MQSeries for AS/400 V5.1

White Paper

Bruce Wassell
IBM MQSeries Technical Strategy
17 March 2000

Abstract

MQSeries for AS/400 V5.1 provides significant new function over previous releases, including support for multiple queue managers, MQSeries clusters and substantially increased message and queue sizes. Enabling this new function has involved some changes to the internal architecture of MQSeries: for example this release makes substantial use of the Integrated File System whereas, previously, all MQSeries data was stored in OS/400 objects in libraries.

These changes mean that in many respects this version can be considered a new product, internally much closer to MQSeries on other platforms than hitherto, rather than a simple enhancement of the previous release. As a consequence some external features have also changed, the most noticeable being a new security model and some necessary modifications to some Control Language commands. Every effort has been made to ensure that existing user programs written to use the MQSeries Interface (MQI) will execute without need for recompilation, however this does not hold true for user exit programs: these will need recompilation and, in some case, some code modification.

Customers that make heavy use of MQSeries on the AS/400 need to be aware that the performance profiles of individual MQI functions have changed with this release. In some cases the performance has improved, in some cases it has degraded. The overall effect will clearly depend upon how applications make use of particular MQ functions, but we anticipate that, in some cases, additional system processor resources will be consumed by existing applications.

For many existing users of the MQSeries for AS/400 product, migration to this new version should be relatively straight forward and a migration tool is provided to automate much of the process. For those making extensive use of the product, or having large numbers of users with individual authorizations to MQSeries objects, the process may be more complex and involve some manual changes. In all cases migration will require special consideration and careful planning.

Introduction

This new version of MQSeries for AS/400 is a substantial enhancement of the product, with a number of new features being added to bring functional parity with the MQSeries V5.1 UNIX products. In addition, some frequently requested AS/400-only functions have been added. Naturally these enhancements involve no compromise to the core design philosophy of all MQSeries products: assured, once-only message delivery and a high degree of platform inter-operability and portability of application source code.

Providing this additional functionality and achieving the parity with the UNIX platforms would have been extremely difficult to achieve without making some substantial changes to the internal architecture of MQSeries for AS/400. Prior to V5.1 the product has made heavy use of OS/400 objects such as User Spaces and User Indexes to implement the shared memory constructs needed for MQSeries objects like queues and the catalogue. While providing an application tightly integrated to the operating system, this architecture made some enhancements - such as increased queue and message sizes - difficult to achieve.

Having AS/400-specific features at the heart of the product can also militate against the easy port of functions already implemented in the code base common to the other distributed platforms.

Taking these considerations into account, the decision was made to rewrite some of the core parts of the product based upon the UNIX model. For the first time MQSeries for AS/400 makes use of a number of UNIX-like, or POSIX, features of OS/400 such as the root of IFS, kernel threads, shared memory and teraspace.

This made the development of V5.1 a major undertaking, far more so than the usual release-to-release upgrade. The potential benefits to the MQSeries for AS/400, in the short term - substantially new function - and for the future - keeping in step with enhancements for the other distributed products - were significant.

The principal design aims for V5.1 of MQSeries for AS/400 were:

- **Achieve parity of base function with the MQSeries UNIX V5.1 platforms**

  This has been frequently requested by our customers and it brings many of the most keenly desired enhancements.

- **Satisfy additional customer requirements**

  For example, to improve the exploitation of OS/400 Work Management functions, allowing users to set, for example, to the run-time priority and time slice for MQSeries jobs.

- **Increase commonality in the MQSeries code base across the distributed MQSeries platforms**

  This will makes it easier for us to bring future improvements for distributed MQSeries to the AS/400 in a timely manner.

- **Retain the best of the current product**

  We did not believe that existing users would be happy to lose the existing MQSeries CL commands with the built-in prompting and context-sensitive help text, to see them replaced by, for example, a more rudimentary UNIX-style interface. Consequently, most of the old commands are still there, enhanced where necessary, together with a number of new ones, for example to support MQSeries clusters.

  We also decided to retain the use of OS/400 Journals to handle MQSeries logging. As tightly integrated OS/400 objects, Journals provide high reliability and excellent performance for this key MQSeries function.

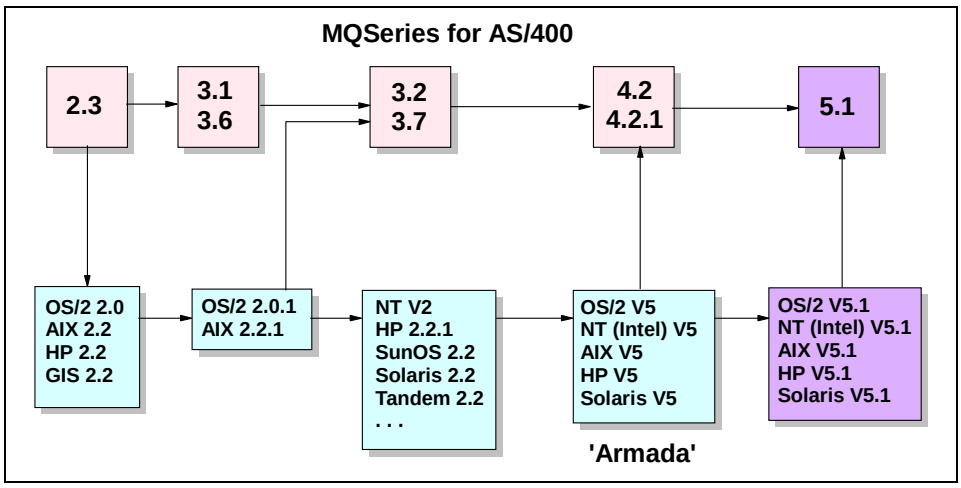
- **Maximize ease of migration for existing users**

Previous versions of MQSeries for AS/400 allowed users to make implicit connections to the (default) queue manager. This is not supported on all platforms. In order for existing programs to run without the need for changes or recompilation, interfaces identical to those provided with the previous release are provided.

To some extent these aims are in conflict - for example, changing the architecture of the product can make migration more difficult or adding function can be at the cost of performance - and some compromises have been necessary. We hope that in MQSeries for AS/400 V5.1 we have achieved these goals to an extent that will satisfy our existing users as well as providing a product that will be attractive to potential new customers.

The purpose of this White Paper is to describe this new function, to say what internal changes have been made to the architecture of the product to enable it, and to provide guidance to existing users of MQSeries for AS/400 on how to plan for migration to V5.1.

Background to V5.1



1Evolution of Distributed MQSeries Platforms

The figure shows the evolution of MQSeries on the distributed platforms, from the early 1990s. AS/400 was one of the earliest platforms to implement MQSeries and the functions implemented on the AS/400 formed the basis for the initial implementations on other platforms. Although the internal architecture of the AS/400 MQSeries product differed in some significant ways from the other platforms, the function provided has kept largely in step, albeit with some delays for the AS/400, explained in some cases by the timing of the MQSeries release being tied to an OS/400 release. This linkage between the MQSeries and OS/400 release (and also the adoption of the OS/400 release number) finished with V4R2M1, which was released with OS/400 V4R3. At this time MQSeries for AS/400 was also offered in shrink-wrap packaging.

When a subset of the distributed platforms: OS/2 Warp, Windows NT, AIX, HP-UX and Sun Solaris implemented version 5 of MQSeries (code name 'Armada'), MQSeries for AS/400

took some of the new function, including grouped and segmented messages, distribution lists and a dead-letter queue handler and released them with V4R2 and V4R2M1. However, there remained functions available in Armada, but not in MQSeries for AS/400, which our customers were requesting. These included:

- Support for Multiple Queue Managers
- Increase the maximum message size above 4 MB (Armada supported 100 MB)
- Increase 320 MB Queue size limit
- Java Bindings
- Two-phase commit

To these could be added a number of other requirements, some unique to the AS/400 environment, such as improved use of OS/400 work management functions.

Many of these requirements have been fulfilled with MQSeries for AS/400 V5.1. Note the numbering of the release, this indicates that the AS/400 product is part of the Armada family and shows the final break with the OS/400 release numbering - OS/400 V4R4 is needed to support this release of MQSeries - and subsequent release numbers will be those of the distributed MQSeries family.

New Function

The following are the major new functions provided in MQSeries for AS/400 V5.1:

- Multiple Queue Managers
- Increased maximum message size (100 MB)
- Increased maximum queue size (2 GB)
- MQSeries Queue Manager Clusters
- Enhanced transactional support
- Support for threaded applications
- Enhanced MQSeries classes for Java
- Enhanced bindings for COBOL
- Enhanced exploitation of OS/400 Work Management

In the following sections, these functions, and their potential benefits to the user of MQSeries, are described.

Multiple Queue Managers

In releases of MQSeries for AS/400 prior to V5.1 only one queue manager can be configured and started at any one time in a given AS/400 logical partition. This restriction can be inconvenient; for example in installations where it is necessary or desirable to run test and production systems in the same partition, or where an application provides multiple company support.

Users can now configure and start multiple concurrent queue managers on the AS/400. Each queue manager is a separate logical entity, and manages its own MQSeries resources, including queues, channels, listeners and so on. Physically these resources are mapped to files

in separate directories stored in the root of the Integrated File System (IFS). Each queue manager also has its own OS/400 library where the Journal is stored that is used to log changes to MQSeries objects and the images of persistent messages.

In order to accommodate multiple queue managers, most MQSeries CL commands now include an optional MQMNAME (Message Queue Manager Name) parameter, allowing the user to specify for which queue manager a queue, for example, should be created. The user can optionally specify that one queue manager be designated the 'default queue manager'. If such a default exists the MQMNAME parameter can be left blank, when that default queue manager becomes the target of the command. The default queue manager can also specified for the queue manager connect (MQCONN) API function. Applications, written for earlier versions of MQSeries, which make an implicit connection to a queue manager, will target this default.

Increased Maximum Message Size

Prior to V5.1 the maximum message size that can be specified in MQSeries for AS/400 is 4 MB. This is certainly large enough for most applications and reference or segmented messages provide means of reliably transporting larger logical parcels of data. For some applications, however, it is inconvenient for the physical size of the message - for example where the content is an image file - to be limited to a value this small. The increased maximum size of 100 MB is a reasonable relaxation of this limit.

Increased Maximum Queue Size

Prior to V5.1 this is 320 MB. The new maximum of 2 GB is particularly useful where large messages are involved or where the message arrival rate is high and a receiving application cannot, for some reason, clear those messages from the queue sufficiently quickly.

MQSeries Queue Manager Clusters

A cluster is a network of queue managers that are logically associated in some way. The queue managers in a cluster may be physically remote. For example, they might represent the branches of an international chain store and be physically located in different countries. Each cluster within an enterprise should have a unique name.

There are two quite different reasons for using clusters.

1. Reduced system administration.

Establishing a network of queue managers in a cluster involves fewer definitions than establishing a network that is to use distributed queuing. Whenever you create a receiver channel or define a queue, the systems administrator automatically creates corresponding sender channels and remote-queue definitions on the other queue managers. With fewer definitions to make, the network can be set up or changed more quickly and easily, and the risk of making an error in the definitions is reduced.

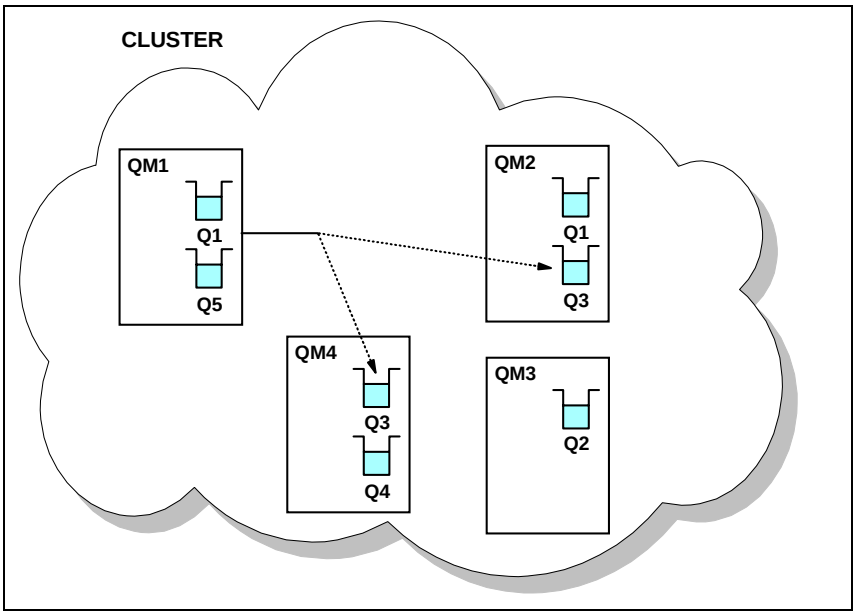
2. Increased availability and workload balancing.

As well as setting up clusters to reduce system administration, there is another way of using them. Clusters can be created in which more than one queue manager hosts an instance of the same queue. Clusters can be organized such that the queue managers in it are clones of each other. This means that they are able to run the same applications and have local definitions of the same queues. Because there can be more than one instance of an application each receiving messages and running independently of each other, the workload can be spread between the queue managers.

The advantages of using clusters in this way are:

- Increased availability of your queues and applications
- Faster throughput of messages
- More even distribution of workload in the network

Any one of the queue managers that hosts an instance of a particular queue can handle messages destined for that queue. This means that applications need not explicitly name the queue manager when sending messages. A workload management algorithm determines which queue manager should handle the message.



A cluster with multiple instances of the same queue

The figure shows a cluster in which there is more than one definition for the queue Q3. When an application at QM1 puts a message to Q3, it does not necessarily know which particular instance of Q3 will process its message. (Note, however, that when an application on QM2 or an application on QM4 puts a message to Q3 the local instance of the queue is used.)

Enhanced Transactional Support

In previous releases MQSeries for AS/400 has relied upon OS/400 to coordinate transactions involving messages operations taking place under syncpoint. MQSeries registers itself with OS/400 as a commitment resource and subsequent commit and rollback operations issued by the application apply to MQSeries as well as other registered resources, such as the database. Up until now MQSeries has registered itself as a *single-phase* commitment resource. This means that the MQSeries operations and, say, the database operations occurring within the commit/rollback boundaries are not treated as a single logical unit of work. It is possible (if unlikely) that a failure at a critical point could leave the database changes committed, but the MQSeries messages uncommitted and perhaps rolled back.

With V5.1 the support has been enhanced to make MQSeries register itself as a *two-phase* commitment resource. Now the commit and rollback operations treat all resources as being part of the same transaction: all are committed or rolled back together.

The MQSeries MQI functions MQCMIT and MQBACK are now supported. These commit and rollback the MQSeries message operations only: they do not affect resources external to MQSeries, such as the database. Note that the MQBEGIN function, provided here to help assist source code portability, does not provide the function in the release of allowing MQSeries to coordinate global units of work.

Support for Threaded Applications

Threaded versions of the service programs providing the MQSeries interfaces for C, C++ , ILE COBOL and ILE RPG are now provided. The bindings for Java also make use of the threaded service programs. The OS/400 threading support used by MQSeries is kernel threads, based on POSIX 1003.1c approved standard (draft 10), not the older draft 4 standard provided by CPA threads. MQSeries does not support the use of CPA threads.

Existing programs will have to be recompiled to bind to the threaded service programs and it is the responsibility of the user to ensure that the code is thread-safe. COBOL and RPG programmers should take special care: it is essential that threaded COBOL applications specify THREAD(SERIALIZE) in the PROCESS statement and threaded RPG applications specify THREAD(*SERIALIZE) in the control specification. The threaded support is only provided for programs using ILE bound procedural calls to the MQI; a threaded dynamic call interface is not available.

Enhanced MQSeries Classes for Java

The MQSeries classes for Java provide an interface to MQSeries, offering similar functionality to the MQI but providing an object-oriented, rather than procedural interface. With the MQI the program will typically call API functions to connect to a queue manager, open one or more queues, put and get messages and so on. With the Java classes the approach is to instantiate queue manager, queue and message objects and execute methods on the queue objects to put and get messages.

The classes come in two flavours:

- The MQSeries Client for Java

  As with other MQSeries clients, this uses a server connection channel to connect to a queue manager, which would usually reside on a remote system. The system on which the client is installed does not require an MQSeries Server to be installed and does not need to be an AS/400.

- MQSeries Bindings for Java

  These require the presence of a local queue manager. Connection with the queue manager is via the Java Native Interface (JNI).

These combined classes for the MQSeries client and bindings for Java are not included in the base V5.1 product, but may be downloaded from the Web. They form part of SupportPac MA88: 'MQSeries classes for Java and MQSeries classes for Java Message Service.'
The URL for this SupportPac is:

http://www.ibm.com/software/mqseries/txppacs/ma88.html

Documentation for the classes, provided in US English, form part of the SupportPac and are provided in html and Adobe Acrobat (TM) Reader format.

Instructions for the installation of the classes are provided with the SupportPac.

Enhanced Bindings for COBOL

In addition to the existing dynamic call COBOL interface, service programs are now supplied to provide ILE bound procedure calls to the MQI. The interfaces are identical: a compiler switch (LINKLIT(*PGM) or LINKLIT(*PRC)) allows you to specify which is to be used. Generally speaking the bound procedure interface should provide superior performance, as well as providing the COBOL programmer with access to the MQI functions new to MQSeries for AS/400 V5.1: MQCONNX, MQCMIT an MQBACK.

Enhanced Exploitation of OS/400 Work Management

During normal operations, an MQSeries queue manager starts a number of batch jobs to perform different tasks. In releases of MQSeries for AS/400 previous to V5.1 these jobs run in the system-supplied subsystem QSYSWRK. This can be inconvenient if, for example, this subsystem needs to be ended for some reason unrelated to the operation of MQSeries. In these releases it is also not possible to change the execution attributes, for example the run-time priority, for MQSeries jobs except on a temporary basis.

This has changed in V5.1. By default these AS/400 batch jobs now run in the QMQM subsystem that is created when MQSeries is installed. You can now tailor attributes of MQSeries tasks to obtain the optimum performance from your system, or to make administration simpler.

For example, you can:

- Change the run-priority of jobs to make one queue manager more responsive than another.
- Redirect the output of a number of jobs to a particular output queue.
- Make all jobs of a certain type run in a specific subsystem.

Work management is carried out by creating or changing the job descriptions associated with the MQSeries jobs, and is configurable for:

- An entire MQSeries installation
- Individual Queue managers
- Individual jobs for individual Queue Managers

Migration Considerations

As has already been mentioned, the internal architecture of MQSeries for AS/400 has undergone significant restructuring, involving a major shift away from use of the OS/400 library system for the storage of MQSeries objects, to exploitation of the Integrated File System with its UNIX-style directories and stream files. Although every attempt has been made to minimize the differences in the external aspects of MQSeries and the impact on user applications, some changes are inevitable. The purpose of this section is to describe some of the areas of major difference, which will require careful consideration and planning for all users migrating to V5.1 from an earlier release of MQSeries for AS/400. An outline description is given here and details can be found in the MQSeries for AS/400 Quick Beginnings and System Administration manuals.

Performance

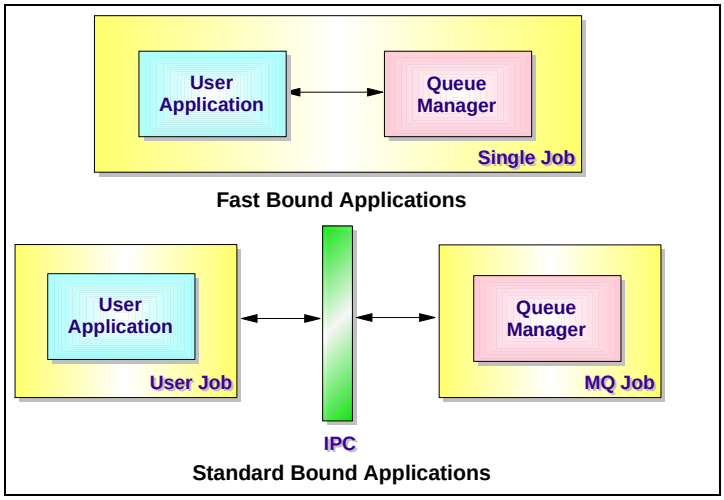
It is inevitable that in the process of making some fundamental changes to the product architecture and in adding significant new function that there would be a change in the performance profile. This has occurred and when the performance is considered of individual MQSeries operations, such as connecting and disconnecting from queue managers, opening and closing queues and putting and getting messages, differences are apparent.

At V5.1 some APIs show an improvement over V4.2.1, in particular MQCONN and MQDISC: this will be of benefit to customers with applications that make frequent and short transactions, say from clients. APIs that show reduced performance are MQCLOSE for an empty queue, MQOPEN and MQCLOSE for dynamic queues and MQPUT and MQGET. For standard bound, single-threaded MQPUT and MQGET of non-persistent messages of 1K in size, a measurement comparing of timings of calls showed an overhead of approximately 100% at V5.1 compared with the (fast bound) V4.2.1 figures. In the case of persistent messages the overhead was much lower: approximately 20%. This difference can be accounted for, in its entirety, by the cost of the process switch, which is described below, for the standard bound operations, since the corresponding measurements at V5.1 for fast bound MQPUTs and MQGETs are almost identical to the V4.2.1 figures.

Any measurements such as these should be interpreted with the greatest caution. Clearly the absolute cost of each operation in seconds and the mix in any particular customer application is paramount. Some customers, particularly those with MQ transactions involving relatively few message puts and gets, may see performance improvements. Environments where the MQI content of applications is relatively high, for example batch applications involving substantial numbers of put and get operations, may see a performance degradation.

Fast bound and Standard bound Calls

Fundamental to the understanding of the performance differences between this and the previous release is the comparison between 'fast bound' or ‘trusted’ execution of MQSeries functions and 'standard bound' execution.



Fast and Standard Bound Applications

The figure shows the difference between these two modes of execution, which are established at the time of connection to the queue manager. In fast bound mode the MQSeries queue manager functions execute as part of the user process. This includes delicate operations such as the manipulation of shared memory segments where queue data may be stored. Clearly this places a significant responsibility on the designer and of the user application both from the point of view of integrity of vital MQSeries data and also in the need to maintain adequate security.

With standard bound execution, the user and queue manager processes are separate and a badly behaved user application cannot directly compromise the queue manager. The user and queue manager processes use inter-process communications - similar to remote procedure calls - to exchange data and to coordinate their behaviour. In this mode each MQSeries operation, for example putting or getting a message, involves one or more process switches between the two sides. Compared with fast bound mode, the process switch adds an inevitable overhead.

MQSeries for AS/400 prior to V5.1 always runs in 'fast bound' mode, with no process switch. Certain aspects of OS/400 architecture are exploited to make this possible and safe and additional routines are incorporated to prevent the interruption of certain critical sections of

code. It is not possible in any easy way to transfer these protective constructions to the new, IFS-based architecture of the V5.1 product. Consequently MQSeries for AS/400 V5.1 runs, by default, in standard bound mode. Any performance comparisons with the previous versions should take this into account.

It is possible, through the use of the MQCONNX MQI function, to run in fast bound mode with V5.1. This is not generally recommended, for two reasons:

1. As stated above, the user application must be designed very carefully to ensure the integrity of MQSeries data, such as the content of queues. Fairly advanced programming techniques would be needed to implement such a design.

2. Fast bound mode puts a severe security constraint on the application. Although OS/400 programming interfaces may be exploited to avoid this, their use is not common practice.

For applications written in RPG and COBOL that use the dynamic program call interfaces to the MQI significant performance improvements may be realized by converting these programs to use ILE bound procedure calls. These performance gains may be sufficient to overcome, or even surpass any of the overheads mentioned above. Making the changes for COBOL applications may involve no more than a recompilation: see the the section above, headed 'Enhanced Bindings for COBOL'. In the case of RPG programs more work will be involved: first the source code must be converted (if necessary) to RPG IV syntax and secondly calls to the MQI must be converted to use prototyped procedural calls.

Change in the Security Model

Distributed MQSeries defines a security model that specifies how users may manipulate MQSeries objects, including queues and process definitions. Broadly speaking, users' authorizations in this model fall into four main categories:

- Authorizations related to MQI calls
- Authorizations related to the administration commands
- Authorizations related to message context
- General authorizations

These are implemented through an MQSeries component called the Object Authority Manager, or OAM. The OAM does not replace or override the platform security, rather it removes the need for the MQSeries security administrator to need to understand the underlying relationships between MQSeries objects and the files to which they map.

Prior to V5.1 MQSeries for AS/400 did not use the OAM, but used its own model. MQSeries authorizations mapped directly to OS/400 object and data authorizations to the underlying user spaces and other OS/400 object types involved. MQSeries V5.1 sees the replacement of these OS/400 objects with files stored in the Integrated File System. A necessary consequence of this architectural change is that the security model must also change, for two main reasons:

- IFS directory/file authorizations are different to the OS/400 library/object authorizations

  IFS authorities are based on the UNIX model, which is simpler than the normal OS/400 one.

- Program adopted authority does not apply to the IFS

  This was used previously to prevent direct user access to the OS/400 objects underlying the MQSeries ones.

For these reasons, as well the objective to increase the compatibility with the other V5.1 MQSeries platforms, it was decided to implement the OAM and dispense with the OS/400 object based security model. The CL commands to grant and revoke MQSeries authorizations, GRTMQMAUT and RVKMQMAUT remain, but their behaviour differs in a number of ways:

- OS/400 authorization lists are no longer supported
- Reference objects are no longer supported
- The 'admin object' is no longer used for context and alternate user authorities

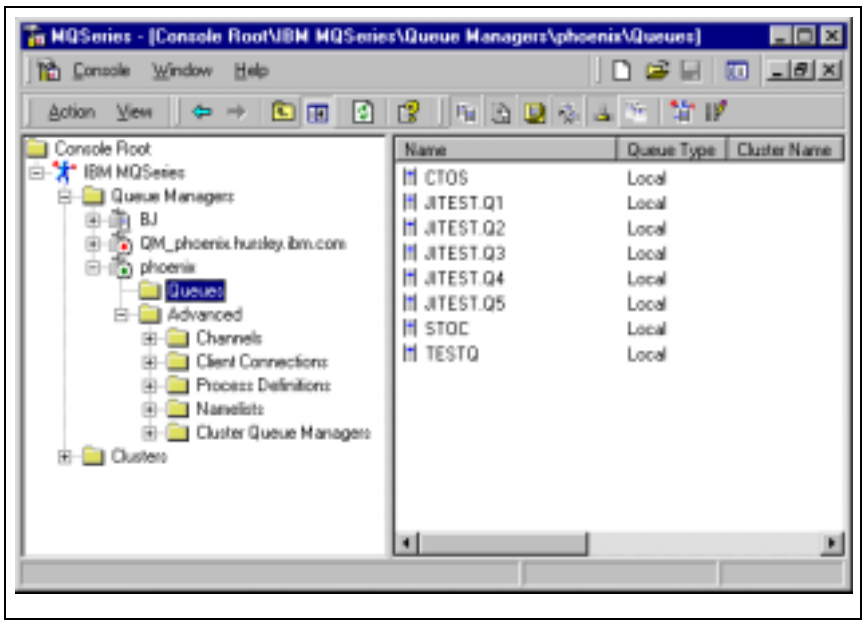
These changes mean that individual and group MQSeries authorizations will need to be examined closely prior to any migration to MQSeries V5.1. Since the detailed implementation of GRTMQMAUT and RVKMQMAUT have changed, any CL programs embodying these commands will need to be modified.

Administration Application

The MQSeries for AS/400 Administration Application, install option 2 on previous releases, is not provided with V5.1. This application provided some additional functionality and ease-of-use compared with the raw MQSeries CL commands and could be used to assist in the administration of local and remote queue manager resources.

Major work would have been required to incorporate some of the new functions, such as support for multiple queue managers and MQSeries Clusters, into the Administration Application. It was also apparent that the 'green-screen' 5250 panel format was no longer adequate for productive administration of these functions.

For users with access to MQSeries for Windows NT V5.1, a recommended alternative is the MQSeries Explorer snap-in to the MMC (Microsoft Management Console) that is supplied with that product. This can be readily configured for remote administration of MQSeries for AS/400 and provides a graphical user interface to many of the common administration tasks.



MQSeries Explorer

It displays a window like the Windows Explorer in which you can use the pane on the left (the *console tree*) to navigate through, and select, MQSeries objects, and the pane on the right (the *results pane*) shows a view of the selected object.

By right-clicking the mouse on items such as queue managers, queues and channels you get a menu where you can select your required action, such as create new, modify properties, delete and so forth.

Other administration products are also available from Third Party organizations.

User Exit Programs

All user exit programs will need to be recompiled for MQSeries for AS/400 V5.1.

MQSeries for AS/400 now makes use of OS/400 teraspace to implement its shared memory architecture. Teraspace, new with OS/400 V4R4, allows the application developer to construct large segments of shared memory, which MQSeries exploits to enable the increased message and queue sizes supported in this release.

This use by MQSeries of teraspace is hidden from user applications, except in the case of user exit programs. User exit programs are called as part of the queue manager job and are passed a parameter pointer to the message buffer, which is stored in teraspace. Any attempt to access the buffer, or de-reference the pointer, will fail unless the program has been compiled to be 'teraspace enabled'. In addition, if the programs may be called as TCP/IP channel programs, which may run threaded, they must be made thread-safe.

In order to enable C programs to use teraspace, TERASPACE(*YES *TSIFC) needs to be specified on the CRTCMOD and CRTBNDC commands. In the case of the ILE RPG and ILE COBOL compilers teraspace enablement is automatic. OPM RPG and COBOL programs will need to be converted to compile with the ILE compilers. In the case of COBOL this may

involve little or no additional effort, but OPM RPG source code will need to be converted, for example with the CRTRPGSRC command, to RPG IV format.

In addition to checking the COBOL and RPG source to ensure that it is thread-safe, with ILE RPG it is necessary to specify THREAD(*SERIALIZE) in the control specification and in the case of ILE COBOL that SERIALIZE be specified for the THREAD option of the PROCESS option.

Conclusion

This is a major upgrade for any existing user of MQSeries for AS/400. The new function will be sufficient reason for many to users to install V5.1, but as the above discussion spells out, careful planning is needed before going into production. Areas which will probably have the most impact are:

- The change in the MQSeries security model
- The possible need to convert, certainly to recompile user exit programs
- Possible performance degradation

In the first two areas the amount of work required for successful migration will be closely related to the complexity of the MQSeries environment. In the case of performance, the likely impact is problematic. As has already been observed, some users may see improvements in performance over the previous release; for many users the effects will be neutral or, if performance is not a critical concern, of no great business impact. There will be a class of environments where there may be a severe impact. These may be characterized as follows:

Where heavy use is made of MQSeries function calls known to carry additional overhead and, interactive response times or batch turn round times are critical, or
MQSeries already uses a significant percentage of available system resources.

In these cases careful thought must be given to decide whether an upgrade to MQSeries V5.1 should be attempted unless additional AS/400 system resources can be made available.