Unit 8. Message integrity

What this unit is about

This unit covers the various ways WebSphere MQ protects the integrity of messages. The differences between circular and linear logging are explained, and the implications of using persistence are examined. Transaction management is also covered. The method of capturing and restoring an object image is explained.

What you should be able to do

After completing this unit, you should be able to:

- Explain the difference between circular and linear logging
- Explain recovery of persistent messages
- Explain how to record and re-create an object media image
- Propose scenarios to application developers where sync point processing is appropriate

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

Unit objectives

After completing this unit, you should be able to:

- Explain the difference between circular and linear logging
- Explain recovery of persistent messages
- Explain how to record and re-create an object media image
- Propose scenarios to application developers where syncpoint processing is appropriate

Figure 8-1. Unit objectives

WM203 / VM2032.0

Notes:

Types of logs

Circular

- Log files viewed as a closed loop
- Amount of disk space required for the log does not increase with time

Linear

- Log files viewed as a sequence
- Log file never deleted but becomes inactive when it contains no entries required to restart the queue manager
- Can be archived when it becomes inactive
- Needed for media recovery

Figure 8-3. Types of logs

Notes:

WebSphere MQ records all significant changes to the data controlled by the queue manager in a recovery log. Changes include creating and deleting objects, persistent message updates, transaction states, changes to object attributes, and channel activities. The log contains the information you need to recover all updates to message queues by:

- Keeping records of queue manager changes
- Keeping records of queue updates for use by the restart process
- Enabling you to restore data after a hardware or software failure

Unless you request linear logging when you create a queue manager, circular logging is provided by default.

Periodically, the queue manager performs a log checkpoint. Information about the last checkpoint, including its location in the log, is held in the checkpoint file, amqalchk.fil.

Circular logging

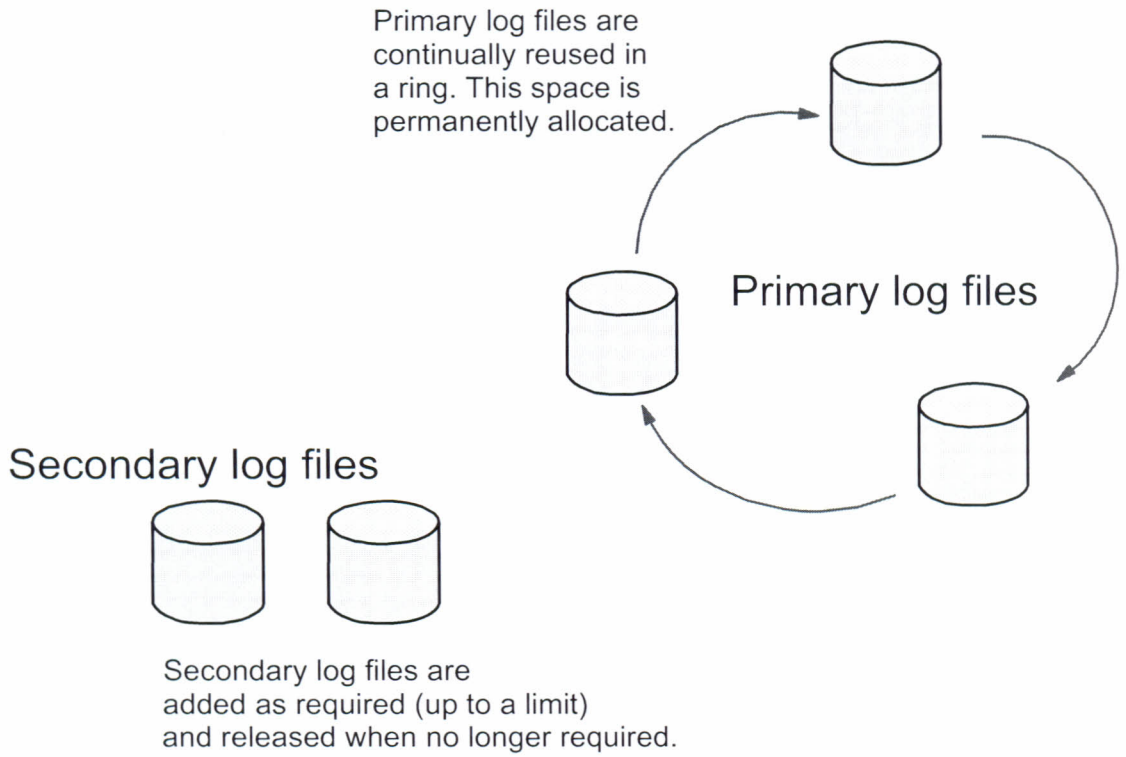


Figure 8-4. Circular logging WM203 / VM2032.0

Notes:

Circular logging is able to recover messages following a system failure but is unable to recover following a media failure. It has the advantage that the amount of disk space required for the log does not increase with time.

Linear logging

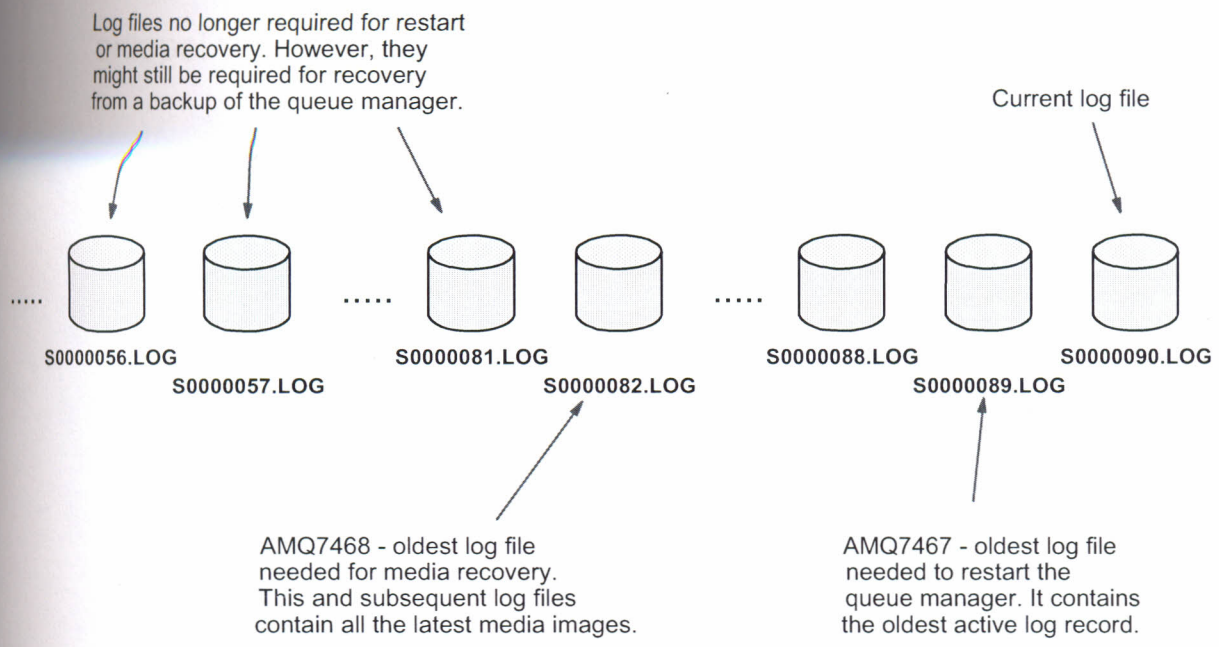


Figure 8-5. Linear logging

Notes:

A linear log can recover from a media failure but it requires inactive log files to be archived on a regular basis.

Circular logging

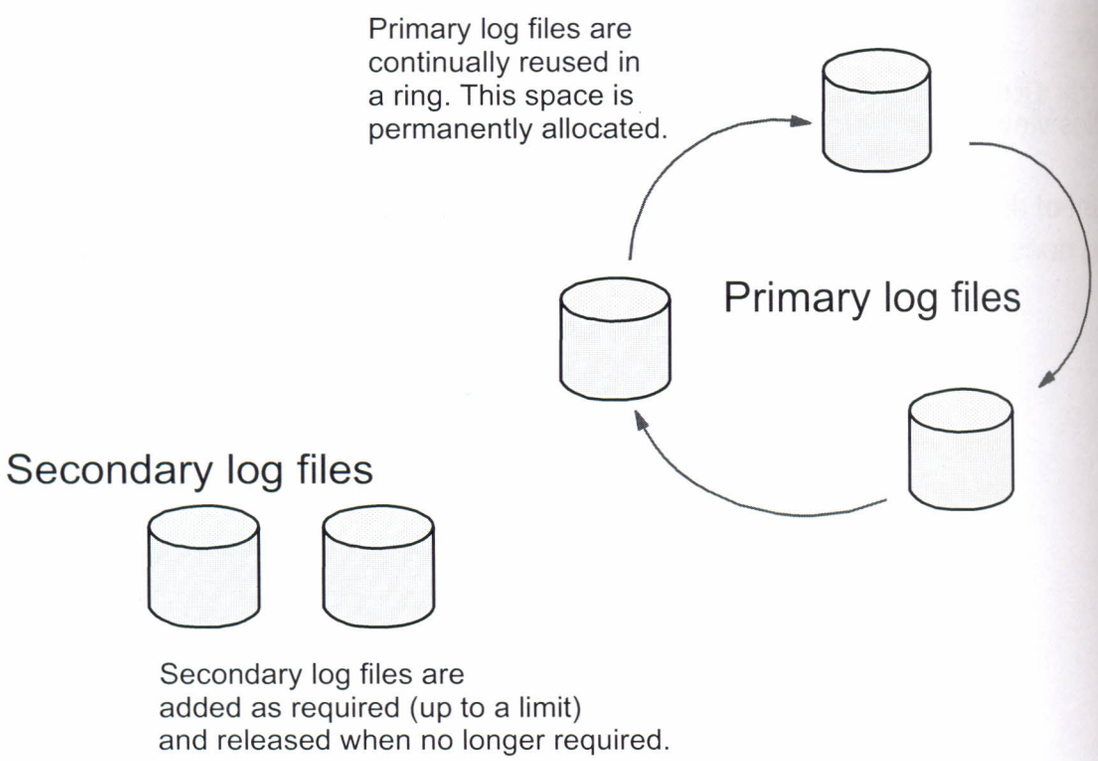


Figure 8-4. Circular logging WM203 / VM2032.0

Notes:

Circular logging is able to recover messages following a system failure but is unable to recover following a media failure. It has the advantage that the amount of disk space required for the log does not increase with time.

Recovering persistent messages

- If necessary, persistent messages are recovered automatically when the queue manager is restarted
- A damaged local queue can only be detected later
  - Reported as "object damaged"
  - Normally needs to be recovered manually
- In order to restart, a queue manager only requires:
  - Log records written since the last checkpoint
  - Log records written by transactions that were still active at the time the queue manager stopped

Figure 8-8. Recovering persistent messages WM203 / VM2032.0

Notes:

The queue manager recovers any damaged object that would prevent it from starting, but it would not normally include a local queue which is damaged. Such a queue may only be detected later when an attempt is made to access it.

In order to restart, a queue manager only requires:

- Log records written since the last checkpoint.
- Log records written by transactions which were still active at the time the queue manager stopped. Uncommitted persistent messages, put, or got inside these transactions, are rolled back during restart.

Hence, recovery can be quick.

Damaged objects and media recovery

- WebSphere MQ objects can be marked as damaged
  - Corrupt data in the queue file
  - Missing queue file
  - Disk failure
- Damaged objects can be deleted
- A damaged object can be re-created from a **linear** log
  - Known as *media recovery*
  - Media images of some objects are recorded automatically by the queue manager at certain times
  - Record the media image of a local queue on a regular basis using the control command `rcdmqimg`
- Media recovery
  - Automatic if a damaged object is detected during restart
  - For a local queue, it is normally done by using the control command `rcrmqobj`

Figure 8-9. Damaged objects and media recovery WM203 / VM2032.0

Notes:

The control command to record a media image is **rcdmqimg**. For example, the following command records a media image of a local queue.

```
rcdmqimg -m QMgrName -t qlocal QName
```

The object name can have a trailing asterisk to record any objects with names matching the portion of the name before the asterisk.

A damaged object can be re-created by using the **rcrmqobj** control command. For example, the following command re-creates a local queue.

```
rcrmqobj -m QMgrName -t qlocal QName
```

Dumping the log

- Use `dmpmqlog` to dump a formatted version of the log
- Queue manager must not be running
- By default, the dump commences from the head of the log. Optionally, the dump can commence from:
  - The base of the log
  - A log record identified by a specified *log sequence number* (LSN)
  - A log file identified by a specified *extent number* (linear logs only)
- Log records formatted include:
  - Put and get of persistent messages
  - Transaction events
  - Creation, alteration, and deletion of WebSphere MQ objects

Figure 8-10. Dumping the log WM203 / VM2032.0

Notes:

The *head* of the log is the checkpoint that commences the active portion of the log. Normally, the head of the log would be the most recent checkpoint. However, the head of the log might be positioned at an earlier checkpoint if there were transactions that were still active when the queue manager stopped and there are uncommitted persistent messages that were put or got inside these transactions before the most recent checkpoint.

- The *base* of the log is the first log record in the log file containing the head of the log.
- Each log record is identified by a unique log sequence number (LSN).
- Each log file has a file name of the form *Snnnnnnn.LOG*, where *nnnnnnn* is the extent number.

Full details, including a sample dump with notes, can be found in the *WebSphere MQ System Administration Guide*.

Syncpoint control

```
MQGET (customer order)
Update DB
MQPUT (dispatch request)
MQPUT (delivery confirmation)
Commit
```

- Option on MQPUT and MQGET calls
  - If `NO_SYNCPOINT`, message is added or removed immediately
  - If `SYNCPOINT`, result of an MQPUT or an MQGET call only becomes visible when the unit of work is committed
- Default is platform-dependent
- Additional option on an MQGET call
  - `SYNCPOINT_IF_PERSISTENT` to get message is only within syncpoint control if it is persistent

Figure 8-13. Syncpoint control WM203 / VM2032.0

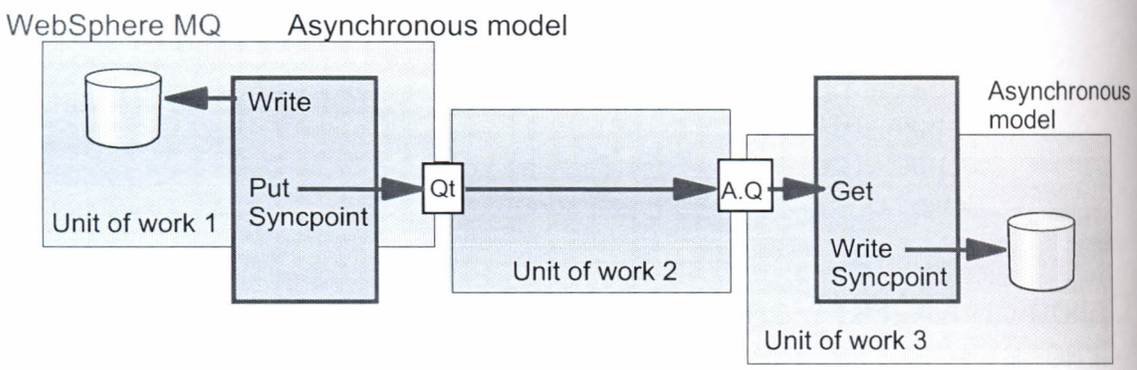
Notes:

An application can specify whether an MQPUT call or an MQGET call is within, or outside of, sync point control or leave it as the default. The default is platform-dependent.

Note

An application can get a message it has put in the same unit of work.

Compensating transactions



Local syncpoint participation = committed changes

Examples:

1. Debit account, Send credit message
   1. Account not known, Send debit reversal
2. Update travel itinerary, Send flight reservation
   2. Fully booked, Send nearest alternative
3. Confirm order, Send shipping request
   3. Out of stock so reorder, Send revised delivery date

Figure 8-14. Compensating transactions WM203 / VM2032.0

Notes:

Local sync point coordinates messages with database updates on that system. But the actual processing of the message may take place on a different system, or at a later time.

When a message is processed later, the application may find an error, in the database, that prevents the message from being processed. It is too late to roll back the original unit of work.

The answer is to include *compensating transactions* which send messages to reverse the original request. Some examples are illustrated.

Coordinating local units of work

- A **local** unit of work is one in which the only resources being updated are those of the queue manager

```
MQGET  message from server queue
MQPUT  extra requests
MQPUT  reply message

if error ...
       MQBACK
if OK ...
       MQCMIT
```


Figure 8-15. Coordinating local units of work

Notes:

The calls MQCMIT and MQBACK are supported on all queue managers except WebSphere MQ for HP NonStop Server.

On WebSphere MQ for HP NonStop Server, TM/MP (TMF) is used to coordinate local units of work. Instead of using the calls MQCMIT and MQBACK, an application changing WebSphere MQ resources within sync point control uses the calls BEGINTRANSACTION, ENDTRANSACTION, and ABORTTRANSACTION.

A WebSphere MQ client application might also use the MQCMIT and MQBACK calls.

Internal coordination of global units of work

- A **global** unit of work is one in which the resources of other resource managers are also being updated

```
MQBEGIN
MQGET  message from server queue
EXEC SQL INSERT data base record
MQPUT  reply message

if error ...
      MQBACK
if OK ...
      MQCMIT
```

AIX, HP-UX, i5/OS, Solaris, Linux, Windows only

Figure 8-16. Internal coordination of global units of work

Notes:

Internal coordination of global units of work is only supported by certain queue managers. Using the X/Open XA interface, with a two-phase commit protocol, a queue manager is able to coordinate changes to its own resources and to other resource managers within a unit of work. An external sync point coordinator is not required under these circumstances.

Database coordination

- Supported database managers

Platform	DB2	Informix	Oracle	Sybase
AIX	✓	✓	✓	✓
HP-UX	✓	✓	✓	✓
Linux	✓	✓	✓	✓
Sun Solaris	✓	✓	✓	✓
Windows	✓	✓	✓	✓

- Restrictions
  - A WebSphere MQ client cannot participate in a global unit of work
  - Only one queue manager can participate in a global unit of work
  - Normally, updates to WebSphere MQ and database resources must be made on the same system
    - Database server may be on a different system provided it can supply an XA compliant client feature

Figure 8-17. Database coordination WM203 / VM2032.0

Notes:

The figure lists the XA-compliant database managers that are supported by the queue managers and can participate in a global unit of work coordinated by WebSphere MQ.

There are some restrictions regarding the internal coordination of global units of work.

- A WebSphere MQ client application cannot participate in a global unit of work and cannot therefore issue the MQBEGIN call.
- Although a queue manager may be XA-compliant, both as a sync point coordinator and as a resource manager, it is not possible to configure two or more queue managers as participants in a global unit of work. An application can only be connected to one queue manager at a time.
- Normally, updates to WebSphere MQ resources and to resources of a database manager must be made on the same system. WebSphere MQ cannot coordinate a distributed unit of work. However, a database manager may reside on a different system to the queue manager provided it can supply an XA-compliant client feature which resides on the same system as the queue manager.

External coordination of global units of work

- AIX
  - TX Series
  - BEA Tuxedo
  - BEA WebLogic Server
  - WebSphere Application Server
- HP-UX
  - TX Series
  - BEA Tuxedo
  - BEA WebLogic Server
  - WebSphere Application Server
- Linux
  - BEA Tuxedo
  - BEA WebLogic Server
  - WebSphere Application Server
- Sun Solaris
  - TX Series
  - BEA Tuxedo
  - BEA WebLogic Server
  - WebSphere Application Server
- HP NonStop Server
  - TM/MP (TMF)
- Windows
  - TX Series
  - BEA Tuxedo
  - BEA WebLogic Server
  - WebSphere Application Server
  - MTS/COM

Figure 8-18. External coordination of global units of work WM203 / VM2032.0

Notes:

The external sync point coordinators listed on the figure use the X/Open XA interface for coordinating changes to WebSphere MQ resources and to other resource managers.

A WebSphere MQ client application can only participate in a global unit of work using the WebSphere MQ extended transactional client and an external sync point coordinator. The extended transactional client is supported on AIX, HP-UX, Linux (on x86 and zSeries platforms), Solaris, and Windows.

As this list occasionally changes, check the following URL for the latest information:
`http://www.ibm.com/software/integration/wmq/support/`

Checkpoint questions

1. For what reasons would linear logging be considered:
   a. Ease of administration
   b. Requirement that persistent queues be recoverable.
   c. Requirement that queues be rolled back to a point particular time.
   d. Production environment

2. True or false: The **dmpmqlog** command should be used to backup WebSphere MQ logs.

3. True or false: WebSphere MQ can be both an XA transaction coordinator and a transaction manager.

Figure 8-20. Checkpoint questions

Notes: