resources.)

6 Click English (default) for the language; click **Next** and **Accept Licence Agreement**;click **Next** and choose **Custom install**.

7 Select **Install a new instance of WebSphere Application Server** and click **Next**.

8 Accept default paths for WebSphere Application Server, HTTP Server, and WebSphere Portal Server installations and click **Next**.

9 On the screen for WebSphere Portal Administrator user id and password enter wpsadmin for user id, password <password>, and confirm password fields and click **Next**.

10 File copying will start. Depending on the system resources, file copying may take a minimum of 180 minutes.

11 After the installation has completed, register the product and click on **Finish**.

12 **Umount** the mount point on SuSE Linux:

```
# umount /mnt
```

Test WebSphere Portal Server installation

After the basic installation is complete, the portal server is started automatically after the install. Check the status of the application server WebSphere_Portal by typing the following command:

```
# cd /opt/WebSphere/AppServer/bin
# ./serverStatus.sh WebSphere_Portal
```

Notice the status of the server running.

If the status of the server shows stopped, start the WebSphere_Portal server by typing the command:

```
# ./startServer.sh WebSphere_Portal
```

After the WebSphere_Portal server is started, access the login page of the Portal on the embedded http server port 9081:

1 Open an Internet Explorer or Netscape browser and enter the URL http://WPSNODE.company.com:9081/wps/portal.

2 Verify that the Portal home page is shown correctly.

3 On the top-right corner, click on **Login**.

4 Enter the user id wpsadmin and password <password>.

5 Verify that the login is completed.

6 Click on **logout** on the right top corner of the Portal page.

Installation of DB2 client on Portal machines

We have the database server on a remote machine for the portal repository. The database client software needs to be installed on the WebSphere Portal machine to communicate with the remote database server and the databases required by WebSphere Portal must be created manually before performing the WebSphere Portal database transfer to work with DB2.

To install the DB2 UDB Client carry out these steps on the 'WPSNODE' machine:

1 Log in on SuSE host WPSNODE:

```
# ssh -X WPSNODE
```

2 **Mount** the shared NFS disk on the SuSE Linux:

```
# mount -t nfs NFSNODE:/opt/setup /mnt
```

3 Change to the following directory and execute the install command:

```
# cd /mnt/db2/linux390
# ./db2_install
```

4 When prompted for the product to be installed, type:

```
DB2.ADC
```

5 The DB2 Client will be installed and you can see the file copying process.

6 (Optionally) open the DB2 installation log to follow the installation's progress and identify any errors while installing.

7 You will be returned to the command prompt when the installation has completed.

8 Proceed to install the DB2 fixpack:

```
# cd /mnt/db2fp/zlinux
# ./installFixPak
```

9 The DB2 fixpack will be installed and you can see the file copying process.

10 (Optionally) open the DB2 fixpack installation log to follow the installation's progress and identify any errors while installing. Type the following command to check the install log:

```
tail -f /tmp/installFixPak.tmp1
```

11 You will be returned to the command prompt when the installation is completed.

12 **Umount** the mount point on SuSE Linux.

13 Type the following commands to create groups and users:

```
# groupadd -g 600 db2grp1
# useradd -u 600 -g db2grp1 -G db2grp1 -d
                            /home/db2admin -m db2admin -p <password>
```

14 Type the following command to create the DB2 instance:

```
# ./opt/IBM/db2/V8.1/instance/db2icrt -u db2fenc1 db2admin
```

This will create the instance environment.

15 Check the ports used on the DB2NODE for the instance:

```
# rsh DB2NODE "cat /etc/services | grep db2admin"
```

You should see output similar to the following at each node:

```
DB2_db2admin    60004/tcp
DB2_db2admin_1  60005/tcp
DB2_db2admin_2  60006/tcp
DB2_db2admin_END        60007/tcp
```

16 Copy the lines shown above (this has to be entered in */etc/services* of the WPSNODE machine).

17 Paste the copied lines in the */etc/services* file in the WPSNODE.

18 Catalog the db2admin instance created at the DB2 server node on the Portal Server so that the Portal Server can connect to the db2admin instance:

```
# su - db2admin
$ db2 catalog tcpip node DBSERVER remote DB2NODE server 60005
```

where

- *DBSERVER* is the node name that we gave to the instance db2admin.
- *DB2NODE* is the host name of the DB2 Server.
- *60005* is the port on which the db2admin instance runs on the DB2 server.

Installation of LDAP client on Portal machines

Since the LDAP server is installed on a remote machine, we will install the LDAP client on the WebSphere Portal machine WPSNODE.

To install the IBM directory client with fixpack, perform the following steps:

1 Log in on SuSE host:

```
# ssh -X WPSNODE
```

2 Ensure that the IBM Java is installed in the directory */opt/IBMJava2-s390.131*.

3 Since the IBM Tivoli Directory Server looks for the Java in the */usr/lib* directory, create a soft link for IBM Java as follows:

```
# cd /usr/lib
# ln -sf /opt/IBMJava2-s390.131 /usr/lib/IBMJava2-s390.131
```

4 Add the following line at the bottom of the */root/.profile*:

```
PATH=/opt/IBMJava2-s390.131:/usr/lib/IBMJava2-s390.131:$PATH
```

5 **Mount** the shared NFS disk on the SuSE Linux:

```
# mount -t nfs NFSNODE:/opt/setup /mnt
```

6 Install the Client package by typing the following command:

```
rpm -ivh ldap-clientd-5-2-1.s390.rpm
```

7 Unpack and install the required fixpacks FP520L-01 and FP520L-02:

```
# tar -xvf FP520L-01.tar
# /tmp/tds52.fp1/FP520L-01/install_update.sh
# tar -xvf FP520L-02.tar
# /tmp/tds52.fp1/FP520L-02/install_update.sh
```

8 To verify that the fixpack(s) are installed successfully, confirm that the following files exist:

```
/usr/ldap/bin/FP520-01.txt
/usr/ldap/bin/FP520-02.txt
```

Note: the *install_update.sh* script saves a back-up of the previous version of the Tivoli Directory Server in the */usr/ldap/data* directory. Do not remove this directory if you intend to uninstall the update.

9 **Umount** the mount point on SuSE Linux:

```
# umount /mnt
```

Editor's note: this article will be concluded next month when we look at IBM HTTP Server installation.

Hozefa Rupawala (UAE)

WebSphere lite?

A little while ago (May 2005) IBM bought a company called Gluecode Software. The question is, why? And what's it got to do with WebSphere?

IBM has dabbled in the Open Source community before –

which it claims promotes the adoption of open standards – with the acquisition of Eclipse, Derby, and the Apache HTTP server (httpd).

Gluecode ships JOE, a Java-certified application stack for the Apache Geronimo application server. It's middleware software that looks like a cut-down version of WebSphere and is ideal for smaller companies. It is a direct competitor to JBoss.

So what was IBM's thinking? Why did they decide to buy Gluecode?

It means that companies that are starting small in terms of their middleware requirements will be able to get a product from IBM. And IBM hopes that as those companies' needs grow, so will their likelihood of upgrading to WebSphere, which has greater capabilities and a higher cost.

It also gives IBM the opportunity to try subscription-based software pricing, which Gluecode has used for some time. Users download the software and then pay an annual fee for support.

It also gives IBM a fighting chance to compete against the open source JBoss – which is very popular and doesn't have all the fancy stuff found in WebSphere, but gets the job done. And JBoss usage is growing at an alarming rate (alarming if you're IBM – that is!). Some people, of course, might ask why IBM didn't go for JBoss; perhaps they did, and Gluecode was second best. Who knows?

It looks like IBM will use Apache Geronimo, the J2EE server from Apache, to fill out the low end of its WebSphere line and make it the open source application server of the future, with JOE as its WebSphere Lite product. They'll hope to gain extra revenue as companies' needs grow and they upgrade to WebSphere itself.

Nick Nourse
Independent Consultant (UK)

Nastel has announced that its AutoPilot/WMQ has been designed to work with the new Version 6 of WebSphere MQ.

AutoPilot/WMQ provides end-to-end, real-time application process monitoring in a distributed application environment. Its modules monitor and control WMQ middleware and application interaction to optimize operations and eliminate redundancies and bottlenecks, they claim. Also, there are tools to help identify problems, which is useful in troubleshooting situations.

For further information contact:
URL: www.nastel.com/news/new_pr43.shtml.

* * *

DST Systems has announced that IBM has validated its Automated Work Distributor as "Ready for WebSphere", which means that AWD can be integrated with IBM's WebSphere application server technology.

AWD is a Business Process Management (BPM) solution. With the certification, AWD can function as a component of the WebSphere middleware stack, thereby providing WebSphere customers with access to AWD BPM components, and AWD customers will be able to take advantage of the WebSphere infrastructure and better interoperate with other applications in a J2EE environment.

For further information contact:
URL: www.awdbpm.com.

* * *

Lattice3D has announced that its ultra-compressed 3D XVL file format and Player have been integrated by IBM into IBM WebSphere Product Center. Lattice3D and the IBM WebSphere Product Center team will market their respective integrated solutions.

This combination enables interactive 2D and 3D user selection and procurement of 3D products, assemblies and parts lists. Lattice3D develops and markets 3D communication and publishing applications to enable the rapid use of 3D data assets downstream of design. 3D CAD models and designs can be used directly in print and online user and employee manuals, parts lists, maintenance instructions, assembly instructions, IETMs, etc.

For further information contact:
URL: www.lattice3d.com/press_websphere060205.htm.

* * *

Following IBM's acquisition of Ascential at the end of April, Ascential has been showing off its current product line. The prediction is that the products will simply have the word 'WebSphere' inserted in front of their name when IBM sells them.

The new product, code-named Hawk, redesigns most of Ascential's core products, turning them into a new bundle that is integrated to a common metadata back end and a common front end. The result is that users will be able to work off common information on how data is being moved, transformed, and integrated.

It also enables users to work on data transformations, cleansing, and profiling without having to switch to different tools. And Hawk adds various point capabilities, such as the ability of all the Hawk tools to read or write data in XML, receive or pass messages using SOAP or JMS, and expose data transformation services as WSDL or EJBs.

For further information contact:
URL: www.ascential.com.