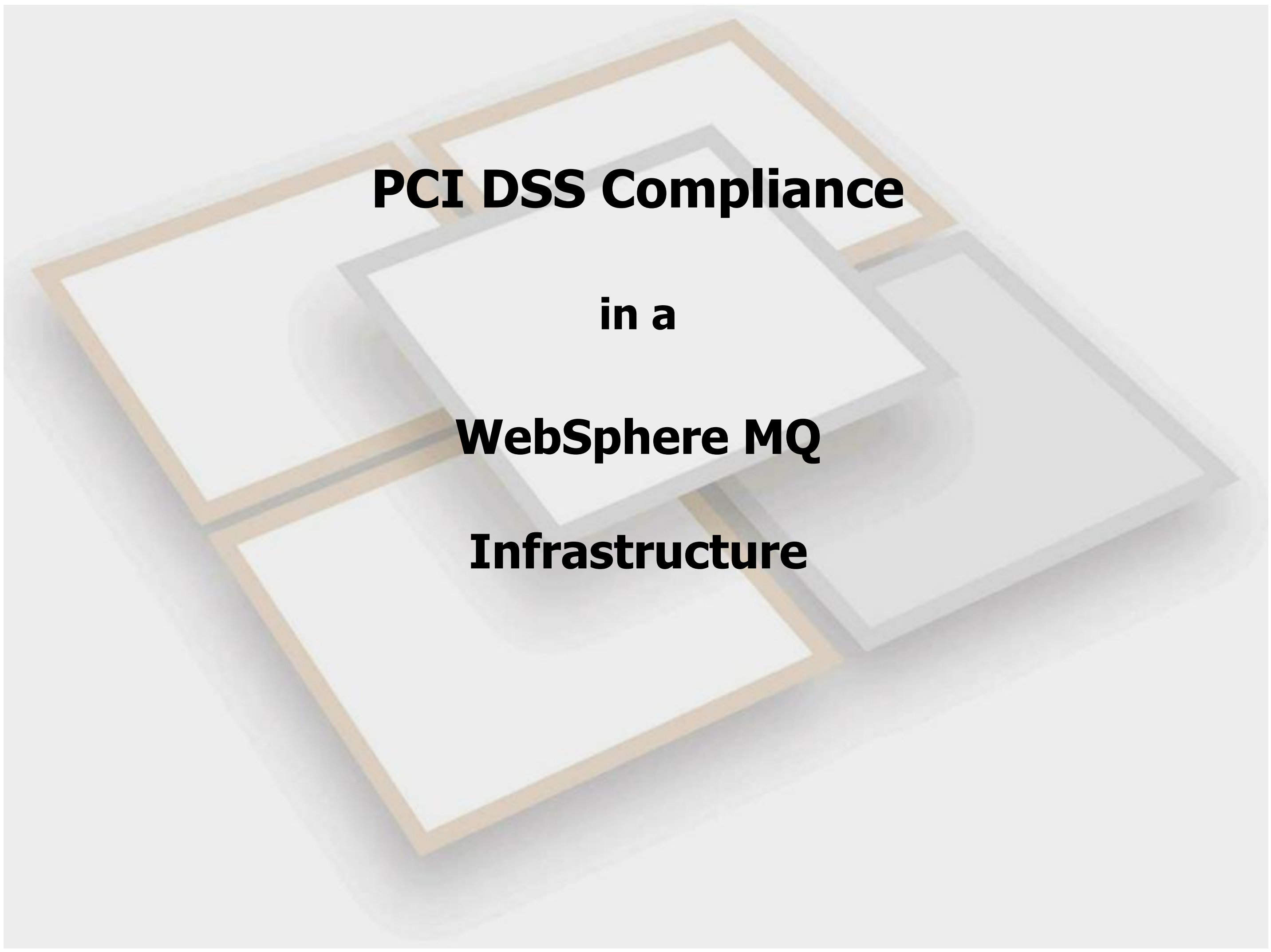
PRIMEUR



PCI DSS Compliance

in a

WebSphere MQ

Infrastructure

Primeur UK Ltd
32 Sackville Street
Royalty House
London W1S 3EA
United Kingdom
☎ +44 (0)20 3239 7456
🖷 +44 (0) 20 7432 6051

Primeur Group
Corso Paganini 3
16125 Genova
Italy

☎ +39 010 27811
🖷 +39 010 868 4913

www.primeur.com

March 2010

Table of Contents

Executive Summary

The Payment Card Industry (PCI) Data Security Standard (DSS) was developed to encourage and enhance cardholder data security and facilitate the broad adoption of consistent data security measures globally. The PCI Security Standards Council (PCI SSC) www.pcisecuritystandards.org was set-up in 2006 and owns and develops the standards and raises awareness of them. The PCI SSC is jointly-owned by the world's major card brands.

Applying PCI standards and being compliant is no longer the exclusive domain of the Banks. Anyone who processes cardholder data is subject to PCI sooner or later, and many deadlines have already passed.

The 'problem' with PCI is that it is both prescriptive and specific.

- Prescriptive – because it outlines standards and rules that must be adhered to
- Specific – because it outlines minimum baseline controls

This has been done deliberately and there is no room for manoeuvre as there is with many other standards.

PCI standards are very clear about data protection (see table below). PCI DSS requirements are applicable if a Primary Account Number (PAN) is stored, processed, or transmitted. If a PAN is not stored, processed, or transmitted, PCI DSS requirements do not apply.

Many organisations, financial and non-financial, use WebSphere MQ to move cardholder data and most will have implemented encryption of data in transmission using the SSL capability provided free of charge with WebSphere MQ.

Unfortunately, with regards to WebSphere MQ, this is not enough. WebSphere MQ itself can and does store messages (data) on the file system in clear text even when SSL is being used. A further exposure in WebSphere MQ is that anyone with access to the file system and some knowledge of WebSphere MQ, can modify the content of messages held in the file system without anyone or WebSphere MQ knowing it has happened.

This paper explores these and other exposures in the use of WebSphere MQ in more detail.

	Data Element	Storage Permitted	Protection Required	PCI DSS Req. 3.4
Cardholder Data	Primary Account Number (PAN)	Yes	Yes	Yes
	Cardholder Name [1]	Yes	Yes [1]	No
	Service Code [1]	Yes	Yes [1]	No
	Expiration Date [1]	Yes	Yes [1]	No
Sensitive Authentication Data [2]	Full Magnetic Stripe Data [3]	No	N/A	N/A
	CAV2/CVC2/CVV2/CID	No	N/A	N/A
	PIN/PIN Block	No	N/A	N/A

Source: PCI DSS Requirements and Security Assessment Procedures, v1.2.1

1. These data elements must be protected if stored in conjunction with the PAN. This protection should be per PCI DSS requirements for general protection of the cardholder data environment. Additionally, other legislation (for example, related to consumer data protection, privacy, identity theft, or data security) may require specific protection of this data, or proper disclosure of a company's practices if consumer-related personal data is being collected during the course of business. PCI DSS, however, does not apply if PANs are not stored, processed, or transmitted.
2. Sensitive authentication data must not be stored after authorisation (even if encrypted).
3. Full track data from the magnetic stripe, magnetic stripe image on the chip, or elsewhere.

Introduction

Protecting sensitive cardholder data is not simply a question of complying with a standard and putting a tick in a box that yet another audit has been passed. There is a cost impact for failing to protect data, and there are financial benefits to be obtained through achieving and maintaining compliance.

Data breaches cost companies real money and, for the individuals who's credit card data is stolen or compromised, untold misery and inconvenience. The real money costs for companies are not insignificant. We have seen estimates of the cost per compromised record as low as £64 and as high as £112.

On-going reductions in the cost of PCI audits and increased operational efficiency are also possible if sensible approaches are taken. Making the right technology and implementation choices around encryption and tokenisation can have a direct impact on whether costs reduce or increase.

Within most of the market sectors where PCI DSS compliance is relevant, the dominant messaging infrastructure is IBM's WebSphere MQ. It is this infrastructure which is moving vast numbers of messages which contain data which should be protected. Thus, the question that should always be asked by any company using WebSphere MQ is; "How does my WebSphere MQ infrastructure impact my PCI DSS compliance?"

It is, however, this question itself that often causes a problem. The members of the Security teams do not necessarily know enough about WebSphere MQ to really assess its compliance to the security standards, and the closest many WebSphere MQ teams get to security is implementation of SSL encryption. How to create an efficient and open bridge between these two teams can be a challenge, but it is a bridge that needs to be built and maintained if cost effective compliance is to be achieved.

Since the early 2000's, IBM have shipped SSL security for free along with WebSphere MQ – but is this enough to meet PCI compliance? On its own, the answer is definitely 'No' as SSL only protects data whilst it is in transit. As one of our customers said to us recently, "we now realise that we do inadvertently store data unprotected and that sometimes it is WebSphere MQ doing this as a normal part of the way it operates".

Myths and Inaccuracies

When we talk to companies using WebSphere MQ about security, some get it and some don't. Some of the more regular comments from those that don't get it include:-

"We have a policy of not storing data in WebSphere queues. All data to be stored must be offloaded from the queues and placed in secure data storage".

- A nice idea and it reads well as a policy to show auditors that you are (on paper) compliant. Unfortunately, the policy potentially has two large holes in it.
    - The first is that WebSphere itself stores persistent messages (data) in the file system. Not only can this data be read by anyone with the appropriate access to the file system, it can also be modified without WebSphere MQ knowing that the data has been tampered with.
    - The second is that mainframe batch applications – such as payment systems - often read data directly from WebSphere MQ queues. Queues are filling up with messages waiting to be processed at the next scheduled batch run and whilst they are filling up the data in them can be read by anyone with access. The same situation arises with transmission queues – if a channel is down or stopped, messages will queue up until the channel is restarted.

"No unauthorised person can access WebSphere queues. We have strict access and authorisation management policies. For sensitive queues it really is a need to know basis".

- Regardless of the controls in place, anyone with root administrator access can read any data stored in clear text on a Windows or Unix system which means that the file system representation of WebSphere MQ queues are open to be accessed and read if they are in clear text.

"Our WebSphere infrastructure is safe as it all sits behind the firewall".

- So why is it that, after Compliance, research shows the biggest security driver for most companies to be Insider Risk?

"Audit has never raised any concerns".

- How knowledgeable about WebSphere MQ are the people in Audit? Have you really got the proper controls and systems in place to be compliant, or is it a case of "in the land of the blind the one eyed man is king"?

"We use SSL throughout our WebSphere MQ infrastructure so we're OK"

- This 'excuse' has been used for many years to avoid the costs and effort of having to implement stronger protection. It sounds good because SSL implies encryption, mutual authentication, and no tampering but anyone with knowledge of WebSphere MQ knows that; a) data encryption and integrity is only provided during its transfer, thus only preventing hackers who are sniffing the network, and b) mutual authentication is only between the two physical machines and doesn't authenticate the persons or applications doing the sending and receiving.

What next?

Primeur has been providing security solutions and consultancy since 1989. We are an IBM Premier level Business Partner with a particular focus on WebSphere and in 1999, soon after WebSphere MQ became available, we developed DataSecure for MQ (DSMQ) to provide both Link level (SSL) and End-to-End (E2E) security.

These two sets of skills allow us to bridge the gap between security and WebSphere MQ. Far too often we see a worrying lack of security in many WebSphere MQ implementations either through ignorance or complacency.

The remainder of this document, using the example of Primeur's DSMQ product, explores what is needed to help make WebSphere MQ a part of a PCI compliant infrastructure.

PCI Data Security Standard (PCI –DSS)

Overview
PCI DSS comprises 6 overall PCI goals. Each goal is broken down into 2 or 3 specific PCI DSS requirements, numbering 12 in total (sometimes referred to as the 'digital dozen').

The goal and requirements are summarised in the following table.

PCI Goal	PCI DSS Requirements
Build and maintain a secure IT network	1. Install and maintain a firewall configuration to protect cardholder data
	2. Do not use vendor-supplied defaults for system passwords and other security parameters
Protect cardholder data	3. Protect stored cardholder data
	4. Encrypt transmission of cardholder data across open, public networks
Maintain a vulnerability management program	5. Use and regularly update anti-virus software on all systems commonly affected by malware
	6. Develop and maintain secure systems and applications
Implement strong access control measures	7. Restrict access to cardholder data by business need-to-know
	8. Assign a unique ID to each person with computer access
	9. Restrict physical access to cardholder data
Regularly monitor and test networks	10. Track and monitor all access to network resources and cardholder data
	11. Regularly test security systems and processes
Maintain an information security policy	12. Maintain a policy that addresses information security

DataSecure for MQ (DSMQ)

In the following table, specific PCI DSS requirements that can be met by using DSMQ are identified and a description of how DSMQ meets this requirement is given. A full list and description of the PCI DSS requirements can be found at PCI Security Standards Council (PCI SSC) web site at this URL:- www.pcisecuritystandards.org

Implementing security policies and using DSMQ may have a positive impact on other PCI DSS requirements. This paper does not attempt to identify these as they will be specific to the individual implementation.

PCI DSS Requirement	DSMQ Capability
3. Protect Card Holder Data	
3.4 Render the Primary Account Number (PAN), at minimum, unreadable anywhere it is stored (including on portable digital media, backup media, in logs) by using any of the following approaches: ▪ One-way hashes based on strong cryptography ▪ Truncation ▪ Index tokens and pads (pads must be securely stored) ▪ Strong cryptography with associated key-management processes and procedures The MINIMUM account information that must be rendered unreadable is the PAN.	DSMQ provides current state of the art strong cryptography, exclusively based on recognized industry and legal standards. ▪ Symmetric: DES , T-DES, AES ▪ Asymmetric: RSA up to 4096 ▪ Hashing: MD5, SHA1, SHA2 ▪ Crypto envelopes: PKCS#7 ▪ Crypto standards: PKCS, PKI, X.509
3.5 Protect cryptographic keys used for encryption of cardholder data against both disclosure and misuse.	Keys are protected within a PKCS#11 compliant Token, encrypted according to PKCS#5 standard. The token can be implemented in software or, for maximum security, within Hardware such as Smart Cards, USB Tokens, or proper HSMs. The latter is the proper way to store crypto keys in the most secure fashion and strongly advisable.
3.5.2 Store cryptographic keys securely in the fewest possible locations and forms.	Tokens can and will be fully reused by all applications needing them in whatever format they have been implemented. Tokens are a piece of security infrastructure and as such application and operating system independent.
3.6 Fully document and implement all key-management processes and procedures for cryptographic keys used for encryption of cardholder data.	DSMQ fully supports PKI key management procedures.
4. Encrypt transmission of cardholder data across open, public networks	
4.1 Use strong cryptography and security protocols such as SSL/TLS or IPSEC to safeguard sensitive cardholder data during transmission over open, public networks.	DSMQ E2E also supports SSL/TLS as a robust security protocol although the WebSphere MQ SSL is more than adequate for link level authentication and encryption if that is all that is needed
7. Restrict access to cardholder data by business need to know	
7.1 Limit access to system components and cardholder data to only those individuals whose job requires such access. Access limitations must include the following:-	WebSphere MQ itself can and does store persistent messages (data) on the file system in clear text even when SSL is being used. A further exposure in WebSphere MQ is that anyone with access to the file system and some knowledge of WebSphere MQ, can by-pass the WMQ API and modify the content of messages held in the file system without anyone or WebSphere MQ knowing it has happened. Tightening up Role Based Access Control for access to the WMQ queue managers does not alone mitigate this risk – at best it controls it to certain trusted individuals. Using DSMQ E2E to add a digital signature adds mitigation to this risk as DSMQ E2E will know that the digital signature has been broken and will stop the message from being processed (the WMQ GET will fail). Additionally if DSMQ E2E was also used to encrypt the message, then obviously the raw data would not be visible to change.
7.1.1 Restriction of access rights to privileged user IDs to least privileges necessary to perform job responsibilities	
7.1.2 Assignment of privileges is based on individual personnel's job classification and function	

PCI DSS Requirement	DSMQ Capability
10. Track and monitor all access to network resources and cardholder data.	
10.1 Establish a process for linking all access to system components (especially access done with administrative privileges such as root) to each individual user.	DSMQ can be used just to log activity (WMQ API calls) on a particular set of queues and writes this audit data to a WMQ queue which can itself be protected by DSMQ. This obviously would not stop somebody being able to add a message but ensuring that all users with access to the queues know that logging and audit is in place will help in deterrence. In addition, if there is a breach, the perpetrator should be identifiable.
10.2 Implement automated audit trails for all system components to reconstruct the following events:	
10.3 Record at least the following audit trail entries for all system components for each event:	
10.5 Secure audit trails so they cannot be altered.	

PRIMEUR

DSMQ Product Description

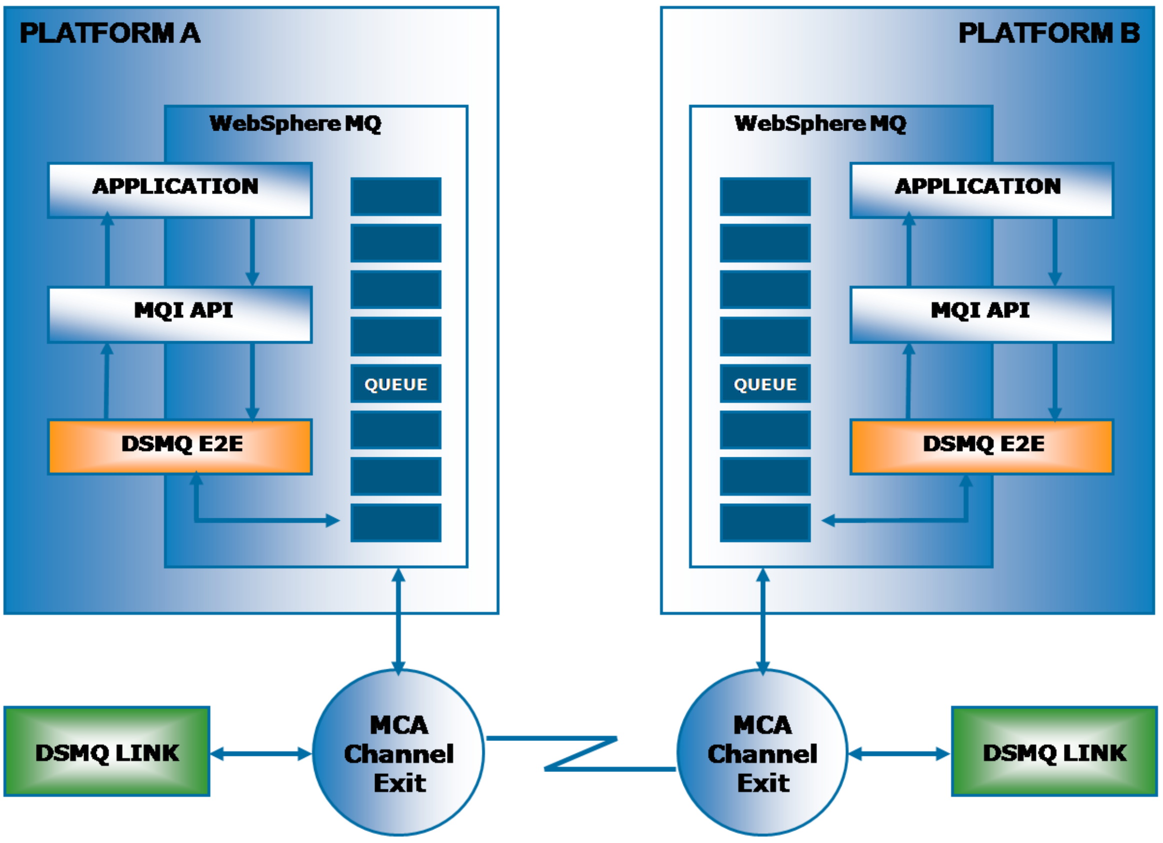
DSMQ is specifically designed to provide cryptographic protection for messages sent using WebSphere MQ without requiring changes to the application. Data is protected in the form of PKCS#7 Data Objects (or messages). WebSphere MQ messages may be either signed or signed and encrypted. In addition to the cryptographic protection of messages, DSMQ E2E can also be used to compress message data using an LZW based compression technique. The compression of messages is particularly relevant when cryptographic processing is to be applied, as a PKCS#7 format message (or Data Object) will be larger than the clear text message from which it is created. A further benefit from compression is that the cost of compressing and then encrypting a message is typically lower than the cost of just encrypting the same message.

All message types including segmented messages and Distribution Lists are fully supported. If messages are encrypted, they are encrypted on the queues, and also in the WebSphere MQ log files. It should be stressed that only the message payload is encrypted – the WebSphere MQ headers are not touched as this would prevent WebSphere MQ processing the messages.

How the product works

DSMQ works by pre and post processing the WebSphere MQ API calls. To achieve this we use a call interception technique for OS/390 batch applications and OS/390 CICS is supported using the WebSphere MQ CICS API crossing exit (CSQCAPX). For WebSphere MQ 5.3 - and later versions - on distributed platforms the WebSphere MQ API crossing exit is used.

Processing is controlled entirely by the DSMQ configuration. DSMQ will only perform its processing on messages when both the logged on user and the queue being opened have appropriate entries in the configuration.



DSMQ End-to-End (E2E) Processing

When DSMQ intercepts an MQPUT issued by an application, it checks whether E2E processing is to be performed for messages put to this queue based on the configuration data read when the queue was opened. If DSMQ E2E processing is not to be performed, the MQPUT call proceeds normally.

Data that is being sent by the application will have a specific format (for example it might be character data (MQFMT_STRING) that is to be converted from the sending queue manager's CCSID to the receiving queue manager's CCSID when the MQGET API is issued with the MQGMO_CONVERT option). When data is compressed or has cryptographic processing applied to it the resulting data will be binary data and will be a different length from the original. It is possible to configure channels to do data conversion as the data passes through the MQ network, but the data that DSMQ sends after processing will not survive such conversion. For this reason, the original message format from the MQMD is preserved at the point where the original application puts the message, and sets the message format to MQFMT_NONE.

The original message format, CCSID, encoding and other information such as the original message length and the actual DSMQ E2E processing that has been applied are saved in DSMQ's own message header (the Data Secure Message Descriptor or DSMD) which is the first part of the data actually put to the queue. This message header is followed by the data resulting from the compression and, or cryptographic processing by DSMQ E2E. Data that is less than 100 bytes is not compressed by DSMQ E2E regardless of the configuration.

Any errors in processing are notified to the application using the completion code and reason code in the normal way. The reason codes that DSMQ E2E uses are outside the range used by WebSphere MQ.

For MQGET processing, as with MQPUT, if no DSMQ E2E processing is to be performed, then the MQGET is allowed to proceed normally.

If the message on the queue was not processed by DSMQ E2E, and the MQGET policy allows this, then the message is returned normally and any data conversion requested by the application is performed. This allows the receiving application to process messages that were sent from applications for which DSMQ E2E is not configured, and from those for which it is. A staged roll out of DSMQ E2E within the enterprise is therefore simply enabled.

If the MQGET policy only allows messages that are signed and encrypted (for example), and the message on the queue is not signed and encrypted, the message from the queue is returned to the application without the removal of our header (if present) and without decryption, decompression or data conversion. The completion code and reason code are set to indicate this.

If the message on the queue is signed or signed and encrypted, and the distinguished name of the sender does not match one of the authorised senders (if specified), the message is returned to the application as it is on the queue (as above). The completion code and reason code are set accordingly.

Otherwise, if the data was signed or signed and encrypted, the P7GETD() API call is used to decrypt the data if necessary and to validate the signature. If this fails (for example the signature is not valid) the data from the queue is returned as above, and the completion code and reason code are set. The data is then decompressed if necessary.

The DSMD header is removed and the original message format, encoding and so on are restored in the MQMD. Finally any data conversion requested by the application is performed.

Additional capabilities of DSMQ End to End

The techniques described above allow an End-to-End data security solution to be delivered for WebSphere MQ that provides the services of message integrity and proof of origin (by using a digital signature) and if so wished message privacy by means of data encryption.

On their own however, these techniques may not provide an adequate solution for meeting regulatory or other requirements for after the event non-repudiation, the provision of audit trails, or to support the use of a message broker. How DSMQ supports these requirements is explained below:

Non-repudiation support

In situations where messages are being exchanged between parties who do not necessarily trust one another, it can be useful for the recipient of a message to be able to prove at a later date that the other party sent a particular message. By using a digital signature on the message the identity of the sender and the authenticity of the message can be confirmed, and therefore the sender of the message is unable to deny (or repudiate) the sending of that message.

If the recipient of a digitally signed or a digitally signed and enveloped (encrypted) message wishes to use the digital signature for this purpose, then they need to archive the message in its signed or signed and enveloped (encrypted) form. If this is done, should a dispute arise at a later date, the recipient can retrieve the message from the archive and check the signature again. This is not the only requirement for ensuring that messages cannot be repudiated, but it is a necessary one.

There are a number of other requirements for the process of non-repudiation. The most important are:

- A suitable legal framework exists to allow the use of digital signatures for this purpose. This may take the form a contract, or be part of national or supra-national legislation.
- If necessary, the communicating parties have agreed to the use of digital signatures for this purpose.
- If required by the legal framework (this is normally the case), the certificates used for checking the signature should include key usage of "non-repudiation".

Non-repudiation is normally only necessary when receiving signed or signed and encrypted messages from another organisation as might occur in a Business to Business or Business to Consumer message exchange. It is not typically required for messages moving within the organisation where the digital signature is used purely for the purposes of authenticating the message sender and that the message has not been tampered with while in transit.

Because only some messages need to be archived for non-repudiation purposes, it is not appropriate for DSMQ End-to-End Security to archive all signed or signed and enveloped message that are received. Typically the requirement will be to archive messages retrieved from a specific queue or queues.

Each customer will typically have unique requirements for archival: Some will want to archive messages to an external database, while others will wish to write these messages to some other form of external storage. For this reason, we write the archive messages to a WebSphere MQ queue (QLOCAL, QALIAS, or QREMOTE) whose name is part of the queue configuration of DSMQ End-to-End Security. For example: We might wish to archive all signed or signed and enveloped messages read from the queue MYAPP.QUEUE. In this case we would modify the queue configuration in DSMQ E2E to specify an archive queue name of MYAPP.ARCHIVE.QUEUE.

If this is done, then once DSMQ End-to-End security has successfully validated the signature on the message, and if so configured has confirmed that the message sender was authorised, the message as it was read from the queue is written to the archive queue. Put in another way, the message written to the archive queue includes the DSMD header and the payload is still in the form of a PKCS#7 Signed or Signed and Enveloped data object or objects.

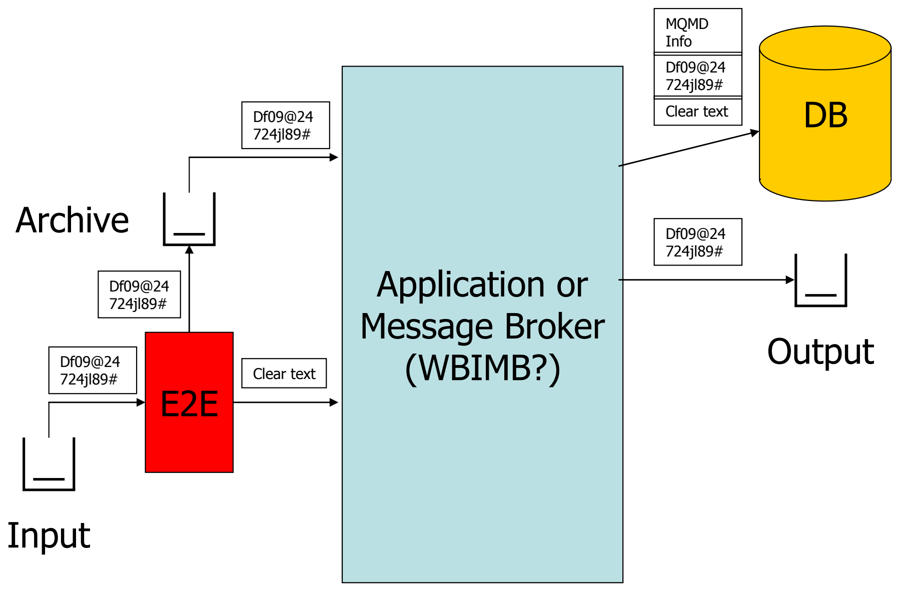
The message is written to the archive queue using MQPMO_SET_ALL_CONTEXT, so that all the information from the original message descriptor is preserved. For this reason, the user running the application reading the messages requires appropriate authority to use "set all context" on the archive queue.

The archive messages are MQPUT to the archive queue with the same syncpoint options that the user application was using to get the messages from the application queue. The archive function does not issue any commits or rollbacks.

The messages may then be read from the archive queue by the same (or another) application and saved to a database for non-repudiation purposes (where the signature is re-validated "after the event" in the event of a dispute).

Support of Message Brokers

This archive facility described above allows solutions to be developed for message brokers where it is wished for the broker to inspect the "clear text" message and yet route the original signed (or signed and encrypted) message through to the final destination. It also allows more complex solutions to be developed where the original (protected) message is archived along with the plain text, and also to transform the message and record the clear text and protected content that has been sent on to the final processing node.



Audit Log

Regulatory requirements, internal practices, or similar often require that an audit log be kept of processing applied to data. This might only require that exception conditions be logged, or that all processing be logged. It might be required for all messages processed on all queues, or just for some queues.

For this reason, DSMQ End-to-End may be configured on either a global basis or a queue by queue basis to write an audit log of its activities. This audit log may record either all DSMQ E2E related processing, or just those DSMQ E2E actions that resulted in an error condition.

The audit records could be written in a platform specific manner (such as for example using SMF records on z/OS), or a platform neutral manner using a format such as XML written to a flat file. We believe that a platform neutral solution such as XML provides a portable solution which has the advantage that it can be read by people without requiring any special processing, and also by programmes.