MQSeries
&
MQSeries Security

Stuart Jones
MQSeries Technical Strategy
IBM Hawthorne

stuartc_jones@uk.ibm.com

Agenda

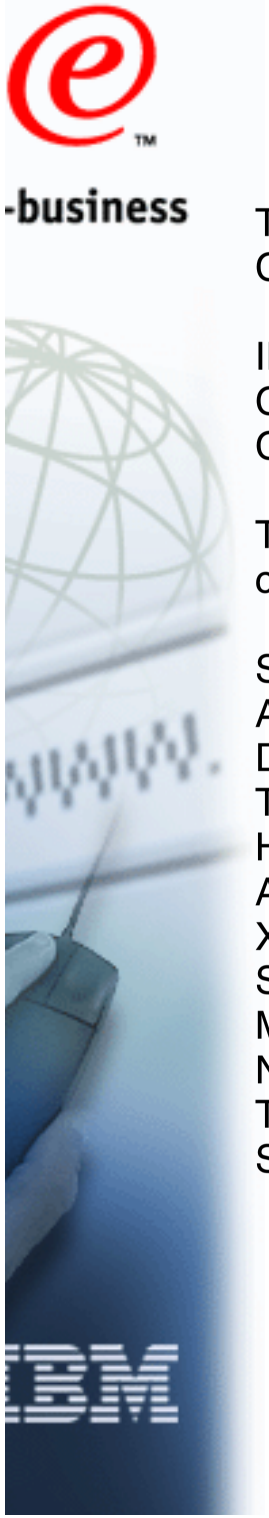
MQSeries Basics

Message Queuing Architecture

Platform Coverage

MQSeries Basics

MQSeries Security overview



Trademarks

The following terms, used in this presentation, are trademarks or registered trademarks of the IBM Corporation in the United States or other countries:

IBM AIX PC-DOS OS/2 OS/400 VTAM MVS/ESA VSE/ESA CICS
CICS/ESA CICS/MVS CICS/VSE CICS OS/2 CICS/6000
CICS/400 SNA MQSeries MQ

The following terms, used in this presentation, are trademarks or registered trademarks of other companies:

Systems Strategies Inc:	ezBRIDGE Transact
Apertus Technologies Inc:	Systems Strategies
Digital Equipment Corporation:	DIGITAL VMS VAX DecNet
Tandem Computers Inc:	Tandem Guardian Himalaya
Hewlett-Packard:	HP-UX
AT&T Company:	AT&T
X/Open Company Limited:	UNIX
SUN Microsystems Inc:	SunOS Solaris
Microsoft Corp:	Windows
Novell Inc:	UnixWare
The Santa Cruz Operation Inc:	SCO
Siemens Nixdorf Information Systeme	SINIX

MQSeries Basic Features

A single, multi-platform API

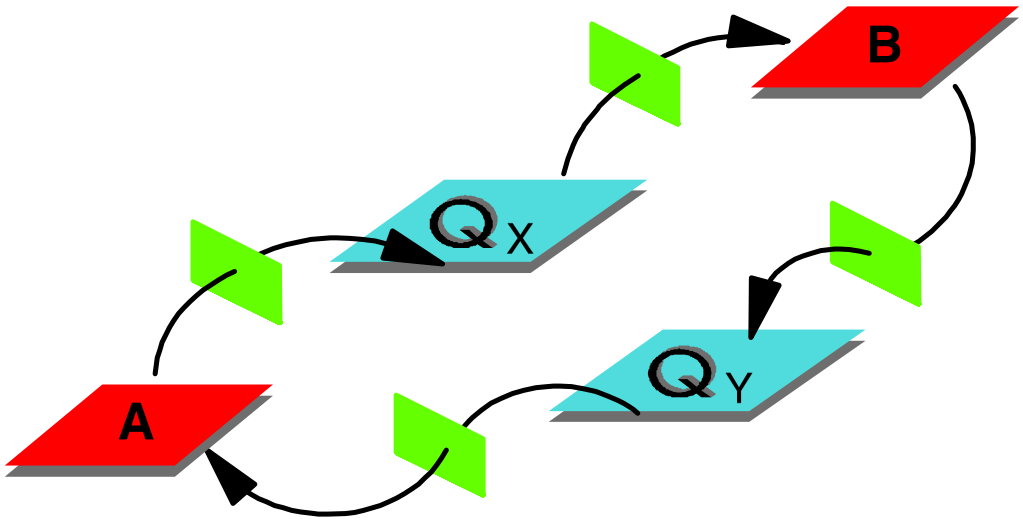
- Easy to use
- Network independent

...faster application development

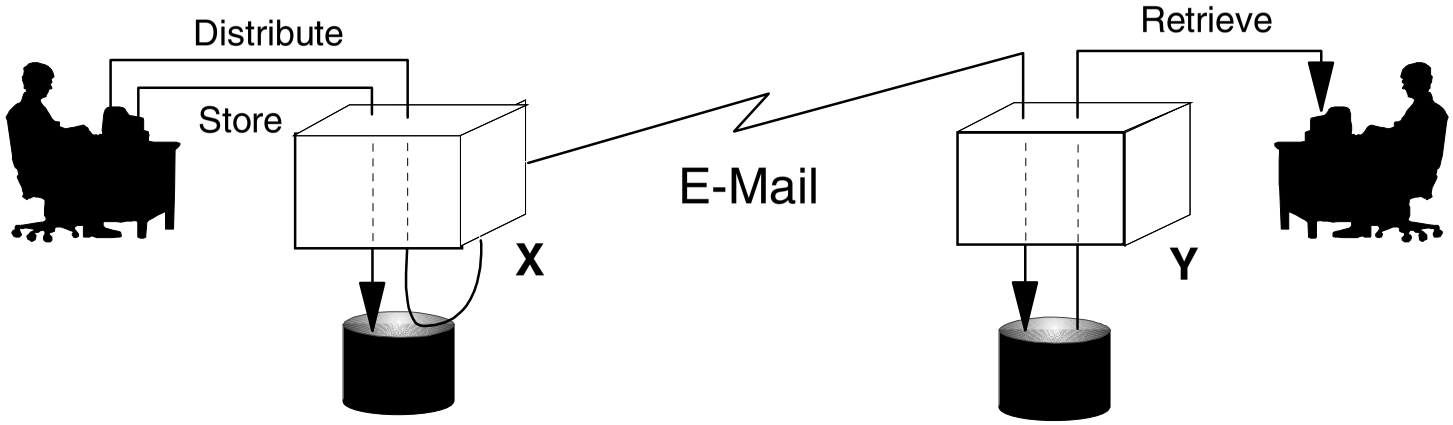
Assured message delivery

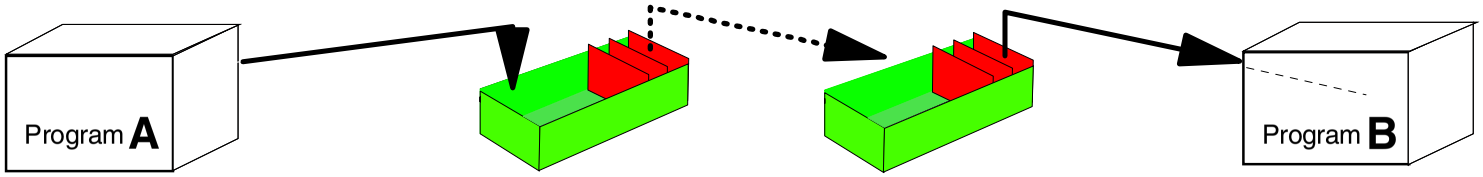
Time independent processing

- Asynchronous messaging



Types of Messaging





Message/Queuing

On-line/Real-time

Mail Systems/Messaging Systems...

Basic philosophy

- User orientation for mail systems
- Program orientation for messaging

Ease of programming

- Simple MQI verbs
- Short development cycles for messaging applications

Message Reliability

- Guaranteed once/once only delivery of data

Transactional Messaging

- Coordination of multiple messages
- Coordination with other Resource Managers

Reducing complexity

Reliable, distributed computing

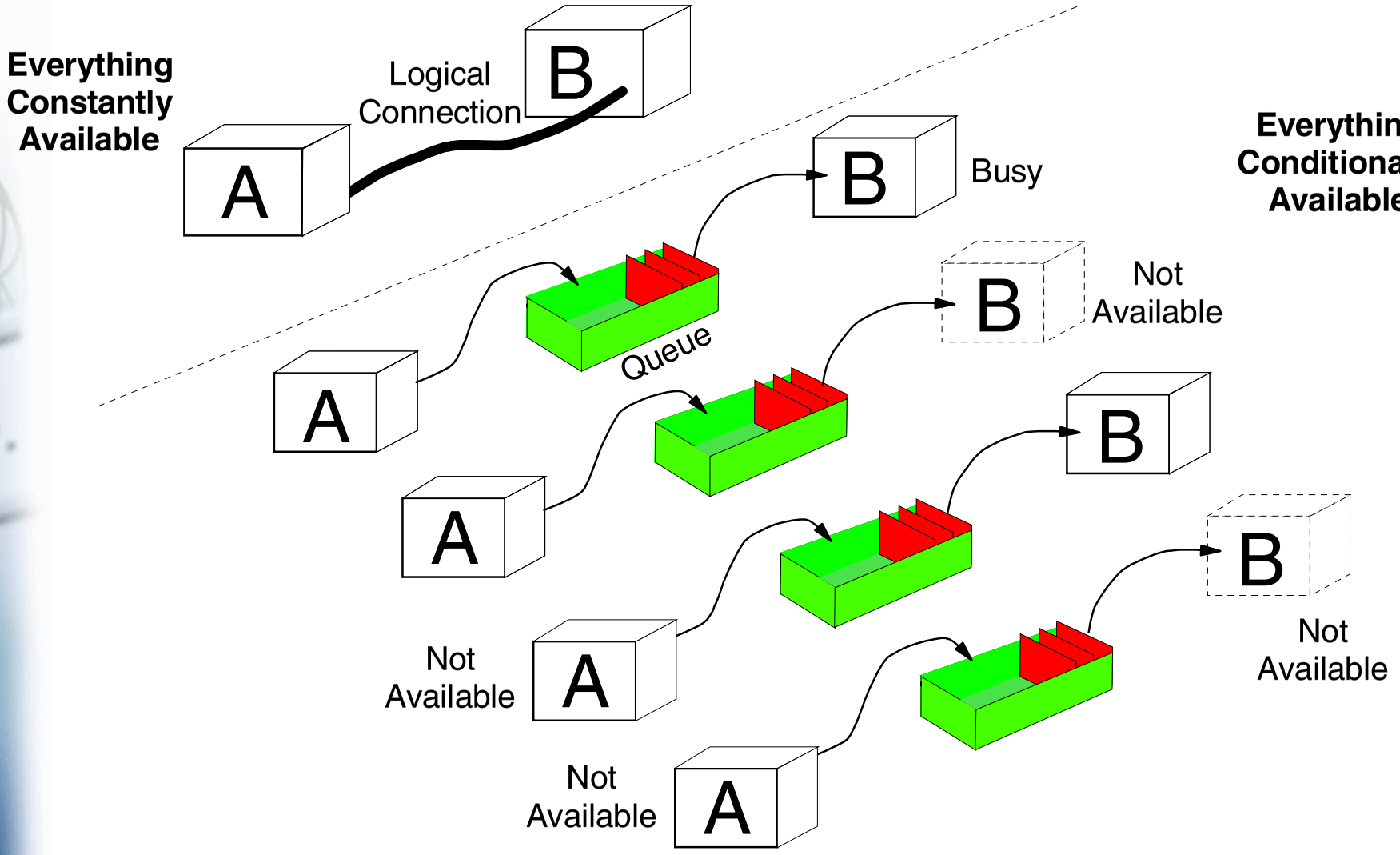
- Complex
- Error prone

MQSeries helps to ease the complexity

MQSeries is not a substitute for:

- Well written applications
- Robust network
- Good operational procedures
- Well managed system

Availability Choices



MQSeries Messaging Platforms

Servers:

- OS/390
- AIX
- Windows NT, 2000
- Solaris
- HP-UX
- OS/400
- OS/2
- OpenVMS
- Tandem NSK
- VSE
- Windows: 3.1,95,98
- Digital UNIX
- Compaq Tru64 UNIX
- Solaris:Intel & SPARC

- \+ SCO: Openserver, UnixWare
- \+ IRIX
- Dynix/ptx
- NCR
- TPF
- SunOS
- DC/OSx
- Sinix
- Linux
- PalmOS
- EPOC
- Java

Clients only:

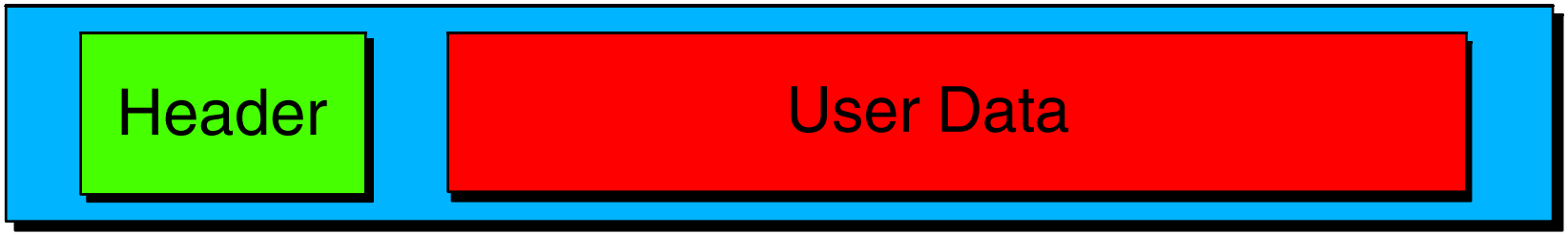
- \+ DG/UX
- \+ HP3000 MPE/iX
- Java
- Windows: 3.1,95,98
- DOS
- VM
- \+ Apple MacOS
- \+ Stratus VOS
- \+ 4690
- \+ Unisys A

39 Platforms

Plus Hitachi and Unisys 2200 offerings

Message

Message = Header + User Data



A Series of Message Attributes Understood and augmented by the Queue Manager
- Unique Message Id
- Correlation Id
- Routing information
- Reply routing information
- Message priority
- Message codepage/encoding
- Message format
- ....etc.

Any sequence of bytes
- Private to the sending and receiving programs
- Not meaningful to the Queue Manager

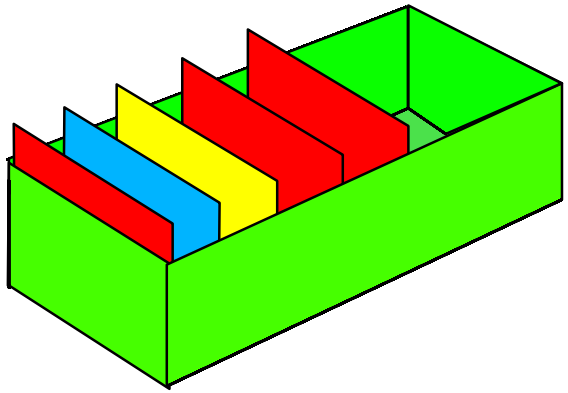
Queue

Message Types

- Persistent
- Non Persistent

Message Access

- FIFO
- Priority
- Direct



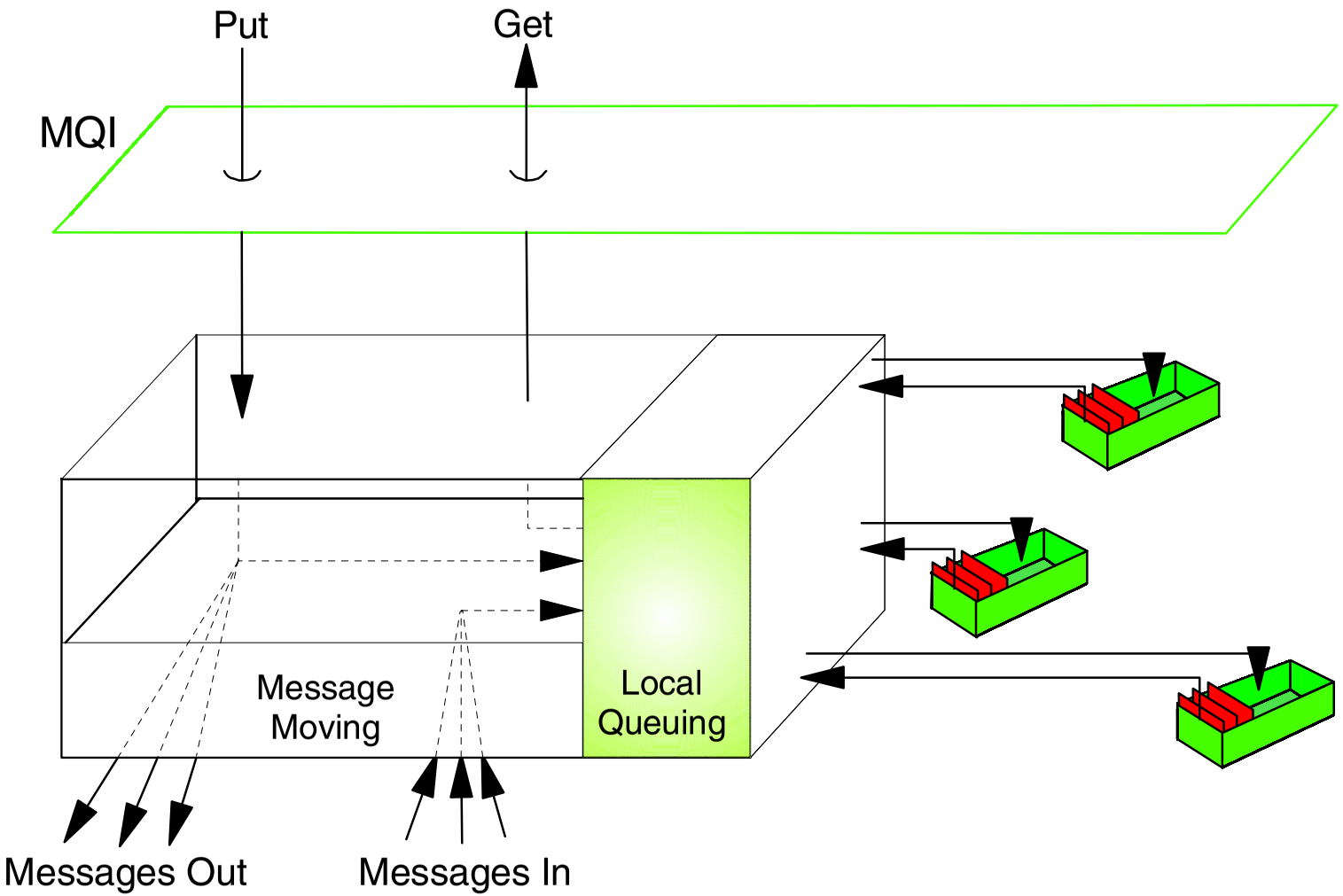
Queue creation

- Pre-defined
- Dynamic definition

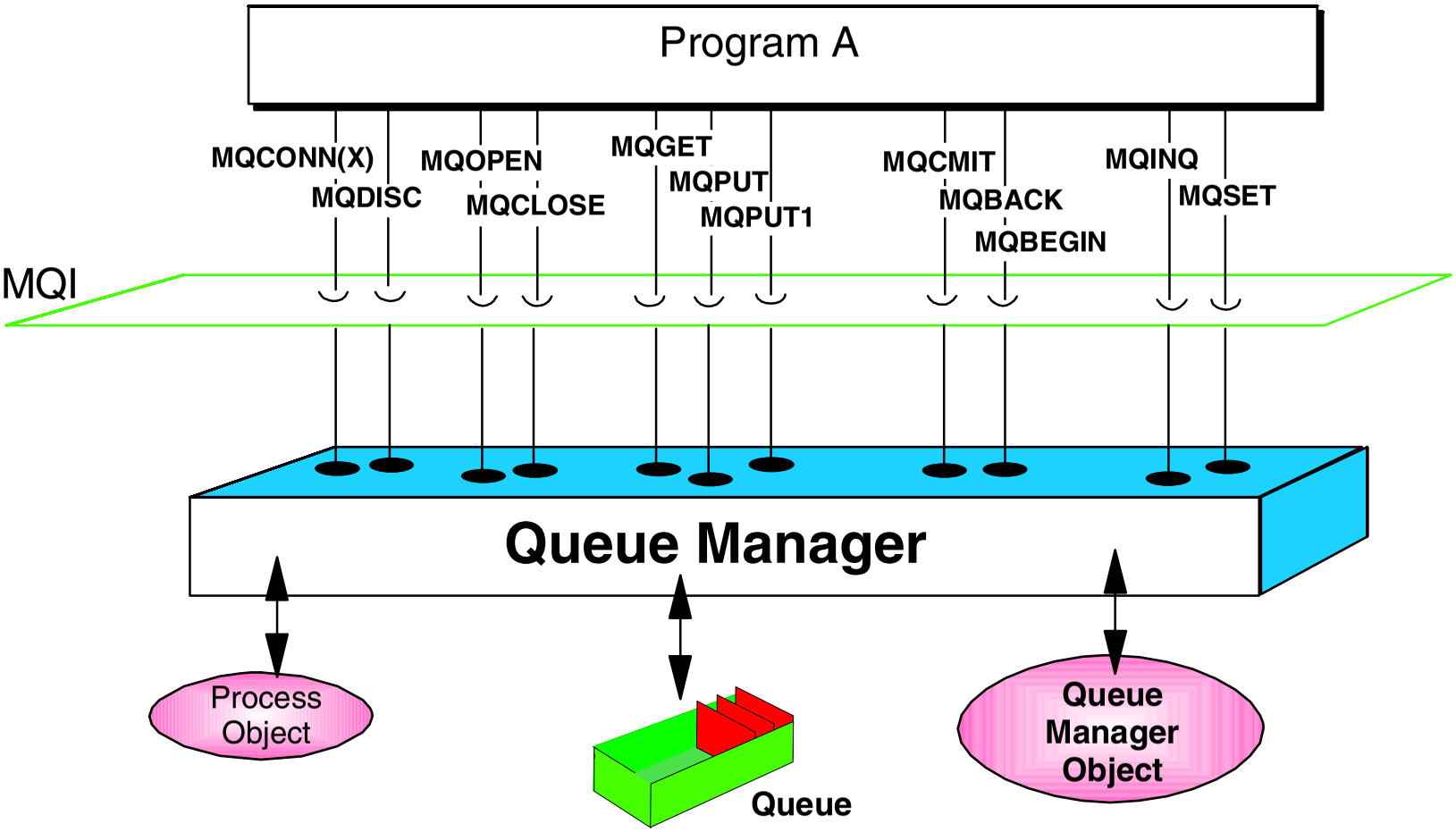
Up to 100MB message length

Parallel access by applications

Queue Manager



MQ API


Program A
MQCONN(X)
MQDISC
MQOPEN
MQCLOSE
MQGET
MQPUT
MQPUT1
MQCMIT
MQBACK
MQBEGIN
MQINQ
MQSET
MQI
Queue Manager
Process Object
Queue
Queue Manager Object

MQ API - Notes

There are 13 verbs in the MQSeries API, four of which are most heavily used and the rest have less frequent use. The most common verbs will be MQOPEN, MQCLOSE, MQPUT and MQGET which are concerned with processing of messages on queues. The other verbs have important uses but will not be used as commonly as these four. There are many, many options associated wth these verbs - approximately 250 ! However, in general, most of these options may be left to take their default values - and MQSeries provides a set of default structures to allow for easy assignment of these default values.

The MQ API is available for most programming languages - ASM, COBOL, C, C++, PL/1, RPG and SmallTalk. There is also support for Visual Basic, ActiveX and LotusScript.

MQSeries Transactional Integration

Message level selection for unit of work

Single UoW active at any one time

MQSeries messages in a unit of work

- MQCMIT and MQBACK control the unit of work

Messages and other resources in a unit of work

- Managed by a Transaction Monitor
  - MVS: CICS, IMS, OS/390 RRS
  - Any XA Transaction Manager

  Transaction Manager verbs control the UoW

- Managed by MQSeries
  - MQSeries V5 is a XA Transaction Manager

  MQBEGIN, MQCMIT and MQBACK control the unit of work

MQSeries Transactional Integration - Notes

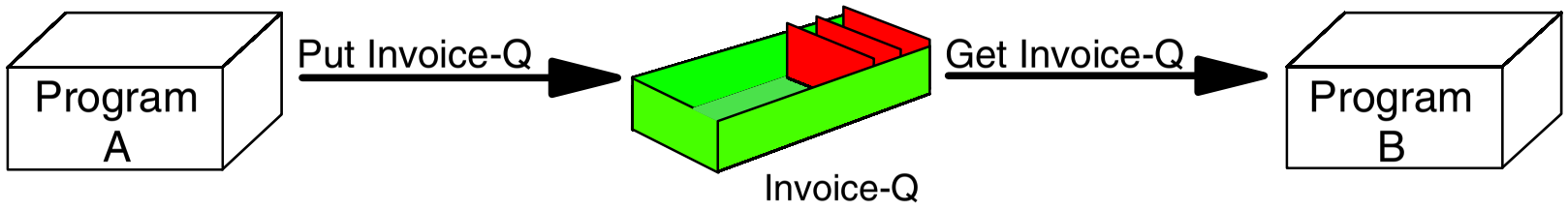
MQSeries supports logical units of work (UoW) where a set of resource updates may be considered as an atomic unit - either all of the changes are made or none of the changes are made. This support is particularly important when using MQSeries in a commercial environment (it's primary focus) as transactions play a major part in this arena.

MQSeries allows messages to be included/excluded from a UoW at the message level. This differs from some other environments where a UoW starts and *all* subsequent actions are included in the UoW. Thus, a set of messages may be considered to be a UoW. More often, it is necessary to include both MQ messages and some other recoverable resources (typically database updates) in a UoW. Typically, this has required the use of some Transaction Monitor and MQSeries works well with CICS and IMS on MVS and with any XA compliaint Transaction Manager. In situations where a Transaction Manager product is not available/suitable, MQSeries V5 may be used as the Transactoin Manager. This **does not** mean that MQSeries is becoming a Transaction Monitor, it is just providing the Transaction Manager aspect of a Transaction Monitor product.

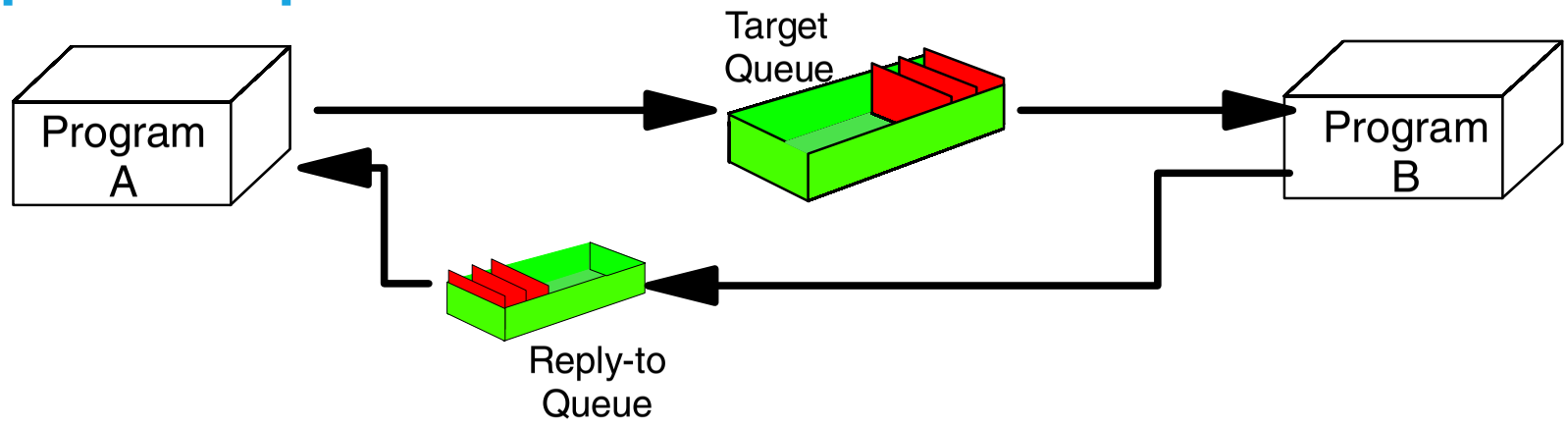
The API used in handling transactions differs according to the environment. MQSeries provides some verbs to handle UoWs. If a Transaction monitor is used, however, it's UoW verbs are used in place of the MQSeries API.

Examples

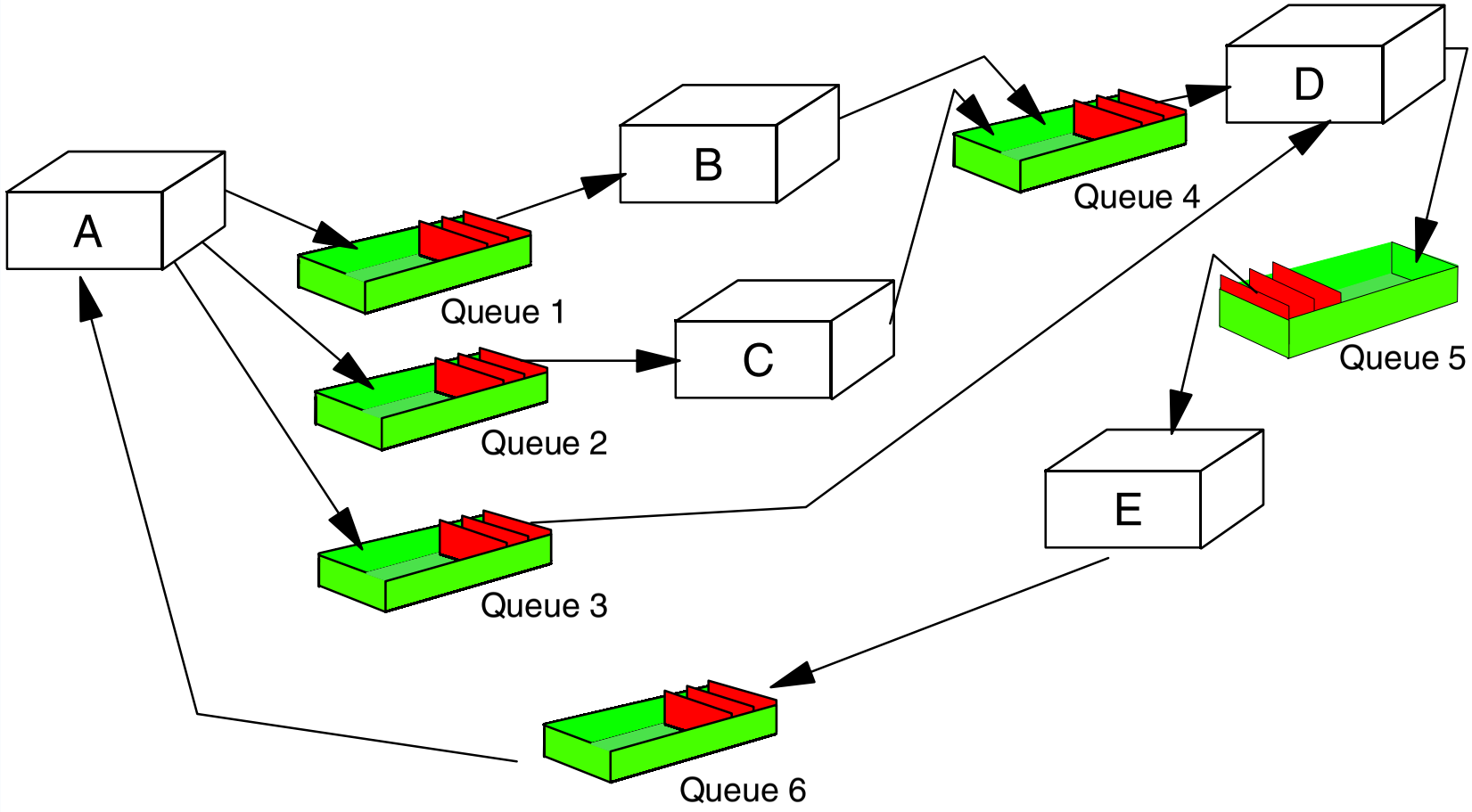
'Fire and Forget'



Request/Response



Topology Choices...



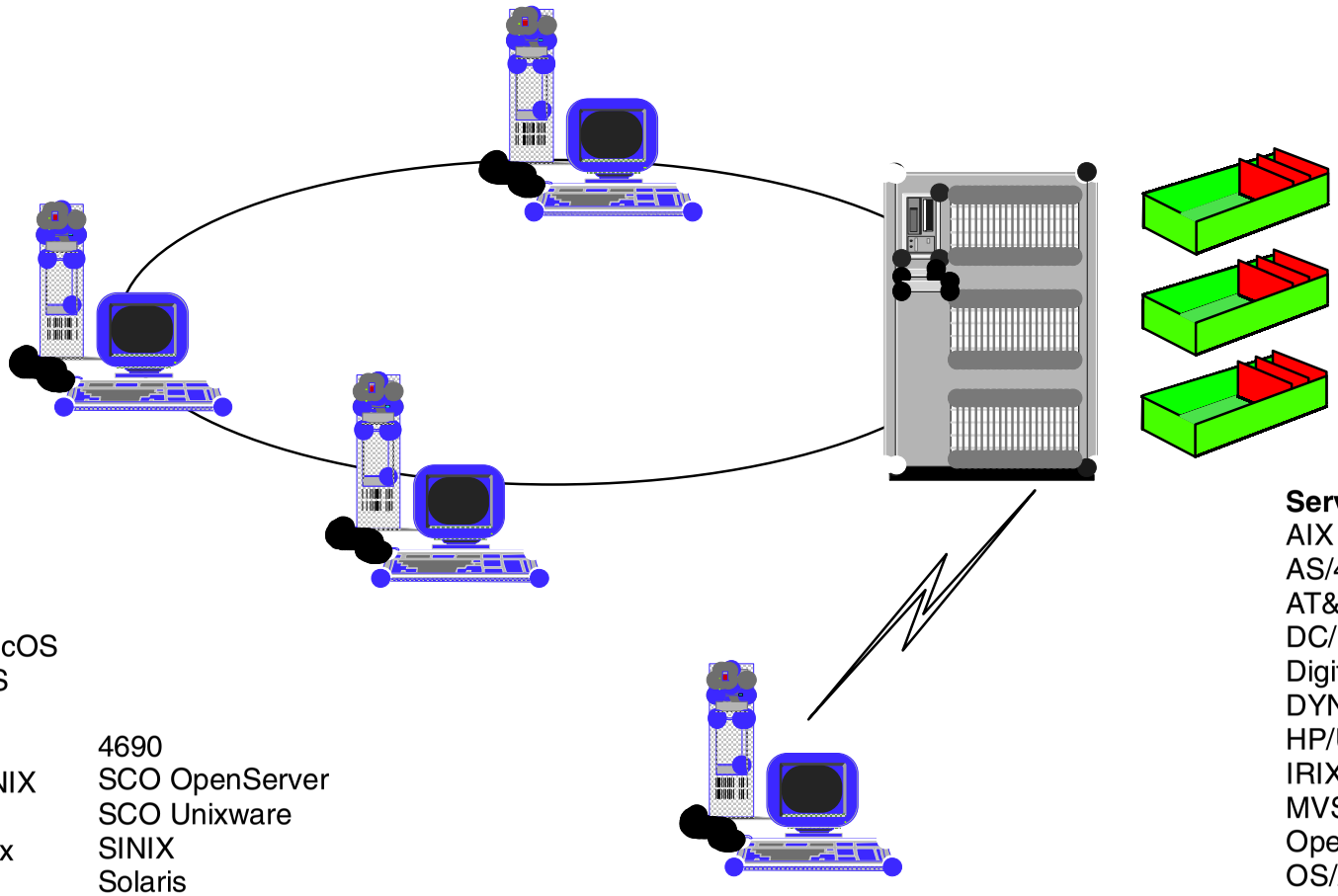
Topology Choices - Notes

There are two styles of access to an MQSeries queue - shared access and exclusive access. With exclusive access only one application may access the queue for a specific operation at any one time. Shared access means that any number of application may access the queue simultaneously. The type of access is governed separately for both PUT and GET. Thus, there may be multiple 'putters', but only one 'getter', a single 'putter' and mulitple 'getters' of multiples of both. This provides the facilities to have parallel access to queues, providing a single 'entry point' (or queue) for a set of applications and load balancing facilities when there are many messages to be handled. The Queue Manager will manage the mulitple applications simultaneously accessing queues, ensuring that the integrity of individual messages and units of work is maintained.

As well as providing for parallel access to applications, this mechanism is also useful where the relative speeds of applications varies; if one application fills a queue faster than receiving application(s) can remove messages then the Queue Manager will act as a buffer for as many applications as are removing the messages. This often prevents overload of receiving systems.

All of the topology choices and styles of operation described here may be combined - as required - to provide as simple or as complex a topology as is required for any particular style of application. In general, it is likely that such complex topologies will be built up over time and MQSeries easily allows this incremental development of applications.

MQSeries Clients



Clients:

AIX
Apple MacOS
AT&T GIS
DC/OSx
DG/UX
Digital UNIX
DOS
DYNIX/ptx
HP-UX
IRIX
Java
Linux
MPE/iX
OpenVMS
OS/2
4690
SCO OpenServer
SCO Unixware
SINIX
Solaris
Stratus VOS
TPF
Unisys A
VM/ESA
Windows 31/95/98
Windows NT

Servers:

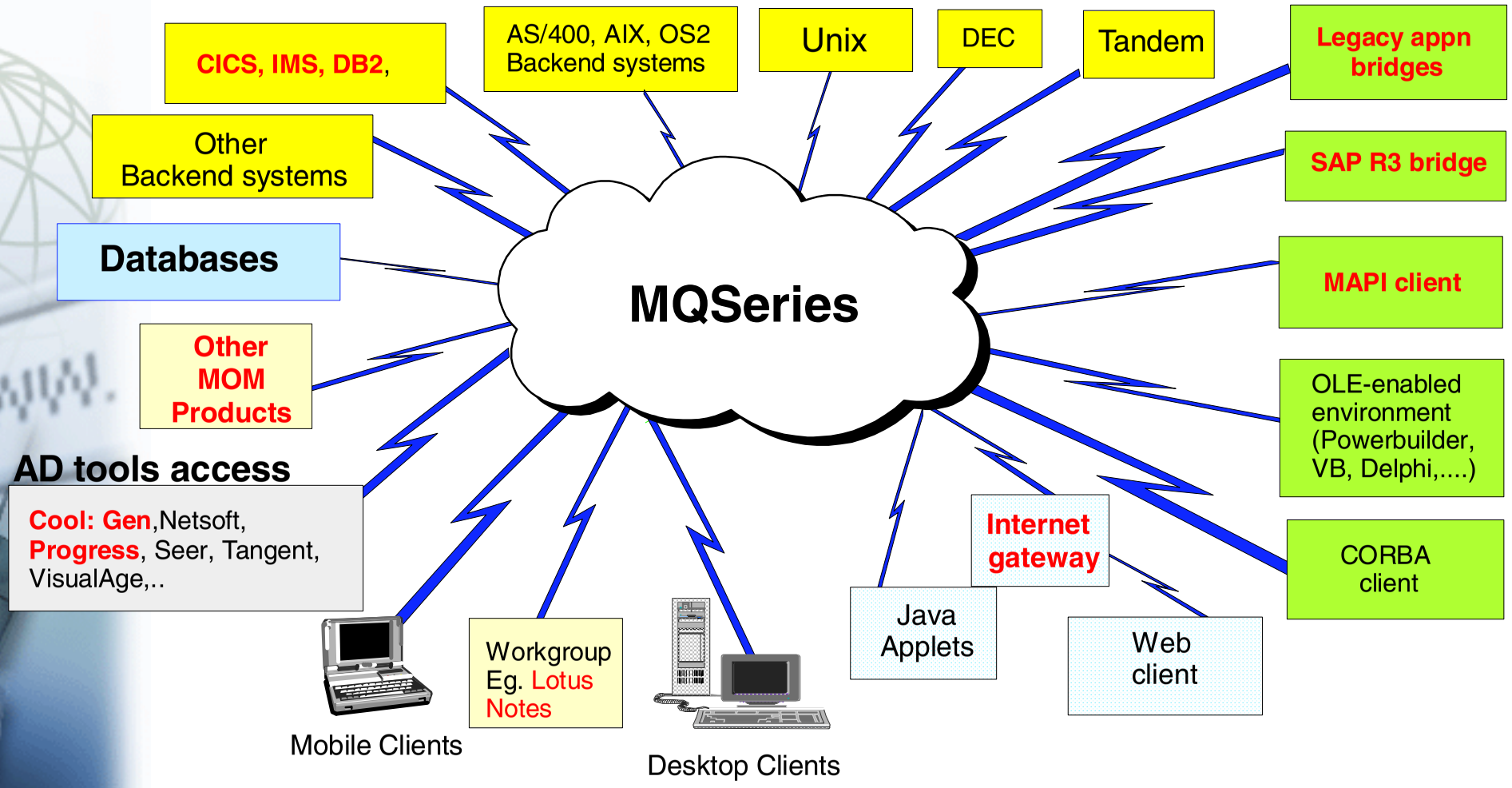
AIX
AS/400
AT&T GIS
DC/OSx
Digital UNIX
DYNIX/ptx
HP/UX
IRIX
MVS/ESA
OpenVMS
OS/2
SCO OpenServer
SCO Unixware
SINIX
Solaris
Windows NT

MQSeries Clients - Notes

MQSeries clients provide a low cost, low resource mechanism to gain access to MQSeries facilities. The client provides a remote API facility, allowing an MQSeries application to run on a machine that does not run a queue manager. Each MQ API command is passed to a Server queue manager where a proxy executes the required API command. The connection between client and server is entirely synchronous providing an 'rpc-like' mechanism - though NO regular (well-known) rpc mechanism is used ! Also, the client machine does not own any MQSeries resources - all resources are held by the Server machine. Thus, if local queuing capability is required then a server (rather than a client) must be used.

The client implementations are available in two groups. The (so-called) desktop clients are available for DOS,OS/2 and Windows (3.1 and 95). These clients are supplied with all queue managers that can act as servers. The other client implementations are *supplied* only with the server for that platform - though any client will connect to any server. Thus the HP-UX client (for example) is supplied only with the HP-UX server. However, the clients are available upon request for any platform, or from the MQSeries Home Page.

MQSeries Environment Bridges



MQSeries Environment Bridges - Notes

Fundamentally, a bridge is a bi-lingual program which 'understands' more than one environment. As a consequence, most bridge programs may be provided by any party - by MQSeries development, by some other part of IBM, by a vendor or by a customer. Most of the bridges listed on this chart are provided by MQSeries development but there are many other programs provided by vendors, listed below:

BEA MessageQ MQSeries Connection	BEA Systems
Crossplex	SofTouch
DARS (MQSeries/IDMS API)	Internaltional Software Products (ISP)
dbQ	Sybase Inc
Enterprise Access MQ edition	Apertus Technologies
Gateways to MQSeries (X.400, DCE, Tuxedo)	Marben Group
Interface for CA-IDMS	Aquisoft
Gateway to MS Falcon	Level 8
MQBroker	MultiConn International
MQBroker/Oracle	MultiConn International
NetEssential	Seer Technologies
Entire Broker	Software AG
ObjectBridge	VisualEdge
Orbix/MQ	Iona
PL/SQL Gateway for Oracle 7	Oracle Corp
TIB2MQ Brdge	TIBCO
MQ/204	CCA

The above list is not intended to be complete but is intended to show the breadth of products available. For a more complete list, consult the MQSeries Home Page.

MQSeries Features Summary

A single, multi-platform API

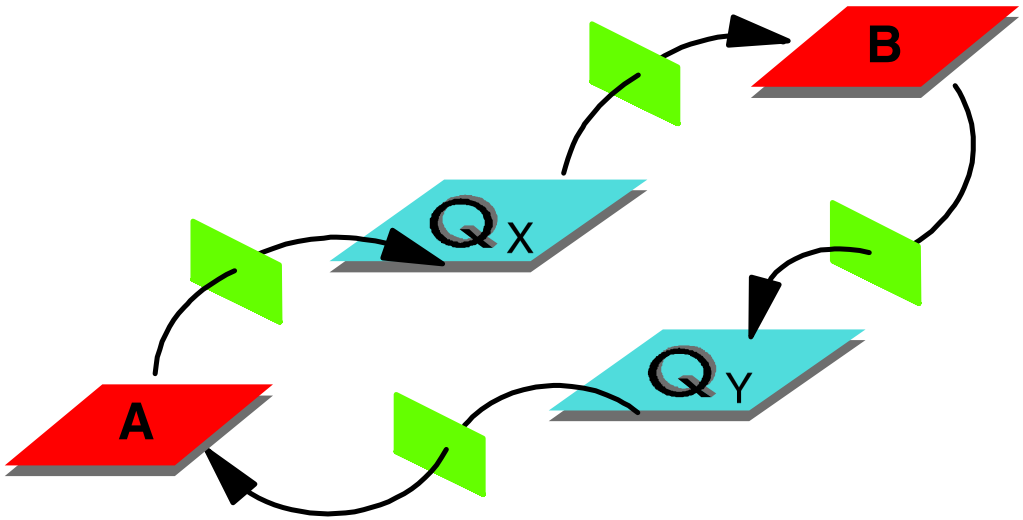
- Easy to use
- Network independent

...faster application development

Assured message delivery

Time independent processing

- Asynchronous messaging



MQSeries Security

Security Agenda

Access Control

- Access to MQSeries
- Access Control
- Context Security
  - User Context
  - Application Context

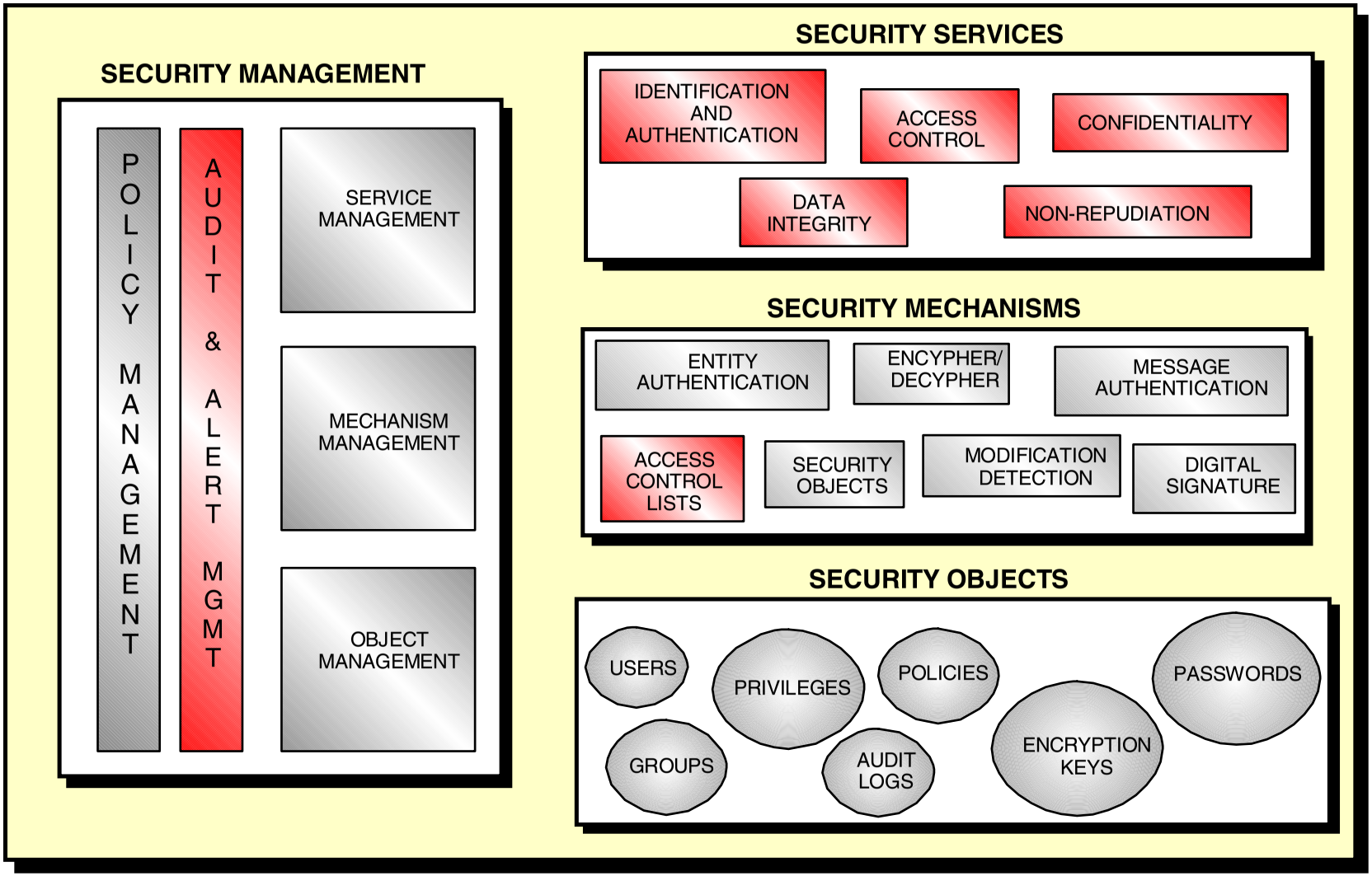
Point-to-Point Security

- Server/Server connections
- Client/Server connections

- Firewalls

Application Level Security

The Security Environment



The Security Environment - Notes

This diagram (based upon ISO standard 7498-2) outlines the security environment - the functions which make up that environment and how they are (inter-)related. For this discussion, it is the set of Security Services which is of primary interest; some, though not all of these servics are applicable to the MQSeries environment, as follows:

- Identification & Authentication (I&A) of users is carried before there is any use of MQSeries and so the Queue Managers are not responsible for providing any functions in terms of signing on and signing off. The Queue Managers are responsible for collecting the user identifier associated with a connecting application and using that userid for subsequent actions detailed below.

  When messages are sent to remote Queue Managers, the local Queue Manager is responsible for providing the I&A service so that the origin of the message can be reliably identified.

- MQSeries is repsonsible for ensuring that acceptable access control facilities are available to restrict access to MQSeries resources to properly authorised users. Note that this does not imply that MQSeries provides an access control service - it simply states that MQSeries must provides access to such services.

- MQSeries is responsible for providing (access to) confidentiality services

- MQSeries is responsible for providing (access to) data intergrity services

- MQSeries is responsible for providing (access to) non-repudiation services

Access To MQSeries

Access to commands

- Physical access control
- Operating system access control facilities
- MQSeries access control

Access to MQSeries applications

- Physical access control
- Operating system access control facilities

Access to the Queue Manager

- MQSeries access control

Access To MQSeries - Notes

There are several levels of protection that can be introduced to restrict access to MQSeries facilities:

- There are several MQSeries commands available - both for controlling the Queue Manager (such as CRTMQM, STRMQM) and for configuration of the Queue Manager (such as SETMQAUT). Either (or both) the operating system or MQSeries facilities may be used to control which users may access these commands. Base operating system facilities may be used to control access to libraries which contain the commands. Such facilities are outside of the scope of MQSeries. Alternatively, MQSeries provides access control facilities to restrict access.

  For the supremely cautious, placing of the MQSeries commands onto diskette will certainly restrict access to diskette holders only !!

- Access to MQSeries applications may be controlled by restricting access to the link libraries (used when link-editing MQSeries applications) and then by restricting access to the compiled and linked executable.

  Both of these controls are base operating system facilities and are outside the scope of MQSeries.

  Again, for the supremely cautious, placing of the MQSeries link libraries onto diskette will certainly restrict access to diskette holders only !!

- MQSeries provides access control facilities to control which users may run applications which issue MQCONN API commands. This will control which users may access the running Queue Manager, even though they may have access to the application libraries.

MQSeries Access Control

Complete access control facilities

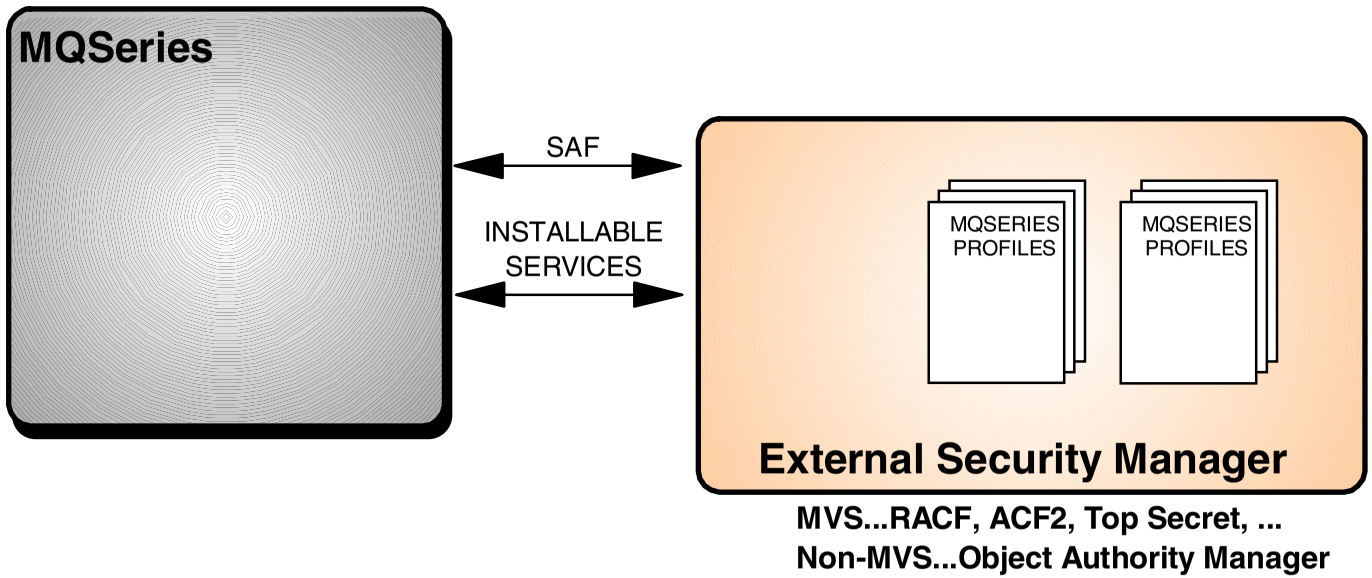
- For all MQSeries resources ... except channels

Two equivalent mechanisms

- MVS uses SAF
- Other platforms use MQSeries Installable Services

Capabilities depend upon profiles provided

- MVS control is more granular than other systems



MQSeries Access Control...

Control at user and/or group level

- Mirror operating system support
  - Tandem and OpenVMS are exceptions here
- MQSeries for Windows NT
  - 12 character userids only
  - **MQSeries for windows NT V5.1**
    20 character/domain qualified full NT userid

Alternate userids may be specified

- By authorised applications

First level name *only* is controlled

- Alias Queues
- Remote Queues

MQSeries Access Control - Notes

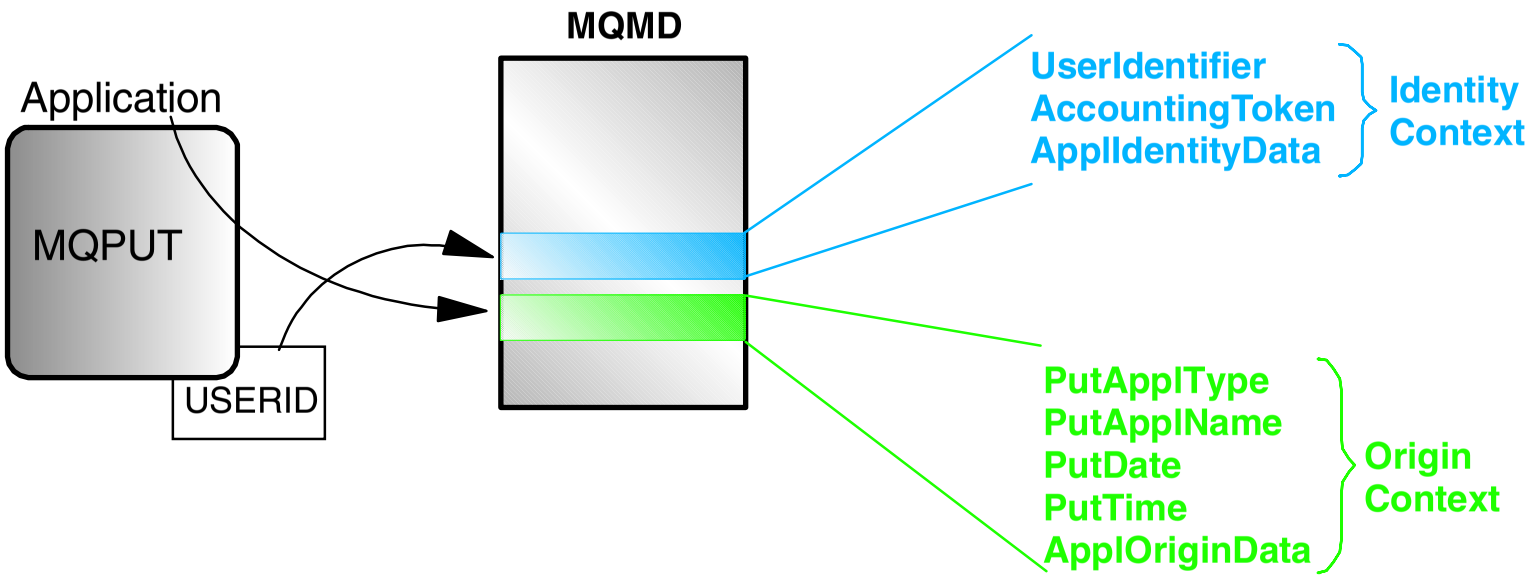
All advanced level Queue Managers provide access control facilities to control which users have access to which MQSeries resources. Note, however, that none of the Queue Managers has an access control component; all Queue Managers make use of an associated security manager to provide access control services.

- MQSeries for MVS/ESA uses the MVS standard System Authorization Facilitiy (SAF) interface to an (external) security manager. This means that MQSeries for MVS/ESA can operate with any security manager which conforms to the SAF interface. Examples of such (conforming) security managers are RACF, Top Secret and ACF2.
- The distributed Queue Managers use the Installable Services component of MQSeries - using the Authorization Service - to provide access control for MQSeries resources. MQSeries supplies an Object Authority Manager (OAM) as an authorization service which conforms to the Installable Services interface. The OAM provides a full set of access control facilities for MQSeries including both the access control checking and commands to set, change and inquire on MQSeries access control information. The OAM, like all Installable Services components, is replaceable by any component - user or vendor supplied - that conforms to the Authorization Service interface.

The set of facilities provided for MVS is richer than the non-MVS platforms. There are many more profiles for the MVS implementation providing more granular control of resources and capabilities. However, the non-MVS
systems do provide some facilties not available on MVS so neither platform is subset of the other.

When MQSeries checks to see if a user is permitted to access a particular resource, it is the name specified in the MQ API command which is used for the check. For the case of an Alias or Remote queue definition, it is still the name of the queue specified in the MQ API commmand and not the resolved-to name. Thus, a user needs access to the first named resource and not the resolved-to resource. For model queues, there may be instances where MQSeries will generate the name of the model queue. In this case it is recommended that generic named profiles are used for access control.

Message Context



Identifies message origin

Authorized applications may control all context fields

MQSeries for windows NT V5.1

- AccountingToken field is used to hold the Security IDentifier

Message Context - Notes

The MQSeries header contains a set of fields which form the context of that message. There are two parts to this set, identifying *who* the messages came from (the Identity Context) and *where* the message came from (the Origin Context). The Identity Context consists of the following fields:

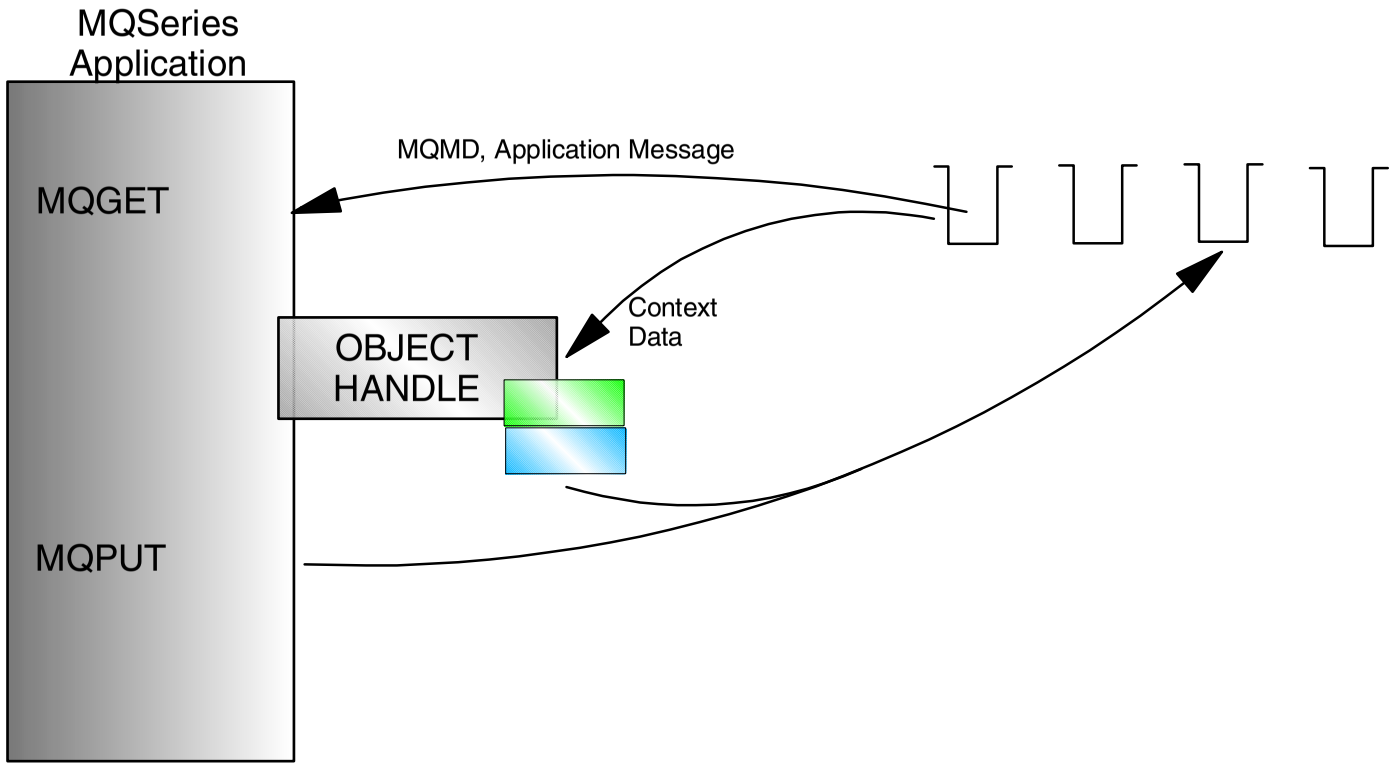
- UserIdentifier. This is a 1-12 character field which contains the userid associated with the putting application. For userids which are longer than 12 characters (particularly Windows NT today) the first 12 characters of the userid are contained in this field.
- Accounting Token. This field is intended to be used by MQSeries applications. If not set, the Queue Manager will set a *platform dependent* value which is documented in the relevant MQSeries Application Programming Reference.
  For the new MQSeries for windows NT V5.1 release, the AccountingToken has changed and will (by default) contain a Security Identifier.
- ApplIdentityData

  This is application dependant data and MQSeres does not define its format or place a default value in it.
  It may only be used by suitably authorised applications.

The Origin Context consists of the following fields:

- PutApplType. This is the type of application which has PUT the message and identifies the environment in which the application was running, such as CICS, IMS, AIX,...
- PutApplName. This is the name of the application which has PUT the message and will be the first 28 bytes of the fully qualified pathname for most environments and a transaction ID where appropriate.
- PutDate. The format is YYYYMMDD, GMT
- PutTime. The format is HHMMSSTH, GMT
- ApplOriginData. his is application dependant data and MQSeres does not define its format or place a default value in it. It may only be used by suitably authorised applications.

Passing Context



User must be appropriately authorised to open

- Output queue

Passing Context - Notes

There are many instances where the context data associated with one message needs to be associated with a subsequent message. For instance, when a mesasge is read by some server application and then forwarded for further processing. MQSeries provides a facility where the context associated with a message can be 'saved' when an MQGET is performed and then retrieved for some subsequent MQPUT. This is achieved by specifying:

- MQOO_SAVE_ALL_CONTEXT when the input queue is opened.
  Note that the authority required to use this option is that same as the authority required to open a queue of input.

- MQOO_PASS_IDENTITY_CONTEXT or MQOO_PASS_ALL_CONTEXT when the output queue is opened.

For MVS, it is possible to control which userids may open queues for input in this manner. For all advanced level systems, it is possible to control which userids can open queues for output with this option. When a message is put to the output queue, there are two PMO* options to control which parts of the context set are passed; these are MQPMO_PASS_IDENTITY CONTEXT and MQPMO_PASS_ALL_CONTEXT. Note that there is no mechanism to save just the identity or origin context, all context is saved.

Audit: MQSeries Security Events

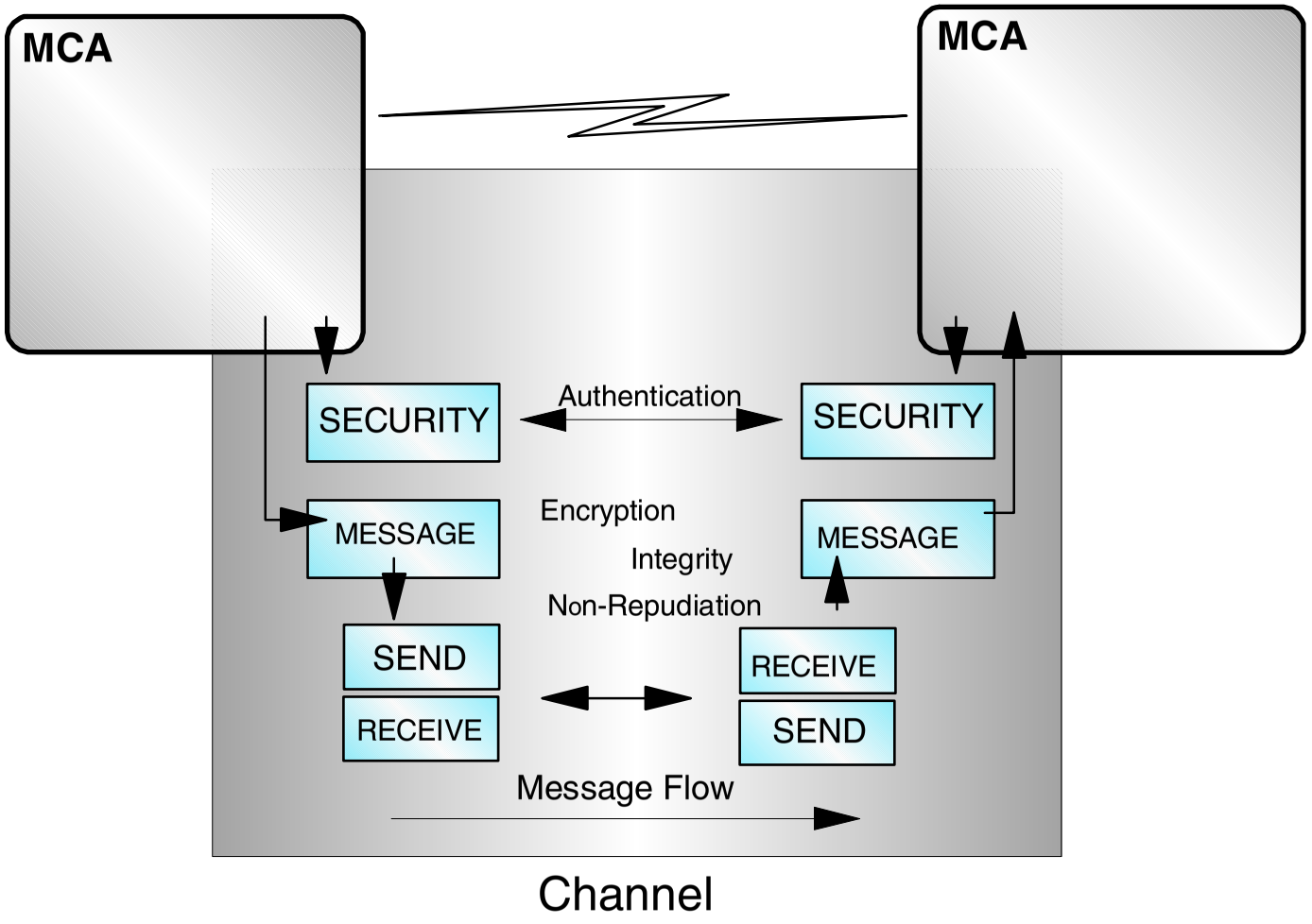
MVS Queue Manager...standard MVS ESM facilities

Distributed Queue Managers...4 'Not Authorized' events:

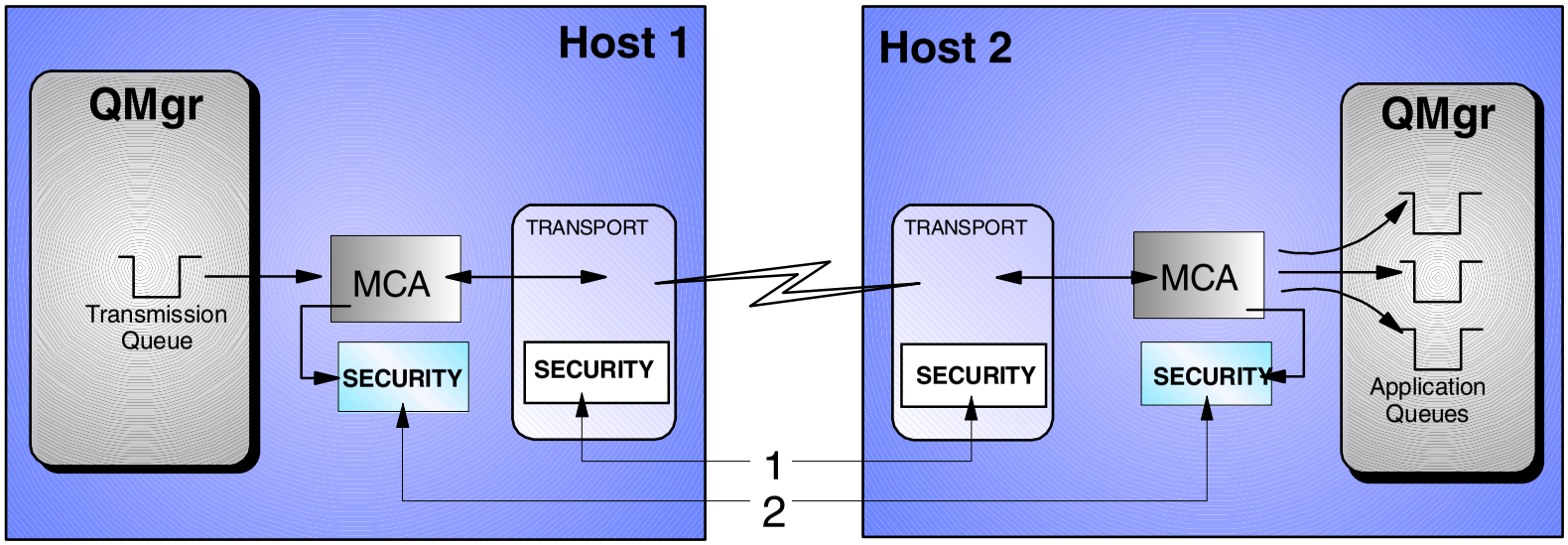
- MQCONN
- MQOPEN/MQPUT1
- MQCLOSE
- MQSeries PCF commands

- All events written to SYSTEM.ADMIN.QMGR.EVENT queue

Point-to-Point Security



Server/Server Security - Authentication



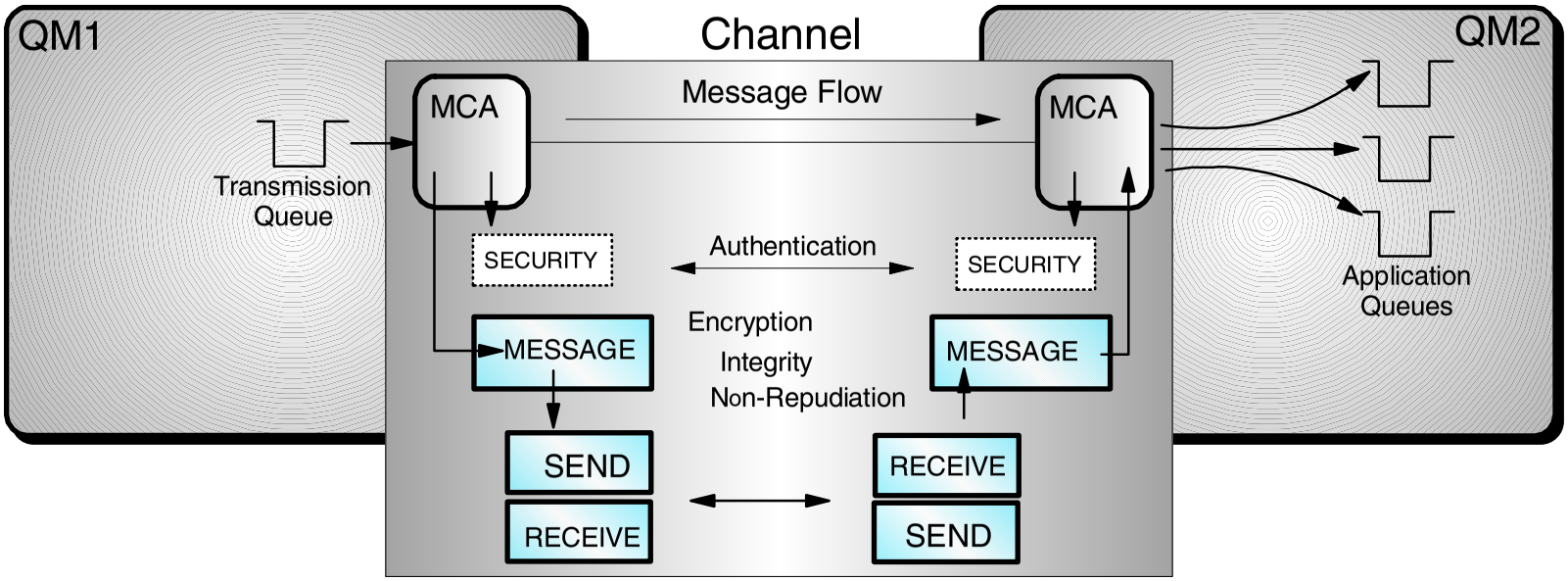
1. Transport level security

- APPC Session Level Security, IPSec ... IPv6

2. MQSeries level security ... security exits

- MQSeries V5 exits (using DCE), Supportpac for Entrust support
- MQSeries ISVs
- RYO
  IBM Redbook
  - MQSeries Security: Example of Using a Channel Security Exit, Encryption and Decryption...SG24 5306

Server/Server Security...



Transport level functions only

- No end-to-end security support

MQSeries level security ... security exits

- MQSeries V5 exits (using DCE), Supportpac for Entrust support
- MQSeries ISVs
- RYO
  IBM Redbook
  - MQSeries Security: Example of Using a Channel Security Exit, Encryption and Decryption...SG24 5306

Server/Server Security - Notes

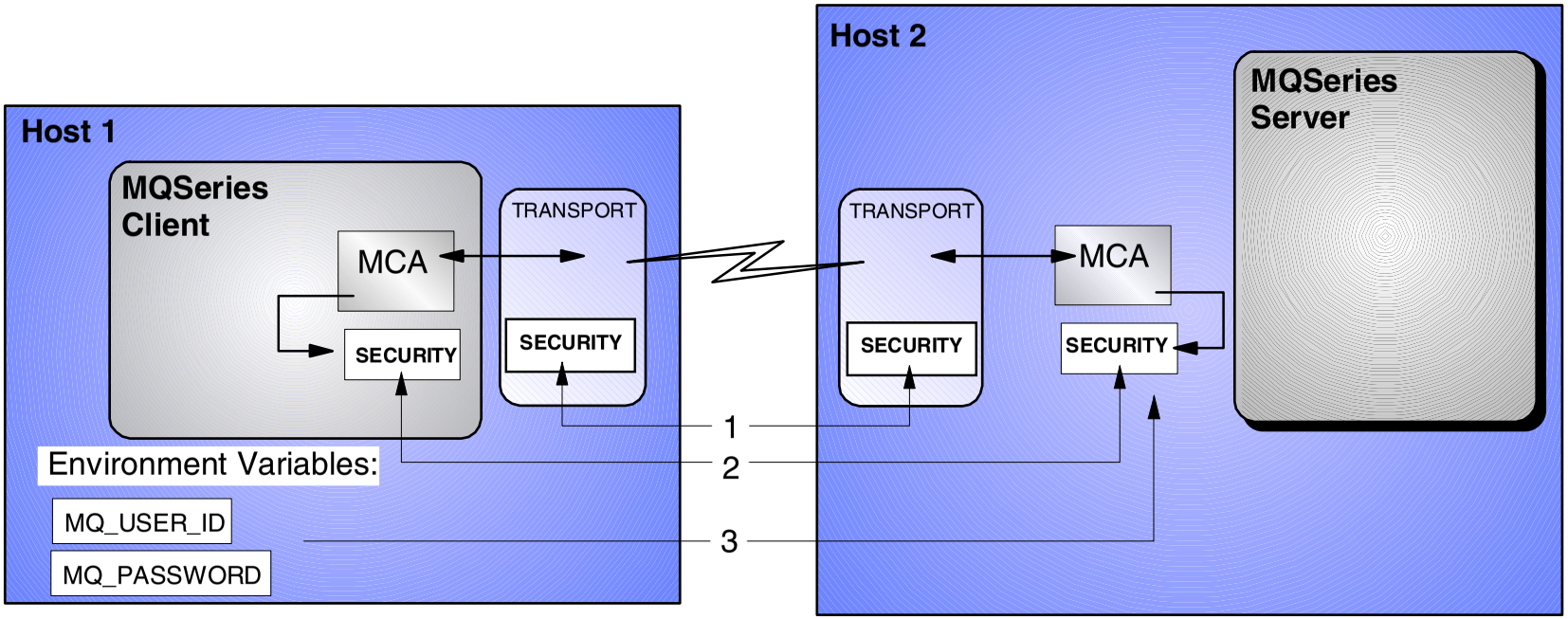
Queue Managers need to ensure that they exchange messages with the correct partner Queue Managers. Note that it is just as important for the sending Queue Manager to be sure of the receiver's identity as it is for the receiving Queue Manager to be sure of the sender's identity. Such an environment is called *mutual authentication*. There are two methods available for the provision of such partner authentication

1. Some transport systems provide for partner authentication when a connection is made between the two systems. For instance, many APPC implementations provide session level authentication (also known as Bind Time Security) to enable mutual authentication of each partner. A similar facility is provided by Secure Sockets Layer (SSL) on TCP/IP,] though not as a built-in part of the protocol. Note that such an authentication provides proof only of the partner transport - and not the partner queue managers. For most environments, this will be sufficient as a node (a physical machine) will usually be trustworthy. However, there will be environments where the authentication of the transport is not sufficient to verify the application using that transport. Further, this kind of facility is not available for all transports or on all platforms supporting a particular transport.

2. The Message Channel Agents (MCAs) provide an exit point which enables an authentication process to be carried out. The exit running at this point is user provided and so any desired style of authentication may be provided. As the exit runs as a part of the MCA - and so runs above the transport layer - the exit may be protocol independent and, thus, be available for any of the underlying transport mechanisms. The security exits may also (optionally) control the userid under which the MCA runs and accesses MQSeries resources.

Note that the Message Retry Exit is not shown in the diagram as it does not support security functions

Client/Server Security - Authentication



1. Transport level security

- Same as Server/Server environment

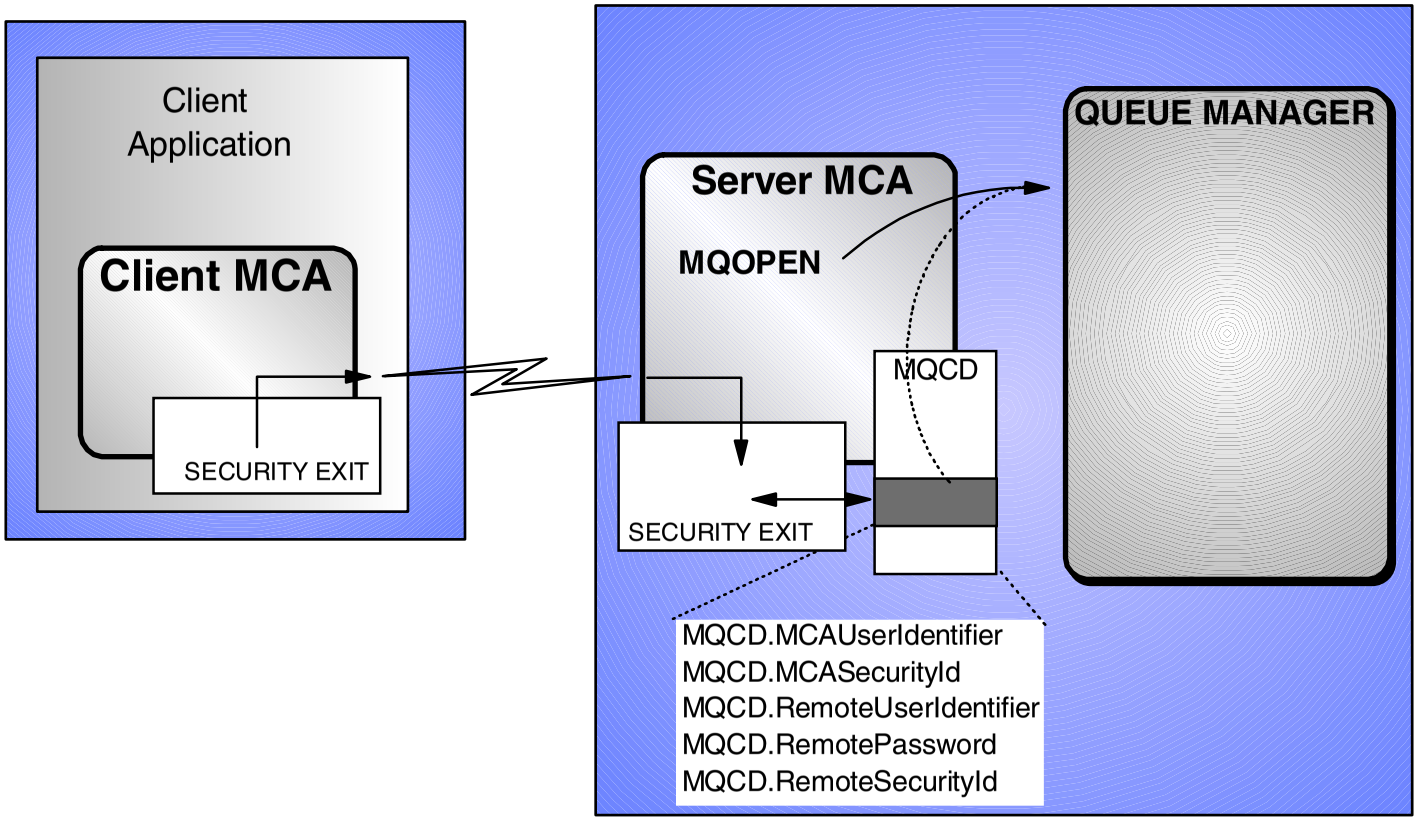
2. MQSeries level security ... security exits

- Same as Server/Server environment

3. MQSeries level security ... environment variables

- Discontinued in MQSeries V5.1 for UNIX and Windows NT

Client/Server Security: Userids

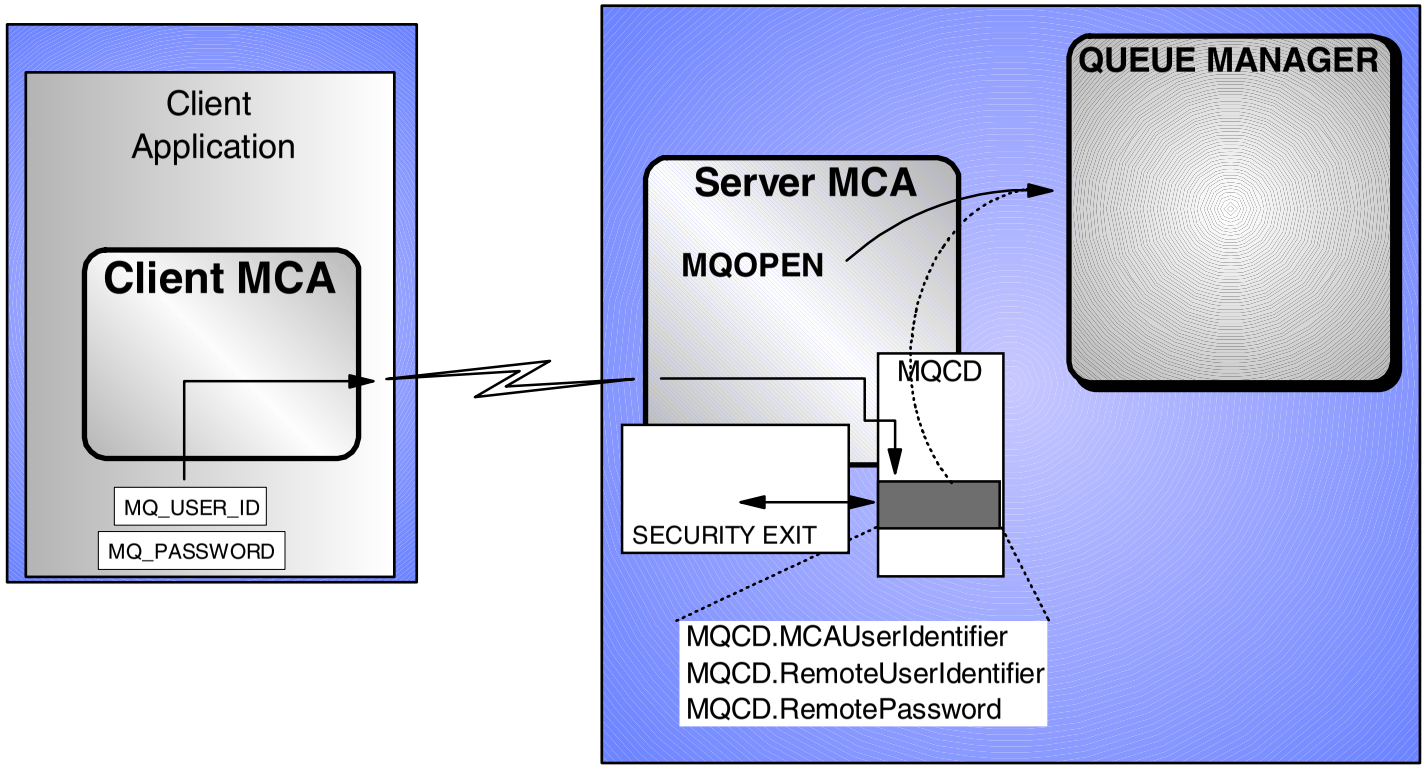


MQSeries V5.1:

- Exits no longer required
  - client userid automatically transferred to SVRCONN
  - windows NT accepts a Security ID from windows NT clients

... only automatic if SVRCONN MCAUSER is blank

Client/Server Security: Userids...



Environment variables:

- Discontinued in MQSeries V5.1 for UNIX and Windows NT
  - Available for all other client platforms
- MQ_PASSWORD is for processing via exits *only*

Client/Server Security: Userids - Notes

In an MQI client-server environment, there is a client MCA which passes MQ API commands to a server MCA. This serves as a proxy to the client application, executing the client API commands . The userid which the server MCA uses to access MQSeries is set when the client MCA connects to the server MCA. There are several alternative ways to set the userid that will be used by the server MCA.

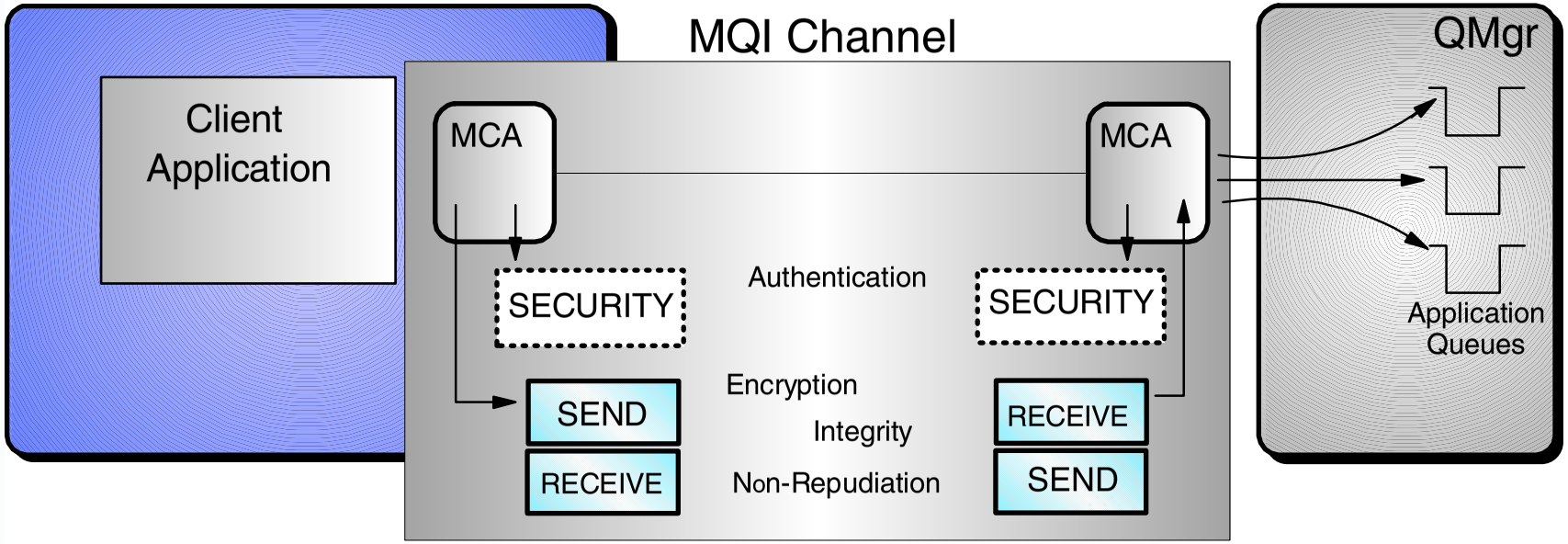
- Set the channel MCAUSER parameter for the SVRCONN channel definition.
- Set the client environment variable MQ_USER_ID. This is used if the MCAUSER parameter is blank or nulls.
- Use the security exit. For the client-server environment, there are two options for the security exit:
  - Server exit, no client exit. In this case, any environment variables are automatically passed to the server and then the security exit is called. The security exit can reset the userid that the MQI Server will use, as required.
  - Server exit, client exit. In this case, environment variables are not automatically passed to the server MCA. The client exit may aps any desired information to the server exit. This will usually be a userid, though does not have to be. The server exit may set the userid as required.

If a client security security exit is defined, then the client environemnt variables are not automatically sent to the server MCA.

After the above processing has taken place, the server MCA will adopt the userid that is contained in theMCAUserIdentifier parameter of the MQCD control block. The MQCD is passed to the security exit as one of the MQCHANNELEXIT parameters (ChannelDefinition).

With MQSeries V5., there is no need to code an exit as the transfer of client userid to server is made automatic. Further, with MQSeries for windows NT V5.1 the client Security Id is also automatically transferred from client to server.

Client/Server Security...



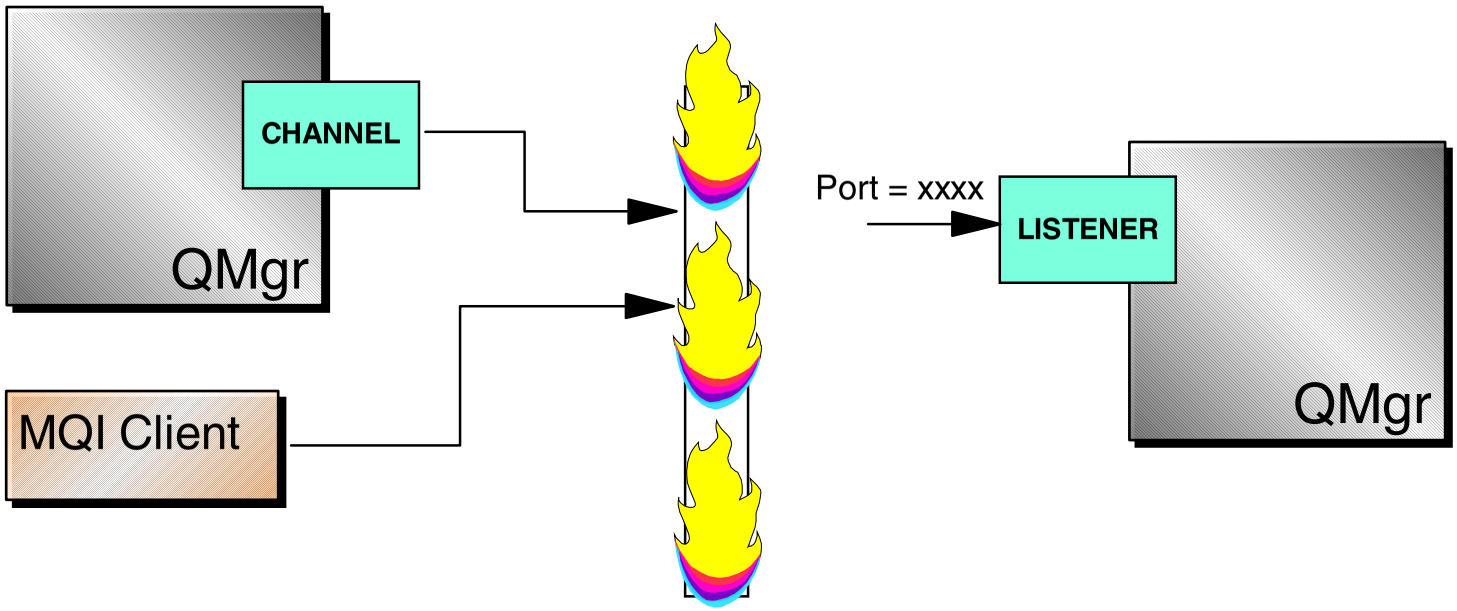
No Message exit

- There are no messages exchanged
- Causes difficulty in *selective* security processing
  - Client FAP is partially exposed

Provision of exits

- Same as Server/Server environment

MQSeries and Firewalls



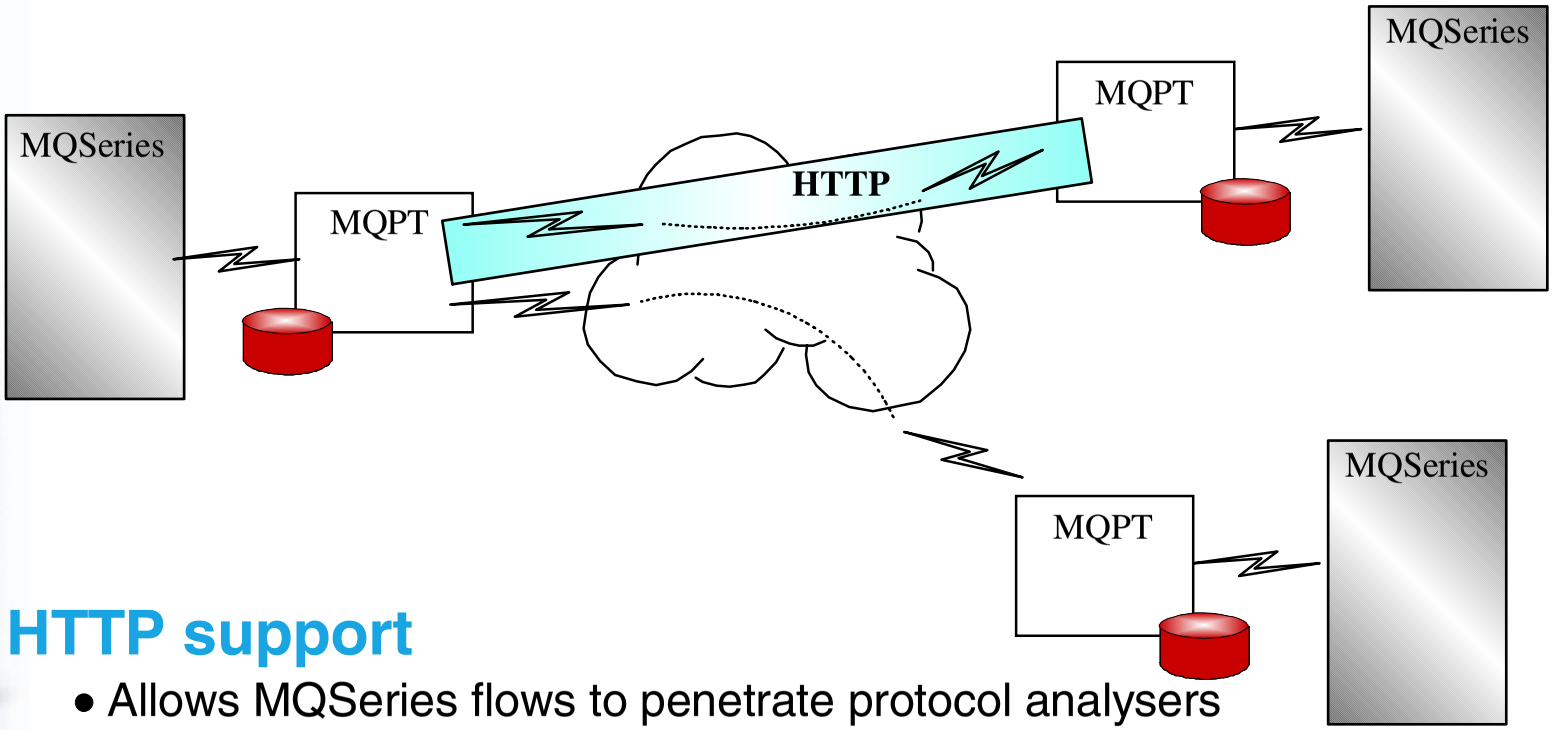
Requires source port configuration

- Configurable for MQI Client

Requires target port configuration

- Configurable for the TCP Listener

MQSeries PassThru



HTTP support

- Allows MQSeries flows to penetrate protocol analysers

Disconnected comuncations channels

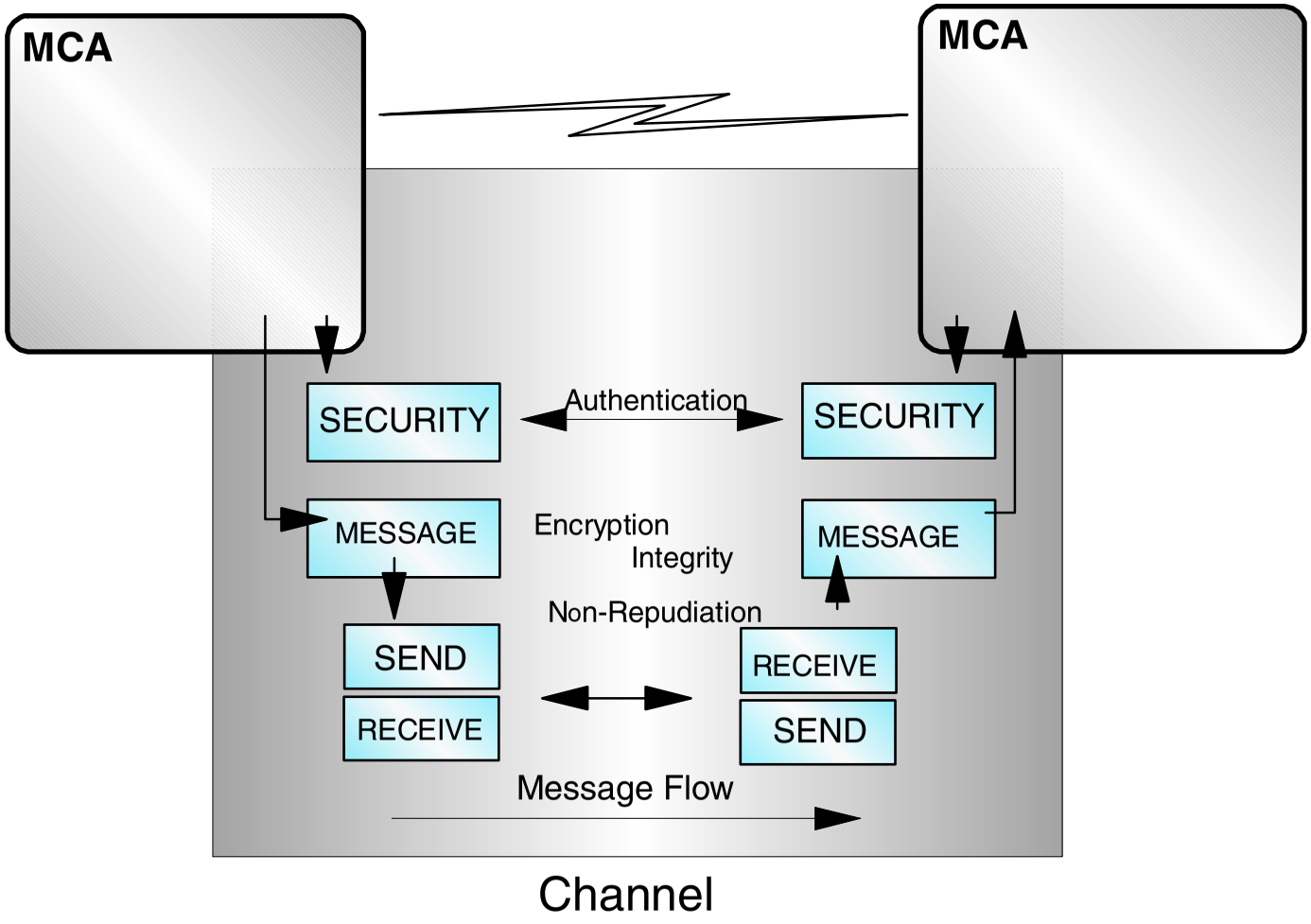
- No direct connection between queue managers

Single MQPT supports multiple targets

- target port on daemon identifies target queue manager

Category 3 supportpac for Windows NT

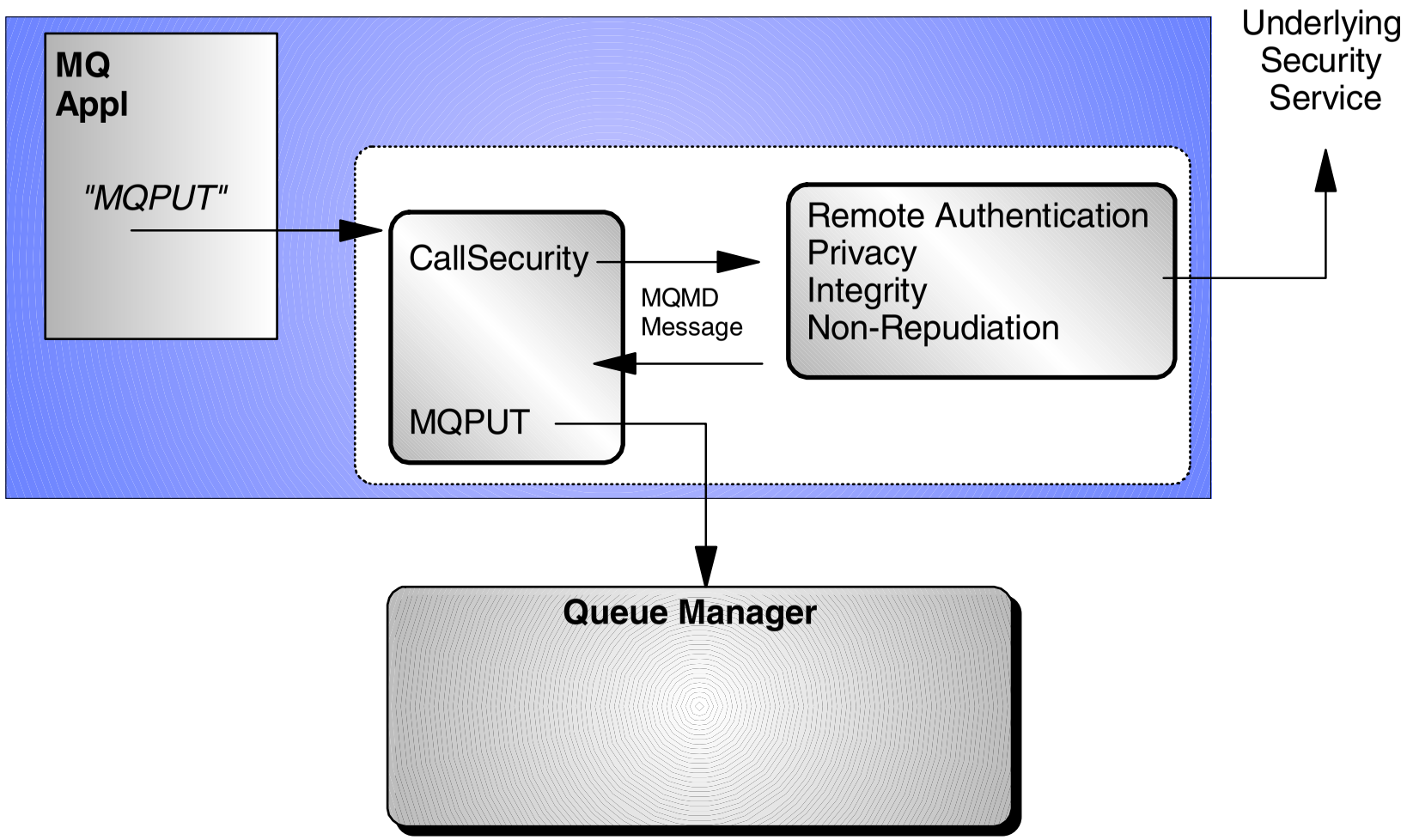
Application Level Security...MCA Exits



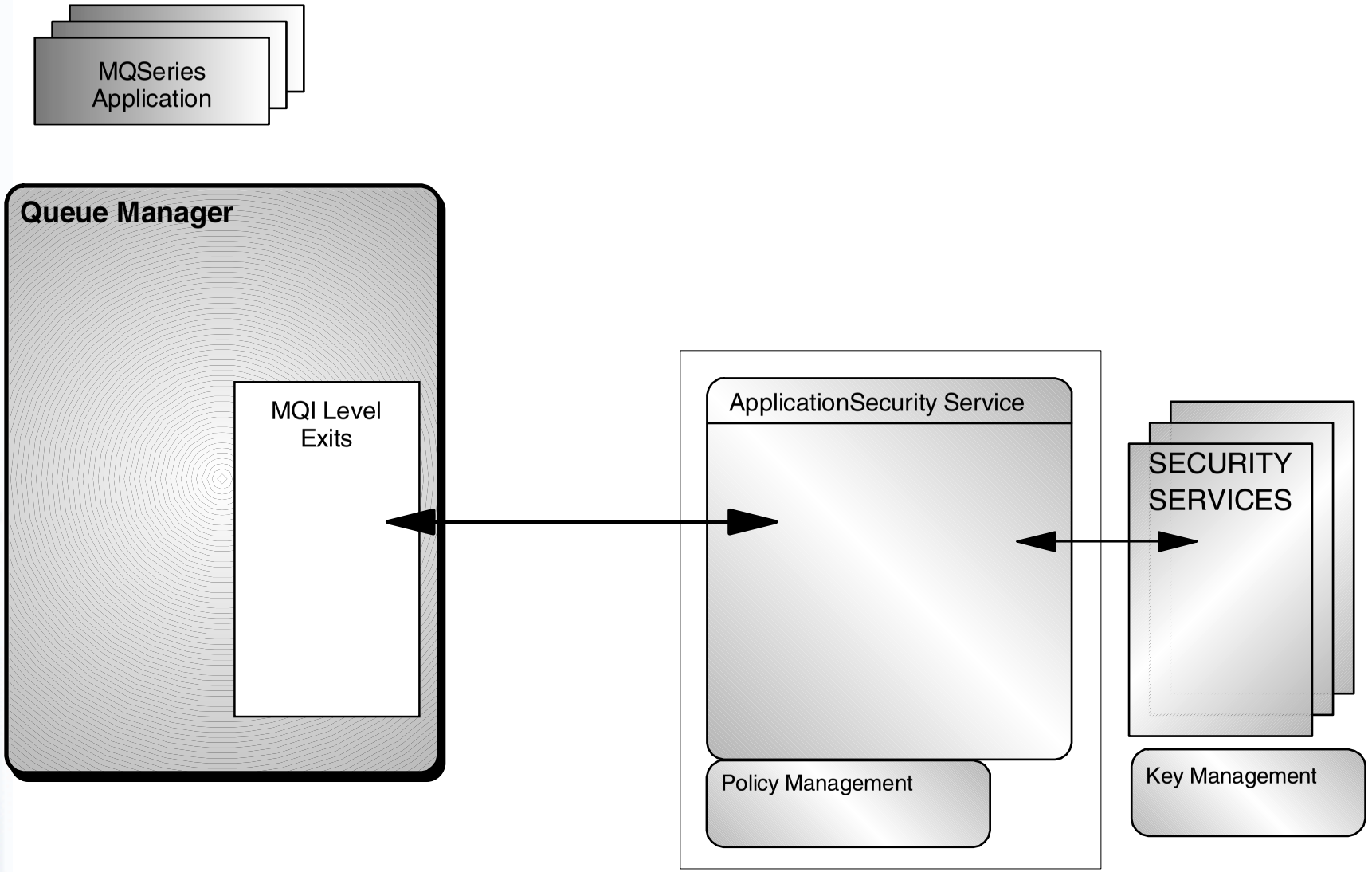
Transport level functions only

- No end-to-end security support

Application Level Security...Embedded API



Future Security Directions

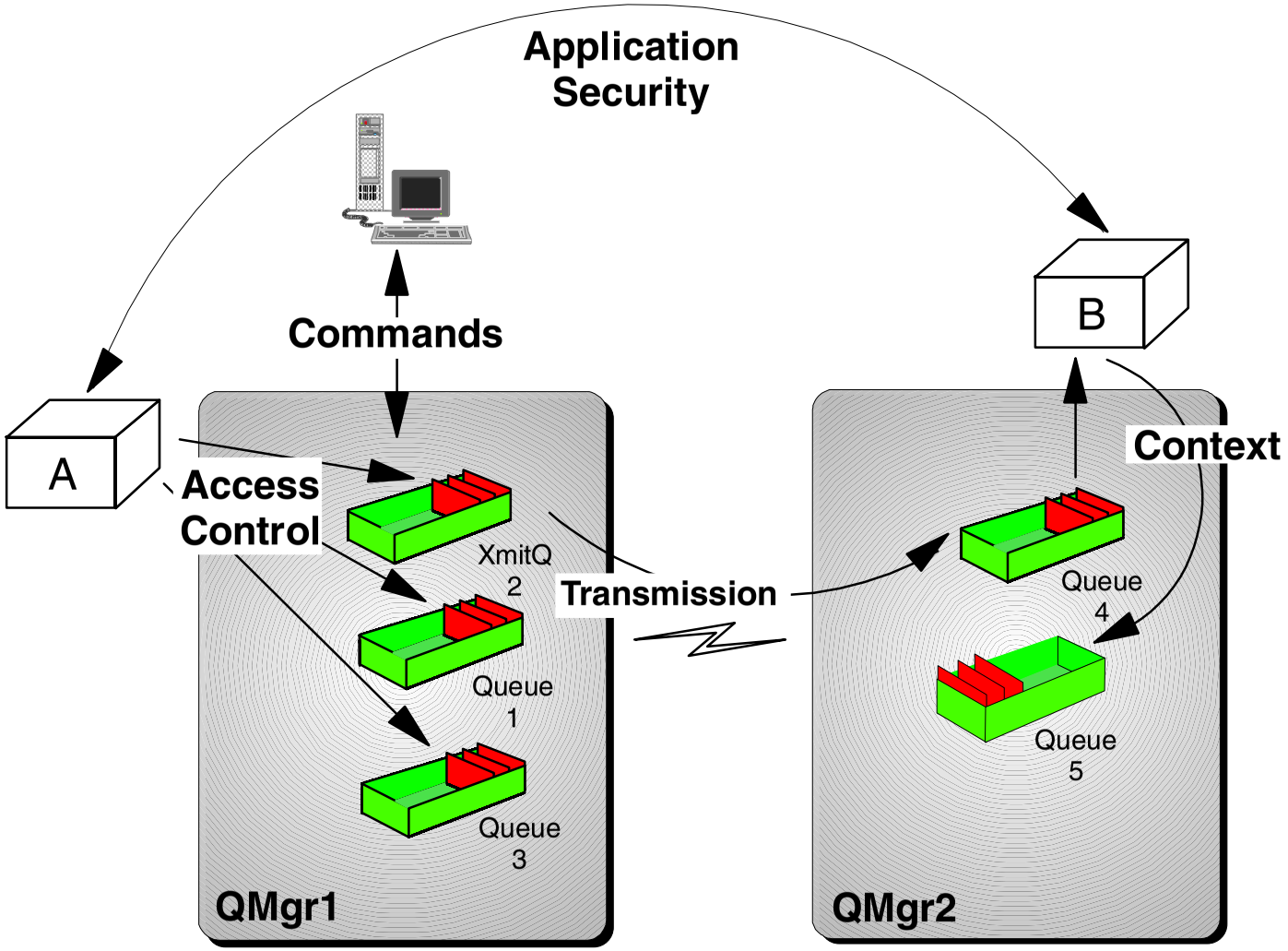


Future Directions - Notes

The principal missing component of MQSeries security is application level security. Any implementation of application level security for MQSeries will (most likely) have the following characteristics:

- Application level security functions are not only concerned with transmission of messages across some network. It is equally important to provide these security functions within a single Queue Manager. The natural implication here is that a solution which is implemented only within the MCA exits will not be complete.
- (Particularly for the distributed platforms), the implementation must not have pre-requisiste security products/mechanisms. There should be no dependence, for example, on DES, DCE, RSA. This means that any implementation should be independent of the underlying security mechanisms and should be entirely replaceable.
- There should be as little security function as possible in the Queue Manager kernel functions - particularly in terms of managing the environment, which should ideally be managed as a part of the overall security management functions.

Summary


Application
Security
Commands
A
B
Access
Control
XmitQ
2
Queue
1
Queue
3
Transmission
Queue
4
Queue
5
Context
QMgr1
QMgr2

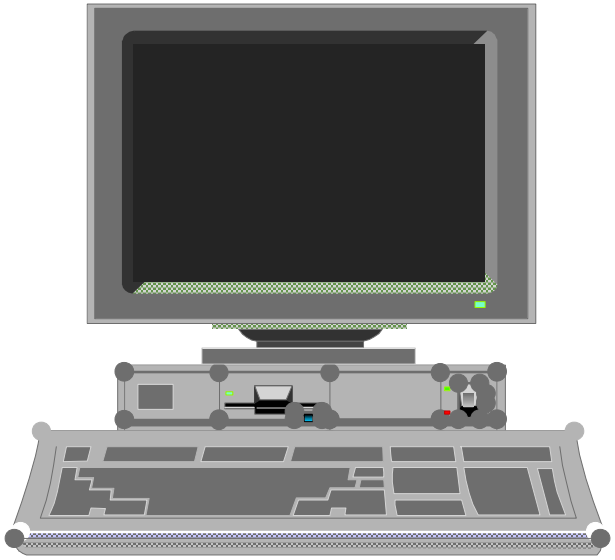
Summary - Notes

For further reading on the subject of MQSeries security, see the following MQSeries publications:

• MQSeries for MVSESA system Management Guide	SC33-0806-03
• MQSeries for AIX System Management Guide	SC33-1373
• MQSeries for OS/2 System Management Guide	SC33-1371
• MQSeries for Windows NT System Management Guide	SC33-1643
• MQSeries Distributed Queuing Guide	SC33-1139-07
• MQSeries Clients	GC33-1632-03

There is also a white paper on the subject of MQ Security, available from IBM, via the MQSeries Home Page as SupportPac MS06. This White Paper has recently been updated.

MQSeries Web Site



http://www.software.ibm.com/ts/mqseries/