

MQSeries® per Compaq OpenVMS Alpha®



# Guida alla gestione del sistema

*Versione 5 Rilascio 1*



MQSeries® per Compaq OpenVMS Alpha®



# Guida alla gestione del sistema

*Versione 5 Rilascio 1*

**Nota**

Prima di utilizzare questo prodotto e le relative informazioni, consultare la sezione “Appendice L. Informazioni particolari” a pagina 391.

**Prima edizione (maggio 2001)**

Questa edizione si riferisce al programma MQSeries per Compaq OpenVMS Alpha, Versione 5.1 e a tutti i successivi rilasci, versioni e modifiche, se non diversamente indicato nelle nuove edizioni.

Come ultima pagina del manuale è stato predisposto un foglio riservato ai commenti del lettore. Se il foglio è stato rimosso, eventuali commenti possono essere inviati alla:

Selfin S.p.A.  
Translation Assurance  
via F. Giordani, 7  
80122 - NAPOLI

Tutti i commenti e i suggerimenti inviati potranno essere utilizzati liberamente dall'IBM e dalla Selfin e diventeranno proprietà esclusiva delle stesse.

© Copyright International Business Machines Corporation 1994, 2001. Tutti i diritti riservati.

# Indice

<b>Figure</b> . . . . .	<b>ix</b>
<b>Tabelle</b> . . . . .	<b>xi</b>
<b>Informazioni su questo manuale</b> . . . . .	<b>xiii</b>
A chi si rivolge questo manuale . . . . .	xiii
Cosa bisogna conoscere per comprendere questo manuale . . . . .	xiii
Come utilizzare questo manuale . . . . .	xiii
Utilizzo delle appendici . . . . .	xiii
Informazioni su MQSeries su Internet . . . . .	xiv

## Novità presenti in MQSeries per Compaq OpenVMS Alpha, V5.1 . . . . xv

### Parte 1. Guida . . . . . 1

#### Capitolo 1. Introduzione a MQSeries . . . 5

MQSeries e accodamento di messaggi . . . . .	5
Applicazioni indipendenti dal tempo . . . . .	5
Elaborazione dei messaggi guidati . . . . .	5
Messaggi e code . . . . .	5
Che cos'è un messaggio? . . . . .	6
Cos'è una coda? . . . . .	6
Oggetti . . . . .	7
Nomi degli oggetti . . . . .	8
Gestione degli oggetti . . . . .	8
Queue Manager MQSeries . . . . .	9
Code di MQSeries . . . . .	9
Definizioni di processo . . . . .	13
Canali . . . . .	13
Cluster . . . . .	14
Namelist . . . . .	14
Oggetti definiti dal sistema . . . . .	14
Gestione locale e remota . . . . .	15
Client e server . . . . .	15
Applicazioni MQSeries in un ambiente client-server . . . . .	15
Estensione delle funzioni Queue Manager . . . . .	16
Uscite utente . . . . .	17
Servizi installabili . . . . .	17
Sicurezza . . . . .	18
Funzione OAM (Object Authority Manager) . . . . .	18
Sicurezza DCE . . . . .	18
Supporti transazionali . . . . .	18

#### Capitolo 2. Introduzione alla gestione di MQSeries . . . . . 19

Gestione locale e in remoto . . . . .	19
Esecuzione delle attività di gestione utilizzando i comandi di controllo . . . . .	19
Esecuzione delle attività di gestione utilizzando i comandi MQSC . . . . .	20

Esecuzione delle attività di gestione utilizzando i comandi PCF . . . . .	20
Attributi MQSC e PCF . . . . .	21
Escape PCF . . . . .	21
Analisi dei nomi di file MQSeries . . . . .	21
Trasformazione del nome di Queue Manager . . . . .	21
Trasformazione del nome di oggetto . . . . .	23
Analisi della sensibilità al minuscolo/maiuscolo . . . . .	23
Sensibilità al minuscolo/maiuscolo nei comandi di controllo . . . . .	23
Sensibilità al minuscolo/maiuscolo nei comandi MQSC . . . . .	25

#### Capitolo 3. Gestione dei Queue Manager utilizzando i comandi di controllo . . . . . 27

Utilizzo dei comandi di controllo . . . . .	27
Utilizzo dei comandi di controllo . . . . .	27
Creazione di un Queue Manager . . . . .	28
Indicazioni per la creazione dei Queue Manager . . . . .	28
Creazione di un Queue Manager predefinito . . . . .	31
Avvio di un Queue Manager . . . . .	32
Rendere predefinito un Queue Manager esistente . . . . .	32
Arresto di un Queue Manager . . . . .	33
Chiusura ad attività completate . . . . .	33
Chiusura immediata . . . . .	33
Chiusura preventiva . . . . .	34
Problemi di chiusura del Queue Manager . . . . .	34
Riavvio di un Queue Manager . . . . .	34
Per eliminare un Queue Manager . . . . .	34

#### Capitolo 4. Gestione di oggetti MQSeries locali . . . . . 37

Supporto dei programmi applicativi che utilizzano MQI . . . . .	37
Esecuzione delle attività di gestione locale utilizzando i comandi MQSC . . . . .	38
Prima di iniziare . . . . .	38
Utilizzo interattivo della funzione MQSC . . . . .	39
Feedback dei comandi MQSC . . . . .	40
Termine dell'input interattivo MQSC . . . . .	40
Visualizzazione degli attributi Queue Manager . . . . .	40
Utilizzo di un Queue Manager diverso dal predefinito . . . . .	41
Modifica degli attributi di Queue Manager . . . . .	42
Esecuzione di comandi MQSC da file di testo . . . . .	42
File di comandi MQSC . . . . .	43
Notifiche MQSC . . . . .	43
Esecuzione dei file di comandi MQSC forniti . . . . .	44
Utilizzo di runmqsc per la verifica dei comandi . . . . .	44
Risoluzione dei problemi con MQSC . . . . .	45
Utilizzo delle code locali . . . . .	47
Definizione di una coda locale . . . . .	47
Definizione di una coda messaggi non recapitati . . . . .	48

Visualizzazione degli attributi predefiniti di oggetti . . . . .	49
Copia della definizione di una coda locale . . . . .	49
Modifica degli attributi di coda locale . . . . .	50
Annullamento di una coda locale . . . . .	50
Eliminazione di una coda locale . . . . .	52
Esame delle code . . . . .	52
Utilizzo di alias di code . . . . .	56
Definizione di un alias di coda . . . . .	56
Utilizzo di altri comandi con gli alias di coda . . . . .	57
Utilizzo delle code di esempio . . . . .	57
Definizione di una coda di esempio . . . . .	58
Utilizzo degli altri comandi con le code di esempio . . . . .	58
Gestione degli oggetti per triggering . . . . .	59
Definizione di una coda di applicazione per triggering . . . . .	59
Definizione di una coda di iniziazione . . . . .	60
Creazione di una definizione di processo . . . . .	60
Visualizzazione della definizione di processo . . . . .	61

## Capitolo 5. Automazione delle attività di gestione . . . . . 63

Comandi PCF . . . . .	63
Attributi MQSC e PCF. . . . .	64
Escape PCF . . . . .	64
Utilizzo di MQAI per semplificare l'impiego di PCF . . . . .	64
Gestione del server di comando per una gestione remota . . . . .	65
Avvio del server di comando . . . . .	65
Visualizzazione dello stato del server di comando . . . . .	65
Arresto del server di comando . . . . .	67

## Capitolo 6. Gestione di oggetti remoti di MQSeries . . . . . 69

Canali, cluster e accodamento remoto. . . . .	69
Gestione remota mediante cluster . . . . .	70
Gestione remota da un Queue Manager locale mediante comandi MQSC . . . . .	71
Preparazione dei Queue Manager alla gestione remota . . . . .	71
Preparazione di canali e code di trasmissione per la gestione remota . . . . .	72
Definizione di canali e code di trasmissione . . . . .	73
Avvio dei canali . . . . .	74
Definizione automatica dei canali . . . . .	75
Emissione remota di comandi MQSC. . . . .	75
Funzionamento dei Queue Manager su MVS/ESA. . . . .	76
Eventuali problemi nell'utilizzo remoto di MQSC . . . . .	77
Creazione di una definizione locale di una coda remota . . . . .	77
Analisi del funzionamento delle definizioni locali di code remote . . . . .	77
Un modo alternativo per inserire messaggi in una coda remota. . . . .	79
Utilizzo di altri comandi con code remote . . . . .	79
Creazione di una coda di trasmissione . . . . .	79
Utilizzo di definizioni di coda remota come alias. . . . .	80

Alias di Queue Manager . . . . .	80
Alias di coda di risposta RTQ . . . . .	81
Conversione dei dati . . . . .	81
Modifica del CCSID del Queue Manager . . . . .	83

## Capitolo 7. Protezione di oggetti di MQSeries . . . . . 85

Perché bisogna proteggere le risorse di MQSeries. . . . .	85
Prima di iniziare. . . . .	85
ID utente in MQSeries per Compaq OpenVMS con MQM identificativo risorse. . . . .	85
Per ulteriori informazioni. . . . .	86
Comprendere Object Authority Manager . . . . .	86
Come funziona OAM . . . . .	87
Gestione dell'accesso mediante gli identificativi di diritti . . . . .	87
Identificativo diritti predefinito. . . . .	88
Le risorse che è possibile proteggere con OAM . . . . .	88
Utilizzo degli identificativi diritti per le autorizzazioni . . . . .	88
Disabilitazione di OAM (Object Authority Manager) . . . . .	89
Utilizzo dei comandi di OAM (Object Authority Manager) . . . . .	89
Cosa specificare durante l'utilizzo dei comandi di OAM . . . . .	89
Utilizzo del comando setmqaut. . . . .	91
Autorizzazioni di accesso. . . . .	92
Visualizzazione del comando di autorizzazione . . . . .	92
Istruzioni per OAM (Object Authority Manager) . . . . .	92
ID utente . . . . .	92
Directory del Queue Manager . . . . .	93
Code . . . . .	93
Autorizzazione per utente alternativo. . . . .	93
Autorizzazione contestuale . . . . .	94
Considerazioni sulla sicurezza remota . . . . .	94
Sicurezza del comando di canale . . . . .	95
Comprendere le tabelle di specifica delle autorizzazioni . . . . .	96
Autorizzazioni MQI . . . . .	96
Autorizzazioni di gestione . . . . .	99
Autorizzazioni per comandi MQSC nei PCF escape. . . . .	100
Comprendere i file di autorizzazione . . . . .	102
Percorsi dei file di autorizzazione . . . . .	102
Cosa contengono i file di autorizzazione . . . . .	103
Gestione dei file di autorizzazione . . . . .	105

## Capitolo 8. Handler di code messaggi non recapitati di MQSeries . . . . . 107

Richiamo del DLQ handler. . . . .	107
Modello di DLQ handler, amqsdldq . . . . .	108
Tabella regole DLQ handler . . . . .	108
Dati di controllo . . . . .	109
Regole (schemi e azioni). . . . .	111
Convenzioni relative alla tabella regole. . . . .	114
Elaborazione della tabella regole . . . . .	115
Metodi per l'elaborazione di tutti i messaggi presenti nella coda DLQ. . . . .	116
Esempio di tabella regole per DLQ handler . . . . .	118

## Capitolo 9. Eventi di strumentazione 121

Cosa sono gli eventi di strumentazione? . . . . .	121
Perché utilizzare gli eventi? . . . . .	122
Tipi di eventi . . . . .	123
Notificazione di evento attraverso code di evento. . . . .	124
Abilitazione e disabilitazione degli eventi . . . . .	124
Messaggi di evento . . . . .	125

## Capitolo 10. Supporto transazionale 127

Coordinazione dei database . . . . .	128
Limitazioni . . . . .	129
Connessioni al database . . . . .	129
Configurazione dei gestori di database . . . . .	129
Configurazione Oracle . . . . .	131
Livelli minimi di supporto per Oracle . . . . .	131
Controllare le impostazioni della variabile di ambiente.. . . . .	131
Abilitazione del supporto XA Oracle . . . . .	131
Creazione del file switch load Oracle . . . . .	132
Aggiunta dell'informazione di configurazione Oracle XAResourceManager . . . . .	134
Modifica dei parametri di configurazione Oracle . . . . .	136
Attività di gestione . . . . .	136
Unità di elaborazione sospese . . . . .	137
Utilizzo del comando dspmqtrn . . . . .	137
Utilizzo del comando rsvmqtrn . . . . .	139
Risultati diversi ed errori . . . . .	139
Modifica delle informazioni di configurazione . . . . .	140

## Capitolo 11. Ripristino e riavvio . . . . . 143

Come assicurarsi che i messaggi non vadano persi (registrazione nei log) . . . . .	143
Che aspetto hanno i log . . . . .	144
Tipi di registrazione nei log . . . . .	144
Checkpoint – garantire un completo ripristino . . . . .	146
Calcolo delle dimensioni del log . . . . .	149
Gestione di log . . . . .	150
Cosa accade quando un disco è pieno . . . . .	151
Gestione di file di log . . . . .	151
Utilizzo del log per il ripristino . . . . .	152
Risoluzione dei problemi . . . . .	152
Ripristino dei supporti . . . . .	153
Ripristino degli oggetti danneggiati durante l'avvio. . . . .	154
Ripristino degli oggetti danneggiati in altri momenti . . . . .	155
Protezione dei file di log di MQSeries . . . . .	155
Backup e ripristino . . . . .	155
Esecuzione del backup di MQSeries . . . . .	155
Ripristino di MQSeries . . . . .	156
Situazioni di ripristino . . . . .	156
Malfunzionamenti dell'unità disco . . . . .	157
Oggetto danneggiato del Queue Manager . . . . .	158
Oggetto singolo danneggiato . . . . .	158
Malfunzionamento del ripristino automatico dei supporti . . . . .	158
Scaricamento del contenuto del log mediante il comando dmpmqlog . . . . .	158

## Capitolo 12. Utilizzo di Name Service 177

Utilizzo di DCE per condividere code relative a Queue Manager diversi . . . . .	177
Operazioni di configurazione per code condivise . . . . .	177
Configurazione DCE . . . . .	178

## Capitolo 13. Configurazione MQSeries 179

File di configurazione di MQSeries . . . . .	179
Modifica dei file di configurazione . . . . .	179
File di configurazione MQSeries, mqs.ini . . . . .	180
File di configurazione Queue Manager, qm.ini . . . . .	181
Attributi per modificare informazioni sulle configurazioni MQSeries . . . . .	181
La voce AllQueueManagers . . . . .	181
La voce ClientExitPath . . . . .	182
La voce DefaultQueueManager . . . . .	183
La voce ExitProperties . . . . .	183
La voce LogDefaults . . . . .	184
La voce QueueManager . . . . .	186
Modifica delle informazioni sulle configurazioni . . . . .	186
La voce Service. . . . .	186
La voce ServiceComponent. . . . .	187
La voce Log . . . . .	188
La voce XAResourceManager . . . . .	190
La voce Channels . . . . .	191
Le voci LU62 e TCP . . . . .	193
La voce ExitPath . . . . .	194
Esempio di file mqs.ini e qm.ini . . . . .	194

## Capitolo 14. Determinazione dei problemi . . . . . 199

Controlli preliminari . . . . .	199
Si è mai avuta in precedenza un'esecuzione con esito positivo di MQSeries?. . . . .	199
Vengono riportati messaggi di errore? . . . . .	199
Vengono riportati codici di errori che spiegano il problema? . . . . .	200
Il problema è riproducibile? . . . . .	200
Sono state effettuate modifiche dall'ultima esecuzione valida?. . . . .	200
Si è mai avuta in precedenza un'esecuzione con esito positivo dell'applicazione? . . . . .	200
Il problema è relativo a parti specifiche della rete? . . . . .	201
Il problema si verifica ad una determinata ora del giorno? . . . . .	201
Il problema si manifesta in modo discontinuo? . . . . .	202
Sono stati eseguiti aggiornamenti del servizio? . . . . .	202
È necessario aggiungere un aggiornamento? . . . . .	202
Errori di programmazione frequenti . . . . .	202
Come procedere . . . . .	203
Viene riportato un output non corretto? . . . . .	203
Non si è ottenuta alcuna risposta utilizzando un comando PCF? . . . . .	203
Si verificano malfunzionamenti in alcune code? . . . . .	204
Il problema è relativo alle sole code remote? . . . . .	205
Considerazioni sulla progettazione delle applicazioni . . . . .	205
Lunghezza del messaggio . . . . .	206
Messaggi permanenti. . . . .	206

Ricerca di un particolare messaggio . . . . .	206
Code contenenti messaggi di diversa lunghezza	206
Frequenza di syncpoint . . . . .	206
Utilizzo della chiamata MQPUT1. . . . .	207
Output non corretto . . . . .	207
Messaggi assenti nella coda . . . . .	207
Messaggi che contengono informazioni inattese o alterate . . . . .	208
Problemi relativi ad output non corretti durante l'accodamento distribuito . . . . .	209
Log di errore . . . . .	209
File di log . . . . .	210
Errori precedenti . . . . .	210
Messaggi per l'operatore. . . . .	211
Esempio di log degli errori . . . . .	211
Code messaggi non recapitati . . . . .	213
File di configurazione e determinazione dei problemi . . . . .	214
Utilizzo della traccia MQSeries . . . . .	214
Nomi dei file di traccia . . . . .	214
Esempio di dati traccia . . . . .	215
FFST (First Failure Support Technology) . . . . .	215
Esame dei FFST . . . . .	215
Determinazione dei problemi relativi ai client . . . . .	220
Client arrestati . . . . .	221
Messaggi di errore relativi ai client . . . . .	221

## Capitolo 15. Ottimizzazione delle prestazioni . . . . . 223

Impostazione del valore dei parametri specifici di processo . . . . .	224
---	-----

## Capitolo 16. MQSeries per OpenVMS e clustering . . . . . 227

Installazione di MQSeries in un cluster OpenVMS	227
Gruppi di ripristino di cluster OpenVMS . . . . .	228
Panoramica dei gruppi di ripristino di cluster OpenVMS . . . . .	228
Nozione relative ai gruppi di ripristino di cluster OpenVMS . . . . .	229
Preparazione alla configurazione di un gruppo di ripristino cluster OpenVMS. . . . .	230
Configurazione di un gruppo di ripristino cluster OpenVMS . . . . .	230
Operazione successive alla configurazione di un gruppo di ripristino cluster OpenVMS . . . . .	232
Modifica del file di configurazione FAILOVER.INI . . . . .	232
Procedure di comando utilizzate dai gruppi di ripristino . . . . .	233
Gestione dei gruppi di ripristino . . . . .	235
Avvio di failover monitor . . . . .	235
Avvio di un Queue Manager in un gruppo di ripristino . . . . .	235
Arresto di un Queue Manager in un gruppo di ripristino . . . . .	236
Spostamento di un Queue Manager in un gruppo di ripristino . . . . .	236
Visualizzazione dello stato di un gruppo di ripristino . . . . .	236

Impostazione dei simboli DCL allo stato di un gruppo di ripristino . . . . .	238
Arresto di un processo failover monitor . . . . .	239
Esecuzione di comandi mentre è in corso un aggiornamento . . . . .	239
Modifica dello stato di un gruppo di ripristino	239
Impostazione di sicurezza per associazioni ICC	240
Risoluzione dei problemi con gruppi di ripristino . . . . .	241
Utilizzo di MultiNet per OpenVMS con gruppi di ripristino . . . . .	241
Un esempio di utilizzo di gruppi di ripristino	242

## Parte 2. Riferimenti . . . . . 249

### Capitolo 17. Comandi di controllo di MQSeries . . . . . 251

Regole per la denominazione degli oggetti di MQSeries. . . . .	251
Analisi dei file di oggetto . . . . .	251
Come leggere i diagrammi di sintassi . . . . .	252
Guida alla sintassi. . . . .	253
Esempi . . . . .	254
Codici di errore di MQSeries . . . . .	255
crtmqcvx (Conversione dei dati) . . . . .	256
crtmqm (Crea Queue Manager) . . . . .	258
dlmqm (Delete queue manager) . . . . .	263
dmpmqlog (Dump log) . . . . .	266
dspmqaut (Display Authority) . . . . .	268
dspmqcsv (Display Command Server) . . . . .	272
dspmqfls (Display MQSeries files) . . . . .	273
dspmqtrc (Visualizza output di traccia formattato di MQSeries) . . . . .	275
dspmqtrn (Display MQSeries transactions) . . . . .	277
endmqcsv (End command server) . . . . .	279
endmqlsr (End listener) . . . . .	282
endmqm (End queue manager) . . . . .	283
endmqtrc (End MQSeries trace) . . . . .	286
failover (Gestione di un gruppo di ripristino). . . . .	287
rcdmqimg (Registra immagine supporto) . . . . .	291
rcrmqobj (Ricrea oggetto) . . . . .	293
rsvmqtrn (risolve transazioni MQSeries) . . . . .	296
runmqchi (Run channel initiator) . . . . .	298
runmqchl (Run channel) . . . . .	299
runmqdlq (Run dead-letter queue handler) . . . . .	300
runmqfm (Avvia un monitor di ripristino). . . . .	302
runmqlsr (Run listener) . . . . .	303
runmqsc (Esegue comandi di MQSeries) . . . . .	305
runmqtrc (Avvia trigger monitor del client) . . . . .	308
runmqtrm (Avvia un trigger monitor) . . . . .	309
setmqaut (Imposta/reimposta l'autorizzazione) . . . . .	310
strmqcsv (Avvia server di comandi) . . . . .	317
strmqm (Avvio Queue Manager) . . . . .	318
strmqtrc (Avvio traccia di MQSeries) . . . . .	320

## Parte 3. Appendici . . . . . 323

### Appendice A. MQSeries per Compaq OpenVMS al primo sguardo . . . . . 325



Numero programma e parte . . . . .	325
Requisiti hardware . . . . .	325
Requisiti software . . . . .	325
Connettività . . . . .	325
Sicurezza . . . . .	325
Funzioni di manutenzione . . . . .	326
Compatibilità . . . . .	326
Compilatori supportati . . . . .	326
Selezione della lingua . . . . .	326
Internazionalizzazione . . . . .	326

**Appendice B. Impostazioni predefinite del sistema . . . . . 327**

**Appendice C. Struttura di directory 329**

Directory e file in MQS_ROOT:[MQM] . . . . .	330
Directory e file della sottodirectory MQS_ROOT:[MQM.QMGRS.QMNAME] . . . . .	330

**Appendice D. Confronto tra gruppi di comandi . . . . . 335**

Comandi per la gestione di Queue Manager . . . . .	335
Comandi per la gestione di server di comandi . . . . .	335
Comandi per la gestione di code . . . . .	336
Comandi per la gestione di processi . . . . .	336
Comandi per la gestione di canali . . . . .	337
Altri comandi di controllo . . . . .	337

**Appendice E. Esempi di programmi MQI e file MQSC . . . . . 339**

Esempi di file di comando MQSC . . . . .	339
Esempi di programmi in C e COBOL . . . . .	339
Strumenti vari . . . . .	340

**Appendice F. OpenVMS modelli di gruppi di ripristino di cluster. . . . . 341**

Modello di file di configurazione FAILOVER.TEMPLATE . . . . .	341
Modello di procedura StartCommand START_QM.TEMPLATE . . . . .	343
Modello di procedura EndCommand END_QM.TEMPLATE . . . . .	344
Modello di procedura TidyCommand TIDY_QM.TEMPLATE . . . . .	347

**Appendice G. Codici supportati da MQSeries per Compaq OpenVMS. . . . . 349**

**Appendice H. Utilità di diagnostica MONMQ. . . . . 351**

Panoramica . . . . .	351
Variabili in MONMQ . . . . .	352
Assegnazione dei valori predefiniti . . . . .	353
Aprire o creare una sezione di traccia e la casella postale associata . . . . .	354
Visualizzazione della definizione di unità logica . . . . .	354
Chiusura ed eliminazione di una LU . . . . .	355
Visualizzazione dei dettagli di canale . . . . .	355

Visualizzazione della maschera di traccia corrente per un canale . . . . .	355
Visualizzazione dei contenuti dello stack dei sottoprocessi di destinazione . . . . .	356
Visualizzazione dei processi attivi relativi a MQSeries e dell'utilizzo della memoria . . . . .	356
Visualizzazione di tutti i messaggi conservati in un canale . . . . .	356
Visualizzazione di tutte le sezioni globali relative a MQSeries sul nodo corrente . . . . .	357
Segnalazioni al sottoprocesso di destinazione per l'invio della tabella mutex al processo di traccia del client . . . . .	358
Segnalazioni al sottoprocesso di destinazione per l'invio della tabella degli eventi interni al processo di traccia del client . . . . .	359
Segnalazioni al sottoprocesso di destinazione per l'invio della tabella interna della memoria condivisa mappata al processo di traccia del client . . . . .	360
Visualizzazione dei componenti attivi MQSeries per nome e ID esadecimali . . . . .	361
Visualizzazione delle funzioni in un componente specificato . . . . .	361
Attivazione della traccia dal punto di avvio di un processo . . . . .	362
Impedire al processo di MQSeries di eseguire la traccia immediatamente all'avvio . . . . .	362
Connessione del sottoprocesso di destinazione al canale specificato . . . . .	362
Disconnessione del sottoprocesso di destinazione dal canale specificato . . . . .	363
Visualizzazione del messaggio di traccia in tempo reale scritto alla casella di posta della traccia delle LU . . . . .	363
Separazione e termine del processo del client corrente . . . . .	363
Specificazione dei dati di traccia . . . . .	363
Rimozione di una singola voce dalla tabella filtri della traccia . . . . .	364
Scrittura di messaggi di traccia in un file binario da parte di un processo di client . . . . .	364
Chiusura di un file binario di messaggi di traccia . . . . .	365
Scrittura di messaggi di traccia in un file di testo da parte di un processo di client . . . . .	365
Chiusura di file di testo di messaggi di traccia . . . . .	365
Messaggi di timestamp . . . . .	365
Arresto dei messaggi di timestamping . . . . .	365
Attivazione traccia . . . . .	365
Disattivazione traccia . . . . .	366
Salvataggio cronologia messaggi . . . . .	366
Disabilitazione della cronologia messaggi . . . . .	366
Eliminazione della cronologia messaggi . . . . .	366
Impostazione della profondità di cronologia . . . . .	366
Azzeramento dello stack e dei dati di cronologia per un canale . . . . .	366
Abilitazione o disabilitazione del bit di maschera . . . . .	367
Impostazione di un colore per un canale . . . . .	368
Reindirizzare l'output a un file . . . . .	368
Analisi del file binario di traccia . . . . .	368
Visualizzazione dello stato corrente dei sottoprocessi di MQSeries . . . . .	371

Chiusura di traccia e uscita da MONMQ . . . . .	371
Abbandono di MONMQ senza chiudere la traccia . . . . .	371
Gestione della memoria condivisa con MONMQ . . . . .	372
Script e macro in MONMQ. . . . .	373
Esempio di sessione di traccia . . . . .	374

**Appendice I. Uscite utente . . . . . 385**

Uscite del canale e del carico di lavoro . . . . .	385
MQSeries Cluster Workload Exit . . . . .	385

**Appendice J. Applicazioni affidabili 387**

Applicazioni per l'utente . . . . .	387
Impostazione delle applicazioni affidabili . . . . .	387
Esecuzione di canali e listener come applicazioni affidabili . . . . .	388
Messaggi non permanenti rapidi . . . . .	388

**Appendice K. Informazioni ausiliarie 389**

Guida di programmazione dell'applicazione . . . . .	389
Triggering dell'applicazione . . . . .	389

**Appendice L. Informazioni particolari 391**

Marchi . . . . .	392
------------------	-----

**Bibliografia . . . . . 395**

MQSeries pubblicazioni multiplatforma . . . . .	395
Pubblicazioni MQSeries per piattaforme specifiche . . . . .	395
Manuali in formato elettronico . . . . .	396
Formato HTML . . . . .	396
Portable Document Format (PDF) . . . . .	396
Formato BookManager . . . . .	397
Formato PostScript . . . . .	397
Formato Windows Help . . . . .	397
Informazioni su MQSeries disponibili su Internet . . . . .	397
Pubblicazioni correlate . . . . .	397

**Glossario dei termini e delle abbreviazioni . . . . . 399**

**Indice analitico. . . . . 411**

---

## Figure

1.	Code, messaggi e applicazioni . . . . .	37
2.	Tipico output di un comando DISPLAY QMGR	41
3.	Estratto dal file di comandi MQSC, myprog.in	43
4.	Estratto dal file di notifica MQSC, myprog.out.	44
5.	Risultati tipici di un browser di coda . . . . .	53
6.	Gestione remota . . . . .	72
7.	Impostazione di canali e code per la gestione remota . . . . .	73
8.	Esempio di regola da una tabella regole DLQ handler . . . . .	111
9.	Analisi degli eventi di strumentazione	122
10.	Controllo dei Queue Manager attraverso diverse piattaforme, da un singolo nodo. . .	123
11.	Codice sorgente per file switch load Oracle, oraswit.c . . . . .	132
12.	Esempio di immissione XAResourceManager per Oracle . . . . .	136
13.	Esempio di output dspmqtrn . . . . .	138
14.	Esempio di output dspmqtrn per una transazione errata . . . . .	140
15.	Esclusione della voce XAResourceManager	141
16.	Checkpoint . . . . .	147
17.	Checkpoint con transazioni in esecuzione da molto tempo . . . . .	148
18.	Output di dmpmqlog di esempio . . . . .	163
19.	Esempio di un file di configurazione MQSeries per sistemi MQSeries per Compaq OpenVMS . . . . .	195
20.	Esempio di file di configurazione di Queue Manager per MQSeries per Compaq OpenVMS . . . . .	196
21.	Esempio di traccia MQSeries per Compaq OpenVMS . . . . .	215
22.	Esempio di voci da immettere in \$SYSTARTUP.COM . . . . .	241
23.	Failover.template per creare un file di configurazione FAILOVER.INI . . . . .	243
24.	procedura di comando start_failover_set	245
25.	procedura di comando end_failover_set	247
26.	Struttura di directory predefinita dopo l'avvio di un Queue Manager . . . . .	329
27.	File di configurazione modello: failover.template . . . . .	342
28.	Modello di procedura StartCommand: Start_QM.template . . . . .	343
29.	Modello di procedura EndCommand: END_QM.template. . . . .	346
30.	Modello di procedura TidyCommand: TIDY_QM.template . . . . .	347



---

## Tabelle

1. Categorie dei comandi di controllo . . . . .	27	15. Come leggere i diagrammi di sintassi	252
2. Autorizzazione di sicurezza necessaria per chiamate MQI. . . . .	97	16. Autorizzazioni di sicurezza del comando dspmqaut. . . . .	269
3. Comandi MQSC e autorizzazioni di sicurezza necessari . . . . .	100	17. Specificazione delle autorizzazioni per i diversi tipi di oggetto. . . . .	313
4. Comandi PCF e autorizzazioni di sicurezza necessari . . . . .	101	18. Oggetti del sistema e oggetti predefiniti per le code . . . . .	327
5. Database relazionali XA compatibili . . . . .	128	19. Oggetti del sistema e oggetti predefiniti per i canali . . . . .	327
6. Dimensioni del sovraccarico del log (tutti i valori sono approssimativi) . . . . .	149	20. Oggetti del sistema e oggetti predefiniti per namelist . . . . .	328
7. Elenco di possibili ISO CCSIDs . . . . .	182	21. Oggetti del sistema e oggetti predefiniti per processi. . . . .	328
8. Richieste predefinite di connessione esterna (TCP) . . . . .	193	22. Comandi per la gestione di Queue Manager	335
9. Descrizione dei campi nel file FAILOVER.INI	232	23. Comandi per la gestione di server di comandi	335
10. Parametri elaborati dalle procedure di comando . . . . .	233	24. Comandi per la gestione di code . . . . .	336
11. Stato dei Queue Manager del gruppo di ripristino . . . . .	236	25. Comandi per la gestione di processi . . . . .	336
12. Stato dei Queue Manager su ciascun nodo del gruppo di ripristino . . . . .	237	26. Comandi per la gestione di canali. . . . .	337
13. Stato dei Node Monitor del gruppo di ripristino . . . . .	237	27. Altri comandi di controllo . . . . .	337
14. simboli DCL e relativa descrizione . . . . .	238	28. File di comando MQSC . . . . .	339
		29. Programmi di esempio - file di origine	339
		30. File vari . . . . .	340
		31. Locali e CCSID . . . . .	349



---

## Informazioni su questo manuale

MQSeries per Compaq OpenVMS Alpha, V5.1 –in questo volume denominato anche MQSeries per Compaq OpenVMS o MQSeries, a seconda del contesto–fa parte della famiglia di prodotti MQSeries. Questi prodotti forniscono servizi di programmazione di applicazioni che consentono ai programmi applicativi di comunicare fra essi mediante *code di messaggi*. Questa forma di comunicazione è denominata *messaggistica commerciale*. Le applicazioni implicate possono esistere su nodi diversi in una vasta gamma di tipi di computer e di sistemi operativi. Queste utilizzano un'interfaccia comune di programmazione di applicazioni, denominata Message Queuing Interface o MQI, in modo che i programmi sviluppati su una determinata piattaforma possano essere prontamente trasferiti a un'altra piattaforma.

Questo volume descrive gli aspetti di gestione del sistema MQSeries per Compaq OpenVMS Alpha, V5.1 e i servizi forniti a supporto della messaggistica commerciale in un ambiente OpenVMS. Ciò include la gestione delle code che le applicazioni utilizzano per ricevere messaggi e la garanzia che le applicazioni dispongano dell'accesso alle code necessarie.

---

## A chi si rivolge questo manuale

In primo luogo, questo manuale è rivolto agli amministratori e ai programmatori di sistema, che gestiscono le attività di configurazione e gestione di MQSeries. Inoltre, il volume è utile ai programmatori di applicazioni, i quali necessitano di alcune cognizioni sulle attività amministrative di MQSeries.

---

## Cosa bisogna conoscere per comprendere questo manuale

Per adoperare questo manuale occorre disporre di una buona competenza del sistema operativo OpenVMS e delle utilità associate. Non è necessario aver già lavorato con prodotti di accodamento messaggi, ma occorre avere alcune cognizioni dei concetti fondamentali dell'accodamento messaggi.

---

## Come utilizzare questo manuale

La parte centrale di questo volume:

- Presenta MQSeries
- Descrive la gestione quotidiana di un sistema OpenVMS, affrontando argomenti quali la gestione di oggetti locali e remoti di MQSeries, la sicurezza, il supporto per le transazioni e la determinazione di problemi.

## Utilizzo delle appendici

Le appendici forniscono materiale di riferimento. Alcune comprendono informazioni che saranno integrate in altri manuali di MQSeries alla prossima occasione.

## Informazioni su MQSeries su Internet

**URL di MQSeries**

L'URL della homepage della famiglia di prodotti MQSeries è:

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



---

## Novità presenti in MQSeries per Compaq OpenVMS Alpha, V5.1

La seguente nuova funzione è descritta in questa edizione del *MQSeries per Compaq OpenVMS Alpha, V5.1 Guida alla gestione del sistema*.

### Cluster del Queue Manager di MQSeries

È possibile connettere i Queue Manager di MQSeries per formare un cluster di Queue Manager. All'interno di un cluster, i Queue Manager possono rendere le code ospitate disponibili a tutti gli altri Queue Manager. Qualsiasi Queue Manager può inviare un messaggio a tutti gli altri Queue Manager nello stesso cluster senza la necessità di definizioni di canale esplicite, definizioni di coda remota o code di trasmissione per ciascuna destinazione. I principali vantaggi dei cluster di MQSeries sono:

- Minor numero di attività di gestione del sistema
- Disponibilità accresciuta
- Bilanciamento del carico di lavoro

Consultare "Cluster" a pagina 14 per ulteriori informazioni.

**Nota:** I cluster di MQSeries non sono uguali ai cluster di OpenVMS.

Quando si utilizza il termine *cluster*, questo si riferisce a un cluster del Queue Manager di MQSeries. Ci si riferisce a un cluster OpenVMS sempre come a un *cluster OpenVMS*. Per ulteriori informazioni sui cluster OpenVMS, consultare il "Capitolo 16. MQSeries per OpenVMS e clustering" a pagina 227.

### MQSeries Application Interface (MQAI)

MQSeries per Compaq OpenVMS ora supporta MQSeries Application Interface (MQAI), un'interfaccia di programmazione che semplifica l'utilizzo dei messaggi PCF per configurare MQSeries. Per ulteriori informazioni su MQAI, incluse descrizioni sui comandi completi, consultare il manuale *MQSeries Administration Interface Programming Guide and Reference*.

### Dimensione coda messaggi

Una coda messaggi può avere dimensioni fino a 2 GB.

### Controllato, attendere arresto di un Queue Manager

È stata aggiunta una nuova opzione al comando **endmqm** per consentire l'arresto sincrono controllato di un Queue Manager.

### Supporto Java<sup>®</sup>

MQSeries per Compaq OpenVMS adesso funziona con i compilatori Java.

### Gestione Web

È ora possibile eseguire in remoto le seguenti attività di gestione per MQSeries per Compaq OpenVMS mediante un browser HTML (ad esempio, Netscape Navigator o Microsoft<sup>®</sup> Internet Explorer) su un sistema Windows NT<sup>®</sup>:

- Connettersi come Amministratore di MQSeries
- Selezionare un Queue Manager ed inviare ad esso comandi MQSC
- Creare, modificare ed eliminare script MQSC.



# Parte 1. Guida

<b>Capitolo 1. Introduzione a MQSeries.</b> . . . . .	5
MQSeries e accodamento di messaggi . . . . .	5
Applicazioni indipendenti dal tempo . . . . .	5
Elaborazione dei messaggi guidati . . . . .	5
Messaggi e code . . . . .	5
Che cos'è un messaggio? . . . . .	6
Dimensione dei messaggi . . . . .	6
Cos'è una coda? . . . . .	6
In che modo le applicazioni inviano e ricevono i messaggi? . . . . .	6
Code predefinite e dinamiche. . . . .	7
Richiamo dei messaggi dalle code . . . . .	7
Oggetti . . . . .	7
Nomi degli oggetti . . . . .	8
Gestione degli oggetti . . . . .	8
Attributi degli oggetti . . . . .	8
Queue Manager MQSeries. . . . .	9
Chiamata MQI. . . . .	9
Code di MQSeries . . . . .	9
Utilizzo di oggetti coda . . . . .	10
Code locali specifiche utilizzate da MQSeries . . . . .	11
Definizioni di processo . . . . .	13
Canali . . . . .	13
Cluster . . . . .	14
Namelist . . . . .	14
Oggetti definiti dal sistema . . . . .	14
Gestione locale e remota . . . . .	15
Client e server . . . . .	15
Applicazioni MQSeries in un ambiente client-server . . . . .	15
Estensione delle funzioni Queue Manager . . . . .	16
Uscite utente . . . . .	17
Servizi installabili . . . . .	17
Sicurezza . . . . .	18
Funzione OAM (Object Authority Manager) . . . . .	18
Sicurezza DCE . . . . .	18
Supporti transazionali . . . . .	18
<b>Capitolo 2. Introduzione alla gestione di MQSeries.</b> . . . . .	19
Gestione locale e in remoto . . . . .	19
Esecuzione delle attività di gestione utilizzando i comandi di controllo . . . . .	19
Esecuzione delle attività di gestione utilizzando i comandi MQSC . . . . .	20
Esecuzione delle attività di gestione utilizzando i comandi PCF. . . . .	20
Attributi MQSC e PCF. . . . .	21
Escape PCF . . . . .	21
Analisi dei nomi di file MQSeries . . . . .	21
Trasformazione del nome di Queue Manager . . . . .	21
Trasformazione del nome di oggetto . . . . .	23
Analisi della sensibilità al minuscolo/maiuscolo . . . . .	23
Sensibilità al minuscolo/maiuscolo nei comandi di controllo . . . . .	23
Sensibilità al minuscolo/maiuscolo nei comandi MQSC . . . . .	25
<b>Capitolo 3. Gestione dei Queue Manager utilizzando i comandi di controllo</b> . . . . .	27
Utilizzo dei comandi di controllo . . . . .	27
Utilizzo dei comandi di controllo . . . . .	27
Creazione di un Queue Manager . . . . .	28
Indicazioni per la creazione dei Queue Manager . . . . .	28
Specificare un nome univoco per il Queue Manager . . . . .	29
Limitare il numero dei Queue Manager . . . . .	29
Specificare il Queue Manager predefinito . . . . .	29
Specificare una coda messaggi non recapitati . . . . .	30
Specificare una coda di trasmissione predefinita. . . . .	31
Specificare i parametri necessari alla registrazione nei log . . . . .	31
Eseguire una copia di riserva dei file di configurazione dopo aver creato un Queue Manager . . . . .	31
Creazione di un Queue Manager predefinito . . . . .	31
Avvio di un Queue Manager . . . . .	32
Rendere predefinito un Queue Manager esistente. . . . .	32
Arresto di un Queue Manager . . . . .	33
Chiusura ad attività completate. . . . .	33
Chiusura immediata . . . . .	33
Chiusura preventiva . . . . .	34
Problemi di chiusura del Queue Manager . . . . .	34
Riavvio di un Queue Manager . . . . .	34
Per eliminare un Queue Manager . . . . .	34
<b>Capitolo 4. Gestione di oggetti MQSeries locali</b> . . . . .	37
Supporto dei programmi applicativi che utilizzano MQI. . . . .	37
Esecuzione delle attività di gestione locale utilizzando i comandi MQSC . . . . .	38
Prima di iniziare. . . . .	38
Nomi degli oggetti MQSeries . . . . .	38
Reindirizzo di input e output . . . . .	39
Utilizzo interattivo della funzione MQSC . . . . .	39
Feedback dei comandi MQSC . . . . .	40
Termine dell'input interattivo MQSC . . . . .	40
Visualizzazione degli attributi Queue Manager . . . . .	40
Utilizzo di un Queue Manager diverso dal predefinito. . . . .	41
Modifica degli attributi di Queue Manager . . . . .	42
Esecuzione di comandi MQSC da file di testo . . . . .	42
File di comandi MQSC . . . . .	43
Notifiche MQSC. . . . .	43
Esecuzione dei file di comandi MQSC forniti . . . . .	44
Utilizzo di runmqsc per la verifica dei comandi . . . . .	44
Risoluzione dei problemi con MQSC . . . . .	45
Utilizzo delle code locali . . . . .	47
Definizione di una coda locale . . . . .	47
Definizione di una coda messaggi non recapitati . . . . .	48

Visualizzazione degli attributi predefiniti di oggetti . . . . .	49	Utilizzo di definizioni di coda remota come alias . . . . .	80
Copia della definizione di una coda locale . . . . .	49	Alias di Queue Manager . . . . .	80
Modifica degli attributi di coda locale . . . . .	50	Alias di coda di risposta RTQ . . . . .	81
Annullamento di una coda locale . . . . .	50	Conversione dei dati . . . . .	81
Eliminazione di una coda locale . . . . .	52	Quando un Queue Manager non è in grado di convertire messaggi in formati incorporati . . . . .	81
Esame delle code . . . . .	52	File ccsid.tbl . . . . .	81
Utilizzo di alias di code . . . . .	56	Conversione di messaggi in formati definiti dall'utente . . . . .	83
Definizione di un alias di coda . . . . .	56	Modifica del CCSID del Queue Manager . . . . .	83
Utilizzo di altri comandi con gli alias di coda . . . . .	57		
Utilizzo delle code di esempio . . . . .	57	<b>Capitolo 7. Protezione di oggetti di MQSeries . . . . .</b>	<b>85</b>
Definizione di una coda di esempio . . . . .	58	Perché bisogna proteggere le risorse di MQSeries. . . . .	85
Utilizzo degli altri comandi con le code di esempio . . . . .	58	Prima di iniziare. . . . .	85
Gestione degli oggetti per triggering . . . . .	59	ID utente in MQSeries per Compaq OpenVMS con MQM identificativo risorse. . . . .	85
Definizione di una coda di applicazione per triggering . . . . .	59	Per ulteriori informazioni. . . . .	86
Definizione di una coda di iniziazione . . . . .	60	Comprendere Object Authority Manager . . . . .	86
Creazione di una definizione di processo . . . . .	60	Come funziona OAM . . . . .	87
Visualizzazione della definizione di processo . . . . .	61	Gestione dell'accesso mediante gli identificativi di diritti . . . . .	87
		Gli identificativi di diritti e l'identificativo di diritti primario . . . . .	87
<b>Capitolo 5. Automazione delle attività di gestione 63</b>		Quando un principal contiene più di un identificativo di diritti . . . . .	87
Comandi PCF . . . . .	63	Identificativo diritti predefinito . . . . .	88
Attributi MQSC e PCF. . . . .	64	Le risorse che è possibile proteggere con OAM . . . . .	88
Escape PCF . . . . .	64	Utilizzo degli identificativi diritti per le autorizzazioni . . . . .	88
Utilizzo di MQAI per semplificare l'impiego di PCF . . . . .	64	Disabilitazione di OAM (Object Authority Manager) . . . . .	89
Gestione del server di comando per una gestione remota . . . . .	65	Utilizzo dei comandi di OAM (Object Authority Manager) . . . . .	89
Avvio del server di comando . . . . .	65	Cosa specificare durante l'utilizzo dei comandi di OAM . . . . .	89
Visualizzazione dello stato del server di comando . . . . .	65	Elenchi di autorizzazioni . . . . .	89
Arresto del server di comando . . . . .	67	Utilizzo del comando setmqaut. . . . .	91
		Comandi di autorizzazione e servizi installabili . . . . .	91
<b>Capitolo 6. Gestione di oggetti remoti di MQSeries. . . . .</b>	<b>69</b>	Autorizzazioni di accesso. . . . .	92
Canali, cluster e accodamento remoto. . . . .	69	Visualizzazione del comando di autorizzazione . . . . .	92
Gestione remota mediante cluster . . . . .	70	Istruzioni per OAM (Object Authority Manager) . . . . .	92
Gestione remota da un Queue Manager locale mediante comandi MQSC . . . . .	71	ID utente . . . . .	92
Preparazione dei Queue Manager alla gestione remota . . . . .	71	Directory del Queue Manager . . . . .	93
Preparazione di canali e code di trasmissione per la gestione remota . . . . .	72	Code . . . . .	93
Definizione di canali e code di trasmissione . . . . .	73	Autorizzazione per utente alternativo. . . . .	93
Avvio dei canali . . . . .	74	Autorizzazione contestuale . . . . .	94
Definizione automatica dei canali . . . . .	75	Considerazioni sulla sicurezza remota . . . . .	94
Emissione remota di comandi MQSC. . . . .	75	Sicurezza del comando di canale . . . . .	95
Funzionamento dei Queue Manager su MVS/ESA . . . . .	76	Comandi PCF . . . . .	95
Consigli per l'accodamento remoto . . . . .	76	Comandi canale MQSC . . . . .	95
Eventuali problemi nell'utilizzo remoto di MQSC . . . . .	77	Comandi di controllo per i canali . . . . .	95
Creazione di una definizione locale di una coda remota . . . . .	77	Comprendere le tabelle di specifica delle autorizzazioni . . . . .	96
Analisi del funzionamento delle definizioni locali di code remote . . . . .	77	Autorizzazioni MQI . . . . .	96
Esempio . . . . .	77	Autorizzazioni di gestione . . . . .	99
Un modo alternativo per inserire messaggi in una coda remota. . . . .	79	Autorizzazioni per comandi MQSC nei PCF escape. . . . .	100
Utilizzo di altri comandi con code remote . . . . .	79	Autorizzazioni per comandi PCF. . . . .	100
Creazione di una coda di trasmissione . . . . .	79	Comprendere i file di autorizzazione . . . . .	102
Code di trasmissione predefinite . . . . .	80	Percorsi dei file di autorizzazione . . . . .	102

Directory dell'autorizzazione . . . . .	102	Come assicurarsi che i messaggi non vadano persi (registrazione nei log) . . . . .	143
Cosa contengono i file di autorizzazione . . . . .	103	Che aspetto hanno i log . . . . .	144
File di autorizzazione classe . . . . .	104	File di controllo dei log . . . . .	144
Tutti i file di autorizzazione classe . . . . .	104	Tipi di registrazione nei log . . . . .	144
Gestione dei file di autorizzazione . . . . .	105	Registrazione nei log circolare . . . . .	144
Autorizzazioni per i file di autorizzazione . . . . .	105	Registrazione nei log lineare . . . . .	145
<b>Capitolo 8. Handler di code messaggi non recapitati di MQSeries . . . . .</b>	<b>107</b>	Checkpoint – garantire un completo ripristino . . . . .	146
Richiamo del DLQ handler . . . . .	107	Calcolo delle dimensioni del log . . . . .	149
Modello di DLQ handler, amqsdlq . . . . .	108	Gestione di log . . . . .	150
Tabella regole DLQ handler . . . . .	108	Cosa accade quando un disco è pieno . . . . .	151
Dati di controllo . . . . .	109	Gestione di file di log . . . . .	151
Regole (schemi e azioni) . . . . .	111	Posizione dei file di log . . . . .	152
Le parole chiave dello schema di confronto . . . . .	111	Utilizzo del log per il ripristino . . . . .	152
Le parole chiave di azione . . . . .	112	Risoluzione dei problemi . . . . .	152
Convenzioni relative alla tabella regole . . . . .	114	Ripristino dei supporti . . . . .	153
Elaborazione della tabella regole . . . . .	115	Ripristino di immagini del supporto . . . . .	154
Metodi per l'elaborazione di tutti i messaggi presenti nella coda DLQ . . . . .	116	Ripristino degli oggetti danneggiati durante l'avvio . . . . .	154
Esempio di tabella regole per DLQ handler . . . . .	118	Ripristino degli oggetti danneggiati in altri momenti . . . . .	155
<b>Capitolo 9. Eventi di strumentazione . . . . .</b>	<b>121</b>	Protezione dei file di log di MQSeries . . . . .	155
Cosa sono gli eventi di strumentazione? . . . . .	121	Backup e ripristino . . . . .	155
Perché utilizzare gli eventi? . . . . .	122	Esecuzione del backup di MQSeries . . . . .	155
Tipi di eventi . . . . .	123	Ripristino di MQSeries . . . . .	156
Notificazione di evento attraverso code di evento . . . . .	124	Situazioni di ripristino . . . . .	156
Utilizzo delle code di evento con la funzione trigger . . . . .	124	Malfunzionamenti dell'unità disco . . . . .	157
Abilitazione e disabilitazione degli eventi . . . . .	124	Oggetto danneggiato del Queue Manager . . . . .	158
Messaggi di evento . . . . .	125	Oggetto singolo danneggiato . . . . .	158
<b>Capitolo 10. Supporto transazionale . . . . .</b>	<b>127</b>	Malfunzionamento del ripristino automatico dei supporti . . . . .	158
Coordinazione dei database . . . . .	128	Scaricamento del contenuto del log mediante il comando dmpmqlog . . . . .	158
Limitazioni . . . . .	129	<b>Capitolo 12. Utilizzo di Name Service . . . . .</b>	<b>177</b>
Connessioni al database . . . . .	129	Utilizzo di DCE per condividere code relative a Queue Manager diversi . . . . .	177
Configurazione dei gestori di database . . . . .	129	Operazioni di configurazione per code condivise . . . . .	177
Creazione dei switch load file . . . . .	130	Configurazione DCE . . . . .	178
Definizione dei gestori di database . . . . .	130	<b>Capitolo 13. Configurazione MQSeries . . . . .</b>	<b>179</b>
Configurazione Oracle . . . . .	131	File di configurazione di MQSeries . . . . .	179
Livelli minimi di supporto per Oracle . . . . .	131	Modifica dei file di configurazione . . . . .	179
Controllare le impostazioni della variabile di ambiente . . . . .	131	Casi in cui occorre modificare un file di configurazione . . . . .	180
Abilitazione del supporto XA Oracle . . . . .	131	Priorità del file di configurazione . . . . .	180
Creazione del file switch load Oracle . . . . .	132	Implementazione delle modifiche ai file di configurazione . . . . .	180
Creazione del file switch load Oracle nei sistemi OpenVMS . . . . .	132	File di configurazione MQSeries, mqs.ini . . . . .	180
Aggiunta dell'informazione di configurazione Oracle XAResourceManager . . . . .	134	File di configurazione Queue Manager, qm.ini . . . . .	181
Modifica dei parametri di configurazione Oracle . . . . .	136	Attributi per modificare informazioni sulle configurazioni MQSeries . . . . .	181
Attività di gestione . . . . .	136	La voce AllQueueManagers . . . . .	181
Unità di elaborazione sospese . . . . .	137	La voce ClientExitPath . . . . .	182
Utilizzo del comando dspmqtrn . . . . .	137	La voce DefaultQueueManager . . . . .	183
Utilizzo del comando rsvmqtrn . . . . .	139	La voce ExitProperties . . . . .	183
Risultati diversi ed errori . . . . .	139	La voce LogDefaults . . . . .	184
Modifica delle informazioni di configurazione . . . . .	140	La voce QueueManager . . . . .	186
Rimozione delle istanze del gestore di database . . . . .	141	Modifica delle informazioni sulle configurazioni . . . . .	186
<b>Capitolo 11. Ripristino e riavvio . . . . .</b>	<b>143</b>	La voce Service . . . . .	186
		La voce ServiceComponent . . . . .	187

La voce Log . . . . .	188
La voce XAResourceManager . . . . .	190
La voce Channels . . . . .	191
Le voci LU62 e TCP . . . . .	193
La voce ExitPath . . . . .	194
Esempio di file mqs.ini e qm.ini . . . . .	194

**Capitolo 14. Determinazione dei problemi . . . 199**

Controlli preliminari . . . . .	199
Si è mai avuta in precedenza un'esecuzione con esito positivo di MQSeries?. . . . .	199
Vengono riportati messaggi di errore? . . . . .	199
Vengono riportati codici di errori che spiegano il problema? . . . . .	200
Il problema è riproducibile? . . . . .	200
Sono state effettuate modifiche dall'ultima esecuzione valida?. . . . .	200
Si è mai avuta in precedenza un'esecuzione con esito positivo dell'applicazione? . . . . .	200
Se l'applicazione in precedenza non è stata eseguita con esito positivo . . . . .	201
Il problema è relativo a parti specifiche della rete? . . . . .	201
Il problema si verifica ad una determinata or del giorno? . . . . .	201
Il problema si manifesta in modo discontinuo? . . . . .	202
Sono stati eseguiti aggiornamenti del servizio? . . . . .	202
È necessario aggiungere un aggiornamento? . . . . .	202
Errori di programmazione frequenti . . . . .	202
Come procedere . . . . .	203
Viene riportato un output non corretto? . . . . .	203
Non si è ottenuta alcuna risposta utilizzando un comando PCF? . . . . .	203
Si verificano malfunzionamenti in alcune code? . . . . .	204
Il problema è relativo alle sole code remote? . . . . .	205
Considerazioni sulla progettazione delle applicazioni . . . . .	205
Lunghezza del messaggio . . . . .	206
Messaggi permanenti . . . . .	206
Ricerca di un particolare messaggio . . . . .	206
Code contenenti messaggi di diversa lunghezza . . . . .	206
Frequenza di syncpoint . . . . .	206
Utilizzo della chiamata MQPUT1 . . . . .	207
Output non corretto . . . . .	207
Messaggi assenti nella coda . . . . .	207
Messaggi che contengono informazioni inattese o alterate . . . . .	208
Problemi relativi ad output non corretti durante l'accodamento distribuito . . . . .	209
Log di errore . . . . .	209
File di log . . . . .	210
Errori precedenti . . . . .	210
Messaggi per l'operatore . . . . .	211
Esempio di log degli errori . . . . .	211
Code messaggi non recapitati . . . . .	213
File di configurazione e determinazione dei problemi . . . . .	214
Utilizzo della traccia MQSeries . . . . .	214
Nomi dei file di traccia . . . . .	214
Esempio di dati traccia . . . . .	215
FFST (First Failure Support Technology) . . . . .	215

Esame dei FFST . . . . .	215
Determinazione dei problemi relativi ai client . . . . .	220
Client arrestati . . . . .	221
Messaggi di errore relativi ai client . . . . .	221
OS/2, UNIX e sistemi client OpenVMS . . . . .	221
Client DOS e Windows® . . . . .	221

**Capitolo 15. Ottimizzazione delle prestazioni 223**

Impostazione del valore dei parametri specifici di processo . . . . .	224
--	-----

**Capitolo 16. MQSeries per OpenVMS e clustering . . . . . 227**

Installazione di MQSeries in un cluster OpenVMS . . . . .	227
Gruppi di ripristino di cluster OpenVMS . . . . .	228
Panoramica dei gruppi di ripristino di cluster OpenVMS . . . . .	228
Nozione relative ai gruppi di ripristino di cluster OpenVMS . . . . .	229
Preparazione alla configurazione di un gruppo di ripristino cluster OpenVMS . . . . .	230
Configurazione di un gruppo di ripristino cluster OpenVMS . . . . .	230
Operazione successive alla configurazione di un gruppo di ripristino cluster OpenVMS . . . . .	232
Modifica del file di configurazione FAILOVER.INI . . . . .	232
Procedure di comando utilizzate dai gruppi di ripristino . . . . .	233
Gestione dei gruppi di ripristino . . . . .	235
Avvio di failover monitor . . . . .	235
Avvio di un Queue Manager in un gruppo di ripristino . . . . .	235
Arresto di un Queue Manager in un gruppo di ripristino . . . . .	236
Spostamento di un Queue Manager in un gruppo di ripristino . . . . .	236
Visualizzazione dello stato di un gruppo di ripristino . . . . .	236
Impostazione dei simboli DCL allo stato di un gruppo di ripristino . . . . .	238
Arresto di un processo failover monitor . . . . .	239
Esecuzione di comandi mentre è in corso un aggiornamento . . . . .	239
Modifica dello stato di un gruppo di ripristino . . . . .	239
Impostazione di sicurezza per associazioni ICC . . . . .	240
Risoluzione dei problemi con gruppi di ripristino . . . . .	241
Utilizzo di MultiNet per OpenVMS con gruppi di ripristino . . . . .	241
Un esempio di utilizzo di gruppi di ripristino . . . . .	242
Personalizzazione di failover.template . . . . .	242
Modifica delle procedure di comando di un gruppo di ripristino . . . . .	244
Esempio di procedura di comando di avvio del gruppo di ripristino, start_failover_set.com . . . . .	244
Esempio di procedura di comando di arresto di un gruppo di ripristino, end_failover_set.com . . . . .	246



---

## Capitolo 1. Introduzione a MQSeries

Questo capitolo esamina MQSeries per Compaq OpenVMS dalla prospettiva di gestione, descrivendo i concetti di base relativi a MQSeries e alla messaggistica. Esso contiene le seguenti sezioni:

- “MQSeries e accodamento di messaggi”
- “Messaggi e code”
- “Oggetti” a pagina 7
- “Oggetti definiti dal sistema” a pagina 14
- “Gestione locale e remota” a pagina 15
- “Client e server” a pagina 15
- “Estensione delle funzioni Queue Manager” a pagina 16
- “Sicurezza” a pagina 18

---

### MQSeries e accodamento di messaggi

MQSeries consente ai programmi applicativi di utilizzare l'accodamento di messaggi per partecipare all'elaborazione di messaggi guidati. I programmi applicativi possono comunicare attraverso diverse piattaforme utilizzando appropriati prodotti software per l'accodamento dei messaggi. Ad esempio, le applicazioni OpenVMS e MVS/ESA<sup>®</sup> possono comunicare tra loro attraverso MQSeries per Compaq OpenVMS e MQSeries per OS/390<sup>®</sup>. Le applicazioni sono protette dai meccanismi delle comunicazioni di base.

I prodotti MQSeries implementano una comune interfaccia di programmazione applicativa definita MQI (Message Queue Interface) su qualunque piattaforma le applicazioni vengano eseguite. Questo facilita il trasferimento delle applicazioni da una piattaforma all'altra.

MQI è descritta in dettaglio nel manuale *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa*.

### Applicazioni indipendenti dal tempo

Grazie all'accodamento, lo scambio dei messaggi tra i programmi di invio e quelli di ricezione può avvenire in modo indipendente dal tempo. In pratica, le applicazioni di invio e di ricezione sono disaccoppiate, così che l'applicazione di invio può continuare le elaborazioni senza dover attendere la conferma di ricezione del messaggio da parte dell'applicazione ricevente. Quindi, non è necessario che l'applicazione di destinazione sia in esecuzione quando viene inviato il messaggio. Tale applicazione potrà ricevere il messaggio quando verrà avviata.

### Elaborazione dei messaggi guidati

Una volta giunti ad una coda, i messaggi possono avviare l'applicazione automaticamente utilizzando un meccanismo denominato *triggering*. Se necessario, le applicazioni possono essere arrestate una volta terminata l'elaborazione del o dei messaggi.

---

### Messaggi e code

I messaggi e le code sono gli elementi di base di un sistema di accodamento.

### Che cos'è un messaggio?

Un *messaggio* è un insieme di byte che assume significato per l'applicazione in cui viene utilizzato. I messaggi vengono utilizzati per il trasferimento di informazioni tra le applicazioni o tra diverse parti della stessa applicazione. Le applicazioni possono essere eseguite sulla stessa piattaforma o su piattaforme diverse.

I messaggi MQSeries si compongono di due parti:

- *dati per l'applicazione*

Il contenuto e la struttura dei dati è definito dal programma applicativo che li utilizza.

- *descrittore messaggio*

Il descrittore messaggio identifica il messaggio e contiene altre informazioni di controllo, quali il tipo di messaggio e la priorità assegnata al messaggio dall'applicazione che lo invia.

Il formato del descrittore messaggio è definito da MQSeries. Per un esame completo del descrittore messaggio, consultare la guida *MQSeries per Tandem Non Stop Kernal, V5.1 Guida operativa*.

### Dimensione dei messaggi

In MQSeries, la dimensione massima di un messaggio è 100 MB (dove 1 MB equivale a 1 048 576 byte). In pratica, la dimensione del messaggio può essere limitata da:

- La dimensione massima di messaggio definita per la coda ricevente.
- La dimensione massima di messaggio definita per il Queue Manager.
- La dimensione massima di messaggio definita da entrambe le applicazioni di ricezione e di invio.
- Lo spazio di memoria disponibile per il messaggio.

Per trasferire tutte le informazioni richieste da un'applicazione potrebbe essere necessario l'invio di più messaggi.

### Cos'è una coda?

Una *coda* è una struttura di dati utilizzata per memorizzare messaggi. I messaggi possono essere inviati ad una coda da programmi applicativi o da Queue Manager come una normale operazione.

Ogni coda è gestita da un *Queue Manager*. Il Queue Manager è responsabile della gestione delle code di sua pertinenza e della memorizzazione nelle code appropriate di tutti i messaggi che riceve.

La dimensione massima di una coda è 2 GB. Per informazioni relative alla pianificazione del quantitativo di memoria necessario alle code, consultare *MQSeries Planning Guide* oppure visitare il seguente sito web per ottenere le relazioni sulle caratteristiche proprie delle piattaforme.

<http://www.ibm.com/software/ts/mqseries/txppacs/txpm1.html>

### In che modo le applicazioni inviano e ricevono i messaggi?

Le applicazioni inviano e ricevono i messaggi utilizzando le *chiamate MQI*. Ad esempio, l'applicazione per inviare un messaggio ad una coda:

1. Apre la coda di destinazione inoltrando una chiamata MQI MQOPEN.
2. Inoltra una chiamata MQI MQPUT per inviare il messaggio alla coda.



3. Un'altra applicazione può richiamare il messaggio dalla stessa coda inoltrando una chiamata MQI MQGET.

Per ulteriori informazioni sulle chiamate MQI, consultare il manuale *MQSeries per Tandem Non Stop Kernal, V5.1 Guida operativa*.

### Code predefinite e dinamiche

Le code possono essere caratterizzate in base al modo in cui vengono create:

- Le *Code predefinite* sono create dall'amministratore del sistema utilizzando l'appropriato gruppo di comandi. Ad esempio, il comando DEFINE QLOCAL crea una coda locale predefinita. Le code predefinite sono permanenti; ciò significa che esistono indipendentemente dalle applicazioni che le utilizzano e sopravvivono al riavvio di MQSeries.
- Le *Code dinamiche* sono create da un'applicazione che inoltra una richiesta OPEN specificando il nome di un *modello di coda*. La coda così creata è basata su un tipo di definizione di coda, che è il modello di coda. È possibile creare un modello di coda utilizzando il comando DEFINE QMODEL. Gli attributi di un modello di coda, ad esempio il numero massimo di messaggi memorizzabili nella coda, vengono trasferiti a tutte le code dinamiche create in base a quel modello.

I modelli di coda hanno un attributo che specifica se le code dinamiche devono essere permanenti o temporanee. Le code permanenti sopravvivono all'applicazione e al riavvio; le code temporanee vanno perse quando si riavvia MQSeries.

### Richiamo dei messaggi dalle code

In MQSeries, le applicazioni opportunamente autorizzate possono richiamare messaggi da una coda sulla base di questi algoritmi di richiamo:

- FIFO (First-In-First-Out).
- Priorità di messaggio, come definito nel descrittore di messaggio. I messaggi che hanno pari priorità, sono richiamati in base al FIFO.
- La richiesta di un programma relativa ad un messaggio specifico.

La richiesta MQGET dell'applicazione determina il metodo utilizzato.

---

## Oggetti

Molte operazioni descritte in questo manuale prevedono la gestione di *oggetti* MQSeries. In MQSeries Versione 5.1, i tipi di oggetti comprendono Queue Manager, code, definizioni di processi, canali, cluster e namelist.

La manipolazione o la *gestione* di oggetti prevede:

- Avvio e arresto dei Queue Manager.
- Creazione di oggetti, in particolare code, per le applicazioni.
- Utilizzo dei canali per creare percorsi di comunicazione verso Queue Manager di altri sistemi remoti. Tutto ciò è spiegato in dettaglio nel manuale *MQSeries Intercommunication*.
- Creazione di *cluster* di Queue Manager per semplificare i processi di gestione generali o per ottenere un bilanciamento del carico di lavoro.

Questo manuale contiene informazioni dettagliate relative alla gestione nei seguenti capitoli:

- "Capitolo 2. Introduzione alla gestione di MQSeries" a pagina 19

## Oggetti

- “Capitolo 3. Gestione dei Queue Manager utilizzando i comandi di controllo” a pagina 27
- “Capitolo 4. Gestione di oggetti MQSeries locali” a pagina 37
- “Capitolo 6. Gestione di oggetti remoti di MQSeries” a pagina 69

## Nomi degli oggetti

La convenzione di denominazione adottata per gli oggetti MQSeries dipende dagli oggetti.

Ogni istanza di un Queue Manager si caratterizza per il nome. Tale nome deve essere unico all'interno della rete dei Queue Manager collegati, in modo che si possa identificare con certezza il Queue Manager di destinazione al quale inviare un dato messaggio.

Per quel che riguarda gli altri oggetti, a ognuno di essi viene associato un nome con il quale potersi riferire all'oggetto. Questo nome deve essere unico in un Queue Manager e in un tipo di oggetto. Ad esempio, è possibile che ci siano una coda e un processo con lo stesso nome, ma non è possibile che ci siano due code con lo stesso nome.

In MQSeries, i nomi possono essere composti da un massimo di 48 caratteri, ad eccezione dei *canali*, che possono essere composti da un massimo di 20 caratteri. Per ulteriori informazioni, consultare la sezione “Regole per la denominazione degli oggetti di MQSeries” a pagina 251.

## Gestione degli oggetti

È possibile creare, modificare, visualizzare ed eliminare oggetti utilizzando:

- Comandi di controllo, che vengono immessi digitandoli da una tastiera.
- Comandi MQSeries (MQSC), che possono essere immessi digitandoli da una tastiera oppure leggendoli da un file.
- Comandi PCF (Programmable Command Format), che possono essere utilizzati in un programma di automazione.
- Chiamata MQAI (MQSeries Administration Interface) in un programma.

Per ulteriori informazioni, consultare il “Capitolo 2. Introduzione alla gestione di MQSeries” a pagina 19.

### Attributi degli oggetti

Le proprietà di un oggetto sono definite dai relativi attributi. Alcuni possono essere specificati, altri solo letti. Ad esempio, la dimensione massima di messaggio che una coda può supportare è definito dall'attributo *MaxMsgLength*; è possibile specificare tale attributo al momento della creazione della coda. L'attributo *DefinitionType* specifica il modo in cui è stata creata la coda; tale attributo può essere solo visualizzato.

In MQSeries, esistono due modi per riferirsi ad un attributo:

- Utilizzando il relativo nome PCF, ad esempio, *MaxMsgLength*.
- Utilizzando il relativo nome MQSC, ad esempio, *MAXMSGL*.

Il nome formale di un attributo è il nome PCF. Dal momento che l'utilizzo della funzione MQSC costituisce una parte fondamentale di questo manuale, il nome MQSC di un dato attributo sarà più ricorrente di quello PCF.

## Queue Manager MQSeries

Un Queue Manager fornisce i servizi di accodamento alle applicazioni e gestisce le relative code. Esso garantisce che:

- Gli attributi degli oggetti vengano modificati in base ai comandi ricevuti.
- Eventi particolari, quali gli eventi trigger o gli eventi di strumentazione, vengano generati solo quando esistono le condizioni ideali.
- I messaggi vengano inviati alla coda appropriata, così come richiesto dall'applicazione che ha generato la chiamata MQPUT. L'applicazione viene informata nel caso ciò risulti impossibile e viene visualizzato un codice di errore.

Ogni coda appartiene ad un determinato Queue Manager e rispetto a questo viene definita *coda locale*. Il Queue Manager, a cui si connette un'applicazione, è definito Queue Manager locale rispetto a quella applicazione. Rispetto all'applicazione, le code che appartengono al Queue Manager locale, si definiscono code locali.

Una *coda remota* è semplicemente una coda che appartiene ad un altro Queue Manager.

Un *Queue manager remoto* è un qualunque Queue Manager diverso da quello locale. Un Queue Manager remoto può risiedere su una macchina remota attraverso la rete oppure può risiedere su una determinata macchina come Queue Manager locale.

MQSeries supporta diversi Queue Manager sulla stessa macchina.

### Chiamata MQI

Un oggetto Queue Manager può essere utilizzato in alcune chiamate MQI. Ad esempio, è possibile richiedere informazioni relative agli attributi del Queue Manager utilizzando la chiamata MQI MQINQ.

**Nota:** non è possibile inviare messaggi ad un oggetto Queue Manager; i messaggi vengono sempre inviati agli oggetti code, piuttosto che agli oggetti Queue Manager.

## Code di MQSeries

Le code vengono definite in MQSeries utilizzando:

- L'apposito comando MQSC DEFINE
- Il comando PCF Create Queue

I comandi specificano il tipo di coda e i relativi attributi. Ad esempio, un oggetto coda locale ha attributi che specificano cosa accade quando le applicazioni si riferiscono a quella coda nelle chiamate MQI. Esempi di attributi sono:

- Se le applicazioni possono ricevere messaggi dalla coda (GET enabled).
- Se l'applicazione può inviare messaggi alla coda (PUT enabled).
- Se l'accesso alla coda è esclusivo per una sola applicazione o è condiviso tra più applicazioni.
- Il numero massimo di messaggi che possono essere memorizzati in una coda nello stesso tempo (capacità massima della coda).
- La lunghezza massima dei messaggi inviabili alla coda.

Per ulteriori dettagli relativi alla definizione degli oggetti coda, consultare il manuale *MQSeries - Guida di riferimento per i comandi* oppure *MQSeries Programmable System Management*.

## Oggetti

### Utilizzo di oggetti coda

In MQSeries, vi sono diversi tipi di oggetti coda. Ogni tipo di oggetto può essere manipolato dai comandi del prodotto ed è associato alle code reali in diversi modi:

- **Oggetto coda locale**

Un oggetto coda locale identifica una coda locale che appartiene al Queue Manager al quale è connessa l'applicazione. Tutte le code sono code locali, nel senso che ogni coda appartiene ad un Queue Manager e, per quel Queue Manager, la coda è una coda locale.

- **Oggetto coda remota**

Un oggetto coda remota identifica una coda che appartiene ad un altro Queue Manager. Questa coda deve essere definita coda locale di quel Queue Manager. Le informazioni specificate durante la definizione di un oggetto coda remota, consentono a Queue Manager locale di individuare il Queue Manager remoto, in modo da destinare a quest'ultimo i messaggi da inviare alla coda remota.

Prima che le applicazioni possano inviare messaggi ad una coda che appartiene ad un altro Queue Manager, occorre aver definito una coda di trasmissione ed un canale tra i Queue Manager, a meno che non siano stati raggruppati uno o più Queue Manager in un cluster. Per ulteriori informazioni sui cluster, consultare la sezione "Gestione remota mediante cluster" a pagina 70.

- **Oggetto alias di coda**

Un oggetto alias di coda consente alle applicazioni di accedere ad una coda riferendosi ad essa indirettamente nelle chiamate MQI. Quando si utilizza un nome alias di coda in una chiamata MQI, il nome viene interpretato sia per una coda locale che remota al momento dell'avvio. Questo consente di cambiare le code utilizzate dalle applicazioni, senza modificare in alcun modo l'applicazione; basterà modificare solo la definizione dell'alias in modo da riflettere il nuovo nome della coda a cui si riferisce l'alias.

Un alias di coda non è una coda, ma un oggetto che consente di accedere ad un'altra coda.

- **Oggetto modello di coda**

Un oggetto modello di coda definisce un gruppo di attributi di coda che vengono utilizzati come esempi per la creazione di una coda dinamica. Le code dinamiche vengono create dal Queue Manager quando un'applicazione inoltra una richiesta MQOPEN specificando un nome di coda che è appunto il nome di un modello di coda. La coda dinamica così creata è una coda locale, i cui attributi sono ricavati dalla definizione del modello di coda. Il nome della coda dinamica può essere specificata dall'applicazione oppure può essere creato dal Queue Manager che lo riporta all'applicazione.

Le code dinamiche così definite possono essere code temporanee che vanno perse la riavvio del prodotto, oppure code permanenti che invece sopravvivono.

### Code locali specifiche utilizzate da MQSeries

MQSeries utilizza alcune code locali per particolari scopi relativi alle proprie operazioni. È *necessario* definirle perché MQSeries possa usarle.

**Code di applicazione:** Una coda che viene utilizzata da un'applicazione (attraverso MQI) si definisce *coda di applicazione*. Questa può essere sia una coda locale rispetto al Queue Manager a cui è connessa l'applicazione, sia una coda remota di pertinenza di un altro Queue Manager.

Le applicazioni possono inviare messaggi sia a code locali che remote. Tuttavia, possono ricevere messaggi solo da una coda locale.

**Coda di iniziazione:** Le *code di iniziazione* sono code che vengono utilizzate nel triggering. Un Queue Manager invia un messaggio trigger ad una coda di iniziazione qualora si verifichi un evento trigger. Un evento trigger è una combinazione logica di condizioni che viene rilevata da un Queue Manager. Ad esempio, può essere generato un evento trigger quando il numero di messaggi in una coda raggiunge la capacità predefinita per quella coda. Questo evento fa sì che il Queue Manager invii un messaggio trigger alla coda di iniziazione interessata. Questo messaggio trigger viene ricevuto da un *trigger monitor*, un'applicazione

## Oggetti

particolare che controlla una coda di iniziazione. Quindi il trigger monitor avvia il programma applicativo specificato nel messaggio trigger.

Prima che un Queue Manager possa utilizzare la funzione triggering, deve essere definita almeno una coda di iniziazione relativa a quel Queue Manager.

Consultare le sezioni "Gestione degli oggetti per triggering" a pagina 59 e "runmqtrm (Avvia un trigger monitor)" a pagina 309. Per ulteriori informazioni relative alla funzione triggering, consultare il manuale *MQSeries Application Programming Guide*.

**Code di trasmissione:** Una *coda di trasmissione* memorizza temporaneamente i messaggi destinati ad un Queue Manager remoto. È necessario definire almeno una coda di trasmissione per ogni Queue Manager remoto, alla quale il Queue Manager locale può inviare messaggi direttamente. Queste code vengono utilizzate anche nella gestione remota; consultare la sezione "Gestione remota da un Queue Manager locale mediante comandi MQSC" a pagina 71. Per informazioni relative all'utilizzo delle code di trasmissione nell'accodamento distribuito, consultare il manuale *MQSeries Intercommunication*.

**Code di trasmissione del cluster:** Ogni Queue Manager all'interno di un cluster possiede una coda di trasmissione del cluster denominata SYSTEM.CLUSTER.TRANSMIT.QUEUE. Nella Versione 5.1, per impostazione predefinita, su ogni Queue Manager viene creata una definizione di questa coda.

Un Queue Manager che compone un cluster può inviare messaggi tramite la coda di trasmissione del cluster a qualsiasi Queue Manager all'interno dello stesso cluster.

I Queue Manager del cluster possono comunicare con i Queue Manager che non fanno parte del proprio cluster. A questo scopo è necessario definire i canali e una coda di trasmissione da un Queue Manager del cluster verso un Queue Manager fuori dal cluster, così come avviene in un normale ambiente di accodamento distribuito.

Durante la risoluzione del nome, la coda di trasmissione del cluster assume prevalenza sulla coda di trasmissione predefinita. Quando un Queue Manager esterno al cluster invia un messaggio ad una coda remota, per impostazione predefinita, viene utilizzata la coda di trasmissione predefinita, a meno che vi sia una coda di trasmissione con lo stesso nome del Queue Manager di destinazione.

Se il Queue Manager fa parte di un cluster, per impostazione predefinita viene utilizzato SYSTEM.CLUSTER.TRANSMIT.QUEUE, a meno che il Queue Manager di destinazione sia estraneo al cluster.

**Code messaggi non recapitati:** Una *coda messaggi non recapitati* memorizza i messaggi che non riescono a raggiungere la loro destinazione. Ciò accade, ad esempio, quando la coda di destinazione è piena. La coda messaggi non recapitati predefinita è denominata SYSTEM.DEAD.LETTER.QUEUE. Su altre piattaforme queste code vengono denominate diversamente (undelivered-message).

Per l'accodamento distribuito è necessario definire una coda messaggi non recapitati su ciascun Queue Manager incluso.

**Code comandi:** La coda comandi, denominata SYSTEM.ADMIN.COMMAND.QUEUE, è una coda locale a cui le applicazioni

opportunamente autorizzate possono inviare i comandi MQSeries da elaborare. Questi comandi vengono poi richiamati da un componente MQSeries denominato server di comandi. Il server di comandi analizza i comandi e trasmette quelli validi al Queue Manager affinché siano elaborati e poi invia le risposte alle appropriate code di risposta.

Una coda comandi viene generata automaticamente per ogni Queue Manager alla creazione dello stesso Queue Manager.

**Code di risposta RTQ (Reply-to queues):** Quando un'applicazione riceve il messaggio richiesto può restituire un messaggio di risposta all'applicazione che lo ha inviato. Tale messaggio viene inviato ad una coda definita coda di risposta, che solitamente è una coda locale rispetto all'applicazione di invio. Il nome della coda di risposta è specificato dall'applicazione di invio come parte del descrittore di messaggio.

**Code di evento:** MQSeries Versione 5.1 supporta eventi di strumentazione, che possono essere utilizzati per controllare i Queue Manager indipendentemente dalle applicazioni MQI. Gli eventi di strumentazione possono essere generati in diversi modi, ad esempio:

- Un'applicazione tenta di inviare un messaggio ad una coda che non è disponibile o non esiste.
- Una coda diviene piena.
- Viene avviato un canale.

Quando si verifica un evento di strumentazione, il Queue Manager invia un messaggio di evento ad una coda di evento. Questo messaggio può così essere rilevato da un'applicazione di controllo, che può informare un amministratore oppure iniziare alcune attività di ripristino nel caso in cui l'evento indichi un problema.

**Nota:** gli eventi di trigger sono diversi dagli eventi di strumentazione poiché sono prodotti da condizioni diverse e non generano messaggi di evento.

Per ulteriori informazioni relative all'evento di strumentazione, consultare il manuale *MQSeries Programmable System Management*.

## Definizioni di processo

Un *oggetto definizione di processo* definisce un'applicazione che viene avviata come conseguenza di un evento di trigger in un Queue Manager MQSeries. Per ulteriori informazioni, consultare la sezione "Coda di iniziazione" a pagina 11.

Gli attributi di questo oggetto sono l'ID applicazione, il tipo di applicazione e i dati specifici per l'applicazione.

Per creare una definizione di processo utilizzare il comando DEFINE PROCESS di tipo MQSC oppure il comando Create Process di tipo PCF.

## Canali

I *canali* sono oggetti che forniscono un percorso di comunicazione tra i Queue Manager. I canali vengono utilizzati nell'accodamento distribuito di messaggi per trasportare i messaggi tra i diversi Queue Manager. Essi separano le applicazioni dai protocolli di comunicazione di base. I Queue Manager possono trovarsi sulla stessa piattaforma o su piattaforme diverse. Per la comunicazione tra Queue



## Oggetti

Manager, è necessario definire un oggetto canale relativo al Queue Manager che invia i messaggi ed un altro canale complementare relativo al Queue Manager che riceve i messaggi.

Per informazioni sui canali e sul loro utilizzo, consultare il manuale *MQSeries Intercommunication* e la sezione "Preparazione di canali e code di trasmissione per la gestione remota" a pagina 72.

## Cluster

In una normale rete di MQSeries che utilizza l'accodamento distribuito, i Queue Manager sono tra loro indipendenti. Affinché un Queue Manager possa inviare un messaggio ad un altro Queue Manager, occorre aver definito una coda di trasmissione, un canale diretto al Queue Manager remoto ed una coda di definizione remota per ogni coda a cui il Queue Manager dovrà inviare messaggi.

Un cluster è un gruppo di Queue Manager impostato in modo che i Queue Manager possano comunicare direttamente fra loro attraverso una rete, senza la necessità di complesse code di trasmissione e definizioni di canali e coda.

**Nota:** i cluster MQSeries sono diversi dai cluster OpenVMS. Quando viene utilizzato il termine *cluster*, esso si riferisce ai cluster di Queue Manager MQSeries. Quando si fa riferimento ai cluster OpenVMS viene sempre utilizzato il termine *cluster OpenVMS*. Per ulteriori informazioni relative ai cluster OpenVMS, consultare il "Capitolo 16. MQSeries per OpenVMS e clustering" a pagina 227.

Per informazioni relative ai cluster, consultare il "Capitolo 6. Gestione di oggetti remoti di MQSeries" a pagina 69 e il manuale *MQSeries Queue Manager Clusters*.

## Namelist

Una namelist è un oggetto MQSeries che contiene un elenco di altri oggetti MQSeries. Solitamente, le namelist vengono utilizzate da applicazioni quali trigger monitor, per identificare un gruppo di code. Il vantaggio dato dall'utilizzo di una namelist consiste nel fatto che la namelist è indipendente dall'applicazione; pertanto può essere aggiornata senza dover arrestare nessuna delle applicazioni che la utilizzano. Inoltre, se un'applicazione ha esito negativo, la namelist non viene coinvolta e può essere ancora utilizzata dalle altre applicazioni.

Le namelist vengono anche utilizzate con i cluster di Queue Manager in modo da poter gestire una lista di cluster a cui si riferisce più di un oggetto MQSeries.

---

## Oggetti definiti dal sistema

Gli *oggetti predefiniti di sistema* sono un insieme di definizioni di oggetti che vengono creati automaticamente ogni volta che viene creato un queue manager. È possibile creare o modificare ciascuna di queste definizioni di oggetti per utilizzarle nelle applicazioni al momento dell'installazione. I nomi degli oggetti predefiniti hanno l'estensione SYSTEM.DEF; ad esempio, SYSTEM.DEFAULT.LOCAL.QUEUE è la coda locale predefinita; SYSTEM.DEF.RECEIVER è il canale ricevente predefinito. Non è possibile ridenominare questi oggetti; l'estensione predefinita di questi nomi è necessaria.

Quando si definisce un oggetto, tutti gli attributi non specificati vengono copiati dal corrispondente oggetto predefinito. Ad esempio, gli attributi, che non vengono specificati durante la creazione di una coda locale, vengono copiati dalla coda predefinita SYSTEM.DEFAULT.LOCAL.QUEUE.



Consultare l'Appendice B. Impostazioni predefinite del sistema" a pagina 327 per ulteriori informazioni relative alle impostazioni predefinite.

---

### Gestione locale e remota

La gestione locale consente di eseguire attività di gestione su ogni Queue Manager precedentemente definito nel sistema locale. È possibile accedere ad altri sistemi, ad esempio, attraverso il programma **telnet** di emulazione TCP/IP, ed eseguirvi attività di gestione. In MQSeries questa può considerarsi una gestione locale poiché non si utilizzano canali, visto che le comunicazioni sono gestite dal sistema operativo.

MQSeries supporta la gestione da un postazione remota definita *gestione remota*. Ciò consente di inviare al proprio sistema locale comandi che vengono elaborati da un altro sistema. Non è necessario collegarsi a quel sistema, sebbene occorra aver definito i canali in maniera corretta. Il Queue Manager e il server di comandi del sistema destinatario devono essere stati avviati. Ad esempio, è possibile comandare la modifica di una definizione di coda in un Queue Manager remoto.

Alcuni comandi non possono essere inviati in questo modo; in particolare la creazione o l'avvio dei Queue Manager e l'avvio dei server di comandi. Per eseguire queste attività, è necessario collegarsi al sistema remoto ed inviare il comando oppure creare un processo in grado di inviare tali comandi.

---

### Client e server

MQSeries supporta configurazioni client-server per applicazioni MQSeries.

Un *client MQSeries* è un componente del prodotto MQSeries installato su una macchina per ricevere le chiamate MQI dalle applicazioni e per inviarle ad una macchina *server MQI*. Qui vengono elaborate da un Queue Manager. Solitamente, il client e il server risiedono su macchine diverse, ma possono anche risiedere sulla stessa macchina.

Un *server MQI* è un Queue Manager che fornisce servizi di accodamento ad uno o più client. Tutti gli oggetti MQSeries, ad esempio le code, si trovano solo sulla macchina Queue Manager e cioè sulla macchina server MQI. Un server può supportare anche normali applicazioni locali MQSeries.

La differenza tra un server MQI e un normale Queue Manager sta nel fatto che un server ha un collegamento per le comunicazioni specifico per ogni client. Per ulteriori informazioni relative alla creazione di canali per client e server, consultare il manuale *MQSeries Intercommunication*.

Per informazioni generali relative ai client, consultare il manuale *MQSeries - Client*.

### Applicazioni MQSeries in un ambiente client-server

Collegate ad un server, le applicazioni client MQSeries possono inviare chiamate MQI così come fanno le applicazioni locali. L'applicazione client inoltra una chiamata MQCONN per collegarsi ad uno specifico Queue Manager. Successivamente vengono elaborate tutte le chiamate MQI aggiuntive che specificano l'handle di connessione restituito dalla richiesta di connessione.

È necessario collegare le applicazioni alle librerie client appropriate. Per ulteriori informazioni consultare il manuale *MQSeries - Client*.

### Estensione delle funzioni Queue Manager

Le funzioni di un Queue Manager possono essere estese da:

- Uscite utente
- Servizi installabili

## Uscite utente

L'uscita utente prevede un meccanismo che consente all'utente di inserire il proprio codice in una funzione Queue Manager. Le uscite utente supportate sono:

- **Uscite del canale**

Queste uscite modificano il funzionamento dei canali. Le uscite del canale sono descritte nel manuale *MQSeries Intercommunication*.

- **Uscite di conversione dati**

Queste uscite creano frammenti di codice sorgente che possono essere inviate ad un programma applicativo per convertire dati da un formato ad un altro. Le uscite di conversione dati sono descritte nel manuale *MQSeries Application Programming Guide*.

- **Cluster workload exit**

La funzione consentita da questa uscita è definita dal fornitore. Le informazioni sulla definizione della chiamata sono contenute nel manuale *MQSeries Queue Manager Clusters*.

Tutti i tipi di uscite sono correlati all'accodamento distribuito. Per ulteriori informazioni su queste uscite e sul loro utilizzo consultare il manuale *MQSeries Intercommunication*.

## Servizi installabili

I servizi installabili realizzano una maggiore estensione delle uscite poiché posseggono interfacce standard (API) ad ingressi multipli.

L'implementazione di un servizio installabile si definisce *service component*. È possibile utilizzare i componenti forniti con il prodotto oppure è possibile compilarne uno per eseguire le funzioni desiderate.

Attualmente vengono forniti i seguenti servizi installabili:

- **Authorization Service.**

L'Authorization Service consente la creazione di una propria funzione di sicurezza.

Il componente di servizio predefinito che implementa il servizio è l'OAM (Object Authority Manager), fornito con il prodotto. Per impostazione predefinita, il componente OAM è attivo e ciò significa che non è necessaria alcuna operazione di configurazione. È possibile utilizzare l'interfaccia Authorization Service per creare altri componenti che sostituiscano o incrementino l'OAM. Per ulteriori informazioni relative a questo componente, consultare il "Capitolo 7. Protezione di oggetti di MQSeries" a pagina 85.

- **Name Service.**

Name service abilita la condivisione delle code consentendo alle applicazioni di identificare le code remote ritenute code locali. Un componente di servizio predefinito che implementa name service è fornito con MQSeries Versione 5.1. Esso utilizza OSF (Open Software Foundation) DCE (Distributed Computing Environment). È tuttavia possibile compilare un componente Name Service, ad esempio, se non si è installato DCE. Per impostazione predefinita, name service non è attivo.

Consultare il "Capitolo 12. Utilizzo di Name Service" a pagina 177 ed anche il manuale *MQSeries Programmable System Management*.

### Sicurezza

Nei prodotti MQSeries Versione 5, esistono due metodi per fornire sicurezza:

- La funzione OAM (Object Authority Manager)
- La sicurezza DCE

#### Funzione OAM (Object Authority Manager)

L'autorizzazione ad utilizzare comandi per le chiamate MQI e per accedere agli oggetti è fornita da OAM (Object Authority Manager), che per impostazione predefinita è abilitata. L'accesso alle entità MQSeries è controllata attraverso il gruppo utenti MQSeries e l'OAM. È prevista un'interfaccia riga di comandi per abilitare gli amministratori a concedere o revocare le autorizzazioni.

Per ulteriori informazioni relative alla creazione di componenti Authorization Service, consultare il manuale *MQSeries Programmable System Management*.

#### Sicurezza DCE

Le uscite del canale che utilizzano il GSS (Generic Security Service) DCE sono fornite da MQSeries. Per ulteriori informazioni, consultare il manuale *MQSeries Intercommunication*.

---

### Supporti transazionali

Un programma applicativo può raggruppare un insieme di aggiornamenti in un'*unità di lavoro*. Tali aggiornamenti sono solitamente in relazione logica e devono essere privi di errori per preservare l'integrità dei dati. Se anche un solo aggiornamento si arresta, l'integrità dei dati verrà compromessa.

Un'unità di lavoro esegue il commit quando ha esito positivo. Da questo momento tutti gli aggiornamenti realizzati in quella unità di lavoro diventano permanenti o irreversibili. Se invece l'unità di lavoro dovesse avere esito negativo, allora di tutti gli aggiornamenti verrà eseguito il back out. Syncpoint coordination è il processo attraverso il quale si esegue il commit o il back out delle intere unità di lavoro.

Un'unità di lavoro *locale* si caratterizza per il fatto che le uniche risorse aggiornate sono quelle del Queue Manager MQSeries. In queste unità, syncpoint coordination è prodotto autonomamente dal Queue Manager utilizzando un processo commit a fase singola.

Un'unità di lavoro *globale* si caratterizza per il fatto che vengono aggiornate anche le risorse relative ad altri gestori di risorse, quali un database XA compatibile. In queste unità deve essere utilizzato un processo commit a doppia fase e l'unità di lavoro deve essere coordinata da Queue Manager autonomamente.

Per ulteriori informazioni, consultare il "Capitolo 10. Supporto transazionale" a pagina 127.

---

## Capitolo 2. Introduzione alla gestione di MQSeries

Questo capitolo ha lo scopo di fornire informazioni di base riguardo la gestione di MQSeries.

Si considerano attività di gestione la creazione, l'avvio, la modifica, la visualizzazione, l'arresto e l'eliminazione di oggetti MQSeries (Queue Manager, code, processi, namelist, cluster e canali).

Questo capitolo contiene le seguenti sezioni:

- "Gestione locale e in remoto"
- "Esecuzione delle attività di gestione utilizzando i comandi di controllo"
- "Esecuzione delle attività di gestione utilizzando i comandi MQSC" a pagina 20
- "Esecuzione delle attività di gestione utilizzando i comandi PCF" a pagina 20
- "Analisi dei nomi di file MQSeries" a pagina 21.
- "Analisi della sensibilità al minuscolo/maiuscolo" a pagina 23

---

### Gestione locale e in remoto

È possibile gestire oggetti MQSeries in locale o in remoto.

La *gestione locale* consente di eseguire attività di gestione su ogni Queue Manager precedentemente definito nel sistema locale. È possibile accedere ad altri sistemi, ad esempio, attraverso il programma telnet di emulazione del terminale TCP/IP ed eseguire attività di gestione. In MQSeries questa può considerarsi una gestione locale poiché non si utilizzano canali, visto che le comunicazioni sono gestite dal sistema operativo.

MQSeries supporta la gestione da un determinato punto attraverso ciò che viene definito gestione in remoto. Ciò consente di inviare dal proprio sistema locale comandi che vengono elaborati da un altro sistema. Non è necessario collegarsi a quel sistema, sebbene occorra aver definito i canali in maniera corretta. Il Queue Manager e il server di comando del sistema destinatario devono essere stati avviati. Ad esempio, è possibile emettere un comando in remoto per modificare una definizione di coda in un Queue Manager remoto.

Alcuni comandi non possono essere inviati in questo modo; in particolare la creazione o l'avvio dei Queue Manager e l'avvio dei server di comando. Per eseguire queste attività, è necessario collegarsi al sistema remoto ed inviare il comando oppure creare un processo in grado di inviare tali comandi.

Il "Capitolo 6. Gestione di oggetti remoti di MQSeries" a pagina 69 descrive nei dettagli le attività di gestione remota.

---

### Esecuzione delle attività di gestione utilizzando i comandi di controllo

I *comandi di controllo* consentono di eseguire attività di gestione sugli stessi Queue Manager.

Consultare il "Capitolo 3. Gestione dei Queue Manager utilizzando i comandi di controllo" a pagina 27 per ulteriori informazioni relative ai comandi di controllo.

### Esecuzione delle attività di gestione utilizzando i comandi MQSC

Utilizzare i comandi MQSeries (MQSC) per gestire gli oggetti Queue Manager, inclusi gli stessi Queue Manager, i canali, le code e le definizioni di processo. Ad esempio, esistono comandi che definiscono, modificano, visualizzano e cancellano una determinata coda.

E' possibile eseguire i comandi MQSC richiamando il comando **runmqsc** da una riga comandi. È possibile eseguire i comandi MQSC:

- In modo interattivo digitandoli sulla tastiera. Consultare la sezione "Utilizzo interattivo della funzione MQSC" a pagina 39.
- Come sequenza di comandi da un file di testo ASCII. Consultare la sezione "Esecuzione di comandi MQSC da file di testo" a pagina 42.

Il comando **runmqsc** può essere eseguito in tre modi che si differenziano in base al tipo di flag impostato per il comando.

- *Verification mode (Modo Verifica)*, in cui i comandi MQSC vengono verificati su un Queue Manager locale, ma non vengono eseguiti.
- *Direct mode (Modo Diretto)*, in cui i comandi MQSC vengono eseguiti su un Queue Manager locale.
- *Indirect mode (Modo Indiretto)*, in cui i comandi vengono eseguiti su un Queue Manager remoto.

Per ulteriori informazioni relative all'utilizzo delle funzioni MQSC e ai file di testo, consultare la sezione "Esecuzione di comandi MQSC da file di testo" a pagina 42. Per ulteriori informazioni relative al comando **runmqsc**, consultare la sezione "runmqsc (Esegue comandi di MQSeries)" a pagina 305.

Gli attributi degli oggetti specificati in MQSC sono riportati in questo manuale in lettere maiuscole (ad esempio RQMNAME), sebbene essi non siano sensibili al maiuscolo/minuscolo. I nomi degli attributi MQSC hanno una lunghezza massima di otto caratteri.

I comandi MQSC sono disponibili su altre piattaforme, incluse AS/400® e OS/390.

I comandi MQSC sono riassunti nell'"Appendice D. Confronto tra gruppi di comandi" a pagina 335.

Consultare il manuale *MQSeries - Guida di riferimento per i comandi* per la descrizione di ogni comando MQSC e della relativa sintassi.

Consultare la sezione "Esecuzione delle attività di gestione locale utilizzando i comandi MQSC" a pagina 38 per ulteriori informazioni relative all'utilizzo dei comandi MQSC nella gestione locale.

---

### Esecuzione delle attività di gestione utilizzando i comandi PCF

La funzione dei comandi PCF (Programmable Command Format) è di consentire la pianificazione delle attività di gestione in un programma di gestione. In questo modo è possibile creare code e definizioni di processi oltre che effettuare modifiche relative ai Queue Manager da un programma.

Le funzioni MQSC sono realizzate in modo analogo dai comandi PCF.

Consultare la sezione "Comandi PCF" a pagina 63 per ulteriori informazioni.

Per una descrizione completa delle strutture dei dati PCF e sulla loro implementazione, consultare il manuale *MQSeries Programmable System Management*.

Per ottenere un accesso più semplice ai messaggi PCF utilizzare MQAI (MQSeries Administration Interface). Maggiori dettagli sono contenuti nella sezione "Utilizzo di MQAI per semplificare l'impiego di PCF" a pagina 64.

## Attributi MQSC e PCF

Gli attributi degli oggetti specificati in MQSC sono riportati in questo manuale in lettere maiuscole, ad esempio RQMNAME, sebbene essi non siano sensibili al maiuscolo/minuscolo. I nomi di questi attributi hanno una lunghezza massima di otto caratteri, e ciò rende spesso difficile comprenderne il significato, come avviene ad esempio con QDPHIEV. Gli attributi in PCF vengono visualizzati con il tipo di carattere "italic", non hanno il limite degli otto caratteri e sono perciò più facili da leggere. L'equivalente in PCF di RQMNAME è *RemoteQMgrName* e l'equivalente di QDPHIEV è *QDepthHighEvent*.

## Escape PCF

Gli Escape PCF sono comandi PCF che contengono i comandi MQSC con il messaggio di testo. È possibile utilizzare i comandi PCF per inviare comandi ad un Queue Manager remoto. Per ulteriori informazioni relative all'utilizzo di escape PCF, consultare il manuale *MQSeries Programmable System Management*.

---

## Analisi dei nomi di file MQSeries

Tutti gli oggetti MQSeries, coda, Queue Manager, namelist o processo, sono rappresentati da un file. Dal momento che non tutti nomi sono necessariamente validi nomi di file, il Queue Manager converte i nomi invalidi.

Il percorso per la directory di un Queue Manager è formata in questo modo:

- Un prefisso - la prima parte del nome:

```
MQS_ROOT:[MQM]
```

Tale prefisso è definito nel file di configurazione del Queue Manager.

- Un'espressione:

```
QMGRS
```

- Un nome codificato di Queue Manager, che consiste nel nome del Queue Manager trasformato in un valido nome di directory. Ad esempio il Queue Manager:

```
QUEUE.MANAGER
```

potrà essere trasformato in:

```
QUEUE$MANAGER
```

Questo processo viene definito *trasformazione del nome*.

## Trasformazione del nome di Queue Manager

In MQSeries è possibile dare ad un Queue Manager un nome che contiene fino a 48 caratteri.

Ad esempio, è possibile denominare un Queue Manager:

```
QUEUE.MANAGER.ACCOUNTING.SERVICES
```

## **Analisi dei nomi MQSeries**

Tuttavia, ogni Queue Manager è rappresentato da un file il cui nome è limitato nella lunghezza e nel tipo di caratteri utilizzabili. Quindi i nomi dei file che rappresentano oggetti sono automaticamente trasformati per concordare con i requisiti imposti dal sistema.



Di seguito vengono riportate le norme che regolano la trasformazione del nome di un Queue Manager, a cui viene dato, a titolo di esempio, il nome `QUEUE.MANAGER`:

1. Trasformazione dei singoli caratteri:
  - viene trasformato in `$`
  - / viene trasformato in `_`
  - % viene trasformato in `_`
2. Se il nome non è ancora valido:
  - a. Eliminazione dei caratteri eccedenti l'ottavo
  - b. Aggiunta di un suffisso numerico di tre caratteri

Ad esempio, assumendo il prefisso predefinito, il nome del Queue Manager viene trasformato in:

```
MQS_ROOT: [MQM.QMGRS.QUEUE$MANAGER]
```

L'algoritmo consente anche di distinguere i nomi che differiscono solo nel carattere maiuscolo o minuscolo, in quei sistemi che non sono sensibili a questa caratteristica.

## Trasformazione del nome di oggetto

I nomi degli oggetti non sono necessariamente dei validi nomi di file system. Pertanto, potrebbe essere necessaria la trasformazione del nome di un oggetto. Il metodo utilizzato in questo caso è diverso rispetto a quello utilizzato per i nomi di Queue Manager, poiché vi sono solo pochi Queue Manager in una macchina, mentre esistono moltissimi oggetti per ogni Queue Manager. Solo le definizioni di processo, le code e i namelist sono rappresentati nel file system; queste considerazioni non riguardano i canali.

Il nome generato dal processo di trasformazione non assomiglia per niente al nome originario. Per convertire i nomi trasformati nei nomi originari, utilizzare il comando `dsqmqls`.

I nomi dei file di coda iniziano con la lettera "Q".

Per ulteriori informazioni sulla denominazione degli oggetti, consultare la sezione "Regole per la denominazione degli oggetti di MQSeries" a pagina 251.

---

## Analisi della sensibilità al minuscolo/maiuscolo

### Sensibilità al minuscolo/maiuscolo nei comandi di controllo

OpenVMS è solitamente descritto come un sistema operativo che non rileva le differenze tra lettere minuscole e maiuscole. Questo significa che, solitamente, i seguenti tre comandi generano tutti un Queue Manager denominato "QUEUEMANAGER".

```
$ crtmqm QueueManager
$ crtmqm queuemanager
$ crtmqm QUEUEMANAGER
```

Con MQSeries per Compaq OpenVMS, è possibile utilizzare le virgolette (o un parametro simile) prima e dopo il nome del Queue Manager in modo da rendere rilevante la differenza tra le lettere maiuscole e le minuscole. Infatti, quando vengono utilizzate le virgolette, i seguenti tre comandi generano tre diversi Queue Manager.

## Analisi dei nomi MQSeries

\$ crtmqm "QueueManager"	crea un Queue Manager denominato	<b>QueueManager</b>
\$ crtmqm "queuemanager"	crea un Queue Manager denominato	<b>queuemanager</b>
\$ crtmqm "QUEUEMANAGER"	crea un Queue Manager denominato	<b>QUEUEMANAGER</b>

OpenVMS Version 7.2 ha introdotto il seguente nuovo comando:

```
$ set process /parse_style = ( traditional | extended )
```

Questo comando modifica la rilevanza che Open VMS riconosce al carattere minuscolo o maiuscolo.

Se non viene utilizzato il comando **set process /parse\_style** oppure viene utilizzato con l'opzione **traditional**, Open VMS riconosce la solita rilevanza al carattere.

Se invece questo comando viene utilizzato con l'opzione **extended**, la run time library routine LIB\$GET\_FOREIGN funziona in modo diverso, cioè in modo da riconoscere la differenza tra i caratteri in maiuscolo e i caratteri in minuscolo. Dal momento che MQSeries utilizza questa routine per acquisire i parametri dalle righe di comando, la differenza del tipo di carattere in questi parametri verrà rilevata anche se essi non vengono posti tra le virgolette.

Ad esempio, la sequenza di comandi che segue crea tre diversi Queue Manager. Si noti che i parametri non sono stati posti tra virgolette.

```
$ set process /parse_style = extended
$ crtmqm QueueManager   crea un Queue Manager denominato QueueManager
$ crtmqm queuemanager   crea un Queue Manager denominato queuemanager
$ crtmqm QUEUEMANAGER   crea un Queue Manager denominato QUEUEMANAGER
```

Il comando OpenVMS `set process /parse_style` produce anche altri cambiamenti. Prima di utilizzare tale comando nel sistema è consigliabile apprendere le ulteriori informazioni fornite nel dizionario OpenVMS DCL.

## Sensibilità al minuscolo/maiuscolo nei comandi MQSC

I comandi di controllo MQSeries (ad esempio, **runmqsc** che richiama la funzione MQSC) non sono sensibili alla differenza tra il minuscolo e il maiuscolo.

I comandi MQSC e i relativi attributi possono essere digitati sia nel carattere minuscolo che nel maiuscolo. I nomi nei comandi MQSC vengono automaticamente trasformati in lettere maiuscole se non sono inclusi tra *singole* virgolette. Se non vengono utilizzate virgolette singole, il nome viene elaborato come se fosse scritto in lettere maiuscole. Consultare il manuale *MQSeries - Guida di riferimento per i comandi* per ulteriori informazioni.



---

## Capitolo 3. Gestione dei Queue Manager utilizzando i comandi di controllo

Questo capitolo spiega come eseguire le operazioni relative ai Queue Manager e ai server di comandi. Esso contiene le seguenti sezioni:

- "Utilizzo dei comandi di controllo"
- "Indicazioni per la creazione dei Queue Manager" a pagina 28
- "Creazione di un Queue Manager predefinito" a pagina 31
- "Avvio di un Queue Manager" a pagina 32
- "Rendere predefinito un Queue Manager esistente" a pagina 32
- "Arresto di un Queue Manager" a pagina 33
- "Riavvio di un Queue Manager" a pagina 34
- "Per eliminare un Queue Manager" a pagina 34
- "Analisi dei file di oggetto" a pagina 251

---

### Utilizzo dei comandi di controllo

I comandi di controllo vengono utilizzati per eseguire le operazioni relative ai Queue Manager, ai server di comando e ai canali. I comandi di controllo possono essere suddivisi in tre categorie, così come mostrato nella Tabella 1 a pagina 27.

Tabella 1. Categorie dei comandi di controllo.

Categoria	Descrizione
Comandi del Queue Manager	I comandi di controllo del Queue Manager consentono di creare, avviare, arrestare ed eliminare Queue Manager e server di comandi.
Comandi del canale	I comandi di controllo del canale consentono di avviare e terminare i canali e i channel initiators (programmi di inizializzazione canale).
Comandi delle utilità	I comandi delle utilità includono comandi associati a: <ul style="list-style-type: none"><li>• Esecuzione di comandi MQSC</li><li>• Uscite di conversione</li><li>• Gestione delle autorizzazioni</li><li>• Registrazione e ripristino delle immagini oggetto delle risorse Queue Manager</li><li>• Visualizzazione e risoluzione di transazioni</li><li>• Trigger monitor</li><li>• Visualizzazione dei nomi di file degli oggetti MQSeries</li></ul>

Per informazioni relative alle attività di gestione dei canali, consultare il manuale *MQSeries Intercommunication*.

### Utilizzo dei comandi di controllo

In MQSeries per Compaq OpenVMS, i comandi di controllo vengono immessi al richiesta comandi DCL. Il nome del comando e i flag possono essere immessi indifferentemente sia in minuscolo che in maiuscolo, ma non è così per i parametri:

## Utilizzo dei comandi di controllo

dipende dall'opzione di elaborazione OpenVMS e dall'aver incluso o meno i parametri tra le virgolette, in modo da proteggere la differenza tra le lettere minuscole e quelle maiuscole. Per ulteriori informazioni sulla sensibilità al minuscolo/maiuscolo nei comandi OpenVMS e sull'utilizzo delle virgolette, consultare la sezione "Analisi della sensibilità al minuscolo/maiuscolo" a pagina 23.

Si noti questo esempio:

```
crtmqm -u system.dead.letter.queue "jupiter.queue.manager"
```

- La coda messaggi non recapitati può essere SYSTEM.DEAD.LETTER.QUEUE, anche se è stata immessa in lettere minuscole. La conversione automatica di questi caratteri in lettere maiuscole dipende dall'impostazione del comando OpenVMS **set process/parse\_style**. Consultare la sezione "Analisi della sensibilità al minuscolo/maiuscolo" a pagina 23.
- Il nome del Queue Manager viene specificato come "jupiter.queue.manager", che è differente da "JUPITER.queue.manager", poiché è incluso tra le virgolette.

Pertanto, occorre immettere i comandi facendo attenzione a copiarli così come sono riportati negli esempi.

---

## Creazione di un Queue Manager

Un Queue Manager gestisce le risorse associate ad esso, in particolare le code di sua pertinenza. Esso fornisce i servizi di accodamento alle applicazioni per le chiamate MQI (Message Queueing Interface) e i comandi per la creazione, la modifica, la visualizzazione e l'eliminazione di oggetti MQSeries.

Prima di utilizzare messaggi e code, è necessario creare almeno un queue manager ed i relativi oggetti associati. Utilizzare il comando di controllo MQSeries **crtmqm** per creare un Queue Manager. Il comando **crtmqm** crea automaticamente gli oggetti predefiniti e gli oggetti di sistema necessari. Gli oggetti predefiniti costituiscono la base di qualsiasi definizione di oggetto che si realizza; gli oggetti di sistema sono necessari alle operazioni del Queue Manager. Dopo aver creato un Queue Manager ed i relativi oggetti, utilizzare il comando **strmqm** per avviare il queue manager.

## Indicazioni per la creazione dei Queue Manager

Prima di creare un Queue Manager occorre essere a conoscenza di alcune informazioni importanti (specialmente in un ambiente di produzione). Esaminare l'elenco che segue:

- Specificare un nome univoco per il Queue Manager.
- Limitare il numero dei Queue Manager.
- Specificare un Queue Manager predefinito.
- Specificare una coda messaggi non recapitati.
- Specificare una coda di trasmissione predefinita.
- Specificare i parametri necessari alla registrazione nei log.
- Eseguire una copia di riserva dei file di configurazione dopo aver creato un Queue Manager.

Le sezioni che seguono spiegano le operazioni riportate in questo elenco.

### Specificare un nome univoco per il Queue Manager

Quando si crea un Queue Manager bisogna fare attenzione a non assegnargli lo stesso nome di un altro Queue Manager *già* presente nella rete. I nomi dei Queue Manager non vengono controllati al momento della loro creazione; a causa dei nomi utilizzati più volte non sarà possibile utilizzare i canali per l'accodamento distribuito.

Per assicurare l'univocità dei nomi è consigliabile includere nel nome del Queue Manager il nome del relativo nodo (univoco). Ad esempio, se un nodo è denominato *accounts*, si consiglia di denominare il Queue Manager *accounts.saturn.queue.manager*, in cui *saturn* identifica un determinato Queue Manager e *queue.manager* è l'estensione che può essere assegnata a qualsiasi Queue Manager. In alternativa, è possibile ignorare tale suggerimento, ma si noti che *accounts.saturn* e *accounts.saturn.queue.manager* sono nomi di Queue Manager *diversi*.

Se si utilizza MQSeries nelle comunicazioni con altre società, è possibile inserire il nome della società come prefisso. Ciò non è mostrato negli esempi, perché altrimenti diventerebbero troppo complessi.

**Nota:** i nomi dei Queue Manager nei comandi di controllo possono essere convertiti o meno in lettere maiuscole, dipende dall'opzione di elaborazione OpenVMS e dall'aver incluso o meno i nomi tra le virgolette, in modo da proteggere la differenza tra le lettere minuscole e quelle maiuscole. Ciò significa che è possibile creare due Queue Manager, uno con il nome *jupiter.queue.manager* e l'altro con il nome *JUPITER.queue.manager*. Per ulteriori informazioni sulla sensibilità al minuscolo/maiuscolo nell'opzione di elaborazione OpenVMS e sull'utilizzo delle virgolette, consultare la sezione "Analisi della sensibilità al minuscolo/maiuscolo" a pagina 23.

### Limitare il numero dei Queue Manager

È possibile creare tanti Queue Manager quanti ne consentono le risorse. Tuttavia, poiché ogni Queue Manager occupa le sue risorse, è solitamente consigliabile definire un queue Manager con 100 code su un nodo, piuttosto che dieci Queue Manager con dieci code ciascuno.

Nei sistemi di produzione, molti nodi saranno eseguiti con un singolo Queue Manager, ma macchine server più potenti potranno lavorare con più Queue Manager.

### Specificare il Queue Manager predefinito

Ogni nodo deve avere un Queue Manager predefinito, anche se è possibile configurare MQSeries su un nodo che ne sia privo.

Per creare un Queue Manager utilizzare il comando **crtmqm**. Per una descrizione dettagliata di questo comando e dei relativi parametri, consultare la sezione "crtmqm (Crea Queue Manager)" a pagina 258.

### Cos'è un Queue Manager predefinito?

Il Queue Manager predefinito è il Queue Manager cui le applicazioni si connettono quando si specifica un nome di Queue Manager nella chiamata MQCONN. Inoltre il Queue Manager predefinito elabora i comandi MQSC quando si richiama il comando **runmqsc** senza specificare un nome di Queue Manager.

## Creazione dei Queue Manager

### Come specificare un Queue Manager predefinito?

Per specificare che il Queue Manager che si sta creando è il Queue Manager predefinito, è necessario includere il flag `-q` nel comando `crtmqm`. Se non si desidera rendere predefinito il queue manager che si sta creando, omettere questo indicatore.

Quando si specifica un Queue Manager come predefinito, si *sostituisce* qualunque specificazione relativa al precedente Queue Manager predefinito per il nodo.

### Cosa accade se si decide di cambiare il Queue Manager predefinito?

Se si decide di cambiare il Queue Manager predefinito, si ricordi che tale modifica avrà effetto sulle altre applicazioni e gli altri utenti. Le modifiche non hanno però effetto sulle applicazioni attualmente connesse, poiché esse nelle successive chiamate MQI utilizzano la connessione originaria. Questa connessione fa sì che le successive chiamate siano tutte dirette allo stesso Queue Manager. Invece, tutte le applicazioni che si connettono dopo la modifica, si conetteranno al nuovo Queue Manager.

Tutto ciò potrebbe corrispondere al risultato voluto; in ogni caso bisogna tenere presente quanto detto prima di effettuare questa modifica.

### Specificare una coda messaggi non recapitati

La coda messaggi non recapitati è una coda locale a cui vengono inoltrati i messaggi che non riescono a raggiungere la loro corretta destinazione.

#### Attenzione:

È di primaria importanza specificare una coda messaggi non recapitati per ogni Queue Manager della rete. Diversamente gli errori che si verificheranno nei programmi applicativi determineranno l'arresto dei canali o la mancata ricezione delle risposte ai comandi di gestione.

Ad esempio, se un'applicazione tenta di inviare un messaggio ad una coda relativa ad un altro Queue Manager, ma il nome del Queue Manager è errato, il canale viene arrestato e il messaggio rimane nella coda di trasmissione. Le altre applicazioni non potranno quindi utilizzare questo canale per i propri messaggi.

Se invece viene definita una coda messaggi non recapitati per ogni Queue Manager, i canali possono continuare le proprie attività. I messaggi non recapitati verranno semplicemente inoltrati alle code appositamente create all'estremità del canale di ricezione, lasciando liberi il canale e la relativa coda di trasmissione.

Pertanto, quando si crea un Queue Manager è necessario utilizzare il flag `-u` per specificare il nome della coda messaggi non recapitati. È inoltre possibile utilizzare i comandi MQSC per modificare gli attributi di un Queue Manager e specificare la coda messaggi non recapitati da utilizzare. Consultare la sezione "Modifica degli attributi di Queue Manager" a pagina 42 per un esempio del comando MQSC ALTER.

Per elaborare i messaggi che vengono rilevati nella coda messaggi non recapitati, è possibile utilizzare il DLQ (Dead-Letter Queue) handler, fornito con MQSeries. Consultare il "Capitolo 8. Handler di code messaggi non recapitati di MQSeries" a pagina 107 per ulteriori informazioni sul DLQ handler e sui metodi per evitare l'accumulo dei messaggi nella coda messaggi non recapitati.



### Specificare una coda di trasmissione predefinita

Una coda di trasmissione è una coda locale nella quale i messaggi in transito per un Queue Manager remoto vengono accodati in attesa della trasmissione. La coda di trasmissione predefinita è la coda utilizzata tutte le volte che non è definita esplicitamente un'altra coda. Ad ogni Queue Manager può essere assegnata una coda di trasmissione predefinita.

Quando si crea un Queue Manager, utilizzare il flag `-d` per specificare il nome della coda di trasmissione predefinita. Questa operazione non crea direttamente la coda; a questo scopo si dovrà intervenire esplicitamente in seguito. Consultare la sezione "Utilizzo delle code locali" a pagina 47 per ulteriori informazioni.

### Specificare i parametri necessari alla registrazione nei log

È possibile specificare i parametri per la registrazione dei log con il comando `crtmqm`, indicando il tipo di registrazione, il percorso e la dimensione dei file di log. In un ambiente di sviluppo, i parametri predefiniti per la registrazione nei log devono essere adeguati. Tuttavia, è possibile modificare i valori predefiniti, ad esempio:

- La configurazione del sistema non supporta log estesi.
- Si prevede la presenza contemporanea nella coda di una gran quantità di messaggi estesi.

Per ulteriori informazioni relative alla specifica dei parametri di registrazione:

- Per il comando `crtmqm`, consultare la sezione "crtmqm (Crea Queue Manager)" a pagina 258.
- Per l'utilizzo dei file di configurazione, consultare la sezione "La voce Log" a pagina 188.

### Eeguire una copia di riserva dei file di configurazione dopo aver creato un Queue Manager

I file di configurazione da considerare sono:

1. Quando si installa il prodotto viene creato il file di configurazione MQSeries (`mqs.ini`). Esso contiene un elenco di Queue Manager, che viene aggiornato in seguito ad ogni successiva creazione o eliminazione di un Queue Manager. Esiste un file `mqs.ini` per ogni nodo.
2. Quando si crea un nuovo Queue Manager, viene creato automaticamente un nuovo file di configurazione per il Queue Manager (`qm.ini`). Esso contiene i parametri di configurazione per il Queue Manager.

È consigliabile eseguire una copia di riserva di questi file. Se, in seguito, viene creato un nuovo Queue Manager che provoca dei problemi, sarà possibile reinstallare le copie di riserva dei file una volta rimossa la causa del problema. Come norma generale, ogni volta che si crea un nuovo Queue Manager deve sempre eseguirsi una copia di riserva dei file di configurazione.

Per ulteriori informazioni sui file di configurazione, consultare il "Capitolo 13. Configurazione MQSeries" a pagina 179.

---

## Creazione di un Queue Manager predefinito

Utilizzare il comando `crtmqm` per creare un Queue Manager predefinito. Specificando il comando `crtmqm` con un flag `q`:

- Si crea un Queue Manager predefinito denominato `saturn.queue.manager`
- Si creano gli oggetti predefiniti e quelli di sistema

## Creazione dei Queue Manager

- Si specifica sia il nome della relativa coda di trasmissione che quello della relativa coda messaggi non recapitati.

```
crtmqm -q -d MY.DEFAULT.XMIT.QUEUE -u SYSTEM.DEAD.LETTER.QUEUE "saturn.queue.manager"
```

in cui:

**-q** Indica che questo è il Queue Manager predefinito.

**-d MY.DEFAULT.XMIT.QUEUE**  
Indica il nome della coda di trasmissione predefinita.

**-u SYSTEM.DEAD.LETTER.QUEUE**  
Indica il nome della coda messaggi non recapitati.

**"saturn.queue.manager"**  
Indica il nome del Queue Manager. Questo deve essere l'ultimo parametro del comando **crtmqm**.

La creazione di un Queue Manager consente di inviare alcuni comandi (come ad esempio **strmqm** e **runmqsc**) senza dover specificare il nome del Queue Manager. Altri comandi (come ad esempio **endmqm** e **dltmqm**) richiedono che si specifichi il nome del Queue Manager.

Si noti che il nome del Queue Manager riportato in questo esempio è scritto in lettere minuscole e che il tipo di carattere è protetto dalle virgolette. Per ulteriori informazioni sui sistemi di impiego della sensibilità al minuscolo/maiuscolo nei parametri, consultare la sezione "Analisi della sensibilità al minuscolo/maiuscolo" a pagina 23.

---

## Avvio di un Queue Manager

Dopo aver creato un Queue Manager, perché questo possa elaborare comandi o chiamate MQI, è necessario avviarlo. Per avviare il Queue Manager digitare questo comando:

```
strmqm "saturn.queue.manager"
```

Il comando **strmqm** non ha effetto fino a quando il Queue Manager è avviato e pronto a ricevere richieste di connessione.

---

## Rendere predefinito un Queue Manager esistente

Quando si crea un Queue Manager predefinito, il nome viene inserito nella voce *DefaultQueueManager* nel file di configurazione MQSeries (mq5.ini). La voce e i relativi contenuti vengono creati automaticamente se non esistono già.

Potrebbe essere necessario modificare tale voce:

- **Modifica per rendere predefinito un Queue Manager esistente.** A questo scopo è necessario modificare il nome del Queue Manager in questa voce nel nome del nuovo Queue Manager predefinito. Questa operazione non avviene in modo automatico, ma occorre utilizzare un editor di testo.

## Creazione dei Queue Manager

- **Modifica per rendere predefinito un Queue Manager esistente, quando non ne esiste già uno.** A questo scopo è necessario creare la voce `DefaultQueueManager`—con il nome richiesto—personalmente.
- **Modifica accidentale che rende predefinito un Queue Manager e ripristino del Queue Manager predefinito originale.** A questo scopo, modificare la voce `DefaultQueueManager` nel file di configurazione MQSeries sostituendo al nome del Queue Manager modificato per errore, quello del Queue Manager che si desidera sia predefinito.

Consultare il “Capitolo 13. Configurazione MQSeries” a pagina 179 per informazioni relative ai file di configurazione.

Quando sono state immesse nella voce le informazioni richieste, arrestare il Queue Manager e riavviarlo.

---

## Arresto di un Queue Manager

Per arrestare un Queue Manager utilizzare il comando `endmqm`. Ad esempio, per arrestare un Queue Manager denominato `saturn.queue.manager` digitare:

```
endmqm "saturn.queue.manager"
```

## Chiusura ad attività completate

Per impostazione predefinita, il comando `endmqm` esegue un arresto *controllato* o *ad attività completate* del Queue Manager specificato. Ciò può richiedere del tempo una chiusura controllata attende la disconnessione di *tutte* le applicazioni collegate.

Utilizzare questo tipo di arresto per notificare l’arresto alle applicazioni. Se si digita:

```
endmqm -c "saturn.queue.manager"
```

l’utente non viene avvisato dell’arresto di tutte le applicazioni (un comando `endmqm -c "saturn.queue.manager"` è equivalente ad un comando `endmqm "saturn.queue.manager"`).

## Chiusura immediata

Con la chiusura immediata le chiamate MQI attive vengono completate, mentre le nuove chiamate hanno esito negativo. questo tipo di chiusura non attende la disconnessione delle applicazioni dal Queue Manager.

Utilizzare questa chiusura come un normale sistema di arresto del Queue Manager dopo un periodo privo di attività per il Queue Manager. Per una chiusura immediata, digitare:

```
endmqm -i "saturn.queue.manager"
```

## Creazione dei Queue Manager

### Chiusura preventiva

**Attenzione:** non utilizzare questo metodo a meno che tutti i tentativi di arrestare il Queue Manager utilizzando il comando **endmqm** abbiano avuto esito negativo. Questo metodo può avere conseguenze imprevedibili per le applicazioni connesse.

Se l'arresto immediato non ha effetto, è possibile ricorrere ad un arresto *forzato*, specificando il flag -p. Ad esempio:

```
endmqm -p "saturn.queue.manager"
```

In questo modo tutte le code del Queue Manager verranno chiuse.

**Nota:** in seguito ad una chiusura forzata o preventiva oppure in seguito ad un malfunzionamento del Queue Manager, la chiusura avverrà senza la cancellazione della memoria condivisa di pertinenza del Queue Manager. Tale condizione può determinare dei problemi al riavvio. Per informazioni relative all'impiego delle utilità MONMQ per la cancellazione della memoria in seguito a chiusure di questo tipo, consultare la sezione "Gestione della memoria condivisa con MONMQ" a pagina 372.

### Problemi di chiusura del Queue Manager

I problemi di chiusura del Queue Manager sono spesso dovuti alle applicazioni. Ad esempio, le applicazioni potrebbero:

- Non controllare in modo appropriato i codici di errore MQI.
- Non richiedere una notifica di arresto.
- Eseguire la chiusura senza disconnettersi dal Queue Manager (immettendo una chiamata MQDISC).

Se la chiusura di un Queue Manager è troppo lenta o si ritiene che la procedura di arresto non sia stata avviata, è possibile interrompere il comando **endmqm** utilizzando CTRL-Y. Quindi, immettere nuovamente il comando **endmqm**, ma questa volta utilizzare un flag che specifichi o una chiusura immediata o una chiusura preventiva.

Per una descrizione dettagliata del comando **endmqm** e delle relative opzioni, consultare la sezione "endmqm (End queue manager)" a pagina 283.

---

## Riavvio di un Queue Manager

Per riavviare un Queue Manager, utilizzare il comando:

```
strmqm "saturn.queue.manager"
```

---

## Per eliminare un Queue Manager

Per eliminare un Queue Manager, dopo averlo chiuso, utilizzare il seguente comando:

```
dltmqm "saturn.queue.manager"
```

## Creazione dei Queue Manager

**Attenzione:** l'eliminazione di un Queue Manager è un'operazione drastica, poiché con esso vengono eliminate anche tutte le risorse associate. Tra queste vi sono le code e i messaggi, ma anche tutte le definizioni di oggetti.

Per una descrizione del comando **dltmqm** e delle relative opzioni, consultare la sezione "dltmqm (Delete queue manager)" a pagina 263. Assicurarsi che solo gli amministratori accreditati abbiano l'autorizzazione necessaria ad utilizzare questo comando.



---

## Capitolo 4. Gestione di oggetti MQSeries locali

Questo capitolo descrive come gestire gli oggetti locali MQSeries in modo da supportare programmi applicativi che utilizzano MQI (Message Queuing Interface). In questo contesto, con gestione locale si intende la creazione, la visualizzazione, le modifiche, la copia e l'eliminazione di oggetti MQSeries.

Questo capitolo contiene le seguenti sezioni:

- "Supporto dei programmi applicativi che utilizzano MQI"
- "Esecuzione delle attività di gestione locale utilizzando i comandi MQSC" a pagina 38
- "Esecuzione di comandi MQSC da file di testo" a pagina 42
- "Risoluzione dei problemi con MQSC" a pagina 45
- "Utilizzo delle code locali" a pagina 47
- "Utilizzo di alias di code" a pagina 56
- "Utilizzo delle code di esempio" a pagina 57
- "Gestione degli oggetti per triggering" a pagina 59

---

### Supporto dei programmi applicativi che utilizzano MQI

I programmi applicativi necessitano di alcuni oggetti per poter essere eseguiti. Ad esempio, la Figura 1 illustra un'applicazione che rimuove i messaggi da una coda, li elabora e poi invia i risultati ad un'altra coda dello stesso Queue Manager.

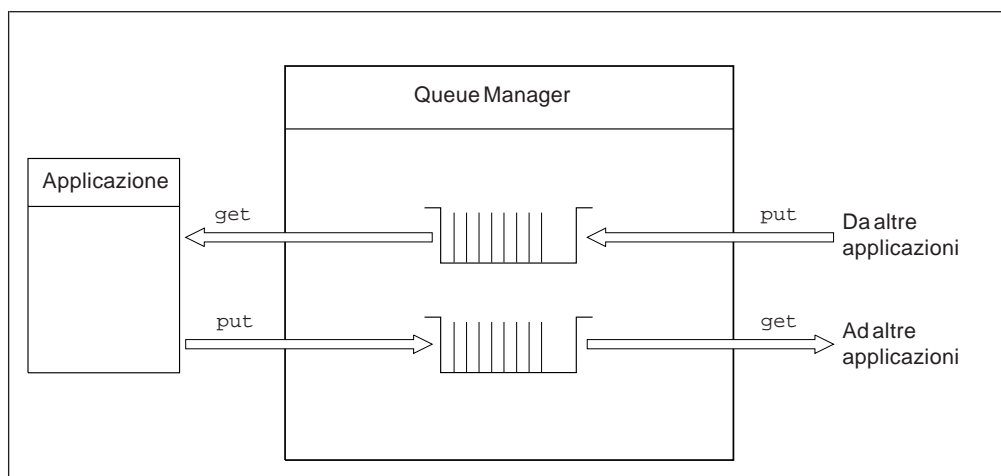


Figura 1. Code, messaggi e applicazioni

Le applicazioni possono inviare messaggi (utilizzando MQPUT) sia su code locali che remote, mentre possono riceverne solo da code locali (utilizzando MQGET).

Prima di poter eseguire questa applicazione, devono essere soddisfatte le seguenti condizioni:

- Deve esistere un Queue Manager e deve essere in esecuzione.
- La prima coda dell'applicazione, quella da cui rimuovere i messaggi, deve essere definita.
- La seconda coda, quella a cui inviare i messaggi, deve essere definita.

## Programmi applicativi

- L'applicazione deve potersi connettere al Queue Manager. A questo scopo occorre il collegamento al codice del prodotto. Consultare il manuale *MQSeries Application Programming Guide* per ulteriori informazioni.
- Le applicazioni che inviano i messaggi alla prima coda devono anche connettersi al Queue Manager. Se si tratta di code remote, occorre impostare anche le code di trasmissione e i canali. Questa parte del sistema non è illustrata nella Figura 1 a pagina 37.

---

## Esecuzione delle attività di gestione locale utilizzando i comandi MQSC

In questa sezione, si presume che l'immissione di comandi avvenga utilizzando il comando **runmqsc**. È possibile far ciò interattivamente immettendo i comandi con la tastiera oppure è possibile reindirizzare SYS\$INPUT per eseguire una sequenza di comandi da un file di testo ASCII. In entrambi i casi, il formato dei comandi è lo stesso.

Il manuale *MQSeries - Guida di riferimento per i comandi* contiene una descrizione di ogni comando MQSC e delle relative sintassi.

È possibile utilizzare i comandi di scrittura MQSeries (MQSC) per gestire gli oggetti del Queue Manager, incluso lo stesso Queue Manager, cluster, canali, code, namelist e processi di definizione. Questa sezione si occupa di Queue Manager, di code e di definizioni di processo; per informazioni relative alla gestione di oggetti canale, consultare l'implementazione DQM nel manuale *MQSeries Intercommunication*.

È possibile immettere i comandi MQSC utilizzando il comando **runmqsc**, in modo interattivo, immettendo cioè i comandi dalla tastiera, oppure reindirizzando un input standard per eseguire una sequenza di comandi da un file di testo ASCII. In entrambi i casi, il formato dei comandi è lo stesso.

È possibile eseguire il comando **runmqsc** in tre modi, in funzione dei flag impostati nel comando:

- *Verification mode (Modo Verifica)*, in cui i comandi MQSC vengono verificati su un Queue Manager locale, ma non vengono eseguiti.
- *Direct mode (Modo Diretto)*, in cui i comandi MQSC vengono eseguiti su un Queue Manager locale.
- *Indirect mode (Modo Indiretto)*, in cui i comandi vengono eseguiti su un Queue Manager remoto.

Gli attributi degli oggetti specificati in MQSC vengono riportati in lettere maiuscole (ad esempio, RQMNAME), sebbene tale caratteristica non abbia rilevanza. Per ulteriori informazioni sulla sensibilità al minuscolo/maiuscolo, consultare la sezione "Sensibilità al minuscolo/maiuscolo nei comandi MQSC" a pagina 25. I nomi degli attributi MQSC hanno una lunghezza massima di otto caratteri.

## Prima di iniziare

Prima di eseguire i comandi MQSC, è necessario creare ed avviare il Queue Manager su cui si dovranno eseguire i comandi, consultare la sezione "Creazione di un Queue Manager predefinito" a pagina 31.

### Nomi degli oggetti MQSeries

Negli esempi, vengono utilizzati dei nomi lunghi per gli oggetti. Ciò per semplificare l'identificazione degli oggetti che si utilizzano.



## Immissione di comandi MQSC

Quando si immettono comandi MQSC, è necessario specificare solo il nome locale della coda. Negli esempi vengono utilizzati nomi quali:

```
ORANGE.LOCAL.QUEUE
```

La parte del nome LOCAL.QUEUE serve ad indicare che si tratta di una coda locale. *Non* è necessario inserirla nei nomi delle code locali.

Inoltre viene utilizzato il nome saturn.queue.manager come nome di un Queue Manager.

La parte del nome queue.manager serve ad indicare che l'oggetto in questione è un Queue Manager. *Non* è necessario inserirla nei nomi dei Queue Manager.

Non è obbligatorio l'utilizzo di questi nomi, ma se si ricorre ad altri nomi, è necessario modificare tutti i comandi degli esempi che si riferiscono a questi nomi.

### Reindirizzamento di input e output

Per semplificare la migrazione degli altri sistemi operativi verso OpenVMS, MQSeries supporta lo stile UNIX<sup>®</sup> degli indicatori di reindirizzamento per sys\$input, sys\$output, e sys\$error del tipo seguente:

- < specifica il sorgente per SYS\$INPUT
- > specifica il sorgente per SYS\$OUTPUT
- 2> specifica il sorgente per SYS\$error

Questa funzione è inclusa anche nella versione eseguibile dei programmi esempio. Non è tuttavia incluso nel sorgente degli esempi e, pertanto, non sarà disponibile se si rigenerano gli esempi dal codice sorgente.

## Utilizzo interattivo della funzione MQSC

Per inserire comandi in modo interattivo, alla richiesta di comandi DCL, digitare:

```
runmqsc
```

In questo comando non è specificato un nome di Queue Manager, pertanto i comandi MQSC verranno elaborati dal Queue Manager predefinito. Ora è possibile digitare qualsiasi comando MQSC. Ad esempio:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE)
```

Per indicare che un comando si estende su più linee, devono essere utilizzati i caratteri di continuazione:

- Un segno meno (-) indica che il comando continua dall'inizio della linea seguente.
- Un segno più (+) indica che il comando continua dal primo carattere presente nella linea seguente.

L'input del comando termina con il carattere finale di una linea non vuota, purché non sia un carattere di continuazione. La fine dell'input di comando può essere anche reso esplicito inserendo un "punto e virgola" (;). Ciò è particolarmente utile se accidentalmente viene immesso un carattere di continuazione al termine dell'ultima linea di un input di comando.

## Immissione di comandi MQSC

### Feedback dei comandi MQSC

Quando si immettono comandi con le funzioni MQSC, il Queue Manager riporta dei messaggi per l'operatore che confermano le attività o rendono informazioni in merito agli errori commessi. Ad esempio:

```
AMQ8006: MQSeries queue created
.
.
.
AMQ8405: Syntax error detected at or near end of command segment below:-
z

AMQ8426: Valid MQSC commands are:

ALTER
CLEAR
DEFINE
DELETE
DISPLAY
END
PING
REFRESH
RESET
RESOLVE
RESUME
START
STOP
SUSPEND
```

Il primo messaggio conferma la creazione di un Queue Manager; il secondo informa che è stato commesso un errore di sintassi. Questi messaggi sono inviati al dispositivo output standard. Se i comandi non sono stati inseriti correttamente, consultare il manuale *MQSeries - Guida di riferimento per i comandi* per la sintassi corretta.

### Termine dell'input interattivo MQSC

Per terminare l'immissione interattiva di comandi MQSC, digitare il comando MQSC END:

```
END
```

In alternativa, è possibile digitare i caratteri EOF <CTRL Z>.

Questa operazione non è necessaria se si sta eseguendo il reindirizzo di input da altri fonti, come un file di testo.

### Visualizzazione degli attributi Queue Manager

Per visualizzare gli attributi di Queue Manager specificati con il comando **runmqsc**, utilizzare il seguente comando MQSC:

```
DISPLAY QMGR ALL
```

Nella Figura 2 viene visualizzato un tipico output.

```

1 : display qmgr all
AMQ8408: Display Queue Manager details.
DESCR( )                DEADQ( )
DEFXMITQ( )             CHADEXIT( )
CLWLEXIT( )             CLWLDATA( )
REPOS( )                REPOSNL( )
COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)  QMNAME(saturn.queue.manager)
CRDATE(2001-01-16)     CRTIME(11.13.56)
ALTDATA(2001-01-16)   ALTTIME(11.13.56)
QMID(saturn.queue.manager_2001-01-16_11.13.56)
TRIGINT(999999999)    MAXHANDS(256)
MAXUMSGS(10000)       AUTHOREV(DISABLED)
INHIBTEV(DISABLED)   LOCALEV(DISABLED)
REMOTEEV(DISABLED)   PERFMEV(DISABLED)
STRSTPEV(ENABLED)    CHAD(DISABLED)
CHADEV(DISABLED)     CLWLLEN(100)
MAXMSGL(4194304)     CCSID(819)
MAXPRTY(9)           CMDLEVEL(510)
PLATFORM(OpenVMS)    SYNCPT
DISTL(YES)

```

Figura 2. Tipico output di un comando DISPLAY QMGR

Il parametro ALL del comando DISPLAY QMGR determina la visualizzazione di tutti gli attributi del Queue Manager. In particolare, poiché nel comando non è stato specificato alcun nome di Queue Manager, l'output riporta il nome del Queue Manager predefinito (saturn.queue.manager), i nomi della coda messaggi non recapitati (SYSTEM.DEAD.LETTER.QUEUE) e della coda comandi (SYSTEM.ADMIN.COMMAND.QUEUE).

Prima di procedere, confermare la creazione di queste code digitando il comando:

```
DISPLAY QUEUE (SYSTEM.*)
```

In questo modo viene visualizzato un elenco di code che riportano l'estensione 'SYSTEM.\*'. Le parentesi sono necessarie.

### Utilizzo di un Queue Manager diverso dal predefinito

È possibile specificare il nome del Queue Manager nel comando `runmqsc` in modo da eseguire comandi MQSC su un Queue Manager locale diverso dal predefinito. Ad esempio, per eseguire comandi MQSC sul Queue Manager `jupiter.queue.manager`, utilizzare il comando:

```
runmqsc "jupiter.queue.manager"
```

Dopo di che, tutti i comandi MQSC digitati verranno elaborati da questo Queue Manager purché sia sullo stesso nodo che in esecuzione.

È inoltre possibile eseguire comandi MQSC su un Queue Manager remoto; consultare la sezione "Emissione remota di comandi MQSC" a pagina 75.

## Immissione di comandi MQSC

### Modifica degli attributi di Queue Manager

Per modificare gli attributi del Queue Manager specificati nel comando **runmqsc**, utilizzare il comando ALTER QMGR, specificando gli attributi e i valori che si desidera modificare. Ad esempio, utilizzare i seguenti comandi per modificare gli attributi del Queue Manager `jupiter.queue.manager`:

```
runmqsc "jupiter.queue.manager"  
ALTER QMGR DEADQ (ANOTHERDLQ) INHIBTEV (ENABLED)
```

Il comando ALTER QMGR modifica la coda messaggi non recapitati utilizzata e abilita eventi inibitori.

---

### Esecuzione di comandi MQSC da file di testo

L'esecuzione di comandi in modo interattivo è ideale per controlli brevi; se invece si ha la necessità di inserire comandi molto lunghi o sequenze di comandi che devono essere ripetute, è opportuno fornire gli input da un file di testo. Consultare la sezione "Reindirizzo di input e output" a pagina 39 per informazioni relative al reindirizzo di indicatori. Per utilizzare i file di testo, occorre prima crearne uno che contenga i comandi MQSC utilizzando un editor di testo. Ad esempio, il seguente comando esegue una sequenza di comandi contenuta nel file di testo `myprog.in`:

```
runmqsc < myprog.in
```

In modo analogo, è possibile reindirizzare l'output ad un file. Un file che contiene i comandi MQSC per l'input viene definito *file di comandi MQSC*. Il file di output che contiene le risposte del Queue Manager viene definito *file di notifica*.

Per reindirizzare sia `SYSS$INPUT` che `SYSS$OUTPUT` sul comando **runmqsc**, utilizzare questo formato del comando:

```
runmqsc < myprog.in > myprog.out
```

Questo comando richiama i comandi MQSC contenuti nel file di comandi MQSC `myprog.in`. Dal momento che non è stato specificato il nome del Queue Manager, i comandi MQSC verranno eseguiti sul Queue Manager predefinito. L'output è inviato al file di notifica `myprog.out`. La Figura 3 a pagina 43 illustra un estratto dal file di comandi MQSC `myprog.in` e la Figura 4 a pagina 44 illustra l'estratto del corrispondente file di notifica `myprog.out`.

Per reindirizzare sia `SYSS$INPUT` che `SYSS$OUTPUT` sul comando **runmqsc**, in relazione al Queue Manager diverso dal predefinito (`saturn.queue.manager`), utilizzare il seguente formato di comando:

```
runmqsc "saturn.queue.manager" < myprog.in > myprog.out
```

## File di comandi MQSC

I comandi MQSC sono scritti in forma leggibile, cioè in formato testo ASCII. La Figura 3 è un estratto da un file di comandi MQSC che illustra il comando DEFINE QLOCAL e i relativi attributi. Il manuale *MQSeries - Guida di riferimento per i comandi* contiene una descrizione di ogni comando MQSC e delle relative sintassi.

```

.
.
.
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) REPLACE +
DESCR(' ') +
PUT(ENABLED) +
DEFPRTY(0) +
DEFPSIST(NO) +
GET(ENABLED) +
MAXDEPTH(5000) +
MAXMSGL(1024) +
DEFSOPT(SHARED) +
NOHARDENBO +
USAGE(NORMAL) +
NOTRIGGER
.
.
.

```

Figura 3. Estratto dal file di comandi MQSC, *myprog.in*

Per la flessibilità d'uso negli ambienti MQSeries, si raccomanda di limitare la lunghezza delle linee a 72 caratteri nei file di comandi MQSC. Il segno più (+) indica che il comando continua nella riga successiva.

Per MQSeries per Compaq OpenVMS, il limite massimo per linea è di 80 caratteri, incluso il carattere di continuazione. Il segno più (+) indica che il comando continua sulla riga successiva.

## Notifiche MQSC

Il comando **runmqsc** restituisce una *notifica*, che viene inviata a SYS\$OUTPUT. La notifica contiene:

- Un'intestazione che identifica MQSC come il sorgente della notifica:  
Starting MQSeries Commands.
- Un elenco numerato dei comandi MQSC immessi (opzionale). Per impostazione predefinita, il testo dell'input viene riportato all'output. Nell'output, ogni comando è ordinato in base ad una sequenza numerica, così come mostrato nella Figura 4 a pagina 44. Tuttavia è possibile utilizzare il flag **-e** nel comando **runmqsc** per eliminare l'output.
- Un messaggio di errore di sintassi per ogni comando in cui viene rilevato un errore.
- Un *messaggio per l'operatore* che indica il risultato dell'esecuzione di ogni comando. Ad esempio, il messaggio per l'operatore in cui viene indicato il completamento con esito positivo di un comando DEFINE QLOCAL è:  
AMQ8006: MQSeries queue created.
- Altri messaggi dovuti ad errori durante l'esecuzione del file di scrittura.

## Esecuzione di comandi MQSC

- Una chiara sintesi statistica della notifica che indica il numero di comandi letti, il numero di comandi con errori di sintassi e il numero di comandi che non è stato possibile elaborare.

**Nota:** il queue Manager inizia l'elaborazione dei soli comandi che non presentano errori di sintassi.

```
Starting MQSeries Commands.
.
.
12:   DEFINE QLOCAL('RED.LOCAL.QUEUE') REPLACE +
:     DESCR(' ') +
:     PUT(ENABLED) +
:     DEFPRTY(0) +
:     DEFPSIST(NO) +
:     GET(ENABLED) +
:     MAXDEPTH(5000) +
:     MAXMSGL(1024) +
:     DEFSOPT(SHARED) +
:     USAGE(NORMAL) +
:     NOTRIGGER
AMQ8006: MQSeries queue created.
:
.
.
15 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.
```

Figura 4. Estratto dal file di notifica MQSC, myprog.out.

## Esecuzione dei file di comandi MQSC forniti

Quando si installa MQSeries per Compaq OpenVMS, vengono forniti i seguenti file di comandi MQSC:

### amqscos0.tst

Definizione di oggetti utilizzati dai programmi di esempio.

Il file è memorizzato nella directory MQS\_EXAMPLES:

## Utilizzo di runmqsc per la verifica dei comandi

È possibile utilizzare il comando **runmqsc** per verificare i comandi MQSC su un Queue Manager locale senza doverli necessariamente eseguire. A questo scopo, impostare il flag **-v** nel comando **runmqsc**, ad esempio:

```
runmqsc -v < myprog.in > myprog.out
```

Quando si richiama **runmqsc** con un file di comandi MQSC, il Queue Manager verifica ogni comando e restituisce una notifica senza eseguire i comandi MQSC. In questo modo è possibile controllare la sintassi di tutti i comandi del file di comandi. Ciò è particolarmente importante se si eseguono molti comandi da un file di comandi.

Questa notifica è simile a quella riportata nella Figura 4.

## Esecuzione di comandi MQSC

Non è possibile utilizzare questo metodo per verificare i comandi MQSC da una postazione remota. Ad esempio, se si immette questo comando:

```
runmqsc -w 30 -v "jupiter.queue.manager" < myprog.in > myprog.out
```

L'indicatore `-w`, che viene utilizzato per indicare che il queue manager è remoto, viene ignorato e il comando viene eseguito in locale in modalità verifica (verification mode).

---

## Risoluzione dei problemi con MQSC

Se non si riesce ad ottenere i comandi MQSC da eseguire, utilizzare il seguente elenco di controllo per riscontrare l'eventuale presenza di uno di questi frequenti problemi. La lettura dell'errore generato non offre sempre conclusioni univoche.

Quando si utilizza il comando **runmqsc**, ricordarsi di:

- Utilizzare il simbolo `<` qualora si reindirizzi un input da un file. Se tale simbolo viene omissso, il Queue Manager interpreta il nome del file come un nome di Queue Manager ed inoltra il seguente messaggio di errore:

```
AMQ8118: MQSeries queue manager does not exist.
```

- Se si reindirizza un output ad un file, utilizzare il simbolo `>`. Per impostazione predefinita, l'output è rivolto alla directory dalla quale viene eseguito il comando **runmqsc**. Specificare un nome di file valido per inviare l'output ad uno specifico file di una specifica directory.
- Verificare di aver creato il Queue Manager che dovrà eseguire i comandi. Effettuare questo controllo nel file di configurazione `mqsc.ini`, che per impostazione predefinita è memorizzato nella directory `MQS_ROOT:[MQM]`. Questo file contiene i nomi dei Queue Manager e il nome del Queue Manager predefinito, se impostato.
- Il Queue Manager deve essere già avviato. Se non lo è già, avviarlo; consultare "Avvio di un Queue Manager" a pagina 32. Si riceve un messaggio di errore nel caso sia già avviato.
- Specificare un nome di Queue Manager nel comando **runmqsc** se non si è definito un Queue Manager predefinito, altrimenti si riceve il seguente errore:

```
AMQ8146: MQSeries queue manager not available.
```

Per risolvere questo tipo di problema, consultare la sezione "Rendere predefinito un Queue Manager esistente" a pagina 32.

- Non è possibile specificare un comando MQSC come parametro **runmqsc**. Ad esempio, la seguente stringa non è valida:

```
runmqsc DEFINE QLOCAL(FRED)
```

## Problemi con MQSC

- Non è possibile inserire comandi MQSC da DCL prima di aver immesso il comando **runmqsc**. Ad esempio:

```
DEFINE QLOCAL(Queue1)
```

```
%DCL-W-PARMDCL, invalid parameter delimiter - check use of special characters
```



- Non è possibile eseguire comandi di controllo da `runmqsc`. Ad esempio, non è possibile avviare un Queue Manager una volta avviata interattivamente l'esecuzione MQSC:

```
$ runmqsc
0790997, 5724-A38 (C) Copyright IBM Corp. 1996, 2001 ALL RIGHTS RESERVED.
Starting MQSeries Commands.

strmqm saturn.queue.manager
  1 : strmqm saturn.queue.manager
AMQ8405: Syntax error detected at or near end of command segment below:-
s

AMQ8426: Valid MQSC commands are:

    ALTER
    CLEAR
    DEFINE
    DELETE
    DISPLAY
    END
    PING
    REFRESH
    RESET
    RESOLVE
    RESUME
    START
    STOP
    SUSPEND

*CANCEL*

One MQSC command read.
One command has a syntax error.
All valid MQSC commands were processed.
$
```

Consultare anche la sezione “Eventuali problemi nell’utilizzo remoto di MQSC” a pagina 77.

---

## Utilizzo delle code locali

Questa sezione contiene esempi dell’utilizzo di alcuni comandi MQSC. Fare riferimento al manuale *MQSeries - Guida di riferimento per i comandi* per una descrizione completa di questi comandi.

### Definizione di una coda locale

Per un’applicazione, il Queue Manager locale è quello a cui è connessa l’applicazione. Le code gestite dal Queue Manager locale vengono definite locali rispetto a questo.

Utilizzare il comando `DEFINE QLOCAL` per creare una definizione di una coda locale ed anche la struttura dei dati della coda. È possibile anche modificare le caratteristiche della coda dalle code locali predefinite.

Nell’esempio che segue, le specifiche della coda denominata `ORANGE.LOCAL.QUEUE` determinano queste caratteristiche:

## Utilizzo delle code locali

- Abilitata al ricevimento, non abilitata all'invio e funzionante su base FIFO (First-In-First-Out).
- Si tratta di una coda 'normale', nel senso che non è una coda di iniziazione o una coda di trasmissione e non genera messaggi trigger.
- La capacità massima della coda è 1000 messaggi; la dimensione massima dei messaggi è 2000 byte.

Osservare il seguente comando MQSC:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) +
  DESCR('Queue for messages from other systems') +
  PUT (DISABLED) +
  GET (ENABLED) +
  NOTRIGGER +
  MSGDLVSQ (FIFO) +
  MAXDEPTH (1000) +
  MAXMSGL (2000) +
  USAGE (NORMAL)
```

### Note:

1. Molti di questi attributi, così come forniti col prodotto, sono preimpostati. Tuttavia sono riportati in figura a scopo dimostrativo. È possibile omettere questi attributi se si è certi che i valori predefiniti corrispondano alle proprie esigenze e che non siano stati modificati. Consultare anche la sezione "Visualizzazione degli attributi predefiniti di oggetti" a pagina 49.
2. USAGE (NORMAL) indica che la coda non è una coda di trasmissione.
3. Se sullo stesso Queue Manager già esiste una coda locale denominata ORANGE.LOCAL.QUEUE, questo comando avrà esito negativo. Utilizzare l'attributo REPLACE, se si desidera sostituire questa definizione di coda a quella già esistente, ma consultare anche la sezione "Modifica degli attributi di coda locale" a pagina 50.

## Definizione di una coda messaggi non recapitati

Ogni Queue Manager deve avere una coda locale da utilizzare come coda messaggi non recapitati, in modo che questi messaggi che non riescono a raggiungere la loro corretta destinazione, possano essere memorizzati per un futuro ripristino. La coda messaggi non recapitati deve essere specificata esplicitamente al Queue Manager. Ciò è possibile utilizzando il comando **crtmqm** oppure il comando ALTER QMGR per specificarne una successivamente. La coda messaggi non recapitati deve essere specificata prima di poterla utilizzare.

Un coda messaggi non recapitati di esempio, denominata SYSTEM.DEAD.LETTER.QUEUE, è fornita con il prodotto. Questa coda viene creata automaticamente quando si esegue l'esempio. Se necessario, è possibile modificare la definizione. Non è necessario, tuttavia, modificare il nome.

Una coda messaggi non recapitati non ha alcuna particolarità, eccetto:

- Deve essere una coda locale.
- Il relativo attributo MAXMSGL (Maximum Message Length) deve consentire la memorizzazione dei messaggi più estesi gestiti dal Queue Manager **più** lo spazio occupato dall'intestazione del messaggio non recapitato (MQDLH).

MQSeries fornisce una coda messaggi non recapitati facilmente gestibile che consente di specificare il modo in cui elaborare o rimuovere i messaggi rilevati sulla coda messaggi non recapitati. Per ulteriori informazioni, consultare il manuale "Capitolo 8. Handler di code messaggi non recapitati di MQSeries" a pagina 107.

### Visualizzazione degli attributi predefiniti di oggetti

Quando si definisce un oggetto MQSeries, tutti gli attributi non specificati vengono copiati dal corrispondente oggetto predefinito. Ad esempio, quando si definisce una coda locale, essa eredita tutti gli attributi che vengono omessi nella definizione dalla coda locale predefinita, denominata SYSTEM.DEFAULT.LOCAL.QUEUE. Per esaminare questi attributi, utilizzare il comando:

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE) ALL
```

**Nota:** la sintassi di questo comando è diversa da quella del corrispondente comando DEFINE.

È possibile scegliere gli attributi da visualizzare specificandoli individualmente. Ad esempio:

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +  
    MAXDEPTH +  
    MAXMSGL +  
    CURDEPTH
```

Questo comando visualizza i tre attributi specificati:

```
AMQ8409: Display Queue details.  
QUEUE(ORANGE.LOCAL.QUEUE)  
MAXDEPTH(1000)  
MAXMSGL(2000)  
CURDEPTH(0)
```

CURDEPTH rappresenta la capacità della coda e, cioè, il numero massimo di messaggi che la coda può contenere. È utile visualizzare questo attributo poiché, controllando la capacità della coda, si può evitare il riempimento della coda.

### Copia della definizione di una coda locale

È possibile copiare la definizione di una coda utilizzando l'attributo LIKE con il comando DEFINE. Ad esempio:

```
DEFINE QLOCAL (MAGENTA.QUEUE) +  
    LIKE (ORANGE.LOCAL.QUEUE)
```

Questo comando crea una coda con gli stessi attributi della coda ORANGE.LOCAL.QUEUE, invece di quelli della coda locale predefinita dal sistema.

## Utilizzo delle code locali

È possibile utilizzare questo formato del comando DEFINE per copiare una definizione di coda e sostituire uno o più modifiche agli attributi della coda di origine. Ad esempio:

```
DEFINE QLOCAL (THIRD.QUEUE) +  
  LIKE (ORANGE.LOCAL.QUEUE) +  
  MAXMSGL(1024)
```

Questo comando copia gli attributi della coda ORANGE.LOCAL.QUEUE nella coda THIRD.QUEUE, ma specifica come capacità massima per i messaggi della nuova coda il valore di 1024 byte, invece di 2000.

### Note:

1. Quando si utilizza l'attributo LIKE con il comando DEFINE, si esegue la copia dei soli attributi della coda. Non è in corso la copia dei messaggi presenti nella coda.
2. Se si definisce una coda locale, senza specificare LIKE, è come se si utilizzasse DEFINE LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE).

## Modifica degli attributi di coda locale

Gli attributi di coda possono essere modificati in due modi: utilizzando il comando ALTER QLOCAL o il comando DEFINE QLOCAL con l'attributo REPLACE. Nella sezione "Definizione di una coda locale" a pagina 47 viene definita la coda ORANGE.LOCAL.QUEUE. Si ipotizzi di voler aumentare la dimensione massima dei messaggi su questa coda fino a 10 000 byte; ciò è possibile:

- Utilizzando il comando ALTER:

```
ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
```

Questo comando modifica un singolo attributo, quello della dimensione massima dei messaggi; tutti gli altri attributi rimangono invariati.

- Utilizzando il comando DEFINE con l'opzione REPLACE, ad esempio:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
```

Questo comando modifica non solo la dimensione massima dei messaggi, ma anche tutti gli altri attributi, a cui vengono assegnati i valori predefiniti. La coda ora è abilitata all'invio, laddove prima tale attributo era di valore opposto. L'abilitazione all'invio è il valore predefinito, così come specificato dalla coda SYSTEM.DEFAULT.LOCAL.QUEUE, se non modificata.

Se si *diminuisce* la dimensione massima dei messaggi su una coda esistente, la modifica non riguarderà i messaggi esistenti. Tutti i nuovi messaggi, invece, devono corrispondere alle nuove impostazioni.

## Annullamento di una coda locale

Per eliminare tutti i messaggi presenti su una coda locale denominata MAGENTA.QUEUE, utilizzare il seguente comando:

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

Non è possibile annullare la coda se:

- Alla coda sono stati inviati messaggi di cui non è stato eseguito il commit sottoposti a syncpoint.
- La coda è stata appena aperta da un'applicazione.

## Utilizzo delle code locali

### Eliminazione di una coda locale

Utilizzare il comando DELETE QLOCAL per eliminare una coda. Una coda non può essere eliminata se vi sono messaggi su cui non è stato eseguito il commit. In ogni caso, anche se sono presenti solo messaggi di cui è già stato eseguito il commit, sarà possibile eliminarla solo se viene specificata l'opzione PURGE. Ad esempio:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

La specifica dell'opzione NOPURGE, piuttosto che PURGE, verifica che non sia stata eliminata la coda, nel caso in cui essa contenga messaggi su cui è stato eseguito il commit.

### Esame delle code

Se si ha la necessità di esaminare il contenuto di messaggi di una coda, MQSeries per OpenVMS fornisce un browser di esempio utile a questo scopo. Il browser è fornito sia come sorgente che modulo eseguibile. Per impostazione predefinita, i nomi dei file e i percorsi sono:

**Sorgente**

```
MQS_EXAMPLES:AMQSBG0.C
```

**Modulo eseguibile**

```
[.BIN]AMQSBG.EXE, in
```

```
MQS_EXAMPLES:
```

.

L'esempio prevede due parametri e cioè:

- Il nome della coda, ad esempio SYSTEM.ADMIN.RESPQ.TEST.
- Il nome del Queue Manager, ad esempio JJJH

così come mostrato nel comando seguente:

```
amqsbcg "SYSTEM.ADMIN.RESPQ.TEST" "JJJH"
```

Non esistono valori predefiniti; entrambi i parametri sono necessari. Nella Figura 5 a pagina 53 vengono mostrati dei risultati tipici di questi comandi.









### Utilizzo di alias di coda

Un alias di coda consente di reindirizzare le chiamate MQI. Un alias di coda non è una coda reale, ma una definizione che si riferisce ad una coda reale. La definizione di alias di coda contiene un nome di coda destinatario che viene specificato dall'attributo TARGQ (*BaseQName* in PCF). Quando un'applicazione specifica un alias di coda in una chiamata MQI, il Queue Manager, al momento dell'esecuzione, si riferisce al nome della coda reale.

Ad esempio, un'applicazione deve inviare un messaggio alla coda denominata MY.ALIAS.QUEUE. Essa specifica il nome di questa coda quando esegue una richiesta MQOPEN e, indirettamente, quando invia un messaggio a questa coda. L'applicazione non rileva che si tratta di un alias di coda. Per ciascuna chiamata MQI che utilizza questo alias, il Queue Manager si riferisce al nome della coda reale, che può essere sia una coda locale che remota rispetto a questo Queue Manager.

Modificando il valore dell'attributo TARGQ, è possibile reindirizzare le chiamate MQI verso un'altra coda, anche su un altro Queue Manager. Ciò è utile nella gestione, nella migrazione e nel bilanciamento del carico di lavoro.

### Definizione di un alias di coda

Il seguente comando crea un alias di coda:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGQ (YELLOW.QUEUE)
```

Questo comando reindirizza le chiamate MQI che specificano MY.ALIAS.QUEUE, verso la coda YELLOW.QUEUE. Il comando non crea la coda di destinazione; le chiamate MQI hanno esito negativo se la coda YELLOW.QUEUE non esiste al momento dell'esecuzione.

Se si modifica la definizione dell'alias, è possibile reindirizzare le chiamate MQI verso un'altra coda. Ad esempio:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGQ (MAGENTA.QUEUE) REPLACE
```

Questo comando reindirizza le chiamate MQI verso un'altra coda, MAGENTA.QUEUE.

Inoltre è possibile utilizzare gli alias di coda per far sì che una singola coda sembri avere diversi attributi per diverse applicazioni. Per far ciò basta definire due alias, uno per ogni applicazione. Si ipotizzino le seguenti due applicazioni:

- L'applicazione ALPHA può inviare i messaggi alla coda YELLOW.QUEUE, ma non può riceverne da questa coda.
- L'applicazione BETA può ricevere messaggi dalla coda YELLOW.QUEUE, ma non può inviarvene.

A questo scopo è sufficiente utilizzare i seguenti comandi:

```
* This alias is put enabled and get disabled for application ALPHA

DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
  TARGQ (YELLOW.QUEUE) +
  PUT (ENABLED) +
  GET (DISABLED)

* This alias is put disabled and get enabled for application BETA

DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
  TARGQ (YELLOW.QUEUE) +
  PUT (DISABLED) +
  GET (ENABLED)
```

ALPHA utilizza il nome di coda ALPHAS.ALIAS.QUEUE nelle relative chiamate MQI; BETA utilizza il nome di coda BETAS.ALIAS.QUEUE. Entrambe hanno accesso alla stessa coda, ma in modi diversi.

È possibile utilizzare gli attributi LIKE e REPLACE quando si definiscono gli alias di code, nello stesso modo in cui si utilizzano tali attributi nella definizione di code locali.

## Utilizzo di altri comandi con gli alias di coda

È possibile utilizzare gli appropriati comandi MQSC per visualizzare o modificare gli attributi dell'alias di coda o per eliminare gli oggetti dell'alias di coda. Ad esempio:

```
* Display the queue alias' attributes
* ALL = Display all attributes

DISPLAY QUEUE (ALPHAS.ALIAS.QUEUE) ALL

* ALTER the base queue name, to which the alias resolves.
* FORCE = Force the change even if the queue is open.

ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGQ(ORANGE.LOCAL.QUEUE) FORCE

* Delete this queue alias, if you can.

DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
```

Non è possibile eliminare un alias di coda se, ad esempio, un'applicazione ha aperto di recente la coda o una coda che richiama questa coda. Consultare il manuale *MQSeries - Guida di riferimento per i comandi* per ulteriori informazioni su questo e sugli altri comandi dell'alias di coda.

---

## Utilizzo delle code di esempio

Un Queue Manager crea una *coda dinamica* se riceve una chiamata MQI da un'applicazione che specifica un nome di coda definito come coda di esempio. Il nome della nuova coda dinamica è generato dal Queue Manager al momento della sua creazione. Una *coda di esempio* è un modello che specifica gli attributi di ogni coda dinamica creata.

## Utilizzo delle code di esempio

Le code di esempio forniscono un metodo utile alle applicazioni per creare code in base alle necessità.

### Definizione di una coda di esempio

Per definire le code di esempio occorre utilizzare un gruppo di attributi, così come avviene per le code locali. Le code di esempio e le code locali hanno lo stesso gruppo di attributi; tuttavia con le code di esempio è possibile specificare se le code dinamiche create devono essere temporanee o permanenti. Le code temporanee vanno perse quando viene riavviato il Queue Manager piuttosto che quelle permanenti. Ad esempio:

```
DEFINE QMODEL (GREEN.MODEL.QUEUE) +
  DESCR('Queue for messages from application X') +
  PUT (DISABLED) +
  GET (ENABLED) +
  NOTRIGGER +
  MSGDLVSQ (FIFO) +
  MAXDEPTH (1000) +
  MAXMSGL (2000) +
  USAGE (NORMAL) +
  DEFTYPE (PERDYN)
```

Questo comando crea una definizione di coda di esempio. In base all'attributo DEFTYPE, le code attualmente create da questo modello sono code dinamiche permanenti.

**Nota:** gli attributi non specificati vengono copiati automaticamente dalla coda predefinita SYSYTEM.DEFAULT.MODEL.QUEUE.

È possibile utilizzare gli attributi LIKE e REPLACE quando si definiscono code di esempio, nello stesso modo in cui si utilizzano tali attributi nella definizione di code locali.

### Utilizzo degli altri comandi con le code di esempio

È possibile utilizzare i comandi MQSC appropriati per visualizzare, modificare gli attributi di una coda di esempio oppure per eliminare l'oggetto coda di esempio. Ad esempio:

```
* Display the model queue's attributes
* ALL = Display all attributes

DISPLAY QUEUE (GREEN.MODEL.QUEUE) ALL

* ALTER the model to enable puts on any
* dynamic queue created from this model.

ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)

* Delete this model queue:

DELETE QMODEL (RED.MODEL.QUEUE)
```

## Gestione degli oggetti per triggering

MQSeries prevede una funzione che consente l'avvio automatico di un'applicazione qualora si verificano determinate condizioni. Un esempio di tali condizioni si ha quando i messaggi presenti su una coda raggiungono un determinato numero. Questa funzione è denominata *triggering* ed è dettagliatamente descritta nel manuale *MQSeries Application Programming Guide*. Questa sezione descrive il modo come impostare gli oggetti in modo da supportare la funzione triggering in MQSeries per Compaq OpenVMS.

### Definizione di una coda di applicazione per triggering

Una coda di applicazione è una coda locale utilizzata da un'applicazione per la messaggistica tramite MQI. La funzione triggering richiede la definizione di alcuni attributi nella coda di applicazione. La stessa funzione triggering viene abilitata dall'attributo *Trigger* (TRIGGER in MQSC).

Nell'esempio che segue, viene generato un evento trigger quando nella coda locale MOTOR.INS.QUEUE sono presenti 100 messaggi di priorità 5 o superiore:

```
DEFINE QLOCAL (MOTOR.INS.QUEUE) +
PROCESS (MOTOR.INS.PROC) +
  MAXMSGL (2000) +
  DEFPSIST (YES) +
  INITQ (MOTOR.INS.INT.Q) +
  TRIGGER +
  TRIGTYPE (DEPTH) +
  TRIGDPH (100)+
  TRIGMPRI (5)
```

Nell'esempio:

#### **QLOCAL (MOTOR.INS.QUEUE)**

Specifica il nome della coda di applicazione di cui si esegue la definizione.

#### **PROCESS (MOTOR.INS.PROC)**

Specifica il nome dell'applicazione che deve essere avviata da un programma di monitoraggio del trigger.

#### **MAXMSGL (2000)**

Specifica la dimensione massima dei messaggi nella coda.

#### **DEFPSIST (YES)**

Specifica che i messaggi sono permanenti su questa coda.

#### **INITQ (MOTOR.INS.INT.Q)**

Indica il nome della coda di iniziazione sulla quale il Queue Manager invia il messaggio trigger.

#### **TRIGGER**

È l'impostazione dell'attributo trigger.

#### **TRIGTYPE (DEPTH)**

Specifica che un evento trigger deve verificarsi qualora i messaggi della priorità indicata (TRIMPRI) raggiunga il numero specificato in TRIGDPH.

#### **TRIGDPH (100)**

Specifica il numero dei messaggi necessari per generare un evento trigger.

## Gestione degli oggetti per triggering

### TRIGMPRI (5)

Indica la priorità dei messaggi che il Queue Manager deve contare nel decidere di generare l'evento trigger. Vengono contati solo i messaggi di priorità 5 o superiore.

## Definizione di una coda di iniziazione

Quando si verifica un evento trigger, il Queue Manager invia un messaggio trigger alla coda di iniziazione specificata nella definizione della coda di applicazione. Le code di iniziazione non hanno impostazioni particolari, ma è consigliabile utilizzare la definizione seguente della coda locale MOTOR.INS.INT:

```
DEFINE QLOCAL(MOTOR.INS.INT.Q) +
  GET (ENABLED) +
  NOSHARE +
  NOTRIGGER +
  MAXMSGL (2000) +
  MAXDEPTH (10)
```

## Creazione di una definizione di processo

Utilizzare il comando DEFINE PROCESS per creare una definizione di processo. Una definizione di processo associa una coda di applicazione all'applicazione che elabora i messaggi dalla coda. Ciò è possibile grazie all'utilizzo dell'attributo PROCESS nella coda di applicazione MOTOR.INS.QUEUE. Il seguente comando MQSC definisce il processo necessario, MOTOR.INS.PROC, indicato nell'esempio:

```
DEFINE PROCESS (MOTOR.INS.PROC) +
  DESCR ('Insurance request message processing') +
  APPLTYPE (OPENVMS) +
  APPLICID ('DKA0:[MQM.ADMIN.TEST]IRMP01.EXE') +
  USERDATA ('open, close, 235')
```

Nell'esempio:

### **MOTOR.INS.PROC**

Indica il nome della definizione di processo, con un massimo di 15 caratteri.

### **DESCR ('Insurance request message processing')**

Con questi caratteri di testo si indica il programma applicativo a cui si riferisce la definizione, in funzione delle parole chiave. Tale testo viene visualizzato utilizzando il comando DISPLAY PROCESS. Ciò può facilitare l'identificazione della funzione del processo. Se si utilizzano spazi nella stringa, sarà necessario racchiudere il testo tra virgolette.

### **APPLTYPE (OPENVMS)**

Indica il tipo di applicazione eseguita su OpenVMS.

### **APPLICID ('DKA0:[MQM.ADMIN.TEST]IRMP01.EXE')**

Indica il nome del programma applicativo eseguibile.

### **USERDATA ('open, close, 235')**

Sono i dati definiti dall'utente, che verranno utilizzati dall'applicazione.

## Visualizzazione della definizione di processo

Utilizzare il comando DISPLAY PROCESS, con la parola chiave ALL, per esaminare i risultati della definizione. Ad esempio:

```
DISPLAY PROCESS (MOTOR.INS.PROC) ALL

      24 : DISPLAY PROCESS (MOTOR.INS.PROC) ALL
AMQ8407: Display Process details.
DESCR ('Insurance request message processing') +
APPLICID ('DKA0:[MQM.ADMIN.TEST]IRMP01.EXE') +
USERDATA (open, close, 235) +
PROCESS (MOTOR.INS.PROC) +
APPLTYPE (OPENVMS)
```

È possibile utilizzare anche il comando ALTER PROCESS per modificare una definizione di processo esistente oppure il comando DELETE PROCESS per eliminare una definizione di processo.





---

## Capitolo 5. Automazione delle attività di gestione

Questo capitolo presume che l'utente abbia già avuto esperienza di gestione degli oggetti MQSeries.

Nel caso in cui si decida di migliorare l'installazione rendendo automatiche alcune attività di gestione e di controllo, è possibile utilizzare i comandi PCF (Programmable Command Format) per i Queue Manager sia locale che remoto.

Questo capitolo descrive:

- Come utilizzare i comandi PCF (Programmable Command Formats) per rendere automatiche le attività di gestione nella sezione "Comandi PCF"
- Come utilizzare il server di comando nella sezione "Gestione del server di comando per una gestione remota" a pagina 65

---

### Comandi PCF

La funzione dei comandi MQSeries PCF (Programmable Command Format) è di consentire la pianificazione delle attività di gestione in un programma di gestione. Utilizzando questo programma sarà possibile creare code, definizioni di processi, canali e nomi elenchi oltre che effettuare modifiche Queue Manager.

Le funzioni MQSC sono realizzate in modo analogo dai comandi PCF.

Infatti, è possibile impostare un programma in modo che un singolo nodo può emettere comandi PCF verso qualsiasi Queue Manager della rete. In questo modo le attività di gestione vengono centralizzate e al tempo stesso automatizzate.

Ogni comando PCF è una struttura di dati che può essere integrato nella parte di un messaggio MQSeries relativa ai dati applicativi. Ogni comando viene inviato al Queue Manager di destinazione utilizzando la funzione MQI MQPUT così come avviene per ogni altro messaggio. Il server di comando del Queue Manager che riceve il messaggio lo riconosce come comando e lo esegue. Per ricevere le risposte, l'applicazione realizza una chiamata MQGET e i dati relativi alla risposta sono inviati in un'altra struttura di dati. A questo punto l'applicazione può analizzare la risposta e eseguirne il contenuto.

**Nota:** diversamente dai comandi MQSC, i comandi PCF e le relative risposte non sono in un formato testo leggibile.

In breve, queste sono note che il programmatore di applicazione deve specificare quando si crea un messaggio relativo ad un comando PCF:

#### Descrittore messaggio

Questo è un descrittore messaggio standard MQSeries, nel quale:  
Tipo di messaggio (*MsgType*) è MQMT\_REQUEST.  
Formati di messaggio (*Format*) è MQFMT\_ADMIN.

#### Dati applicativi

Contiene i messaggi PCF incluso l'intestazione PCF, nel quale:

Il tipo di messaggio PCF (*Type*) specifica MQCFT\_COMMAND.

L'identificativo del comando specifica il comando, per esempio, *Cambia coda* (MQCMD\_CHANGE\_Q).

## comandi PCF

Per una descrizione completa delle strutture dei dati PCF e sulla loro implementazione, consultare il manuale *MQSeries Programmable System Management*.

### Attributi MQSC e PCF

Gli attributi degli oggetti specificati in MQSC sono riportati in questo manuale in lettere maiuscole (ad esempio RQMNAME), sebbene essi non siano sensibili al maiuscolo/minuscolo. I nomi degli attributi MQSC hanno una lunghezza massima di otto caratteri.

Gli attributi degli oggetti in PCF, il cui nome è formato da più di otto caratteri, è riportato in questo manuale con il tipo di carattere "italics". Ad esempio, il comando PCF equivalente a RQMNAME è *RemoteQMgrName*.

### Escape PCF

Gli Escape PCF sono comandi PCF che contengono i comandi MQSC con il messaggio di testo. È possibile utilizzare i comandi PCF per inviare comandi ad un Queue Manager remoto. Per ulteriori informazioni relative all'utilizzo di escape PCF, consultare il manuale *MQSeries Programmable System Management*.

### Utilizzo di MQAI per semplificare l'impiego di PCF

MQAI è un'interfaccia di gestione per MQSeries disponibile sulla piattaforma OpenVMS.

Essa realizza attività di gestione su un Queue Manager grazie all'impiego di *data bags* (cartella dati). La cartella dati (Data bag) consente di gestire proprietà o parametri di oggetti in modo più semplice rispetto a PCF.

MQAI può essere utilizzato:

- **Per semplificare l'utilizzo dei messaggi PCF**

MQAI consente una facile gestione di MQSeries; non è necessario scrivere i messaggi PCF e questo evita i problemi che nascono dalla complessità delle strutture dei dati.

Per passare parametri in programmi scritti utilizzando le chiamate MQI, i messaggi PCF devono contenere il comando e i dettagli dei dati String o Integer. A questo scopo, per ogni struttura occorre fornire alcune specifiche al programma e destinare un apposito spazio nella memoria. Questa operazione è complessa ed impiega molto tempo.

Del resto, i programmi vengono compilati utilizzando i parametri MQAI pass nell'appropriata data bag (cartella dati) ed è richiesta una sola istruzione per ogni struttura. L'utilizzo della cartella dati MQAI esclude la necessità di fornire specifiche e assegnare spazio nella memoria e aumenta il grado di isolamento dai dettagli del PCF.

- **Per implementare le applicazioni di autogestione e gli strumenti di gestione**

Ad esempio, Active Directory Services fornito con MQSeries per Windows NT e Windows 2000 Versione 5.2 utilizza MQAI. Attualmente non esistono esempi di impiego sulla piattaforma OpenVMS.

- **Per gestire il caso in cui si verifica un errore in modo più semplice**

È difficile risalire al codice dai comandi MQSC, tuttavia MQAI rende più facile al programma la gestione quando si verifica un errore.

Dopo aver creato e inserito dati nella cartella dati (data bag), è possibile inviare un messaggio del comando di gestione al server di comando di un Queue Manager, utilizzando una chiamata mqExecute che attenderà un messaggio di risposta. La chiamata mqExecute gestisce lo scambio con il server di comandi e invia la risposta in una response bag (cartella risposte).

Per ulteriori informazioni relative all'utilizzo di MQAI, consultare il manuale *MQSeries Administration Interface Programming Guide and Reference*.

Per informazioni generali relative a PCFs, consultare il manuale *MQSeries Programmable System Management*.

---

## Gestione del server di comando per una gestione remota

È possibile associare un server di comando a ciascun Queue Manager. Un server di comando elabora tutti i comandi inviati da una Queue Manager remota o i comandi PCF delle applicazioni. Esso fornisce i comandi al Queue Manager per l'elaborazione e restituisce un codice di completamento o un messaggio operatore in base all'origine del comando.

Un server di comando è necessario ad ogni gestione che utilizzano PCF, MQAI ed anche per la gestione in remoto.

**Nota:** per le gestioni in remoto, assicurarsi che il Queue Manager di destinazione sia in esecuzione. Altrimenti, i messaggi, che contengono i comandi, non possono lasciare il Queue Manager da cui sono inviati. Invece, questi messaggi sono accodati alla trasmissione locale che invia dati al Queue Manager remoto. Questa situazione deve essere evitata, se possibile.

## Avvio del server di comando

Per avviare il server di comando utilizzare questo comando:

```
strmqcsv "saturn.queue.manager"
```

dove saturn.queue.manager indica il Queue Manager per il quale si avvia il server di comando.

## Visualizzazione dello stato del server di comando

Per le gestioni in remoto, assicurarsi che il server di comando del Queue Manager di destinazione sia in esecuzione. Diversamente i comandi in remoto non potranno essere elaborati. Tutti i messaggi che contengono comandi vengono accodati alla coda comandi del Queue Manager di destinazione.

Per visualizzare lo stato del server di comando relativo ad un Queue Manager, qui definito saturn.queue.manager, il comando è:

```
dspmqcsv "saturn.queue.manager"
```

È necessario inviare questo comando alla macchina di destinazione. Se il server di comando è in esecuzione viene inviato il seguente messaggio:

## Gestione remota del server di comando

```
AMQ8027    MQSeries Command Server Status ...: Running
```

## Arresto del server di comando

Per chiudere un server di comando, utilizzando l'esempio precedente, il comando è:

```
endmqcsv "saturn.queue.manager"
```

È possibile arrestare il server di comando in due differenti modi:

- Per un arresto controllato, modo predefinito, utilizzare il comando **endmqcsv** con il flag **-c**.
- Per un arresto immediato, utilizzare il comando **endmqcsv** con il flag **-i**.

**Nota:** arrestando un Queue Manager si chiude anche il server di comando ad esso associato (se avviato).

## Gestione remota del server di comando

---

## Capitolo 6. Gestione di oggetti remoti di MQSeries

Questo capitolo descrive come gestire oggetti di MQSeries su un altro Queue Manager. In esso si descrive come utilizzare gli oggetti di coda remota per controllare la destinazione dei messaggi e dei messaggi di risposta.

Esso contiene le seguenti sezioni:

- “Canali, cluster e accodamento remoto”
- “Gestione remota da un Queue Manager locale mediante comandi MQSC” a pagina 71
- “Creazione di una definizione locale di una coda remota” a pagina 77
- “Utilizzo di definizioni di coda remota come alias” a pagina 80

Per ulteriori informazioni sui canali, i relativi attributi e le modalità di impostazione, consultare il manuale *MQSeries Intercommunication*.

---

### Canali, cluster e accodamento remoto

Un Queue Manager comunica con un altro Queue Manager inviando un messaggio e, se necessario, ricevendo una risposta. Il Queue Manager ricevente potrebbe essere:

- Sullo stesso computer
- Su un altro computer nella stessa ubicazione o dall'altra parte del mondo
- In esecuzione sulla stessa piattaforma del Queue Manager locale
- In esecuzione su un'altra piattaforma supportata da MQSeries

Questi messaggi potrebbero essere originati da:

- Programmi applicativi scritti dall'utente, che trasferiscono dati da un nodo a un altro.
- Applicazioni di gestione scritte dall'utente, che utilizzano i PCF, MQAI o ADSI.
- Queue Manager che inviano:
  - Messaggi di evento di strumentazione inviati a un altro Queue Manager.
  - Comandi MQSC emessi da un comando **runmqsc** in modo indiretto (in cui i comandi sono eseguiti su un altro Queue Manager).

Prima che sia possibile inviare un messaggio a un Queue Manager remoto, il Queue Manager locale necessita di un meccanismo per rilevare l'arrivo di messaggi e trasportarli consistenti in:

- Almeno un canale
- Una coda di trasmissione
- Un agente del canale di messaggio (MCA)
- Un listener di canale
- Un channel initiator

Un canale è una connessione a senso unico tra due Queue Manager e può trasportare messaggi destinati a un qualsiasi numero di code presso il Queue Manager remoto.

Ciascuna estremità del canale dispone di una distinta definizione. Ad esempio, se un'estremità è un mittente o un server, l'altra estremità dovrà essere un ricevente o un richiedente. Un semplice canale consiste in una *definizione di canale mittente*

## Gestione di oggetti remoti

sull'estremità del Queue Manager e in una *definizione di canale ricevente* sull'estremità del Queue Manager remoto. Le due definizioni devono avere lo stesso nome e costituire assieme un unico canale.

Se si presume che il Queue Manager remoto risponda ai messaggi inviati dal Queue Manager locale, sarà necessario impostare un secondo canale per inviare risposte al Queue Manager locale.

I canali vengono definiti mediante il comando MQSC DEFINE CHANNEL. In questo capitolo, gli esempi relativi ai canali utilizzano gli attributi di canale predefiniti se non diversamente specificato.

In ciascuna estremità di un canale esiste un MCA (Message Channel Agent) che controlla l'invio e la ricezione di messaggi. È compito di MCA prelevare i messaggi dalla coda di trasmissione e inserirli nella connessione tra i Queue Manager.

Una coda di trasmissione è una coda locale specializzata che contiene temporaneamente i messaggi prima che siano prelevati da MCA e inviati al Queue Manager remoto. Specificare il nome della coda di trasmissione su una *definizione di coda remota*.

La sezione "Preparazione di canali e code di trasmissione per la gestione remota" a pagina 72 illustra l'utilizzo di tali definizioni per impostare la gestione remota.

Per ulteriori informazioni sull'impostazione dell'accodamento distribuito in generale, consultare il manuale *MQSeries Intercommunication*.

## Gestione remota mediante cluster

In una normale rete MQSeries che utilizza l'accodamento distribuito, tutti i Queue Manager sono indipendenti. Affinché un Queue Manager possa inviare un messaggio a un altro Queue Manager, occorre aver definito una coda di trasmissione, un canale diretto al Queue Manager remoto e una definizione di coda remota per ogni coda a cui inviare messaggi.

Un *cluster* è un gruppo di Queue Manager impostato in modo che i Queue Manager possano comunicare direttamente tra loro tramite un'unica rete, senza necessità di definizioni di coda di trasmissione complessa, di canale e di coda. È possibile impostare facilmente i cluster, che di norma contengono Queue Manager correlati in maniera logica e necessitano la condivisione dei dati o delle applicazioni.

Una volta creato un cluster, i Queue Manager al suo interno possono comunicare tra loro *senza la necessità di complicate definizioni di canale o di code remote*. Persino il cluster più piccolo ridurrà i rischi relativi alla gestione del sistema.

Stabilendo una rete di Queue Manager in un cluster si avranno meno definizioni rispetto allo stabilire un ambiente tradizionale di accodamento distribuito. Con un numero inferiore di definizioni da effettuare, sarà possibile impostare o modificare la propria rete più rapidamente e facilmente e si ridurrà il rischio di errori nelle definizioni.

Per impostare un cluster, di norma è necessario una definizione di mittente cluster (CLUSSDR) e una definizione di ricevente cluster (CLUSRCVR) per Queue Manager. Non saranno necessarie alcune definizioni di coda di trasmissione o di



coda remota. I principi della gestione remota sono gli stessi all'interno di un cluster, ma le definizioni stesse sono enormemente semplificate.

Per ulteriori informazioni sui cluster, i relativi attributi e le modalità di impostazione, consultare il manuale *MQSeries Queue Manager Clusters*.

---

## Gestione remota da un Queue Manager locale mediante comandi MQSC

Questa sezione indica come gestire un Queue Manager remoto da un Queue Manager locale. È possibile implementare la gestione remota da un nodo locale mediante:

- Comandi MQSC
- Comandi PCF

La preparazione delle code e dei canale è fondamentalmente identica per entrambi i metodi. In questo volume, gli esempi illustrano i comandi MQSC, poiché sono più semplici da comprendere. Tuttavia, è possibile convertire gli esempi in PCF, se lo si desidera. Per ulteriori informazioni sulla scrittura di programmi di gestione, consultare il manuale *MQSeries Programmable System Management*.

Con la gestione remota si inviano comandi MQSC a un Queue Manager remoto in maniera interattiva o da un file di testo contenente i comandi. Il Queue Manager remoto potrebbe risiedere sullo stesso computer o, più comunemente, su un computer diverso. È possibile gestire in modo remoto i Queue Manager in diversi ambienti MQSeries, inclusi AIX®, AS/400, MVS/ESA e OS/2®.

Per implementare la gestione remota, occorre creare determinati oggetti. A meno che non siano necessari requisiti specializzati, i valori predefiniti dovrebbero risultare sufficienti (ad esempio, per la lunghezza del messaggio).

### Preparazione dei Queue Manager alla gestione remota

La Figura 6 a pagina 72 illustra la configurazione dei Queue Manager e dei canali che sono necessari per la gestione remota mediante il comando **runmqsc**. `source.queue.manager` è il Queue Manager *source* da cui è possibile emettere comandi MQSC e a cui vengono restituiti i risultati di tali comandi (messaggi per l'operatore), se possibile. `target.queue.manager` è il Queue Manager di destinazione, che elabora i comandi e genera qualsiasi messaggio per l'operatore.

**Nota:** `source.queue.manager` *deve* essere il Queue Manager predefinito. Per ulteriori informazioni sulla creazione del Queue Manager, consultare la sezione "crtmqm (Crea Queue Manager)" a pagina 258.

## Gestione remota

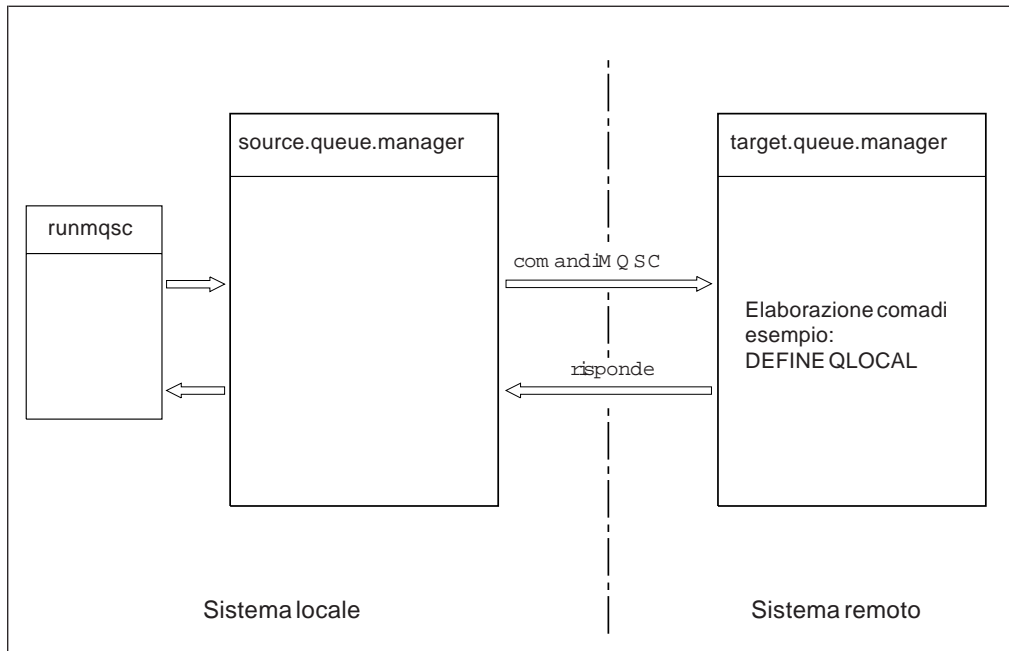


Figura 6. Gestione remota

Se non sono ancora state effettuate le seguenti operazioni, su entrambi i sistemi occorre:

- Creare il Queue Manager, mediante il comando **crtmqm**.
- Avviare il Queue Manager, mediante il comando **strmqm**.

Occorre eseguire tali comandi localmente o su una rete, ad esempio Telnet.

Sul Queue Manager di destinazione:

- Deve essere presente la coda di comando, **SYSTEM.ADMIN.COMMAND.QUEUE**. Per impostazione predefinita, questa coda viene creata quando viene creato un queue manager.
- È necessario avviare il server di comandi mediante il comando **strmqcsv**.

## Preparazione di canali e code di trasmissione per la gestione remota

Per eseguire in remoto i comandi MQSC, occorre impostare due canali, uno per ciascuna direzione e per le code di trasmissione associate. Tale esempio presume che si sta utilizzando il protocollo TCP/IP come tipo di trasporto e che si è al corrente del coinvolgimento dell'indirizzo TCP/IP.

Il canale **source.to.target** serve a inviare comandi MQSC dal Queue Manager di origine alla destinazione. Il relativo mittente è all'indirizzo **source.queue.manager** e il relativo ricevente è al Queue Manager **target.queue.manager**. Il canale **target.to.source** serve a restituire l'output dai comandi e qualsiasi messaggio per l'operatore generato per il Queue Manager di origine. Occorre inoltre definire una coda di trasmissione per ciascun mittente. Questa coda è una coda locale a cui viene assegnato il nome del Queue Manager ricevente. Per consentire la gestione in remoto, il nome **XMITQ** deve corrispondere al nome del queue manager remoto, a meno che per il queue manager non si stia utilizzando un alias. Figura 7 a pagina 73 riepiloga tale configurazione.

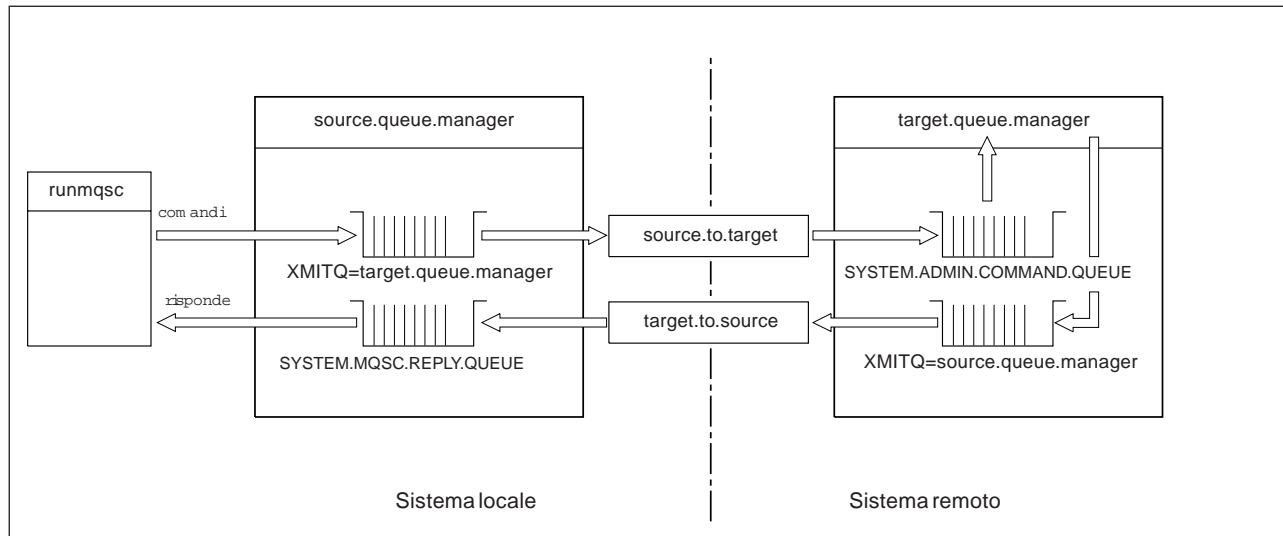


Figura 7. Impostazione di canali e code per la gestione remota

Consultare il manuale *MQSeries Intercommunication* per ulteriori informazioni sull'impostazione di canali remoti.

## Definizione di canali e code di trasmissione

Sul Queue Manager di origine, emettere questi comandi MQSC per definire i canali e la coda di trasmissione:

\* Define the sender channel at the source queue manager

```
DEFINE CHANNEL ('source.to.target') +
  CHLTYPE(SDR) +
  CONNAME (RHX5498) +
  XMITQ ('target.queue.manager') +
  TRPTYPE(TCP)
```

\* Define the receiver channel at the source queue manager

```
DEFINE CHANNEL ('target.to.source') +
  CHLTYPE(RCVR) +
  TRPTYPE(TCP)
```

\* Define the transmission queue on the source

```
DEFINE QLOCAL ('target.queue.manager') +
  USAGE (XMITQ)
```

## Gestione remota

Emettere tali comandi sul Queue Manager di destinazione (`target.queue.manager`), per creare i canali e la coda di trasmissione:

```
* Define the sender channel on the destination queue manager

DEFINE CHANNEL ('target.to.source') +
  CHLTYPE(SDR) +
  CONNAME (RHX7721) +
  XMITQ ('source.queue.manager') +
  TRPTYPE(TCP)

* Define the receiver channel on the destination queue manager

DEFINE CHANNEL ('source.to.target') +
  CHLTYPE(RCVR) +
  TRPTYPE(TCP)

* Define the transmission queue on the destination queue manager

DEFINE QLOCAL ('source.queue.manager') +
  USAGE (XMITQ)
```

**Nota:** i nomi della connessione TCP/IP specificati per l'attributo `CONNAME` nelle definizioni di canale mittente sono intesi solo a scopo illustrativo. Questo è il nome della rete del computer all'*altra* estremità della connessione. Utilizzare i valori appropriati per la propria rete.

## Avvio dei canali

La seguente descrizione presuppone che entrambe le estremità del canale siano in esecuzione su MQSeries per Compaq OpenVMS. In caso contrario, fare riferimento alla relativa documentazione per l'estremità diversa da OpenVMS del canale.

Per avviare i due canali, assicurarsi dapprima che i servizi TCP/IP siano stati configurati per MQSeries su entrambi i nodi e siano in esecuzione su entrambe le estremità delle connessioni.

Avviare un listener sull'estremità ricevente di ciascun canale.

- Sul Queue Manager di origine, digitare:

```
runmqlsr -m "source.queue.manager" -t tcp
```

- Sul Queue Manager di destinazione, digitare:

```
runmqlsr -m "target.queue.manager" -t tcp
```

Il nome del Queue Manager non è necessario sul comando **runmqchl** per il Queue Manager di origine, poiché questo deve essere il Queue Manager predefinito. Il nome del Queue Manager è necessario sul comando **runmchl** per il Queue Manager di destinazione se questo non è il Queue Manager predefinito sul relativo nodo.

Quindi, avviare i canali:

- Sul Queue Manager di origine, digitare:

```
runmqchl -m "source.queue.manager" -c "source.to.target"
```

- Sul Queue Manager di destinazione, digitare:

```
runmqchl -m "target.queue.manager" -c "target.to.source"
```

I comandi **runmqlsr** e **runmqchl** sono comandi di controllo di MQSeries. Non è possibile emetterli mediante **runmqsc**. È possibile avviare i listener e i canali mediante i comandi **runmqsc** o gli script (start channel e start listener).

## Definizione automatica dei canali

La definizione automatica si applica solo se il Queue Manager di destinazione è in esecuzione sui prodotti MQSeries Versione 5.1 o successive. Se viene ricevuta una richiesta di allegato in arrivo e non è possibile rilevare una definizione di un ricevente appropriato o di connessione al server nel file di definizione canale (CDF), MQSeries crea automaticamente una definizione e la aggiunge al CDF. Le definizioni automatiche si basano su due definizioni predefinite fornite con MQSeries: SYSTEM.AUTO.RECEIVER e SYSTEM.AUTO.SVRCONN.

Abilitare la definizione automatica delle definizioni di ricevente e connessione al server aggiornando l'oggetto Queue Manager mediante il comando MQSC, ALTER QMGR (o il comando PCF Change Queue Manager).

Per ulteriori informazioni sulla creazione automatica delle definizioni di canale, consultare il manuale *MQSeries Intercommunication*.

Per informazioni sulla definizione automatica di canali per i cluster, consultare il manuale *MQSeries Queue Manager Clusters*.

## Emissione remota di comandi MQSC

Il server di comandi *deve* essere in esecuzione sul Queue Manager di destinazione, se sta per elaborare i comandi MQSC in remoto (ciò non è necessario sul Queue Manager di origine).

- Sul Queue Manager di destinazione, digitare:

```
strmqcsv "target.queue.manager"
```

- Sul Queue Manager di origine, è possibile eseguire MQSC interattivamente in modalità accodamento digitando:

```
runmqsc -w 30 "target.queue.manager"
```

Questo formato del comando **runmqsc**, con il flag **-w**, esegue i comandi MQSC in modalità accodamento, in cui i comandi vengono inseriti (in formato modificato) nella coda di input del server di comandi ed eseguiti in ordine.

## Gestione remota

Quando si digita un comando MQSC, questo viene reindirizzato al Queue Manager remoto, in questo caso `target.queue.manager`. Il timeout è impostato su 30 secondi; se non viene ricevuta alcuna risposta entro 30 secondi, sarà generato il seguente messaggio sul Queue Manager locale (di origine):

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

Al termine della sessione di MQSC, il Queue Manager locale visualizza le eventuali risposte scadute giunte. Quando la sessione di MQSC è terminata, le eventuali ulteriori risposte vengono eliminate.

In modo indiretto, è inoltre possibile eseguire un file di comandi MQSC su un Queue Manager remoto. Ad esempio:

```
runmqsc -w 60 "target.queue.manager" < mycomds.in > report.out
```

dove `mycomds.in` è un file contenente comandi MQSC e `report.out` è il file di notifica.

## Funzionamento dei Queue Manager su MVS/ESA

È possibile emettere comandi MQSC su un Queue Manager MVS/ESA da un Queue Manager di MQSeries per Compaq OpenVMS. Tuttavia, a tale scopo, è necessario modificare il comando `runmqsc` e le definizioni di canale al mittente.

In particolare, si aggiunge il flag `-x` al comando `runmqsc` su un nodo OpenVMS:

```
runmqsc -w 30 -x "target.queue.manager"
```

Sul canale mittente, impostare l'attributo `CONVERT` su `YES`. Ciò specifica che la conversione dati necessaria tra i sistemi sarà eseguita al termine di OpenVMS. Il comando di definizione canale diviene ora:

```
* Define the sender channel at the source queue manager on OpenVMS  
  
DEFINE CHANNEL ('source.to.target') +  
  CHLTYPE(SDR) +  
  CONNAME (RHX5498) +  
  XMITQ ('target.queue.manager') +  
  TRPTYPE(TCP) +  
  CONVERT (YES)
```

È necessario definire il canale ricevente e la coda di trasmissione presso il Queue Manager di origine come in precedenza. Ancora una volta, quest'esempio presuppone che il protocollo di trasmissione utilizzato sia TCP/IP.

### Consigli per l'accodamento remoto

Quando si implementa l'accodamento remoto:

1. Inserire i comandi MQSC da eseguire sul sistema remoto in un file di comandi.
2. Verificare localmente i comandi MQSC, specificando il flag `-v` sul comando `runmqsc`.

Non sarà possibile utilizzare `runmqsc` per verificare i comandi MQSC su un altro Queue Manager.

3. Verificare, quando possibile, che il file di comandi sia eseguito localmente senza errori.
4. Infine, eseguire il file di comandi sul sistema remoto.

### Eventuali problemi nell'utilizzo remoto di MQSC

Se si incontrano difficoltà nell'esecuzione remota di comandi MQSC, avvalersi del seguente elenco di controllo per verificare di aver:

- Avviato il server di comandi sul Queue Manager di destinazione.
- Definito una coda di trasmissione valida.
- Definito le due estremità dei canali di messaggio per:
  - Il canale assieme al quale si inviano i comandi.
  - Il canale assieme al quale si restituiscono le risposte.
- Specificato il nome di connessione corretto (CONNNAME) nella definizione di canale.
- Avviato i listener prima di aver avviato i canali di messaggio.
- Verificato che l'intervallo di disconnessione non sia scaduto, ad esempio, se un canale è stato avviato, ma arrestato dopo qualche tempo. Ciò è particolarmente importante se si avviano i canali manualmente.
- Controllato che non si stiano inviando richieste da un Queue Manager di origine che il Queue Manager di destinazione non può interpretare (ad esempio, richieste che includano nuovi parametri).

Consultare anche la sezione "Risoluzione dei problemi con MQSC" a pagina 45.

---

### Creazione di una definizione locale di una coda remota

È possibile utilizzare una definizione di coda remota come definizione locale di una coda remota. Creare un oggetto coda remota sul Queue Manager locale per identificare una coda locale di un altro Queue Manager.

### Analisi del funzionamento delle definizioni locali di code remote

Un'applicazione si connette a un Queue Manager locale e quindi emette una chiamata MQOPEN. Nella chiamata aperta, il nome di coda specificato è quello di una definizione di coda remota sul Queue Manager locale. La definizione di coda remota fornisce i nomi della coda di destinazione, il Queue Manager di destinazione e, facoltativamente, una coda di trasmissione. Per inserire un messaggio sulla coda remota, l'applicazione emette una chiamata MQPUT, specificando l'handle restituito dalla chiamata MQOPEN. Il Queue Manager aggiunge il nome di coda remota e il nome del Queue Manager remoto a un'intestazione di trasmissione nel messaggio. Queste informazioni sono utilizzate per indirizzare il messaggio alla destinazione corretta nella rete.

Come amministratore, l'utente può controllare la destinazione del messaggio modificando la definizione della coda remota.

### Esempio

**Scopo:** È necessario che un'applicazione inserisca un messaggio in una coda appartenente a un Queue Manager remoto.

## Definizione locale di una coda remota

**Come funziona:** L'applicazione si connette a un Queue Manager, ad esempio, saturn.queue.manager. La coda di destinazione appartiene a un altro Queue Manager.

Sulla chiamata MQOPEN, l'applicazione specifica questi campi:

Valore campo	Descrizione
<i>ObjectName</i> CYAN.REMOTE.QUEUE	Specifica il nome locale dell'oggetto coda remota. Ciò definisce la coda di destinazione e il Queue Manager di destinazione.
<i>ObjectType</i> (Coda)	Identifica quest'oggetto come una coda.
<i>ObjectQmgrName</i> Vuoto o saturn.queue.manager	Questo campo è facoltativo.  Se lasciato vuoto, il nome del Queue Manager locale viene presupposto (cioè il Queue Manager su cui è stata effettuata la definizione di coda remota e al quale l'applicazione è connessa).  Se non è vuoto, è necessario specificare il nome del Queue Manager locale.

Al termine, l'applicazione emette una chiamata MQPUT per inserire un messaggio su questa coda.

Sul Queue Manager locale, è possibile creare una definizione loca di una coda remota mediante i seguenti comandi MQSC:

```
DEFINE QREMOTE (CYAN.REMOTE.QUEUE) +
  DESCR ('Queue for auto insurance requests from the branches') +
  RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE) +
  RQMNAME ('jupiter.queue.manager') +
  XMITQ (INQUOTE.XMIT.QUEUE)
```

Dove:

### **QREMOTE (CYAN.REMOTE.QUEUE)**

Specifica il nome locale dell'oggetto coda remota. Questo è il nome che le applicazioni connesse a questo Queue Manager deve specificare nella chiamata MQOPEN per aprire la coda AUTOMOBILE.INSURANCE.QUOTE.QUEUE sul Queue Manager remoto jupiter.queue.manager.

### **DESCR ('Queue for auto insurance requests from the branches')**

Testo aggiuntivo che descrive l'utilizzo della coda.

### **RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)**

Specifica il nome della coda di destinazione sul Queue Manager remoto. Questa è la coda di destinazione effettiva per i messaggi che vengono inviati dalle applicazioni che specificano il nome coda CYAN.REMOTE.QUEUE. La coda AUTOMOBILE.INSURANCE.QUOTE.QUEUE deve essere definita come una coda locale sul Queue Manager remoto.

### **RQMNAME ('jupiter.queue.manager')**

Specifica il nome del Queue Manager remoto a cui appartiene la coda di



## Definizione locale di una coda remota

destinazione AUTOMOBILE.INSURANCE.QUOTE.QUEUE. Questo nome deve essere racchiuso fra virgolette **single**.

### XMITQ (INQUOTE.XMIT.QUEUE)

Specifica il nome della coda di trasmissione. Questo è facoltativo; se non specificato, viene utilizzata una coda con lo stesso nome del Queue Manager remoto.

In entrambi i casi, occorre definire la coda di trasmissione appropriata come coda locale con un attributo *Usage* specificando che si tratta di una coda di trasmissione (USAGE(XMIT) in MQSC).

## Un modo alternativo per inserire messaggi in una coda remota

Utilizzare una definizione locale di una coda remota non è l'unico modo di inserire messaggi su una coda remota. Le applicazioni possono specificare il nome di coda completo, che include il nome del Queue Manager remoto, come parte della chiamata MQOPEN. In tal caso, non è necessaria una definizione locale di coda remota. Tuttavia, tale alternativa significa che le applicazioni devono conoscere o avere accesso al nome del Queue Manager remoto al runtime.

## Utilizzo di altri comandi con code remote

È possibile utilizzare i comandi MQSC appropriati per visualizzare o modificare gli attributi di un oggetto coda remota o eliminare l'oggetto coda remota. Ad esempio:

```
* Display the remote queue's attributes.  
* ALL = Display all attributes  
  
DISPLAY QUEUE (CYAN.REMOTE.QUEUE) ALL  
  
* ALTER the remote queue to enable puts.  
* This does not affect the destination queue,  
* only applications that specify this remote queue.  
  
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)  
  
* Delete this remote queue  
* This does not affect the destination queue  
* only its local definition  
  
DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

**Nota:** eliminando una coda remota, si elimina solo la rappresentazione locale della coda remota. Non eliminare la coda remota stessa o eventuali messaggi a essa appartenenti.

## Creazione di una coda di trasmissione

Una coda di trasmissione è una coda locale che viene utilizzata quando un Queue Manager inoltra messaggi a un Queue Manager remoto tramite un canale di messaggi. Il canale fornisce un collegamento a senso unico al Queue Manager remoto. I messaggi vengono accodati nella coda di trasmissione finché il canale sarà in grado di accettarli. Nel definire un canale, è necessario specificare un nome di coda di trasmissione all'estremità mittente del canale di messaggi.

## Definizione locale di una coda remota

L'attributo *Usage* (USAGE in MQSC) definisce se una coda è una coda di trasmissione o una coda normale.

### Code di trasmissione predefinite

Facoltativamente, è possibile specificare una coda di trasmissione in un oggetto coda remota, mediante l'attributo *XmitQName* (XMITQ in MQSC). Se non viene definita alcuna coda di trasmissione, sarà utilizzata una coda predefinita. Quando le applicazioni inseriscono messaggi su una coda remota, se esiste una coda di trasmissione con lo stesso nome della coda di destinazione, sarà utilizzata questa coda. Se questa coda non esistesse, sarà utilizzata la coda specificata dall'attributo *DefaultXmitQ* (DEFXMITQ in MQSC) sul Queue Manager locale.

Ad esempio, il seguente comando MQSC crea una coda di trasmissione predefinita su `source.queue.manager` per messaggi inviati a `target.queue.manager`:

```
DEFINE QLOCAL ('target.queue.manager') +
  DESCR ('Default transmission queue for target qm') +
  USAGE (XMITQ)
```

Le applicazioni possono inserire messaggi direttamente in una coda di trasmissione, con un'intestazione appropriata oppure è possibile inserirli direttamente, ad esempio, tramite una definizione di coda remota. Consultare anche la sezione "Creazione di una definizione locale di una coda remota" a pagina 77.

---

## Utilizzo di definizioni di coda remota come alias

Oltre a individuare una coda su un altro Queue Manager, è anche possibile utilizzare una definizione locale di coda remota per entrambi:

- Alias di Queue Manager
- Alias di coda di risposta RTQ

Entrambi i tipi di alias vengono risolti tramite la definizione locale di una coda remota.

Come di norma nell'accodamento remoto, occorre impostare i canali appropriati se il messaggio deve giungere alla relativa destinazione.

### Alias di Queue Manager

Un alias è il processo per il quale un nome del Queue Manager di destinazione come specificato in un messaggio è modificato da un Queue Manager sulla route del messaggio. Le alias di Queue Manager sono importanti perché è possibile utilizzarle per controllare la destinazione di messaggi all'interno di una rete di Queue Manager.

A tale scopo, modificare la definizione di coda remota sul Queue Manager al punto di controllo. L'applicazione mittente non è consapevole del fatto che il nome del Queue Manager specificato sia un alias.

Per ulteriori informazioni sugli alias del Queue Manager, consultare il manuale *MQSeries Intercommunication*.

## Alias di coda di risposta RTQ

Facoltativamente, un'applicazione può specificare il nome di una coda di risposta RTQ quando inserisce un *messaggio di richiesta* su una coda. Se l'applicazione che elabora il messaggio estrae il nome della coda di risposta RTQ, sa dove inviare il *messaggio di risposta*, se necessario.

Un alias di coda di risposta RTQ è il processo per il quale una coda di risposta RTQ come specificato in un messaggio di richiesta è modificato da un Queue Manager sulla route del messaggio. L'applicazione mittente non è consapevole del fatto che il nome della coda di risposta RTQ specificato sia un alias.

Un alias di una coda di risposta RTQ consente di modificare il nome della coda di risposta e facoltativamente il relativo Queue Manager. A sua volta, questo consente di controllare la route da utilizzare per i messaggi di risposta.

Per ulteriori informazioni su messaggi di richiesta, messaggi di risposta e code di risposta, consultare il manuale *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa*. Per ulteriori informazioni sugli alias delle code di risposta RTQ, consultare il manuale *MQSeries Intercommunication*.

## Conversione dei dati

Il Queue Manager può convertire i dati dei messaggi in formati definiti da MQSeries (noti anche come formati incorporati) da un insieme di caratteri codificati a un altro, a condizione che entrambi gli insiemi di caratteri si riferiscano a un'unica lingua o a un gruppo di lingue simili.

Ad esempio, è supportata la conversione tra insiemi di caratteri codificati i cui identificativi (CCSID) sono 850 e 500, poiché entrambi si applicano alle lingue europee occidentali.

Per conversioni di caratteri NL (New Line) EBCDIC in ASCII, consultare la sezione "La voce AllQueueManagers" a pagina 181.

Le conversioni supportate sono definite nelle tabelle di conversione Code page nel manuale *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa*.

### Quando un Queue Manager non è in grado di convertire messaggi in formati incorporati

Il Queue Manager non è in grado di convertire automaticamente i messaggi in formati incorporati se i relativi CCSID rappresentano diversi gruppi di lingue nazionali. Ad esempio, la conversione tra CCSID 850 e CCSID 1025 (che è un insieme di caratteri codificati EBCDIC per lingue che utilizzano script in cirillico) non è supportata perché molti dei caratteri in un insieme di caratteri codificati non possono essere rappresentati nell'altro. Se si dispone di una rete di Queue Manager funzionanti in diverse lingue nazionali e non è supportata la conversione di dati tra alcuni insiemi di caratteri codificati, sarà possibile abilitare una conversione predefinita. La conversione predefinita dei dati è descritta nella sezione "Conversione di dati predefinita" a pagina 82.

### File ccsid.tbl

Il file ccsid.tbl specifica:

- Qualsiasi insieme di codici aggiuntivo. Per specificare insiemi di codici aggiuntivi, occorre modificare ccsid.tbl (le istruzioni a tale scopo sono fornite nel file).
- Qualsiasi conversione di dati predefinita.

## Alias

È possibile aggiornare le informazioni registrate nel file `ccsid.tbl`. Ciò potrebbe essere opportuno se, ad esempio, un futuro rilascio del proprio sistema operativo supportasse insiemi di caratteri codificati aggiuntivi.

In MQSeries per Compaq OpenVMS, un file `ccsid.tbl` di esempio è fornito come `MQS_EXAMPLES:CCSID.TBL`

e il file `ccsid.tbl` attivo è ubicato nella directory

`MQS_ROOT:[MQM.CONV.TABLE]`

**Conversione di dati predefinita:** Per implementare la conversioni di dati predefinita, modificare il file `ccsid.tbl` per specificare un CCSID EBCDIC e un CCSID ASCII predefiniti e inoltre per specificare i CCSID predefiniti. Le istruzioni a tale scopo sono incluse nel file.

Se si aggiorna `ccsid.tbl` per implementare la conversione di dati predefinita, Queue Manager dovrà essere riavviato prima che le modifiche divengano effettive.

Il processo di conversione di dati predefinita è il seguente:

- Se la conversione tra i CCSID di origine e di destinazione non è supportata, ma i CCSID degli ambienti di origine e di destinazione sono entrambi EBCDIC o entrambi ASCII, i dati dei caratteri vengono spostati all'applicazione di destinazione senza conversione.
- Se un CCSID rappresenta un insieme di caratteri codificati ASCII e l'altro rappresenta un insieme di caratteri codificati EBCDIC, MQSeries convertirà i dati mediante i CCSID di conversione di dati predefinita indicati in `ccsid.tbl`.

**Nota:** l'utente dovrebbe provare a limitare i caratteri che vengono convertiti a quelli caratterizzati dagli stessi valori di codici nell'insieme di caratteri codificati specificato per il messaggio e nell'insieme di caratteri codificati predefinito. Se si utilizza solo l'insieme di caratteri valido per i nomi oggetto di MQSeries tale requisito sarà generalmente soddisfatto. Alcune eccezioni si verificano con i CCSID EBCDIC 290, 930, 1279 e 5026 utilizzati in Giappone, dove i caratteri minuscoli hanno codici diversi rispetto a quelli utilizzati in altri CCSID EBCDIC.

### **Conversione di messaggi in formati definiti dall'utente**

Non è possibile convertire messaggi in formati definiti dall'utente da un insieme di caratteri codificati dal Queue Manager. Se i dati in un formato definito dall'utente richiedono la conversione, occorre fornire un'uscita della conversione dei dati per ciascun formato simile. L'utilizzo dei CCSID predefiniti per la conversione di dati di caratteri in formati definiti dall'utente è sconsigliato, sebbene sia possibile. Per ulteriori informazioni sulla conversione di dati in formati definiti dall'utente e sulla scrittura di uscite di conversione di dati, consultare il manuale *MQSeries Application Programming Guide*.

## **Modifica del CCSID del Queue Manager**

Si consiglia di arrestare e riavviare il Queue Manager quando si modifica il CCSID del Queue Manager, utilizzando l'attributo CCSID del comando ALTER QMGR.

Ciò garantisce che tutte le applicazioni in esecuzione, inclusi il server di comandi e i programmi di canale, siano arrestate e riavviate.

Ciò è necessario, poiché qualsiasi applicazione, che sia in esecuzione quando viene modificato il CCSID del Queue Manager, continua a utilizzare il CCSID esistente.



---

## Capitolo 7. Protezione di oggetti di MQSeries

Questo capitolo descrive le funzioni del controllo di sicurezza in MQSeries per Compaq OpenVMS e come è possibile implementare tale controllo.

Esso contiene le seguenti sezioni:

- “Perché bisogna proteggere le risorse di MQSeries”
- “Prima di iniziare”
- “Comprendere Object Authority Manager” a pagina 86
- “Utilizzo dei comandi di OAM (Object Authority Manager)” a pagina 89
- “Istruzioni per OAM (Object Authority Manager)” a pagina 92
- “Comprendere le tabelle di specifica delle autorizzazioni” a pagina 96
- “Comprendere i file di autorizzazione” a pagina 102

---

### Perché bisogna proteggere le risorse di MQSeries

Poiché i Queue Manager di MQSeries gestiscono il trasferimento delle informazioni potenzialmente preziose, occorre salvarle mediante un sistema di autorizzazioni. Ciò garantisce che le risorse appartenenti a un Queue Manager e da questo gestite siano protette dall'accesso non autorizzato, che potrebbe condurre alla perdita o alla divulgazione delle informazioni. In un sistema protetto, è fondamentale che nessun utente o applicazione non autorizzati possano accedere a o modificare quanto segue:

- Connessioni a un Queue Manager.
- Accesso a oggetti di MQSeries, quali code, cluster, canali e processi.
- Comandi di gestione del Queue Manager, inclusi i comandi MQSC e PCF.
- Accesso a messaggi di MQSeries.
- Informazioni contestuali associate ai messaggi.

Occorre sviluppare i propri criteri per quanto riguarda i particolari utenti che accederanno a determinate risorse.

---

### Prima di iniziare

Tutte le risorse del Queue Manager eseguite con VMS Rights Identifier:

MQM

Questo identificativo diritti viene creato durante l'installazione di MQSeries ed è **necessario** concedere tale attributo di risorse a tutti gli utenti che dovranno controllare le risorse di MQSeries.

### ID utente in MQSeries per Compaq OpenVMS con MQM identificativo risorse

Se l'ID utente detiene l'MQM identificativo diritti di OpenVMS, si dispone di tutte le autorizzazioni per tutte le risorse di MQSeries. L'ID utente *deve* detenere l'identificativo diritti MQM di OpenVMS per essere in grado di utilizzare tutti i comandi di controllo di MQSeries per Compaq OpenVMS tranne **crmtmqcvx**. In particolare, quest'autorizzazione è necessaria per:

- Utilizzare il comando **runmqsc** per eseguire comandi MQSC.

## Prima di iniziare

- Gestire le autorizzazioni su MQSeries per Compaq OpenVMS mediante il comando **setmqaut**.

Se si inviano comandi di canale ai Queue Manager su un sistema remoto, occorre verificare che il proprio ID utente contenga l'MQM identificativo diritti di OpenVMS sul sistema di destinazione. Per un elenco di comandi PCF e comandi di canale MQSC, consultare la sezione "Sicurezza del comando di canale" a pagina 95.

Inoltre, l'installazione di MQSeries crea un identificativo MQS\_SERVER. Questa è la proprietà concessa del dominio di risorse in cui VMS conserva le informazioni di blocco per MQSeries. Per impostazione predefinita, le autorizzazioni di accesso a tale identificativo sono concesse agli utenti che:

- Fanno parte dello stesso gruppo utenti di MQM, o
- Sono utenti del sistema, o
- Dispongono dell'insieme di privilegi SYSPRV, SYSLCK o BYPASS

Per consentire ad altri utenti di accedere alle risorse MQ, occorre verificare che l'identificativo MQS\_SERVER disponga dell'appropriato privilegio WORLD eseguendo il comando:

```
SET SECURITY/CLASS=RESOURCE [MQS_SERVER] /PROTECTION=(W:RWL)
```

**Nota:** ciò non è fondamentale affinché il proprio ID utente contenga l'MQM identificativo diritti per emettere:

- comandi PCF, inclusi i PCF Escape, da un programma di gestione
- chiamate MQI da un programma applicativo

## Per ulteriori informazioni

Per ulteriori informazioni su:

- insiemi di comandi di MQSeries per Compaq OpenVMS, consultare il "Capitolo 2. Introduzione alla gestione di MQSeries" a pagina 19.
- comandi di controllo di MQSeries per Compaq OpenVMS, consultare il "Capitolo 17. Comandi di controllo di MQSeries" a pagina 251.
- comandi PCF e Escape PCF, consultare il manuale *MQSeries Programmable System Management*.
- chiamate MQI, consultare i manuali *MQSeries Application Programming Guide* e *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa*.

---

## Comprendere Object Authority Manager

Per impostazione predefinita, l'accesso alle risorse del Queue Manager viene controllato tramite un componente installabile del servizio di autorizzazione. Tale componente è formalmente denominato Object Authority Manager (OAM) per MQSeries per Compaq OpenVMS. Esso è fornito con MQSeries per Compaq OpenVMS ed è automaticamente installato e attivato per ciascun Queue Manager creato, se non diversamente specificato. In questo capitolo, il termine OAM è adoperato per denotare Object Authority Manager fornito con questo prodotto.

OAM è un *componente installabile* del servizio di autorizzazione. La possibilità di installare il servizio OAM offre la flessibilità di:

- Sostituire OAM con il proprio componente di servizio di autorizzazione mediante l'interfaccia fornita.



- Aumentare le funzioni fornite da OAM con quelle del proprio componente di servizio di autorizzazione, usando sempre l'interfaccia fornita.
- Rimuovere o disabilitare OAM ed eseguire il programma senza alcun servizio di autorizzazione.

Per ulteriori informazioni sui servizi installabili, consultare il manuale *MQSeries Programmable System Management*.

OAM gestisce le autorizzazioni degli utenti per la manipolazione di oggetti MQSeries, inclusi code, definizioni di processo e canali. Esso fornisce inoltre un'interfaccia di comando tramite la quale è possibile concedere o revocare le autorizzazioni di accesso a un oggetto per uno specifico gruppo di utenti. OAM decide di concedere l'accesso a una risorsa e il Queue Manager si attiene a tale decisione. Se OAM non è in grado di prendere decisioni, il Queue Manager impedisce l'accesso a tale risorsa.

### Come funziona OAM

OAM funziona sfruttando le funzioni di sicurezza del sottostante sistema operativo OpenVMS. In particolare, OAM utilizza l'utente, le ID di gruppo e gli identificativi diritti di OpenVMS. Gli utenti possono accedere agli oggetti del Queue Manager solo se dispongono dell'autorizzazione richiesta.

### Gestione dell'accesso mediante gli identificativi di diritti

Nell'interfaccia dei comandi, si utilizza il termine *principal* piuttosto che ID utente. Il motivo per ciò è che le autorizzazioni concesse a un ID utente possono essere concesse anche ad altre entità, ad esempio un programma applicativo che emette chiamate MQI o un programma applicativo che emette comandi PCF. In tali casi, il principal associato al programma non è necessariamente l'ID utente che è stato utilizzato all'avvio del programma. Tuttavia, in questa discussione, i principal e le ID utente sono sempre ID utente di OpenVMS.

#### Gli identificativi di diritti e l'identificativo di diritti primario

La gestione dei permessi di accesso alle risorse di MQSeries si basa sugli *identificativi di diritti* di OpenVMS, ossia gli identificativi contenuti dai principal. Un principal può contenere un o più OpenVMS identificativi di diritti. Un gruppo è definito come insieme di tutti i principal a cui è stato concesso uno specifico identificativo di diritti.

OAM mantiene le autorizzazioni al livello degli identificativi di diritti piuttosto che dei principal individuali. La mappatura dei principal ai nomi dell'identificativo è svolta nell'ambito di OAM e le operazioni vengono compiute al livello dell'identificativo di diritti. È possibile, tuttavia, visualizzare le autorizzazioni di un principal individuale.

#### Quando un principal contiene più di un identificativo di diritti

Le autorizzazioni possedute da un principal rappresentano l'unione delle autorizzazioni di tutti gli identificativi di diritti che contiene, ossia i relativi diritti di elaborazione. Ogni qual volta un principal richiede l'accesso a una risorsa, OAM calcola tale unione e verifica l'autorizzazione associata. È possibile utilizzare il comando di controllo **setmqaut** per impostare le autorizzazioni per uno specifico principal, o identificativo.

**Nota:** qualsiasi modifica apportata mediante il comando **setmqaut** implica un effetto immediato, a meno che l'oggetto non sia in uso. In tal caso, la modifica avrà effetto quando l'oggetto sarà aperto successivamente. Tuttavia,

## Object authority manager

le modifiche apportate all'elenco di identificativi di diritti di un principal non diverranno effettive finché non si reimposta il Queue Manager, ossia dopo averlo arrestato e riavviato.

Le autorizzazioni associate a un principal sono memorizzate nella cache quando vengono calcolate da OAM. Qualsiasi modifica apportata alle autorizzazioni di un identificativo dopo che è stato memorizzato nella cache non viene riconosciuta fino al riavvio del Queue Manager. Evitare di modificare qualsiasi autorizzazione quando il Queue Manager è in esecuzione.

## Identificativo diritti predefinito

OAM riconosce un'impostazione predefinita a cui tutti gli utenti sono assegnati nominalmente. Tale gruppo è definito dallo pseudo identificativo diritti di 'NOBODY'. 'NOBODY' può essere utilizzato come se fosse un identificativo diritti valido per assegnare autorizzazioni mediante comandi di MQSeries. Per impostazione predefinita, nessuna autorizzazione è concessa a questo identificativo. Gli utenti con autorizzazioni specifiche possono ottenere l'accesso alle risorse di MQSeries tramite questo identificativo diritti.

## Le risorse che è possibile proteggere con OAM

Tramite OAM è possibile controllare:

- L'accesso a oggetti di MQSeries tramite MQI. Quando un programma applicativo tenta di accedere a un oggetto, OAM verifica se l'ID utente che sta effettuando la richiesta sia dotato dell'autorizzazione (tramite l'identificativo contenuto) per l'operazione richiesta.  
In particolare, ciò significa che le code e i messaggi in coda possono essere protetti dall'accesso non autorizzato.
- Permesso di utilizzare i comandi MQSC; solo i principal che contengono l'identificativo diritti MQM possono eseguire comandi di gestione del Queue Manager, ad esempio, per creare una coda.
- Permesso di utilizzare i comandi di controllo; solo i principal che contengono l'identificativo diritti MQM possono eseguire comandi di controllo, ad esempio, creare una Queue Manager, avviare un server di comandi o utilizzare **runmqsc**.
- Permesso di utilizzare comandi PCF.

È possibile concedere a diversi utenti tipi differenti di autorizzazione di accesso allo stesso oggetto. Ad esempio, per una coda specifica, gli utenti in possesso di un identificativo sono autorizzati ad eseguire operazioni put e get; gli utenti con un altro identificativo saranno autorizzati solo ad esplorare la coda (MQGET con opzione browse). In modo simile, gli utenti con identificativi potrebbero disporre di autorizzazioni put e get in una coda, ma non essere autorizzati a modificare o eliminare la coda.

## Utilizzo degli identificativi diritti per le autorizzazioni

L'utilizzo degli identificativi, piuttosto che dei principal individuali, per l'autorizzazione riduce la quantità di gestione richiesta. Di norma, è richiesto un tipo di accesso particolare da più principal. Ad esempio, si potrebbe definire un identificativo consistente in utenti finali che desiderano eseguire un'applicazione particolare. I nuovi utenti potrebbero ottenere l'accesso semplicemente concedendo l'identificativo appropriato al proprio ID utente di OpenVMS.

Provare a tenere più basso possibile il numero di identificativi. Ad esempio, si consiglia di iniziare dividendo i principal in un gruppo per gli utenti delle applicazioni e in un gruppo per gli amministratori.

### Disabilitazione di OAM (Object Authority Manager)

Per impostazione predefinita, OAM è abilitato. È possibile disabilitarlo impostando il nome logico MQSNOAUT prima di creare il Queue Manager, come segue:

```
$ DEFINE/SYSTEM MQSNOAUT TRUE
```

Tuttavia, in tal caso non sarà possibile, generalmente, riavviare OAM in seguito. Un miglior approccio consiste nel mantenere OAM abilitato e assicurarsi che tutti gli utenti e le applicazioni dispongano dell'accesso tramite un ID utente appropriato.

È inoltre possibile disabilitare OAM a scopo di prova solo rimuovendo la voce del servizio di autorizzazione nel file di configurazione del Queue Manager (qm.ini).

---

### Utilizzo dei comandi di OAM (Object Authority Manager)

OAM fornisce un'interfaccia di comandi per concedere e revocare le autorizzazioni. Prima di poter utilizzare tali comandi, occorre essere adeguatamente autorizzati il proprio ID utente deve contenere l'identificativo diritti MQM di OpenVMS. Tale identificativo dovrebbe essere stato impostato durante l'installazione del prodotto.

Se l'ID utente è dotato di identificativo MQM, si dispone dell'autorizzazione di gestione del Queue Manager. Ciò significa che si è autorizzati a emettere qualsiasi richiesta o comando MQI dal proprio ID utente.

OAM fornisce due comandi che è possibile richiamare dal DCL di OpenVMS per gestire le autorizzazioni degli utenti. Questi sono:

- **setmqaut** (imposta o reimposta l'autorizzazione)
- **dspmqa** (Display authority)

La verifica dell'autorizzazione di verifica nelle seguenti chiamate: MQCONN, MQOPEN, MQPUT1 e MQCLOSE.

La verifica dell'autorizzazione viene eseguita solo alla prima istanza di una qualsiasi chiamata e l'autorizzazione non viene corretta finché l'oggetto non viene reimpostato (ovvero, chiuso e riaperto).

Di conseguenza, qualsiasi modifica apportata all'autorizzazione di un oggetto che utilizza **setmqaut** non diverrà effettiva finché non si reimposta l'oggetto.

### Cosa specificare durante l'utilizzo dei comandi di OAM

I comandi di autorizzazione si applicano allo specifico Queue Manager; se non si specifica un Queue Manager, sarà utilizzato il Queue Manager predefinito. Su tali comandi, è necessario specificare unicamente l'oggetto, cioè il nome e il tipo dell'oggetto. Occorre inoltre specificare il nome del principal o dell'identificativo a cui si applica l'autorizzazione.

#### Elenchi di autorizzazioni

Specificare un elenco di autorizzazioni sul comando **setmqaut**. Questo è un modo breve per specificare se l'autorizzazione deve essere concessa o revocata e a quali

## Utilizzo dei comandi di OAM

risorse si applica l'autorizzazione. Ciascuna autorizzazione nell'elenco è specificata come parola chiave a caratteri minuscoli, preceduta da un segno + o -. Utilizzare un segno + per aggiungere l'autorizzazione specificata o un segno - per rimuoverla. È possibile specificare qualsiasi numero di autorizzazioni in un unico comando. Ad esempio:

```
+browse -get +put
```

## Utilizzo del comando setmqaut

A condizione di disporre l'autorizzazione richiesta, è possibile utilizzare il comando **setmqaut** per concedere o revocare l'autorizzazione di un principal o identificativo diritti ad accedere a un particolare oggetto. Il seguente esempio illustra l'utilizzo del comando **setmqaut**:

```
setmqaut -m "saturn.queue.manager" -t queue -n RED.LOCAL.QUEUE -g GROUPA +browse -get +put
```

In questo esempio:

Questo termine...	Specifica il...
<b>saturn.queue.manager</b>	Nome del Queue Manager.
<b>queue</b>	Tipo di oggetto.
<b>RED.LOCAL.QUEUE</b>	Nome dell'oggetto.
<b>GROUPA</b>	ID del gruppo a cui concedere le autorizzazioni.
<b>+browse -get +put</b>	Elenco di autorizzazioni per la coda specificata. Non devono esserci spazi tra i segni '+' o '-' e la parola chiave.

L'elenco di autorizzazioni specifica le autorizzazioni da concedere, dove:

Questo termine...	Effettua la seguente operazione...
<b>+browse</b>	Aggiunge l'autorizzazione per esplorare (MQGET con opzione browse) messaggi nella coda.
<b>-get</b>	Rimuove l'autorizzazione per ricevere (MQGET) messaggi dalla coda.
<b>+put</b>	Aggiunge l'autorizzazione per inserire (MQPUT) messaggi in coda.

Ciò significa che le applicazioni avviate con l'ID utente che contiene l'identificativo di OpenVMS GROUPA dispongono di tali autorizzazioni.

È possibile specificare uno o più principal e, contemporaneamente, uno o più identificativi. Ad esempio, il comando seguente revoca l'autorizzazione per il comando put sulla coda MyQueue al principal FVUSER e agli identificativi GROUPA e GROUPB.

```
setmqaut -m "saturn.queue.manager" -t queue -n "MyQueue" -p FVUSER -g GROUPA -g GROUPB -put
```

**Nota:** questo comando revoca inoltre l'autorizzazione put per tutti gli identificativi diritti contenuti in FVUSER, cioè tutti i gruppi a cui FVUSER appartiene.

Per una definizione formale del comando e della relativa sintassi, consultare la sezione "setmqaut (Imposta/reimposta l'autorizzazione)" a pagina 310.

### Comandi di autorizzazione e servizi installabili

Il comando **setmqaut** assume un parametro supplementare che specifica il nome del componente di servizio installabile a cui si applica l'aggiornamento. È necessario specificare tale parametro se si dispone di più componenti installabili in esecuzione simultaneamente. Per impostazione predefinita, non risulta tale

## Utilizzo dei comandi di OAM

situazione. Se il parametro viene omesso, l'aggiornamento viene effettuato sul primo servizio installabile di quel tipo, se disponibile. Il servizio predefinito è OAM, che è incluso.

## Autorizzazioni di accesso

Le autorizzazioni definite dall'elenco di autorizzazioni associato al comando **setmqaut** possono essere divise in categorie come segue:

- Autorizzazioni relative alle chiamate MQI
- Autorizzazioni relative ai comandi di gestione
- Autorizzazioni contestuali
- Autorizzazioni generali, ossia, per chiamate MQI, per i comandi o per entrambi

Ciascuna autorizzazione è specificata da una parola chiave utilizzata assieme ai comandi **setmqaut** e **dspmqaut**. Questi comandi sono descritti nella sezione "setmqaut (Imposta/reimposta l'autorizzazione)" a pagina 310.

## Visualizzazione del comando di autorizzazione

È possibile utilizzare il comando **dspmqaut** per visualizzare le autorizzazioni che uno specifico principal o identificativo per un oggetto particolare. I flag possiedono lo stesso significato di quelli nel comando **setmqaut**. L'autorizzazione può essere visualizzata per un solo identificativo o principal per volta. Consultare la sezione "dspmqaut (Display Authority)" a pagina 268 per una specifica formale di questo comando.

Ad esempio, il seguente comando visualizza le autorizzazioni concesse al gruppo GpAdmin per accedere a una definizione di processo denominata Annuities sul Queue Manager QueueMan1.

```
dspmqaut -m "QueueMan1" -t process -n "Annuities" -g "GpAdmin"
```

Le parole chiave visualizzate come risultato di questo comando identificano le autorizzazioni che sono attive.

---

## Istruzioni per OAM (Object Authority Manager)

Alcune operazioni sono particolarmente riservate e dovrebbero essere limitate a utenti privilegiati. Ad esempio:

- Avvio e arresto di Queue Manager.
- Accesso a determinate code speciali, quali le code di trasmissione o la coda di comando SYSTEM.ADMIN.COMMAND.QUEUE.
- Programmi che utilizzano opzioni contestuali MQI complete.
- In generale, la creazione e la copia di code di applicazione.

## ID utente

Lo speciale MQM ID utente, creato nel corso dell'installazione del prodotto, si intende per l'utilizzo esclusivo del prodotto. Non dovrebbe mai essere disponibile agli utenti non privilegiati.

L'ID utente utilizzato per le verifiche di autorizzazione, associato a un processo MQ, è l'ID utente di OpenVMS.

## Directory del Queue Manager

La directory contenente code e altri dati del Queue Manager è per l'uso privato del prodotto. Gli oggetti presenti in questa directory dispongono di autorizzazioni utente di OpenVMS correlate alle rispettive autorizzazioni di OAM. Tuttavia, non utilizzare comandi standard di OpenVMS per concedere o revocare autorizzazioni alle risorse MQI poiché:

- Gli oggetti di MQSeries non sono necessariamente identici al corrispondente nome dell'oggetto di sistema. Consultare la sezione "Analisi dei nomi di file MQSeries" a pagina 21 per ulteriori informazioni in proposito.
- Tutti gli oggetti appartengono all'ID MQM risorsa.

## Code

L'autorizzazione per l'accesso a una coda dinamica si basa su quella della coda modello da cui deriva, ma non è necessariamente uguale ad essa. Consultare la nota 1 a pagina 99 per ulteriori informazioni.

Per code di alias e code remote, l'autorizzazione è quella dell'oggetto stesso, non la coda in cui l'alias o la coda remota si trasforma. Di conseguenza, è possibile autorizzare un ID utente ad accedere ad un alias di coda che si trasforma in una coda locale per accedere alla quale l'ID utente non dispone di permessi.

Occorre limitare l'autorizzazione affinché crei code per utenti privilegiati. In caso contrario, alcuni utenti potrebbero aggirare il normale controllo di accesso creando un alias.

## Autorizzazione per utente alternativo

L'autorizzazione dell'utente alternativo controlla se un ID utente può utilizzare l'autorizzazione di un altro ID utente durante l'accesso a un oggetto di MQSeries. Ciò è essenziale laddove un server riceva richieste da un programma e il server desidera assicurarsi che il programma disponga dell'autorizzazione necessaria per la richiesta. Il server potrebbe disporre dell'autorizzazione necessaria, ma occorre sapere se il programma dispone dell'autorizzazione per le azioni richieste.

Ad esempio:

- Un programma del server in esecuzione sotto l'ID utente PAYSERV recupera un messaggio di richiesta da una coda che era stata inserita in coda dall'ID utente USER1.
- Quando il programma del server riceve il messaggio di richiesta, elabora la richiesta e inserisce la risposta nella coda di risposta specificata con il messaggio di richiesta.
- Invece di utilizzare il proprio ID utente (PAYSERV) per autorizzare l'apertura della coda di risposta, il server potrà specificare alcuni altri ID utente, in tal caso, USER1. In questo esempio, è possibile utilizzare l'autorizzazione utente alternativo per controllare se a PAYSERV è concesso specificare USER1 come ID utente alternativo quando apre la coda di risposta.

L'ID utente alternativo è specificato nel campo *AlternateUserId* della descrizione dell'oggetto.

**Nota:** è possibile utilizzare ID utente alternativi su qualsiasi oggetto di MQSeries. L'utilizzo di un ID utente alternativo non influisce sull'ID utente utilizzato da altri gestori di risorse.



### Autorizzazione contestuale

Il contesto è l'informazione applicata a un messaggio particolare ed è contenuta nella descrizione del messaggio, MQMD, che fa parte del messaggio. Le informazioni contestuali si dividono in due sezioni:

#### Sezione Identità

Questa parte specifica il mittente del messaggio. Consiste nei seguenti campi:

- *UserIdentifier*
- *AccountingToken*
- *ApplIdentityData*

#### Sezione Origine

Questa sezione specifica da dove proviene il messaggio e quando è stato inserito in coda. Consiste nei seguenti campi:

- *PutApplType*
- *PutApplName*
- *PutDate*
- *PutTime*
- *ApplOriginData*

Le applicazioni possono specificare i dati contestuali quando viene eseguita una chiamata MQOPEN o MQPUT. Questi dati possono essere generati dall'applicazione, trasferiti da un altro messaggio o generati dal Queue Manager per impostazione predefinita. Ad esempio, i dati contestuali possono essere utilizzati dai programmi del server per verificare l'identità del richiedente, controllando se il messaggio proviene da un'applicazione, eseguita sotto un ID utente autorizzato.

Un programma di server può utilizzare *UserIdentifier* per determinare l'ID utente di un utente alternativo.

Utilizzare l'autorizzazione contestuale per controllare se l'utente può specificare qualsiasi opzione di contesto su qualsiasi chiamata MQOPEN o MQPUT1. Per informazioni sulle opzioni di contesto, consultare il manuale *MQSeries Application Programming Guide*. Per un'illustrazione dei campi di descrizione del messaggio relativi al contesto, consultare il manuale *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa*.

### Considerazioni sulla sicurezza remota

Per la sicurezza remota, occorre considerare:

#### Autorizzazione put

Per la sicurezza in tutti i Queue Manager è possibile specificare l'autorizzazione put utilizzata quando un canale riceve un messaggio inviato da un altro Queue Manager.

Specificare l'attributo di canale PUTAUT come segue:

**DEF** ID utente predefinito. Questo è l'ID utente sotto cui è in esecuzione l'agente del canale di messaggio.

**CTX** L'ID utente nel contesto del messaggio.

#### Code di trasmissione

I Queue Manager inseriscono automaticamente messaggi in una coda di trasmissione; non è richiesta alcuna autorizzazione speciale a tale scopo.



Tuttavia, l'inserimento di un messaggio direttamente in una coda di trasmissione richiede un'autorizzazione speciale; consultare la Tabella 2 a pagina 97.

### Uscite di canale

Le uscite dal canale possono essere utilizzate per una maggiore sicurezza.

Per ulteriori informazioni, consultare il manuale *MQSeries Intercommunication*.

## Sicurezza del comando di canale

I comandi di canale possono essere emessi come comandi PCF, tramite comandi MQSC, MQAI e comandi di controllo.

### Comandi PCF

È possibile emettere comandi di canale PCF inviando un messaggio PCF a SYSTEM.ADMIN.COMMAND.QUEUE su un sistema OpenVMS remoto. L'ID utente, come specificato nella descrizione del messaggio PCF, deve contenere l'identificativo diritti MQM sul sistema di destinazione. Tali comandi sono:

- *ChangeChannel*
- *CopyChannel*
- *CreateChannel*
- *DeleteChannel*
- *PingChannel*
- *ResetChannel*
- *StartChannel*
- *StartChannelInitiator*
- *StartChannelListener*
- *StopChannel*
- *ResolveChannel*

Consultare il manuale *MQSeries Programmable System Management* per i requisiti di sicurezza PCF.

### Comandi canale MQSC

È possibile emettere comandi di canale MQSC a un sistema remoto OpenVMS inviando il comando direttamente in un messaggio escape PCF oppure emettendo il comando mediante **runmqsc** in modo indiretto. L'ID utente come specificato nella descrizione del messaggio PCF associato deve contenere l'identificativo diritti MQM sul sistema di destinazione (i comandi PCF sono impliciti nei comandi MQSC emessi da **runmqsc** in modo indiretto). Tali comandi sono:

- ALTER CHANNEL
- DEFINE CHANNEL
- DELETE CHANNEL
- PING CHANNEL
- RESET CHANNEL
- START CHANNEL
- START CHINIT
- START LISTENER
- STOP CHANNEL
- RESOLVE CHANNEL

Per i comandi MQSC emessi dal comando **runmqsc**, l'ID utente nel messaggio PCF è di norma quello dell'attuale utente.

### Comandi di controllo per i canali

Per quanto riguarda i comandi di controllo dei canali, l'ID utente che li emette deve contenere un identificativo diritti MQM. Tali comandi sono:

## Istruzioni per OAM

- **runmqchi** (esegue channel initiator)
- **runmqchl** (esegue channel)

---

## Comprendere le tabelle di specifica delle autorizzazioni

Le tabelle di specifica delle autorizzazioni che iniziano a pagina 97 definiscono in maniera precisa il funzionamento delle autorizzazioni e le restrizioni applicate. Le tabelle si applicano alle seguenti situazioni:

- Applicazioni che emettono chiamate MQI.
- Programmi di gestione che emettono comandi MQSC come PCF escape.
- Programmi di gestione che emettono comandi PCF.

In questa sezione, le informazioni sono presentate come un insieme di tabelle che specificano quanto segue:

### Azione da eseguire

Opzione MQI, comando MQSC o comando PCF.

### Oggetto di controllo accessi

Coda, processo o Queue Manager.

### Autorizzazione richiesta

Espressa come una costante 'MQZAO\_'.

Nelle tabelle, le costanti precedute dal prefisso MQZAO\_ corrispondono alle parole chiave nell'elenco di autorizzazioni per il comando **setmqaut** relativo alla particolare entità. Ad esempio, MQZAO\_BROWSE corrisponde alla parola chiave +browse; similmente, la parola chiave MQZAO\_SET\_ALL\_CONTEXT corrisponde alla parola chiave +setall e così via. Queste costanti sono definite nel file di intestazione cmqzc.h, incluso nel prodotto. Consultare la sezione "Cosa contengono i file di autorizzazione" a pagina 103 per ulteriori informazioni.

## Autorizzazioni MQI

Un'applicazione può emettere solo determinate chiamate e opzioni MQI se all'identificativo utente sotto il quale è in esecuzione (o le cui autorizzazioni è in grado di assumere) è stata concessa la relativa autorizzazione.

Quattro chiamate MQI potrebbero richiedere verifiche di autorizzazione: MQCONN, MQOPEN, MQPUT1 e MQCLOSE.

Per MQOPEN e MQPUT1, la verifica di autorizzazione viene effettuata sul nome dell'oggetto aperto e non sul nome o i nomi, risultanti quando un nome è stato risolto. Ad esempio, è possibile concedere a un'applicazione l'autorizzazione per aprire un alias di coda senza che questa disponga dell'autorizzazione per aprire la coda base in cui si trasforma l'alias. Di regola la verifica viene effettuata sulla prima definizione incontrata durante il processo di risoluzione del nome che non sia un alias di Queue Manager, a meno che la relativa definizione non sia aperta direttamente; in altre parole, il nome appare nel campo *ObjectName* della descrizione dell'oggetto. L'autorizzazione è sempre necessaria per il particolare oggetto che viene aperto; in alcuni casi sono necessarie autorizzazioni supplementari indipendenti dalla coda che si ottengono tramite l'autorizzazione per l'accesso all'oggetto del Queue Manager.

La Tabella 2 a pagina 97 riassume le autorizzazioni necessarie per ciascuna chiamata.

## Tabelle di specifica delle autorizzazioni

Tabella 2. Autorizzazione di sicurezza necessaria per chiamate MQI

Autorizzazione richiesta per:	Oggetto coda (1)	Oggetto processo	Oggetto gestore code	Namelist
Opzione MQCONN	Non applicabile	Non applicabile	MQZAO_CONNECT	Non applicabile
Opzione MQOPEN				
MQOO_INQUIRE	MQZAO_INQUIRE (2)	MQZAO_INQUIRE (2)	MQZAO_INQUIRE (2)	MQZAO_INQUIRE (2)
MQOO_BROWSE	MQZAO_BROWSE	Non applicabile	Nessuna verifica	Non applicabile
MQOO_INPUT_*	MQZAO_INPUT	Non applicabile	Nessuna verifica	Non applicabile
MQOO_SAVE_ALL_CONTEXT (3)	MQZAO_INPUT	Non applicabile	Nessuna verifica	Non applicabile
MQOO_OUTPUT (coda normale) (4)	MQZAO_OUTPUT	Non applicabile	Nessuna verifica	Non applicabile
MQOO_PASS_IDENTITY_CONTEXT (5)	MQZAO_PASS_IDENTITY_CONTEXT	Non applicabile	Nessuna verifica	Non applicabile
MQOO_PASS_ALL_CONTEXT (5, 6)	MQZAO_PASS_ALL_CONTEXT	Non applicabile	Nessuna verifica	Non applicabile
MQOO_SET_IDENTITY_CONTEXT (5, 6)	MQZAO_SET_IDENTITY_CONTEXT	Non applicabile	MQZAO_SET_IDENTITY_CONTEXT (7)	Non applicabile
MQOO_SET_ALL_CONTEXT (5, 8)	MQZAO_SET_ALL_CONTEXT	Non applicabile	MQZAO_SET_ALL_CONTEXT (7)	Non applicabile
MQOO_OUTPUT (coda di trasmissione) 9)	MQZAO_SET_ALL_CONTEXT	Non applicabile	MQZAO_SET_ALL_CONTEXT (7)	Non applicabile
MQOO_SET	MQZAO_SET	Non applicabile	Nessuna verifica	Non applicabile
MQOO_ALTERNATE_USER_AUTHORITY	(10)	(10)	MQZAO_ALTERNATE_USER_AUTHORITY (10, 11)	(10)
Opzione MQPUT1				
MQPMO_PASS_IDENTITY_CONTEXT	MQZAO_PASS_IDENTITY_CONTEXT (12)	Non applicabile	Nessuna verifica	Non applicabile
MQPMO_PASS_ALL_CONTEXT	MQZAO_PASS_ALL_CONTEXT (12)	Non applicabile	Nessuna verifica	Non applicabile
MQPMO_SET_IDENTITY_CONTEXT	MQZAO_SET_IDENTITY_CONTEXT (12)	Non applicabile	MQZAO_SET_IDENTITY_CONTEXT (7)	Non applicabile
MQPMO_SET_ALL_CONTEXT	MQZAO_SET_ALL_CONTEXT (12)	Non applicabile	MQZAO_SET_ALL_CONTEXT (7)	Non applicabile
(Coda di trasmissione) (9)	MQZAO_SET_ALL_CONTEXT	Non applicabile	MQZAO_SET_ALL_CONTEXT (7)	Non applicabile

## Tabelle di specifica delle autorizzazioni

Tabella 2. Autorizzazione di sicurezza necessaria per chiamate MQI (Continua)

Autorizzazione richiesta per:	Oggetto coda (1)	Oggetto processo	Oggetto gestore code	Namelist
MQPMO_ALTERNATE_USER_AUTHORITY	(13)	Non applicabile	MQZAO_ALTERNATE_USER_AUTHORITY (11)	Non applicabile
Opzione MQCLOSE				
MQCO_DELETE	MQZAO_DELETE (14)	Non applicabile	Non applicabile	Non applicabile
MQCO_DELETE_PURGE	MQZAO_DELETE (14)	Non applicabile	Non applicabile	Non applicabile

### Note specifiche:

1. Se una coda modello viene aperta:
  - È necessaria l'autorizzazione MQZAO\_DISPLAY per la coda modello, oltre a qualsiasi altra autorizzazione (anche per la coda modello) che sia richiesta per le opzioni aperte specificate.
  - L'autorizzazione MQZAO\_CREATE non è necessaria per creare la coda dinamica.
  - All'identificativo utente utilizzato per aprire la coda modello sono automaticamente concesse tutte le autorizzazioni specifiche della coda (equivalenti a MQZAO\_ALL) per la coda dinamica creata.
2. Vengono verificati la coda, il processo, la namelist o il Queue Manager, a seconda del tipo di oggetto aperto.
3. Occorre specificare anche MQOO\_INPUT\_\*. Ciò è valido per una coda locale, modello o di alias.
4. Questa verifica viene eseguita per tutti i casi di output, tranne il caso specificato nella nota 9.
5. Occorre specificare anche MQOO\_OUTPUT.
6. Anche MQOO\_PASS\_IDENTITY\_CONTEXT è implicato in questa opzione.
7. Quest'autorizzazione è richiesta sia dall'oggetto Queue Manager che dalla coda particolare.
8. Anche MQOO\_PASS\_IDENTITY\_CONTEXT, MQOO\_PASS\_ALL\_CONTEXT e MQOO\_SET\_IDENTITY\_CONTEXT sono implicati in questa opzione.
9. Questa verifica viene eseguita per una coda locale o modello che dispone di un attributo di coda *Usage* di MQUS\_TRANSMISSION e viene aperta direttamente per l'output. Ciò non si applica se una coda remota viene aperta (specificando i nomi del Queue Manager remoto e della coda remota oppure specificando il nome di una definizione locale della coda remota).
10. Occorre inoltre specificare almeno uno fra MQOO\_INQUIRE (per qualsiasi tipo di oggetto) o (per le code) MQOO\_BROWSE, MQOO\_INPUT\_\*, MQOO\_OUTPUT o

## Tabelle di specifica delle autorizzazioni

MQOO\_SET. La verifica eseguita, come per le altre opzioni specificate, utilizza l'identificativo utente alternativo fornito per l'autorizzazione all'oggetto dal nome specifico e l'autorizzazione dell'applicazione corrente per la verifica MQZAO\_ALTERNATE\_USER\_IDENTIFIER.

11. Questa autorizzazione consente a qualsiasi *AlternateUserId* di essere specificato.
12. Viene eseguita anche una verifica MQZAO\_OUTPUT, se la coda non dispone di un attributo di coda *Usage* di MQUS\_TRANSMISSION.
13. La verifica eseguita, come per le altre opzioni specificate, utilizza l'identificativo utente alternativo fornito per l'autorizzazione alla coda dal nome specifico e l'autorizzazione della corrente applicazione per la verifica MQZAO\_ALTERNATE\_USER\_IDENTIFIER.
14. La verifica viene eseguita solo se entrambi le seguenti affermazioni sono vere:
  - Si sta chiudendo ed eliminando una coda dinamica permanente.
  - La coda non è stata creata da MQOPEN che ha restituito l'handle dell'oggetto in uso.

In caso contrario, non vi sarà alcuna verifica.

### Note generali:

1. L'autorizzazione speciale MQZAO\_ALL\_MQI include tutte le seguenti che sono relative al tipo di oggetto:
  - MQZAO\_CONNECT
  - MQZAO\_INQUIRE
  - MQZAO\_SET
  - MQZAO\_BROWSE
  - MQZAO\_INPUT
  - MQZAO\_OUTPUT
  - MQZAO\_PASS\_IDENTITY\_CONTEXT
  - MQZAO\_PASS\_ALL\_CONTEXT
  - MQZAO\_SET\_IDENTITY\_CONTEXT
  - MQZAO\_SET\_ALL\_CONTEXT
  - MQZAO\_ALTERNATE\_USER\_AUTHORITY
2. MQZAO\_DELETE (vedere nota la 14 a pagina 99) e MQZAO\_DISPLAY sono classificate come autorizzazioni di gestione. Esse non sono pertanto incluse in MQZAO\_ALL\_MQI.
3. 'Nessuna verifica' significa che non è stata eseguita alcuna verifica dell'autorizzazione.
4. 'Non applicabile' significa che la verifica dell'autorizzazione non è importante per questa operazione. Ad esempio, non è possibile emettere una chiamata MQPUT per un oggetto processo.

## Autorizzazioni di gestione

Queste autorizzazioni consentono a un utente di emettere comandi di gestione, ovvero un comando MQSC come messaggio PCF escape o come comando PCF stesso. Tali metodi consentono a un programma di inviare un comando di gestione come messaggio a un Queue Manager, affinché lo esegua per conto di tale utente.

## Tabelle di specifica delle autorizzazioni

### Autorizzazioni per comandi MQSC nei PCF escape

La Tabella 3 riassume le autorizzazioni necessarie per ciascun comando MQSC contenuto nel PCF Escape.

Tabella 3. Comandi MQSC e autorizzazioni di sicurezza necessari

(2)Autorizzazione richiesta per:	Oggetto coda	Oggetto processo	Oggetto gestore code	Namelist
Comando MQSC				
ALTER object	MQZAO_CHANGE	MQZAO_CHANGE	MQZAO_CHANGE	MQZAO_CHANGE
CLEAR QLOCAL	MQZAO_CLEAR	Non applicabile	Non applicabile	Non applicabile
DEFINE object NOREPLACE (3)	MQZAO_CREATE (4)	MQZAO_CREATE (4)	Non applicabile	MQZAO_CREATE (4)
DEFINE object REPLACE (3, 5)	MQZAO_CHANGE	MQZAO_CHANGE	Non applicabile	MQZAO_CHANGE
DELETE object	MQZAO_DELETE	MQZAO_DELETE	Non applicabile	MQZAO_DELETE
DISPLAY object	MQZAO_DISPLAY	MQZAO_DISPLAY	MQZAO_DISPLAY	MQZAO_DISPLAY

#### Note specifiche:

1. L'identificativo utente, sotto il quale il programma (ad esempio, **runmqsc**) che invia il comando è in esecuzione, deve disporre inoltre dell'autorizzazione MQZAO\_CONNECT per accedere al Queue Manager.
2. Vengono verificati la coda, il processo, la namelist o il Queue Manager, a seconda del tipo di oggetto.
3. Per i comandi DEFINE, anche l'autorizzazione MQZAO\_DISPLAY è necessaria per l'oggetto LIKE se specificato, o sull'oggetto SYSTEM.DEFAULT.xxx appropriato se LIKE viene omissso.
4. L'autorizzazione MQZAO\_CREATE non è specifica di un particolare oggetto o tipo di oggetto. La creazione di autorizzazioni è concessa a tutti gli oggetti, per un Queue Manager specificato, specificando un tipo di oggetto di QMGR sul comando SETMQAUT.
5. Ciò si applica se l'oggetto da sostituire è già effettivamente esistente. In caso contrario, la verifica è la stessa di DEFINE object NOREPLACE.

#### Note generali:

1. Per eseguire qualsiasi comando PCF, è necessario disporre dell'autorizzazione DISPLAY sul Queue Manager.
2. L'autorizzazione per eseguire un PCF escape dipende dal comando MQSC all'interno del testo del messaggio PCF escape.
3. 'Non applicabile' significa che la verifica dell'autorizzazione non è importante per questa operazione. Ad esempio, non è possibile emettere CLEAR QLOCAL su un oggetto Queue Manager.

### Autorizzazioni per comandi PCF

La Tabella 4 a pagina 101 riepiloga le autorizzazioni necessarie per ciascun comando PCF.

## Tabelle di specifica delle autorizzazioni

Tabella 4. Comandi PCF e autorizzazioni di sicurezza necessari

(2)Autorizzazione richiesta per:	Oggetto coda	Oggetto processo	Oggetto gestore code	Namelist
<b>comando PCF</b>				
Modifica oggetto	MQZAO_CHANGE	MQZAO_CHANGE	MQZAO_CHANGE	MQZAO_CHANGE
Cancella coda	MQZAO_CLEAR	Non applicabile	Non applicabile	Non applicabile
Copia oggetto (senza sostituire) 3)	MQZAO_CREATE (4)	MQZAO_CREATE (4)	Non applicabile	MQZAO_CREATE (4)
Copia oggetto (e sostituisci) 3, 6)	MQZAO_CHANGE	MQZAO_CHANGE	Non applicabile	MQZAO_CHANGE
Crea oggetto (senza sostituire) 5)	MQZAO_CREATE (4)	MQZAO_CREATE (4)	Non applicabile	MQZAO_CREATE (4)
Crea oggetto (e sostituisci) 5, 6)	MQZAO_CHANGE	MQZAO_CHANGE	Non applicabile	MQZAO_CHANGE
Elimina oggetto	MQZAO_DELETE	MQZAO_DELETE	Non applicabile	MQZAO_DELETE
Richiedi informazioni sull'oggetto	MQZAO_DISPLAY	MQZAO_DISPLAY	MQZAO_DISPLAY	MQZAO_DISPLAY
Richiedi informazioni sui nomi dell'oggetto	Nessuna verifica	Nessuna verifica	Nessuna verifica	Nessuna verifica
Reimposta le statistiche di coda	MQZAO_DISPLAY e MQZAO_CHANGE	Non applicabile	Non applicabile	Non applicabile

### Note specifiche:

1. Anche l'identificativo utente sotto il quale il programma che invia il comando è in esecuzione deve disporre dell'autorizzazione per connettersi al relativo gestore di code locali e per aprire la coda di gestione del comando per l'output.
2. Vengono verificati la coda, il processo, la namelist o il Queue Manager, a seconda del tipo di oggetto.
3. Per i comandi Copy, è necessaria l'autorizzazione MQZAO\_DISPLAY anche per l'oggetto From.
4. L'autorizzazione MQZAO\_CREATE non è specifica di un particolare oggetto o tipo di oggetto. La creazione di autorizzazioni è concessa a tutti gli oggetti, per un Queue Manager specificato, specificando un tipo di oggetto di QMGR sul comando SETMQAUT.
5. Per i comandi Create, è necessaria l'autorizzazione MQZAO\_DISPLAY anche per l'appropriato oggetto SYSTEM.DEFAULT.\*.
6. Ciò si applica se l'oggetto da sostituire è già esistente. In caso contrario, la verifica è la stessa di Copy or Create.

### Note generali:

1. Per eseguire qualsiasi comando PCF, è necessario disporre dell'autorizzazione DISPLAY sul Queue Manager.
2. L'autorizzazione speciale MQZAO\_ALL\_ADMIN include tutte le seguenti che sono relative al tipo di oggetto:
  - MQZAO\_CHANGE



## Tabelle di specifica delle autorizzazioni

- MQZAO\_CLEAR
- MQZAO\_DELETE
- MQZAO\_DISPLAY

MQZAO\_CREATE non è inclusa, perché non è specifica ad un particolare o tipo di oggetto.

3. 'Nessuna verifica' significa che non è stata eseguita alcuna verifica dell'autorizzazione.
4. 'Non applicabile' significa che la verifica dell'autorizzazione non è importante per questa operazione. Ad esempio, non è possibile utilizzare un comando **Clear Queue** su un oggetto di processo.

---

## Comprendere i file di autorizzazione

**Nota:** le informazioni contenute in questa sezione sono d'ausilio nella determinazione dei problemi. In circostanze normali, utilizzare i comandi di autorizzazione per visualizzare e modificare le informazioni di autorizzazione.

MQSeries per Compaq OpenVMS utilizza una struttura di file specifica per implementare la sicurezza. Non occorre effettuare alcuna operazione con questi file, tranne assicurarsi che tutti i file di autorizzazione siano protetti.

La sicurezza è implementata dai file di autorizzazione. Da tale prospettiva, esistono tre tipi di autorizzazione:

- Le autorizzazioni che si applicano a un singolo oggetto, ad esempio, l'autorizzazione di inserire un messaggio in una coda.
- Le autorizzazioni che si applicano a una classe di oggetti, ad esempio, l'autorizzazione per creare una coda.
- Le autorizzazioni che si applicano a tutte le classi di oggetti, ad esempio, l'autorizzazione per eseguire operazioni per conto di diversi utenti.

## Percorsi dei file di autorizzazione

Il percorso di un file di autorizzazione dipende dal tipo. Nello specificare un'autorizzazione per un oggetto, ad esempio, il Queue Manager crea i file di autorizzazione appropriati. Questo inserisce i file in una sottodirectory, il cui percorso è definito dal nome del Queue Manager, dal tipo di autorizzazione e, laddove appropriato, il nome dell'oggetto.

Non tutte le autorizzazioni si applicano direttamente alle istanze degli oggetti. Ad esempio, l'autorizzazione per creare un oggetto si applica alla classe di oggetti piuttosto che a una singola istanza. Inoltre, alcune autorizzazioni si applicano all'intero Queue Manager, ad esempio, l'autorizzazione dell'utente alternativo significa che un utente può assumere le autorizzazioni associate a un altro utente.

### Directory dell'autorizzazione

Per impostazione predefinita, le directory di autorizzazione, per un Queue Manager denominato saturn, sono:

**MQS\_ROOT: [MQM.QMGRS.SATURN.AUTH.QUEUES]**

File di autorizzazione per le code.

**MQS\_ROOT: [MQM.QMGRS.SATURN.AUTH.PROCDEF]**

File di autorizzazione per definizioni di processo.



**MQS\_ROOT: [MQM.QMGRS.SATURN.AUTH.QMANAGER]**

File di autorizzazione per il Queue Manager.

**MQS\_ROOT: [MQM.QMGRS.SATURN.AUTH]\$ACCLASS**

Autorizzazioni che si applicano a tutte le classi.

**MQS\_ROOT: [MQM.QMGRS.SATURN.AUTH.NAMELIST]**

Autorizzazioni che si applicano a tutte le namelist.

Nelle directory dell'oggetto, i file \$CLASS contengono le autorizzazioni relative all'intera classe.

**Nota:** Esiste una differenza tra \$CLASS (file di autorizzazioni che specifica un'autorizzazione per una classe in particolare) e \$ACCLASS (file di autorizzazioni che specifica le autorizzazioni per tutte le classi).

I percorsi dei file di autorizzazione oggetto si basano su quelli dell'oggetto stesso, in cui AUTH viene inserito davanti alla directory di tipo oggetto. È possibile utilizzare il comando **dspmqls** per visualizzare il percorso di un oggetto specificato.

Ad esempio, se il nome e il percorso di SYSTEM.DEFAULT.LOCAL.QUEUE è:

```
MQS_ROOT: [MQM.QMGRS.SATURN.QUEUES.SYSTEM$DEFAULT$LOCAL$QUEUE]
```

il nome e il percorso del file di autorizzazione corrispondente è:

```
MQS_ROOT: [MQM.QMGRS.SATURN.AUTH.QUEUES.SYSTEM$DEFAULT$LOCAL$QUEUE]
```

**Nota:** in questo caso, i nomi effettivi dei file associati alla coda non sono uguali al nome della coda stessa. Consultare la sezione "Analisi dei nomi di file MQSeries" a pagina 21 per ulteriori informazioni.

## Cosa contengono i file di autorizzazione

Le autorizzazioni di un identificativo particolare sono definite da un insieme di voci nel file di autorizzazione. Consultare la sezione "Comprendere i file di autorizzazione" a pagina 102 per ulteriori informazioni. Le autorizzazioni si applicano all'oggetto associato a questo file. Ad esempio:

```
groupb:  
  Authority=0x0040007
```

Questa voce definisce l'autorizzazione per l'identificativo GROUPB. La specifica dell'autorizzazione è costituita dall'unione degli schemi di bit individuali basati sulle seguenti assegnazioni:

## File di autorizzazione

Authorization keyword	Formal name	Hexadecimal Value
connect	MQZAO_CONNECT	0x00000001
browse	MQZAO_BROWSE	0x00000002
get	MQZAO_INPUT	0x00000004
put	MQZAO_OUTPUT	0x00000008
inq	MQZAO_INQUIRE	0x00000010
set	MQZAO_SET	0x00000020
passid	MQZAO_PASS_IDENTITY_CONTEXT	0x00000040
passall	MQZAO_PASS_ALL_CONTEXT	0x00000080
setid	MQZAO_SET_IDENTITY_CONTEXT	0x00000100
setall	MQZAO_SET_ALL_CONTEXT	0x00000200
altusr	MQZAO_ALTERNATE_USER_AUTHORITY	0x00000400
allmqi	MQZAO_ALL_MQI	0x000007FF
crt	MQZAO_CREATE	0x00010000
dlt	MQZAO_DELETE	0x00020000
dsp	MQZAO_DISPLAY	0x00040000
chg	MQZAO_CHANGE	0x00080000
clr	MQZAO_CLEAR	0x00100000
chgaut	MQZAO_AUTHORIZE	0x00800000
alladm	MQZAO_ALL_ADMIN	0x009E0000
none	MQZAO_NONE	0x00000000
all	MQZAO_ALL	0x009E07FF

Queste definizioni sono effettuate nel file di intestazione cmqzc.h. Nel seguente esempio, a GROUPB sono state concesse le autorizzazioni basate sul numero esadecimale 0x40007. Ciò corrisponde a:

MQZAO_CONNECT	0x00000001
MQZAO_BROWSE	0x00000002
MQZAO_INPUT	0x00000004
MQZAO_DISPLAY	0x00040000
-----	
Authority is:	0x00040007

Questi diritti di accesso significano che chiunque nel GROUPB può emettere le chiamate MQI:

MQCONN  
MQGET (con browse)

Essi dispongono anche dell'autorizzazione DISPLAY per l'oggetto associato a tale file di autorizzazione.

### File di autorizzazione classe

I *file di autorizzazione classe* contengono le autorizzazioni relative all'intera classe. Tali file sono denominati "\$CLASS" ed esistono nella stessa directory dei file per gli oggetti specifici. La voce MQZAO\_CRT nel file \$CLASS fornisce l'autorizzazione per creare un oggetto nella classe. Questa è l'unica autorizzazione di classe.

### Tutti i file di autorizzazione classe

Il *file di autorizzazione per tutte le classi* contiene autorizzazioni che si applicano a un intero Queue Manager. Questo file è denominato \$AClass ed esiste nella sottodirectory auth del Queue Manager.

Le seguenti autorizzazioni si applicano all'intero Queue Manager e sono contenute nel file di autorizzazione per tutte le classi:

**La voce... autorizza a...**

**MQZAO\_ALTERNATE\_USER\_AUTHORITY**

Assumere l'identità di un altro utente durante l'interazione con oggetti MQSeries.

**MQZAO\_SET\_ALL\_CONTEXT**

Impostare il contesto di un messaggio quando si emette MQPUT.

**MQZAO\_SET\_IDENTITY\_CONTEXT**

Impostare il contesto di identità di un messaggio quando si emette MQPUT.

## Gestione dei file di autorizzazione

Ecco alcune indicazioni da tenere in considerazione nel gestire i propri file di autorizzazione:

1. È necessario assicurarsi che i file di autorizzazione siano protetti e non accessibili alla scrittura da parte di utenti generali non attendibili. Consultare la sezione "Autorizzazioni per i file di autorizzazione".
2. Per poter riprodurre i propri file di autorizzazione, assicurarsi di eseguire almeno una delle seguenti operazioni:
  - Eseguire la copia di riserva della sottodirectory AUTH dopo qualsiasi aggiornamento significativo
  - Conservare i file di comando DCL che contengono i comandi utilizzati
3. È possibile copiare e modificare i file di autorizzazione. Tuttavia, non dovrebbe essere di norma necessario crearli o ripararli manualmente. Qualora dovesse verificarsi un'emergenza, utilizzare le informazioni qui fornite per recuperare i file di autorizzazione persi o danneggiati.

### Autorizzazioni per i file di autorizzazione

I file di autorizzazione devono essere leggibili da qualsiasi principal. Tuttavia, solo al responsabile di sistema e all'utente con l'identificativo MQM dovrebbe essere concesso di aggiornare questi file.

I permessi sui file di autorizzazione, creati da OAM, sono:

S:RWD, O:RWD, G:RWD, W:R (ID=MQM, ACCESS=R+W+E+D+C)

Non modificare questi permessi senza aver revisionato attentamente la presenza di eventuali sezioni non protette.

Per modificare le autorizzazioni mediante il comando fornito con MQSeries per Compaq OpenVMS, il processo dovrà disporre dell'identificativo diritti MQM.



---

## Capitolo 8. Handler di code messaggi non recapitati di MQSeries

Una coda messaggi non recapitati DLQ (*Dead-Letter Queue* ma talvolta denominata anche *undelivered-message queue*) è una coda che contiene i messaggi che non riescono a raggiungere la loro coda di destinazione. Ogni queue Manager in una rete dovrebbe essere associato ad una DLQ.

I messaggi sono inviati alla DLQ dai Queue Manager, dai MCA (Message Channel Agent) e dalle applicazioni. Tutti i messaggi della DLQ devono avere la struttura *dead-letter header*, MQDLH. I messaggi inviati alla DLQ da un Queue Manager o da un MCA hanno sempre un MQDLH; le applicazioni che inviano messaggi alla DLQ devono sempre fornire tale struttura. Il campo *Reason* della struttura MQDLH contiene un codice di errore che identifica il motivo per cui il messaggio è stato inviato alla DLQ.

In tutti gli ambienti MQSeries, deve essere impostata una routine da eseguire ad intervalli regolari per elaborare i messaggi presenti nella DLQ. MQSeries fornisce una routine predefinita, denominata *DLQ handler* (dead-letter queue handler), che può essere richiamata utilizzando il comando **runmqdlq**.

Le istruzioni per l'elaborazione dei messaggi presenti nella DLQ vengono fornite al DLQ handler dai contenuti immessi dall'utente nella *tabella regole*. Il DLQ handler confronta i messaggi presenti nella coda messaggi non recapitati con le indicazioni della tabella regole: quando rileva una corrispondenza tra un messaggio e un'indicazione, il DLQ handler esegue l'azione associata a quella determinata indicazione.

Questo capitolo si compone delle seguenti sezioni:

- "Richiamo del DLQ handler"
- "Tabella regole DLQ handler" a pagina 108
- "Elaborazione della tabella regole" a pagina 115
- "Esempio di tabella regole per DLQ handler" a pagina 118

---

### Richiamo del DLQ handler

Per richiamare il DLQ handler occorre utilizzare il comando **runmqdlq**. È possibile denominare la coda messaggi non recapitati che si desidera elaborare e il Queue Manager che si desidera utilizzare in due modi:

- Come parametri di **runmqdlq** alla richiesta comandi. Ad esempio:

```
runmqdlq ABC1.DEAD.LETTER.QUEUE ABC1.QUEUE.MANAGER < qrule.rul
```

- Nella tabella regole. Ad esempio:

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE) INPUTQM(ABC1.QUEUE.MANAGER)
```

## DLQ handler

Gli esempi riportati si riferiscono alla coda DLQ denominata ABC1.DEAD.LETTER.QUEUE, di pertinenza del Queue Manager ABC1.QUEUE.MANAGER.

se non vengono specificati i nomi della DLQ e del Queue Manager come sopra riportato, viene utilizzato il Queue Manager predefinito per l'installazione e la coda DLQ di pertinenza.

Il comando **runmqdlq** rileva i propri input da SYS\$INPUT : associare pertanto la tabella regole con **runmqdlq** reindirizzando SYS\$INPUT dalla tabella regole.

**Attenzione:** l'esecuzione del DLQ handler senza il preventivo reindirizzo di SYS\$INPUT provoca il loop del DLQ handler.

Per poter eseguire il DLQ handler, è necessario l'autorizzazione ad accedere sia alla coda messaggi non recapitati, sia alle code cui erano destinati i messaggi. Pertanto, se si abilita il DLQ handler ad inviare i messaggi alle code con l'autorizzazione dell'ID utente nel contesto del messaggio, si dovrà avere l'autorizzazione ad assumere l'identità di altri utenti.

Per ulteriori informazioni relative al comando **runmqdlq**, consultare la sezione "runmqdlq (Run dead-letter queue handler)" a pagina 300.

## Modello di DLQ handler, amqsdq

In aggiunta al DLQ handler richiamato utilizzando il comando **runmqdlq**, MQSeries fornisce il sorgente di un modello di DLQ handler, **amqsdq**, le cui funzioni sono simili a quelle prodotte con **runmqdlq**. I sorgenti sono forniti solo come modello e devono essere personalizzati in modo da fornire un DLQ handler che soddisfi dei requisiti specifici. Ad esempio, si potrebbe voler utilizzare un DLQ handler in grado di elaborare messaggi senza l'intestazione di messaggi non recapitati. Sia il DLQ handler predefinito che il modello di DLQ handler, **amqsdq**, elaborano solo i messaggi che iniziano con l'intestazione dei messaggi non recapitati, MQDLH. I messaggi che non hanno tale intestazione vengono identificati come messaggi affetti da errore e restano nella coda messaggi non recapitati per un tempo indefinito.

Il sorgente di **amqsdq** è memorizzato nella directory:

[.DLQ], under MQS\_EXAMPLES

e la versione compilata è memorizzata nella directory:

[.BIN], under MQS\_EXAMPLES

---

## Tabella regole DLQ handler

La tabella regole DLQ handler definisce il tipo di elaborazione dei messaggi che arrivano nella coda messaggi non recapitati. Vi sono due tipi di voci da poter immettere nella tabella:

- La prima, facoltativa, contiene *dati di controllo*.
- Tutte le altre sono *regole* che il DLQ handler deve seguire. Ogni regola è costituita di uno *schema* (un gruppo di caratteristiche del messaggio) cui un messaggio deve corrispondere e di un'*azione* che deve essere eseguita, quando nella coda messaggi non recapitati viene rilevato un messaggio che corrisponde alle caratteristiche dello schema. Deve esserci almeno una regola nella tabella.

Ogni voce della tabella comprende una o più parole chiave.

## Dati di controllo

Questa sezione descrive le parole chiave che si possono inserire in una voce di dati di controllo nella tabella regole di un DLQ handler. Si noti quanto segue:

- L'impostazione predefinita dell'eventuale parola chiave è sottolineata.
- La linea verticale (|) separa le alternative, di cui una sola può essere specificata.
- Tutte le parole chiave sono facoltative.

## DLQ handler

### INPUTQ (*QueueName* | ' \_')

Consente di denominare la coda messaggi non recapitati che si desidera elaborare:

1. Se si specifica un valore INPUTQ come un parametro del comando **runmqdlq**, viene sovrascritto ogni valore INPUTQ presente nella tabella regole.
2. Se non si specifica un valore INPUTQ come parametro del comando **runmqdlq**, ma si specifica un valore nella tabella, il valore INPUTQ viene utilizzato nella tabella regole.
3. Se non si specifica una coda DLQ o si specifica INPUTQ(' ') nella tabella regole, viene utilizzato il nome della coda DLQ di pertinenza del Queue Manager il cui nome è immesso come parametro del comando **runmqdlq**.
4. Se si specifica un valore INPUTQ come parametro del comando **runmqdlq** o come valore nella tabella regole, viene utilizzata la coda DLQ di pertinenza del Queue Manager denominato nella parola chiave INPUTQM.

### INPUTQM (*QueueManagerName* | ' \_')

Consente di denominare il Queue Manager che gestisce la coda DLQ denominata con la parola chiave INPUTQ:

1. Se si specifica un valore INPUTQM come parametro del comando **runmqdlq**, viene sovrascritto ogni valore INPUTQM presente nella tabella regole.
2. Se non si specifica un valore INPUTQM come parametro del comando **runmqdlq**, viene utilizzato il valore INPUTQM.
3. Se non si specifica nessun Queue Manager o si specifica INPUTQM(' ') nella tabella regole, viene utilizzato il Queue Manager predefinito.

### RETRYINT (*Interval* | 60)

Indica, in secondi, l'intervallo di tempo che separa i tentativi di elaborazione da parte del DLQ handler di quei messaggi che non è riuscito ad elaborare al primo tentativo e per i quali è stato richiesto un nuovo tentativo. Per impostazione predefinita, l'intervallo di tempo è di 60 secondi.

### WAIT (YES | NO | *nnn*)

Indica se il DLQ handler deve attendere l'arrivo di ulteriori messaggi sulla coda DLQ quando rileva l'assenza di messaggi da elaborare.

**YES** Determina un'attesa a tempo indeterminato.

**NO** Determina l'arresto del DLQ handler quando rileva che non vi sono altri messaggi da elaborare sulla coda DLQ.

*nnn* Determina un'attesa del DLQ handler di *nnn* secondi dall'elaborazione dell'ultimo messaggio prima dell'arresto.

Si consiglia di specificare WAIT (YES) per le code DLQ piene di messaggi e WAIT (NO) o WAIT (*nnn*) per la code DLQ con un basso livello di attività. Se il DLQ handler si arresta, è necessario richiamarlo con la funzione trigger.

In alternativa all'immissione di dati di controllo nella tabella regole, è possibile inserire il nome della coda DLQ e del relativo Queue Manager come parametro di immissione del comando **runmqdlq**. Se vengono specificati entrambi, il parametro di immissione del comando **runmqdlq** ha la precedenza sul valore specificato nella tabella regole.

**Nota:** se nella tabella regole è inclusa una voce dati di controllo, *deve* essere la prima voce della tabella.



## Regole (schemi e azioni)

La Figura 8 mostra un esempio di regole da una tabella regole DLQ handler.

```
PERSIST(MQPER_PERSISTENT) REASON(MQRC_PUT_INHIBITED) +
ACTION(RETRY) RETRY(3)
```

Figura 8. Esempio di regola da una tabella regole DLQ handler. Questa regola ordina al DLQ handler di effettuare tre tentativi per inviare alla propria coda di destinazione ogni messaggio inviato alla coda DLQ in seguito all'inibizione di MQPUT e MQPUT1.

Tutte le parole chiave utilizzabili in una regola sono descritte nella rimanente parte di questa sezione. Si noti quanto segue:

- L'impostazione predefinita dell'eventuale parola chiave è sottolineata. Per molte parole chiave, il valore predefinito è \* (asterisco), che comprende ogni valore.
- La linea verticale (|) separa alternative, di cui una sola può essere specificata.
- Tutte le parole chiave sono facoltative, tranne ACTION.

Questa sezione descrive prima le parole chiave dello schema di confronto (quelle con le quali vengono confrontati i messaggi presenti nella coda DLQ) e poi quelle relative alle azioni (quelle che determinano il tipo di elaborazione dei messaggi da parte del DLQ handler).

### Le parole chiave dello schema di confronto

Di seguito vengono descritte le parole chiave dello schema di confronto, utilizzate per specificare i valori con i quali confrontare i messaggi nella coda DLQ. Tutte le parole chiave sono facoltative.

#### APPLIDAT (*ApplIdentityData* | \*)

Indica il valore *ApplIdentityData* specificato nel descrittore di messaggio MQMD del messaggio presente nella coda DLQ.

#### APPLNAME (*PutAppName* | \*)

Indica il nome dell'applicazione che inoltra la chiamata MQPUT o MQPUT1, come specificato nel campo *PutAppName* del descrittore di messaggio MQMD, del messaggio nella coda DLQ.

#### APPLTYPE (*PutApplType* | \*)

Indica il valore *PutApplType* specificato nel descrittore di messaggio del messaggio nella coda DLQ.

#### DESTQ (*QueueName* | \*)

Indica il nome della coda destinataria del messaggio.

#### DESTQM (*QueueManagerName* | \*)

Indica il nome del Queue Manager che gestisce la coda destinataria del messaggio.

#### FEEDBACK (*Feedback* | \*)

Quando il valore *MsgType* è MQFB\_REPORT, *Feedback* descrive la natura della notifica.

Possono essere utilizzati nomi simbolici. Ad esempio, per identificare i messaggi nella coda DLQ che richiedono la conferma del proprio arrivo alla coda di destinazione, è possibile utilizzare il nome simbolico MQFB\_COA.

#### FORMAT (*Format* | \*)

Indica il nome utilizzato dal mittente per descrivere il formato dei dati del messaggio inviato.

## DLQ handler

### **MSGTYPE** (*MsgType* | \*)

Indica il tipo di messaggio presente nella coda DLQ.

È possibile utilizzare nomi simbolici. Ad esempio, per identificare i messaggi presenti nella coda DLQ che richiedono risposte, è possibile utilizzare il nome simbolico MQMT\_REQUEST.

### **PERSIST** (*Persistence* | \*)

Indica il valore di permanenza del messaggio. La permanenza determina la sopravvivenza o meno del messaggio al riavvio del Queue Manager.

È possibile utilizzare nomi simbolici. Ad esempio, per identificare i messaggi nella coda DLQ permanenti, è possibile utilizzare il nome MQPER\_PERSISTENT.

### **REASON** (*ReasonCode* | \*)

Indica il codice di errore che spiega il motivo dell'invio del messaggio alla coda DLQ.

Possono essere utilizzati nomi simbolici. Ad esempio, per identificare i messaggi che sono stati inviati alla coda DLQ poiché la loro coda di destinazione era piena, è possibile utilizzare il nome simbolico MQRC\_Q\_FULL.

### **REPLYQ** (*QueueName* | \*)

Indica il nome della coda di risposta specificata nel descrittore di messaggio, MQMD, relativo al messaggio presente nella coda DLQ.

### **REPLYQM** (*QueueManagerName* | \*)

Indica il nome del Queue Manager che gestisce la coda di risposta specificata nel descrittore di messaggio, MQMD, del messaggio presente nella coda DLQ.

### **USERID** (*UserIdentifier* | \*)

Indica l'ID utente specificato nel descrittore di messaggio; MQMD, del messaggio presente nella coda DLQ.

## **Le parole chiave di azione**

Di seguito vengono descritte le parole chiave che indicano il tipo di elaborazione dei messaggi, che rientrano nello schema di confronto.

### **ACTION (DISCARD | IGNORE | RETRY | FWD)**

Indica l'azione da eseguire per quei messaggi nella coda DLQ che rientrano nello schema definito in questa regola.

#### **DISCARD**

Determina l'eliminazione del messaggio dalla coda DLQ.

#### **IGNORE**

Determina la permanenza del messaggio nella coda DLQ.

#### **RETRY**

Determina un nuovo tentativo del DLQ handler di inviare il messaggio alla coda di destinazione.

**FWD** Fa sì che il DLQ handler inoltri il messaggio alla coda denominata con la parola chiave FWDQ.

La parola chiave ACTION deve essere specificata. Il numero tentativi volti al compimento di un'azione è determinato dalla parola chiave RETRY. L'intervallo di tempo tra i vari tentativi è indicato dalla parola chiave RETRYINT dei dati di controllo.

**FWDQ** (*QueueName* | **&DESTQ** | **&REPLYQ**)

Indica il nome della coda a cui inoltrare il messaggio quando è impostato ACTION (FWD).

*QueueName*

Indica il nome di una coda messaggi. FWDQ(' ') non è valido.

**&DESTQ**

Determina l'assunzione del nome della coda dal campo *DestQName* nella struttura MQDLH.

**&REPLYQ**

Determina l'assunzione del nome dal campo *ReplyToQ* nel descrittore di messaggio MQMD.

Per evitare messaggi di errore, che si verificherebbero qualora una regola che specifica FWDQ (&REPLYQ) riscontri un messaggio con il campo *ReplyToQ* vuoto, è consigliabile specificare REPLYQ (?\*) nello schema messaggi.

**FWDQM** (*QueueManagerName* | **&DESTQM** | **&REPLYQM** | ' ')

Identifica il Queue Manager che gestisce la coda cui deve essere inoltrato un messaggio.

*QueueManagerName*

Indica il nome del Queue Manager che gestisce la coda cui inoltrare un messaggio quando è impostato ACTION (FWD).

**&DESTQM**

Determina l'assunzione del nome del Queue Manager dal campo *DestQMgrName* nella struttura MQDLH.

**&REPLYQM**

Determina l'assunzione del nome dal campo *ReplyToQMgr* nel descrittore di messaggi MQMD.

' ' FWDQM(' '), valore predefinito, identifica il Queue Manager locale.

**HEADER** (**YES** | **NO**)

Specifica se la struttura MQDLH debba rimanere su un messaggio per cui è stata specificata la parola chiave ACTION (FWD). Per impostazione predefinita, la struttura MQDLH rimane sul messaggio. La parola chiave HEADER non è valida per altre azioni diverse da FWD.

**PUTAUT** (**DEF** | **CTX**)

Definisce l'autorizzazione in base alla quale il DLQ handler deve eseguire l'invio dei messaggi:

**DEF** Determina l'invio dei messaggi con l'autorizzazione propria del DLQ handler.

**CTX** Determina l'invio dei messaggi con l'autorizzazione dell'ID utente nel messaggio context. Se si specifica PUTAUT (CTX), è necessaria l'autorizzazione ad assumere l'identità di altri utenti.

**RETRY** (*RetryCount* | **1**)

Indica il numero di tentativi che possono essere realizzati per il compimento di un'azione, in uno spazio tra 1-999,999,999 (l'intervallo i tempo è specificato con la parola chiave RETRYINT dei dati di controllo).

**Nota:** il conteggio dei tentativi fatti dal DLQ handler per completare una determinata azione è relativo all'istanza in corso; il conteggio non viene

ripreso dopo un riavvio. Quando il DLQ handler viene riavviato, il conteggio dei tentativi eseguiti è azzerato.

### Convenzioni relative alla tabella regole

La tabella regole deve rispettare le seguenti convenzioni relative alla sintassi, alla struttura e ai contenuti:

- Una tabella regole deve contenere almeno una regola.
- Le parole chiave possono ricorrere in qualsiasi ordine.
- Una parola chiave può essere inserita solo una volta in ciascuna regola.
- Le parole chiave non sono sensibili al minuscolo/maiuscolo.
- Una parola chiave e i relativi valori dei parametri devono essere separate dalle altre parole chiave con almeno uno spazio o una virgola.
- All'inizio e alla fine della regola, tra le parole chiave, la punteggiatura e i valori può essere inserito un qualsiasi numero di spazi vuoti.
- Ogni regola deve iniziare su una sua nuova linea.
- Per ragioni di flessibilità d'uso, la dimensione di una linea non deve superare i 72 caratteri.
- Utilizzare il segno più (+) come ultimo carattere di una linea per indicare che la regola prosegue con il primo carattere digitato nella linea successiva. Utilizzare il segno meno (-) come ultimo carattere di una linea per indicare che la regola prosegue dall'inizio della linea successiva. I caratteri di continuazione possono essere inseriti all'interno di parole chiave e parametri.
- Linee di commento, che iniziano con un asterisco (\*), possono trovarsi in qualsiasi punto della tabella regole.
- Le linee vuote sono ignorate.
- Ogni voce della tabella regole del DLQ handler comprende una o più parole chiave e i relativi parametri associati. I parametri devono rispettare le seguenti regole di sintassi:
  - Ogni parametro deve includere almeno un carattere valido. I segni di delimitazione in cui vengono ricompresi i valori tra parentesi non sono considerati caratteri validi. Ad esempio, di seguito sono riportati dei caratteri validi:  
**FORMAT ('ABC')** 3 caratteri validi  
**FORMAT(ABC)** 3 caratteri validi  
**FORMAT('A')** 1 carattere valido  
**FORMAT(A)** 1 carattere valido  
**FORMAT(' ')** 1 carattere valido

I seguenti parametri invece non sono validi, poiché non contengono caratteri validi:

**FORMAT('')**  
**FORMAT( )**  
**FORMAT()**  
**FORMAT**

- I caratteri globali sono supportati: è possibile utilizzare il punto interrogativo (?) al posto di ogni singolo carattere, tranne gli spazi vuoti; è possibile utilizzare l'asterisco (\*) al posto dello zero o di altri caratteri simili. L'asterisco (\*) e il punto interrogativo (?) sono *sempre* considerati caratteri globali nell'impostazione dei parametri.
- I caratteri globali non possono essere utilizzati nei parametri delle seguenti parole chiave: ACTION, HEADER, RETRY, FWDQ, FWDQM e PUTAUT.

- Gli spazi vuoti nell'impostazione dei parametri e dei corrispondenti campi nei messaggi presenti nella coda DLQ, non sono rilevanti nel confronto con i caratteri globali. Tuttavia, gli spazi vuoti inclusi tra le virgolette assumono rilevanza nel confronto con i caratteri globali.
- I parametri numerici non possono includere il punto interrogativo (?). L'asterisco (\*) può invece essere utilizzato al posto di un intero parametro numerico, ma non può essere incluso come parte di un parametro numerico. Ad esempio, di seguito sono riportati parametri numerici validi:
  - MSGTYPE(2)** Sono considerati solo i messaggi di risposta
  - MSGTYPE(\*)** Sono considerati tutti i tipi di messaggi.
  - MSGTYPE('\*')** Sono considerati tutti i tipi di messaggi.

Tuttavia, **MSGTYPE('2\*')** non è valido, perché include l'asterisco (\*) come una parte di un parametro numerico.

- I parametri numerici devono essere compresi tra 0–999,999,999. Se il valore è compreso in questa gamma viene accettato, anche se non è valido per la parola chiave a cui si riferisce. Possono essere utilizzati nomi simbolici per parametri numerici.
- Se l'impostazione di una stringa è più breve del campo nel MQDLH o MQDM cui la parola chiave si riferisce, al valore viene aggiunta una serie di spazi fino al termine del campo. Se invece l'impostazione, compresa di un asterisco, è più lunga del campo che deve ospitarla, viene diagnosticato un errore. Ad esempio, di seguito vengono riportate impostazioni valide per un campo di 8 caratteri:
 

<b>'ABCDEFGH'</b>	8 caratteri
<b>'A*C*E*G*I'</b>	5 caratteri escluso gli asterischi
<b>'*A*C*E*G*I*K*M*O*'</b>	8 caratteri esclusi gli asterischi
- Le stringhe che contengono spazi vuoti, caratteri minuscoli o caratteri speciali diversi dal punto (.), dallo slash inclinato in avanti (/), dai punti sospensivi (.) o dal segno percentuale (%) devono essere inclusi tra virgolette. I caratteri minuscoli non racchiusi tra virgolette vengono considerati maiuscoli. Se nella stringa sono previste le virgolette, queste dovranno essere racchiuse tra le virgolette doppie, per segnalare l'inizio e la fine del testo racchiuso nelle virgolette. Il carattere delle virgolette doppie viene contato come singolo carattere.

---

## Elaborazione della tabella regole

Il DLQ handler ricerca una tabella regole il cui schema corrisponda alle caratteristiche del messaggio presente nella coda DLQ. La ricerca comincia dalla prima regola della tabella e prosegue in sequenza. Quando viene rilevata una regola con uno schema corrispondente alle caratteristiche del messaggio, viene tentata l'azione prevista da quella regola. Il DLQ handler incrementa il conteggio dei tentativi di 1, ogni volta che prova l'esecuzione di quella regola. Se il primo tentativo fallisce, se ne eseguiranno degli altri, fino a raggiungere il numero massimo stabilito con la parola chiave RETRY. Se tutti tentativi hanno esito negativo, il DLQ ricerca nella tabella la successiva regola corrispondente alle caratteristiche del messaggio.

Questo processo viene ripetuto in sequenza fino a quando un'azione avrà esito positivo. Quando tutte le regole corrispondenti alle caratteristiche del messaggio sono state eseguite, per il numero di volte specificato con la parola chiave RETRY, e dopo che tutte hanno avuto esito negativo viene utilizzato ACTION (IGNORE). Tale parola chiave viene utilizzato anche nel caso non venga rilevata alcuna regola corrispondente alle caratteristiche del messaggio.

## DLQ handler

### Note:

1. Gli schemi di confronto delle regole vengono ricercati solo per i messaggi che iniziano con un MQDLH presenti sulla coda DLQ. I messaggi che non hanno tale intestazione vengono identificati come messaggi affetti da errore e restano nella coda messaggi non recapitati per un tempo indefinito.
2. Per tutte le parole chiave dello schema possono essere utilizzate quelle predefinite, così come una regola può consistere di una sola azione. Si noti, tuttavia, che le regole composte di una sola azione verranno applicate a tutti i messaggi con MQDDLH che non sono stati già elaborati in funzione di altre regole della tabella.
3. All'avvio del DLQ handler viene rilevata la tabella regole e vengono segnalati gli errori. È possibile effettuare modifiche in qualunque momento, ma tali modifiche non avranno effetto fino al successivo riavvio del DLQ handler.
4. Il DLQ handler non modifica il contenuto dei messaggi, del MQDLH o del descrittore dei messaggi. Il DLQ handler invia sempre i messaggi alle altre code con l'opzione di messaggio MQPMO\_PASS\_ALL\_CONTEXT.
5. Il DLQ handler apre la coda messaggi non recapitati con l'opzione MQOO\_INPUT\_AS\_Q\_DEF.
6. È possibile eseguire più istanze del DLQ handler contemporaneamente verso la stessa coda, utilizzando la stessa tabella regole. Tuttavia, accade più spesso che venga elaborata un'unica istanza per volta.

## Metodi per l'elaborazione di tutti i messaggi presenti nella coda DLQ

Il DLQ handler conserva un record di tutti i messaggi rilevati nella coda DLQ e non ancora rimossi. Se si utilizza il DLQ handler come filtro di estrazione dalla coda DLQ di un sottogruppo di messaggi, il DLQ handler conserverà ancora il record di quei messaggi della coda DLQ non elaborati. Inoltre, il DLQ handler non può garantire che i nuovi messaggi che arrivano nella coda DLQ saranno rilevati, anche se è stata definita l'impostazione FIFO (First-In-First-Out). Perciò, a meno che la coda sia vuota, viene realizzata un'analisi periodica della coda DLQ per controllare tutti i messaggi. Per queste ragioni, è necessario fare in modo che la coda DLQ contenga il minor numero di messaggi possibile; se i messaggi che non possono essere eliminati o inoltrati ad altre code (per qualsiasi ragione) si accumulano, il carico di lavoro del DLQ handler aumenta e la stessa coda DLQ corre il rischio di riempirsi.

Esistono operazioni apposite per consentire al DLQ handler di svuotare la coda messaggi non recapitati. Ad esempio, si consiglia di evitare l'utilizzo del parametro ACTION (IGNORE), poiché questo lascia accumulare i messaggi nella coda DLQ. Questo parametro è utilizzato per i messaggi che non hanno un'esplicita corrispondenza nella tabella regole. Al posto di questo parametro, per i messaggi che si desidera ignorare, utilizzare un'azione che sposti i messaggi verso un'altra coda. Ad esempio:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

Analogamente, l'ultima regola della tabella dovrebbe essere impostata in modo da corrispondere a tutti i messaggi; così i messaggi che non hanno trovato corrispondenza con le regole precedenti verranno elaborate in base all'ultima

regola. Ad esempio, l'ultima regola della tabella potrebbe assomigliare a quella che segue:

ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
--

Questa azione fa sì che tutti i messaggi, che non sono stati elaborati in base alle precedenti regole, vengano inoltrati alla coda `REALLY.DEAD.QUEUE`, dove poi saranno elaborate manualmente. se non si imposta una regola simile a questa, i messaggi probabilmente rimarranno nella coda `DLQ` indefinitamente.



## Esempio di tabella regole per DLQ handler

Di seguito viene riportato un esempio di tabella regole che contiene una singola voce dati di controllo e diverse regole:

```
*****
*           An example rules table for the runmqdlq command           *
*****
* Control data entry
* -----
* If no queue manager name is supplied as an explicit parameter to
* runmqdlq, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to runmqdlq,
* use the DLQ defined for the local queue manager.
*
inputqm(' ') inputq(' ')

* Rules
* ----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.

* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation are always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
  ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never do things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
  action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.

REPLYQM(CCCC.*) +
  ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it should be able
* to cope with the message being lost, so we can afford to
* discard the message.
```



```
PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)
```

```
* For performance and efficiency reasons, we like to keep  
* the number of messages on the DLQ small.  
* If we receive a message that has not been processed by  
* an earlier rule in the table, we assume that it  
* requires manual intervention to resolve the problem.  
* Some problems are best solved at the node where the  
* problem was detected, and others are best solved where  
* the message originated. We don't have the message origin,  
* but we can use the REPLYQM to identify a node that has  
* some interest in this message.  
* Attempt to put the message onto a manual intervention  
* queue at the appropriate node. If this fails,  
* put the message on the manual intervention queue at  
* this node.
```

```
REPLYQM('?*') +  
  ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)
```

```
ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

## DLQ handler

---

## Capitolo 9. Eventi di strumentazione

È possibile utilizzare gli eventi di strumentazione MQSeries per controllare le operazioni dei Queue Manager. Questo capitolo fornisce un breve introduzione agli eventi di strumentazione. Per una descrizione completa, consultare il manuale *MQSeries Programmable System Management*.

---

### Cosa sono gli eventi di strumentazione?

Gli eventi di strumentazione generano messaggi particolari, denominati *messaggi di eventi*, ogni volta che i Queue Manager rilevano il verificarsi di determinate condizioni predefinite. Ad esempio, le seguenti condizioni danno luogo ad un evento *Coda Piena*:

- Gli eventi Coda Piena possono verificarsi solo per una determinata coda.
- Un'applicazione inoltra una chiamata MQPUT per inviare un messaggio ad una coda, ma la chiamata ha esito negativo poiché la coda è piena.

Altre condizioni che possono dar luogo ad un evento di strumentazione sono:

- Viene ricevuto il numero limite di messaggi predeterminato per una coda.
- Una coda non viene servita nell'intervallo di tempo specificato.
- Viene avviata o arrestata un'istanza del canale.
- In un sistema MQSeries per Compaq OpenVMS, un'applicazione tenta di aprire una coda specificando un>ID utente non autorizzato.

Con l'eccezione degli eventi canale, tutti gli eventi di strumentazione devono essere abilitati prima che si verifichino.

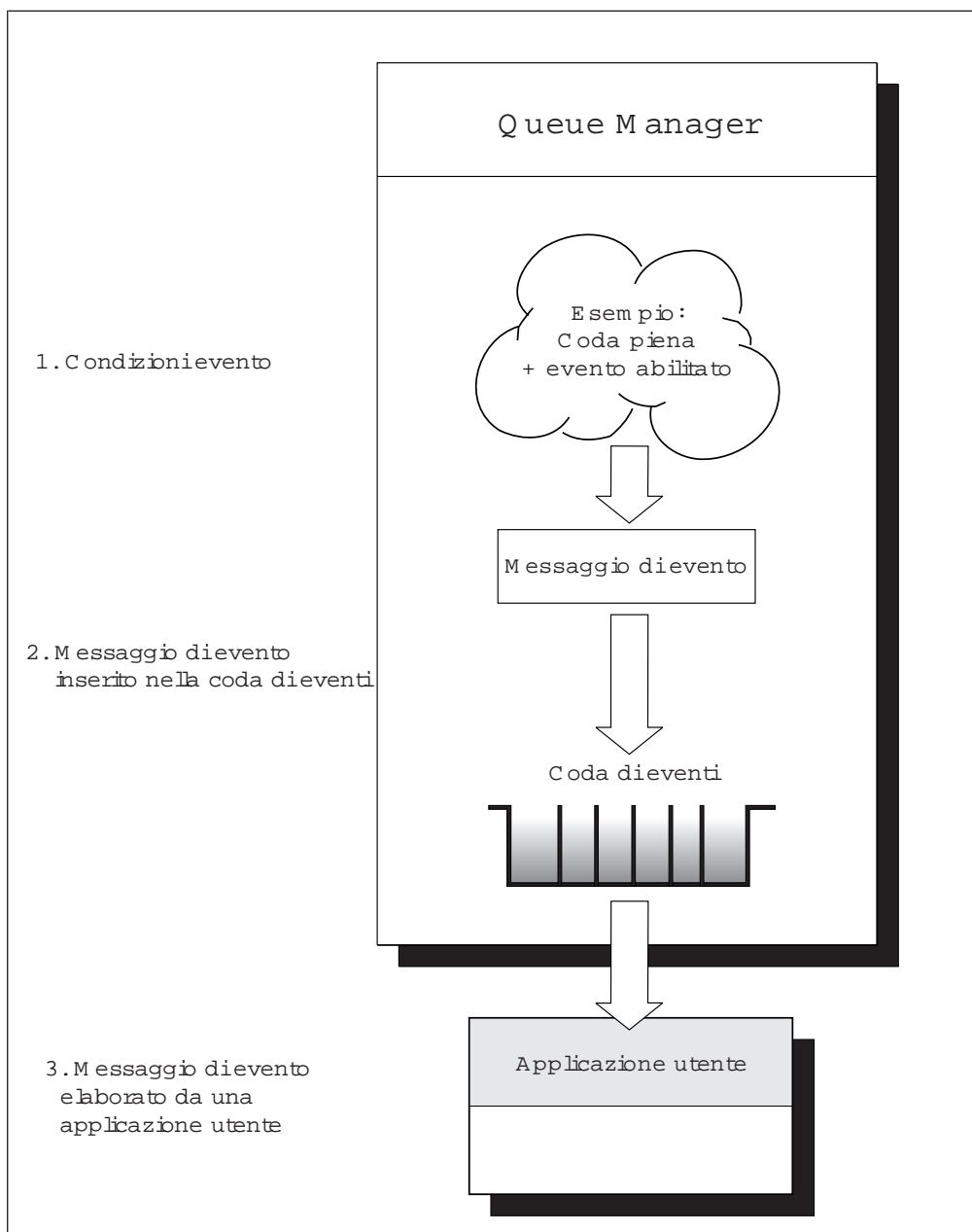


Figura 9. Analisi degli eventi di strumentazione. Quando un Queue Manager rileva il verificarsi delle condizioni per un evento, invia un messaggio di evento alla coda di evento appropriata.

Alla coda di evento viene inviato il messaggio di evento, che contiene le informazioni relative alle condizioni che danno luogo all'evento. Un'applicazione può così richiamare il messaggio di evento da questa coda per analizzarlo.

## Perché utilizzare gli eventi?

Se si specificano le code di evento come code remote, è possibile inserire tutte le code di evento in un singolo Queue Manager (per i nodi che supportano gli eventi di strumentazione). Sarà così possibile utilizzare gli eventi prodotti per controllare una rete di Queue Manager da un singolo nodo. Questo è quanto viene illustrato nella Figura 10 a pagina 123.

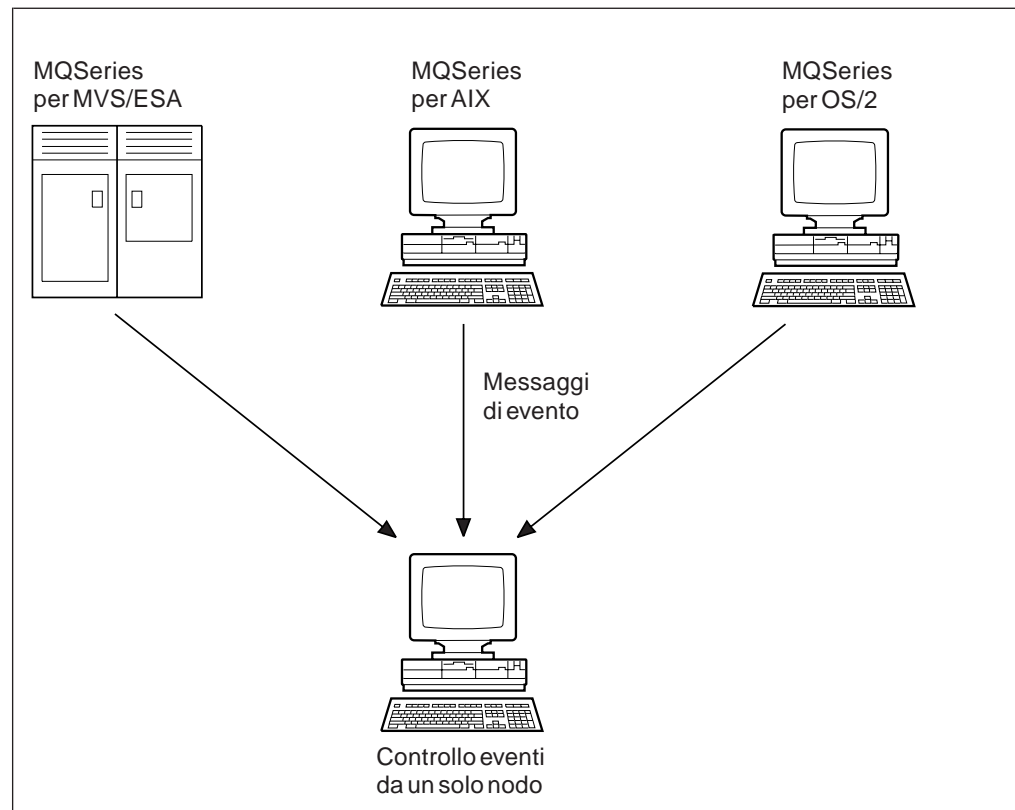


Figura 10. Controllo dei Queue Manager attraverso diverse piattaforme, da un singolo nodo.

## Tipi di eventi

Gli eventi MQSeries possono essere suddivisi nelle seguenti categorie:

### Eventi Queue Manager

Questi eventi si riferiscono alle definizioni di risorse nei Queue Manager. Ad esempio, un'applicazione tenta di inviare un messaggio ad una coda che non esiste.

### Eventi prestazioni

Questi eventi costituiscono notifiche del fatto che una risorsa ha raggiunto una determinata condizione. Ad esempio, si è raggiunto il limite di capacità di una coda oppure, durante un'estrazione, la coda non è stata servita entro il limite di tempo prestabilito.

### Eventi canale

Questi eventi sono riportati dai canali come il risultato delle condizioni rilevate durante le loro operazioni. Ad esempio, l'arresto di un'istanza del canale.

### Eventi trigger

Quando si tratta la funzione trigger in questo ed in altri manuali MQSeries, talvolta si fa riferimento all'*evento trigger*, che si verifica quando un Queue Manager riscontra le condizioni per un evento trigger. Ad esempio, una coda può essere configurata in modo da produrre un evento trigger tutte le volte che arriva un messaggio. Le condizioni per un evento trigger e quelle per un evento di strumentazione sono leggermente diverse.

Un evento trigger fa sì che un messaggio trigger sia inviato ad una coda di iniziazione e, facoltativamente, venga avviato un programma applicativo.

## Notificazione di evento attraverso code di evento

Quando si verifica un evento, il Queue Manager inserisce un messaggio di evento nella coda di evento appropriata, se definita. Il messaggio di evento contiene informazioni relative all'evento che è possibile richiamare compilando un apposito programma applicativo MQI che:

- Estrae il messaggio dalla coda.
- Elabora il messaggio per estrarre i dati dell'evento. Per una descrizione dei formati dei messaggi di evento, consultare il manuale *MQSeries Programmable System Management*.

Ogni categoria di eventi ha la propria coda di evento. Tutti gli eventi di una categoria risultano in messaggio di evento che viene inviato alla stessa coda.

**Questa coda di evento...                      Contiene messaggi da...**

**SYSTEM.ADMIN.QMGR.EVENT**

Eventi Queue Manager

**SYSTEM.ADMIN.PERFM.EVENT**

Eventi prestazioni

**SYSTEM.ADMIN.CHANNEL.EVENT**

Eventi canali

È possibile definire code di evento sia come code locali che remote. Definendo tutte le code di evento come code remote sullo stesso Queue Manager, è possibile centralizzare le attività di controllo.

### Utilizzo delle code di evento con la funzione trigger

È possibile impostare le code di evento con la funzione trigger in modo tale che quando viene generato un evento, il messaggio di evento che viene inserito nella coda evento, avvia un'applicazione di controllo (compilata dall'utente). Questa applicazione può elaborare i messaggi di evento ed avviare le dovute azioni. Ad esempio, alcuni eventi possono prevedere una notifica all'operatore, altri eventi possono avviare un'applicazione che esegue determinate attività di gestione in modo automatico.

## Abilitazione e disabilitazione degli eventi

Per abilitare e disabilitare gli eventi occorre specificare i valori appropriati per il Queue Manager o gli attributi di coda o entrambi, in base al tipo di evento. A questo scopo utilizzare i seguenti comandi:

- Comandi MQSC. Per ulteriori informazioni, consultare il manuale *MQSeries - Guida di riferimento per i comandi*.
- Comandi PCF per Queue Manager su sistemi UNIX, sistemi OpenVMS e sistemi OS/". Per ulteriori informazioni, consultare il manuale *MQSeries Programmable System Management*.
- Comandi MQAI. Per ulteriori informazioni, consultare il manuale *MQSeries Administration Interface Programming Guide and Reference*.

Abilitazione di un evento in base alla sua categoria:

- Gli eventi Queue Manager si abilitano impostando gli attributi del Queue Manager.
- Gli eventi prestazioni devono essere integralmente abilitati sul Queue Manager o non potranno verificarsi eventi prestazioni. Per abilitare uno specifico evento prestazione, quindi, occorrerà impostare l'attributo di coda opportuno. Inoltre bisognerà anche specificare le condizioni che danno luogo all'evento, come ad esempio, un limite di capacità della coda.
- Gli eventi canale si verificano in modo automatico; non occorre abilitarli. Se si desidera escludere il controllo degli eventi canale, è possibile inibire la coda evento canale.

## Messaggi di evento

I messaggi di evento contengono informazioni relative all'origine di un evento, incluso il tipo di evento, il nome dell'applicazione che ha determinato l'evento e, nel caso degli eventi prestazioni, uno breve rapporto statistico della coda.

Il formato dei messaggi di evento è simile a quello dei messaggi di risposta PCF. I dati dei messaggi possono essere richiamati grazie ad un programma di gestione compilato dall'utente, utilizzando le strutture dei dati descritte nel manuale *MQSeries Programmable System Management*.





---

## Capitolo 10. Supporto transazionale

Il manuale *MQSeries Application Programming Guide* contiene un'esauriente trattazione relativa all'argomento di questo capitolo. Di seguito è riportata solo una concisa introduzione.

Un programma applicativo può riunire un gruppo di aggiornamenti in un'*unità di elaborazione*. Tali aggiornamenti sono solitamente in relazione logica e devono essere privi di errori per preservare l'integrità dei dati. Se anche un solo aggiornamento si arresta l'integrità dei dati verrà compromessa.

Un'unità di elaborazione esegue il **commit** quando ha esito positivo. Da questo momento tutti gli aggiornamenti realizzati in quella unità di elaborazione diventano permanenti o irreversibili. Se invece l'unità di elaborazione dovesse avere esito negativo allora di tutti gli aggiornamenti verrà invece eseguito il *back out*. *Syncpoint coordination* è il processo attraverso il quale si esegue il commit o il back out delle intere unità di lavoro.

Un'unità di elaborazione *locale* si caratterizza per il fatto che le uniche risorse aggiornate sono quelle del Queue Manager MQSeries. In queste unità, syncpoint coordination è prodotto autonomamente dal Queue Manager utilizzando un processo commit a fase singola.

Un'unità di elaborazione *globale* si caratterizza per il fatto che vengono aggiornate anche le risorse relative ad altri gestori di risorse, quali un database XA compatibile. In queste unità deve essere utilizzato un processo commit a doppia fase e l'unità di elaborazione deve essere coordinata da Queue Manager autonomamente.

In breve, le risorse Queue Manager possono essere aggiornate come parte di un'unità di elaborazione locale o globale:

### Unità di lavoro locale:

Utilizzare le unità di lavoro locale quando le uniche risorse da aggiornare sono quelle del Queue Manager MQSeries. Il commit degli aggiornamenti si esegue utilizzando il termine MQCMIT, mentre per il back out si utilizza MQBACK.

### Unità di elaborazione globale

Utilizzare le unità di elaborazione globale quando si ha la necessità di includere aggiornamenti ai gestori di database XA compatibili. In questi casi la coordinazione può essere interna o esterna al Queue Manager.

### Coordinazione Queue Manager

Le unità di elaborazione globale vengono avviate utilizzando il termine MQBEGIN e se ne esegue il commit con il termine MQCMIT o il back out con il termine MQBACK. Viene utilizzato un processo commit a doppia fase grazie al quale viene come prima cosa richiesto a tutti gestori di risorse XA compatibili come Oracle® di prepararsi al commit. La richiesta relativa all'esecuzione del commit avverrà solo se la preparazione è completa. Se un gestore di risorsa segnala che non è in grado di prepararsi al commit, verrà richiesto se si desidera eseguire il back out per quel gestore.

## Supporto transazionale

### Coordinazione esterna

In questi casi, la coordinazione è realizzata da un gestore di transazione Xa compatibile come IBM CICS®, Transarc Encina o BEA Tuxedo. Il gestore di transazione gestisce l'avvio e il commit delle unità di elaborazione. I termini MQBEGIN, MQCMIT e MQBACK non sono disponibili.

Questo capitolo spiega come abilitare il supporto per le unità di elaborazione globale (il supporto per le unità di elaborazione locale non necessita di specifica abilitazione).

Esso contiene le seguenti sezioni:

- "Coordinazione dei database"
- "Configurazione Oracle" a pagina 131
- "Attività di gestione" a pagina 136

---

## Coordinazione dei database

Quando Queue Manager coordina autonomamente unità di elaborazione globali è possibile integrare aggiornamenti di database nelle unità di elaborazione MQ. Così, un'applicazione mista MQI e SQL può essere compilata e possono essere utilizzati i termini MQCMIT e MQBACK per eseguire il commit o il roll back delle modifiche sia alle code che ai database.

Per questo processo il Queue Manager utilizza un protocollo commit a due fasi. Prima di eseguire il commit per un'unità di elaborazione, il Queue Manager richiede ad ogni gestore di database in connessione se ha eseguito la preparazione al commit dei propri aggiornamenti. Solo nel caso in cui tutti gli elementi in connessione, compreso lo stesso Queue Manager, risultino preparati, verrà eseguito il commit di tutti gli aggiornamenti di code e database. Invece del commit verrà eseguito il back out dell'unità di elaborazione che non ha preparato i propri aggiornamenti.

È previsto un supporto per il completo ripristino se il Queue Manager perde la connessione con qualsiasi dei gestori di database durante il protocollo commit. Se un gestore di database diviene indisponibile durante il processo, così che mentre gli viene richiesto di prepararsi ha già ricevuto una decisione di commit o back out, il Queue Manager ricorda questo risultato fino a quando non viene recapitato con esito positivo. Analogamente, se il Queue Manager si chiude con operazioni commit incomplete ancora in corso, ciò viene ricordato fino al riavvio del Queue Manager.

Il termine MQI, MQBEGIN, deve essere utilizzato per denotare gli aggiornamenti di database delle unità di elaborazione da includere. *MQSeries Application Programming Guide* indica programmi di esempio che elaborano aggiornamenti MQSeries e di database nella stessa unità di elaborazione.

Il Queue Manager comunica con i gestori risorse utilizzando l'interfaccia XA come descritto in *X/Open Distributed Transaction Processing: The XA Specification* (ISBN 1 872630 24 3). Da ciò si deduce che il Queue Manager può comunicare con gestori di database compatibili con questo standard. Tali gestori di database sono definiti *XA compatibili*.

La Tabella 5 a pagina 129 identifica i gestori di database XA compatibili supportati dai prodotti MQSeries Versione 5.1.

Tabella 5. Database relazionali XA compatibili

Prodotto MQSeries	DB2®	Oracle	Sybase®
MQSeries per AIX	Sì	Sì	Sì
MQSeries per HP-UX	Sì	Sì	No
MQSeries per Sun Solaris	Sì	Sì	Sì
MQSeries per Compaq OpenVMS Alpha	No	Sì	No
MQSeries per Windows NT e Windows 2000	Sì	No	Sì

## Limitazioni

Le seguenti limitazioni si applicano al supporto di coordinazione di database:

- La funzione di coordinare gli aggiornamenti di database nelle unità di elaborazione MQSeries **non** è supportata in un'applicazione MQI client.
- Gli aggiornamenti MQI e gli aggiornamenti di database devono essere eseguiti sulla stessa macchina server Queue Manager.
- Il server database può risiedere su una diversa macchina rispetto al server Queue Manager. In questo caso, è necessario accedere al database attraverso una funzione client XA compatibile fornita dallo stesso gestore di database.
- Sebbene il Queue Manager sia di per sé XA compatibile, non è possibile configurare un altro Queue Manager come componente di unità di elaborazione globali. Questo perché può essere realizzata una sola connessione per volta.

## Connessioni al database

Un'applicazione che stabilisce una connessione standard al Queue Manager verrà associata con un sottoprocesso in un diverso agente Queue Manager locale. Quando l'applicazione emette l'ordine MQBEGIN entrambi dovranno stabilire la connessione ai database da includere nell'unità di elaborazione. Le connessioni al database restano attive durante tutto il tempo in cui è attiva la connessione tra l'applicazione e il Queue Manager. Tale considerazione è da tenere presente se il database supporta un numero limite di utenti o di connessioni.

Un sistema per ridurre il numero delle connessioni è utilizzare con la chiamata MQCONNX per richiedere l'esecuzione di un fastpath binding. In questo modo l'applicazione e l'agente locale Queue manager si fondono in un solo processo e possono di conseguenza utilizzare una singola connessione al database. Prima di effettuare questa operazione, consultare il manuale *MQSeries Application Programming Guide* per un elenco delle limitazioni che si applicano alle applicazioni fastpath.

## Configurazione dei gestori di database

È necessario compiere alcune operazioni perché un gestore di database possa diventare un componente dell'unità di elaborazione globale coordinata dal Queue Manager:

1. Creare un *XA switch load file*<sup>1</sup> per il gestore di database.
2. Definire il gestore di database nel 'file di configurazione Queue Manager, *qm.ini*.

1. Un file switch load XA è un oggetto caricato dinamicamente che consente al Queue Manager e al gestore di database di comunicare fra loro.

## Configurazione dei gestori di database

Sono diversi gli oggetti che devono essere definiti in `qm.ini`.

### Creazione dei switch load file

Le istruzioni relative alla creazione dei file switch load per i gestori di database supportati sono riportate in "Creazione del file switch load Oracle" a pagina 132.

Consultare la documentazione relativa all'installazione MQSeries per ulteriori informazioni sulla procedura di installazione.

I moduli esemplificativi di origine utilizzati per la produzione di file switch load contengono tutti una singola funzione denominata `MQStart`. Una volta caricato il switch load file, il Queue Manager richiama questa funzione e ripristina l'indirizzo di una struttura definita come funzione `XA`. Il file switch load è collegato ad una libreria fornita dal gestore di database, che consente il collegamento tra MQSeries e quel gestore di database.

I moduli esemplificativi di origine utilizzati per la creazione di switch load files per Oracle sono del tipo `oraswit.c`.

### Definizione dei gestori di database

Una volta creato un file switch load per il gestore di database, occorre specificare la sua posizione al Queue Manager. Questa operazione viene realizzata nel file `qm.ini` del Queue Manager nella voce `XAResourceManager`.

È necessario aggiungere una voce `XAResourceManager` per ciascun gestore di database che il Queue Manager dovrà coordinare.

Gli attributi della voce `XAResourceManager` sono i seguenti.

#### **Name=name**

Stringa definita dall'utente che identifica la denominazione del gestore di database.

Il nome è obbligatorio e può avere una lunghezza massima di 31 caratteri. Deve essere unico. Può essere ricavato dalla semplificazione del nome del gestore di database, sebbene, per mantenere l'unicità in situazioni più complesse, converrebbe includervi anche il nome del database dopo l'aggiornamento.

È opportuno che il nome scelto sia significativo poiché il Queue Manager utilizza tale nome per fare riferimento a questa istanza del gestore di database sia nei messaggi che nell'output quando si utilizza il comando `dspmqrn`.

Una volta scelto un nome, **non modificare questo attributo**. Istruzioni relative alla modifica delle informazioni di configurazione sono riportate nella sezione "Modifica delle informazioni di configurazione" a pagina 140.

#### **SwitchFile=name**

Questo è il nome completo del file switch load `XA` del gestore di database. Tale attributo è obbligatorio.

#### **XAOpenString=string**

Questa è una stringa di dati trasmessa nelle chiamate al punto di immissione `xa_open` del gestore di database. Il formato di questa stringa dipende dal gestore di database, ma potrebbe identificare il nome del database da aggiornare.

Tale attributo è facoltativo; se viene omissso verrà acquisita una stringa vuota.

### **XACloseString=string**

Questa è una stringa di dati trasmessa nelle chiamate al punto di immissione `xa_close` del gestore di database. Il formato di questa stringa dipende dal gestore di database.

Tale attributo è facoltativo; se viene omesso verrà acquisita una stringa vuota.

### **ThreadOfControl=THREAD | PROCESS**

Il valore *ThreadOfControl* può essere `THREAD` o `PROCESS`. Viene utilizzato dal Queue Manager per le attività di serializzazione.

Se il gestore di database è “thread-safe”, il valore per *ThreadOfControl* potrà essere `THREAD`, e il Queue Manager potrà collegarsi al gestore di database da sottoprocessi multipli contemporaneamente.

Se il gestore di database non è thread-safe, il valore per *ThreadOfControl* potrà essere `PROCESS`. Il Queue Manager serializza tutte le chiamate al gestore di database in modo che solo una chiamata per volta potrà essere eseguita da un determinato processo.

Consultare la sezione “La voce `XAResourceManager`” a pagina 190 per descrizioni dettagliate di questi attributi.

La sezione “Configurazione Oracle” fornisce ulteriori informazioni relative alle operazioni da effettuare per configurare MQSeries con ognuno dei gestori di database supportati.

---

## Configurazione Oracle

È necessario effettuare le seguenti operazioni:

- Controllare i livelli Oracle e applicare i patch se tale operazione non è già stata eseguita.
- Controllare le impostazioni della variabile di ambiente.
- Abilitare il supporto Oracle XA.
- Creare il file `switch load Oracle`.
- Aggiungere l’informazione di configurazione `XAResourceManager` al file `qm.ini`.
- Modificare i parametri di configurazione Oracle, se necessario.

### **Livelli minimi di supporto per Oracle**

Il livello minimo di Oracle supportato in OpenVMS è 8.1.6.

### **Controllare le impostazioni della variabile di ambiente.**

Verificare che le variabili di ambiente Oracle siano impostate per i processi Queue Manager così come nei processi dell’applicazione. In particolare, le seguenti variabili di ambiente dovrebbero essere sempre impostate **prima** di avviare il Queue Manger:

#### **ORACLE\_HOME**

È la home directory di Oracle

#### **ORACLE\_SID**

È la SID utilizzata da Oracle

### **Abilitazione del supporto XA Oracle**

Assicurarsi che il supporto XA Oracle sia abilitato. In particolare, è necessario creare una libreria Oracle condivisa; ciò avviene durante l’installazione della libreria XA Oracle.

## Configurazione Oracle

Durante l'installazione di Oracle8, le librerie è creata automaticamente. Se è necessario ricreare la libreria, consultare la pubblicazione *Oracle8 Administrator's Reference* relativa alla piattaforma utilizzata.

### Creazione del file switch load Oracle

Il metodo più facile per la creazione di un file switch load Oracle consiste nell'utilizzare il file esemplificativo. Il codice sorgente utilizzato per creare il file switch load Oracle è riportato nella Figura 11.

```
#include <cmqc.h>
#include "xa.h"

extern struct xa_switch_t xaosw;

struct xa_switch_t * MQENTRY MQStart(void)
{
    return(&xaosw);
}
```

Figura 11. Codice sorgente per file switch load Oracle, *oraswit.c*

Il file di intestazione *xa.h* incluso è fornito con MQSeries nella stessa directory in cui è memorizzato *oraswit.c*.

### Creazione del file switch load Oracle nei sistemi OpenVMS

Per creare il file switch load Oracle sui sistemi OpenVMS, *oraswit.c* deve essere compilato e collegato alla libreria client Oracle "*oraclient\_v816.exe*".

1. Creare la directory in cui il file switch load di Oracle, *oraswit*, verrà realizzato.
2. Copiare i seguenti file da *mqs\_examples:[xatm]* in questa directory:
  - *xa.h*
  - *oraswit.c*
3. Compilare il file sorgente copiato (*oraswit.c*). Ad esempio:

```
$ cc oraswit0.c
```

4. Generare il file switch load:

```
$ link/share oraswit0.obj, sys$input/options
ora_root:[util]oraclient_v816.exe/share
SYMBOL_VECTOR=(MQStart=PROCEDURE)
```

5. La procedura *MQStart* è caricata automaticamente in fase di avvio dall'immagine generata. Pertanto non deve essere specificato neanche un nome logico nella tabella di sistema per riferirsi al file generato senza l'estensione ".exe" o al file copiato in *sys\$share*. Ad esempio, se la posizione del file creato è *disk\$a\_device:[a\_directory]oraswit0.exe*, utilizzare uno dei seguenti:

```
$ define/sys/exec oraswit0 disk$a_device:[a_directory]oraswit0
```

o

```
$ copy disk$a_device:[a_directory]oraswit0.exe sys$share:oraswit0.exe
```

## Configurazione Oracle

### Aggiunta dell'informazione di configurazione Oracle XAResourceManager

L'operazione successiva consiste nel modificare il file di configurazione `qm.ini` del Queue Manager, per definire Oracle come componente di unità di elaborazione globali. È necessario aggiungere una voce XAResourceManager con i seguenti attributi:

**Name=name**

Questo attributo è obbligatorio. Scegliere un nome adatto per questo componente. È possibile includere il nome del database che viene aggiornato.

**SwitchFile=name**

Questo attributo è obbligatorio. Il nome completo del file switch load Oracle.



### XAOpenString=string

La stringa open XA per Oracle ha il seguente formato:

```
Oracle_XA+Acc=P//|P/userName/passWord
      +SesTm=sessionTimeLimit
      [+DB=dataBaseName]
      [+GPwd=P/groupPassWord]
      [+LogDir=logDir]
      [+MaxCur=maximumOpenCursors]
      [+SqlNet=connectString]
```

in cui:

**Acc=** è obbligatorio ed è utilizzato per specificare l'informazione di accesso utente. **P//** indica non è prevista nessuna informazione relativa all'utente o alla password e che deve essere utilizzata la forma **ops\$login**. **P/userName/passWord** indica un valido ID utente Oracle e la corrispondente password.

#### SesTm=

È obbligatoria ed è utilizzata per specificare il tempo massimo di inattività per una transazione prima che venga automaticamente eliminata dal sistema. L'unità di tempo è il secondo.

#### DB=

È utilizzata per specificare il nome del database, dove DataBaseName è il nome che i precompilatori Oracle utilizzano per identificare il database. Questo campo è necessario solo se le applicazioni specificano il nome del database (in questo caso, utilizzare la clausola AT nelle relative impostazioni SQL).

#### GPwd=

GPwd è utilizzato per specificare la password di sicurezza del server, dove P/groupPassWord indica il nome del gruppo di password di sicurezza del server. I gruppi di sicurezza del server forniscono un ulteriore livello di protezione per le diverse applicazioni eseguite verso la medesima istanza Oracle. L'impostazione predefinita è un gruppo di sicurezza ORACLE.

#### LogDir=

LogDir è utilizzato per specificare la directory di una macchina locale cui è possibile collegare la libreria di errore XA Oracle e l'informazione sulla traccia. Se non è specificato un valore, viene utilizzata la directory corrente. Accertare che l'utente mqm abbia accesso alla scrittura in questa directory.

#### MaxCur=

MaxCur è utilizzato per specificare il numero di cursori da assegnare quando si apre il database. Tale impostazione svolge la medesima funzione dell'opzione precompilatore, maxopencursors.

#### SqlNet=

SqlNet è utilizzato per specificare la stringa di connessione SQL\*Net utilizzata per collegarsi al sistema. La stringa di connessione può essere del tipo SQL\*Net V1 string, SQL\*Net V2 string o SQL\*Net V2 alias. Questo campo è necessario quando si imposta Oracle su una macchina diversa da Queue Manager.

Consultare *Oracle8 Server Application Developer's Guide* (Numero Parte A54642-01) per ulteriori informazioni.

### XACloseString=string

Oracle non necessita di questo attributo.

## Configurazione Oracle

### **ThreadOfControl=THREAD | PROCESS**

Non è necessario specificare questo parametro sulle piattaforme MQSeries per Compaq OpenVMS.

Per una descrizione più completa di ognuno di questi attributi, consultare la sezione "La voce XAResourceManager" a pagina 190.

Nella Figura 12, il database da aggiornare è denominato MQBankDB. Si raccomanda di aggiungere un attributo LogDir alla stringa XA open in modo che tutte le informazioni relative agli errori e alla traccia vengano collegati alla stessa posizione. Si suppone che il file switch load Oracle sia stato copiato nella directory sys\$share dopo la sua creazione.

```
XAResourceManager:  
Name=Oracle MQBankDB  
SwitchFile=sys$share:oraswit0  
XAOpenString=Oracle_XA+Acc=P/jim/tiger+SesTm=35+LogDir=/tmp/ora.log+DB=MQBankDB
```

Figura 12. Esempio di immissione XAResourceManager per Oracle

## Modifica dei parametri di configurazione Oracle

Il Queue Manager e le applicazioni quando si collegano ad Oracle utilizzano l'ID utente specificato nella stringa open XA.

- **Privilegi del database** L'ID utente specificato nella stringa open XA deve avere i privilegi di accesso alla visualizzazione DBA\_PENDING\_TRANSACTIONS.

Il privilegio necessario può essere assegnato utilizzando il comando seguente, dove userID sta per l'ID utente che deve ricevere l'accesso.

```
grant select on DBA_PENDING_TRANSACTIONS to userID;
```

Consultare il "Capitolo 7. Protezione di oggetti di MQSeries" a pagina 85 per ulteriori informazioni relative alla sicurezza.

---

## Attività di gestione

Per la normale gestione è sufficiente una minima attività di gestione una volta completate le operazioni di configurazione. Il lavoro di gestione è reso più semplice dal fatto che il Queue Manger non è vincolato ai gestori di database. In particolare ciò significa che:

- Il Queue Manager può essere avviato in qualsiasi momento senza prima dover avviare tutti gestore di database.
- Non è necessario arrestare e poi riavviare il Queue Manager se uno dei gestori di database diviene indisponibile.

Questo consente di avviare e arrestare il Queue Manager indipendentemente dal gestore di database e viceversa ciò è possibile anche per il gestore di database se la funzione è supportata.

Se si interrompe il collegamento tra il Queue Manager e il gestore di database, essi dovranno risincronizzarsi non appena torneranno disponibili entrambi.

La risincronizzazione è il processo grazie al quale viene completata l'elaborazione di tutte le unità in sospenso che utilizzano quel database. Solitamente ciò avviene

automaticamente senza la necessità di intervento dell'utente. Il Queue Manager richiede al gestore di database l'elenco delle unità in cui la relativa elaborazione è sospesa. Dopo di che ordina al gestore di database di eseguire il commit o il rollback di ognuna di queste unità di elaborazione.

Dopo l'arresto, il Queue Manager, al riavvio, deve risincronizzarsi con ogni istanza dei gestori di database. Quando un singolo gestore di database diventa indisponibile, sarà necessario risincronizzare solo quel gestore non appena il Queue Manager noterà che esso è nuovamente disponibile.

Il Queue Manager tenta di ristabilire automaticamente il collegamento con un gestore di database non appena una nuova unità di elaborazione globale viene avviata. In alternativa, può essere utilizzato il comando **rsvmqtrn** per risolvere direttamente tutte le unità di elaborazione sospese.

### Unità di elaborazione sospese

Le unità di elaborazione possono trovarsi in uno stato di sospensione se il collegamento con Queue Manager viene interrotto dopo che il gestore di database ha ricevuto l'ordine PREPARE. Il gestore di database deve mantenere i vincoli del database associati agli aggiornamenti finché riceve dal Queue Manager il risultato del COMMIT o del ROLLBACK.

Dal momento che questo blocco impedisce l'aggiornamento alle altre applicazioni ed anche la lettura e la registrazione di database, la risincronizzazione deve avvenire al più presto possibile.

Se per una ragione qualsiasi non è possibile attendere la risincronizzazione automatica, l'utente può utilizzare le funzioni fornite dal gestore di database per eseguire il commit o il rollback degli aggiornamenti manualmente. Questa si definisce una decisione *euristica* e deve essere presa solo come ultima risorsa poiché può compromettere l'integrità dei dati; può accadere infatti che si esegua il commit degli aggiornamenti, mentre tutti gli altri partecipanti eseguono il rollback o viceversa.

È decisamente consigliabile riavviare il Queue Manager oppure utilizzare il comando **rsvmqtrn** una volta riavviato il database, per iniziare la risincronizzazione automatica.

### Utilizzo del comando dspmqtrn

Quando un gestore di database è indisponibile, è possibile utilizzare il comando **dspmqtrn** per controllare lo stato di quelle unità bloccate in cui si sta elaborando un database.

Quando un gestore di database diviene indisponibile, prima di entrare nel processo di commit a doppia fase, viene eseguito il rollback di ogni unità di lavoro in corso coinvolta nell'elaborazione del database. Al successivo riavvio, il gestore di database esegue automaticamente il roll back delle proprie unità di elaborazione in-flight.

Il comando **dspmqtrn** visualizza solo quelle unità di elaborazione in cui uno o più partecipanti sono sospesi, in attesa del COMMIT o del ROLLBACK ordinato dal Queue Manager.

## Attività di gestione

Viene visualizzato lo stato di ogni partecipante di queste unità di elaborazione. Il gestore, le cui risorse non sono state aggiornate dall'unità di elaborazione, non viene visualizzato.

Rispetto ad un'unità di elaborazione sospesa, un gestore risorse indica di aver eseguito una delle seguenti operazioni:

- Prepared** Il gestore risorse è pronto al commit dei propri aggiornamenti.
- Committed** Il gestore risorse ha eseguito il commit dei propri aggiornamenti.
- Rolled-back** Il gestore risorse ha eseguito il rollback dei propri aggiornamenti.
- Participated** Il gestore risorse è un partecipa all'elaborazione, ma non ha ancora eseguito la preparazione, il commit o il rollback dei propri aggiornamenti.

Si noti che il Queue Manager al proprio riavvio non ricorda lo stato dei singoli partecipanti. Al riavvio il Queue Manager deve potersi collegare al gestore di database che partecipava all'elaborazione delle unità sospese, altrimenti tali unità non saranno ripristinate durante la fase di riavvio. In questo caso, il gestore di database riporta lo stato *prepared* fino alla successiva risincronizzazione.

Quando il comando **dspmqtrn** visualizza un'unità di elaborazione sospesa, per prima cosa elenca tutti i gestori di risorse che probabilmente partecipano a quell'elaborazione. A questi viene assegnato un unico identificativo, *RMIId*, utilizzato al posto del *Name* dei gestori risorse, per riportare il loro stato rispetto all'unità di elaborazione sospesa.

La Figura 13 mostra il risultato dell'invio del seguente comando:

```
dspmqtrn -m MY_QMGR
```

```
AMQ7107: Resource manager 0 is MQSeries.  
AMQ7107: Resource manager 1 is Oracle MQBankDB  
AMQ7107: Resource manager 2 is Oracle MQFeeDB  
  
AMQ7056: Transaction number 0,1.  
XID: formatID 5067085, gtrid_length 12, bqual_length 4  
gtrid [3291A5060000201374657374]  
bqual [00000001]  
AMQ7105: Resource manager 0 has committed.  
AMQ7104: Resource manager 1 has prepared.  
AMQ7104: Resource manager 2 has prepared.
```

Figura 13. Esempio di output *dspmqtrn*

L'output dalla Figura 13 mostra che vi sono tre gestori risorse associati al Queue Manager. Il primo gestore risorse, 0, è lo stesso Queue Manager. Gli altri due gestori sono i database Oracle MQBankDB e MQFeeDB.

L'esempio mostra una sola unità sospesa. Viene inviato un messaggio a tutti i tre gestori di risorse e ciò significa che sono stati eseguiti aggiornamenti nell'unità di elaborazione dal Queue Manager e da entrambi i database Oracle.

Gli aggiornamenti eseguiti dal Queue Manager, il gestore risorse 0, risultano *committed*. Gli aggiornamenti eseguiti dai database Oracle risultano nello stato *prepared*, il che significa che Oracle è divenuto indisponibile prima di ricevere l'ordine di eseguire il commit degli aggiornamenti da parte dei database *MQBankDB* e *MQFeeDB*.

L'unità di elaborazione sospesa ha come identificativo esterno XIX. Questo è l'identificativo associato da Oracle agli aggiornamenti.

## Utilizzo del comando `rsvmqtrn`

L'output mostrato nella Figura 13 a pagina 138 riporta una singola unità di elaborazione sospesa nella quale la decisione di eseguire il commit è già stata inviata ad entrambi i database Oracle.

Per poter ripristinare questa unità di elaborazione, è necessario risincronizzare il Queue Manager ed Oracle quando Oracle ritorna disponibile. Il Queue Manager utilizza l'avvio di una nuova unità di elaborazione come occasione in cui tentare di ripristinare il collegamento con Oracle. In alternativa, è possibile comandare esplicitamente al Queue Manager di eseguire la risincronizzazione utilizzando il comando `rsvmqtrn`. È opportuno utilizzare questo comando subito dopo il riavvio di Oracle, in modo che ogni vincolo di database associato all'unità di lavoro in sospeso venga rilasciato quanto prima.

Ciò avviene utilizzando l'opzione `-a` che ordina al Queue Manager di ripristinare tutte le unità di elaborazione sospese. Nel seguente esempio, Oracle viene riavviato in modo che il Queue Manager può ripristinare l'unità di elaborazione sospesa:

```
rsvmqtrn -m MY_QMGR -a
```

Tutte le transazioni sospese sono state ripristinate.

## Risultati diversi ed errori

Il fatto che il Queue Manager utilizzi un protocollo di commit a doppia fase non esclude la possibilità che alcune unità di elaborazione possano essere ripristinate con risultati diversi. Questo può accadere se alcuni partecipanti eseguono il commit dei propri aggiornamenti ed altri eseguono invece il roll back.

Se le unità di elaborazione vengono ripristinate con risultati diversi si ha la grave conseguenza che le risorse condivise non saranno più in uno stato congruente.

Questi risultati sono principalmente dovuti alle decisioni euristiche di agire direttamente sulle unità di elaborazione invece di lasciare che il Queue Manager ripristini da solo le unità.

Qualora il Queue Manager rilevi un danno euristico produce l'informazione FFST<sup>®</sup> e documenta il problema nei relativi log di errore, con uno di questi due messaggi:

- Se il gestore di database esegue il rollback invece del commit:

```
AMQ7606 A transaction has been committed but one or more resource
managers have rolled back.
```

- Se il gestore di database esegue il commit invece del rollback:

```
AMQ7607 A transaction has been rolled back but one or more resource
managers have committed.
```

## Attività di gestione

Vengono inviati ulteriori messaggi che identificano i database danneggiati. È responsabilità dell'utente eseguire attività di ripristino in locale per recuperare la congruenza dei database danneggiati. Si tratta di una procedura complessa, nella quale è necessario prima isolare i database che hanno subito erroneamente il rollback o il commit; quindi si dovrà annullare o ripetere manualmente la modifica effettuata.

Il danno che si verifica a causa degli errori del software è probabilmente più ridotto. Le unità di elaborazione colpite da questo danno riportano il numero delle relative transazioni con il messaggio AMQ7112. I partecipanti potrebbero trovarsi in uno stato incongruente.

```
rsvmqtrn -m MY_QMGR

AMQ7107: Resource manager 0 is MQSeries.
AMQ7107: Resource manager 1 is Oracle MQBankDB
AMQ7107: Resource manager 2 is Oracle MQFeedB

AMQ7112: Transaction number 0,1 has encountered an error.
        XID: formatID 5067085, gtrid_length 12, bqual_length 4
            gtrid [3291A5060000201374657374]
            bqual [00000001]
AMQ7105: Resource manager 0 has committed.
AMQ7104: Resource manager 1 has prepared.
AMQ7104: Resource manager 2 has rolled back.
```

Figura 14. Esempio di output `dspmqrn` per una transazione errata

Il Queue Manager non tenterà il ripristino da questa situazione fino al successivo avvio di Queue Manager. Nella Figura 14 viene mostrato che gli aggiornamenti al gestore risorse **1**, il database MQBankDB, rimangono nello stato *prepared* anche se viene inviato il comando `rsvmqtrn` per ripristinare l'unità di elaborazione.

## Modifica delle informazioni di configurazione

Dopo che il Queue Manager ha iniziato nuovamente a coordinare le unità di elaborazione globali, occorre essere cauti nell'effettuare modifiche ad una qualsiasi voce XAResourceManager nel file `qm.ini`.

Se necessarie, le modifiche al file `qm.ini` possono essere effettuate in qualsiasi momento, ma non diverranno effettive fino al successivo riavvio del Queue Manager. Ad esempio, se occorre modificare la stringa XA open trasmessa al gestore di database, è necessario riavviare il Queue Manager perché il cambiamento produca effetto.

si noti che modificando una voce XAResourceManager si esclude la possibilità che il Queue Manager contatti un determinato gestore di database.

Non modificare *mai* l'attributo *Name* di qualsiasi voce XAResourceManager. Questo attributo identifica in modo esclusivo il collegamento di un determinato gestore di database al Queue Manager. Se si modifica questo elemento distintivo, il Queue Manager deduce che il collegamento con il gestore di database è stato rimosso e che se ne è aggiunto un altro completamente nuovo. Il Queue Manager continuerà ad associare i risultati dell'unità di elaborazione al vecchio *Name*, con la probabilità di lasciare il database in uno stato di sospensione.

### Rimozione delle istanze del gestore di database

Prima di rimuovere in modo permanente un database o un gestore di database dalla configurazione, è necessario accertare che il database non sia in uno stato di sospensione. Effettuare questo controllo prima di riavviare il Queue Manager. Molti gestori di database prevedono appositi comandi per elencare le transazioni sospese. Se esistono transazioni in dubbio, prima di rimuovere la relativa voce XAResourceManager occorre consentire al Queue Manger di effettuare la risincronizzazione con il gestore di database.

Se non si osserva questa raccomandazione il Queue Manager continuerà a ricordare tutte le unità di elaborazione sospese in cui era presente quel database. Viene inviato un messaggio di avviso, AMQ7623, ad ogni riavvio del Queue Manager. Se si è certi di non voler configurare in seguito il Queue Manger con questo gestore di database, è possibile ordinare al Queue Manger di ignorare la partecipazione del database nelle relative transazioni sospese, utilizzando l'opzione -r del comando `rsvmqtrn`.

**Nota:** il Queue Manager ignorerà le transazioni solo quando sarà completato il processo syncpoint con tutti partecipanti.

Può verificarsi il caso in cui si ha la necessità di rimuovere una voce XAResourceManager temporaneamente. Per far ciò il modo migliore è di escludere la voce in modo da poterla poi reinstallare facilmente in seguito. Se si verificano continuamente errori tutte le volte che il Queue Manager contatta un determinato database o gestore di database, potrebbe essere il caso di compiere questa operazione. Rimuovendo temporaneamente la relativa voce XAResourceManager si consente al Queue Manager di avviare le unità di elaborazione globali impegnate dal resto dei partecipanti. Di seguito è riportato un esempio di esclusione della voce *XAResourceManager*:

```
# This database has been temporarily removed
#XAResourceManager:
# Name=Oracle MQBankDB
# SwitchFile=sys$share:oraswit0
# XAOpenString=MQBankDB
```

Figura 15. Esclusione della voce XAResourceManager





---

## Capitolo 11. Ripristino e riavvio

Un sistema di messaggistica garantisce che i messaggi immessi nel sistema siano consegnati alla relativa destinazione. Ciò significa che esso deve fornire un metodo di verifica dei messaggi nel sistema e di ripristino dei messaggi se il sistema dovesse subire un guasto per un qualsiasi motivo.

MQSeries garantisce che i messaggi non vengano persi dai record di manutenzione (log) delle attività dei Queue Manager che gestiscono la ricezione, la trasmissione e la consegna dei messaggi. Tali log sono utilizzati per tre tipi di ripristino:

1. *Ripristino di riavvio*, quando si arresta MQSeries in maniera pianificata.
2. *Ripristino di sistema arrestato in modo anomalo*, quando MQSeries viene arrestato per un malfunzionamento inatteso.
3. *Ripristino supporti*, per ripristinare oggetti danneggiati.

In ogni caso, il ripristino riporta Queue Manager allo stato in cui si trovava quando è stato arrestato. Su tutte le transazioni in corso viene effettuato un roll back rimuovendo dalle code qualsiasi messaggio sui cui è stato eseguito il commit al momento in cui il Queue Manager è stato arrestato. Il ripristino recupera tutti i messaggi permanenti; i messaggi non permanenti vengono persi durante il processo.

La parte successiva di questo capitolo illustra i concetti di ripristino e riavvio più dettagliatamente e quindi indica le modalità di ripristino nell'eventualità si verificassero dei problemi. Essa si compone dei seguenti argomenti:

- “Come assicurarsi che i messaggi non vadano persi (registrazione nei log)”
- “Checkpoint – garantire un completo ripristino” a pagina 146
- “Calcolo delle dimensioni del log” a pagina 149
- “Gestione di log” a pagina 150
- “Utilizzo del log per il ripristino” a pagina 152
- “Protezione dei file di log di MQSeries” a pagina 155
- “Backup e ripristino” a pagina 155
- “Situazioni di ripristino” a pagina 156
- “Scaricamento del contenuto del log mediante il comando `dmpmqlog`” a pagina 158.

---

### Come assicurarsi che i messaggi non vadano persi (registrazione nei log)

MQSeries registra tutte le modifiche significative apportate ai dati controllati dal Queue Manager in un log. Ciò include la creazione e l'eliminazione di oggetti, tutti gli aggiornamenti ai messaggi permanenti, gli stati di transazione, le modifiche agli attributi di oggetto e le attività di canale. Di conseguenza, il log contiene le informazioni necessarie a ripristinare tutti gli aggiornamenti delle code di messaggi nei seguenti modi:

- Tenendo dei record delle modifiche del Queue Manager.
- Tenendo dei record degli aggiornamenti della coda affinché siano utilizzati dal processo di riavvio.
- Consentendo all'utente di ripristinare i dati in seguito al malfunzionamento di hardware o software.

## Registrazione nei log

### Che aspetto hanno i log

Un log di MQSeries consiste in due componenti:

1. Uno o più file di dati di log
2. Un file di controllo del log

Esiste una serie di file di log che contengono i dati che si stanno registrando. È possibile definire il numero e le dimensioni (come illustrato nella sezione “Calcolo delle dimensioni del log” a pagina 149), oppure assumere il valore predefinito di sistema, che è di 3 file, ciascuno delle dimensioni di 4 MB.

Creando un Queue Manager, la quantità di file di log definita corrisponde al numero di file di log *primari* allocati. Se non si specifica alcun numero, sarà utilizzato il valore predefinito. Se non si è modificato il percorso del log, questi saranno creati nella directory:

```
MQS_ROOT:[MQM.LOG.QmName.ACTIVE]
```

MQSeries viene avviato con questi file di log primari, ma, se il log comincia ad essere pieno, alloca file di log *secondari*. Ciò viene eseguito in maniera dinamica e questi saranno rimossi quando si ridurrà la richiesta di spazio per i log. Per impostazione predefinita, è possibile allocare fino a due file di log secondari, fornendo ulteriori 8 MB di spazio su disco. È possibile modificare anche il numero predefinito, consultare il “Capitolo 13. Configurazione MQSeries” a pagina 179.

### File di controllo dei log

Il file di controllo dei log contiene le informazioni necessarie a monitorare l'utilizzo dei file di log: la dimensione e la posizione, il nome del successivo file disponibile e così via.

**Nota:** occorre assicurarsi che i log creati all'avvio di un Queue Manager siano abbastanza grandi da accogliere le dimensioni e il volume di messaggi che le applicazioni gestiranno. Sarà necessario apportare delle modifiche alle quantità e alle dimensioni predefinite dei log al fine di soddisfare i propri requisiti. La procedura di modifica dei valori predefiniti è descritta a pagina 149.

## Tipi di registrazione nei log

In MQSeries, il numero di file utilizzati per la registrazione di log dipende dalle dimensioni del file, dal numero di messaggi ricevuti e dalla lunghezza degli stessi. Esistono due metodi per la manutenzione dei record delle attività dei Queue Manager: la registrazione circolare e la registrazione lineare.

### Registrazione nei log circolare

Adoperare la registrazione nei log circolare se si desidera solo riavviare il ripristino, utilizzando il log per eseguire il roll back delle transazioni in corso al momento dell'arresto del sistema.

La registrazione circolare conserva tutti i dati di riavvio in un ring di file di log. La registrazione di log riempie il primo file nel ring, quindi passa al successivo e così via finché tutti i file sono riempiti. La registrazione torna quindi al primo file nel ring e ricomincia. Ciò continua finché il prodotto è in uso e offre il vantaggio di avere sempre a disposizione file di log.

La precedente è un semplice spiegazione della registrazione nei log circolare. Tuttavia, esiste una complicazione. Le voci di log richieste per il riavvio del Queue Manager senza perdita di dati vengono conservate finché non sono più necessarie

a garantire il ripristino dei dati del Queue Manager. Il meccanismo di rilascio di file di log da riutilizzare è descritto nella sezione “Checkpoint – garantire un completo ripristino” a pagina 146. Per il momento, occorre sapere che MQSeries utilizza file di log secondari per ampliare la capacità del log secondo le esigenze.

### Registrazione nei log lineare

Adoperare la registrazione nei log lineare se si desidera riavviare il ripristino e i supporti o inoltrare il ripristino (ricreando i dati danneggiati o perduti mediante la riproduzione del contenuto del log).

La registrazione lineare conserva i dati di log in una sequenza continua di file. Lo spazio non sarà riutilizzato, in tal modo sarà sempre possibile recuperare qualsiasi record registrato dal momento in cui è stato creato il Queue Manager.

Poiché lo spazio su disco è limitato, potrebbe essere opportuno prendere in considerazione una forma di archiviazione. L'amministratore dovrà occuparsi della gestione dello spazio su disco per la registrazione di log, riutilizzando o ampliando lo spazio esistente secondo le esigenze.

Il numero di file di log utilizzati con la registrazione lineare può essere davvero ingente, a seconda del flusso di messaggi e dall'età del Queue Manager. Tuttavia, esiste una serie di file che dovrebbero essere attivi. I file attivi contengono le voci di log necessarie a riavviare il Queue Manager. Il numero di file di log attivi è di norma identico a quello dei file di log primari, come definito nei file di configurazione (consultare la sezione “Calcolo delle dimensioni del log” a pagina 149 per ulteriori dettagli sulla definizione di tale numero).

L'evento più importante che determina la denominazione di “attivo” per un file è un *checkpoint*. Un MQSeries checkpoint è un gruppo di record di log contenenti informazioni che consentono di riavviare con buon esito il Queue Manager. Qualsiasi informazioni registrata precedentemente non è necessaria a riavviare il Queue Manager e può quindi essere considerata inattiva (consultare la sezione “Checkpoint – garantire un completo ripristino” a pagina 146 per ulteriori informazioni sul checkpoint).

È necessario decidere quando i file di log inattivi non servono più. Si potrebbe decidere di archivarli o di eliminarli, poiché non più interessanti allo scopo delle proprie operazioni. Consultare la sezione “Gestione di log” a pagina 150 per ulteriori informazioni sullo smaltimento dei file di log.

Se viene registrato un checkpoint nel secondo o più recente file di log primario, il primo file diverrà inattivo e sarà formattato un nuovo file primario, che verrà poi aggiunto al termine del pool primario, ripristinando il numero di file primari disponibili per la registrazione di log. In tal modo, il pool di file di log primari può essere considerato come un insieme corrente di file in un elenco infinito di file di log. È compito dell'amministratore di sistema, anche in questo caso, gestire i file inattivi secondo i requisiti delle proprie operazioni.

Sebbene i file di log secondari siano definiti per la registrazione nei log lineare, essi non vengono utilizzati nelle normali operazioni. Qualora non dovesse essere possibile, probabilmente a causa di transazioni in esecuzione da molto tempo, liberare un file dal pool inattivo poiché potrebbe essere ancora necessario a riavviare, i file secondari vanno formattati e aggiunti al pool dei file di log attivi.

## Registrazione nei log

Se il numero di file secondari disponibile è esaurito, le richieste per la maggior parte delle ulteriori operazioni che richiedono attività di log saranno rifiutate con la restituzione di MQRC\_RESOURCE\_PROBLEM all'applicazione.

Entrambi i tipi di registrazione possono far fronte a un'imprevista interruzione di corrente, a condizione che non si verifichi un malfunzionamento dell'hardware.

---

## Checkpoint – garantire un completo ripristino

Gli aggiornamenti permanenti alle code di messaggi si verificano in due fasi. Innanzitutto, i record che rappresentano l'aggiornamento vengono scritti nel log, quindi il file di coda viene aggiornato. I file di log possono quindi essere più aggiornati dei file di coda. Per garantire l'elaborazione ricominci da un punto congruente, MQSeries utilizza i checkpoint. Un checkpoint è un punto nel tempo in cui il record descritto nel log è uguale al record nella coda. Il checkpoint stesso consiste nella serie di record di log necessari per riavviare il Queue Manager; ad esempio, lo stato di tutte le transazioni (cioè, le unità di lavoro) attive al momento del checkpoint.

I checkpoint sono generati automaticamente da MQSeries. Essi sono assunti all'avvio del Queue Manager, all'arresto, quando lo spazio di registrazione dei log è insufficiente e ogni 1000 operazioni registrate. Mentre le code gestiscono ulteriori messaggi, il record del checkpoint diviene incongruente rispetto allo stato corrente delle code.

Quando MQSeries viene riavviato, individua il più recente record di checkpoint nel log. Tali informazioni vengono memorizzate nel file di checkpoint che viene aggiornato al termine di ogni checkpoint. Il record di checkpoint rappresenta il punto più recente di congruenza tra il log e i dati. I dati provenienti da tale checkpoint vengono utilizzati per ricostruire le code così com'erano al momento del checkpoint. Quando le code sono state ricreate, il log viene riprodotto in avanti per riportare le code allo stato in cui erano prima del malfunzionamento o dell'arresto del sistema.

MQSeries mantiene i puntatori interni all'inizio e alla fine del log. Esso sposta il puntatore dall'inizio fino al più recente checkpoint che è congruente con il ripristino dei dati del messaggio.

I checkpoint sono utilizzati per rendere più efficiente il ripristino e per controllare il riutilizzo dei file di log primari e secondari.

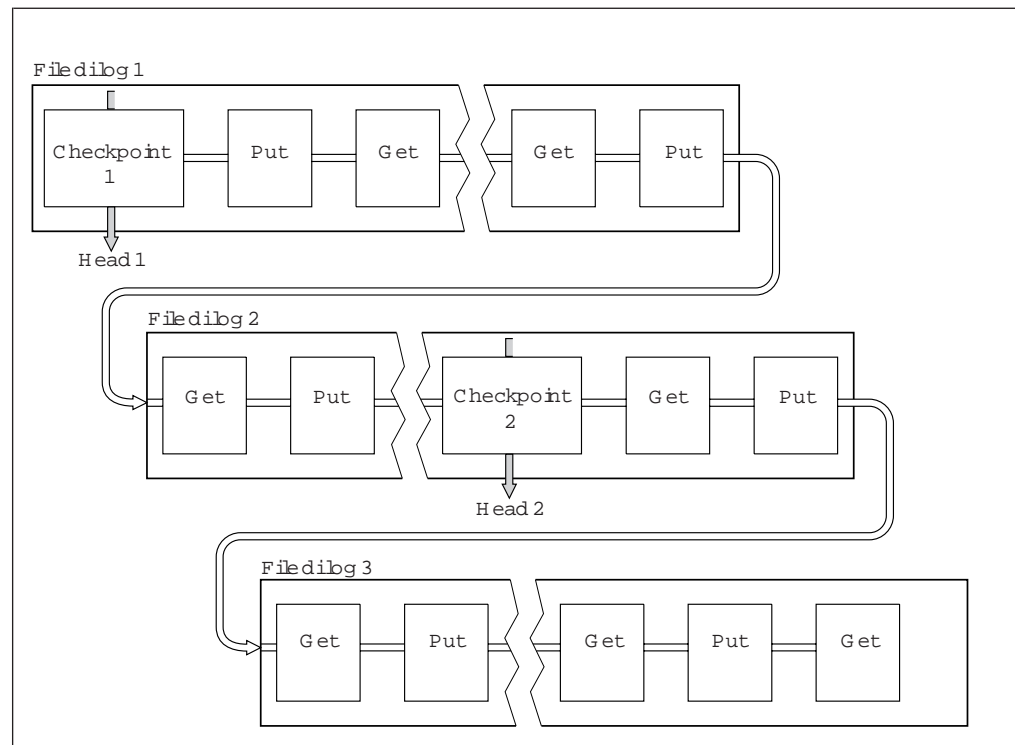


Figura 16. Checkpoint. Per semplicità, solo le estremità dei file di log sono illustrate.

Nella Figura 16, tutti i record prima del più recente checkpoint, ossia il checkpoint 2, non sono più necessari per MQSeries. È possibile ripristinare le code dalle informazioni di checkpoint e da tutte le voci di log recenti. Per la registrazione nei log circolare, è possibile riutilizzare qualsiasi file liberato precedente al checkpoint. Per un log lineare, non è più necessario accedere ai file di log liberati per le normali operazioni; essi divengono quindi inattivi. In questo esempio, il puntatore iniziale della coda viene spostato fino a puntare al più recente checkpoint, Checkpoint 2, il quale diventa il nuovo inizio della coda, head 2. È ora possibile riutilizzare il file di log 1.

## Checkpoint

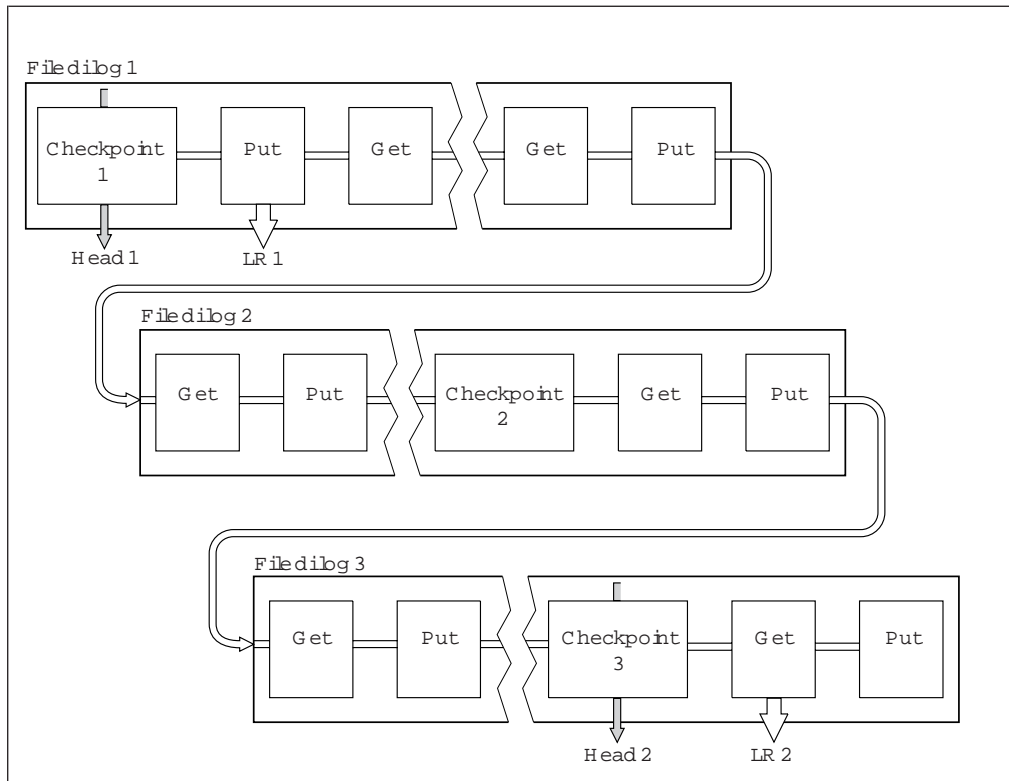


Figura 17. Checkpoint con transazioni in esecuzione da molto tempo. Per semplicità, solo le estremità dei file di log sono illustrate.

La Figura 17 illustra in che modo una transazione in esecuzione da molto tempo influenza il riutilizzo dei file di log. Nell'esempio, una transazione in esecuzione da molto tempo ha dato origine a una voce nel log, mostrata come LR 1, dopo il primo checkpoint illustrato. La transazione non sarà completata, come illustrato da LR 2, fino a dopo il terzo checkpoint. Tutte le informazioni di log da LR 1 in poi sono conservate per consentire il ripristino di tale transazione, se necessario, fino al suo completamento.

Al completamento della transazione in esecuzione da molto tempo, in LR 2, la parte iniziale del log viene spostata al checkpoint 3, l'ultimo checkpoint registrato. I file contenenti record di log precedenti al checkpoint 3, Head 2, non sono più necessari. Se si sta utilizzando la registrazione nei log circolare, lo spazio potrà essere riutilizzato.

Se i file di log primari sono completamente riempiti prima che la transazione in esecuzione da molto tempo sia completata, i file di log secondari saranno utilizzati per impedire l'insorgere del rischio di una situazione di log pieno, se possibile.

Quando la parte iniziale del log viene spostata e si sta utilizzando la registrazione circolare, i file di log primari diventano idonei al riutilizzo e la registrazione, dopo aver riempito il file corrente, riutilizza a tale scopo il primo file primario disponibile. Se invece si sta utilizzando la registrazione lineare, la parte iniziale del log viene sempre spostata in basso nel pool attivo e il primo file diventa inattivo. Un nuovo file primario sarà formattato e aggiunto alla base del pool, pronto per future attività di registrazione nei log.

## Calcolo delle dimensioni del log

Dopo aver deciso se il Queue Manager debba utilizzare la registrazione circolare o lineare, l'attività successiva consiste nello stimare le dimensioni del log di cui avrà bisogno Queue Manager. Le dimensioni del log sono determinate dalla seguente configurazione del log:

### LogFilePages

Le dimensioni di ciascun file di log primario e secondario in unità di pagine da 4 KB

### LogPrimaryFiles

Il numero di file di log primari pre-allocati

### LogSecondaryFiles

Il numero di file di log secondari che è possibile creare affinché siano utilizzati quando i file di log primari sono pieni

La Tabella 6 mostra la quantità di dati che Queue Manager registra per varie operazioni. La maggior parte delle operazioni eseguite da Queue manager richiede una quantità minima di spazio di log, tuttavia, quando si inserisce un messaggio permanente in una coda, tutti i dati del messaggio devono essere scritti nel log per rendere possibile il ripristino del messaggio. Pertanto, le dimensioni del log dipendono, di norma, dal numero e dalle dimensioni dei messaggi permanenti che il Queue Manager deve gestire.

Tabella 6. Dimensioni del sovraccarico del log (tutti i valori sono approssimativi)

Operazione	Dimensione
Inserire un messaggio permanente	750 byte + lunghezza del messaggio. Se il messaggio è di grandi dimensioni, sarà diviso in segmenti da 15700 byte, ciascuno con un sovraccarico di 300 byte.
Ricevere un messaggio	260 byte
Syncpoint, commit	750 byte
Syncpoint, roll back	1000 byte + 12 byte per ogni get o put di cui eseguire il roll back
Creare un oggetto	1500 byte
Elimina oggetto	300 byte
Modificare gli attributi	1024 byte
Registra immagine supporto	800 byte + immagine. L'immagine viene divisa in segmenti da 15700 byte, ciascuno avente un sovraccarico di 300 byte
Checkpoint	750 byte + 200 byte per ciascuna unità attiva di lavoro. È possibile registrare dati aggiuntivi per qualsiasi comando put o get su cui non è stato eseguito il commit e che è stato inserito nel buffer per motivi di prestazioni.

### Note:

1. Il numero di file di log primari e secondari può essere modificato ogni volta che viene avviato il Queue Manager.
2. Le dimensioni del file di log non possono essere modificate e devono essere determinate **prima** della creazione del Queue Manager.

## Checkpoint

3. Il numero di file di log primari e le dimensioni del file di log determinano la quantità di spazio di log che viene pre-allocata alla creazione del Queue Manager. Si consiglia di organizzare tale spazio come un numero minore di file di log più grandi piuttosto che come un numero maggiore di file di log piccoli.
4. Il numero totale di file di log primari e secondari non può superare 63, il quale, in presenza di transazioni in esecuzione da molto tempo, limita la quantità massima di spazio di log che è possibile rendere disponibile al Queue Manager per il ripristino del riavvio. La quantità di spazio di log che il Queue Manager potrebbe necessitare per il ripristino dei supporti non condivide tale limite.
5. Quando si utilizza la registrazione *circolare*, il Queue Manager riutilizza lo spazio di log primario. Ciò significa che il log del Queue Manager può essere inferiore alla quantità stimata di dati che il Queue Manager dovrà registrare. Il Queue Manager allocherà, fino a un certo limite, un file di log secondario quando un file di log si riempie e il successivo file di log primario in sequenza non è disponibile.
6. I file di log primari sono resi disponibili per il riutilizzo durante checkpoint. Il Queue Manager prende in considerazione sia lo spazio di log primario che secondario prima di un checkpoint dal momento che la quantità di spazio è insufficiente.

Se non si definiscono più file di log primari che file di log secondari, il Queue Manager potrebbe allocare file di log secondari prima che sia raggiunto un checkpoint. Ciò rende i file di log primari disponibili per il riutilizzo.

---

## Gestione di log

Nel tempo, alcuni record di log scritti divengono superflui per il riavvio del Queue Manager, che reclama spazio disponibile nei file di log. Tale attività è trasparente all'utente e di norma non si nota la riduzione della quantità di spazio su disco poiché lo spazio allocato viene rapidamente riutilizzato.

Fra i record di log, solo quelli scritti dall'avvio dell'ultimo checkpoint completo e quelli scritti da qualsiasi transazione attiva sono necessari per riavviare il Queue Manager. In questo modo, il log potrebbe riempirsi se un checkpoint non è stato raggiunto da un lungo periodo o se una transazione in esecuzione prolungata ha scritto un record di log molto tempo prima. Il Queue Manager tenta di raggiungere i checkpoint in maniera sufficientemente frequente da evitare il primo problema.

Quando una transazione in esecuzione da molto tempo riempie un log, i tentativi di scrivere i record di log non riescono e alcune chiamate MQI restituiscono MQRC\_RESOURCE\_PROBLEM (viene riservato dello spazio per eseguire il commit o il rollback di tutte le transazioni in corso, in modo che MQCMIT o MQBACK abbiano un buon esito).

Il Queue Manager esegue il roll back delle transazioni che consumano troppo spazio di log. Un'applicazione della cui transazione viene eseguito il roll back in questo modo non è in grado di eseguire operazioni MQPUT o MQGET successive specificando syncpoint sotto la stessa transazione. Un tentativo di inserire o ricevere un messaggio sotto syncpoint in questo stato restituisce MQRC\_BACKED\_OUT. L'applicazione potrebbe quindi emettere MQCMIT, che restituisce MQRC\_BACKED\_OUT o MQBACK e avvia una nuova transazione. Quando la transazione è stato eseguito il roll back di una transazione che consuma troppo spazio di log, il relativo spazio di log viene rilasciato e il Queue Manager continua a funzionare normalmente.



Se il log si riempie, viene emesso un messaggio (AMQ7463). Inoltre, se il log si riempie perché una transazione in esecuzione da molto tempo ha impedito il rilascio dello spazio, viene emesso il messaggio AMQ7465.

Infine, se i record che vengono scritti nel log più velocemente di quanto i processi di manutenzione possano gestirli, viene emesso il messaggio AMQ7466. Se si riceve questo messaggio, occorre aumentare il numero dei file di log o ridurre la quantità di dati elaborati dal Queue Manager.

### Cosa accade quando un disco è pieno

Il componente di registrazione del Queue Manager può far fronte a un disco pieno e a file di log pieni. Se il disco contenente il log si riempie, Queue Manager emette il messaggio AMQ6708 e viene registrato un record di errore.

Vengono creati i file di log nelle loro massime dimensioni, piuttosto che essere ampliati mentre i record di log vengono scritti in essi. Ciò significa che MQSeries può esaurire lo spazio su disco solo quando sta creando un nuovo file. Pertanto, non può esaurire lo spazio quando sta scrivendo un record nel log. MQSeries è sempre al corrente della quantità di spazio disponibile dei file di log esistenti e lo gestisce in maniera conseguente.

Se si riempie l'unità contenente i file di log, si potrebbe essere in grado di liberare spazio su disco. Se si utilizza una registrazione lineare, nella directory di registrazione potrebbero esserci alcuni file di log inattivi che è possibile copiare in un'altra unità o periferica. Se ancora non si dispone di spazio sufficiente, verificare che la configurazione del log nel file di configurazione del Queue Manager sia corretta. Potrebbe essere possibile ridurre il numero di file di log primari o secondari in modo che il log non superi lo spazio disponibile. Si noti che non è possibile modificare le dimensioni dei file di log per un Queue Manager esistente. Il Queue Manager presuppone che tutti i file di log siano della stessa dimensione.

### Gestione di file di log

Se si utilizza la registrazione circolare, assicurarsi che vi sia spazio sufficiente per memorizzare i file di log durante la configurazione del sistema (vedere le sezioni "La voce LogDefaults" a pagina 184 e "La voce Log" a pagina 188). La quantità di spazio su disco utilizzato dal log, incluso lo spazio necessario per i file secondari da creare quando necessario, è limitata dalle dimensioni configurate del disco.

Se si utilizza una registrazione lineare, i file di log vengono aggiunti continuamente durante la registrazione dei dati e la quantità di spazio su disco utilizzata aumenta nel tempo. Se la proporzione di dati registrati è alta, lo spazio su disco viene consumato rapidamente dai nuovi file di log.

Nel tempo, i file di log più obsoleti per la registrazione lineare non saranno più necessari a riavviare il Queue Manager o a eseguire il ripristino dei supporti di qualsiasi oggetto danneggiato. Periodicamente, il Queue Manager emette un paio di messaggi per indicare quale dei file di log è necessario:

- Il messaggio AMQ7467 indica il nome del file di log più obsoleto necessario a riavviare il Queue Manager. Questo file di log e tutti i file di log più recenti devono essere disponibili durante il riavvio di Queue Manager.
- Il messaggio AMQ7468 indica il nome del file di log più obsoleto necessario a eseguire il ripristino dei supporti.

## Gestione di log

Tutti i file di log più obsoleti di questi non devono essere necessariamente in linea. È possibile copiarli in un supporto di archiviazione, come ad esempio il nastro per il ripristino d'emergenza e rimuoverli dalla directory di log attiva. Tutti i file di log necessari per il ripristino dei supporti, ma non per il riavvio, possono essere disattivati da un archivio.

Se un file di log necessario non è stato trovato, viene emesso il messaggio AMQ6767. Rendere disponibile il file di log e tutti i file di log successivi al Queue Manager e ritentare l'operazione.

**Nota:** durante l'esecuzione del ripristino dei supporti, tutti i file di log necessari devono essere disponibili nella directory dei file di log contemporaneamente. Assicurarsi di creare periodicamente immagini dei supporti di qualsiasi oggetto che si potrebbe desiderare di ripristinare, al fine di evitare di esaurire lo spazio su disco necessario a memorizzare tutti i file di log richiesti.

### Posizione dei file di log

Nello scegliere una posizione per i file di log, rammentare che le operazioni saranno gravemente influenzate se MQSeries non riesce a formattare un nuovo log a causa dell'insufficienza di spazio su disco.

Se si utilizza la registrazione circolare, assicurarsi che vi sia spazio sufficiente sull'unità per memorizzare almeno i file di log primari configurati. Occorre inoltre lasciare spazio per almeno un file di log secondario che è necessario per l'eventuale aumento del log.

Se si utilizza una registrazione lineare, occorre prevedere un considerevole spazio disponibile; lo spazio su disco consumato aumenta continuamente durante la registrazione dei dati.

In maniera ideale, i file di log dovrebbero essere memorizzati in un'unità disco separata dai dati del Queue Manager. Ciò implica dei vantaggi in termini di prestazioni. Potrebbe inoltre essere possibile collocare file di log su più unità disco in una disposizione mirror. Ciò assicura protezione contro i malfunzionamenti dell'unità contenente il log. Senza mirroring, potrebbe essere obbligatorio tornare all'ultima copia di riserva del sistema MQSeries.

---

## Utilizzo del log per il ripristino

Esistono diversi modi in cui i dati potrebbero subire danni. MQSeries per Compaq OpenVMS è d'aiuto nel ripristinare il sistema dopo:

- Un danno dell'oggetto dati
- Un'interruzione di corrente nel sistema
- Un malfunzionamento delle comunicazioni
- Un danno del volume del log

Questa sezione illustra l'utilizzo dei log per rimediare a tali problemi e ripristinare il sistema.

### Risoluzione dei problemi

MQSeries può ripristinare il sistema sia dopo un malfunzionamento delle comunicazioni che dopo un'interruzione di corrente. In aggiunta, è spesso possibile ripristinare il sistema in seguito ad altri tipi di problemi, come ad esempio l'eliminazione involontaria di un file.

Nel caso di un malfunzionamento delle comunicazioni, i messaggi restano nelle code finché non vengono rimossi da un'applicazione ricevente. Se il messaggio è in corso di trasmissione, esso rimane in coda di trasmissione finché non sarà trasmesso con esito positivo. Per eseguire il ripristino in seguito a un malfunzionamento delle comunicazioni, è di norma sufficiente riavviare semplicemente i canali mediante la connessione non riuscita.

In caso di interruzione di corrente, quando il Queue Manager viene riavviato, MQSeries ripristina le code nello stato in cui erano al momento del guasto. Ciò garantisce che non si perderanno messaggi permanenti. I messaggi non permanenti vengono eliminati; non sopravvivono all'arresto di MQSeries.

Esistono alcuni modi in cui un oggetto di MQSeries può diventare inutilizzabile, ad esempio a causa di danni involontario. È necessario dunque eseguire il ripristino del sistema completo o di parte di esso. L'azione richiesta dipende da quando viene rilevato il danno, dall'eventualità che il metodo di registrazione preveda il ripristino dei supporti e da quali oggetti sono danneggiati.

### Ripristino dei supporti

Il ripristino dei supporti consiste nella nuova creazione di oggetti a partire dalle informazioni registrate solo in un log lineare. Il ripristino dei supporti non è previsto nella registrazione circolare. Ad esempio, se un file di oggetto viene inavvertitamente eliminato o diviene inutilizzabile per qualche altro motivo, è possibile avvalersi del ripristino dei supporti per ricrearlo. L'informazione di log necessaria al ripristino dei supporti di un oggetto è denominata *immagine del supporto*. Le immagini del supporto possono essere registrate manualmente, mediante il comando `rcdmqimg` o automaticamente in determinate circostanze.

Un'immagine del supporto è una sequenza di record di log contenente un'immagine di un oggetto dalla quale è possibile ricreare l'oggetto stesso.

Il primo record di log necessario a ricreare un oggetto è noto come *record di ripristino supporti*; è da esso che si può ricostruire l'immagine del supporto più recente dell'oggetto. Il record di ripristino supporti di ciascun oggetto è costituito da una delle informazioni registrate durante un checkpoint.

Nel ricreare un oggetto partendo dalla sua immagine del supporto, è inoltre necessario riprodurre qualsiasi record di log che descriva gli aggiornamenti eseguiti dal momento in cui è stata creata l'ultima immagine.

Si consideri, ad esempio, una coda locale che disponga di un'immagine dell'oggetto coda creata prima che un messaggio permanente fosse inserito nella coda. Allo scopo di ricreare l'immagine dell'oggetto più recente, è necessario riprodurre le voci del log che registrano l'inserimento del messaggio nella coda, come pure la riproduzione dell'immagine stessa.

Quando viene creato un oggetto, i record di log scritti contengono informazioni sufficienti a ricreare completamente l'oggetto. Tali record costruiscono la prima immagine del supporto dell'oggetto. Successivamente, le immagini del supporto sono registrate automaticamente dal Queue Manager quando:

- Le immagini di tutti gli oggetti processo e le code non locali sono create ad ogni arresto.
- Le immagini di coda locale sono create quando la coda diventa piena.

## Utilizzo del log

Le immagini del supporto possono essere registrate anche manualmente, mediante il comando **rcdmqimg**, descritto nella sezione “rcdmqimg (Registra immagine supporto)” a pagina 291.

### Ripristino di immagini del supporto

MQSeries ripristina automaticamente alcuni oggetti dalla relativa immagine del supporto nel caso in cui fossero corrotti o danneggiati. Ciò si applica in particolare agli oggetti danneggiati rilevati durante l'avvio di Queue Manager. Se durante l'ultimo arresto del Queue Manager una transazione era incompleta, viene recuperata automaticamente anche tutta la coda interessata, in modo da completare l'operazione di avvio.

È necessario recuperare gli altri oggetti manualmente, utilizzando il comando **rcrmqobj**. Questo comando riproduce i record nel log in modo da creare nuovamente l'oggetto MQSeries. L'oggetto viene ricreato dalla sua immagine più recente trovata nel log, insieme a tutti gli eventi di log occorsi dall'ultimo salvataggio dell'immagine fino all'emissione del comando **recreate**. Se un oggetto MQSeries dovesse danneggiarsi, le uniche azioni valide e possibili sono l'eliminazione oppure una nuova creazione tramite questo metodo. Si noti, tuttavia, che non è possibile ripristinare in tal modo i messaggi non permanenti.

Consultare la sezione “rcrmqobj (Ricrea oggetto)” a pagina 293 per ulteriori dettagli sul comando **rcrmqobj**.

È importante ricordare che, nel tentativo di ripristino di supporti di un oggetto, è necessario disporre del file di log contenente il record di ripristino di supporti e tutti i file di log successivi, disponibili nella directory dei file di log. Se un file di log necessario non è stato trovato, viene emesso il messaggio per l'operatore AMQ6767 e l'operazione di ripristino di supporti ha esito negativo. Se non si creano periodicamente immagini di supporti per gli oggetti che si potrebbe voler creare di nuovo, lo spazio su disco potrebbe essere insufficiente per contenere tutti i file di log necessari a ricreare un oggetto.

### Ripristino degli oggetti danneggiati durante l'avvio

Se il Queue Manager scopre un oggetto danneggiato durante l'avvio, l'azione intrapresa dipenderà dal tipo di oggetto e dall'eventuale configurazione del Queue Manager per supportare il ripristino del supporto.

Se l'oggetto del Queue Manager è danneggiato, il Queue Manager non è in grado di avviarsi fino a quando non ripristina l'oggetto. Se il Queue Manager è configurato con la registrazione lineare, in modo da supportare il ripristino di supporti, MQSeries tenta automaticamente di ricreare l'oggetto di MQSeries dalle relative immagini di supporto. Se il metodo di log selezionato non prevede il ripristino di supporti, è possibile eliminare il Queue Manager oppure ripristinare una copia di riserva.

Se all'arresto del Queue Manager erano attive delle transazioni, saranno necessarie per avviare Queue Manager anche le code locali che contengono messaggi permanenti e non assegnati, inseriti o estratti all'interno di queste transazioni. Se qualcuna di queste code locali è danneggiata e il Queue Manager supporta il ripristino di supporti, esso tenta automaticamente di ricrearli dalle relative immagini di supporti. Se non è possibile ripristinare nessuna di tali code, non sarà possibile avviare MQSeries.

Se durante la procedura di avvio di un Queue Manager che non supporta il ripristino dei supporti vengono rilevate code locali danneggiate contenenti

messaggi non assegnati, le code vengono contrassegnate come oggetti danneggiati e vengono ignorati i relativi messaggi non assegnati. Ciò avviene perché non è possibile eseguire un ripristino degli oggetti danneggiati su tale Queue Manager e l'unica azione possibile consiste nell'eliminarle. Viene emesso il messaggio AMQ7472 per segnalare qualsiasi danno.

### Ripristino degli oggetti danneggiati in altri momenti

Il ripristino di supporti di oggetti è automatico soltanto all'avvio. In altri momenti, quando viene rilevato un oggetto danneggiato, viene emesso il messaggio per l'operatore AMQ7472 e la maggior parte delle operazioni che utilizzano l'oggetto ha esito negativo. Se l'oggetto del Queue Manager viene danneggiato in qualsiasi momento successivo all'avvio, Queue Manager esegue un arresto preventivo. Quando un oggetto è stato danneggiato è possibile eliminarlo oppure, se il Queue Manager utilizza un log lineare, tentarne il ripristino dalla relativa immagine di supporti utilizzando il comando `rcrmqobj` (consultare la sezione "rcrmqobj (Ricrea oggetto)" a pagina 293 per ulteriori dettagli).

---

### Protezione dei file di log di MQSeries

È importante non rimuovere i file di log manualmente quando un Queue Manager di MQSeries è in esecuzione. Se un utente elimina involontariamente (o intenzionalmente) i file di log necessari per il riavvio di Queue Manager, MQSeries non emette alcun errore e continua a elaborare i dati, inclusi i messaggi permanenti. Il Queue Manager sarà arrestato normalmente, ma non sarà possibile riavviarlo. Il ripristino di supporti di messaggi diventa quindi impossibile.

Qualsiasi utente autorizzato a rimuovere i file di log utilizzati da un Queue Manager attivo dispone anche dell'autorizzazione a eliminare altre importanti risorse del Queue Manager (quali i file di autorizzazione, i file di coda, il catalogo di oggetti e i file eseguibili di MQSeries). È quindi possibile danneggiare, forse per inesperienza o persino intenzionalmente, un Queue Manager in esecuzione o inattivo in un modo contro il quale MQSeries non è in grado di proteggersi.

Prestare massima attenzione nel concedere agli utenti privilegi elevati o l'identificativo diritti MQM.

---

### Backup e ripristino

Periodicamente, è possibile effettuare un backup dei dati del Queue Manager per proteggere il sistema da possibili danni dovuti a malfunzionamenti dell'hardware. Tuttavia, essendo spesso breve la durata dei dati di messaggio, si può scegliere di non effettuare backup.

### Esecuzione del backup di MQSeries

Per effettuare un backup dei dati di un Queue Manager, è necessario:

1. Assicurarsi che il Queue Manager non sia in esecuzione.

Se il Queue Manager è in esecuzione, arrestarlo con il comando `endmqm`.

**Nota:** se si tenta di eseguire il backup di un Queue Manager in esecuzione, il backup può non essere congruente a causa degli aggiornamenti in corso quando è stata effettuata la copia dei file.

2. Individuare le directory nelle quali il Queue Manager posiziona i relativi dati e file di log.

## Backup e ripristino

Per individuare tali directory è possibile utilizzare le informazioni contenute nei file di configurazione. Per ulteriori informazioni, consultare il “Capitolo 13. Configurazione MQSeries” a pagina 179.

**Nota:** può risultare difficile comprendere i nomi che compaiono nella directory. Ciò avviene perché i nomi vengono trasformati allo scopo di assicurarne la compatibilità con la piattaforma su cui si utilizza MQSeries. Per ulteriori informazioni sulle trasformazioni del nome, consultare il “Analisi dei nomi di file MQSeries” a pagina 21.

3. Effettuare copie di tutti i dati e delle directory di file di log di Queue Manager, incluse tutte le directory secondarie.  
Assicurarsi di non omettere alcun file, specialmente il file di controllo di log e i file di configurazione. Alcune directory potrebbero essere vuote, ma saranno tutte necessarie nel caso di un ripristino di backup successivo, per cui è consigliabile salvare anche queste.
4. Assicurarsi di conservare le proprietà dei file. È possibile farlo con il comando BACKUP e il parametro /BY\_OWNER.

## Ripristino di MQSeries

Per ripristinare un backup di dati di un Queue Manager, è necessario:

1. Assicurarsi che il Queue Manager non sia in esecuzione.
2. Individuare le directory nelle quali il Queue Manager posiziona i relativi dati e file di log. Tale informazione è conservata nel file di configurazione.
3. Svuotare le directory nelle quali si andranno a posizionare i dati del backup.
4. Copiare i dati di backup del Queue Manager e i file di log nelle corrette posizioni.

Controllare la struttura della directory risultante per assicurarsi di disporre di tutte le directory necessarie.

Consultare l’“Appendice C. Struttura di directory” a pagina 329 per ulteriori informazioni su directory primarie e secondarie di MQSeries.

Assicurarsi di disporre di un file di controllo di log e dei file di log. Inoltre, controllare che i file di configurazione di MQSeries e di Queue Manager siano congruenti in modo che MQSeries possa cercare i dati ripristinati nelle posizioni corrette.

Se il precedente backup e il successivo ripristino dei dati sono avvenuti correttamente, il Queue Manager si avvierà.

**Nota:** anche se i dati e i file di log di Queue Manager sono conservati in differenti directory, è necessario eseguire il backup e il ripristino delle directory contemporaneamente. Se i dati del Queue Manager e i file di log sono stati creati in date differenti, il Queue Manager non è in uno stato valido e probabilmente non si avvierà. In caso di avvio, i dati saranno quasi sicuramente corrotti.

---

## Situazioni di ripristino

Questa sezione prende in considerazione una serie di possibili problemi, indicando come risolverli ed eseguire il ripristino.



## Malfunzionamenti dell'unità disco

Si potrebbero riscontrare alcuni problemi con un'unità disco contenente i dati o il log del Queue Manager, o entrambi. Tra i problemi, sono incluse la perdita o la corruzione dei dati. I tre casi differiscono solo nella parte di dati eventualmente restante.

In *tutti* i casi occorre prima verificare la struttura della directory per rilevare eventuali danni e, se necessario, ripararli. Se si perdono i dati del Queue Manager, esiste il pericolo che la struttura della directory del Queue Manager sia stata danneggiata. In tal caso, è necessario ricreare manualmente la struttura ad albero prima di provare a riavviare il Queue Manager. Dopo aver verificato la presenza di danni strutturali, esiste una serie di misure alternative da adottare, a seconda del tipo di registrazione in uso.

- **Qualora vi fossero danni consistenti alla struttura della directory o eventuali danni al log**, rimuovere tutti i file obsoleti al livello QMgrName, inclusi i file di configurazione, il log e la directory del Queue Manager, ripristinare l'ultimo backup e provare a riavviare Queue Manager.
- **Per la registrazione lineare con il ripristino dei supporti**, verificare che la struttura della directory sia intatta e provare a riavviare Queue Manager. Se il Queue Manager non viene riavviato, ripristinare un backup. Se il Queue Manager viene riavviato, verificare se altri oggetti sono stati danneggiati utilizzando MQSC. Ripristinare quelli rilevati, mediante il comando `rcrmobj`, ad esempio:

```
rcrmobj -m QMgrName -t * *
```

dove QMgrName è il Queue Manager che si sta recuperando. `-t * *` indica che qualsiasi oggetto di qualunque tipo sarà recuperato. Se solo uno o due oggetti sono stati riportati come danneggiati, potrebbe essere opportuno specificare tali oggetti per nome e tipo.

**Nota:** tali comandi non si applicano ai canali.

- **Per la registrazione lineare con il ripristino dei supporti e con un log non danneggiato**, si potrebbe ripristinare un backup dei dati del Queue Manager lasciando invariati i file di log esistenti e il file di controllo del log. L'avvio del Queue Manager applica le modifiche dal log per riportare il Queue Manager nello stato in cui si trovava al momento in cui si è verificato il malfunzionamento.

Tale metodo si basa su due fatti: in primo luogo, è fondamentale che il file di checkpoint sia ripristinato come parte dei dati del Queue Manager. Questo file contiene le informazioni che determinano la quantità di dati nel log da applicare per ottenere un Queue Manager congruente.

In secondo luogo, è necessario disporre del file di log più obsoleto che era necessario per avviare il Queue Manager al momento del backup e di tutti i successivi file di log disponibili nella directory dei file di log.

Qualora ciò non fosse possibile, occorre ripristinare un backup sia dei dati del Queue Manager sia del log, entrambi creati contemporaneamente.

- **Per la registrazione circolare, o per la registrazione lineare senza ripristino dei supporti**, è necessario ripristinare il Queue Manager a partire dall'ultimo backup disponibile. Una volta ripristinato il backup, riavviare Queue Manager e

## Situazioni di ripristino

individuare gli eventuali oggetti danneggiati come sopra. Tuttavia, poiché non si dispone del ripristino dei supporti, sarà necessario considerare altri metodi per ricreare gli oggetti danneggiati.

### Oggetto danneggiato del Queue Manager

Se l'oggetto del Queue Manager è riportato come danneggiato durante il normale funzionamento, Queue Manager esegue un arresto preventivo. Esistono due metodi di ripristino in tali circostanze, a seconda del tipo di registrazione utilizzata:

- **Solo per la registrazione lineare**, eliminare manualmente il file contenente l'oggetto danneggiato e riavviare Queue Manager. Il ripristino di supporti dell'oggetto danneggiato è automatico.
- **Per la registrazione circolare o lineare**, ripristinare l'ultimo backup dei dati del Queue Manager e del log e riavviare il Queue Manager.

### Oggetto singolo danneggiato

Se durante il normale funzionamento viene riportato un oggetto come danneggiato, esistono due metodi di ripristino, a seconda del tipo di registrazione utilizzata:

- **Per la registrazione lineare**, ricreare l'oggetto dalla relativa immagine del supporto.
- **Per la registrazione circolare**, ripristinare l'ultimo backup dei dati del Queue Manager e del log e riavviare il Queue Manager.

### Malfunzionamento del ripristino automatico dei supporti

Se una coda locale necessaria per l'avvio del Queue Manager con un log lineare viene danneggiata e il ripristino automatico dei supporti non riesce, ripristinare l'ultimo backup dei dati del Queue Manager e del log e riavviare il Queue Manager.

---

## Scaricamento del contenuto del log mediante il comando `dmpmqlog`

Il comando `dmpmqlog` può essere utilizzato per scaricare il contenuto del log del Queue Manager. Per impostazione predefinita, vengono scaricati tutti i record di log, cioè il comando inizia a scaricare dall'inizio del log. Di norma ciò avviene dall'inizio dell'ultimo checkpoint completato.

È possibile scaricare il log quando il Queue Manager non è in esecuzione. Poiché il Queue Manager raggiunge un checkpoint durante l'arresto, la parte attiva del log di norma contiene un piccolo numero dei record di log. Tuttavia, è possibile istruire il comando `dmpmqlog` per scaricare ulteriori record di log mediante una delle seguenti opzioni allo scopo di modificare la posizione di avvio del comando `dump`:

- L'opzione più semplice consiste nell'iniziare lo scaricamento dalla *base* del log. La base del log è il primo record nel file di log che ne contiene la parte iniziale. La quantità di dati aggiuntivi in questo caso dipende dalla posizione della parte iniziale del log all'interno del file di log. Se si trova all'inizio del file di log, solo una piccola quantità di dati aggiuntivi viene scaricata. Se la parte iniziale è vicina alla parte finale del file di log, sarà scaricata una quantità più consistente di dati.
- Un'altra opzione consente di specificare la posizione di avvio del dump come record di log individuale. Ogni record di log viene identificato da un *LSN (Log Sequence Number)* univoco. In caso di registrazione circolare, questo record di log iniziale non può essere precedente alla base del log; tale limitazione non si



applica ai log lineari. Potrebbe essere necessario reinstallare i file di log inattivi prima di eseguire il comando. È necessario specificare un LSN valido come posizione iniziale per questa opzione, che può essere ricavato dal precedente output `dmpmqlog`.

Ad esempio, è possibile specificare con la registrazione lineare il `nextlsn` dall'ultimo output `dmpmqlog`. Sarà visualizzato Next LSN in Log File Header, che indica l'LSN del record di log successivo da scrivere. Questo potrà quindi essere utilizzato come posizione iniziale per formattare tutti i record di log che sono stati scritti dall'ultima volta in cui è stato scaricato il log.

- La terza opzione è riservata ai log lineari. È possibile istruire il dumper per avviare la formattazione dei record di log dall'estensione di qualsiasi file di log. In questo caso il dumper del log prevede di trovare tale file di log, e tutti i file successivi, nella stessa directory dei file di log attivi. Tale opzione non si applica ai log circolari, poiché in tal caso il dumper del log non può accedere ai record del log precedenti alla base del log.

L'output dal comando `dmpmqlog` è il Log File Header e una serie di record di log formattati. Il Queue Manager utilizza diversi record di log per registrare le modifiche ai relativi dati.

Alcune delle informazioni che vengono formattate sono utilizzate solo internamente. Il seguente elenco include i record di log più utili:

### Log File Header

Ciascun log dispone di un'unica intestazione del file di log, che viene sempre formattata per prima dal comando `dmpmqlog`. Essa contiene i seguenti campi:

<i>logactive</i>	Il numero di estensioni dei log primari.
<i>loginactive</i>	Il numero di estensioni dei log secondari.
<i>logsize</i>	Il numero di pagine da 4 KB per estensione.
<i>baselsn</i>	Il primo LSN nell'estensione del log contenente la parte iniziale del log.
<i>nextlsn</i>	L'LSN del record di log successivo da scrivere.
<i>headlsn</i>	L'LSN del record di log nella parte iniziale del log.
<i>tailsn</i>	L'LSN che identifica la posizione finale del log.
<i>hflag1</i>	Identifica se il log è CIRCULAR o LOG RETAIN (lineare).
<i>HeadExtentID</i>	L'estensione del log contenente la parte iniziale del log.

### Log Record Header

Ciascun record di log dispone di un'intestazione fissa contenente le seguenti informazioni:

<i>LSN</i>	Il numero di sequenza del log.
<i>LogRecdType</i>	Il tipo del record di log.
<i>XTranid</i>	L'identificativo di transazione associato a questo record di log (se esistente).

A *TranType* di MQI indica una transazione solo MQ. Un *TranType* di XA è implicata in altri gestori di risorse. Gli aggiornamenti implicati all'interno della stessa unità di lavoro sono caratterizzati dallo stesso *XTranid*.

<i>QueueName</i>	La coda associata a questo record log (se esistente).
------------------	---

## Utilizzo di dmpmqlog

<i>Qid</i>	L'identificativo interno univoco per la coda.
<i>PrevLSN</i>	L'LSN del record di log precedente nell'ambito della stessa transazione (se esistente).

### Start Queue Manager

Questo log registra che Queue Manager è stato avviato.

<i>StartDate</i>	La data in cui Queue Manager è stato avviato.
<i>StartTime</i>	L'ora in cui Queue Manager è stato avviato.

### Stop Queue Manager

Questo log registra che Queue Manager è stato arrestato.

<i>StopDate</i>	La data in cui Queue Manager è stato arrestato.
<i>StopTime</i>	L'ora in cui Queue Manager è stato arrestato.
<i>ForceFlag</i>	Il tipo di arresto utilizzato.

### Start Checkpoint

Ciò denota l'avvio di un checkpoint di un Queue Manager.

### End Checkpoint

Ciò denota il termine di un checkpoint di un Queue Manager.

<i>ChkPtLSN</i>	L'LSN del record di log che ha avviato questo checkpoint.
-----------------	---

### Put Message

Registra un messaggio permanente inserito in una coda. Se il messaggio è stato inserito sotto syncpoint, l'intestazione del record del log contiene un *XTranid* nonnull. La parte restante del record contiene:

<i>SpcIndex</i>	Un identificativo del messaggio sulla coda. È possibile utilizzarlo per confrontare il corrispondente MQGET che è stato utilizzato per ricevere questo messaggio dalla coda. In tal caso, sarà rilevato un successivo record di log <i>Get Message</i> contenente gli stessi <i>QueueName</i> e <i>SpcIndex</i> . A questo punto l'identificativo <i>SpcIndex</i> potrà essere riutilizzato per un messaggio put successivo a tale coda.
<i>Data</i>	Nel dump esadecimale relativo a tale record di log sono contenuti vari dati interni seguiti dal Message Descriptor (eyecatcher MD) e gli stessi dati del messaggio.

### Put Part

I messaggi permanenti che sono troppo grandi per un unico record di log sono registrati come record *Put Message* singolo, seguito da più record di log *Put Part*.

<i>Data</i>	Continuazione dei dati del messaggio nel punto in cui il precedente record di log si arrestava.
-------------	---

### Get Message

Vengono registrati solo i get dei messaggi permanenti. Se il messaggio è stato ricevuto sotto syncpoint, l'intestazione del record del log contiene un *XTranid* nonnull. La parte restante del record contiene:

<i>SpcIndex</i>	Identifica il messaggio ricevuto dalla coda. Il record di log <i>Put Message</i> più recente contenente gli stessi <i>QueueName</i> e <i>SpcIndex</i> identifica il messaggio ricevuto.
<i>QPriority</i>	La priorità del messaggio ricevuto dalla coda.

**Start Transaction**

Indica l'avvio di una nuova transazione. Un TranType di MQI indica una transazione solo MQ. Un TranType di XA indica una transazione che implica altri gestori di risorse. Tutti gli aggiornamenti effettuati da tale transazione avranno lo stesso *XTranid*.

**Prepare Transaction**

Indica che il Queue Manager viene preparato a eseguire il commit degli aggiornamenti associati al *XTranid* specificato. Questo record di log viene scritto come parte di un commit a due fasi che implica altri gestori di risorse.

**Commit Transaction**

Indica che il Queue Manager ha eseguito il commit di tutti gli aggiornamenti effettuati da una transazione.

**Rollback Transaction**

Questo record di log denota l'intenzione del Queue Manager di eseguire il roll back di una transazione.

**End Transaction**

Questo log denota il termine di una transazione di roll back.

**Transaction Table**

Questo record viene scritto durante il syncpoint. Esso registra lo stato di ciascuna transazione che abbia apportato aggiornamenti permanenti. Per ogni transazione vengono registrate le seguenti informazioni:

- XTranid*            Identificativo di transazione.
- FirstLSN*        LSN del primo record di log associato alla transazione.
- LastLSN*        LSN dell'ultimo record di log associato alla transazione.

**Transaction Participants**

Questo record di log viene scritto dal componente XA Transaction Manager del Queue Manager. Esso registra i gestori di risorse esterne che partecipano alle transazioni. Per ogni partecipante vengono registrate le seguenti informazioni:

- RMName*        Il nome del gestore risorse.
- RMIId*         L'identificativo del gestore risorse. Esso viene inoltre registrato in record di log *Transaction Prepared* successivi, i quali registrano transazioni globali a cui partecipa il gestore di risorse.
- SwitchFile*    Il file di scambio relativo a tale gestore risorse.
- XAOpenString* La stringa aperta XA relativa a tale gestore risorse.
- XACloseString* La stringa aperta XA relativa a tale gestore risorse.

**Transaction Prepared**

Questo record di log viene scritto dal componente XA Transaction Manager del Queue Manager. Esso indica che la transazione globale specificata è stata preparata con esito positivo. Ciascun gestore risorse partecipante sarà istruito per eseguire il commit. Il *RMIId* di ciascun gestore risorse preparato verrà registrato nel record di log. Se il Queue Manager stesso partecipa alla transazione, sarà presente un *Participant Entry* con un *RMIId* uguale a zero.

**Transaction Forget**

Questo record di log viene scritto dal componente XA Transaction Manager del Queue Manager. Esso segue il record di log *Transaction Prepared* quando la decisione di commit è stata consegnata a ciascun partecipante.

## Utilizzo di dmpmqlog

### Purge Queue

Questo registra il fatto che tutti i messaggi su una coda sono stati eliminati, ad esempio, mediante il comando RUNMQSC CLEAR.

### Queue Attributes

Registra l'inizializzazione o la modifica degli attributi di una coda.

### Create Object

Registra la creazione di un oggetto di MQSeries.

*ObjName*            Il nome dell'oggetto che è stato creato.

*UserId*             L'ID utente che esegue la creazione.

### Delete Object

Registra l'eliminazione di un oggetto di MQSeries.

*ObjName*            Il nome dell'oggetto che è stato eliminato.

La Figura 18 a pagina 163 mostra un output di esempio da un comando **dmpmqlog**. Il dump, che ha avviato l'LSN di un record di log specifico, è stato prodotto utilizzando il seguente comando:

```
dmpmqlog -m "testqm" -s 0:0:0:44162
```

```

AMQ7701: DMPMQLOG command is starting.
LOG FILE HEADER
*****

counter1 . . . . : 23                counter2 . . . . : 23
FormatVersion . . : 2                logtype . . . . : 10
logactive . . . . : 3                loginactive . . . : 2
logsize . . . . . : 1024            pages
base1sn . . . . . : <0:0:0:0>
next1sn . . . . . : <0:0:0:60864>
lowtran1sn . . . . : <0:0:0:0>
minbuff1sn . . . . : <0:0:0:58120>
head1sn . . . . . : <0:0:0:58120>
tail1sn . . . . . : <0:0:0:60863>
logfilepath . . . : ""
hflag1 . . . . . : 1
                    -> CONSISTENT
                    -> CIRCULAR
HeadExtentID . . . : 1                LastEID . . . . . : 846249092
LogId . . . . . : 846249061           LastCommit . . . . : 0
FirstArchNum . . . : 4294967295       LastArchNum . . . . : 4294967295
nextArcFile . . . . : 4294967295       firstRecFile . . . . : 4294967295
firstDlFile . . . . : 4294967295       lastDeleteFile . . . : 4294967295
RecHeadFile . . . . : 4294967295       FileCount . . . . . : 3
frec_trunc1sn . . . : <0:0:0:0>
frec_read1sn . . . : <0:0:0:0>
frec_extnum . . . . : 0                LastCid . . . . . : 0
onlineBkupEnd . . . : 0                softmax . . . . . : 4194304

LOG RECORD - LSN <0:0:0:44162>
*****

HLG Header: lreclsize 212, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ALM Start Checkpoint (1025)
Eyecatcher . . . . : ALRH                Version . . . . . : 1
LogRecdLen . . . . : 192                LogRecdOwnr . . . . : 1024 (ALM)
XTranid . . . . . : TranType: NULL
QueueName . . . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . . : <0:0:0:0>
PrevLSN . . . . . : <0:0:0:0>

No data for Start Checkpoint Record

```

Figura 18. Output di dmpmqlog di esempio (Numero 1 di 13)

## Utilizzo di dmpmqlog

```
LOG RECORD - LSN <0:0:0:44374>
*****

HLG Header: lreclsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ATM Transaction Table (773)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 200                      LogRecdOwnc . . . : 768 (ATM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 1
TranCount . . . : 0

LOG RECORD - LSN <0:0:0:44594>
*****

HLG Header: lreclsize 1836, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : Transaction Participants (1537)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 1816                    LogRecdOwnc . . . : 1536 (T)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Id . . . . . : TLPH
Version . . . . : 1                        Flags . . . . . : 3
Count . . . . . : 2

Participant Entry 0
RMName . . . . : DB2 MQBankDB
RMID . . . . . : 1
SwitchFile . . . : /Development/sbolam/build/devlib/tstxasw
XAOpenString . . :
XACloseString . . :

Participant Entry 1
RMName . . . . : DB2 MQBankDB
RMID . . . . . : 2
SwitchFile . . . : /Development/sbolam/build/devlib/tstxasw
XAOpenString . . :
XACloseString . . :
```

Figura 18. Output di dmpmqlog di esempio (Numero 2 di 13)

```

LOG RECORD - LSN <0:0:0:46448>
*****

HLG Header: lrecsize 236, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ALM End Checkpoint (1026)
Eyecatcher . . . : ALRH                      Version . . . . : 1
LogRecdLen . . . : 216                      LogRecdOwnr . . : 1024   (ALM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

ChkPtLSN . . . . : <0:0:0:44162>
OldestLSN . . . . : <0:0:0:0>
MediaLSN . . . . : <0:0:0:0>

LOG RECORD - LSN <0:0:0:52262>
*****

HLG Header: lrecsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ATM Start Transaction (769)
Eyecatcher . . . : ALRH                      Version . . . . : 1
LogRecdLen . . . : 200                      LogRecdOwnr . . : 768   (ATM)
XTranid . . . . : TranType: MQI      TranNum{High 0, Low 1}
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . . : 1
SoftLogLimit . . : 10000

```

Figura 18. Output di dmpmqlog di esempio (Numero 3 di 13)

## Utilizzo di dmpmqlog

```

LOG RECORD - LSN <0:0:0:52482>
*****

HLG Header: lreclsize 730, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : AQM Put Message (257)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 710                      LogRecdOwnr . . . : 256   (AQM)
XTranid . . . . : TranType: MQI      TranNum{High 0, Low 1}
QueueName . . . : Queue1
Qid . . . . . : {Hash 196836031, Counter: 0}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:52262>

Version . . . . . : 3
SpcIndex . . . . : 1
PrevLink.Locn . . : 36                      PrevLink.Length : 8
PrevDataLink . . : {High 0, Low 2048}
Data.Locn . . . . : 2048                    Data.Length . . . : 486
Data . . . . . :
00000: 41 51 52 48 00 00 00 04 FF FF FF FF FF FF FF FF   AQRH.....
00016: 00 00 00 00 00 00 00 00 00 00 00 01 00 01 01 C0   .....&#192;
00032: 00 00 00 00 00 00 00 01 00 00 00 22 00 00 00 00   .....".
00048: 00 00 00 00 41 4D 51 20 74 65 73 74 71 6D 20 20   ....AMQ testqm
00064: 20 20 20 20 33 80 2D D2 00 00 10 13 00 00 00 00   3C-&#30;.....
00080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .....
00096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .....
00112: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01   .....
00128: 00 00 00 00 00 00 00 22 00 00 00 00 00 00 00 00   .....".
00144: 00 00 00 00 00 00 00 C9 2C B5 C0 25 FF FF FF FF   .....&#26;,&#181;&#192;%....
00160: 4D 44 20 20 00 00 00 01 00 00 00 00 00 00 00 08   MD .....
00176: 00 00 00 00 00 00 01 11 00 00 03 33 20 20 20 20   .....3
00192: 20 20 20 20 00 00 00 00 00 00 00 01 20 20 20 20   .....
00208: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00224: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00240: 20 20 20 20 20 20 20 20 20 20 20 20 20 74 65 73 74   test
00256: 71 6D 20 20 20 20 20 20 20 20 20 20 20 20 20 20   qm
00272: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00288: 20 20 20 20 20 20 20 20 20 20 20 20 73 62 6F 6C   sbol
00304: 61 6D 20 20 20 20 20 04 37 34 38 30 00 00 00 00   am .7480...
00320: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .....
00336: 00 00 00 00 00 00 00 00 20 20 20 20 20 20 20 20   .....
00352: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00368: 20 20 20 20 20 20 20 00 00 00 06 75 74 7A 61   ....utza
00384: 70 69 20 20 20 20 20 20 20 20 20 20 20 20 20 20   pi
00400: 20 20 20 20 20 20 20 31 39 39 37 30 35 31 39   19970519
00416: 31 30 34 32 31 35 32 30 20 20 20 20 00 00 00 00   10421520 ....
00432: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .....
00448: 50 65 72 73 69 73 74 65 6E 74 20 6D 65 73 73 61   Persistent messa
00464: 67 65 20 70 75 74 20 75 6E 64 65 72 20 73 79 6E   ge put under syn
00480: 63 70 6F 69 6E 74   cpoint

```

Figura 18. Output di dmpmqlog di esempio (Numero 4 di 13)



LOG RECORD - LSN <0:0:0:53458>  
 \*\*\*\*\*

HLG Header: lreccsize 734, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : AQM Put Message (257)  
 Eyecatcher . . . : ALRH Version . . . . . : 1  
 LogRecdLen . . . : 714 LogRecdOwnr . . . : 256 (AQM)  
 XTranid . . . . . : TranType: NULL  
 QueueName . . . . : Queue2  
 Qid . . . . . : {Hash 184842943, Counter: 2}  
 ThisLSN . . . . . : <0:0:0:0>  
 PrevLSN . . . . . : <0:0:0:0>

Version . . . . . : 3  
 SpcIndex . . . . . : 1  
 PrevLink.Locn . . . : 36 PrevLink.Length : 8  
 PrevDataLink . . . : {High 0, Low 2048}  
 Data.Locn . . . . . : 2048 Data.Length . . . : 490  
 Data . . . . . :

```

00000: 41 51 52 48 00 00 00 04 FF FF FF FF FF FF FF FF AQRH.....
00016: 00 00 00 00 00 00 00 00 00 00 00 01 00 01 01 C0 .....&#192;
00032: 00 00 00 00 00 00 00 01 00 00 00 26 00 00 00 00 .....&;...
00048: 00 00 00 00 41 4D 51 20 74 65 73 74 71 6D 20 20 ....AMQ testqm
00064: 20 20 20 20 33 80 2D D2 00 00 10 13 00 00 00 00      3C-#30;.....
00080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00112: 00 00 00 00 00 00 00 00 26 00 00 00 00 00 00 01 .....
00128: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....&;.....
00144: 00 00 00 00 00 00 00 C9 2C B6 D8 DD FF FF FF FF .....&#26;,.&#216;.....
00160: 4D 44 20 20 00 00 00 01 00 00 00 00 00 00 00 08 MD .....
00176: 00 00 00 00 00 00 01 11 00 00 03 33 20 20 20 20 .....3
00192: 20 20 20 20 00 00 00 00 00 00 00 01 20 20 20 20 .....
00208: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 .....
00224: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 .....
00240: 20 20 20 20 20 20 20 20 20 20 20 20 74 65 73 74          test
00256: 71 6D 20 20 20 20 20 20 20 20 20 20 20 20 20 20      qm
00272: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 .....
00288: 20 20 20 20 20 20 20 20 20 20 20 20 73 62 6F 6C          sbol
00304: 61 6D 20 20 20 20 20 04 37 34 38 30 00 00 00 00      am      .7480...
00320: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00336: 00 00 00 00 00 00 00 00 20 20 20 20 20 20 20 20 .....
00352: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 .....
00368: 20 20 20 20 20 20 20 20 00 00 00 06 75 74 7A 61          ....utza
00384: 70 69 20 20 20 20 20 20 20 20 20 20 20 20 20 20      pi
00400: 20 20 20 20 20 20 20 20 31 39 39 37 30 35 31 39          19970519
00416: 31 30 34 33 32 37 30 36 20 20 20 20 00 00 00 00      10432706 ....
00432: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00448: 50 65 72 73 69 73 74 65 6E 74 20 6D 65 73 73 61      Persistent messa
00464: 67 65 20 6E 6F 74 20 70 75 74 20 75 6E 64 65 72      ge not put under
00480: 20 73 79 6E 63 70 6F 69 6E 74          syncpoint
  
```

Figura 18. Output di dmpmqlog di esempio (Numero 5 di 13)

## Utilizzo di dmpmqlog

```
LOG RECORD - LSN <0:0:0:54192>
*****

HLG Header: lreclsize 216, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ATM Commit Transaction (774)
Eyecatcher . . . : ALRH                      Version . . . . : 1
LogRecdLen . . . : 196                      LogRecdOwnr . . . : 768 (ATM)
XTranid . . . . : TranType: MQI      TranNum{High 0, Low 1}
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:52482>

Version . . . . : 1
LOG RECORD - LSN <0:0:0:54408>
*****

HLG Header: lreclsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ATM Start Transaction (769)
Eyecatcher . . . : ALRH                      Version . . . . : 1
LogRecdLen . . . : 200                      LogRecdOwnr . . . : 768 (ATM)
XTranid . . . . : TranType: MQI      TranNum{High 0, Low 3}
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 1
SoftLogLimit . . : 10000

LOG RECORD - LSN <0:0:0:54628>
*****

HLG Header: lreclsize 240, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : AQM Get Message (259)
Eyecatcher . . . : ALRH                      Version . . . . : 1
LogRecdLen . . . : 220                      LogRecdOwnr . . . : 256 (AQM)
XTranid . . . . : TranType: MQI      TranNum{High 0, Low 3}
QueueName . . . : Queue1
Qid . . . . . : {Hash 196836031, Counter: 0}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:54408>

Version . . . . : 2
SpIndex . . . . : 1                        QPriority . . . . : 0
PrevLink.Locn . . : 36                     PrevLink.Length : 8
PrevDataLink . . . : {High 4294967295, Low 4294967295}
```

Figura 18. Output di dmpmqlog di esempio (Numero 6 di 13)

```

LOG RECORD - LSN <0:0:0:54868>
*****

HLG Header: lreclsize 240, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : AQM Get Message (259)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 220                      LogRecdOwnr . . . : 256   (AQM)
XTranid . . . . : TranType: NULL
QueueName . . . : Queue2
Qid . . . . . : {Hash 184842943, Counter: 2}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . . : 2
SpIndex . . . . . : 1                      QPriority . . . . . : 0
PrevLink.Locn . . : 36                    PrevLink.Length : 8
PrevDataLink . . : {High 4294967295, Low 4294967295}
LOG RECORD - LSN <0:0:0:55108>
*****

HLG Header: lreclsize 216, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ATM Commit Transaction (774)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 196                      LogRecdOwnr . . . : 768   (ATM)
XTranid . . . . : TranType: MQI   TranNum{High 0, Low 3}
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:54628>

Version . . . . . : 1

LOG RECORD - LSN <0:0:0:55324>
*****

HLG Header: lreclsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ATM Start Transaction (769)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 200                      LogRecdOwnr . . . : 768   (ATM)
XTranid . . . . : TranType: XA
      XID: formatID 5067085, gtrid_length 14, bqual_length 4
            gtrid [3270BDB40000102374657374716D]
            bqual [00000001]
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . . : 1
SoftLogLimit . . : 10000

```

Figura 18. Output di dmpmqlog di esempio (Numero 7 di 13)

## Utilizzo di dmpmqlog

```

LOG RECORD - LSN <0:0:0:55544>
*****

HLG Header: lreclsize 738, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : AQM Put Message (257)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 718                      LogRecdOwnr . . . : 256   (AQM)
XTranid . . . . : TranType: XA
  XID: formatID 5067085, gtrid_length 14, bqual_length 4
      gtrid [3270BDB40000102374657374716D]
      bqual [00000001]
QueueName . . . . : Queue2
Qid . . . . . : {Hash 184842943, Counter: 2}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:55324>

Version . . . . . : 3
SpIndex . . . . . : 1
PrevLink.Locn . . : 36                      PrevLink.Length : 8
PrevDataLink . . : {High 0, Low 2048}
Data.Locn . . . . : 2048                   Data.Length . . . : 494
Data . . . . . :
00000: 41 51 52 48 00 00 00 04 FF FF FF FF FF FF FF FF   AQRH.....
00016: 00 00 00 00 00 00 00 00 00 00 00 01 00 01 01 C0   .....&#192;
00032: 00 00 00 00 00 00 00 01 00 00 00 2A 00 00 00 00   .....*....
00048: 00 00 00 01 41 4D 51 20 74 65 73 74 71 6D 20 20   ....AMQ testqm
00064: 20 20 20 20 33 80 2D D2 00 00 10 13 00 00 00 00   3E-&#30;.....
00080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .....
00096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .....
00112: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01   .....
00128: 00 00 00 00 00 00 00 2A 00 00 00 00 00 00 00 00   .....*.....
00144: 00 00 00 00 00 00 00 C9 2C B8 3E E8 FF FF FF FF   .....&#26;,&#184;>.....
00160: 4D 44 20 20 00 00 00 01 00 00 00 00 00 00 00 08   MD .....
00176: 00 00 00 00 00 00 01 11 00 00 03 33 20 20 20 20   .....3
00192: 20 20 20 20 00 00 00 00 00 00 00 00 01 20 20 20 20   .....
00208: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00224: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00240: 20 20 20 20 20 20 20 20 20 20 20 20 74 65 73 74   test
00256: 71 6D 20 20 20 20 20 20 20 20 20 20 20 20 20 20   qm
00272: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00288: 20 20 20 20 20 20 20 20 20 20 20 20 73 62 6F 6C   sbol
00304: 61 6D 20 20 20 20 20 04 37 34 38 30 00 00 00 00   am .7480...
00320: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .....
00336: 00 00 00 00 00 00 00 20 20 20 20 20 20 20 20 20   .....
00352: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00368: 20 20 20 20 20 20 20 00 00 00 06 75 74 7A 61   ....utza
00384: 70 69 20 20 20 20 20 20 20 20 20 20 20 20 20 20   pi
00400: 20 20 20 20 20 20 20 31 39 39 37 30 35 31 39   19970519
00416: 31 30 34 34 35 38 37 32 20 20 20 20 00 00 00 00   10445872 ....
00432: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .....
00448: 41 6E 6F 74 68 65 72 20 70 65 72 73 69 73 74 65   Another persiste
00464: 6E 74 20 6D 65 73 73 61 67 65 20 70 75 74 20 75   nt message put u
00480: 6E 64 65 72 20 73 79 6E 63 70 6F 69 6E 74   nder syncpoint

```

Figura 18. Output di dmpmqlog di esempio (Numero 8 di 13)

```

LOG RECORD - LSN <0:0:0:56282>
*****

HLG Header: lreclsize 216, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ATM Prepare Transaction (770)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 196                      LogRecdOwnr . . . : 768   (ATM)
XTranid . . . . : TranType: XA
  XID: formatID 5067085, gtrid_length 14, bqual_length 4
      gtrid [3270BDB40000102374657374716D]
      bqual [00000001]
QueueName . . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . . : <0:0:0:0>
PrevLSN . . . . . : <0:0:0:55544>

Version . . . . . : 1

LOG RECORD - LSN <0:0:0:56498>
*****

HLG Header: lreclsize 708, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : Transaction Prepared (1538)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 688                      LogRecdOwnr . . . : 1536 (T)
XTranid . . . . : TranType: XA
  XID: formatID 5067085, gtrid_length 14, bqual_length 4
      gtrid [3270BDB40000102374657374716D]
      bqual [00000001]
QueueName . . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . . : <0:0:0:0>
PrevLSN . . . . . : <0:0:0:0>

Id. . . . . : TLPR
Version . . . . : 1                      Flags . . . . . : 1
Count . . . . . : 3

Participant Entry 0
RMID . . . . . : 0                      State . . . . . : 2

Participant Entry 1
RMID . . . . . : 1                      State . . . . . : 2

Participant Entry 2
RMID . . . . . : 2                      State . . . . . : 2

```

Figura 18. Output di dmpmqlog di esempio (Numero 9 di 13)

## Utilizzo di dmpmqlog

```
LOG RECORD - LSN <0:0:0:57206>
*****

HLG Header: lreclsize 216, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ATM Commit Transaction (774)
Eyecatcher . . . : ALRH                      Version . . . . : 1
LogRecdLen . . . : 196                      LogRecdOwnr . . : 768   (ATM)
XTranid . . . . : TranType: XA
  XID: formatID 5067085, gtrid_length 14, bqual_length 4
      gtrid [3270BDB40000102374657374716D]
      bqual [00000001]
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:56282>

Version . . . . : 1
LOG RECORD - LSN <0:0:0:57440>
*****

HLG Header: lreclsize 224, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : Transaction Forget (1539)
Eyecatcher . . . : ALRH                      Version . . . . : 1
LogRecdLen . . . : 204                      LogRecdOwnr . . : 1536 (T)
XTranid . . . . : TranType: XA
  XID: formatID 5067085, gtrid_length 14, bqual_length 4
      gtrid [3270BDB40000102374657374716D]
      bqual [00000001]
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Id. . . . . : TLFG
Version . . . . : 1                      Flags . . . . . : 0
```

Figura 18. Output di dmpmqlog di esempio (Numero 10 di 13)

```

LOG RECORD - LSN <0:0:0:58120>
*****

HLG Header: lrecsize 212, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ALM Start Checkpoint (1025)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 192                      LogRecdOwnr . . . : 1024 (ALM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

No data for Start Checkpoint Record

LOG RECORD - LSN <0:0:0:58332>
*****

HLG Header: lrecsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ATM Transaction Table (773)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 200                      LogRecdOwnr . . . : 768 (ATM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . . : 1
TranCount . . . . : 0

```

Figura 18. Output di dmpmqlog di esempio (Numero 11 di 13)

## Utilizzo di dmpmqlog

```
LOG RECORD - LSN <0:0:0:58552>
*****

HLG Header: lreclsize 1836, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : Transaction Participants (1537)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 1816                      LogRecdOwnc . . . : 1536 (T)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Id. . . . . : TLPH
Version . . . : 1                            Flags . . . . . : 3
Count . . . . : 2

Participant Entry 0
RMName . . . . : DB2 MQBankDB
RMId . . . . . : 1
SwitchFile . . : /Development/sbolam/build/devlib/tstxasw
XAOpenString . :
XACloseString . :

Participant Entry 1
RMName . . . . : DB2 MQFeeDB
RMId . . . . . : 2
SwitchFile . . : /Development/sbolam/build/devlib/tstxasw
XAOpenString . :
XACloseString . :

LOG RECORD - LSN <0:0:0:60388>
*****

HLG Header: lreclsize 236, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ALM End Checkpoint (1026)
Eyecatcher . . . : ALRH                      Version . . . . . : 1
LogRecdLen . . . : 216                      LogRecdOwnc . . . : 1024 (ALM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

ChkPtLSN . . . . : <0:0:0:58120>
OldestLSN . . . . : <0:0:0:0>
MediaLSN . . . . : <0:0:0:0>
```

Figura 18. Output di dmpmqlog di esempio (Numero 12 di 13)



```

LOG RECORD - LSN <0:0:0:60624>
*****

HLG Header: lrecsize 240, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . . : ALM Stop Queue Manager (1028)
Eyecatcher . . . : ALRH                               Version . . . . . : 1
LogRecdLen . . . : 220                               LogRecdOwnr . . . : 1024 (ALM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 1
StopDate . . . : 19970519                            StopTime . . . . : 10490868
SessionNumber . : 0                                  ForceFlag . . . . : Quiesce

AMQ7702: DMPMQLOG command has finished successfully.

```

Figura 18. Output di dmpmqlog di esempio (Numero 13 di 13)

#### Note per la Figura 18 a pagina 163:

1. Il *headlsn* in *Log File Header* ha un valore di <0:0:0:58120>. Questo è il punto in cui il dump sarebbe iniziato se non fosse stato richiesto un LSN di avvio differente.
2. Il *nextlsn* è <0:0:0:60864> il quale sarà l'LSN del primo record di log che il Queue Manager scriverà al successivo riavvio.
3. Il *HeadExtentID* è 1, che indica che la parte iniziale del log risiede attualmente nel file di log S0000001.LOG.
4. Il primo record di log formattato è un record di log *Start Checkpoint*. Il checkpoint occupa una serie di record di log fino al record *End CheckPoint* allo <0:0:0:46448>.
5. Uno dei record registrati durante il checkpoint è il record di log *Transaction Participants* allo <0:0:0:44594>. Esso descrive dettagliatamente i gestori risorse che partecipano alle transazioni globali coordinate dal Queue Manager.
6. Il record di log *Start Transaction* allo <0:0:0:52262> indica l'avvio di una transazione. Il *XTranid* mostra un *TranType* di MQI, che indica che esiste una transazione locale che include solo aggiornamenti MQ.
7. Il record di log successivo è un *Put Message* che registra gli MQPUT permanenti sotto il syncpoint che ha avviato la transazione. MQPUT è stato eseguito sulla coda *Queue1* e i dati del messaggio vengono registrati come Persistent message put under syncpoint. A questo messaggio è stato allocato uno *SpcIndex* di 1, che verrà confrontato con il successivo MQGET di questo messaggio.
8. Anche il record di log successivo all'LSN <0:0:0:53458> è un record *Put Message*. Questo messaggio permanente è stato inserito in una coda diversa, *Queue2*, ma non è stato eseguito sotto syncpoint dal momento che *XTranid* è *NULL*. Anch'esso ha uno *SpcIndex* di 1, il quale è un identificativo univoco per tale coda particolare.
9. Il record di log successivo all'LSN <0:0:0:54192> esegue il commit del messaggio inserito sotto syncpoint.
10. Nei record di log <0:0:0:54408> e <0:0:0:54628> MQGET avvia una nuova transazione sotto syncpoint per la coda *Queue1*. Lo *SpcIndex* nel record di log *Get Message* è 1 e indica che questo era lo stesso messaggio che era stato inserito nella coda *Queue1* in <0:0:0:52262>.

## Utilizzo di dmpmqlog

11. Il record di log successivo riceve il messaggio che era stato inserito nella coda *Queue2* dall'altro record di log *Put Message*.
12. È stato eseguito il commit di MQGET sotto syncpoint come indicato dal record di log *Commit Transaction* allo <0:0:0:55108>.
13. Infine, un MQBEGIN viene utilizzato per avviare una transazione globale nel record di log *Start Transaction* allo <0:0:0:55324>. Il *XTranid* in questo record di log ha un *TranType* di XA.
14. Il seguente *Put Message* registra un messaggio permanente inserito nella coda *Queue2*. Ciò condivide lo stesso *XTranid* del record di log precedente.
15. Se un record di log *Transaction Prepared* viene scritto per questo *Xtranid*, occorre eseguire il commit dell'intera transazione. L'assenza di un tale record di log può essere considerata come un'indicazione del rollback della transazione. In tal caso viene rilevato un record di log *Transaction Prepared* allo <0:0:0:56498>. Ciò registra il Queue Manager stesso come partecipante con un *RMIId* uguale a zero. Esistono due ulteriori partecipanti, e i relativi *RMIId*s di 1 e 2 possono essere confrontati con il precedente record di log *Transaction Participants*.
16. Durante la fase di commit, il componente XA Transaction Manager del Queue Manager non registra le risposte individuali dei partecipanti. Il log indica solo se è stato eseguito o no il commit degli aggiornamenti del Queue Manager. Il record di log *Commit Transaction* allo <0:0:0:57206> indica che è stato effettivamente eseguito il commit del messaggio sulla coda *Queue2*.
17. Il record di log *Transaction Forget* allo <0:0:0:57440> indica che anche la decisione di commit è stata consegnata agli altri due gestori di risorse. Eventuali malfunzionamenti di questi gestori risorse nell'eseguire il commit degli aggiornamenti saranno stati diagnosticati nei log di errore di Queue Manager.

---

## Capitolo 12. Utilizzo di Name Service

Il Name Service è un servizio installabile che abilita un'applicazione connessa al Queue Manager di aprire quella che ritiene essere una coda locale. Queste code sono invece definite presso un altro Queue Manager di un'altra macchina con l'attributo SCOPE impostato su CELL.

L'applicazione può così eseguire tutte le operazioni possibili per le code remote sulle code aperte in questo modo. L'implementazione fornita utilizza DCE (Distributed Computing Environment), anche se l'utente può compilare un diverso componente che non utilizza DCE.

Per utilizzare il componente Name Service, è necessario definire Name Service ed i relativi componenti installati per un Queue Manager. A questo scopo inserire la voce appropriata nel file di configurazione Queue Manager (qm.ini). Consultare il manuale *MQSeries Programmable System Management* per ulteriori dettagli. Saranno inoltre necessarie anche alcune configurazioni DCE.

---

### Utilizzo di DCE per condividere code relative a Queue Manager diversi

Se i nodi in cui risiedono i Queue Manager fanno parte di un insieme DCE (Distributed Computing Environment), è possibile configurarli in modo da condividere le code. Le applicazioni così possono collegarsi ad un Queue Manager e aprire una coda relativa ad un *altro* Queue Manager di un altro nodo. L'applicazione non riscontra questa differenza; non rileva che la coda attualmente risiede presso un altro Queue Manager (normalmente, il Queue Manager respinge le richieste di apertura provenienti da applicazioni locali se la coda non risiede su quel Queue Manager).

### Operazioni di configurazione per code condivise

Questa sezione descrive le impostazioni per la condivisione di code fra Queue Manager che risiedono su nodi di un insieme DCE.

Per ogni Queue Manager:

1. Configurare tale funzione aggiungendo la voce Name Service al file di configurazione Queue Manager. I contenuti di questa voce sono descritti nel manuale *MQSeries Programmable System Management*. Per richiamare la funzione Name Service, è necessario riavviare il Queue Manager.
2. Utilizzare il comando **endmqm** per arrestare il Queue Manager, se in esecuzione.
3. Per riavviare il Queue Manager utilizzare il comando **strmqm**.
4. Impostare i canali per la messaggistica tra i Queue Manager; consultare la sezione "Preparazione di canali e code di trasmissione per la gestione remota" a pagina 72.

## Condivisione di code

Per ogni coda che si desidera condividere, specificare CELL per l'attributo SCOPE. Ad esempio, utilizzare i seguenti comandi MQSC:

```
DEFINE QLOCAL(GREY.PUBLIC.QUEUE) SCOPE(CELL)
```

oppure

```
ALTER QLOCAL(PINK.LOCAL.QUEUE) SCOPE(CELL)
```

La coda creata o modificata deve appartenere ad un Queue Manager che risiede in un nodo dell'insieme DCE.

---

## Configurazione DCE

Per utilizzare il componente Name Service, è necessario aver installato OSF per DCE (Distributed Computing Environment). Questo servizio abilita applicazioni connesse ad un Queue Manager ad aprire code appartenenti ad altri Queue Manager dell'insieme DCE.

Un esempio DCL, che consente l'esecuzione del Name Service, è riportato in `mqs_examples:dcesetu.com`.

---

## Capitolo 13. Configurazione MQSeries

Questo capitolo spiega come modificare il comportamento di uno specifico Queue Manager o di un nodo, per adattarlo alle proprie esigenze di installazione.

Le informazioni sulle configurazioni di MQSeries vengono modificate cambiando i valori specificati in un gruppo di attributi (o parametri) di configurazione che governa MQSeries.

Le modifiche alle informazioni sulle configurazioni e lo spazio della memoria utilizzato da MQSeries per registrare tali modifiche, costituiscono una piattaforma specifica. Gli utenti di MQSeries per Compaq OpenVMS modificano le informazioni sulle configurazioni modificando i **MQSeries file di configurazione**.

Questo capitolo:

- Descrive gli attributi che possono essere utilizzati per modificare le MQSeries informazioni sulle configurazioni nella sezione "Attributi per modificare informazioni sulle configurazioni MQSeries" a pagina 181.
- Descrive gli attributi che possono essere utilizzati per modificare le informazioni sulle configurazioni di Queue Manager nella sezione "Modifica delle informazioni sulle configurazioni" a pagina 186.
- Fornisce esempi di file `mqs.ini` e `qm.ini` per MQSeries per Compaq OpenVMS nella sezione "Esempio di file `mqs.ini` e `qm.ini`" a pagina 194.

---

### File di configurazione di MQSeries

Utilizzando MQSeries per Compaq OpenVMS è possibile modificare gli attributi di configurazione MQSeries con:

- Un file di configurazione MQSeries (`mqs.ini`) per effettuare modifiche a MQSeries su un nodo integralmente. Esiste un file `mqs.ini` per ogni nodo.
- Un file di configurazione Queue Manager (`qm.ini`) per effettuare modifiche ai singoli Queue Manager. Esiste un file `qm.ini` per ogni Queue Manager del nodo.

Un file di configurazione (cui può farsi riferimento anche come file *stanza* o file *.ini*) contiene una o più voci, che sono gruppi semplificati di linee nel file che insieme svolgono una comune funzione o definiscono una parte del sistema, come ad esempio funzioni log, funzioni canale e servizi installabili.

La modifica al file di configurazione avrà effetto solo al successivo avvio del Queue Manager.

### Modifica dei file di configurazione

Prima di iniziare le operazioni di modifica dei file di configurazione, effettuarne una copia di riserva da utilizzare in caso di necessità!

È possibile modificare i file di configurazione:

- Automaticamente, utilizzando comandi che modificano la configurazione dei Queue Manager del nodo
- Manualmente, utilizzando un editor di testo standard

## File di configurazione

È possibile modificare i valori predefiniti nei file di configurazione MQSeries dopo l'installazione.

Se viene impostato un valore non corretto in un attributo del file di configurazione, il valore impostato è ignorato e viene visualizzato un messaggio operatore che indica il problema. In questo caso è come se mancasse del tutto l'attributo.

Quando si crea un nuovo Queue Manager, è necessario:

- Eseguire il backup del file di configurazione MQSeries
- Eseguire il backup del file di configurazione del nuovo Queue Manager

### Casi in cui occorre modificare un file di configurazione

È necessario modificare un file di configurazione se, ad esempio:

- Si è perso un file di configurazione; ripristinarlo dalla copia di riserva se possibile.
- Si ha l'esigenza di spostare uno o più Queue Manager in una nuova directory.
- Si ha l'esigenza di modificare il Queue Manager predefinito; ciò può accadere se involontariamente si elimina il Queue Manager esistente.
- Il Centro di supporto IBM raccomanda questa operazione.

### Priorità del file di configurazione

I valori di attributo di un file di configurazione sono impostati in base alle seguenti priorità:

- I parametri immessi nella linea del comando hanno prevalgono sui valori definiti nei file di configurazione.
- I valori definiti nei file `qm.ini` prevalgono sui valori definiti nei file `mqs.ini`.

### Implementazione delle modifiche ai file di configurazione

Quando si modifica un file di configurazione, i cambiamenti non vengono implementati immediatamente dal Queue Manager. Le modifiche apportate al file di configurazione MQSeries vengono implementate solo al successivo avvio di MQSeries. Le modifiche apportate al file di configurazione di Queue Manager viene implementato solo al successivo avvio di Queue Manager. Se il Queue Manager è in esecuzione mentre vengono effettuate le modifiche, è necessario arrestare e poi riavviare il Queue Manager perché le modifiche vengano riconosciute dal sistema.

## File di configurazione MQSeries, mqs.ini

Il file di configurazione MQSeries, `mqs.ini`, contiene informazioni rilevanti per tutti i Queue Manager del nodo. Esso viene creato automaticamente durante l'installazione. In particolare, il file `mqs.ini` è utilizzato per cercare i dati associati a ciascun Queue Manager.

Il file `mqs.ini` è memorizzato nella directory di dati predefinita, **MQS\_ROOT:[MQM]**.

Il file `mqs.ini` contiene:

- I nomi dei Queue Manager
- Il nome del Queue Manager predefinito
- La posizione dei file associati con ognuno di essi.

Per ulteriori informazioni relative ai contenuti `mqs.ini`, consultare la sezione "Attributi per modificare informazioni sulle configurazioni MQSeries" a pagina 181.

## File di configurazione Queue Manager, qm.ini

Un file di configurazione Queue Manager, `qm.ini`, contiene informazioni rilevanti per uno specifico Queue Manager. Esiste un file di configurazione per ogni Queue Manager. Il file `qm.ini` viene creato automaticamente insieme al Queue Manager cui è associato.

Un file `qm.ini` è conservato nella struttura della directory relativa al Queue Manager.

Ad esempio, in MQSeries per Compaq OpenVMS, il percorso ed il nome relativo ad un file di configurazione di Queue Manager definito `QMNAME` é:

```
MQS_ROOT:[MQM.QMGRS.QMNAME]QM.INI
```

**Nota:** il nome di un Queue Manager può avere la lunghezza massima di 48 caratteri. requisito non è in grado di garantire l'unicità o la validità del nome. Pertanto il nome della directory viene creato basandosi sul nome del Queue Manager. Questo processo è definito **trasformazione del nome**. Per una descrizione di questo processo, consultare la sezione "Analisi dei nomi di file MQSeries" a pagina 21.

Per ulteriori informazioni relative a `qm.ini`, consultare la sezione "Modifica delle informazioni sulle configurazioni" a pagina 186.

---

## Attributi per modificare informazioni sulle configurazioni MQSeries

I seguenti gruppi di attributi vengono visualizzati in `mq.s.ini`:

- La voce `AllQueueManagers`
- "La voce `ClientExitPath`" a pagina 182
- "La voce `DefaultQueueManager`" a pagina 183
- "La voce `ExitProperties`" a pagina 183
- "La voce `LogDefaults`" a pagina 184
- "La voce `QueueManager`" a pagina 186

Un esempio di `mq.s.ini` è mostrato nella sezione "Esempio di file `mq.s.ini` e `qm.ini`" a pagina 194.

## La voce `AllQueueManagers`

La voce `AllQueueManagers` può indicare:

- Il percorso per la directory `qmgrs` in cui sono memorizzati i file associati con un Queue Manager.
- Il metodo di conversione dei dati di formato EBCDIC in formato ASCII

**DefaultPrefix**=*directory\_name*

Questo attributo specifica il percorso per la directory `qmgrs`, al cui interno sono conservati i dati del Queue Manager.

Se viene modificato il prefisso predefinito per il Queue Manager, è necessario replicare la struttura della directory creata al momento dell'installazione (consultare l'"Appendice C. Struttura di directory" a pagina 329).

In particolare, per creare la struttura `qmgrs`: arrestare MQSeries prima di cambiare il prefisso predefinito; quindi, dopo aver spostato le strutture e cambiato il prefisso predefinito, riavviare MQSeries.

In alternativa è possibile utilizzare `MQSPREFIX` logico per sovrascrivere il `DefaultPrefix` per il comando `crtmqm`.

## Modifica del file di configurazione MQSeries

### ConvEBCDICNewline=NL\_TO\_LF|TABLE|ISO

EBCDIC code page contiene un carattere NL (New Line) non supportato dai code page ASCII; tuttavia, alcune varianti ISO del formato ASCII contengono un equivalente.

Utilizzare l'attributo ConvEBCDICNewline per specificare il metodo MQSeries da utilizzare per la conversione del carattere EBCDIC NL nel formato ASCII.

### NL\_TO\_LF

Specificare NL\_TO\_LF se si desidera convertire il carattere EBCDIC NL (X'15') nel carattere ASCII LF (Line Feed) (X'0A'), per tutte le conversioni EBCDIC in ASCII.

NL\_TO\_LF è predefinita.

### TABLE

Specificare TABLE se si desidera convertire il carattere EBCDIC NL in base alle tabelle di conversione utilizzate sulla piattaforma per tutte le conversioni EBCDIC in ASCII.

Notare che l'effetto di questo tipo di conversione può variare da piattaforma a piattaforma e da linguaggio a linguaggio, mentre sulla medesima piattaforma può variare il carattere in base all'utilizzo di differenti CCSIDs.

### ISO

Specificare ISO se si desidera:

- Convertire ISO CCSIDs utilizzando il metodo TABLE
- Convertire tutti gli altri CCSIDs utilizzando il metodo NL\_TO\_LF.

Esempi di possibili ISO CCSIDs sono riportati nella Tabella 7.

Tabella 7. Elenco di possibili ISO CCSIDs

CCSID	Code Set
819	ISO8859-1
912	ISO8859-2
915	ISO8859-5
1089	ISO8859-6
813	ISO8859-7
916	ISO8859-8
920	ISO8859-9
1051	roman8

Nel caso in cui ASCII CCSID non sia un gruppo secondario ISO, ConvEBCDICNewline viene modificato per impostazione predefinita in NL\_TO\_LF.

Per ulteriori informazioni sulla conversione dei dati, consultare il manuale *MQSeries Application Programming Guide* o la sezione "Conversione dei dati" a pagina 81.

## La voce ClientExitPath

La voce ClientExitPath specifica il percorso predefinito per la posizione dell'uscita del canale del client.



**ExitsDefaultPath=***defaultprefix*

L'attributo ExitsDefaultPath specifica il prefisso predefinito per la piattaforma.

### La voce DefaultQueueManager

La voce DefaultQueueManager specifica il Queue Manager per il nodo.

**Name=***default\_queue\_manager*

Il Queue Manager predefinito elabora ogni comando per il quale non è espressamente specificato il nome del Queue Manager. L'attributo DefaultQueueManager è aggiornato automaticamente se viene creato un nuovo Queue Manager predefinito. Se il nuovo Queue Manager viene creato involontariamente, e si desidera riabilitare il precedente, è necessario modificare l'attributo DefaultQueueManager manualmente.

### La voce ExitProperties

La voce ExitProperties specifica le opzioni di configurazione utilizzate dai programmi di uscita di Queue Manager.

**CLWLMode=**SAFE | **FAST**

CLWL (CLuster WorkLoad exit) consente di specificare quale coda cluster nel cluster deve essere aperta per rispondere ad una chiamata MQAPI (MQOPEN o MQPUT e così via). CLWL exit opera in FAST mode o in SAFE mode, dipende dal valore specificato nell'attributo CLWLMode. Se non è stata inserita alcuna specifica nell'attributo CLWLMode, il cluster opera in SAFE mode.

#### SAFE

L'opzione SAFE specifica che il CLWL exit esegue un processo separato rispetto al Queue Manager. Questa è l'impostazione predefinita.

Se si verifica un problema durante l'esecuzione di CLWLexit in SAFE mode, si verificano le seguenti conseguenze:

- Il processo server CLWL (amqzlw0) si interrompe
- Queue Manager riavvia il processo server CLWL
- Il problema viene riportato nel log degli errori. Se una chiamata MQAPI è in elaborazione, viene inviato un codice di errore.

L'integrità del Queue Manager è preservata.

**Nota:** esiste un rischio associato all'esecuzione di CLWL exit in un processo separato, che può influenzare le prestazioni.

#### **FAST**

Specificare FAST se si desidera eseguire il cluster exit in linea con il processo del Queue Manager.

Specificando questa opzione si migliorano le prestazioni, allontanando i rischi associati all'esecuzione in SAFE mode, ma ciò avviene a discapito dell'integrità di Queue Manager. Per questo motivo è consigliabile eseguire CLWL exit in FAST mode solo se si è sicuri che **non** si verificheranno problemi con CLWL exit e si è particolarmente preoccupati dei rischi relativi alla prestazione.

Se si verifica un problema durante l'esecuzione di CLWL exit in FAST mode, il Queue Manager si arresterà con il rischio che la sua integrità venga compromessa.

## Modifica del file di configurazione MQSeries

### La voce LogDefaults

La voce `LogDefaults` specifica gli attributi log predefiniti per il nodo. Gli attributi log vengono utilizzati come valori predefiniti quando si crea un Queue Manager, ma possono essere sovrascritti specificando gli attributi log con il comando `crtmqm`. Consultare la sezione “`crtmqm` (Crea Queue Manager)” a pagina 258 per dettagli relativi a questo comando.

Una volta creato un Queue Manager, gli attributi log vengono letti dalla relativa voce log nel file `qm.ini`.

L'attributo `DefaultPrefix` (nella voce `AllQueueManagers`) e l'attributo `LogPath` nella voce `LogDefaults` consentono di porre su Queue Manager su differenti unità. Questo è il metodo raccomandato, sebbene, per impostazione predefinita, essi sono sulla stessa unità.

Per informazioni relative al calcolo delle dimensioni di log consultare la sezione “Calcolo delle dimensioni del log” a pagina 149.

**Nota:** i limiti riportati nel seguente elenco di parametri sono limiti imposti da MQSeries. I limiti del sistema operativo possono ridurre le dimensioni massime di log.

#### **LogPrimaryFiles=3 | 2-62**

Primary log files sono i file assegnati durante la creazione per un utilizzo futuro.

Il numero minimo di file di log primari è 2 ed il numero massimo è 62. Il numero predefinito è 3.

Il numero totale di file di log primari e secondari non deve essere superiore a 63 ed inferiore a 3.

Tale valore è sovrascritto dal parametro `-lp` del comando `crtmqm` quando si esegue la creazione del Queue Manager.

#### **LogSecondaryFiles=2 | 1-61**

Secondary log files sono file di log assegnati dopo l'esaurimento dei file primari.

Il numero minimo di file secondari è 1 ed il numero massimo è 61. Il numero predefinito è 2.

Il numero totale di file di log primari e secondari non deve essere superiore a 63 ed inferiore a 3.

Tale valore è sovrascritto dal parametro `-ls` del comando `crtmqm` quando si esegue la creazione del Queue Manager.

#### **LogFilePages=number**

I dati di log sono conservati in una serie di file definiti file di log. La dimensione dei file di log è fissata in unità di pagine da 4 KB.

Per MQSeries per Compaq OpenVMS, il numero predefinito di pagine di file di log è 1024, riconoscendo uno spazio al file di log di 4 MB. Il numero minimo è 64 ed quello massimo è 16 384.

Tale valore è sovrascritto dal parametro `-lf` del comando `crtmqm` quando si esegue la creazione del Queue Manager.

## Modifica del file di configurazione MQSeries

### **LogType=CIRCULAR|LINEAR**

L'attributo LogType è utilizzato per definire il tipo da utilizzare. Il tipo predefinito è CIRCULAR.

#### **CIRCULAR**

Impostare questo valore se si desidera eseguire il riavvio del ripristino utilizzando il log per eseguire il roll back delle transazioni che erano in elaborazione nel momento in cui si è verificato l'arresto del sistema.

Consultare la sezione "Registrazione nei log circolare" a pagina 144 per una spiegazione esauriente della registrazione dei log circular.

#### **LINEAR**

Impostare questo valore se si desidera sia il ripristino riavvio (restart recovery) sia il ripristino oggetto (media recovery), creando i dati persi o danneggiati mediante la sostituzione con i contenuti del log.

Consultare la sezione "Registrazione nei log lineare" a pagina 145 per una spiegazione esauriente relativa alla registrazione dei log linear.

Se si desidera modificare il tipo di log predefinito modificare l'attributo LogType nel file `mqs.ini`. In alternativa è possibile sovrascrivere l'impostazione predefinita specificando la registrazione dei log linear utilizzando il parametro `-ll` del comando `crtmqm`. Non è possibile modificare il metodo della registrazione dei log dopo aver eseguito la creazione di un Queue Manager.

### **LogBufferPages=17|4-32**

La quantità di memoria assegnata alla registrazione del buffer di scrittura è configurabile. La dimensione dei buffer è prefissata in unità di pagine da 4 KB.

Il numero minimo di pagine buffer è 4 ed il massimo è 32. Maggiori buffer provocano maggiore elaborazione, specialmente per messaggi estesi.

Il numero predefinito di pagine buffer è 17, corrispondente a 68 KB.

Il valore viene esaminato quando il Queue Manager viene creato o avviato e può essere aumentato o diminuito in entrambi questi momenti. Tuttavia, una modifica del valore non sarà attiva prima del successivo riavvio.

### **LogDefaultPath=directory\_name**

È possibile specificare la directory nella quale risiedono i file di log relativi ad un Queue Manager. La directory dovrebbe risiedere in un dispositivo locale al quale il Queue Manager può accedere e, preferibilmente, dovrebbe trovarsi in un'unità diversa rispetto alle code di messaggio. Specificando un'unità differente aumenta la protezione in caso di malfunzionamento del sistema.

L'impostazione predefinita per MQSeries per Compaq OpenVMS è `MQS_ROOT:[MQM.LOG]`.

In alternativa, è possibile specificare il nome di una directory sul comando `crtmqm` utilizzando il flag `-ld`. Quando viene creato un Queue Manager, viene creata anche una directory sotto la directory del Queue Manager, nella quale vengono conservati i file di log. Il nome di questa directory deriva dal nome del Queue Manager. Questo assicura un unico percorso per i file di log ed anche il rispetto delle regole relative alla lunghezza del nome delle directory.

Se non si specifica `-ld` nel comando `crtmqm`, viene utilizzato il valore predefinito dell'attributo LogDefaultPath nel file `mqs.ini`, che è `MQS_ROOT:[MQM.LOG]`.

## Modifica del file di configurazione MQSeries

Il nome del Queue Manager dipende dal nome della directory del file di log in modo da assicurare che i diversi Queue Manager utilizzino diverse directory di log.

Quando viene creato un Queue Manager, viene creato anche un valore LogPath nella voce Log del file `qm.ini` fornendo il nome completo della directory per i file di log del Queue Manager. Questa impostazione è utile per localizzare i file di log quando il Queue Manager viene avviato o eliminato.

## La voce QueueManager

Esiste una voce `QueueManager` per ogni Queue Manager. Questi attributi specificano il nome del Queue Manager e il nome della directory che contiene i file associati a quel Queue Manager. Il nome di questa directory deriva dal nome del Queue Manager, ma viene modificato se il nome del Queue Manager non è un valido nome di file.

Consultare la sezione “Analisi dei nomi di file MQSeries” a pagina 21 per ulteriori informazioni relative alla trasformazione del nome.

**Name**=*queue\_manager\_name*

Questo attributo specifica il nome del Queue Manager.

**Prefix**=*prefix*

Questo attributo specifica dove vengono memorizzati i file del Queue Manager. Per impostazione predefinita, viene utilizzato lo stesso valore specificato nell'attributo `DefaultPrefix` della voce `AllQueueManager` nel file `mq5.ini`.

**Directory**=*name*

Questo attributo specifica il nome della sottodirectory dove vengono memorizzati i file del Queue Manager. Normalmente viene utilizzata la sottodirectory `MQS_ROOT:[MQM.QMGRS]` a meno che non venga specificato un diverso valore per il prefisso. Questo nome dipende dal nome del Queue Manager, ma può essere modificato se quel nome esiste già o se il nome del Queue Manager non è un valido nome di file.

---

## Modifica delle informazioni sulle configurazioni

I seguenti gruppi di attributi possono essere visualizzati in un particolare file `qm.ini` relativo ad un dato Queue Manager o utilizzate per sovrascrivere i valori impostati in `mq5.ini`.

- “La voce `Service`”
- “La voce `ServiceComponent`” a pagina 187
- “La voce `Log`” a pagina 188
- “La voce `XAResourceManager`” a pagina 190
- “La voce `Channels`” a pagina 191
- “Le voci `LU62` e `TCP`” a pagina 193
- “La voce `ExitPath`” a pagina 194

## La voce Service

La voce `Service` specifica il nome di un servizio installabile e il numero di punti di accesso a quel servizio. Deve esserci una voce `Service` per ogni servizio utilizzato.

Per tutti componenti di un servizio deve esserci una voce `ServiceComponent`, che identifica il nome e il percorso del modulo che contiene il codice per quel componente. Consultare la sezione `La voce ServiceComponent` per ulteriori informazioni.

## Modifica del file di configurazione MQSeries

### **Name=AuthorizationService | NameService**

Specifica il nome del servizio richiesto.

#### **AuthorizationService**

In MQSeries, il componente Authorization Service è definito Object Authority Manager o OAM.

In MQSeries per Compaq OpenVMS, la voce AuthorizationService e la relativa voce associata ServiceComponent sono aggiunte automaticamente quando viene eseguita la creazione del Queue Manager, ma può essere sovrascritto attraverso l'utilizzo del comando mqsnoaut, impostando mqsnoaut prima di creare il Queue Manager (consultare la sezione "Disabilitazione di OAM (Object Authority Manager)" a pagina 89 per ulteriori informazioni). Ogni ulteriore voce ServiceComponent deve essere aggiunta manualmente.

#### **NameService**

La voce NameService deve essere aggiunta manualmente al file qm.ini per abilitare la funzione name service.

### **EntryPoints=number-of-entries**

Specificare il numero dei punti di immissione definiti per il servizio. Questo include i punti di immissione di inizializzazione e di terminazione.

Per ulteriori informazioni relative ai servizi e componenti installabili, consultare il manuale *MQSeries Programmable System Management*.

Per ulteriori informazioni relative ai servizi di sicurezza, consultare il "Capitolo 7. Protezione di oggetti di MQSeries" a pagina 85.

## La voce ServiceComponent

La voce ServiceComponent identifica il nome e il percorso del modulo che contiene il codice per quel componente.

È possibile che ci sia più di una voce ServiceComponent per ogni servizio, ma ogni voce ServiceComponent deve corrispondere alla relativa voce Service.

In MQSeries per Compaq OpenVMS, la voce authorization service è riportata come impostazione predefinita e il componente associato, OAM, è attivo.

### **Service=service\_name**

Specifica il nome del servizio richiesto. Questo nome deve corrispondere al valore specificato nell'attributo Name della voce Service.

### **Name=component\_name**

Specifica il nome che descrive il componente di servizio. Questo nome non può essere utilizzato altre volte, e deve contenere solo i caratteri validi per i nomi degli oggetti MQSeries (ad esempio, nomi utilizzati per le code). Questo nome viene riportato nei messaggi per l'operatore generati dal servizio. Si raccomanda pertanto, di fare in modo che il nome inizi con il marchio della società o un altro simile elemento di distinzione.

### **Module=module\_name**

Specifica il nome del modulo per contenere il codice di questo componente.

**Nota:** specificare il percorso completo del nome.

## Modifica del file di configurazione MQSeries

### **ComponentDataSize=***size*

Specifica la dimensione, in byte, dell'area dati del componente trasmessa al componente per ogni chiamata. Specificare zero se non sono richiesti dati per il componente.

Per ulteriori informazioni relative ai servizi e ai componenti installabili consultare il manuale *MQSeries Programmable System Management*.

## La voce Log

La voce Log specifica gli attributi log per un particolare Queue Manager. Per impostazione predefinita, questi vengono ereditati dalle impostazioni specificate nella voce LogDefaults del file `mq5.ini` al momento della creazione del Queue Manager, a meno che non vengano sovrascritte da specifici parametri nel comando **crtmqm**. Per ulteriori informazioni, consultare le sezioni "La voce LogDefaults" a pagina 184 e "crtmqm (Crea Queue Manager)" a pagina 258.

Modificare solo gli attributi di questa voce se un particolare Queue Manager necessita di essere configurato in modo diverso rispetto agli altri.

I valori specificati negli attributi del file `qm.ini` vengono letti all'avvio del Queue Manager. Il file viene creato contestualmente alla creazione del Queue Manager.

Per informazioni relative al calcolo delle dimensioni di log consultare la sezione "Calcolo delle dimensioni del log" a pagina 149.

**Nota:** i limiti riportati nel seguente elenco di parametri sono limiti imposti da MQSeries. I limiti del sistema operativo possono ridurre le dimensioni massime di log.

### **LogPrimaryFiles=**3 | 2-62

Primary log files sono i file assegnati durante la creazione per un utilizzo futuro.

Il numero minimo di file di log primari è 2 ed il numero massimo è 62. Il numero predefinito è 3.

Il numero totale di file di log primari e secondari non deve essere superiore a 63 ed inferiore a 3.

Il valore viene esaminato quando si esegue la creazione o l'avvio di un Queue Manager. È possibile modificarlo una volta creato il Queue Manager. Tuttavia, una modifica del valore non sarà attiva prima del successivo riavvio e l'effettività potrebbe non essere immediata.

### **LogSecondaryFiles=**2 | 1-61

Secondary log files sono file di log assegnati dopo l'esaurimento dei file primari.

Il numero minimo di file secondari è 1 ed il numero massimo è 61. Il numero predefinito è 2.

Il numero totale di file di log primari e secondari non deve essere superiore a 63 ed inferiore a 3.

Il valore viene esaminato quando si avvia il Queue Manager. È possibile cambiare questo valore, ma le modifiche non saranno effettive prima del successivo riavvio del Queue Manager ed anche allora gli effetti potrebbero non essere immediati.

## Modifica del file di configurazione MQSeries

### **LogFilePages=number**

I dati di log sono conservati in una serie di file definiti file di log. La dimensione dei file di log è fissata in unità di pagine da 4 KB.

Per MQSeries per Compaq OpenVMS, il numero predefinito di pagine di file di log è 1024, riconoscendo uno spazio al file di log di 4 MB. Il numero minimo è 64 ed quello massimo è 16 384.

**Nota:** la dimensione dei file di log specificati durante la creazione del Queue Manager non può essere modificata successivamente.

### **LogType=CIRCULAR | LINEAR**

L'attributo LogType definisce il tipo di logging che il Queue Manager deve utilizzare. Il tipo di logging da utilizzare non può essere modificato una volta creato il Queue Manager. Fare riferimento alla descrizione dell'attributo LogType nella sezione "La voce LogDefaults" a pagina 184 per informazioni relative alla creazione di Queue Manager con il tipo di logging desiderato.

#### **CIRCULAR**

Impostare questo valore se si desidera eseguire il riavvio del ripristino utilizzando il log per eseguire il roll back delle transazioni che erano in elaborazione nel momento in cui si è verificato l'arresto del sistema.

Consultare la sezione "Registrazione nei log circolare" a pagina 144 per una spiegazione esauriente della registrazione dei log circular.

#### **LINEAR**

Impostare questo valore se si desidera sia il ripristino riavvio (recovery restart) sia il ripristino oggetto (media recovery), creando i dati persi o danneggiati mediante la sostituzione con i contenuti del log.

Consultare la sezione "Registrazione nei log lineare" a pagina 145 per una spiegazione esauriente relativa alla registrazione dei log linear.

### **LogBufferPages=17 | 4-32**

La quantità di memoria assegnata alla registrazione del buffer di scrittura è configurabile. La dimensione dei buffer è prefissata in unità di pagine da 4 kb.

Il numero minimo di pagine buffer è 4 ed il massimo è 32. Maggiori buffer provocano maggiore elaborazione, specialmente per messaggi estesi.

Il numero predefinito di pagine buffer è 17, corrispondente a 68 KB.

Il valore viene esaminato quando il Queue Manager viene avviato e può essere aumentato o diminuito. Tuttavia, una modifica del valore non sarà attiva prima del successivo riavvio.

### **LogPath=directory\_name**

È possibile specificare la directory nella quale risiedono i file di log relativi ad un Queue Manager. La directory dovrebbe risiedere in un dispositivo locale al quale il Queue Manager può accedere e, preferibilmente, dovrebbe trovarsi in un'unità diversa rispetto alle code di messaggio. Specificando un'unità differente aumenta la protezione in caso di malfunzionamento del sistema.

L'impostazione predefinita è MQS\_ROOT: [MQM.LOG].

È possibile specificare il nome di una directory sul comando **crtmqm** utilizzando il flag **-ld**. Quando viene creato un Queue Manager, viene creata anche una directory sotto la directory del Queue Manager, nella quale vengono conservati i file di log. Il nome di questa directory deriva dal nome del Queue Manager. Questo assicura un unico percorso per i file di log ed anche il rispetto delle regole relative alla lunghezza del nome delle directory.



## Modifica del file di configurazione MQSeries

Se non si specifica `-ld` nel comando `crtmqm`, viene utilizzato il valore dell'attributo `LogDefaultPath` nel file `mqs.ini`.

**Nota:** in MQSeries per Compaq OpenVMS, l'ID utente `mqm` e il gruppo `mqm` devono avere pieno accesso ai file di log. Se si spostano questi file, è necessario disporre di questi accessi. Ciò non è necessario se i file di log si trovano nelle posizioni predefinite dal prodotto.

## La voce XAResourceManager

La voce `XAResourceManager` specifica i gestori di risorse da coinvolgere nelle unità globali di elaborazione coordinate dal Queue Manager.

È necessaria una voce `XAResourceManager` in `qm.ini` per ogni richiesta di un gestore risorse incluso nelle unità globali di elaborazione; non vengono forniti valori predefiniti attraverso `mqs.ini`.

Consultare la sezione "Coordinazione dei database" a pagina 128 per ulteriori informazioni relative all'aggiunta di attributi `XAResourceManager` in `qm.ini`.

### **Name=name (mandatory)**

Questo attributo identifica le richieste del gestore risorse.

Il valore `Name` può avere una lunghezza massima di 31 caratteri, e deve essere unico all'interno di `qm.ini`. È possibile utilizzare il nome del gestore risorse come definito nella relativa struttura `XA-switch`. Tuttavia, se si sta utilizzando più di un'istanza dello stesso gestore di risorse, è necessario assegnare un nome specifico ad ogni istanza. A questo scopo è consigliabile includere il nome del database nella stringa `Name`.

MQSeries utilizza il valore `Name` nei messaggi e negli output dal comando `dspmqrn`.

Si raccomanda di non modificare il nome di un'istanza del gestore di risorse e di non cancellare la relativa immissione da `qm.ini` una volta che il Queue Manager associato è stato avviato e il nome del gestore risorse è effettivo.

### **SwitchFile=name (obbligatorio)**

Questo attributo specifica il nome completo del file di caricamento che contiene la struttura `XA switch` del gestore risorse.

### **XAOpenString=string (facoltativo)**

Questo attributo specifica la stringa di dati da trasmettere al punto di immissione `xa_open` del gestore risorse. Il contenuto della stringa dipende dallo stesso gestore di risorse. AD esempio la stringa può identificare il database cui accede una determinata istanza del gestore risorse. Per ulteriori informazioni relative alla definizione di questo attributo, consultare sia la sezione "Aggiunta dell'informazione di configurazione Oracle XAResourceManager" a pagina 134 che la documentazione del gestore risorse per la stringa appropriata.

### **XACloseString=string (facoltativo)**

Questo attributo specifica la stringa di dati da trasmettere al punto di immissione `xa_close` del gestore risorse. Il contenuto della stringa dipende dallo stesso gestore di risorse. Per ulteriori informazioni relative alla definizione di questo attributo, consultare sia la sezione "Aggiunta dell'informazione di configurazione Oracle XAResourceManager" a pagina 134 che la documentazione del database per la stringa appropriata.



## Modifica del file di configurazione MQSeries

### **ThreadOfControl=THREAD | PROCESS**

Il valore impostato nell'attributo ThreadOfControl è utilizzato da Queue Manager per le attività di serializzazione quando necessita di contattare il gestore risorse da uno dei suoi processi multisessione.

#### **THREAD**

Indica che il gestore risorse è completamente "thread aware". In un processo MQSeries multisessione, la funzione chiamata XA può essere seguita verso un gestore risorse esterno da sottoprocessi multipli nello stesso momento.

#### **PROCESS**

Indica che il gestore risorse non è "thread safe". In un processo MQSeries multisessione, può essere eseguita una sola funzione di chiamata XA alla volta verso il gestore risorse.

L'immissione ThreadOfControl non si applica alla funzione di chiamata XA inviata dal Queue Manager in un processo applicativo multisessione. Di solito, un'applicazione che ha unità di elaborazione parallele su differenti sottoprocessi richiede questa modalità operativa per essere supportata da ciascun gestore risorse.

## La voce Channels

La voce Channels contiene informazioni relative ai canali.

### **MaxChannels=100 | *number***

Questo attributo specifica il numero massimo di canali consentito. Il valore predefinito è 100.

### **MaxActiveChannels=MaxChannels\_value**

Questo attributo specifica il numero massimo di canali consentito da poter attivare in qualsiasi momento. Il valore predefinito è quello indicato nell'attributo MaxChannels.

### **MaxInitiators=3 | *number***

Questo attributo specifica il numero massimo di iniziatori.

### **MQIBINDTYPE=FASTPATH | STANDARD**

Questo attributo specifica il binding per le applicazioni.

#### **FASTPATH**

I canali stabiliscono connessioni utilizzando MQCONNX FASTPATH. In questo modo non è necessario un agente di elaborazione.

#### **STANDARD**

I canali stabiliscono connessioni utilizzando STANDARD.

### **AdoptNewMCA=NO | SVR | SDR | RCVR | CLUSRCVR | ALL | FASTPATH**

Se MQSeries riceve la richiesta di avviare un canale, ma rileva un processo amqcrsta in corso su quel canale, è necessario arrestare il processo in corso per poterne avviare un altro. L'attributo AdoptNewMCA consente di controllare l'arresto di un processo in corso e l'avvio di un nuovo processo per un tipo di canale specifico.

Se si specifica l'attributo AdoptNewMCA per un dato tipo di canale, ma l'avvio del nuovo canale ha esito negativo poiché è ancora in esecuzione:

1. Il nuovo canale tenta di arrestare il precedente inviando una richiesta di arresto.

## Modifica del file di configurazione MQSeries

2. Se il canale precedente non risponde a questa richiesta nel tempo concessogli da `AdoptNewMCATimeout`, il processo o il sottoprocesso di quel canale viene terminato.
3. Se il canale precedente non si arresta dopo l'operazione descritta nella fase 2 e dopo che `AdoptNewMCATimeout` attende un secondo intervallo di tempo, MQSeries arresta il canale con un errore "CHANNEL IN USE".

Specificare uno o più valori tra quelli del seguente elenco, separandoli con virgole o spazi vuoti:

**NO** La funzione `AdoptNewMCA` non è richiesta. Questo è il valore predefinito.

**SVR** Adotta canali server

**SDR** Adotta canali di invio

**RCVR** Adotta canali di ricezione

**CLUSRCVR**

Adotta canali di ricezione cluster

**ALL** Adotta qualunque tipo di canale, eccetto i canali FASTPATH

**FASTPATH**

Adotta il canale FASTPATH. Ciò è possibile solo se è selezionato il tipo di canale appropriato, ad esempio,

`AdoptNewMCA=RCVR,SVR,FASTPATH`

### Attenzione!

L'attributo `AdoptNewMCA` può avere effetti imprevedibili con i canali FASTPATH a causa della progettazione interna del Queue Manager. Quindi prestare molta attenzione nell'abilitare l'attributo `AdoptNewMCA` per i canali FASTPATH.

### `AdoptNewMCATimeout=60 | 1—3600`

Questo attributo specifica la quantità di tempo (in secondi) necessaria per l'arresto del precedente processo. Specificare un valore (in secondi) compreso tra 1—3600. Il valore predefinito è 60.

### `AdoptNewMCACheck=QM | ADDRESS | NAME | ALL`

L'attributo `AdoptNewMCACheck` consente di specificare il tipo di controllo richiesto durante l'abilitazione dell'attributo `AdoptNewMCA`. È importante effettuare tutti i controlli seguenti, per evitare che i canali vengano arrestati. Se non è possibile effettuarli tutti, controllare almeno che i nomi corrispondano.

## Modifica del file di configurazione MQSeries

Specificare uno o più valori tra quelli del seguente elenco, separandoli con virgole o spazi vuoti:

**QM** Questo indica che il processo listener deve controllare che i nomi del Queue Manager corrispondano.

### ADDRESS

Questo indica che il processo listener deve controllare gli indirizzi delle comunicazioni come, ad esempio, l'indirizzo TCP/IP.

### NAME

Questo indica che il processo listener deve controllare che i nomi dei canali corrispondano.

**ALL** Specificando questo valore si richiedono tutti i controlli sopra riportati.

AdoptNewMCACheck=NAME,ADDRESS è il valore predefinito per FAP1, FAP2 e FAP3, mentre AdoptNewMCACheck=NAME,ADDRESS,QM è il valore predefinito per FAP4 e successivi.

## Le voci LU62 e TCP

Queste voci specificano i parametri di configurazione del protocollo di rete. Esse sovrascrivono gli attributi predefiniti per i canali.

**Nota:** occorre specificare solo gli attributi che costituiscono modifiche dei valori predefiniti.

### LU62

Possono essere specificati i seguenti attributi:

#### TPName

Questo attributo specifica il nome TP da avviare su un sito in remoto.

#### LocalLU

Questo è il nome dell'unità da utilizzare sui sistemi locali.

### TCP

Possono essere specificati i seguenti attributi:

#### Port=1414 | *port\_number*

Questo attributo specifica il numero di porta predefinito (in decimali) per le sessioni TCP/IP. Il numero di porta noto per MQSeries è 1414.

#### KeepAlive=YES | NO

Utilizzare questo attributo per attivare o disattivare la funzione KeepAlive. L'impostazione KeepAlive=YES fa sì che TCP/IP controlli periodicamente la disponibilità dell'altra terminazione della connessione. Se non è disponibile, il canale è chiuso.

#### ListenerBacklog=number

Durante la ricezione TCP/IP, viene impostato un numero massimo di richieste di connessioni esterne. Questo potrebbe essere considerato un *backlog* di richieste in attesa sulla porta TCP/IP per il listener che deve accettare tali richieste. I valori del backlog per il listener sono illustrati in Tabella 8.

Tabella 8. Richieste predefinite di connessione esterna (TCP)

Piattaforma	Valore predefinito ListenerBacklog
OS/390	255
OS/2 Warp	10
Windows NT Server	100

## Modifica del file di configurazione MQSeries

Tabella 8. Richieste predefinite di connessione esterna (TCP) (Continua)

Piattaforma	Valore predefinito ListenerBacklog
Windows NT Workstation	5
AS/400	255
Sun Solaris	100
HP-UX	20
AIX V4.2 o successive	100
AIX V4.1 o precedenti	10
Tutte le altre piattaforme	5

Se il backlog raggiunge i valori mostrati in Tabella 8 a pagina 193, la connessione TCP/IP è rifiutata e il canale non potrà essere avviato.

I canali MCA, in questo caso, entrano nello stato RETRY e il tentativo di connessione viene riprovato dopo poco tempo.

Per le connessioni client, il client riceve un codice di errore MQRC\_Q\_MGR\_NOT\_AVAILABLE da MQCONN e ritenta la connessione dopo poco tempo.

L'attributo ListenerBacklog consente di sovrascrivere il numero predefinito di richieste in corso per il listener TCP/IP.

**Nota:** alcuni sistemi operativi supportano un valore maggiore rispetto a quello predefinito. Se necessario, questo può essere utilizzato per aumentare i limiti della connessione.

## La voce ExitPath

**ExitDefaultPath**=string

L'attributo ExitDefaultPath specifica la posizione di:

- Uscite del canale per client
- Uscite del canale e uscite della conversione dei dati per server

Il percorso di uscita è analizzato dalla voce ClientExitPath nel file `mqs.ini` per client e per server.

---

## Esempio di file `mqs.ini` e `qm.ini`

La Figura 19 a pagina 195 mostra un esempio di un file `mqs.ini` MQSeries per Compaq OpenVMS.

## Modifica del file di configurazione MQSeries

```
#####  
** Module Name: mqs.ini                                **  
** Type      : MQSeries Configuration File            **  
** Function   : Define MQSeries resources for the node **  
**                                                    **  
#####  
** Notes      :                                       **  
** 1) This is an example MQSeries configuration file   **  
**                                                    **  
#####  
AllQueueManagers:  
#####  
** The path to the qmgrs directory, below which queue manager data **  
** is stored                                                    **  
#####  
DefaultPrefix=mqs_root:[mqm]  
  
ClientExitPath:  
  ExitsDefaultPath=mqs_root:[mqm.exits]  
  
LogDefaults:  
  LogPrimaryFiles=3  
  LogSecondaryFiles=2  
  LogFilePages=1024  
  LogType=CIRCULAR  
  LogBufferPages=17  
  LogDefaultPath=mqs_root:[mqm.log]  
QueueManager:  
  Name=saturn.queue.manager  
  Prefix=mqs_root:[mqm]  
  Directory=saturn$queue$manager  
DefaultQueueManager:  
  Name=saturn.queue.manager  
QueueManager:  
  Name=pluto.queue.manager  
  Prefix=mqs_root:[mqm]  
  Directory=pluto$queue$manager
```

Figura 19. Esempio di un file di configurazione MQSeries per sistemi MQSeries per Compaq OpenVMS

La Figura 20 a pagina 196 mostra come poter disporre gruppi di attributi in un file di configurazione di Queue Manager in MQSeries per Compaq OpenVMS.

## Modifica del file di configurazione MQSeries

```
#####  
#* Module Name: qm.ini                                     *#  
#* Type       : MQSeries queue manager configuration file *#  
# Function    : Define the configuration of a single queue manager *#  
#*                                                  *#  
#####  
#* Notes      :                                           *#  
#* 1) This file defines the configuration of the queue manager *#  
#*                                                  *#  
#####  
ExitPath:  
  ExitsDefaultPath=mqm_root:[mqm.exits]  
  
Service:  
  Name=AuthorizationService  
  EntryPoints=9  
  
ServiceComponent:  
  Service=AuthorizationService  
  Name=MQSeries.UNIX.auth.service  
  Module=amqzfu  
  ComponentDataSize=0  
  
Service:  
  Name=NameService  
  EntryPoints=5  
  
ServiceComponent:  
  Service=NameService  
  Name=MQSeries.DCE.name.service  
  Module=amqzfa  
  ComponentDataSize=0  
  
Log:  
  LogPrimaryFiles=3  
  LogSecondaryFiles=2  
  LogFilePages=1024  
  LogType=CIRCULAR  
  LogBufferPages=17  
  LogPath=mqm_root:[mqm.log.saturn$queue$manager]  
  
XAResourceManager:  
  Name=Oracle Resource Manager Bank  
  SwitchFile=sys$share:oraswit0.exe  
  XAOpenString=MQBankDB  
  XACloseString=  
  ThreadOfControl=PROCESS  
  
CHANNELS:  
  MaxChannels = 20           ; Maximum number of Channels allowed.  
                             ; Default is 100.  
  MaxActiveChannels = 10    ; Maximum number of Channels allowed to be  
                             ; active at any time. The default is the  
                             ; value of MaxChannels.  
  
TCP:  
  KeepAlive = Yes           ; TCP/IP entries.  
                             ; Switch KeepAlive on
```

Figura 20. Esempio di file di configurazione di Queue Manager per MQSeries per Compaq OpenVMS

### Note:

MQSeries sul nodo utilizza le posizioni predefinite per Queue Manager e per i log.

## Modifica del file di configurazione MQSeries

Il Queue Manager saturn.queue.manager quello predefinito per il nodo. Il nome della directory che comprende i file associati a questo Queue Manager viene automaticamente trasformato in un nome di file compatibile con OpenVMS file system.

Dal momento che il file di configurazione MQSeries viene utilizzato per localizzare i dati associati ai Queue manager, un file di configurazione inesistente o non valido provocherebbe il mancato funzionamento di alcuni o tutti i comandi MQSeries. Inoltre, le applicazioni non potrebbero connettersi ad un Queue Manager non definito in un file di configurazione MQSeries.





---

## Capitolo 14. Determinazione dei problemi

Questo capitolo offre alcuni suggerimenti per la gestione dei problemi che possono verificarsi utilizzando MQSeries per Compaq OpenVMS.

Non tutti i problemi possono risolversi immediatamente; vi sono problemi, infatti, problemi che possono dipendere da limitazioni hardware. Se si ha ragione di credere che il problema dipenda dal codice MQSeries, contattare il centro di supporto della IBM®. Questo capitolo si compone delle seguenti sezioni:

- “Controlli preliminari”
- “Errori di programmazione frequenti” a pagina 202
- “Come procedere” a pagina 203
- “Considerazioni sulla progettazione delle applicazioni” a pagina 205
- “Output non corretto” a pagina 207
- “Log di errore” a pagina 209
- “Code messaggi non recapitati” a pagina 213
- “File di configurazione e determinazione dei problemi” a pagina 214
- “Utilizzo della traccia MQSeries” a pagina 214
- “FFST (First Failure Support Technology)” a pagina 215
- “Determinazione dei problemi relativi ai client” a pagina 220

---

### Controlli preliminari

Solitamente i problemi relativi a MQSeries si verificano durante l'utilizzo dei seguenti componenti:

- MQSeries
- La rete
- L'applicazione
- Il sistema operativo di base

Le sezioni seguenti suggeriscono alcuni controlli utili alla determinazione del problema.

### Si è mai avuta in precedenza un'esecuzione con esito positivo di MQSeries?

Se non si è mai avuta un'esecuzione con esito positivo di MQSeries, il problema potrebbe dipendere da un'errata impostazione. Consultare il manuale *MQSeries per Compaq OpenVMS Alpha, Quick Beginnings, Version 5.1* per verificare la corretta installazione e impostazione di MQSeries.

### Vengono riportati messaggi di errore?

MQSeries utilizza log di errori per catturare messaggi relativi alle operazioni MQSeries, ai Queue Manager avviati e dati di errori provenienti dai canali utilizzati. Verificare l'eventuale registrazioni di log di errori associati al problema.

Consultare la sezione “Log di errore” a pagina 209 per informazioni relative ai contenuti dei log di errore e alle loro posizioni.

### **Vengono riportati codici di errori che spiegano il problema?**

Se un'applicazione riceve un codice di errore in cui è indicato che una chiamata MQI (Message Queue Interface) ha avuto un esito negativo, consultare il manuale *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa* per una descrizione di questo codice di errore.

### **Il problema è riproducibile?**

Se è possibile riprodurre il problema, fare attenzione alle condizioni in cui viene generato:

- È determinato da un comando o un'altra attività di gestione?  
L'operazione funziona se immessa utilizzando un altro metodo? Se il comando funziona solo immettendolo nella linea di comando e non in altro modo, verificare che il server di comando non sia stato arrestato e che non sia stata modificata la definizione di coda del SYSTEM.ADMIN.COMMAND.QUEUE.
- È determinato da un programma? Si verifica su tutti i sistemi MQSeries e su tutti i Queue Manager o solo su alcuni?
- È possibile individuare un'applicazione che sembra in esecuzione tutte le volte che si verifica il problema? In questo caso, esaminare l'applicazione per verificare la presenza di eventuali errori.

### **Sono state effettuate modifiche dall'ultima esecuzione valida?**

Nel riesaminare le eventuali modifiche apportate di recente, considerare il sistema MQSeries, i programmi con cui interagisce, l'hardware e le nuove applicazioni. Considerare inoltre che una nuova applicazione sconosciuta potrebbe essere stata eseguita sul sistema.

- Sono state aggiunte, modificate o eliminate definizioni di code?
- Sono stati aggiunte o modificate definizioni di canali? Potrebbero essere state effettuate modifiche alle definizioni di canali MQSeries o alle definizioni di comunicazioni di base richieste dal sistema.
- Le applicazioni sono occupate a gestire i codici di errore ricevuti in seguito a qualche modifica effettuata?

### **Si è mai avuta in precedenza un'esecuzione con esito positivo dell'applicazione?**

Se il problema sembra relativo ad una specifica applicazione, controllare che l'applicazione abbia avuto almeno per una volta esito positivo nell'esecuzione.

Prima di rispondere **Sì** a questo interrogativo, prendere in esame le seguenti considerazioni?:

- Sono state apportate modifiche all'applicazione dopo l'ultima esecuzione valida?  
In questo caso, è probabile che l'errore dipenda dalle modifiche dell'applicazione. Controllare le modifiche e verificare l'eventuale presenza di motivazioni ovvie del problema. Provare ad utilizzare l'applicazione nello stato precedente la modifica.
- Sono state utilizzate tutte le funzioni dell'applicazione in precedenza?  
Il problema si verifica quando viene utilizzata una parte dell'applicazione mai impiegata in precedenza? In questo caso, è probabile che l'errore risieda in quella parte dell'applicazione. Individuare l'operazione compiuta dall'applicazione al momento del malfunzionamento e controllare in quella parte del programma il codice sorgente relativo agli errori.

Se il programma non ha mai determinato problemi durante le precedenti esecuzioni, controllare lo stato della coda e dei file elaborati al momento del verificarsi dell'errore. Può accadere che essi contengano dei valori inconsueti che provocano il richiamo di un percorso raramente utilizzato nel programma.

- L'applicazione controlla tutti i codici di errore?

È possibile che il sistema MQSeries sia stato modificato, magari lievemente, ma in modo che l'applicazione non controlli più i codici di errore che riceve. Ad esempio, l'applicazione è in grado di desumere che le code cui accede sono condivise? Se una coda è stata ridefinita esclusiva, l'applicazione elabora il codice che le indica l'impossibilità di accedere a quella coda?

- L'applicazione viene eseguita su altri sistemi MQSeries?

Il problema potrebbe dipendere dal tipo di impostazioni del sistema MQSeries? Ad esempio, sono state definite le stesse lunghezze e priorità di messaggi per le code?

### **Se l'applicazione in precedenza non è stata eseguita con esito positivo**

Se l'applicazione non ha mai avuto un'esecuzione valida, è necessario ricercare con cura la presenza di eventuali problemi.

Prima di controllare il codice e il linguaggio di programmazione in cui è stato scritto, esaminare l'output di traduzione, di compilazione e del redattore di collegamento, laddove possibile, per controllare se vengono riportati errori.

Se il malfunzionamento si verifica quando l'applicazione tenta di tradurre, compilare o redigere collegamenti nella libreria di caricamento, si verificherà di conseguenza anche durante i tentativi di esecuzione. Consultare il manuale *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa* per informazioni relative alla realizzazione dell'applicazione.

Se la documentazione dimostra che tutte queste operazioni sono state eseguite senza errori, il problema potrebbe dipendere dalla codificazione logica dell'applicazione. I sintomi del problema indicano la funzione in cui si verifica il malfunzionamento e, di conseguenza, il segmento di codice in errore? Consultare la sezione "Errori di programmazione frequenti" a pagina 202 per alcuni esempi di errori frequenti che provocano problemi con le applicazioni MQSeries.

### **Il problema è relativo a parti specifiche della rete?**

Il problema potrebbe verificarsi solo in alcune parti della rete (code remote, ad esempio). Se il collegamento ad un Message Queue Manager remoto non è attivo, i messaggi non possono giungere ad una coda remota.

Verificare la disponibilità del collegamento tra i due sistemi e controllare che il componente di intercomunicazione di MQSeries sia in esecuzione.

Controllare che i messaggi ricevano la coda di trasmissione; controllare inoltre la definizione di coda locale della coda di trasmissione e ogni coda remota.

Sono state effettuate modifiche relative alla rete o nelle definizioni MQSeries tali da poter provocare problemi?

### **Il problema si verifica ad una determinata ora del giorno?**

Se il problema si verifica ad una determinata ora del giorno potrebbe dipendere dall'eccessivo caricamento del sistema. Solitamente, il sistema riceve il massimo

## Controlli preliminari

carico nelle fasce orarie di metà mattina e metà pomeriggio: è in questi orari che si verifica la maggior parte dei problemi dipendenti dal sovraccarico del sistema. Se la rete MQSeries è in funzione anche in altre fasce orarie, momenti di intenso carico del sistema potrebbero verificarsi anche in altri periodi del giorno.

### Il problema si manifesta in modo discontinuo?

Un problema discontinuo può verificarsi se non si considera che i processi possono essere eseguiti in modo autonomo l'uno dall'altro. Ad esempio, un programma può inviare una chiamata MQGET, senza specificare un'opzione di attesa, prima che sia terminato un processo precedente. Un problema discontinuo può verificarsi anche nel caso in cui l'applicazione tenti di ricevere un messaggio da una coda, mentre la chiamata che invia il messaggio è sospesa (e ciò si verifica prima dell'esecuzione del commit o del back out).

### Sono stati eseguiti aggiornamenti del servizio?

Se sono stati eseguiti aggiornamenti del servizio ad MQSeries, controllare che le operazioni di aggiornamento siano completate con esito positivo e che non vengano riportati messaggi di errore.

- Esistono particolari istruzioni per l'aggiornamento?
- Sono state eseguite delle prove per verificare l'applicazione corretta e completa dell'aggiornamento?
- Il problema si verifica anche se si ripristina il precedente livello di servizio MQSeries?
- Nel caso di installazione con esito positivo, rivolgersi al Centro supporto della IBM per verificare l'esistenza di eventuali errori patch.
- Nel caso in cui venga applicato un patch ad un altro programma, considerare l'effetto che può avere sul modo in cui MQSeries interagisce con esso.

### È necessario aggiungere un aggiornamento?

Il funzionamento di MQSeries è dipendente dal sistema operativo di base (OpenVMS) e da vari prodotti per la comunicazione dei dati, tra i quali TCP/IP. Consultare il proprio negoziante per accertarsi di aver installato tutti gli aggiornamenti di servizio necessari per questi prodotti.

---

## Errori di programmazione frequenti

Gli errori riportati nel seguente elenco mostrano le cause più comuni dei problemi che si verificano durante l'esecuzione di programmi MQSeries. Il problema verificatosi al sistema MQSeries potrebbe essere dovuto ad uno o più di questi errori:

- Considerare condivisibili, code che invece sono state assegnate come esclusive.
- Utilizzare parametri non corretti in una chiamata MQI.
- Utilizzare parametri insufficienti in una chiamata MQI. Ciò comporta che MQI non può impostare il completamento ed i codici di errori relativi all'applicazione da elaborare.
- Errato controllo dei codici richiesti da MQI.
- Utilizzo di variabili di lunghezza non valida.
- Utilizzo di parametri in ordine errato.
- Inizializzazione errata di *MsgId* e *CorrelId*.

## Come procedere

Probabilmente i controlli preliminari hanno consentito di individuare la causa del problema. In questo caso, è ora possibile risolverli, magari con l'aiuto di altri manuali della libreria MQSeries (consultare la Bibliografia) e delle librerie di altri programmi su licenza.

Se invece non si è ancora riusciti ad individuare la causa, è necessario compiere un esame più specifico del problema.

Lo scopo della presente sezione è quello di consentire all'utente di individuare la causa del problema, laddove non vi sia riuscito eseguendo i controlli preliminari.

Una volta stabilito che non sono state apportate modifiche al sistema e che non vi sono problemi relativi ai programmi applicativi, esaminare le seguenti opzioni e scegliere quella che meglio descrive i sintomi del problema.

- "Viene riportato un output non corretto?"
- "Non si è ottenuta alcuna risposta utilizzando un comando PCF?"
- "Il problema è relativo alle sole code remote?" a pagina 205

Se nessuna delle seguenti opzioni è in grado di descrivere i sintomi del problema, considerare l'ipotesi che il problema sia dovuto ad un altro componente del sistema.

### Viene riportato un output non corretto?

In questa pubblicazione, "output non corretto" si riferisce all'applicazione:

- Non viene riportato il messaggio atteso.
- Viene riportato un messaggio che contiene informazioni inattese o errate.
- Viene riportato un messaggio inatteso, come ad esempio un messaggio destinato ad un'applicazione diversa.

In ogni caso, controllare che ogni coda o gli alias Queue Manager utilizzati dall'applicazione siano correttamente specificati e rivedere tutte le modifiche apportate alla rete.

Se viene riportato un messaggio di errore MQSeries, controllare nel log di errore tutto ciò che ha come prefisso le lettere "AMQ". Per ulteriori informazioni, consultare la sezione "Log di errore" a pagina 209.

### Non si è ottenuta alcuna risposta utilizzando un comando PCF?

Se, impostando un comando, non si riceve alcuna risposta, esaminare le seguenti considerazioni:

- Il server di comando è in esecuzione?
  - Utilizzare il comando **dspmqcsv** per verificare lo stato del server di comando.
  - Se la risposta a questo comando segnala che il server non è in esecuzione, utilizzare il comando **strmqcsv** per avviarlo.
  - Se invece la risposta segnala che SYSTEM.ADMIN.COMMAND.QUEUE non è abilitato alle richieste MQGET, abilitare la coda alle richieste MQGET.
- È stata inviata una risposta alla coda messaggi non recapitati?

La struttura di intestazione della coda messaggi non recapitati contiene un codice di errore o di feedback che descrive il problema. Consultare il manuale

## Come procedere

*MQSeries per Tandem Non Stop Kernal, V5.1 Guida operativa* per informazioni relative alla struttura di intestazione della coda messaggi non recapitati (MQDLH).

Se questa coda contiene messaggi, utilizzare l'applicazione di esempio browse (amqsbcg) fornita per sfogliare i messaggi utilizzando la chiamata MQGET. L'applicazione di esempio guida attraverso tutti i messaggi di una data coda relativa ad un dato Queue Manager, visualizzando sia la descrizione, che i campi relativi al contesto di tutti i messaggi di una data coda.

- È stato inviato un messaggio al log di errore.  
Per ulteriori informazioni, consultare la sezione "Log di errore" a pagina 209.
- Le code sono abilitate alle operazioni di inserimento ed estrazione?
- L'intervallo di tempo (*WaitInterval*) è abbastanza lungo?  
Se una coda MQGET è scaduta, vengono riportati un codice di completamento di MQCC\_FAILED e un codice di errore di MQRC\_NO\_MSG\_AVAILABLE. Consultare il manuale *MQSeries per Tandem Non Stop Kernal, V5.1 Guida operativa* per informazioni relative campo *WaitInterval* e ai codici di completamento e di errore di MQGET.
- È necessario utilizzare un syncpoint quando si utilizza il programma applicativo per inserire comandi nel SYSTEM.ADMIN.COMMAND.QUEUE?  
Fin quando non si specifica l'esclusione dal syncpoint del messaggio di richiesta, è necessario utilizzare un syncpoint prima di ricevere i messaggi di riposta.
- L'impostazione degli attributi delle code MAXDEPTH e MAXMSGL è abbastanza alta?
- I campi *CorrelId* e *MsgId* vengono utilizzati correttamente?  
Impostare i valori di *MsgId* e *CorrelId* nell'applicazione in modo da poter ricevere tutti i messaggi dalla coda.

Per rispondere a qualsiasi messaggio di errore riprodotto, provare ad arrestare e poi riavviare il server di comando.

Se il sistema ancora non risponde, probabilmente il problema riguarda un Queue Manager o l'intero sistema MQSeries. Provare prima ad arrestare ogni singolo Queue Manager per individuare quello in cui si verifica il malfunzionamento. Se in questo modo non si riesce ad individuare la causa del problema, provare ad arrestare e poi riavviare MQSeries, in risposta ad un messaggio prodotto nel log di errore.

Se all'avvio il problema si verifica ancora, richiedere l'assistenza del centro di supporto IBM.

## Si verificano malfunzionamenti in alcune code?

Se si ha motivo di ritenere che il problema riguardi solo un gruppo secondario di code, controllare la coda locale sospetta causa del problema:

1. Visualizzare le informazioni relative a ciascuna coda. A questo scopo, utilizzare il comando MQSC.
2. Utilizzare i dati visualizzati per eseguire i seguenti controlli:
  - Se CURDEPTH è a MAXDEPTH, indica che la coda non viene elaborata. Controllare che tutte le applicazioni vengano eseguite normalmente.
  - Se CURDEPTH non è a MAXDEPTH, controllare che i seguenti attributi di coda siano impostati correttamente:
    - Se viene utilizzata la funzione triggering:

- Il trigger monitor è in esecuzione?
- L'intensità di trigger è troppo elevata? In questo caso vengono generati eventi trigger troppo spesso?
- Il nome del processo è corretto?
- Il processo è disponibile e operativo?
- La coda può essere condivisa? Se non può essere condivisa, un'altra applicazione potrebbe già averla impegnata per l'input.
- La coda è abilitata correttamente per l'inserimento e l'estrazione?
- Se non vi sono processi applicativi che ricevono messaggi dalla coda, occorre determinarne il motivo. Ciò potrebbe essere dovuto al fatto che l'applicazione deve essere avviata o al fatto che una connessione è stata eliminata o ancora al fatto che per motivi vari la chiamata MQOPEN ha avuto esito negativo.

Controllare gli attributi di coda IPPROCS e OPPROCS. Questi attributi indicano se la coda è stata aperta per l'input e l'output. Un valore pari a zero indica che tale operazione non è possibile. Si noti che i valori possono essere cambiati e perciò la coda può essere stata aperta e poi dopo chiusa.

È necessario esaminare lo stato della coda al momento in cui si prevede di inserire o estrarre un messaggio.

Se non si riesce a risolvere il problema, richiedere l'assistenza al centro di supporto IBM.

## Il problema è relativo alle sole code remote?

Se il problema è relativo alle sole code remote, effettuare i seguenti controlli:

- Controllare che i canali richiesti siano avviati, che possono essere sottoposti al trigger e che siano in esecuzione tutti gli iniziatori richiesti.
- Controllare che i programmi che dovrebbero inserire messaggi nelle code remote non abbiano riportato problemi.
- Se si utilizza la funzione triggering per avviare il processo di accodamento distribuito, controllare che la coda di trasmissione è stata impostata per il triggering. Controllare, inoltre, che il channel initiator sia in esecuzione.
- Controllare l'eventuale presenza nei log degli errori di messaggi che indicano errori o problemi relativi al canale.
- Se necessario avviare il canale manualmente. Per informazioni relative a quest'operazione, consultare il manuale *MQSeries Intercommunication*.

Consultare il manuale *MQSeries Intercommunication* per informazioni relative alla definizione dei canali.

---

## Considerazioni sulla progettazione delle applicazioni

Sono numerose i casi in cui le prestazioni di un programma sono influenzate dai limiti della progettazione. Alle volte è difficile rilevare queste limitazioni, poiché il programma sembra avere un normale funzionamento, mentre invece danneggia le prestazioni di altre attività. Di seguito sono riportate alcuni dei problemi propri dei programmi che eseguono chiamate MQSeries.

Per ulteriori informazioni relative alla progettazione delle applicazioni, consultare il manuale *MQSeries Application Programming Guide*.



## Considerazioni sulla progettazione delle applicazioni

### Lunghezza del messaggio

Sebbene MQSeries supporti messaggi che occupano fino a 100MB di dati, una notevole quantità di dati in un messaggio influisce sull'applicazione che lo elabora. Per ottenere le migliori prestazioni dall'applicazione, è opportuno inviare solo i dati essenziali in un messaggio; ad esempio, nella richiesta di addebito in conto corrente bancario, le uniche informazioni che dovrebbero essere inviate dal client all'applicazione server sono il numero del conto corrente e l'importo dell'addebito.

### Messaggi permanenti

I messaggi permanenti vengono collegati. I messaggi collegati riducono le prestazioni dell'applicazione; per questo motivo è consigliabile utilizzare i messaggi permanenti solo per i dati essenziali. Se i dati in un messaggio possono essere cancellati in caso di arresto o malfunzionamento di un Queue Manager, utilizzare messaggi non permanenti.

### Ricerca di un particolare messaggio

La chiamata MQGET generalmente richiama il primo messaggio da una coda. Se si utilizzano gli identificativi di messaggio e di correlazione (*MsgId* e *CorrelId*) nel descrittore di messaggio per specificare un particolare messaggio, il Queue Manager deve iniziare una ricerca che termina solo quando rileva quel messaggio. Questo tipo di utilizzo della chiamata MQGET influisce sulle prestazioni dell'applicazione.

### Code contenenti messaggi di diversa lunghezza

Se i messaggi in una coda hanno lunghezze diverse, per determinare la lunghezza di un messaggio l'applicazione potrebbe utilizzare la chiamata MQGET con il campo *BufferLength* impostato su zero, in modo che, se la chiamata dovesse avere esito negativo, verrebbe riportata la dimensione dei dati del messaggio. L'applicazione potrebbe allora ripetere la chiamata, specificando l'identificativo del messaggio misurato nella precedente chiamata e un buffer di dimensioni corrette. Tuttavia se vi sono altre applicazioni relative alla stessa coda, le prestazioni dell'applicazione saranno ridotte poiché la seconda chiamata MQGET impiegherà del tempo nella ricerca del messaggio che un'altra applicazione avrà richiamato nello spazio di tempo fra le due chiamate.

Se l'applicazione non può utilizzare messaggi di lunghezza fissa, un'altra soluzione potrebbe essere quella di utilizzare la chiamata MQINQ per trovare la dimensione massima di messaggi accettabile dalla coda, per poi utilizzare questo valore nella chiamata MQGET. La dimensione massima dei messaggi è memorizzata nell'attributo *MaxMsgLength* della coda. Tuttavia, questo metodo utilizza una gran quantità di memoria, poiché il valore di questo attributo può giungere fino a 100 MB, il massimo supportato da MQSeries per Compaq OpenVMS.

### Frequenza di syncpoint

I programmi che inviano numerose chiamate MQPUT in un syncpoint, senza eseguirne il commit, possono causare problemi di prestazioni. Le code interessate possono essere riempite di questi messaggi attualmente inaccessibili, mentre altre attività sono in attesa di ricevere tali messaggi. Questo ha implicazioni in termini di memoria e in termini di vincolo di sottoprocessi ad attività che tentano di ricevere messaggi.



### Utilizzo della chiamata MQPUT1

Utilizzare la chiamata MQPUT1 solo quando si deve inviare un singolo messaggio ad una coda. Se invece si devono inviare più messaggi, utilizzare la chiamata MQOPEN, seguite da una serie di chiamate MQPUT e una singola MQCLOSE.

---

### Output non corretto

Il termine “output non corretto” può avere diverse interpretazioni. L’accezione utilizzata in questo capitolo del manuale è quella riportata nella sezione “Viene riportato un output non corretto?” a pagina 203.

In questa sezione vengono illustrati due tipi di output non corretti:

- Il caso in cui non si trovano i messaggi attesi
- Il caso di messaggi che contengono informazioni errate o alterate

Inoltre vengono riportati anche ulteriori problemi relativi alle applicazioni tra cui l’utilizzo di code di distribuzione.

### Messaggi assenti nella coda

Se non si trovano i messaggi attesi nella coda, effettuare i seguenti controlli:

- Il messaggio è stato inviato con esito positivo alla coda?
  - La coda è stata definita in modo corretto? Ad esempio, MAXMSGL è abbastanza esteso?
  - La coda è abilitata all’invio?
  - La coda è già piena? Questo potrebbe rendere impossibile ad un’applicazione l’invio di un messaggio alla coda.
- È possibile ricevere messaggi dalla coda?
  - È necessario fissare un syncpoint?

Se sono stati inviati o richiamati messaggi all’interno di un syncpoint, non saranno disponibili per le altre operazioni finché non sarà eseguito il commit dell’unità di ripristino.
  - L’intervallo di attesa è abbastanza lungo?

È possibile impostare come opzione per la chiamata MQGET l’intervallo di attesa. Occorre assicurarsi che il tempo di attesa sia lungo abbastanza per ottenere una risposta.
  - Si è in attesa di un messaggio particolare, determinato da un identificativo di messaggio o di correlazione (*MsgId* o *CorrelId*)?

Controllare che gli identificativi *MsgId* o *CorrelId* siano corretti per il messaggio. Una chiamata MQGET con esito positivo imposta entrambi questi valori in base alla risposta del messaggio; per questo potrebbe essere necessario reimpostare questi valori per poter ricevere altri messaggi.

Controllare, inoltre, se è possibile ricevere altri messaggi dalla coda.
  - Le altre applicazioni ricevono messaggi dalla coda?
  - Il messaggio atteso era stato definito permanente?

In caso contrario, se MQSeries viene riavviato, il messaggio è perso.
  - Esiste un’altra applicazione che ha accesso esclusivo alla coda?

Se si riesce ad individuare un problema relativo alla coda durante il funzionamento di MQSeries, effettuare i seguenti controlli sul processo che deve inviare il messaggio alla coda:

- L’applicazione è stata avviata?

## Output non corretto

Nel caso in cui venga sottoposta a trigger, controllare che siano state specificate le corrette opzioni.

- L'applicazione è stata arrestata?
- È in esecuzione un trigger monitor?
- Il processo trigger è stato definito correttamente?
- L'applicazione termina correttamente?

Verificare la presenza di una vistosa anomalia della chiusura nel log di lavoro.

- L'applicazione esegue il commit delle proprie modifiche o queste vengono ritirate (backed out)?

Se esistono transazioni multiple relative alla coda, esse possono trovarsi in conflitto tra loro. Si supponga, ad esempio, che una transazione invii una chiamata MQGET con un buffer di lunghezza zero per individuare la lunghezza del messaggio e poi invii una chiamata MQGET che specifica il *MsgId* di quel messaggio. Nello stesso tempo, un'altra transazione invia una chiamata MQGET relativa a quel messaggio che ha esito positivo; in questo caso la prima applicazione riceverà il codice di errore MQRC\_NO\_MSG\_AVAILABLE. Le applicazioni che si desidera eseguire in ambiente multi-server devono essere progettate per adattarsi a queste situazioni.

Si noti che il messaggio può essere stato ricevuto, anche se l'applicazione per qualche motivo non riesce ad elaborarlo. Ad esempio, un errore nel formato del messaggio atteso potrebbe determinare il rifiuto da parte del programma? In questo caso, consultare la sezione "Messaggi che contengono informazioni inattese o alterate".

## Messaggi che contengono informazioni inattese o alterate

Se l'informazione contenuta nel messaggio non è ciò che l'applicazione attendeva o è stata alterata in qualche modo, prendere in esame le seguenti considerazioni:

- È stata modificata l'applicazione utilizzata o quella che invia il messaggio alla coda?

Assicurarsi che ogni modifica sia stata comunicata a tutti i sistemi che devono venire a conoscenza del cambiamento.

Ad esempio, se viene modificato il formato dei dati del messaggio, entrambe le applicazioni devono essere ricomilate per integrare le modifiche. Se un'applicazione non viene ricompilata, all'altra i dati sembreranno alterati.

- L'applicazione sta inviando messaggi alla coda sbagliata?

Controllare che i messaggi ricevuti dall'applicazione non siano in realtà destinati ad un'applicazione relativa ad una diversa coda. Se necessario, modificare le definizioni di sicurezza, in modo da impedire alle applicazioni non autorizzate di inviare messaggi alle code.

Se l'applicazione utilizza un alias di coda, controllare che l'alias si riferisca alla coda appropriata.

- Le informazioni del trigger relative a questa coda sono state specificate correttamente?

Controllare che l'applicazione da utilizzare sia stata avviata; è stata avviata un'applicazione diversa?

Se questi controlli non consentono di risolvere il problema, potrebbe essere necessario controllare la logica dell'applicazione, sia per quanto riguarda il programma che invia il messaggio, che per quello che lo riceve.

## Problemi relativi ad output non corretti durante l'accodamento distribuito

Se l'applicazione utilizza l'accodamento distribuito, è consigliabile prendere in esame i seguenti controlli:

- MQSeries è stato correttamente installato sui sistemi di invio e di ricezione ed è stato correttamente configurato per l'accodamento distribuito?

- I collegamenti tra i due sistemi, sono disponibili?

Controllare la disponibilità dei sistemi e la loro connessione a MQSeries. Controllare che la connessione tra i due sistemi e i canali tra i due Queue Manager siano attivi.

- La funzione triggering è stata attivata nel sistema di invio?
- Il messaggio atteso è un messaggio di risposta da un sistema remoto? Controllare che nel sistema remoto sia attiva la funzione triggering.

- La coda è già piena?

Questo potrebbe rendere impossibile ad un'applicazione l'invio di un messaggio alla coda. In questo caso, controllare se il messaggio è stato inviato ad una coda messaggi non recapitati.

L'intestazione della coda messaggi non recapitati contiene un codice di errore o di feedback che spiega il motivo del fallito invio alla coda di destinazione.

Controllare il manuale *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa* per informazioni relative alla struttura dell'intestazione della coda messaggi non recapitati.

- Vi è un errore di corrispondenza tra il Queue Manager di invio e quello di ricezione?

Ad esempio, la dimensione del messaggio potrebbe essere superiore a quella che il Queue Manager ricevente può gestire.

- Le definizioni dei canali di invio sono compatibili con quelle dei canali di ricezione?

Ad esempio, una differenza nel numero di sequenza arresta il componente dell'accodamento distribuito. Consultare il manuale *MQSeries Intercommunication* per ulteriori informazioni relative all'accodamento distribuito.

- È necessaria la conversione dei dati? Se i formati dei dati differiscono tra l'applicazione di invio e quella di ricezione, è necessaria la conversione dei dati. La conversione automatica si verifica all'invio di MQGET se il formato viene riconosciuto come uno di quelli preinstallati.

Se il gruppo dei dati non è riconosciuto per la conversione, il risultato della conversione viene utilizzato per consentire la trasformazione manuale.

Un'eccezione a questa regola si ha nel caso di invio di dati a MQSeries per MVS/ESA.

Consultare il manuale *MQSeries Intercommunication* per ulteriori informazioni sulla conversione dei dati.

---

## Log di errore

MQSeries utilizza diversi log di errore per registrare messaggi relativi alle operazioni dello stesso MQSeries, di un qualunque Queue Manager avviato e i dati di errori provenienti dai canali utilizzati.

L'associazione dell'errore ad un client o la conoscenza del nome del Queue Manager determinano la posizione dei log degli errori.

- Se il nome del Queue Manager è noto e il Queue Manager è disponibile:

## Log di errore

MQS\_ROOT:[MQM.QMGRS.QMgrName.ERRORS]AMQERR01.LOG

- Se il Queue Manager non è disponibile:

MQS\_ROOT:[MQM.QMGRS.\$SYSTEM.ERRORS]AMQERR01.LOG

- Se si è verificato un errore in un'applicazione client:

MQS\_ROOT:[MQM.ERRORS]AMQERR01.LOG

- First Failure Support Technology<sup>®</sup> (FFST) – consultare la sezione “Esame dei FFST” a pagina 215.

**Nota:** nel caso di client, gli errori vengono memorizzati nell'unità principale.

## File di log

Al momento dell'installazione viene creata una directory [MQM.QMGRS.\$SYSTEM.ERRORS] nel percorso per il file QMGRS. La sottodirectory per gli errori può contenere fino a tre file di log degli errori denominati:

- AMQERR01.LOG
- AMQERR02.LOG
- AMQERR03.LOG

Dopo la creazione di un Queue Manager, in funzione delle sue necessità, vengono creati tre file di log degli errori. Questi file vengono denominati nello stesso modo di quelli \$SYSTEM e cioè AMQERR01, AMQERR02 e AMQERR03; ognuno ha una capacità di 256 KB. I file vengono memorizzati nella sottodirectory di errore di ogni Queue Manager creato.

Una volta prodotti, i messaggi di errore vengono memorizzati in AMQERR01. Quando AMQERR01 supera i 256 KB viene copiato in AMQERR02. Prima di questa operazione, AMQERR02 viene copiato in AMQERR03.LOG. L'eventuale contenuto di AMQERR03 viene eliminato.

I messaggi di errore più recenti sono quelli memorizzati in AMQERR01; gli altri file vengono utilizzati per conservare una storia dei messaggi di errore.

Tutti i messaggi relativi ai canali vengono inoltre memorizzati nei file di errore del proprio Queue Manager, a meno che il nome del Queue Manager è sconosciuto o il Queue Manager è indisponibile. In questi casi i messaggi relativi ai canali vengono memorizzati nella sottodirectory [MQM.QMGRS.\$SYSTEM.ERRORS].

Per esaminare il contenuto dei file di log degli errori, utilizzare l'editor OpenVMS.

## Errori precedenti

Esistono diverse situazioni particolari in cui possono verificarsi errori quando i log di errore di cui sopra non sono ancora stati creati. MQSeries cerca di registrare ogni errore in un log di errore. La posizione del log dipende da quanti Queue Manager vengono stabiliti.

Se non possono essere determinate informazioni sulla posizione, a causa, ad esempio, dell'alterazione di file di configurazione, gli errori vengono collegati ad una directory di errori creata al momento dell'installazione nella directory principale, mqm.

Se il file di configurazione MQSeries e l'attributo DefaultPrefix della voce AllQueueManagers sono leggibili, gli errori vengono collegati nella directory DefaultPrefix[.errors].

Per ulteriori informazioni relative alla configurazione, consultare il “Capitolo 13. Configurazione MQSeries” a pagina 179.

## Messaggi per l'operatore

In MQSeries per Compaq OpenVMS, i messaggi per l'operatore indicano errori generalmente causati dall'utente che immette parametri non validi per un determinato comando. I messaggi per l'operatore sono abilitate al supporto lingue nazionali (NLS), grazie ai cataloghi di messaggio installati nelle posizioni standard.

Questi messaggi vengono compilati nelle eventuali finestre associate ed anche nel log di errore AMQERR01.LOG nella directory del Queue Manager. Ad esempio:

```
MQS_ROOT:[MQM.QMGRS.QUEUE$MANAGER.ERRORS]
```

Alcuni errori vengono collegati al file AMQERR01.LOG nella directory del Queue Manager ed altri nella directory \$SYSTEM copia del log di errore.

## Esempio di log degli errori

Questo esempio mostra parte di un log degli errori MQSeries per Compaq OpenVMS:

```

06/29/00 09:41:39 AMQ7467: The oldest log file required to start queue
manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record
required to restart the queue manager. Log records older than this may
be required for media recovery.
ACTION: You can move log files older than S0000000.LOG to an archive
medium to release space in the log directory. If you move any of the log
files required to recreate objects from their media images, you will
have to restore them to recreate the objects.
-----
06/29/00 09:41:39 AMQ7468: The oldest log file required to perform media
recovery of queue manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record
required to recreate any of the objects from their media images. Any
log files prior to this will not be accessed by media recovery operations.
ACTION: You can move log files older than S0000000.LOG to an archive
medium to release space in the log directory.
-----
06/29/00 09:42:05 AMQ7467: The oldest log file required to start queue
manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record
required to restart the queue manager. Log records older than this
may be required for media recovery.
ACTION: You can move log files older than S0000000.LOG to an archive
medium to release space in the log directory. If you move any of the log
files required to recreate objects from their media images,
you will have to restore them to recreate the objects.
-----
06/29/00 09:42:05 AMQ7468: The oldest log file required to perform media
recovery of queue manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record
required to recreate any of the objects from their media images. Any
log files prior to this will not be accessed by media recovery operations.
ACTION: You can move log files older than S0000000.LOG to an archive
medium to release space in the log directory.
-----
06/29/00 09:42:06 AMQ8003: MQSeries queue manager started.

EXPLANATION: MQSeries queue manager BKM1 started.
ACTION: None.
-----
06/29/00 09:42:06 AMQ7467: The oldest log file required to start queue
manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record
required to restart the queue manager. Log records older than this

```

## Log di errore

may be required for media recovery.  
ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory. If you move any of the log files required to recreate objects from their media images, you will have to restore them to recreate the objects.

-----  
06/29/00 09:42:06 AMQ7468: The oldest log file required to perform media recovery of queue manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record required to recreate any of the objects from their media images. Any log files prior to this will not be accessed by media recovery operations.  
ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory.

-----  
06/29/00 09:46:27 AMQ7030: Request to quiesce the queue manager accepted. The queue manager will stop when there is no further work for it to perform.

EXPLANATION: You have requested that the queue manager end when there is no more work for it. In the meantime, it will refuse new applications that attempt to start, although it allows those already running to complete their work.  
ACTION: None.

-----  
06/29/00 09:46:43 AMQ7467: The oldest log file required to start queue manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record required to restart the queue manager. Log records older than this may be required for media recovery.  
ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory. If you move any of the log files required to recreate objects from their media images, you will have to restore them to recreate the objects.

-----  
06/29/00 09:46:43 AMQ7468: The oldest log file required to perform media recovery of queue manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record required to recreate any of the objects from their media images. Any log files prior to this will not be accessed by media recovery operations.  
ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory.

-----  
06/29/00 09:46:44 AMQ8004: MQSeries queue manager ended.

EXPLANATION: MQSeries queue manager BKM1 ended.  
ACTION: None.

-----  
06/29/00 09:46:59 AMQ7467: The oldest log file required to start queue manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record required to restart the queue manager. Log records older than this may be required for media recovery.  
ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory. If you move any of the log files required to recreate objects from their media images, you will have to restore them to recreate the objects.

-----  
06/29/00 09:47:00 AMQ7468: The oldest log file required to perform media recovery of queue manager BKM1 is S0000000.LOG.

EXPLANATION: The log file S0000000.LOG contains the oldest log record required to recreate any of the objects from their media images. Any log files prior to this will not be accessed by media recovery operations.  
ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory.

-----  
06/29/00 09:47:08 AMQ7472: Object TEST1, type queue damaged.

EXPLANATION: Object TEST1, type queue has been marked as damaged. This indicates that the queue manager was either unable to access the object in the file system, or that some kind of inconsistency with the data in the object was detected.  
ACTION: If a damaged object is detected, the action performed depends on whether the queue manager supports media recovery and when the damage was detected. If the queue manager does not support media recovery,



you must delete the object as no recovery is possible. If the queue manager does support media recovery and the damage is detected during the processing performed when the queue manager is being started, the queue manager will automatically initiate media recovery of the object. If the queue manager supports media recovery and the damage is detected once the queue manager has started, it may be recovered from a media image using the rcrmobj command or it may be deleted.

-----  
06/29/00 09:47:09 AMQ8003: MQSeries queue manager started. --

EXPLANATION: MQSeries queue manager BKM1 started.  
ACTION: None.

-----  
06/29/00 09:47:09 AMQ7467: The oldest log file required to start queue manager BKM1 is S0000000.LOG. --

EXPLANATION: The log file S0000000.LOG contains the oldest log record required to restart the queue manager. Log records older than this may be required for media recovery.

ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory. If you move any of the log files required to recreate objects from their media images, you will have to restore them to recreate the objects.

-----  
06/29/00 09:47:10 AMQ7468: The oldest log file required to perform media recovery of queue manager BKM1 is S0000000.LOG. --

EXPLANATION: The log file S0000000.LOG contains the oldest log record required to recreate any of the objects from their media images. Any log files prior to this will not be accessed by media recovery operations.

ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory.

-----  
06/29/00 09:47:47 AMQ7081: Object TEST1, type queue recreated. --

EXPLANATION: The object TEST1, type queue was recreated from its media image.  
ACTION: None.

-----  
06/29/00 11:22:10 AMQ7467: The oldest log file required to start queue manager BKM1 is S0000000.LOG. --

EXPLANATION: The log file S0000000.LOG contains the oldest log record required to restart the queue manager. Log records older than this may be required for media recovery.

ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory. If you move any of the log files required to recreate objects from their media images, you will have to restore them to recreate the objects.

-----  
06/29/00 11:22:10 AMQ7468: The oldest log file required to perform media recovery of queue manager BKM1 is S0000000.LOG. --

EXPLANATION: The log file S0000000.LOG contains the oldest log record required to recreate any of the objects from their media images. Any log files prior to this will not be accessed by media recovery operations.

ACTION: You can move log files older than S0000000.LOG to an archive medium to release space in the log directory.

-----  
06/29/00 11:22:11 AMQ8004: MQSeries queue manager ended. --

EXPLANATION: MQSeries queue manager BKM1 ended.  
ACTION: None.

-----  
...

---

## Code messaggi non recapitati

I messaggi che per qualsiasi motivo non vengono recapitati vengono inviati alla coda messaggi non recapitati. Immettendo il comando MQSC DISPLAY QUEUE è possibile controllare la presenza di messaggi nella coda. Se la coda contiene messaggi, utilizzare l'applicazione di esempio browse (amqsbcg) fornita per sfogliare i messaggi utilizzando la chiamata MQGET. L'applicazione di esempio

## Code messaggi non recapitati

guida attraverso tutti i messaggi di una data coda relativa ad un dato Queue Manager, visualizzando sia la descrizione, che i campi relativi al contesto di tutti i messaggi di una data coda.

È necessario decidere come disporre dei messaggi trovati nella coda messaggi non recapitati, in base al motivo che ha determinato questa loro destinazione.

Possono verificarsi dei problemi nel caso in cui non sia definita una coda messaggi non recapitati su ciascun Queue Manager utilizzato. Quando si crea un Queue Manager utilizzando il comando **crtmqm**, viene automaticamente creata come oggetto predefinito una coda messaggi non recapitati, denominata SYSTEM.DEAD.LETTER.QUEUE. Tuttavia, questa coda non è definita come la coda messaggi non recapitati relativa al Queue Manager. Consultare la sezione "Definizione di una coda messaggi non recapitati" a pagina 48.

---

## File di configurazione e determinazione dei problemi

Gli errori dei file di configurazione escludono la possibilità che vengano rilevati errori del tipo "queue manager unavailable".

Vi sono diversi controlli da eseguire sui file di configurazione:

- Verificare che i file di configurazione siano stati creati.
- Verificare che siano state assegnate le necessarie autorizzazioni, ad esempio:  

```
MQS.INI;1 MQM (RWED, RWED, RW, R)
(identifier=MQM, ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)
```
- Verificare che il file di configurazione MQSeries si riferisca al Queue Manager e alle directory di log appropriate.

---

## Utilizzo della traccia MQSeries

MQSeries per Compaq OpenVMS utilizza i seguenti comandi per le funzioni di traccia:

- **strmqtrc** – consultare la sezione "strmqtrc (Avvio traccia di MQSeries)" a pagina 320
- **dspmqtrc** – consultare la sezione "dspmqtrc (Visualizza output di traccia formattato di MQSeries)" a pagina 275
- **endmqtrc** – consultare la sezione "endmqtrc (End MQSeries trace)" a pagina 286

La funzione di traccia utilizza un file per ogni entità di cui si esegue la traccia e registra le relative informazioni nel file appropriato.

Nella directory MQS\_ROOT:[MQM.TRACE] vengono creati file associati alla traccia.

I file di questa directory contengono dettagli relativi ai Queue Manager, insieme a tutti i dati di traccia più recenti e ai dati di traccia relativi al \$SYSTEM.

## Nomi dei file di traccia

I nomi dei file di traccia sono strutturati nel seguente modo:

```
AMQppppppp.TRC
```

*pppppppp* indica l'identificativo di processo (PID) relativo al processo che produce la traccia.



**Note:**

1. In MQSeries per Compaq OpenVMS, il valore dell'identificativo di processo è sempre di otto caratteri.
2. Dovrà esserci un file di traccia per ogni processo in esecuzione come parte dell'entità di cui viene eseguita la traccia.

**Esempio di dati traccia**

L'esempio che segue rappresenta un estratto di traccia OpenVMS:

ID	ELAPSED_MSEC	DELTA_MSEC	APPL	SYSCALL	KERNEL	INTERRUPT
...						
30d	0 0	MQS CEI	Exit!	12484.1	xcsWaitEventSem	rc=10806020
30d	0 0	MQS CEI	Exit!	12484.1	zcpReceiveOnLink	rc=20805311
30d	0 0	MQS FNC	Entry	12484.1	zxcProcessChildren	
30d	0 0	MQS CEI	Entry.	12484.1	xcsRequestMutexSem	
30d	1 0	MQS CEI	Entry..	12484.1	xcsHSHMEMBtoPTR	
30d	1 0	MQS CEI	Exit...	12484.1	xcsHSHMEMBtoPTR	rc=00000000
30d	1 0	MQS FNC	Entry..	12484.1	xllSemGetVal	
30d	1 0	MQS FNC	Exit...	12484.1	xllSemGetVal	rc=00000000
30d	1 0	MQS FNC	Entry..	12484.1	xllSemReq	
30d	1 0	MQS FNC	Exit...	12484.1	xllSemReq	rc=00000000
30d	1 0	MQS CEI	Exit..	12484.1	xcsRequestMutexSem	rc=00000000
30d	2 0	MQS CEI	Entry.	12484.1	xcsReleaseMutexSem	
30d	2 0	MQS CEI	Entry..	12484.1	xcsHSHMEMBtoPTR	
30d	2 0	MQS CEI	Exit...	12484.1	xcsHSHMEMBtoPTR	rc=00000000
30d	2 0	MQS FNC	Entry..	12484.1	xllSemRel	
30d	2 0	MQS FNC	Exit...	12484.1	xllSemRel	rc=00000000
30d	2 0	MQS CEI	Exit..	12484.1	xcsReleaseMutexSem	rc=00000000
30d	2 0	MQS CEI	Entry.	12484.1	xcsHSHMEMBtoPTR	
...						

Figura 21. Esempio di traccia MQSeries per Compaq OpenVMS

**Note:**

1. Nell'esempio viene riportata solo una sezione di dati. In una traccia completa vengono riportati per intero i nomi di funzione e i codici di errore.
2. I codici di errore vengono riportati in valori e non in lettere.

**FFST (First Failure Support Technology)**

Le informazioni che solitamente sono registrate nei log FFST, in MQSeries per Compaq OpenVMS, vengono registrate in un file della directory MQS\_ROOT:[MQM.ERRORS].

Questi errori sono molto gravi, generalmente non ripristinabili e indicano un problema di configurazione del sistema o un problema interno di MQSeries.

**Esame dei FFST**

I file sono denominati AMQnnnnnnnn\_mm.FDC, dove:  
nnnnnnnn

indica l'identificativo del processo che riporta l'errore

mm è un numero di sequenza, solitamente 0

Quando un processo crea un FFST immette anche una voce nel log degli errori del sistema. Il record contiene il nome del file FFST da utilizzare nella traccia automatica dei problemi.

## MQSeries First Failure Symptom Report

```

=====
Date/Time      :- Monday January 29 21:32:03 GMT 2001
Host Name     :- CELERY (Unknown)
PIDS          :- 5697175
LVLS          :- 510
Product Long Name :- MQSeries for OpenVMS Alpha
Vendor        :- IBM
Probe Id      :- ZX005025
Application Name :- MQM
Component     :- zxcProcessChildren
Build Date    :- Jan 8 2001
Userid        :- [400,400] (SJACKSON)
Program Name  :- AMQZXMA0.EXE
Process       :- 202001DA
Thread        :- 00000001
QueueManager  :- JJJH
Major Errorcode :- zrcX_PROCESS_MISSING
Minor Errorcode :- OK
Probe Type    :- MSGAMQ5008
Probe Severity :- 2
Probe Description :- AMQ5008: An essential MQSeries process 538968541
                  cannot be found and is assumed to be terminated.
Arith1        :- 538968541 202001dd
VMS Errorcode :- -SYSTEM-W-NONEXPR, nonexistent process (000008E8)

```

## JPI Quota information:

```

=====
ASTCNT=247/250(98%) *          BIOCNT=500/500(100%) *
BYTCNT=183616/183616(100%) *   DIOCNT=250/250(100%) *
ENQCNT=4885/5000(97%) *        FILCNT=241/250(96%) *
PAGFILCNT=975280/1000000(97%) * TQCNT=246/250(98%) *
FREPTCNT=2147483647           APTCNT=0
GPGCNT=5808                   PPGCNT=5872
VIRTPEAK=203264               DFWSCNT=1392
WSAUTH=2784                   WSAUTHEXT=65536
WSEXTENT=65536                WSPEAK=11680
WSQUOTA=2784                   WSSIZE=15792
CPULIM=0                       MAXDETACH=0
MAXJOBS=0                      JOBPRCCNT=2
PAGEFLTS=2895                  PRCCNT=2/100(2%) +
(*) - % resource remaining, (+) - % resource used

```

## Privilege and rights information:

```

=====
CURPRIV=bugchk detach netmbx prmgbl sysgbl sysprv tmpmbx world
IMAGPRIV=bugchk prmgbl sysgbl world
AUTHPRIV=bugchk detach netmbx prmgbl sysgbl sysprv tmpmbx world
SJACKSON                                INTERACT
REMOTE                                  MQM
SYS
IMAGE RIGHTS=
SYS$NODE_CELERY

```

## SYI information:

```

=====
ACTIVE CPU=1/1(100%) +          CLUSTER NODES=1
FREE_GBLPAGES=16000528/16174643(98%) * GBLPAGFIL=1000000
FREE_GBLSECTS=936/1550(60%) *      MEMSIZE=16384
PAGEFILE_FREE=16888/16888(100%) *   PAGE_SIZE=8192
SWAPFILE_FREE=936/936(100%) *       MAXPROCESSCNT=102
PROCSECTCNT=64                    BALSETCNT=100
WSMAX=65536                        NPAGEDYN=2269184
NPAGEVIR=9437184                  PAGEDYN=1597440
VIRTUALPAGECNT=2147483647          LOCKIDTBL_MAX=109437
PQL_DASTLM=24                      PQL_MASTLM=100
PQL_DBIOLM=32                      PQL_MBIOLM=100
PQL_DBYTLM=65536                   PQL_MBYTLM=100000
PQL_DCPULM=0                       PQL_MCPULM=0
PQL_DDIOLM=32                      PQL_MDIOLM=100
PQL_DFILLM=128                     PQL_MFILLM=100
PQL_DPGFLQUOTA=65536               PQL_MPGFLQUOTA=32768
PQL_DPRCLM=32                      PQL_MPRCLM=10
PQL_DTQELM=16                      PQL_MTQELM=0
PQL_DWSDEFAULT=1392                PQL_MWSDEFAULT=1392
PQL_DWSQUOTA=2784                  PQL_MWSQUOTA=2784
PQL_DWSEXTENT=65536                PQL_MWSEXTENT=65536

```

PQL_DENQLM=128	PQL_MENQLM=300
PQL_DJTQUOTA=4096	PQL_MJTQUOTA=0
CLISYMTBL=750	DEFMBXMSG=256
DEFMBXBUFQUO=1056	CHANNELCNT=5000
DLCKEXTRASTK=2560	PIOPAGES=575
CTLPAGES=256	CTLIMGLIM=35
(*) - % resource remaining, (+) - % resource used	

MQM Function Stack  
zxcProcessChildren  
xcsFFST

MQM Trace History

```

--> x11FreeSem
<-- x11FreeSem rc=OK
--> xcsFreeQuickCell
--> x11SpinLockRequest
<-- x11SpinLockRequest rc=OK
--> xstFreeCell
<-- xstFreeCell rc=OK
--> x11SpinLockRelease
<-- x11SpinLockRelease rc=OK
<-- xcsFreeQuickCell rc=OK
<-- xcsCloseEventSem rc=OK
--> xcsFreeMemBlock
--> xstFreeMemBlock
--> xcsRequestThreadMutexSem
<-- xcsRequestThreadMutexSem rc=OK
--> xcsReleaseThreadMutexSem
<-- xcsReleaseThreadMutexSem rc=OK
--> xstFreeBlockFromSharedMemSet
--> x11SpinLockSlowRequest
<-- x11SpinLockSlowRequest rc=OK
--> x11SpinLockRelease
<-- x11SpinLockRelease rc=OK
--> xstFreeBlockInExtent
--> xcsQueryMutexSem
<-- xcsQueryMutexSem rc=OK
--> xcsRequestMutexSem
--> x11SemReq
--> vms_mtx
--> vms_get_lock
<-- vms_get_lock rc=OK
<-- vms_mtx rc=OK
<-- x11SemReq rc=OK
<-- xcsRequestMutexSem rc=OK
--> x11DeleteMutexMem
--> x11CSCloseMutex
--> xihHANDLEtoSUBPOOLFn
--> xihGetConnSPDetailsFromList
--> xihGetConnSPDetails
<-- xihGetConnSPDetails rc=OK
<-- xihGetConnSPDetailsFromList rc=OK
<-- xihHANDLEtoSUBPOOLFn rc=OK
--> x11SpinLockSlowRequest
<-- x11SpinLockSlowRequest rc=OK
--> x11SpinLockRelease
<-- x11SpinLockRelease rc=OK
--> x11FreeSem
<-- x11FreeSem rc=OK
--> vms_mtx
--> vms_get_lock
<-- vms_get_lock rc=OK
<-- vms_mtx rc=OK
--> xcsFreeQuickCell
--> x11SpinLockRequest
<-- x11SpinLockRequest rc=OK
--> xstFreeCell
<-- xstFreeCell rc=OK
--> x11SpinLockRelease
<-- x11SpinLockRelease rc=OK
<-- xcsFreeQuickCell rc=OK
<-- x11CSCloseMutex rc=OK
<-- x11DeleteMutexMem rc=OK
--> xstSerialiseExtent
--> x11SpinLockRequest
<-- x11SpinLockRequest rc=OK
<-- xstSerialiseExtent rc=OK

```

## FFST

```
--> xstFreeChunk
--> xstDeleteChunk
<-- xstDeleteChunk rc=OK
--> xstInsertChunk
<-- xstInsertChunk rc=OK
<-- xstFreeChunk rc=OK
--> xstReleaseSerialisationOnExtent
--> xllSpinLockRelease
<-- xllSpinLockRelease rc=OK
<-- xstReleaseSerialisationOnExtent rc=OK
<-- xstFreeBlockInExtent rc=OK
<-- xstFreeBlockFromSharedMemSet rc=OK
<-- xstFreeMemBlock rc=OK
<-- xcsFreeMemBlock rc=OK
<-- zcpDeleteIPC rc=OK
--> xcsReleaseMutexSem
--> xllSemRel
--> vms_mtx
--> vms_get_lock
<-- vms_get_lock rc=OK
<-- vms_mtx rc=OK
<-- xllSemRel rc=OK
<-- xcsReleaseMutexSem rc=OK
--> xcsFreeMemBlock
--> xstFreeMemBlock
--> xcsRequestThreadMutexSem
<-- xcsRequestThreadMutexSem rc=OK
--> xcsReleaseThreadMutexSem
<-- xcsReleaseThreadMutexSem rc=OK
--> xstFreeBlockFromSharedMemSet
--> xllSpinLockSlowRequest
<-- xllSpinLockSlowRequest rc=OK
--> xllSpinLockRelease
<-- xllSpinLockRelease rc=OK
--> xstFreeBlockInExtent
--> xcsQueryMutexSem
<-- xcsQueryMutexSem rc=OK
--> xcsRequestMutexSem
--> xllSemReq
--> vms_mtx
--> vms_get_lock
<-- vms_get_lock rc=OK
<-- vms_mtx rc=OK
<-- xllSemReq rc=OK
<-- xcsRequestMutexSem rc=OK
--> xclDeleteMutexMem
--> xllCSCloseMutex
--> xihHANDLEtoSUBPOOLFn
--> xihGetConnSPDetailsFromList
--> xihGetConnSPDetails
<-- xihGetConnSPDetails rc=OK
<-- xihGetConnSPDetailsFromList rc=OK
<-- xihHANDLEtoSUBPOOLFn rc=OK
--> xllSpinLockSlowRequest
<-- xllSpinLockSlowRequest rc=OK
--> xllSpinLockRelease
<-- xllSpinLockRelease rc=OK
--> xllFreeSem
<-- xllFreeSem rc=OK
--> vms_mtx
--> vms_get_lock
<-- vms_get_lock rc=OK
<-- vms_mtx rc=OK
--> xcsFreeQuickCell
--> xllSpinLockRequest
<-- xllSpinLockRequest rc=OK
--> xstFreeCell
<-- xstFreeCell rc=OK
--> xllSpinLockRelease
<-- xllSpinLockRelease rc=OK
<-- xcsFreeQuickCell rc=OK
<-- xllCSCloseMutex rc=OK
<-- xclDeleteMutexMem rc=OK
--> xstSerialiseExtent
--> xllSpinLockRequest
<-- xllSpinLockRequest rc=OK
<-- xstSerialiseExtent rc=OK
--> xstFreeChunk
--> xstDeleteChunk
<-- xstDeleteChunk rc=OK
```

```

--> xstInsertChunk
<-- xstInsertChunk rc=OK
<-- xstFreeChunk rc=OK
--> xstReleaseSerialisationOnExtent
--> xllSpinLockRelease
<-- xllSpinLockRelease rc=OK
<-- xstReleaseSerialisationOnExtent rc=OK
<-- xstFreeBlockInExtent rc=OK
<-- xstFreeBlockFromSharedMemSet rc=OK
<-- xstFreeMemBlock rc=OK
<-- xcsFreeMemBlock rc=OK
<-- zxcCleanupAgent rc=OK
--> xcsReleaseMutexSem
--> xllSemRel
--> vms_mtx
--> vms_get_lock
<-- vms_get_lock rc=OK
<-- vms_mtx rc=OK
<-- xllSemRel rc=OK
<-- xcsReleaseMutexSem rc=OK
--> xcsCheckProcess
--> kill
<-- kill rc=OK
<-- xcsCheckProcess rc=OK
--> xcsCheckProcess
--> kill
<-- kill rc=OK
<-- xcsCheckProcess rc=OK
--> xcsRequestMutexSem
--> xllSemReq
--> vms_mtx
--> vms_get_lock
<-- vms_get_lock rc=OK
<-- vms_mtx rc=OK
<-- xllSemReq rc=OK
<-- xcsRequestMutexSem rc=OK
--> xcsReleaseMutexSem
--> xllSemRel
--> vms_mtx
--> vms_get_lock
<-- vms_get_lock rc=OK
<-- vms_mtx rc=OK
<-- xllSemRel rc=OK
<-- xcsReleaseMutexSem rc=OK
--> xcsCheckProcess
--> kill
<-- kill rc=Unknown(FFFF)
<-- xcsCheckProcess rc=xecP_E_INVALID_PID
--> xcsBuildDumpPtr
--> xcsGetMem
<-- xcsGetMem rc=OK
<-- xcsBuildDumpPtr rc=OK
<-- xcsBuildDumpPtr rc=OK
<-- xcsBuildDumpPtr rc=Unknown(4B)
--> xcsFFST

```

```

ECAnchor
6A91B0          5A584541          ZXEA
6A91C0  03000000  E8030000  03220000  0100DA01  ...&#232;...&#218;.
6A91D0  2C05A500  D4040000  0A00DA01  01000000  ,.¥.&#212;...&#218;....
6A91E0  B0000000  0A00DA01  03000000  E8030000  °....&#218;....&#232;...
6A91F0  03220000  0100DA01  07000000  00000000  ."....&#218;.....
6A9200  283F9700  03000000  F0030000  08410000  (?-.....A..
6A9210  0300DA01  03000000  F0030000  08410000  ..&#218;.....A..
6A9220  0300DA01  01000000  B4000000  0200DA01  ..&#218;....&#218;.
6A9230  03000000  E8030000  03220000  0100DA01  ...&#232;...&#218;.
6A9240  00000000  00000000  00000000  00000000  .....
6A9250  00000000  00000000  00000000  00000000  .....
6A9260  00000000  01000000  B4000000  0200DA01  .....&#218;.
6A9270  03000000  E8030000  03220000  0100DA01  ,...&#232;...&#218;.
6A9280  B41F0000  0200DA01  01000000  B4000000  ....&#218;.....
6A9290  0200DA01  03000000  E8030000  03220000  ..&#218;....&#232;...".
6A92A0  0100DA01  DA012020  00000000  EFCD0300  ..&#218;.&#218;. ....&#205;.
6A92B0  00000000  00000000  00000000  00000000  .....
6A92C0  00000000  00000000  01000000  00000000  .....
6A92D0  44454641  554C5400  00000000  00000000  DEFAULT.....
6A92E0  00000000  00000000  00000000  00000000  .....
6A92F0  00000000  00000000  00000000  00000000  .....
6A9300  2F6D7173  5F726F6F  742F6D71  6D000000  /mqs_root/mqm...
6A9310  00000000  00000000  00000000  00000000  .....

```

## FFST

```
6A9320 to 6A93F0 suppressed, lines same as above
6A9400 5A435048 01000000 B8000000 0400DA01 ZCPH....&#184;....&#218;.
6A9410 03000000 F0030000 08410000 0300DA01 .....A....&#218;.
6A9420 DC040000 0400DA01 01000000 B8000000 &#220;....&#218;....&#184;...
6A9430 0400DA01 03000000 F0030000 08410000 ..&#218;.....A..
6A9440 0300DA01 00000000 00000000 00000000 ..&#218;.....
6A9450 00000000 00000000 00000000 00000000 .....
6A9460 to 6A9470 suppressed, lines same as above
6A9480 00000000 00000000 5A435048 01000000 .....ZCPH....
6A9490 B8000000 0500DA01 03000000 F0030000 &#184;....&#218;.....
6A94A0 08410000 0300DA01 DC040000 0500DA01 .A....&#218;. &#220;....&#218;.
6A94B0 01000000 B8000000 0500DA01 03000000 ....&#184;....&#218;....
6A94C0 F0030000 08410000 0300DA01 4C4B0000 .....A....&#218;.LK..
6A94D0 0500DA01 01000000 B8000000 0500DA01 ..&#218;....&#184;....&#218;.
6A94E0 03000000 F0030000 08410000 0300DA01 .....A....&#218;.
6A94F0 07000000 00090000 50140000 0100DA01 .....P....&#218;.
6A9500 01000000 B0000000 0100DA01 03000000 .....°....&#218;....
6A9510 E8030000 03220000 0100DA01 98170000 &#232;...."....&#218;.~...
6A9520 0200DA01 01000000 B4000000 0200DA01 ..&#218;....&#218;.
6A9530 03000000 E8030000 03220000 0100DA01 ....&#232;...."....&#218;.
6A9540 00000000 00000000 08000000 3C0A0000 .....<...
6A9550 50140000 0100DA01 01000000 B0000000 P....&#218;....°...
6A9560 0100DA01 03000000 E8030000 03220000 ..&#218;....&#232;...."..
6A9570 0100DA01 08180000 0200DA01 01000000 ,&#218;....&#218;....
6A9580 B4000000 0200DA01 03000000 E8030000 .....&#218;....&#232;...
6A9590 03220000 0100DA01 00000000 00000000 ."....&#218;.....
6A95A0 00000000 00000000 00000000 00000000 .....
6A95B0 to 6A9650 suppressed, lines same as above
6A9660 01000000 00000000 78180000 0200DA01 .....x....&#218;.
6A9670 01000000 B4000000 0200DA01 03000000 ....&#218;....
6A9680 E8030000 03220000 0100DA01 09000000 &#232;...."....&#218;....
6A9690 780B0000 50140000 0100DA01 01000000 x...P....&#218;....
6A96A0 B0000000 0100DA01 03000000 E8030000 °....&#218;....&#232;...
6A96B0 03220000 0100DA01 00000000 00000000 ."....&#218;.....
6A96C0 00000000 00000000 00000000 00000000 .....
6A96D0 to 6A9750 suppressed, lines same as above
6A9760 00000000 00000000 DD012020 00000000 .....&#221;. ....
6A9770 00000000 00000000 01000000 09000000 .....
6A9780 00000000
```

Function Stack e Trace History vengono utilizzati da IBM per semplificare la determinazione dei problemi. Nella maggior parte dei casi l'amministratore di sistema può fare ben poco qualora si verifichi un errore FFST, se non riportare il problema ai Centri supporto.

Tuttavia, vi è un gruppo di problemi che potrebbero essere risolti dagli amministratori. Se, richiamando una delle funzioni interne, FFST riporta la descrizione "quota exceeded" o "out of space on device", è probabile che i limiti relativi al parametro SYSGEN siano stati superati.

Per risolvere il problema, modificare i parametri di sistema in modo da aumentarne i limiti interni. Consultare il "Capitolo 13. Configurazione MQSeries" a pagina 179 per ulteriori informazioni.

---

## Determinazione dei problemi relativi ai client

Un'applicazione client MQI riceve codici di errore MQR\*\_\* così come le applicazioni non-client MQI. Tuttavia, sono adesso previsti ulteriori codici di errore associati a situazioni specifiche dei client. Ad esempio:

- La macchina remota non risponde
- Errore relativo alla linea di comunicazioni
- Indirizzo di macchina non valido

I casi più frequenti di errori si hanno quando un'applicazione inoltra un MQCONN e riceve la risposta MQR\*\_Q\_MQR\_NOT\_AVAILABLE. Un messaggio di errore compilato nel file di log client, spiega la causa dell'errore. Possono essere collegati messaggi anche al server, in funzione del tipo di malfunzionamento.

### Client arrestati

Anche dopo l'arresto di un client, il processo di un server può rimanere impegnato nell'elaborazione delle code relative al client. Solitamente, questo può accedere per un periodo di tempo ristretto, cioè fin quando il supporto di comunicazione rileva che il componente è stato escluso.

### Messaggi di errore relativi ai client

Quando si verifica un errore relativo ad un sistema client, i messaggi di errore vengono inviati ai file di errore associati al server. Se un errore non può essere memorizzato in quei file, il codice client tenterà di memorizzare il messaggio di errore in un log degli errori nella directory principale della macchina client.

#### **OS/2, UNIX e sistemi client OpenVMS**

Messaggi di errore per OS/2, UNIX e sistemi client OpenVMS vengono memorizzati nei log degli errori dei rispettivi sistemi server MQSeries. Di solito, questi file vengono visualizzati nella directory MQS\_ROOT:[MQM.ERRORS] sui sistemi OpenVMS ed in /var/mqm/errors nei sistemi UNIX.

#### **Client DOS e Windows®**

La posizione del file di log AMQERR01.LOG è impostata dalla variabile di ambiente MQDATA. Se non viene modificata da MQDATA, la posizione predefinita è:

C:\

L'elaborazione in ambiente DOS richiede la variabile di ambiente MQDATA.

Questa è la libreria predefinita utilizzata dal codice client per memorizzare informazioni relative alle tracce e agli errori; inoltre contiene il nome della directory in cui è memorizzato il file qm.ini (necessario all'impostazione di NetBIOS). Se non specificato diversamente, viene utilizzata l'unità C predefinita.

I nomi di file predefiniti contenuti in questa libreria sono:

#### **AMQERR01.LOG**

per i messaggi di errore.

#### **AMQERR01.FDC**

per i messaggi FFDC (First Failure Data Capture).

## Determinazione dei problemi relativi ai client



---

## Capitolo 15. Ottimizzazione delle prestazioni

In questo capitolo si discutono le modalità di ottimizzazione del sistema OpenVMS per ottenere le migliori prestazioni da MQSeries.

Non è possibile per un prodotto come MQSeries definire i valori per i vari parametri di ottimizzazione di OpenVMS che siano corretti per tutte le circostanze. I valori più efficaci vengono determinati dal carico di lavoro per lo stesso MQSeries e dal sistema OpenVMS nell'insieme. Sebbene le impostazioni di parametro indicate nel volume *MQSeries per Compaq OpenVMS Alpha, Version 5.1 Quick Beginnings* forniscano il minimo ragionevole o i valori iniziali, potrebbe essere necessario aumentare tali valori quando si sviluppa il carico di lavoro per i Queue Manager. Tale processo viene indicato come "ottimizzazione".

L'ottimizzazione delle prestazioni di qualsiasi sistema OpenVMS è descritta nel volume *OpenVMS Performance Management*. Utilizzare le informazioni presenti in tale volume così come i punti seguenti relativi in modo specifico a MQSeries:

- Alcuni parametri di ottimizzazione si applicano all'intero sistema (ad esempio, GBLPAGES). Questi parametri vengono controllati dall'utilità SYSGEN e sono pertanto talvolta denominati parametri SYSGEN. Se utilizzato regolarmente, il meccanismo AUTOGEN FEEDBACK controlla le risorse utilizzate dal sistema e regola automaticamente i parametri SYSGEN per tracciare il carico di lavoro in continuo cambiamento. Ciò può notevolmente ridurre l'intervento manuale necessario a mantenere la corretta regolazione del sistema contribuisce a evitare gli errori provocati dall'esaurimento delle risorse. I parametri SYSGEN relativi a MQSeries sono:

### **GBLPAGES, GBLSECTIONS e GBLPAGFIL**

Il Queue Manager viene implementato come un insieme di processi cooperanti che comunicano tramite la memoria condivisa (globale). Pertanto, è importante assicurarsi che i parametri SYSGEN che controllano la memoria globale (GBLPAGES, GBLSECTIONS e GBLPAGFIL) siano sufficientemente grandi. Il volume *MQSeries per Compaq OpenVMS Alpha, Version 5.1 Quick Beginnings* fornisce valori iniziali ragionevoli per tali parametri. Quando il numero di utenti di MQSeries cresce, aumenta la richiesta di memoria globale e può essere necessario aumentare anche i corrispondenti parametri SYSGEN.

### **CHANNELCNT**

Il Queue Manager elabora l'utilizzo delle caselle di posta OpenVMS come una comunicazione interprocesso e meccanismo di sincronizzazione. L'accesso a tali caselle di posta avviene tramite i canali, rendendo in tal modo necessario assicurarsi che il parametro CHANNELCNT di SYSGEN sia abbastanza grande. Nella maggior parte dei casi sarà sufficiente il valore impostato durante l'installazione. Tuttavia, in sistemi sovraccaricati con molti processi attivi di MQSeries, può essere necessario un aumento del valore.

Per impostare esplicitamente i parametri SYSGEN, modificare il file MODPARAMS.DAT come descritto nel volume *OpenVMS Performance Management*.

- Alcuni parametri di ottimizzazione di OpenVMS si applicano a singoli nomi utente o processi (ad esempio, PGFLQUOTA). Questi parametri sono controllati

## Ottimizzazione delle prestazioni

tipicamente (ma non sempre) dall'utilità AUTHORIZE. Non esistono metodi automatici per la regolazione di tali parametri. Tuttavia, poiché questi parametri rappresentano un limite sulla quantità di alcune risorse che un particolare processo può utilizzare, è possibile impostarli su un valore più alto di quello strettamente necessario, in modo da fornire maggiore capacità negli occasionali carichi di punta. I parametri specifici del processo relativi a MQSeries sono:

### PGFLQUOTA

Esso controlla la quantità di spazio di file di paging utilizzabile da un processo. Poiché i processi di MQSeries di norma trasportano messaggi che possono essere molto grandi o numerosi o entrambi, potenzialmente potrebbero consumare grandi quantità di spazio di file di paging.

### PRCLM

Questo parametro controlla il numero di sottoprocessi che possono essere creati da un determinato processo. Poiché la maggior parte dei processi MQSeries viene creata come sottoprocessi del controllore d'esecuzione, il sistema avrà bisogno di un valore elevato per PRCLM.

### ENQLM, ASTLM, TQELM

Come già accennato, il Queue Manager viene implementato come un insieme di processi cooperanti. Tali processi sincronizzano le relative attività utilizzando il lock manager, le AST (Asynchronous System Traps) e i timer di OpenVMS. I tre parametri che limitano l'utilizzo di queste risorse devono essere impostati con valori abbastanza grandi da far fronte alle necessità del Queue Manager.

---

## Impostazione del valore dei parametri specifici di processo

Il modo più comune per impostare questi parametri è di impiegare l'utilità AUTHORIZE per regolare i valori per il nome utente appropriato (che per MQSeries è di solito MQM).

Tuttavia, su OpenVMS alcune quote di processo vengono condivise da tutti i processi nello stesso lavoro, cioè, un processo parent e tutti gli altri processi che sono sottoprocessi del parent. Le quote di processo in questa categoria includono BYTLM, FILLM, PGFLQUOTA, PRCLM, TQELM e ENQLM e sono denominate *quote di pool*.

Poiché la maggior parte dei processi di Queue Manager vengono creati come sottoprocessi del controllore di esecuzione, ne consegue che le quote di pool sono condivise da tutti i processi di Queue Manager. Pertanto, è di norma necessario impostare tali quote su valori che sembrano eccessivi per un singolo processo. Ciò è in particolar modo vero per PGFLQUOTA, poiché questo parametro limita la quantità di memoria virtuale che i processi di Queue Manager possono creare **collettivamente**. Per tale motivo, quando il controllore di esecuzione si avvia, non ottiene i relativi valori di quota iniziali dal file di autorizzazione conservato da AUTHORIZE, ma li imposta esplicitamente su valori ragionevoli. Di conseguenza, non è più possibile utilizzare AUTHORIZE per modificare tali valori. È possibile invece sovrascrivere i valori di quota espliciti utilizzando i seguenti nomi logici:

```
MQS_ASTLM
MQS_BIOLM
MQS_BYTLM
MQS_DIOLM
MQS_ENQLM
MQS_FILLM
MQS_PGFLQUOTA
MQS_PRCLM
MQS_TQELM
```

## Impostazione dei parametri specifici di processo

È possibile impostare tali nomi logici utilizzando un file denominato `SYSDMANAGER:MQS_SYSTARTUP.COM`. MQSeries fornisce un file denominato `SYSDMANAGER:MQS_SYSTARTUP.TEMPLATE`, che può essere modificato e rinominato. Ad esempio, per fornire un valore differente per il parametro `PGFLQUOTA`:

1. Copiare il file `.TEMPLATE`, `MQS_SYSTARTUP.TEMPLATE` in un file `.COM`.
2. Modificare il file `MQS_SYSTARTUP.COM` appena creato in modo da attivare le righe che definiscono i nomi logici corrispondenti alle quote del processo, come `mqs_pgflquota`.
3. Definire nuovi valori.

Ad esempio:

```
$! DEFINE/SYSTEM MQS_PGFLQUOTA 1000000
```

potrebbe diventare

```
$ DEFINE/SYSTEM MQS_PGFLQUOTA 5000000
```

4. Richiamare il file `mqs_systartup` per definire i nomi logici. Ad esempio:

```
$ @sys$manager:mqs_systartup
```

Di norma, ciò viene effettuato come parte della procedura di avvio del sistema.

La carenza di quote di pool diventa di norma visibile quando una nuova applicazione client non riesce a connettersi al Queue Manager oppure qualche altra applicazione subisce un malfunzionamento poiché la nuova connessione ha consumato troppe risorse.

Inoltre, si noti che essendo stato avviato il controllore di esecuzione con impostazioni esplicite per molte delle relative quote, i parametri `SYSGEN PQL_D*` non si applicano a EC.



---

## Capitolo 16. MQSeries per OpenVMS e clustering

*I cluster OpenVMS e i cluster di Queue Manager MQSeries sono cose diverse, indipendenti fra loro.*

**Nota:** quando si utilizza il termine *cluster*, ci si riferisce ad un cluster di Queue Manager MQSeries. Per riferirsi ad un cluster OpenVMS si utilizza sempre l'espressione *cluster OpenVMS*.

I cluster di Queue Manager MQSeries non utilizzano necessariamente protocolli di intercomunicazione cluster OpenVMS, né OpenVMS cluster distributed lock manager o OpenVMS cluster file system. Tutte le comunicazioni tra i Queue Manager in un cluster MQSeries avvengono attraverso i canali MQSeries utilizzando uno dei protocolli supportati. Così è possibile configurare cluster Queue Manager MQSeries con Queue Manager eseguiti su sistemi OpenVMS, purché non siano parti dello stesso cluster OpenVMS.

Se un Queue Manager MQSeries viene configurato all'interno di un cluster OpenVMS, potrà essere eseguito solo su un nodo OpenVMS (di seguito, in questo capitolo, nodo) del cluster OpenVMS. LA funzione di un singolo Queue Manager MQSeries non può essere distribuito tra più nodi OpenVMS di un cluster OpenVMS. Se si tenta di avviare un Queue Manager MQSeries su più di un nodo, verrà riportato un messaggio di errore. Tuttavia, se vengono configurati più Queue Manager MQSeries su un cluster OpenVMS, è possibile eseguirli su differenti nodi del cluster OpenVMS.

Per fornire un maggior livello di disponibilità di Queue Manager MQSeries in un cluster OpenVMS, è stata aggiunta una nuova funzione definita Failover Set (Gruppi di Ripristino) in MQSeries V5.1. Tale funzione consente di riavviare automaticamente un Queue Manager su un altro nodo di un cluster OpenVMS, nel caso si verificano malfunzionamenti. Questa funzione può essere utilizzata con o senza cluster Queue Manager MQSeries. Consultare la sezione "Gruppi di ripristino di cluster OpenVMS" a pagina 228.

---

### Installazione di MQSeries in un cluster OpenVMS

L'installazione di MQSeries per Compaq OpenVMS Alpha, V5.1 in un cluster OpenVMS è molto simile all'installazione di MQSeries su un sistema autonomo OpenVMS. Tuttavia, prima di eseguire l'installazione, considerare quanto segue:

- Se sono presenti più dischi di sistema nel cluster OpenVMS, è necessario installare MQSeries su ciascun disco di sistema che possiede un nodo avviato da esso e che deve eseguire MQSeries. MQSeries deve essere installato su ogni disco e non su ogni nodo.
- Il disco che contiene la struttura di directory MQS\_ROOT deve essere montato su tutti i nodi OpenVMS che devono eseguire i Queue Manager contenuti nella struttura di directory. È possibile che ci siano strutture di directory MQS\_ROOT diverse per ciascun nodo. Tuttavia, nel configurare gruppi di ripristino, ogni nodo OpenVMS in un gruppo di ripristino deve corrispondere alla medesima struttura di directory MQS\_ROOT. Per ciascuna installazione di MQSeries è necessario specificare la directory MQS\_ROOT (come risposta alla richiesta: 'Enter the root device for the MQSeries datafiles:') .

## Installazione in un cluster OpenVMS

- Se il disco che contiene la struttura di directory MQS\_ROOT è diverso da quello che contiene i file di log per un Queue Manager MQSeries, quest'ultimo deve essere mounted system-wide on all nodes nell'impostazione di ripristino.
- MQSeries utilizza un MQM che ha come directory predefinita SYS\$SPECIFIC:[MQS\_SERVER]. Questa directory viene creata solo sul nodo in cui viene installato MQSeries. La directory deve essere creata per ogni ulteriore nodo che viene avviato dallo stesso disco di sistema e che deve eseguire MQSeries. A questo scopo, possono essere utilizzati i seguenti comandi DCL su ogni ulteriore nodo:

```
$create/directory sys$specific:[mqs_server]/owner=[mqs_server] -  
/protection=(s:rwed,o:rwed,g,w)  
$set sec/acl=(identifier=mqm,options=default,access=r+w+e+d+c) -  
sys$specific:[000000]mqs_server.dir  
$set sec/acl=(identifier=mqm,access=r+w+e+d+c) -  
sys$specific:[000000]mqs_server.dir
```

---

## Gruppi di ripristino di cluster OpenVMS

### Panoramica dei gruppi di ripristino di cluster OpenVMS

I gruppi di ripristino di cluster OpenVMS costituiscono una nuova funzione disponibile in MQSeries per Compaq OpenVMS, V5.1. In caso di malfunzionamento di un Queue Manager MQSeries, grazie a questa funzione, è possibile riavviare automaticamente il Queue Manager su un altro nodo in un cluster OpenVMS. I gruppi di ripristino di cluster OpenVMS supportano i seguenti tipi di malfunzionamento:

- Arresto di un nodo che esegue un Queue Manager MQSeries
- Grave malfunzionamento del sistema di un nodo che esegue un Queue Manager MQSeries
- Chiusura di un nodo che esegue un Queue Manager MQSeries, senza la corretta chiusura di quest'ultimo
- Malfunzionamento di un processo Queue Manager MQSeries Execution Controller

I seguenti tipi di malfunzionamento non sono supportati dai gruppi di ripristino:

- Errore verificatosi su un nodo che esegue un Queue Manager MQSeries, che tuttavia non causa il malfunzionamento del nodo o del Queue Manager.
- Il malfunzionamento di un processo Queue Manager MQSeries, tranne il caso del processo Execution Controller. Un Queue Manager MQSeries viene riavviato automaticamente, ma mai sullo stesso nodo.
- Malfunzionamento hardware o software del disco che contiene i file di coda e i dati di log del Queue Manager MQSeries.
- Danneggiamento dei file di coda o dei dati di log del Queue Manager MQSeries.

I gruppi di ripristino di cluster OpenVMS sono supportati solo dai Queue Manager che utilizzano il protocollo TCP/IP per canali MQSeries. I seguenti gruppi TCP/IP sono supportati:

- Digital® TCP/IP Services per OpenVMS V5.0A
- Process Software's TCPware® per OpenVMS V5.4
- Process Software's Multinet® per OpenVMS 4.3

## Nozione relative ai gruppi di ripristino di cluster OpenVMS

Un gruppo di ripristino di cluster OpenVMS è un insieme di nodi in grado di eseguire un Queue Manager MQSeries. Possono esservi da uno a quattro nodi in un gruppo di ripristino di cluster OpenVMS e tutti i nodi devono essere membri del cluster OpenVMS. Un gruppo di ripristino di cluster OpenVMS è relativo ad un solo Queue Manager MQSeries. È possibile che vi sia più di un gruppo di ripristino di cluster OpenVMS all'interno di un cluster OpenVMS. Si noti che la lunghezza massima consentita dai gruppi di ripristino di cluster OpenVMS per il nome di un Queue Manager è di 25 caratteri.

*Failover* è il processo grazie al quale un Queue Manager MQSeries viene riavviato su un altro nodo OpenVMS nel caso in cui si verifichi uno dei malfunzionamenti supportati. Terminata questa procedura, si dice che si è eseguito il fail over del Queue Manager MQSeries.

*Failback* è il processo grazie al quale il Queue Manager MQSeries viene riavviato sul suo originale nodo OpenVMS dopo la risoluzione di un malfunzionamento. L'esecuzione automatica del failback non è supportata ai gruppi di ripristino di cluster OpenVMS e deve essere eseguita manualmente. Terminata questa procedura, si dice che si è eseguito il fail back del Queue Manager MQSeries.

*failover monitor* è un processo che viene eseguito su ogni membro di un gruppo di ripristino di cluster OpenVMS. I failover monitor sono responsabili dell'esecuzione di tutte le funzioni dei gruppi di ripristino. Essi cooperano tra loro per fornire queste funzioni. Un failover monitor viene avviato con il comando **runmqfm** (per ulteriori informazioni su questo comando, consultare la sezione "runmqfm (Avvia un monitor di ripristino)" a pagina 302).

Uno dei failover monitor è denominato *watcher failover monitor* (supervisore delle attività di ripristino) e il suo stato viene denominato *watching* (supervisione). Il failover monitor che per primo si attiva in un gruppo di ripristino assume tale denominazione. Un gruppo di ripristino diventa *operativo* quando si avvia il primo failover monitor. Se si verifica un malfunzionamento al supervisore delle attività di ripristino o al nodo sul quale viene eseguito, viene automaticamente nominato un altro supervisore delle attività di ripristino. Il "watcher failover monitor" ha il compito di controllare che il Queue Manager MQSeries sia in esecuzione e che si attivino le operazioni di ripristino qualora si verifichi un malfunzionamento del tipo supportato. Ogni operazione che deve essere eseguita su un altro nodo viene inoltrata dal supervisore delle attività di ripristino al failover monitor che opera sul nodo OpenVMS interessato.

I gruppi di recupero di cluster OpenVMS vengono gestiti utilizzando il comando DCL **failover**. Il comando **failover** può essere utilizzato in qualunque nodo del gruppo di ripristino di cluster OpenVMS. Tutti i comandi sono inviati al supervisore delle attività di ripristino, che a sua volta invia il comando al failover monitor che deve elaborare quel comando; se necessario lo inoltra poi ad un altro failover monitor.

Il file di configurazione del gruppo di ripristino cluster OpenVMS contiene i dettagli del gruppo, tra cui il numero e i nomi dei nodi OpenVMS. Questo file è denominato **FAILOVER.INI** ed è memorizzato nella directory **MQS\_ROOT:[MQM.QMGRS.queuemanagename]**. È un file di testo, modificabile con un editor di testo; esso deve essere creato prima di avviare il primo failover monitor. Un file modello di configurazione denominato **FAILOVER.TEMPLATE** è contenuto nella directory **MQS\_EXAMPLES**. I parametri del file di configurazione non possono



## nozioni relative ai gruppi di ripristino di cluster OpenVMS

essere modificati dinamicamente. Perché una modifica produca effetto, occorre arrestare e poi riavviare tutti i failover monitor. Fare attenzione durante questa operazione poiché il ripristino automatico del Queue Manager MQSeries non può avvenire se non sono avviati i failover monitor.

Tutti i comandi MQSeries, eccetto i comandi **strmqm** e **endmqm**, funzionano normalmente anche su un Queue Manager MQSeries incluso in un gruppo di ripristino. I due comandi citati, se utilizzati in un gruppo di ripristino attivo, provocano un messaggio di errore. Il comando **failover** deve essere utilizzato per avviare e arrestare il Queue Manager MQSeries.

La priorità nodo OpenVMS stabilisce l'ordine dato a ciascun nodo in un gruppo di ripristino cluster OpenVMS e viene utilizzata per determinare il nodo sul quale riavviare il Queue Manager quando si verifica un malfunzionamento. Il nodo con il numero di priorità più basso ha la maggiore priorità.

L'indirizzo TCP/IP del gruppo di ripristino cluster OpenVMS è l'indirizzo TCP/IP assegnato al gruppo di ripristino. Tutti i canali relativi al Queue Manager in un gruppo di ripristino devono essere configurati in modo da specificare tale indirizzo TCP/IP nel nome di connessione. Ogni gruppo di ripristino cluster OpenVMS deve utilizzare un indirizzo TCP/IP unico. Tutti i nodi del gruppo di ripristino devono avere un'interfaccia per l'indirizzo TCP/IP nella stessa rete secondaria e l'indirizzo TCP/IP del gruppo di ripristino cluster OpenVMS deve essere in questa stessa rete secondaria.

## Preparazione alla configurazione di un gruppo di ripristino cluster OpenVMS

Le seguenti operazioni devono essere compiute prima di configurare un gruppo di ripristino cluster OpenVMS:

1. Se non esiste già, creare un Queue Manager utilizzando il comando **crtmqm**.
2. Ottenere un indirizzo TCP/IP per il gruppo di ripristino cluster OpenVMS.
3. Creare o modificare i canali in modo da utilizzare l'indirizzo TCP/IP del gruppo di ripristino.
4. Scegliere i nodi da includere nel gruppo di ripristino e le relative priorità.
5. Assicurarsi che struttura logica MQS\_ROOT faccia riferimento alla stessa directory in tutti i nodi del gruppo di ripristino OpenVMS e che il disco sia montato su tutti i nodi. I dischi che contengono la directory MQS\_ROOT e i file di log non devono utilizzare MSCP da un nodo del failover set ad un altro nodo, poiché se il nodo che serve il disco diventa non disponibile, il nodo servito dal disco non sarà più in grado di accedere al disco.

## Configurazione di un gruppo di ripristino cluster OpenVMS

Per configurare un gruppo di ripristino cluster OpenVMS è necessario effettuare le seguenti operazioni:

1. Copiare il file `MQS_EXAMPLES:FAILOVER.TEMPLATE` in `MQS_ROOT:[MQM.QMGRS.queumanagername]FAILOVER.INI`.
2. Modificare il file `MQS_ROOT:[MQM.QMGRS.queumanagername]FAILOVER.INI` in modo da configurarlo per questo gruppo di ripristino cluster OpenVMS (consultare la sezione "Modifica del file di configurazione FAILOVER.INI" a pagina 232).
3. Aprire le procedure di comando **START\_QM.COM**, **END\_QM.COM** e **TIDY\_QM.COM** (consultare la sezione "Procedure di comando utilizzate dai gruppi di ripristino" a pagina 233).



## Configurazione di gruppi di ripristino cluster OpenVMS

4. Impostare la sicurezza ICC per la Associazione ICC utilizzata dai failover monitor (consultare la sezione "Impostazione di sicurezza per associazioni ICC" a pagina 240).
5. Avviare un failover monitor su ogni nodo del gruppo di ripristino cluster OpenVMS utilizzando il comando **runmqfm -m** *queuemanagername*.
6. Avviare il Queue Manager utilizzando il comando **failover -m** *queuemanagername -n nodename -s*.

## Configurazione di gruppi di ripristino cluster OpenVMS

7. Modificare la chiusura specifica del sito in:
  - Chiudere o spostare il Queue Manger in esecuzione su un nodo quando viene arrestato.
  - Arrestare il failover monitor.

## Operazione successive alla configurazione di un gruppo di ripristino cluster OpenVMS

Le seguenti operazione possono essere compiute solo dopo la configurazione del gruppo di ripristino:

- Aprire il file di avvio di sistema per avviare i processi di failover monitor su ogni nodo del gruppo di ripristino cluster OpenVMS utilizzando il comando **runmqfm**. Posizionare il comando **runmqfm** dopo il comando per l'avvio di MQSeries.
- Se necessario, per avviare il Queue Manger automaticamente all'avvio del sistema, posizionare un comando sul nodo rilevante per avviare il Queue Manager dopo l'avvio del failover monitor. Il comando per avviare il Queue Manager su un nodo è **failover -m queuemanagername -n nodename -s**.
- Modificare la chiusura specifica del sito in modo da arrestare il failover monitor alla chiusura del sistema. Inoltre chiudere o spostare il Queue Manager in esecuzione su un nodo quando viene arrestato.

## Modifica del file di configurazione FAILOVER.INI

Il file FAILOVER.INI deve essere configurato per ogni gruppo di ripristino cluster OpenVMS. Il significato di ognuno dei campi è riportato nell'elenco della Tabella 9 a pagina 232. Il modello di file di configurazione memorizzato in MQS\_EXAMPLES è riportato nell'Appendice F. OpenVMS modelli di gruppi di ripristino di cluster" a pagina 341. Tutte le linee del file che iniziano con il carattere '#' vengono ignorate se lette durante l'esecuzione di un processo failover monitor. Il carattere nei campi del file deve essere minuscolo o maiuscolo così come specificato nel modello di file. Tutti i nomi dei campi devono essere seguiti dal carattere '=' e dal valore associato. Tutti i campi del modello di file sono obbligatori e pertanto non devono essere omessi.

Tabella 9. Descrizione dei campi nel file FAILOVER.INI

Nome del campo	Descrizione
IpAddress	L'indirizzo TCP/IP da utilizzare per il gruppo di ripristino
PortNumber	Il numero di porta TCP/IP utilizzata dal listener per il Queue Manager
TimeOut	Questo valore di time out viene inviato alla procedura EndCommand. Consultare la sezione "Procedure di comando utilizzate dai gruppi di ripristino" a pagina 233.
StartCommand	La procedura del comando utilizzato per avviare il Queue Manager
EndCommand	La procedura del comando utilizzato per arrestare il Queue Manager
TidyCommand	La procedura del comando utilizzato per riordinare un nodo dopo il verificarsi di un malfunzionamento del Queue Manager che non coinvolge il nodo OpenVMS.
LogDirectory	La directory che contiene i file di log creati dalle procedure StartCommand, EndCommand e TidyCommand

Tabella 9. Descrizione dei campi nel file FAILOVER.INI (Continua)

Nome del campo	Descrizione
NodeCount	Il numero dei nodi del gruppo di ripristino. Il numero massimo di nodi supportati è quattro.
NodeName	Il nome del nodo. Questo è il valore specificato per il parametro di sistema SCSNODE OpenVMS.
Interface	Il nome dell'interfaccia TCP/IP per il nodo se si utilizza Digital TCP/IP Services per gruppi OpenVMS TCP/IP. Questo può ricavarsi dall'output del comando \$tcpip show interface. Questo campo non viene utilizzato se si utilizza TCPware per OpenVMS TCP/IP o Multinet per gruppi OpenVMS TCP/IP, ma il valore predefinito di we0 deve comunque essere specificato (non rimuovere il campo dal file di configurazione).
Priority	La priorità data a questo nodo nel gruppo di ripristino. Il valore deve essere compreso tra 1 e 10. Il valore 1 attribuisce la massima priorità. È possibile attribuire la medesima priorità a più nodi. Quando si verifica un malfunzionamento oppure quando non viene specificato un determinato nodo in un comando <b>failover -s</b> o <b>-f</b> , il Queue Manager viene avviato dal nodo a cui è stata assegnata la priorità più elevata, purché sia disponibile.

## Procedure di comando utilizzate dai gruppi di ripristino

I gruppi di ripristino utilizzano tre procedure di comando per migliorare alcune delle loro funzioni. La posizione di queste procedure di comando sono specificate nel file di configurazione FAILOVER.INI nel campo denominato StartCommand, EndCommand e TidyCommand. I file modello per queste procedure di comando, denominati rispettivamente **START\_QM.TEMPLATE**, **END\_QM.TEMPLATE** e **TIDY\_QM.TEMPLATE**, sono memorizzati in MQS\_EXAMPLES. Questi file sono elencati nell'Appendice F. OpenVMS modelli di gruppi di ripristino di cluster" a pagina 341.

La procedura del comando elabora cinque o sei parametri. Questi parametri sono elencati nella Tabella 10 a pagina 233:

Tabella 10. Parametri elaborati dalle procedure di comando

Parametro	Valore
P1	Nome del Queue Manager
P2	Nome della directory del Queue Manager
P3	Indirizzo TCP/IP del cluster
P4	Nome dell'interfaccia del nodo
P5	numero di porta del listener
P6	Chiusura del timeout Queue Manager (solo per la procedura del comando EndCommand)

La procedura StartCommand viene utilizzata nei seguenti casi per avviare il Queue Manager:

- Quando appositamente specificato con il flag **-s** del comando **failover**
- Quando il Queue Manager viene spostato su un altro nodo OpenVMS utilizzando il flag **-f** del comando **failover**.
- Quando viene riavviato automaticamente dopo un malfunzionamento del Queue Manager

## Procedure di comando

Per impostazione predefinita la procedura StartCommand configura l'indirizzo TCP/IP del gruppo sul nodo che deve eseguire il Queue Manager e poi avvia il Queue Manager utilizzando il comando **strmqm -m** *queuemanagername*. In base ai requisiti del sistema, la procedura del comando può essere modificato in uno dei seguenti modi:

- Modificare il comando **strmqm**
- Aggiungere comandi per avviare ulteriori processi MQSeries come il listener
- Aggiungere comandi per avviare processi applicativi

La procedura StartCommand deve terminare con uno stato pari a 1 perché il Queue Manager possa essere controllato dopo il suo avvio.

La procedura EndCommand viene utilizzata nei seguenti casi per arrestare il Queue Manager:

- Quando appositamente specificato con il flag -e del comando **failover**
- Quando il Queue Manager viene spostato su un altro nodo OpenVMS utilizzando il flag -f del comando **failover**

Per impostazione predefinita la procedura EndCommand tenta di arrestare il Queue Manager con il comando **endmqm -i** *queuemanagername*. Se il Queue Manager non si arresta entro il tempo impostato nel file di configurazione, la procedura tenta di arrestare il Queue Manager con il comando **endmqm -p** *queuemanagername*. Se, trascorso un ulteriore timeout, il Queue Manager non si arresta, esso viene chiuso eliminando il processo Execution Controller. Una volta arrestato il Queue Manager, l'indirizzo TCP/IP del gruppo di ripristino non è più configurato. Se la chiusura del Queue Manager con il comando **endmqm** ha esito positivo, viene riportato lo stato SS\$\_NORMAL. Se invece la chiusura avviene mediante l'eliminazione di Execution Controller, viene riportato lo stato SS\$\_ABORT. Se infine il Queue Manager non viene chiuso dopo il terzo timeout, viene riportato lo stato SS\$\_TIMEOUT. Queste informazioni di stato vengono utilizzate dal supervisore delle attività di ripristino per stabilire il risultato della procedura EndCommand ed impostare conseguentemente lo stato del gruppo di ripristino. In base ai requisiti del sistema, la procedura del comando può essere modificato in uno dei seguenti modi:

- Aggiungere comandi per arrestare ulteriori processi MQSeries come i listener
- Aggiungere comandi per arrestare processi applicativi

La procedura TidyCommand viene utilizzata per riorganizzare un nodo OpenVMS dopo il verificarsi di un malfunzionamento del Queue Manager, nel caso in cui il nodo OpenVMS sia ancora in esecuzione.

Per impostazione predefinita la procedura TidyCommand elimina dalla configurazione l'indirizzo TCP/IP del gruppo di ripristino. In base ai requisiti del sistema, la procedura del comando può essere modificato in uno dei seguenti modi:

- Aggiungere comandi per arrestare qualsiasi processo MQSeries ancora in esecuzione come il listener
- Aggiungere comandi per arrestare processi applicativi ancora in esecuzione

I file modello per impostazione predefinita utilizzano i comandi Digital TCP/IP Services per OpenVMS per configurare e de-configurare l'indirizzo TCP/IP. Se si utilizza TCPware per OpenVMS o MultiNet per OpenVMS, disattivare i comandi Digital TCP/IP Services per OpenVMS e attivare i comandi TCPware per OpenVMS o MultiNet per OpenVMS.

## Gestione dei gruppi di ripristino

I gruppi di ripristino devono essere gestiti da un SYSTEM account o da un account MQSeries Administration. La gestione avviene utilizzando i due comandi DCL **runmqfm** e **failover**. Il comando **runmqfm** viene utilizzato per avviare i controllori di **ripristino** e il comando **failover** esegue tutte le altre attività di gestione. Questi comandi sono descritti nelle sezioni “runmqfm (Avvia un monitor di ripristino)” a pagina 302 e “failover (Gestione di un gruppo di ripristino)” a pagina 287.

### Avvio di failover monitor

Per avviare i failover monitor eseguire il comando **runmqfm** sul nodo in cui si desidera avviare il failover monitor. Ad esempio per avviare un failover monitor per il Queue Manager TESTQM, utilizzare il comando:

```
$ runmqfm -m TESTQM
```

In questo modo viene creato un processo separato il cui nome è basato sul nome del Queue Manager e termina in **\_FM**. In questo esempio il nome del processo è **TESTQM\_FM**. Questo processo è elencato in un active display **monmq**.

Se è necessario un file di log, lo si può specificare reindirizzando l’output del comando **runmqfm** e specificando il flag **-d** è possibile visualizzare ulteriori informazioni di debug nel file di log. Ad esempio:

```
$ runmqm -m TESTQM -d > sys$manager:fm.log
```

Si noti che il comando **runmqfm** avvia solo il failover monitor e non anche il Queue Manager.

### Avvio di un Queue Manager in un gruppo di ripristino

Per avviare un Queue Manager in un gruppo di ripristino, è necessario che sia in esecuzione almeno un failover monitor e che ve ne sia uno in esecuzione sul nodo in cui si desidera avviare il Queue Manager. Utilizzare il flag **-s** del comando **failover** per avviare il Queue Manager. Il comando può essere eseguito da qualsiasi nodo OpenVMS del gruppo di ripristino. Ad esempio, se si desidera avviare il Queue Manager TESTQM sul nodo BATMAN, utilizzare il seguente comando:

```
$ failover -m TESTQM -n BATMAN -s
```

Se è necessario avviare il Queue Manager sul nodo OpenVMS con la maggiore priorità disponibile, omettere il flag **-n** dal comando. Ad esempio:

```
$ failover -m TESTQM -s
```

Si noti che dopo l’avvio del failover monitor relativo ad un Queue Manager (su qualsiasi nodo), non sarà possibile avviare il Queue Manager con il comando **strmqm**. Tuttavia, dopo l’arresto di tutti i failover monitor relativi ad un Queue Manager, sarà di nuovo possibile utilizzare il comando **strmqm**.

## arresto gruppo di ripristino

### Arresto di un Queue Manager in un gruppo di ripristino

Per arrestare un Queue Manager in un gruppo di ripristino, è necessario che vi sia un failover monitor in esecuzione sul nodo in cui è in esecuzione il Queue Manager. Utilizzare il flag **-e** del comando **failover** per arrestare il Queue Manager. Il comando può essere eseguito da qualsiasi nodo OpenVMS del gruppo di ripristino. Ad esempio se si desidera arrestare il Queue Manager TESTQM, utilizzare il comando:

```
$ failover -m TESTQM -e
```

Si noti che dopo l'avvio del failover monitor relativo ad un Queue Manager (su qualsiasi nodo), non sarà possibile arrestare il Queue Manager con il comando **endmqm**. Tuttavia, dopo l'arresto di tutti failover monitor relativi ad un Queue Manager, sarà di nuovo possibile utilizzare il comando **endmqm**.

### Spostamento di un Queue Manager in un gruppo di ripristino

Spostare un Queue Manager in un gruppo di ripristino significa arrestarne l'esecuzione in corso su un nodo e riavviarla su un altro nodo del gruppo di ripristino. Per spostare il Queue Manager all'interno di un gruppo di ripristino, è necessario che sia in esecuzione un failover monitor sul nodo in cui è attualmente in esecuzione il Queue Manager e che sia in esecuzione anche un failover monitor sul nodo in cui si desidera spostare il Queue Manager.

Utilizzare il flag **-f** del comando **failover** per spostare un queue Manager. Il comando può essere eseguito da qualsiasi nodo OpenVMS del gruppo di ripristino. Ad esempio, se si desidera spostare il Queue Manager TESTQM nel nodo ROBIN, utilizzare il seguente comando:

```
$ failover -m TESTQM -n ROBIN -f
```

Se è necessario spostare il Queue Manager nel nodo OpenVMS con la maggiore priorità disponibile, omettere il flag **-n** dal comando. Ad esempio:

```
$ failover -m TESTQM -f
```

### Visualizzazione dello stato di un gruppo di ripristino

Esistono tre tipi di stato che descrivono lo stato complessivo del gruppo di ripristino:

- Lo stato dei Queue Manager del gruppo di ripristino
- Lo stato dei Queue Manager su ciascun nodo del gruppo di ripristino (uno per ogni nodo)
- Lo stato dei Node Monitor del gruppo di ripristino (uno per ogni nodo)

I valori possibili di ogni stato sono descritti nelle seguenti tre tabelle.

Tabella 11. Stato dei Queue Manager del gruppo di ripristino

Stato	Descrizione
STOPPED	Il Queue Manager non è mai stato avviato nel gruppo di ripristino oppure è stato correttamente arrestato.

## Spostamento di un Queue Manager

Tabella 11. Stato dei Queue Manager del gruppo di ripristino (Continua)

Stato	Descrizione
STARTED	Il Queue Manager è stato avviato nel gruppo di ripristino. Il gruppo di ripristino tenterà di riavviare il Queue Manager se si dovesse verificare un malfunzionamento.

Tabella 12. Stato dei Queue Manager su ciascun nodo del gruppo di ripristino

Stato	Descrizione
AVAILABLE	Il nodo è disponibile all'avvio del Queue Manager se dovesse verificarsi un malfunzionamento su un altro nodo.
RUNNING	Il Queue Manager è in esecuzione su questo nodo.
EXCLUDED	Il Queue Manager è stato arrestato in modo non corretto su questo nodo senza tuttavia causare il malfunzionamento del nodo. Se si verifica un malfunzionamento di un Queue Manager su un altro nodo, il Queue Manager non verrà riavviato su questo.

Tabella 13. Stato dei Node Monitor del gruppo di ripristino

Stato	Descrizione
STARTED	Un failover monitor è in esecuzione su questo nodo, ma non è il supervisore.
WATCHING	Su questo nodo è in esecuzione il failover monitor supervisore.
STOPPED	Non vi sono failover monitor in esecuzione su questo nodo.

Utilizzare il flag `-q` del comando **failover** per visualizzare lo stato del gruppo di ripristino. È necessario che vi sia almeno un processo failover monitor in esecuzione; il comando può essere eseguito da qualsiasi nodo del gruppo di ripristino. Ad esempio, per visualizzare lo stato del gruppo di ripristino relativo al Queue Manager TESTQM, utilizzare il seguente comando:

```
$ failover -m TESTQM -q
```

Di seguito è riportato un esempio di output del comando:

## Impostazione simboli DCL

```
83H8439, 5697-270 (C) Copyright IBM Corp. 1996. ALL RIGHTS RESERVED.

OpenVMS Cluster Failover Set - Configuration and State.

Queue Manager Name           : TESTQM
Sequence No                   : 11
TCP/IP Address                 : 10.20.30.40
Listener Port Number          : 1414
Timeout to end the Queue Manager : 30
Queue Manager state in Failover Set : STARTED

OpenVMS Node - Configuration and State

Node name                      : BATMAN
Priority                        : 2
TCP/IP Interface               : we0
Queue Manager state            : RUNNING
Failover Monitor state         : WATCHING

Node name                      : ROBIN
Priority                        : 1
TCP/IP Interface               : we0
Queue Manager state            : EXCLUDED
Failover Monitor state         : STARTED
```

## Impostazione dei simboli DCL allo stato di un gruppo di ripristino

In alcuni casi è necessario scrivere le procedure di comando DCL per controllare i gruppi di ripristino. Il flag `-l` del comando **failover** imposta tre simboli locali DCL che indicano lo stato del gruppo di ripristino. Questi simboli possono essere utilizzati per eseguire azioni in base allo stato del Queue Manager. È necessario che vi sia almeno un processo failover monitor in esecuzione; il comando può essere eseguito da qualsiasi nodo del gruppo di ripristino. I simboli impostati sono descritti nella Tabella 14.

Tabella 14. simboli DCL e relativa descrizione

Nome del simbolo DCL	Descrizione
MQS\$QMGR_NODE	Impostazione per il nodo OpenVMS che esegue il Queue Manager; se non vi è alcun Queue Manager in esecuzione la stringa è nulla.
MQS\$AVAILABLE_NODES	Impostazione per l'elenco dei nodi OpenVMS su cui è possibile eseguire il Queue Manager. Tali sono i nodi che visualizzano lo stato AVAILABLE e quelli su cui è in esecuzione un failover monitor.
MQS\$MONITOR_NODES	Impostazione per l'elenco dei nodi OpenVMS su cui è in esecuzione un failover monitor.

Ad esempio, per impostare i simboli allo stato del gruppo di ripristino relativo al Queue Manager TESTQM, utilizzare il seguente comando:

```
$ failover -m TESTQM -l
```



Di seguito viene riportato un esempio che illustra i risultati dell'impostazione dei simboli:

```
MQS$AVAILABLE_NODES = ""
MQS$MONITOR_NODES = "BATMAN,ROBIN"
MQS$QMGR_NODE = "BATMAN"
```

### Arresto di un processo failover monitor

Il processo failover monitor su un nodo OpenVMS può essere arrestato utilizzando il flag -h del comando **failover**. Il comando può essere eseguito da qualsiasi nodo del gruppo di ripristino. Ad esempio, per arrestare il failover monitor relativo al Queue Manager TESTQM sul nodo BATMAN utilizzare il comando:

```
$ failover -m TESTQM -n BATMAN -h
```

Se si arresta il failover monitor supervisore ed esiste un altro failover monitor, quest'ultimo diverrà quello supervisore. Il gruppo di ripristino non sarà più attivo se si arresta il suo ultimo failover monitor. In questo caso il Queue Manager potrà essere avviato ed arrestato utilizzando i comandi **strmqm** e **endmqm**. Il flag -h del comando **failover** non è in grado di arrestare il Queue Manager. Il Queue Manager rimarrà in esecuzione, qualora venga arrestato il failover monitor del nodo OpenVMS sul quale si sta eseguendo il Queue Manager.

### Esecuzione di comandi mentre è in corso un aggiornamento

I comandi **failover** e i flag -s, -e, -f e -c sono considerati aggiornamenti. Durante l'elaborazione di questi comandi, il failover monitor supervisore imposta un flag di aggiornamento in corso. Quando viene impostato questo flag, ogni altro aggiornamento o comando non avranno effetto, poiché non sono supportati gli aggiornamenti simultanei. Invece, i comandi che non integrano aggiornamenti, quali i flag -q e -l, continuano a produrre i loro effetti durante l'elaborazione degli aggiornamenti.

In casi rari, se un aggiornamento ha esito negativo potrebbe lasciare impostata la flag di aggiornamento in corso. Il flag -u del comando **failover** elimina il flag di aggiornamento in corso. Utilizzare con cautela questo comando. Ad esempio, per eliminare il flag di aggiornamento in corso relativa al Queue Manager TESTQM, utilizzare il seguente comando:

```
$ failover -m TESTQM -u
```

### Modifica dello stato di un gruppo di ripristino

In alcuni casi potrebbe essere necessario modificare lo stato di un gruppo di ripristino. A questo scopo può essere utilizzato il flag -c del comando **failover**. Il caso più frequente si ha quando un Queue Manager su un nodo visualizza lo stato EXCLUDED dopo un malfunzionamento e si desidera riportare lo stato AVAILABLE dopo il ripristino del normale funzionamento del nodo. Ad esempio, per modificare in AVAILABLE lo stato del Queue Manager TESTQM sul nodo BATMAN, utilizzare il seguente comando:

## Modifica dello stato di un gruppo di ripristino

```
$ failover -m TESTQM -n BATMAN -c -qmgr available
```

Potrebbe darsi anche il caso che si desideri escludere la possibilità che su un nodo venga eseguito il Queue Manager; in questo caso è necessario modificare lo stato del nodo per il Queue Manager da AVAILABLE in EXCLUDED. Ad esempio, per modificare in EXCLUDED lo stato del Queue Manager TESTQM sul nodo BATMAN, utilizzare il seguente comando:

```
$ failover -m TESTQM - n BATMAN -c -qmgr excluded
```

Inoltre è possibile modificare anche tutti gli altri stati, ma ogni modifica produce effetto solo se la modifica richiesta è compatibile con il sistema in esecuzione. Ad esempio, se un failover monitor è in esecuzione su un nodo e si tenta di modificarne lo stato in STOPPED, tale modifica non avrà effetto. Esclusa l'ipotesi della modifica dallo stato EXCLUDED in AVAILABLE, potrebbe non essere necessario utilizzare il comando di modifica dello stato, poichè il failover monitor supervisore esegue un controllo dell'integrità ogni 30 secondi, effettuando le modifiche di stato qualora rilevi divari con il sistema in esecuzione.

## Impostazione di sicurezza per associazioni ICC

I failover monitor del gruppo di ripristino e i programmi client utilizzano le chiamate OpenVMS ICC (Intra Cluster Communication) per il trasferimento dei messaggi. Per impedire ad utenti non autorizzati l'invio di messaggi al processo di failover monitor, è possibile impostare la sicurezza per le Associazioni ICC nella procedura del comando **SYS\$STARTUP:ICC\$SYSTARTUP.COM** .

Ogni gruppo di ripristino utilizza due nomi di associazioni: il primo ha il nome del Queue Manager utilizzato per la comunicazione con il failover monitor supervisore; l'altro ha il nome del Queue Manager con l'aggiunta di **\_MQ\_FM**, utilizzato per la comunicazione con ciascun failover monitor.

In Figura 22 a pagina 241 è riportato un esempio delle voci da immettere in **ICC\$SYSTARTUP.COM** per ciascun nodo del gruppo di ripristino. Vi sono due nodi nel gruppo di ripristino denominati BATMAN e ROBIN; il Queue Manager è denominato TESTQM.

```

$! ----- List Nodes with Special Actions -----
$!
$ nodeactions = "/BATMAN/ROBIN/"
$ if f$locate("/"+nodename+"/",nodeactions) .eq. f$length(nodeactions) -
then goto exit ! No action for this node
$ goto 'nodename' ! Go to action code for this node
$!
$! ----- Major Nodes -----
$BATMAN:
$ROBIN:
$!
$!
$! Place in here calls to @SYS$MANAGER:ICC$CREATE_SECURITY_OBJECT and
$! @SYS$MANAGER:ICC$ADD_REGISTRY_TABLE that apply to FAILOVER nodes in the
$! cluster
$!
$!
$ @SYS$MANAGER:ICC$CREATE_SECURITY_OBJECT ICC$::"TESTQM" -
"/owner=MQM/acl=((id=MQM,access=open+access),(id=*,access=none))"
$!
$ @SYS$MANAGER:ICC$CREATE_SECURITY_OBJECT 'nodename'::"TESTQM" -
"/owner=MQM/acl=((id=MQM,access=open+access),(id=*,access=none))"
$!
$ @SYS$MANAGER:ICC$CREATE_SECURITY_OBJECT 'nodename'::"TESTQM_MQ_FM" -
"/owner=MQM/acl=((id=MQM,access=open+access),(id=*,access=none))"
$!
$ set security/class=logicial_name_table icc$registry_table -
/acl=(id=MQM,access=read+write)
$!
$ GOTO EXIT
$!

```

Figura 22. Esempio di voci da immettere in \$SYSTARTUP.COM

Si noti che il nome delle Associazioni ICC può avere la lunghezza massima di 31 caratteri, pertanto la lunghezza massima supportata per il nome del Queue Manager MQSeries è di 25 caratteri se utilizzato in un gruppo di ripristino. Ulteriori informazioni sull'impostazione della sicurezza per Associazioni ICC sono disponibili nel manuale di *OpenVMS System Manager*.

## Risoluzione dei problemi con gruppi di ripristino

Quando si eseguono le procedure **start\_qm.com**, **end\_qm.com** e **tidy\_qm.com**, un file di log viene creato nella directory di log specificata nel file di configurazione **failover.ini**. I nomi dei file di log sono *qmgrname\_procedurename.log*. Ad esempio, per un Queue Manager il cui nome è TESTQM la procedura del comando **start\_qm.com** compilerà un file di log denominato **testqm\_start\_qm.log**.

Per impostazione predefinita il failover monitor non produce un file di log, ma è possibile specificarne uno utilizzando i parametri di reindirizzamento del comando **runmqfm**. Ulteriori informazioni di debug possono essere aggiunte al file specificando il parametro **-d** nel comando **runmqfm**.

Controllare se sono stati creati file FDC in **MQS\_ROOT:[MQM.ERRORS]**.

## Utilizzo di MultiNet per OpenVMS con gruppi di ripristino

Per utilizzare Multinet per OpenVMS con gruppi di ripristino, il cluster alias service deve essere abilitato. Per abilitarlo utilizzare il comando:

## Utilizzo di MultiNet

```
$ MULTINET CONFIGURE/SERVERS
SERVER-CONFIG> ENABLE CLUSTERALIAS
SERVER-CONFIG> EXIT
```

I modelli di file comando presuppongono che ci sia un solo indirizzo per cluster alias, utilizzato dal gruppo di ripristino. Tuttavia, laddove si utilizzino altri indirizzi cluster alias, si dovranno modificare le procedure di comando in modo che gli altri indirizzi rimangano nel nome logico MULTINET\_CLUSTER\_IP\_ALIASES.

## Un esempio di utilizzo di gruppi di ripristino

Di seguito viene riportato un esempio di configurazione di due nodi, BATMAN e ROBIN, in un cluster OpenVMS di un gruppo di ripristino per il Queue Manager TESTQM. L'indirizzo TCP/IP del gruppo di ripristino è 10.20.30.40 e la porta TCP/IP del listener è 1414. Il nodo primario è BATMAN; quello secondario è ROBIN. Inizialmente il Queue Manager verrà avviato su BATMAN; se si verifica un malfunzionamento verrà riavviato su ROBIN. Al riavvio di ROBIN, non sarà possibile eseguire il fail back del Queue Manager su BATMAN, se il Queue Manager era in esecuzione su ROBIN. Se il nodo che esegue il Queue Manager viene arrestato, il Queue Manager viene chiuso e il failover monitor interrotto. Se il nodo non è in esecuzione, il Queue Manager non viene chiuso, ma viene interrotto solo il failover monitor.

### Personalizzazione di failover.template

Il file failover.template viene modificato come segue e copiato  
comemqs\_root:[mqm.qmgrs.testqm] failover.ini .

```

# FAILOVER.TEMPLATE
# Template for creating a FAILOVER.INI configuration file
# All lines beginning with a '#' are treated as comments
#
# OpenVMS Cluster Failover Set Configuration information
# -----
#
# The TCP/IP address used by the OpenVMS Cluster Failover Set
#
IpAddress=10.20.30.40
#
# The TCP/IP port number used by the MQSeries Queue Manager
#
PortNumber=1414
#
# The timeout used by the EndCommand command procedure
#
TimeOut=30
#
# The command procedure used to start the Queue Manager
#
StartCommand=@sys$manager:start_qm
#
# The command procedure used to end the Queue Manager
#
EndCommand=@sys$manager:end_qm
#
# The command procedure used to tidy up on a node after a
# Queue Manager failure but the OpenVMS node did not fail
#
TidyCommand=@sys$manager:tidy_qm
#
# The directory in which the log files for the start, end and
# tidy commands are written
#
LogDirectory=mqs_root:[mqm.errors]
#
# The number of nodes in the OpenVMS Cluster Failover Set. The
# number of nodes defined below must agree with this number
#
NodeCount=2
#
# The Name of the OpenVMS node
#
NodeName=BATMAN
#
# The TCP/IP interface name for the node
#Interface=we0
#
# The priority of the node
#
Priority=1
#
# The Name of the OpenVMS node
#
NodeName=ROBIN
#
# The TCP/IP interface name for the node
#
Interface=we0
#
# The priority of the node
#
Priority=2

```

Figura 23. Failover.template per creare un file di configurazione FAILOVER.INI

## Esempio di gruppo di ripristino

### Modifica delle procedure di comando di un gruppo di ripristino

Le procedure di comando vengono copiate dai file modello. Le uniche differenze sono relative all'attivazione dell'avvio del listener in **start\_qm.com** e alla chiusura del listener in **end\_qm.com**. Se vi sono applicazioni da arrestare o avviare, potrebbe essere necessario aggiungere i comandi appropriati alle procedure di comando.

### Esempio di procedura di comando di avvio del gruppo di ripristino, **start\_failover\_set.com**

La procedura di comando **start\_failover\_set.com** viene utilizzata per avviare il failover monitor su ogni nodo e di conseguenza anche il Queue Manager. Dopo l'esecuzione della procedura di comando **MQS\_STARTUP.COM** la procedura viene richiamata all'avvio del sistema. La procedura riporta due parametri: il nome del Queue Manager e quello del nodo primario. In questo caso viene richiamata nel seguente modo:

```
$@start_failover_set testqm batman
```

La procedura di comando **start\_failover\_set.com** avvia il failover monitor e quindi utilizza il parametro **-l** del comando **failover** per rilevare lo stato del gruppo di ripristino. si noti che il failover monitor potrebbe non aver completato il suo avvio al momento dell'esecuzione del comando **failover**; in questo caso il comando viene ripetuto tre volte con un secondo di intervallo tra ogni tentativo. Quindi, se il nodo è quello primario e il Queue Manager non è avviato, verrà avviato utilizzando il parametro **-s** del comando **failover**.

## Esempio di gruppo di ripristino

```
$on error then exit
$@sy$manager:mqs_symbols
$!
$! start_failover_set.com
$! -----
$! Command procedure to start a Failover Set Queue Manager during startup
$!
$! p1 = Queue Manager name
$! p2 = Primary Node name
$!
$! Check that the Queue Manager has been specified
$!
$if p1 .eqs ""
$then
$ Write sys$output "Queue Manager name omitted"
$ exit
$else
$ qmgr_name = p1
$endif
$!
$! Check that the primary node name has been specified
$!
$if p2 .eqs ""
$then
$ Write sys$output "Node name omitted"
$ exit
$else
$ primary_node = p2
$endif
$!
$! Get the node name of this node
$!
$this_node=f$getsyi("nodename")
$!
$! Start the Failover Monitor on this node
$!
$runmqfm -m 'qmgr_name'
$!
$! Check that the Failover Monitor has fully started
$! Wait up to 3 seconds
$!
$count = 0
$check_start:
$on error then continue
$!
$! Set the MQS$* symbols to the state of Failover Set
$! Wait up to 3 seconds
$!
```

Figura 24. procedura di comando start\_failover\_set (Numero 1 di 2)

## Esempio di gruppo di ripristino

```
$failover -m 'qmgr_name' -l
$!
$! If an error is returned wait a second and try again
$!
$if ( ($status/8) .and %xffff ) .ne. 0 then goto wait
$!
$! If this node is not listed as running a monitor wait a second and try again
$!
$if f$locate( this_node, mqs$monitor_nodes ) .ne. f$length( mqs$monitor_nodes )
$then
$ goto start_qm
$endif
$wait:
$on error then exit
$count = count + 1
$!
$! If we have waited 3 seconds display an error and exit
$!
$if count .ge. 3
$then
$ write sys$output "Failover Monitor not started"
$ exit
$else
$ wait 00:00:01
$ goto check_start
$endif
$start_qm:
$!
$! Only start the Queue Manager on the primary node
$!
$if this_node .nes. primary_node
$then
$ write sys$output "Queue Manager not started on Secondary node"
$ exit
$endif
$!
$! Start the Queue Manager on the primary node if it is not already running.
$!
$if mqs$qmgr_node .eqs. ""
$then
$ failover -m 'qmgr_name' -n 'this_node' -s
$else
write sys$output "Queue Manager already started"
$endif
$exit
```

Figura 24. procedura di comando `start_failover_set` (Numero 2 di 2)

## Esempio di procedura di comando di arresto di un gruppo di ripristino, `end_failover_set.com`

La procedura di comando `end_failover_set.com` per arrestare il Queue Manager e il failover monitor in successione su ogni nodo. Prima che venga eseguita la procedura di comando `MQS_SHUTDOWN.COM`, la procedura viene richiamata dal sito che si deve chiudere. La procedura riporta un solo parametro: il nome del Queue Manager. In questo caso è richiamata nel seguente modo:

```
$@start_failover_set testqm
```

La procedura di comando `end_failover_set.com` rileva lo stato del gruppo di ripristino utilizzando il parametro `-l` del comando `failover`. Quindi se il Queue



## Esempio di gruppo di ripristino

Manager è in esecuzione su questo nodo, viene arrestato. Di conseguenza il failover monitor viene interrotto.

```
on error then exit
$@sys$manager:mqs_symbols
$!
$! end_failover_set.com
$! -----
$! Command procedure to end a Failover Set Queue Manager during shutdown
$!
$! p1 = Queue Manager name
$!
$! Check that the Queue Manager has been specified
$!
$if p1 .eqs ""
$then
$ Write sys$output "Queue Manager name omitted"
$ exit
$else
$ qmgr_name = p1
$endif
$!
$! Get the node name of this node
$!
$this_node=f$getsyi("nodename")
$!
$! Set the MQS$* symbols to the state of the Failover Set
$!
$failover -m 'qmgr_name' -l
$!
$! If an error then exit
$!
$if ( ($status/8) .and %xffff ) .ne. 0
$then
$ write sys$output "Error querying Failover Set"
$ exit
$endif
$!
$! If the Queue Manager is not running on this node then exit
$!
$ if mqs$qmgr_node .nes. this_node
$then
$ write sys$output "Queue Manager not running on this node"
$ goto halt_fm
$endif
$!
$! End the Queue Manager
$!
$failover -m gjtest -e
$halt_fm:
$!
$! Halt the Failover Monitor
$!
$failover -m gjtest -n 'this_node' -h
```

Figura 25. procedura di comando `end_failover_set`



---

## Parte 2. Riferimenti

<b>Capitolo 17. Comandi di controllo di MQSeries</b>	<b>251</b>
Regole per la denominazione degli oggetti di MQSeries . . . . .	251
Analisi dei file di oggetto . . . . .	251
Come leggere i diagrammi di sintassi . . . . .	252
Guida alla sintassi . . . . .	253
Esempi . . . . .	254
Codici di errore di MQSeries . . . . .	255
crtmqcvx (Conversione dei dati) . . . . .	256
crtmqm (Crea Queue Manager) . . . . .	258
dltmqm (Delete queue manager) . . . . .	263
dmpmqlog (Dump log) . . . . .	266
dspmqaout (Display Authority) . . . . .	268
dspmqcsv (Display Command Server) . . . . .	272
dspmqlfs (Display MQSeries files) . . . . .	273
dspmqrtrc (Visualizza output di traccia formattato di MQSeries) . . . . .	275
dspmqrtrn (Display MQSeries transactions) . . . . .	277
endmqcsvg (End command server) . . . . .	279
endmqslr (End listener) . . . . .	282
endmqm (End queue manager) . . . . .	283
endmqtrc (End MQSeries trace) . . . . .	286
failover (Gestione di un gruppo di ripristino) . . . . .	287
rctmqimg (Registra immagine supporto) . . . . .	291
rcrmqobj (Ricrea oggetto) . . . . .	293
rsvmqtrn (risolve transazioni MQSeries) . . . . .	296
runmqchi (Run channel initiator) . . . . .	298
runmqchl (Run channel) . . . . .	299
runmqdlq (Run dead-letter queue handler) . . . . .	300
runmqfm (Avvia un monitor di ripristino) . . . . .	302
runmqslr (Run listener) . . . . .	303
runmqsc (Esegue comandi di MQSeries) . . . . .	305
runmqtmc (Avvia trigger monitor del client) . . . . .	308
runmqtrm (Avvia un trigger monitor) . . . . .	309
setmqaut (Imposta/reimposta l'autorizzazione) . . . . .	310
strmqcsvg (Avvia server di comandi) . . . . .	317
strmqm (Avvio Queue Manager) . . . . .	318
strmqtrc (Avvio traccia di MQSeries) . . . . .	320



---

## Capitolo 17. Comandi di controllo di MQSeries

Questo capitolo contiene materiale di riferimento per i comandi di controllo utilizzati con MQSeries per Compaq OpenVMS. Tutti i comandi in questo capitolo possono essere emessi da un prompt DCL di OpenVMS.

I nomi dei comandi e i relativi flag non sono sensibili al maiuscolo/minuscolo: è possibile immetterli in maiuscolo, minuscolo o in una combinazione di entrambi. Tuttavia, i parametri per controllare i comandi (come ad esempio i nomi di code) possono essere sensibili al maiuscolo/minuscolo. Consultare la sezione "Sensibilità al minuscolo/maiuscolo nei comandi di controllo" a pagina 23 per ulteriori informazioni.

Prima di utilizzare qualsiasi comando di controllo, occorre aver eseguito `mqs_startup` una volta dall'ultimo riavvio.

---

### Regole per la denominazione degli oggetti di MQSeries

In generale, i nomi degli oggetti di MQSeries possono contenere fino a 48 caratteri. Questa regola si applica a tutti i seguenti oggetti:

- Queue manager (tuttavia, se il Queue Manager è supportato da un insieme di ripristino cluster di OpenVMS, la lunghezza massima è di 25 caratteri).
- Code
- Definizioni di processo
- Namelist
- Cluster

La lunghezza massima dei nomi di canale è di 20 caratteri.

I caratteri che è possibile impiegare per tutti i nomi di MQSeries sono:

- Lettere maiuscole dalla A alla Z
- Lettere minuscole dalla a alla z
- Cifre da 0 a 9
- Punto (.)
- Sottolineatura (\_)
- Barra (/) (vedere nota 1)
- Simbolo di percentuale (%) (vedere nota 1)

#### Note:

1. La barra e il simbolo di percentuale sono caratteri speciali. Tuttavia, non è possibile utilizzare la barra e il simbolo di percentuale come primo carattere di un nome. Se si utilizza uno di questi caratteri in un nome, questo dovrà essere racchiuso fra virgolette ogni qual volta viene utilizzato.
2. Non sono consentiti spazi vuoti iniziali o integrati.
3. Non sono consentiti caratteri di lingua nazionale.
4. È possibile racchiudere i nomi fra virgolette, ma ciò è fondamentale solo qualora nel nome siano inclusi caratteri speciali oppure fosse necessario mantenere il maiuscolo/minuscolo.

### Analisi dei file di oggetto

Ciascun oggetto coda, gestore coda o processo di MQSeries è rappresentato da un file. Dal momento che non tutti nomi oggetto sono necessariamente nomi di file

## Nomi

validi, il Queue Manager converte il nome oggetto in un nome file valido, se necessario. Questa procedura è descritta nella sezione "Analisi dei nomi di file MQSeries" a pagina 21.

Per scoprire come visualizzare il nome file effettivo di un oggetto, consultare la sezione "dspmqfls (Display MQSeries files)" a pagina 273.

---

## Come leggere i diagrammi di sintassi

Questo capitolo contiene diagrammi di sintassi (talvolta denominati diagrammi "railroad").

Tutti i diagrammi di sintassi iniziano con una doppia freccia a destra e terminano con una coppia di frecce a destra e sinistra. Le righe che iniziano con un'unica freccia a destra sono righe di continuazione. Il diagramma di sintassi va letto da sinistra a destra e dall'alto verso il basso, seguendo la direzione delle frecce.

Altre convenzioni utilizzate nei diagrammi di sintassi sono:

Tabella 15. Come leggere i diagrammi di sintassi

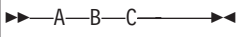
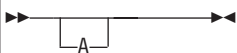

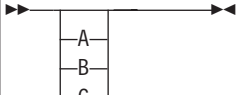
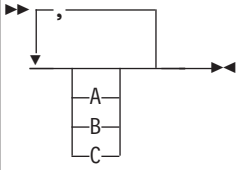
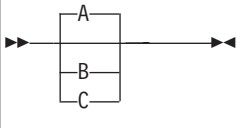
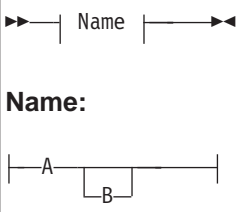
Convenzione	Significato
	Occorre specificare i valori A, B e C. I valori richiesti sono mostrati sulla riga principale di un diagramma di sintassi.
	È possibile specificare il valore A. I valori facoltativi sono illustrati sotto la riga principale di un diagramma di sintassi.
	I valori A, B e C sono alternativi; occorre specificarne almeno uno.
	I valori A, B e C sono alternativi; è possibile specificarne almeno uno.

Tabella 15. Come leggere i diagrammi di sintassi (Continua)

Convenzione	Significato
	<p>È possibile specificare un o più valori A, B e C. Qualsiasi separatore necessario per valori multipli o ripetuti (in questo esempio, la virgola (,)) è mostrato sulla freccia.</p>
	<p>I valori A, B e C sono alternativi; è possibile specificarne almeno uno. Se non si specifica nessuno dei valori mostrati, verrà utilizzato il valore predefinito A (il valore mostrato sopra la riga principale).</p>
	<p>Il frammento di sintassi Name è mostrato separatamente dal diagramma di sintassi principale.</p>

## Guida alla sintassi

È possibile ottenere la guida per la sintassi di tutti i comandi di questo capitolo immettendo il comando seguito da un punto interrogativo. MQSeries risponde elencando la sintassi richiesta corrispondente al comando selezionato. La sintassi illustra tutti i parametri e le variabili associati al comando. Vengono utilizzate diverse forme di parentesi per indicare se un parametro è necessario o no. Ad esempio:

```
CmdName [-x OptParam ] ( -c | -b ) { -p principal } argument
```

in cui:

**CmdName**

È il nome del comando per cui è stata richiesta la guida.

**[-x OptParam ]**

Le parentesi quadre indicano che si tratta di un parametro facoltativo.

**( -c | -b )**

Un campo obbligatorio. In questo caso, è necessario selezionare uno dei flag c o b.

**{ -p principal }**

Un elenco facoltativo di variabili che è possibile fornire, ma, se mostrato, è necessario fornire almeno una variabile quando si immette il comando.

## guida alla sintassi

### argomento

Un argomento necessario da fornire con questo comando, obbligatorio se mostrato nella risposta alla query.

## Esempi

1. Risultato dell'immissione di endmqm ?

```
endmqm [-z] [-c | -i | -p] QMgrName
```

2. Risultato dell'immissione di rcdmqimg ?

```
rcdmqimg [-z] [-m QMgrName] -t ObjType [GenericObjName]
```



## Codici di errore di MQSeries

La maggior parte dei comandi di MQSeries, ad esempio **crtmqm**, scrive una riga di stato al termine per indicare l'esito positivo o negativo del comando.

Se lo stato di un comando deve essere verificato in un file di comandi DCL, potrebbe essere necessario interpretare il valore di stato restituito da un programma MQSeries.

I codici di errore di MQSeries sono definiti in un file di messaggi denominato `SYS$MESSAGE:MQS_MSG.EXE`.

Per accedere al testo del messaggio associato al codice di errore nel file, è *necessario* utilizzare il comando DCL SET MESSAGE. Questo comando carica i codici del messaggio nella tabella del messaggio relativa al processo. Ad esempio:

```
$ SET MESSAGE SYS$MESSAGE:MQS_MSG.EXE
```

Al termine, sarà possibile utilizzare la funzione lessicale F\$MESSAGE per stampare il testo di un codice di errore di MQSeries. Ad esempio:

```
$ strmqm)(*bad-qm-name&%$#
Il nome del Queue Manager non è valido o è sconosciuto
$ WRITE SYS$OUTPUT F$MESSAGE($STATUS)
%MQS-F-CSPRC_Q_MGR_NAM, errore del nome del Queue Manager
```

Per convertire il codice di errore di OpenVMS in un valore di codice di errore utilizzato in MQSeries per sistemi OS/2 o UNIX, sarà possibile utilizzare la seguente equazione DCL:

```
$ RC = $STATUS / 8 .AND. %xFFFF
```

Ad esempio:

```
$ crtmqm &*)*(
Il nome del Queue Manager non è valido o è sconosciuto
$ RC = $STATUS / 8 .AND. %xFFFF
$ SHOW SYMBOL RC
RC = 72 Hex = 00000048 Octal = 0000000110
```

---

## crtmqcvx (Conversione dei dati)

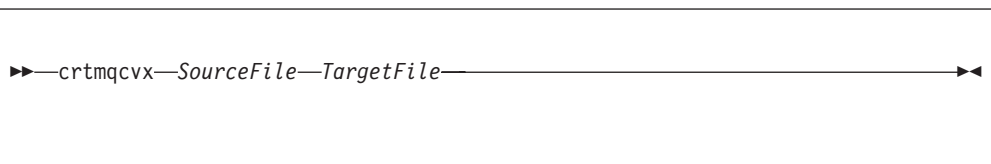
### Scopo

Utilizzare il comando **crtmqcvx** per creare un frammento di codice che converte dati nelle strutture del tipo di dati. Il comando genera una funzione C che può essere utilizzato in un uscita per convertire le strutture C.

Il comando rileva un file di input che contiene una o più strutture da convertire. Quindi realizza un file di output che contiene un o più frammenti di codice per convertire queste strutture.

Per ulteriori informazione relative a questo comando e al suo utilizzo, consultare *MQSeries Application Programming Guide*.

### Sintassi



### Parametri richiesti

#### *SourceFile*

Specifica il file di input che contiene le strutture C da convertire.

#### *TargetFile*

Specifica il file di output che contiene i frammenti di codice creati per convertire le strutture.

### Codici di errore

- 0 Comando completato normalmente
- 10 Comando completato con risultato inatteso
- 20 Si è verificato un errore durante la procedura

### Esempi

L'esempio che segue mostra l'effetto dell'utilizzo del comando di conversione dei dati in una struttura di origine C. Il comando inviato è:

```
crtmqcvx source.tmp target.c
```

Il file di input source.tmp verrà riportato così:

```

/* This is a test C structure which can be converted by the */
/* crtmqcvx utility                                         */

struct my_structure
{
    int    code;
    MQLONG value;
};

```

Il file di output, target.c, prodotto dal comando è riportato di seguito. È possibile utilizzare questi frammenti di codice nell'applicazione per convertire la struttura dei dati. Tuttavia, procedendo in questo modo, si noterà che il frammento utilizza delle macro fornite nel MQSeries file di intestazioneamqsvmha.h.

```

MQLONG Convertmy_structure(
    PMQBYTE *in_cursor,
    PMQBYTE *out_cursor,
    PMQBYTE in_lastbyte,
    PMQBYTE out_lastbyte,
    MQHCONN hConn,
    MQLONG  opts,
    MQLONG  MsgEncoding,
    MQLONG  ReqEncoding,
    MQLONG  MsgCCSID,
    MQLONG  ReqCCSID,
    MQLONG  CompCode,
    MQLONG  Reason)
{
    MQLONG ReturnCode = MQRC_NONE;

    ConvertLong(1); /* code */

    AlignLong();
    ConvertLong(1); /* value */

Fail:
    return(ReturnCode);
}

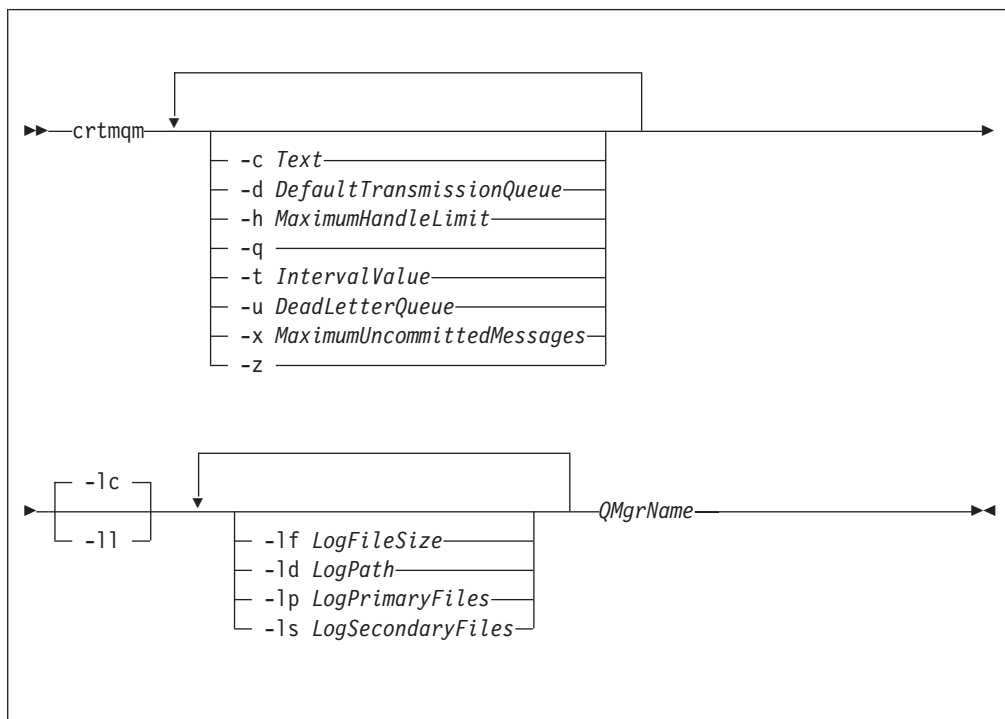
```

## crtmqm (Crea Queue Manager)

### Scopo

Utilizzo del comando **crtmqm** per creare un Queue Manager locale. Una volta creato un Queue Manager, utilizzare il comando **strmqm** per avviarlo.

### Sintassi



### Parametri richiesti

#### *QMgrName*

Specifica il nome del Queue Manager da creare. Il nome può contenere fino a 48 caratteri. Questo deve essere l'ultimo parametro del comando.

### Parametri opzionali

#### *-c Text*

Specifica con alcuni caratteri una descrizione del Queue Manager. Per impostazione predefinita il campo è vuoto.

È possibile utilizzare fino a 64 caratteri. Se si desidera utilizzare alcune lettere maiuscole e altre minuscole è necessario includere il testo tra virgolette.

#### *-d DefaultTransmissionQueue*

Specifica il nome della coda di trasmissione locale su cui vengono posizionati i messaggi remoti, quando non viene loro specificamente assegnata come destinazione una coda di trasmissione. Non esistono valori predefiniti.

#### *-h MaximumHandleLimit*

Specifica il numero massimo di gestioni che ogni applicazione può compiere contemporaneamente.

Specificare un valore compreso tra 1 e 999 999 999. Il valore predefinito è 256.

- q Specifica che questo Queue Manager viene reso predefinito. Il nuovo Queue Manager che ne sostituisce uno esistente diviene quello predefinito.

Se questa flag viene utilizzata accidentalmente e si desidera ripristinare come predefinito il precedente Queue Manager, è necessario digitare la voce *DefaultQueueManager* nel file di configurazione MQSeries. Consultare “Capitolo 13. Configurazione MQSeries” a pagina 179 per informazioni relative ai file di configurazione.

- t *IntervalValue*

Specifica l'intervallo di tempo di trigger in millisecondi delle code controllate da questo Queue Manager. Questo valore specifica il tempo dopo la ricezione di un messaggio che genera un trigger quando il triggering è sospeso. Di conseguenza, se la ricezione di un messaggio fa sì che la coda invii un messaggio di trigger alla coda di avvio, tutti i messaggi in arrivo sulla stessa coda all'interno dell'intervallo di tempo specificato non provocano un altro messaggio di trigger.

È possibile utilizzare l'intervallo di tempo di trigger per assicurare all'applicazione un sufficiente spazio di tempo per gestire una situazione relativa ad un trigger prima dell'avviso della necessità di gestirne un'altra sulla stessa coda. Potrebbe essere necessario visualizzare tutti gli eventi trigger che si verificano; a questo scopo impostare un valore pari o inferiore a zero in questo campo.

Specificare un valore tra 0 e 999 999 999. Il valore predefinito è 999 999 999 millisecondi, un tempo superiore a 11 giorni. Impostando il valore predefinito il triggering verrà disabilitato dopo la ricezione del primo messaggio di trigger. Tuttavia potrà essere riabilitato grazie ad un'applicazione servente la coda utilizzando un comando di modifica della coda che reimposta l'attributo trigger.

- u *DeadLetterQueue*

Specifica il nome della coda locale che deve essere utilizzata come coda messaggi non recapitati. I messaggi vengono inviati a questa coda se non riescono a raggiungere la loro corretta destinazione.

Se tale attributo è omesso, per impostazione predefinita non vi sarà alcuna coda messaggi non recapitati.

- x *MaximumUncommittedMessages*

Specifica il numero massimo di messaggi non assegnati relativi ad ogni syncpoint, cioè, la somma di:

- Il numero di messaggi che possono essere richiamati dalla code.
- Il numero di messaggi che possono essere inviati alle code.
- Tutti i messaggi di trigger generati all'interno di quest'unità di elaborazione.

Questa limitazione non si applica ai messaggi richiamati o espulsi da un syncpoint.

Specificare un valore tra 1 e 10 000. Il valore predefinito è 1000.

- z Sopprime i messaggi di errore.

Questa flag viene utilizzata solitamente in MQSeries per eliminare i messaggi non desiderati. Dal momento che l'utilizzo di questa flag comporta la perdita di dati, si raccomanda di non utilizzarla quando si inseriscono comandi in una riga di comandi.

## crtmqm

Il seguente gruppo di flag viene utilizzato per definire la registrazione nei log che il Queue Manager creato deve utilizzare. Per ulteriori informazioni relative ai log, consultare "Utilizzo del log per il ripristino" a pagina 152.

- lc Viene utilizzata la registrazione dei log circolare. Questo è il metodo predefinito di registrazione dei log.
- ll Viene utilizzata la registrazione dei log lineare.
- lf *LogFileSize*  
Specifica la dimensione dei file di log in unità di 4 KB. Il valore minimo è 64, quello massimo 16384. Il valore predefinito è 1024 data la dimensione predefinita del log in 4 MB.
- ld *LogPath*  
Specifica la directory da utilizzare per contenere i file di log. Il valore predefinito è MQS\_ROOT:[MQM.LOG]. Il valore predefinito può essere modificato qualora MQSeries viene personalizzato.  
  
L'ID utente mqm e il gruppo mqm devono avere pieno accesso ai file di log. Se si spostano questi file, è necessario disporre di questi accessi. Ciò avviene automaticamente se i file di log si trovano nella loro posizione predefinita.
- lp *LogPrimaryFiles*  
Specifica il numero di file di log primari da assegnare. Il valore predefinito è 3, il minimo 2 ed il massimo è 3.
- ls *LogSecondaryFiles*  
Specifica il numero di file di log secondari da assegnare. Il valore predefinito è 2, il minimo 1 ed il massimo è 61.

**Nota:** il numero totale di file di log è limitato a 63, indipendentemente dal numero richiesto.

I limiti riportati nella descrizione del precedente parametro sono limiti impostati da MQSeries. I limiti del sistema operativo possono ridurre le dimensioni massime di log.

## Codici di errore

- 0 Queue Manager creato
- 8 Queue Manager già esistente
- 49 Queue Manager arrestato
- 69 Memoria non disponibile
- 70 Spazio coda non disponibile
- 71 Errore imprevisto
- 72 Errore relativo al nome del Queue Manager
- 100 Posizione di log invalida
- 111 Queue Manager creato. Tuttavia, si è verificato un problema durante l'elaborazione della definizione del Queue Manager predefinito nel file di configurazione del prodotto. La specifica del Queue Manager predefinito potrebbe essere errata.
- 115 Dimensione di log invalida

## Esempi

1. Questo comando crea un Queue Manager predefinito denominato `Paint.queue.manager`, a cui viene data la descrizione `Paint shop`. Inoltre specifica che deve essere utilizzata la registrazione di log lineare.

```
crtmqm -c "Paint shop" -ll -q "Paint.queue.manager"
```

## crtmqm

2. L'esempio richiede un numero di file di log. Sono specificati 2 file di log primari e 3 secondi.

```
crtmqm -c "Paint shop" -ll -lp 2 -ls 3 -q "Paint.queue.manager"
```

3. In questo esempio, viene creato un altro Queue Manager, travel. L'intervallo di trigger è definito in 5000 millisecondi (5 secondi) ed è specificata SYSTEM.DEAD.LETTER.QUEUE. come coda messaggi non recapitati,

```
crtmqm -t 5000 -u SYSTEM.DEAD.LETTER.QUEUE "travel"
```

Una volta verificatosi un evento trigger, i successivi eventi vengono disabilitati per 5 secondi.

## Comandi correlati

<b>strmqm</b>	Avvia Queue Manager
<b>endmqm</b>	Chiudi Queue Manager
<b>dltmqm</b>	Elimina Queue Manager

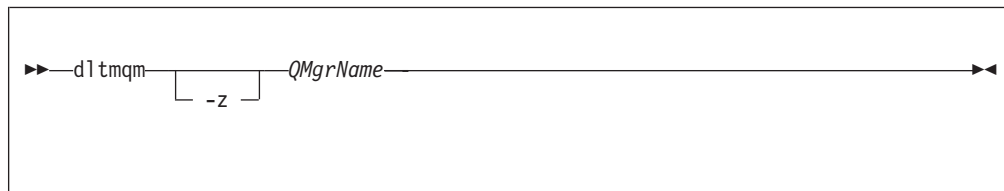


## dltmqm (Delete queue manager)

### Scopo

Utilizzare il comando **dltmqm** eliminare il Queue Manager specificato. Con questo comando si eliminano anche tutti gli oggetti associati a questo Queue Manager. Per poter eliminare un Queue Manager è necessario prima chiuderlo con il comando **endmqm**.

### Sintassi



### Parametri richiesti

*QMgrName*

Specifica il nome del Queue Manager da eliminare.

### Parametri opzionali

**-z** Sopprime i messaggi di errore.

### Codici di errore

0	Queue Manager eliminato
3	Creazione di Queue Manager
5	Queue Manager in esecuzione
16	Queue Manager inesistente
49	Queue Manager in chiusura
69	Memoria non disponibile
71	Errore imprevisto
72	Errore relativo al nome del Queue Manager
100	Posizione di log invalida
112	Queue Manager eliminato. Tuttavia, si è verificato un problema durante l'elaborazione del Queue Manager predefinito nel file di configurazione del prodotto. La specifica del Queue Manager predefinito potrebbe essere errata.

### Esempi

1. Il seguente comando elimina il Queue Manager `saturn.queue.manager`.

```
dltmqm "saturn.queue.manager"
```

2. Il seguente comando elimina il Queue Manager `travel` e sopprime ogni messaggio relativo al comando.

## dltmqm

```
dltmqm -z "travel"
```

**Comandi correlati**

<b>crtmqm</b>	Crea un Queue Manager
<b>strmqm</b>	Avvia un Queue Manager
<b>endmqm</b>	Arresta un Queue Manager

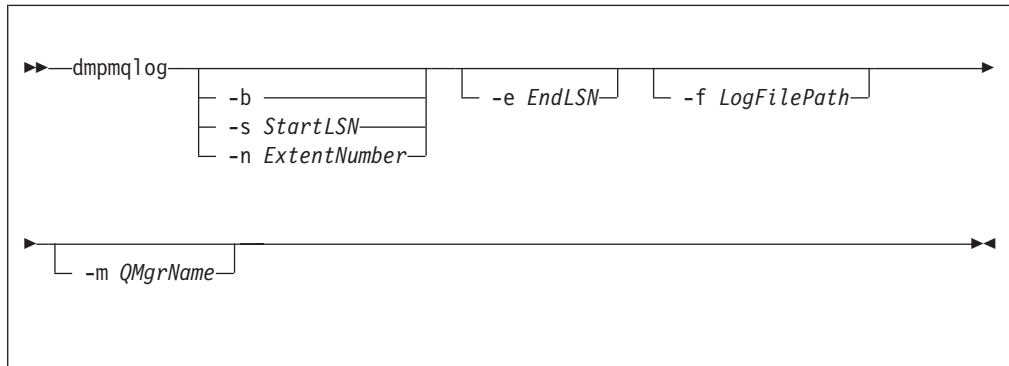
## dmpmqlog (Dump log)

### Scopo

Utilizzare il comando **dmpmqlog** scaricare una copia formattata del log di sistema MQSeries.

Il log da scaricare deve essere stato creato sullo stesso tipo di sistema operativo utilizzato per immettere il comando.

### Sintassi



### Parametri opzionali

#### Punto di inizio del dump

Utilizzare i seguenti parametri per specificare il numero di sequenza di log LSN (Log Sequence Number) dal quale il dump deve avere inizio. Se non viene indicato alcun punto di inizio, per impostazione predefinita, lo scaricamento inizia dal LSN del primo record nella parte attiva del log.

**-b** Specifica che lo scaricamento deve iniziare dalla base LSN. La base LSN identifica l'inizio dell'estensione del log che contiene la prima parte attiva del log.

**-s** *StartLSN*

Specifica che lo scaricamento deve iniziare dal numero di sequenza log specificato. Questo viene specificato nel formato nnnn:nnnn:nnnn:nnnn.

Se si utilizza un log circular, il valore LSN deve essere pari o superiore al valore base LSN del log.

**-n** *ExtentNumber*

Specifica che lo scaricamento deve iniziare dal numero di estensione specificato. Il numero di estensione deve essere compreso tra 0-9 999 999.

Questi parametri sono validi solo per i Queue Manager il cui *LogType* è LINEAR (così come registrato nel file di configurazione, qm.ini).

**-e** *EndLSN*

Indica che lo scaricamento terminerà al numero di sequenza LSN specificato. Tale specificazione avviene nel formato 0nnnn:nnnn:nnnn:nnnn.

**-f** *LogFilePath*

Indica il nome del percorso assoluto, non relativo, della directory dei file di log. La directory specificata deve contenere l'intestazione del file di log (amqh1ctl.lfh) e una sottodirectory denominata active. La sottodirectory

attiva deve contenere i file di log. Per impostazione predefinita, si presume che i file di log siano memorizzati nelle directory specificate nei file `mq5.ini` e `qm.ini`. Se si utilizza questa opzione, i nomi di coda, associati agli identificatori di coda, verranno mostrati soltanto nello scaricamento, se un nome di Queue Manager viene specificato esplicitamente con l'opzione `-m` e quel Queue Manager ha il file di catalogo di oggetto nel percorso della propria directory.

Nei sistemi che supportano nomi di file estesi, questo file è denominato `mqmqobjcat` e, per poter localizzare gli identificatori di coda relativi ai nomi di coda, deve essere il file utilizzato quando i file di log sono stati creati. Ad esempio, per un Queue Manager denominato `qm1`, il file di catalogo di oggetto è memorizzato nella directory `MQS_ROOT:[MQM.QMGRS.QM1.QMANAGER]`. Per realizzare questa mappatura, può essere necessario creare un Queue Manager temporaneo, denominato ad esempio `tmpq`, sostituire il relativo catalogo di oggetto con quello associato ai file di log specifico e quindi avviare `dmpmqlog`, specificando `-m tmpq` e `-f` con il nome del percorso assoluto della directory ai file di log.

**--m *QMgrName***

Indica il nome del Queue Manager. Se questo parametro è omissso, viene utilizzato il nome del Queue Manager predefinito.

Il Queue Manager specificato, o quello predefinito, non devono essere in esecuzione nel momento in cui si immette il comando **dmpmqlog**. Ovviamente, il Queue Manager non deve essere avviato neanche quando il comando **dmpmqlog** è in esecuzione.

## dspmqaout (Display Authority)

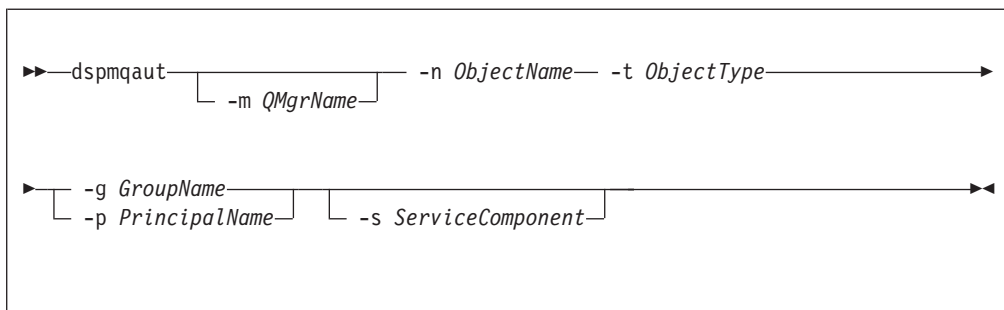
### Scopo

Utilizzare il comando **dspmqaout** per visualizzare le correnti autorizzazioni relative ad un oggetto specificato.

Può essere specificato un solo gruppo.

Se l'ID utente partecipa in più di un gruppo, questo comando visualizza le autorizzazioni combinate di tutti i gruppi.

### Sintassi



### Parametri richiesti

#### -n *ObjectName*

Specifica il nome dell'oggetto sul quale deve essere fatta la richiesta.

Questo è un parametro necessario *a meno che* la richiesta non sia relativa allo stesso Queue Manager, nel qual caso non deve essere incluso.

Occorre specificare il nome del Queue Manager, della coda o della definizione di processo.

#### -t *ObjectType*

Specifica il tipo di oggetto sul quale deve essere fatta la richiesta. I possibili valori sono:

**queue o q** Una o più code che corrispondano al parametro del tipo di oggetto.

**qmgr** Un oggetto Queue Manager

**process o prcs** Un processo

**namelist o nl** Un namelist

### Parametri opzionali

#### -m *QMgrName*

Specifica il nome del Queue Manager sul quale eseguire la richiesta.

#### -g *GroupName*

Specifica il nome o il gruppo utente sul quale eseguire la richiesta. È possibile specificare *un solo* nome, che deve essere il nome di un valido identificativo esistente.

#### -p *PrincipalName*

Specifica il nome di un utente le cui autorizzazioni relative all'oggetto specificato devono essere visualizzate.

**-s ServiceComponent**

Questo parametro si applica solo se si stanno utilizzando i servizi installabili Authorization Service, altrimenti viene ignorato.

Se tali servizi sono supportati, questo parametro specifica il nome dell'Authorization Service a cui si applicano le autorizzazioni. Questo parametro è facoltativo; se non viene specificato, la richiesta di autorizzazione viene eseguita per il primo componente installabile per il servizio.

## Parametri restituiti

Questo comando riporta un elenco di autorizzazione, che può contenere nessuno, uno o più parametri di autorizzazione. Ciascun parametro riportato indica che un ID utente nel gruppo specificato ha il potere di eseguire l'operazione indicata da quel parametro.

Tabella 16 mostra le autorizzazioni che possono essere date a ciascun tipo di oggetto.

Tabella 16. Autorizzazioni di sicurezza del comando dspmqaout

Autorizzazione	Coda	Processo	Qmgr	Namelist
all	✓	✓	✓	✓
alladm	✓	✓	✓	✓
allmqi	✓	✓	✓	✓
altusr			✓	
browse	✓			
chg	✓	✓	✓	✓
clr	✓			
connect			✓	
cpy	✓	✓	✓	✓
crt	✓	✓	✓	✓
dlt	✓	✓	✓	✓
dsp	✓	✓	✓	✓
get	✓			
inq	✓	✓	✓	✓
passall	✓			
passid	✓			
put	✓			
set	✓	✓	✓	
setall	✓		✓	
setid	✓		✓	

L'elenco che segue riporta le autorizzazioni associati ai rispettivi parametri:

- all** Utilizza tutte le operazioni relative all'oggetto.
- alladm** Esegue tutte le operazioni di gestione relative all'oggetto.
- allmqi** Utilizza tutte le chiamate MQI relative all'oggetto.
- altusr** Specifica un ID utente alternativo su una chiamata MQI.

## dspmqaout

<b>browse</b>	Richiama un messaggio da una coda inviando una chiamata MQGET con l'opzione BROWSE.
<b>chg</b>	Modifica gli attributi dell'oggetto specificato, utilizzando l'appropriato gruppo di comandi.
<b>clr</b>	Ripulire una coda (comando PCF: Clear queue only).
<b>connect</b>	Collega l'applicazione al Queue Manager specificato inviando una chiamata MQCONN.
<b>cpy</b>	Copia la definizione di un oggetto, ad esempio il comando PCF Copy queue.
<b>crt</b>	Crea oggetti del tipo specificato, utilizzando l'appropriato gruppo di comandi.
<b>dlt</b>	Elimina l'oggetto specificato, utilizzando l'appropriato gruppo di comandi.
<b>dsp</b>	Visualizza gli attributi dell'oggetto specificato, utilizzando l'appropriato gruppo di comandi.
<b>get</b>	Richiama un messaggio da una coda inviando una chiamata MQGET.
<b>inq</b>	Effettua una richiesta su uno specifico Queue Manager inviando una chiamata MQINQ.
<b>passall</b>	Passa tutto il contesto.
<b>passid</b>	Passa il contesto identità.
<b>put</b>	Invia un messaggio su una specifica coda inviando una chiamata MQPUT.
<b>set</b>	Imposta attributi su una coda dal MQI inviando una chiamata MQSET.
<b>setall</b>	Imposta tutto il contesto su una coda.
<b>setid</b>	Imposta il contesto identità su una coda.

Le autorizzazioni per le operazioni di gestione si applicano, se supportate, a questi gruppi di comandi:

- Comandi di controllo
- Comandi MQSC
- Comandi PCF

## Codici di errore

0	Operazione eseguita
36	Elemento immesso non valido
40	Queue Manager non disponibile
49	Queue Manager arrestato
69	Memoria non disponibile
71	Errore imprevisto
72	Errore relativo al nome del Queue Manager
133	Nome oggetto sconosciuto
145	Nome oggetto imprevisto
146	Nome oggetto assente
147	Tipo oggetto assente
148	Tipo oggetto invalido
149	Nome entità assente



## Esempi

L'esempio seguente mostra comando di visualizzazione delle autorizzazioni relative al Queue Manager saturn.queue.manager associato al gruppo utente staff:

```
dspmqaout -m "saturn.queue.manager" -t qmgr -g staff
```

Gli effetti di questo comando sono:

```
Entity staff has the following authorizations for object:  
  get  
  browse  
  put  
  inq  
  set  
  connect  
  altusr  
  passid  
  passall  
  setid
```

## Comandi correlati

**setmqaut**      Imposta o reimposta l'autorizzazione

## dspmqcsv (Display Command Server)

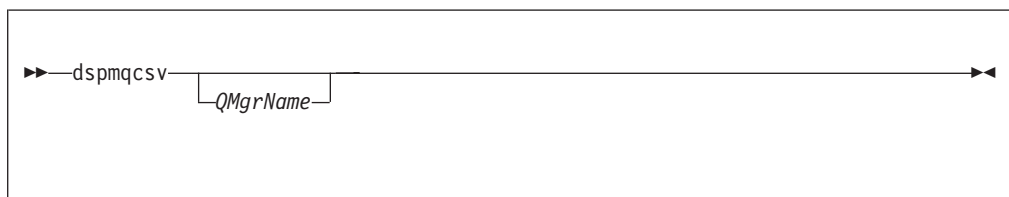
### Scopo

Utilizzare il comando **dspmqcsv** per visualizzare lo stato del server di comando relativo ad uno specifico Queue Manager.

Verrà visualizzato uno dei seguenti stati:

- Avvio
- Esecuzione
- Esecuzione con SYSTEM.ADMIN.COMMAND.QUEUE non abilitato per ricevere
- Chiusura
- Arresto

### Sintassi



### Parametri opzionali

*QMgrName*

Specifica il nome del Queue Manager locale per il quale viene richiesto lo stato del server di comando.

### Codici di errore

- |    |  |
|----|--|
| 0  | Comando completato normalmente                   |
| 10 | Comando completato con risultato inatteso        |
| 20 | Si è verificato un errore durante l'elaborazione |

### Esempi

Il seguente comando visualizza lo stato del server di comando associato al `venus.q.mgr`:

```
dspmqcsv "venus.q.mgr"
```

### Comandi correlati

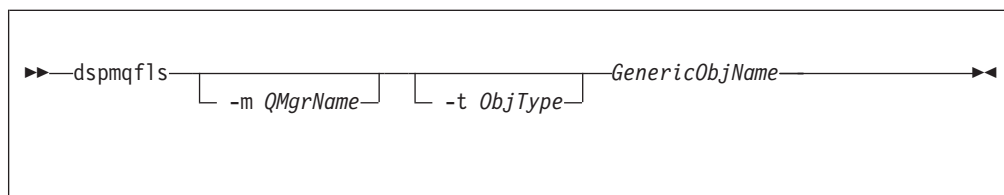
- |                 |                             |
|-----------------|-----------------------------|
| <b>strmqcsv</b> | Avvia un server di comando  |
| <b>endmqcsv</b> | Chiude un server di comando |

## dspmqls (Display MQSeries files)

### Scopo

Utilizzare il comando **dspmqls** per visualizzare il vero nome del file system per tutti gli oggetti MQSeries che rispondono ad un determinato parametro. È possibile utilizzare tale comando per identificare i file associati ad un particolare oggetto MQSeries. In questo modo è più agevole la realizzazione di copie di riserva di determinati oggetti. Consultare "Analisi dei nomi di file MQSeries" a pagina 21 per ulteriori informazioni relative alla trasformazione del nome.

### Sintassi



### Parametri richiesti

#### *GenericObjName*

Specifica il nome dell'oggetto MQSeries. Il nome è una stringa priva di flag ed è un parametro obbligatorio. Se omissso, viene riportato un errore.

Questo parametro supporta il carattere globale \* alla fine della stringa.

### Parametri opzionali

#### **-m** *QMgrName*

Specifica il nome del Queue Manager i cui file devono essere esaminati. Se omissso, il comando viene eseguito sul Queue Manager predefinito.

#### **-t** *ObjType*

Specifica il tipo di oggetto MQSeries. L'elenco che segue mostra i tipi di oggetto validi. I nomi abbreviati sono riportati seguiti dalla loro estensione.

- \* o all** Tutti i tipi di oggetto; questa è l'impostazione predefinita
- q o queue** Una o più code corrispondenti al parametro nome oggetto
- ql o qlocal** Una coda locale
- qa o qalias** Un alias di coda
- qr o qremote** Una coda remota
- qm o qmodel** Un modello di coda
- qmgr** Un oggetto Queue Manager
- prcs o process** Un processo
- ctlg o catalog** Un oggetto catalogo
- nl o namelist** Un namelist

**Nota:** il comando **dspmqls** visualizza la directory che contiene la coda e *non* il nome della coda.

## dspmqfls

### Codici di errore

0	Comando completato normalmente
10	Comando completato con risultato inatteso
20	Si è verificato un errore durante l'elaborazione

### Esempi

1. Il seguente comando visualizza i dettagli di tutti gli oggetti il cui nome inizia con SYSTEM.ADMIN specificati nel Queue Manager predefinito.

```
dspmqfls SYSTEM.ADMIN*
```

2. Il seguente comando visualizza i dettagli dei file di tutti i processi il cui nome inizia con PROC definiti nel Queue Manager RADIUS.

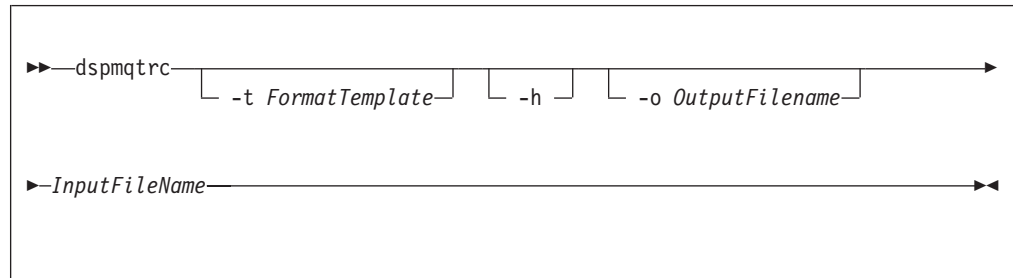
```
dspmqfls -m RADIUS -t prcs PROC*
```

## dspmqtrc (Visualizza output di traccia formattato di MQSeries)

### Scopo

Utilizzare il comando **dspmqtrc** per visualizzare l'output di traccia formattata di MQSeries.

### Sintassi



### Parametri richiesti

*InputFileName*

Specifica il nome del file contenente la traccia non formattata. Ad esempio  
 MQS\_ROOT:[MQM.TRACE]AMQ20202345.TRC.

### Parametri opzionali

**-t** *FormatTemplate*

Specifica il nome del file modello contenente dettagli su come visualizzare la traccia. Il valore predefinito è SYS\$SHARE:AMQTRC.FMT.

**-h** Omette le informazioni dell'intestazione dal prospetto.

**-o** *output\_filename*

Il nome del file in cui scrivere i dati formattati.

### Esempi

1. Il seguente comando mostra il reindirizzamento dell'output:

```
dspmqtrc mqs_root:[mqm.trace]amq20202345.trc > mqs_root:[mqm.trace]amq20202345.fmt
```

**dspmqtrc**

## **Comandi correlati**

<b>endmqtrc</b>	Fine traccia di MQSeries
<b>strmqtrc</b>	Avvia traccia di MQSeries



## **dspmqtrn**

49	Queue Manager in chiusura
69	Memoria non disponibile
71	Errore imprevisto
72	errore relativo al nome del Queue Manager
102	Nessuna transazione rilevata

## **Comandi correlati**

<b>rsvmqtrn</b>	Risolvi transazione MQSeries
-----------------	------------------------------

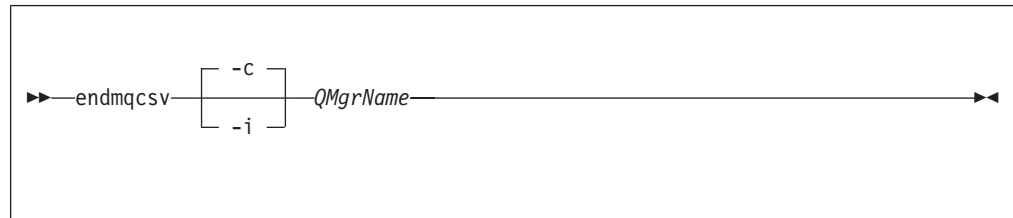


## endmqcsv (End command server)

### Scopo

Utilizzare il comando **endmqcsv** per arrestare il server di comandi relativo al Queue Manager specificato.

### Sintassi



### Parametri richiesti

*QMGrName*

Specifica il nome del Queue Manager il cui server di comandi deve essere arrestato.

### Parametri opzionali

**-c** Specifica che il server di comandi deve essere arrestato in modo controllato. Al server di comandi sarà così possibile completare le elaborazioni dei messaggi di comando già avviate. Nessuno nuovo messaggio viene letto dalla coda comandi.

Questa è l'impostazione predefinita.

**-i** Specifica che il server di comandi deve essere arrestato immediatamente. Non verranno completate le azioni associate ad un messaggio di comando ancora in elaborazione.

### Codici di errore

0	Comando completato normalmente
10	Comando completato con risultato inatteso
20	Si è verificato un errore durante l'elaborazione

### Esempi

1. Il seguente comando arresta il server di comandi relativo al Queue Manager saturn.queue.manager:

```
endmqcsv -c "saturn.queue.manager"
```

Il server di comandi, prima dell'arresto, completa l'elaborazione dei comandi già avviati. I nuovi comandi ricevuti rimangono non elaborati nella coda comandi fino al nuovo riavvio del server di comandi.

2. Il seguente comando arresta immediatamente il server di comandi relativo al Queue Manager pluto:

## endmqcsv

```
endmqcsv -i "pluto"
```

## Comandi correlati

<code>strmqcsv</code>	Avvia un server di comandi
<code>dspmqcsv</code>	Visualizza lo stato di un server di comandi

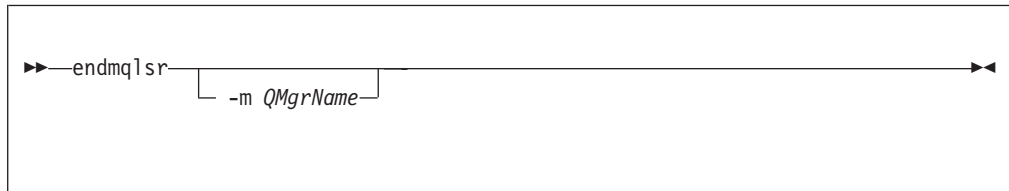
## endmq1sr (End listener)

### Scopo

Il comando **endmq1sr** arresta ogni processo listener relativo al Queue Manager specificato.

Il Queue Manager deve essere arrestato prima di immettere il comando **endmq1sr**.

### Sintassi



### Parametri opzionali

**-m** *QMgrName*

Specifica il nome del Queue Manager. Se non viene specificato alcun nome, l'elaborazione riguarderà il Queue Manager predefinito.

### Codici di errore

- 0 Comando completato normalmente
- 10 Comando completato con risultato inatteso
- 20 Si è verificato un errore durante l'elaborazione

## endmqm (End queue manager)

### Scopo

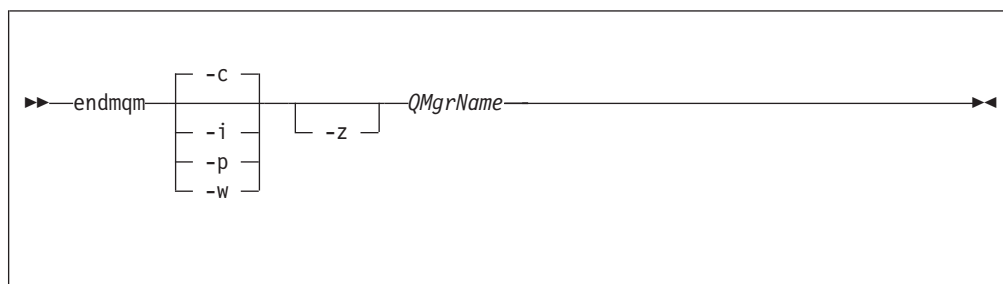
Utilizzare il comando **endmqm** per arrestare un Queue Manager locale specificato. Questo comando arresta il Queue Manager in uno dei seguenti tre modi:

- Chiusura normale o ad attività completate
- Chiusura immediata
- Chiusura preventiva

Gli attributi del Queue Manager e gli oggetti associati non vengono coinvolti. È possibile riavviare il Queue Manager utilizzando il comando **strmqm** (Start queue manager).

Per eliminare un Queue Manager, è necessario prima arrestarlo e poi utilizzare il comando **dltmqm** (Delete queue manager).

### Sintassi



### Parametri richiesti

*QMgrName*

Specifica il nome del Queue Manager da arrestare.

### Parametri opzionali

- c Chiusura controllata (o ad attività completate). Il Queue Manager viene chiuso solo dopo la disconnessione di tutte le applicazioni. Tutte le chiamate MQI in corso di elaborazione vengono completate. Questa è l'impostazione predefinita.

La risposta è immediata e non viene notificata alcuna notizia relativa al momento di chiusura del Queue Manager.

- w Chiusura in attesa

Questo tipo di chiusura è equivalente alla chiusura controllata, salvo che il controllo viene restituito solo dopo la chiusura del Queue Manager. Durante l'elaborazione della chiusura, si riceve il messaggio "Waiting for queue manager *QMgrName* to end".

- i Chiusura immediata. Il Queue Manager viene chiuso dopo il completamento di tutte le chiamate MQI attualmente in elaborazione. Ogni richiesta di chiamata MQI successiva al comando ha esito negativo. Al successivo riavvio del Queue Manager viene eseguito il roll back di tutte le unità di elaborazione incomplete.
- p Chiusura preventiva.

## endmqm

*Utilizzare questo tipo di chiusura solo in circostanze eccezionali.* Ad esempio, qualora un Queue Manager non si arresta in seguito ad un normale comando **endmqm**.

Il Queue Manager viene chiuso senza attendere la disconnessione delle applicazioni o il completamento delle chiamate MQI. Questo metodo può avere conseguenze imprevedibili per le applicazioni MQSeries. Tutti i processi la cui chiusura ha esito negativo vengono arrestati 30 secondi dopo l'immissione del comando.

**Nota:** in seguito ad una chiusura forzata o preventiva oppure in seguito ad un malfunzionamento del Queue Manager, la chiusura avverrà senza la cancellazione della memoria condivisa di pertinenza del Queue Manager. Questo può determinare dei problemi al riavvio. Per informazioni relative all'impiego delle utilità MONMQ per la cancellazione della memoria in seguito a chiusure di questo tipo, consultare "Gestione della memoria condivisa con MONMQ" a pagina 372.

-z Sopprime i messaggi di errore relativi al comando

## Codici di errore

0	Queue Manager arrestato
3	Creazione di Queue Manager
16	Queue Manager inesistente
40	Queue Manager non disponibile
49	Queue Manager in chiusura
69	Memoria non disponibile
71	Errore imprevisto
72	Errore relativo al nome del Queue Manager

## Esempi

Gli esempi seguenti mostrano comandi che arrestano il Queue Manager specificato:

1. Questo comando arresta il Queue Manager denominato `mercury.queue.manager` con il metodo controllato. Tutte le applicazioni attualmente connesse possono disconnettersi.

```
endmqm "mercury.queue.manager"
```

2. Questo comando arresta il Queue Manager denominato `saturn.queue.manager` con il metodo immediato. Tutte le chiamate MQI in corso sono completate, ma non ne vengono accettate nuove.

```
endmqm -i "saturn.queue.manager"
```

## Comandi correlati

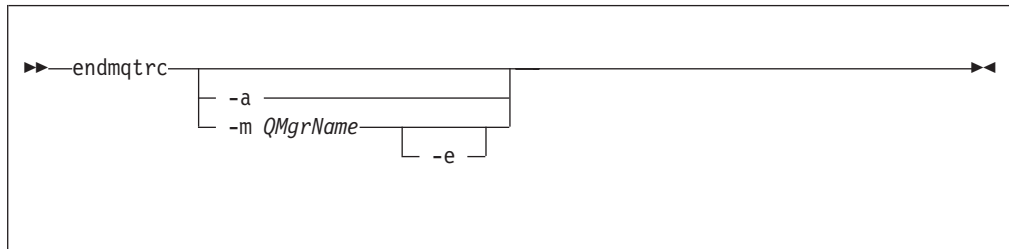
<code>crtmqm</code>	Crea un Queue Manager
<code>strmqm</code>	Avvia un Queue Manager
<code>dltmqm</code>	Elimina un Queue Manager

## endmqtrc (End MQSeries trace)

### Scopo

Utilizzare il comando **endmqtrc** per arrestare il tracciamento dell'entità specificata o di tutte le entità.

### Sintassi



### Parametri opzionali

**-m** *QMgrName*

Indica il nome del Queue Manager la cui funzione di traccia deve essere arrestata.

Al comando può essere fornita una sola flag -m ed il nome del Queue Manager associato.

Nello stesso comando possono essere specificati un nome di Queue Manager e la flag -m, così come avviene per la flag -e.

**-e** Specificando tale flag, viene arrestata la traccia più recente.

**-a** Specificando questa flag, vengono arrestate tutte le tracce.

Questa flag *deve* essere specificata da sola.

### Codici di errore

**AMQ5611** Questo messaggio è emesso nel caso vengano forniti elementi invalidi al comando.

### Esempi

Questo comando arresta la funzione di traccia dei dati di un Queue Manager denominato QM1.

```
endmqtrc -m QM1
```

### Comandi correlati

**dspmqtrc** Visualizza l'output di traccia formattato  
**strmqtrc** Avvia traccia MQSeries

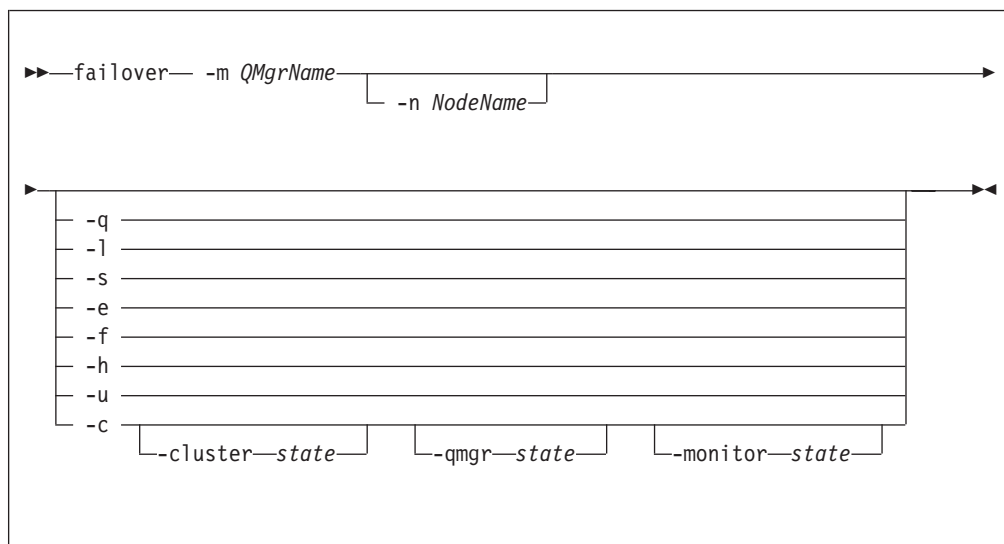


## failover (Gestione di un gruppo di ripristino)

### Scopo

Utilizzare il comando **failover** per gestire un gruppo di ripristino. Il comando **failover** sia i parametri di aggiornamento sia quelle di query. Il comando **failover** può essere eseguito da un qualunque nodo OpenVMS del gruppo di ripristino.

### Sintassi



### Parametri richiesti

**-m** *QMgrName*

Specifica il nome del Queue Manager cui applicare il comando **failover**. La dimensione massima consentita per il nome *QMgrName* è pari a 25 caratteri.

**-n** *NodeName*

Specifica il nome del nodo OpenVMS al quale applicare il comando. Questo è obbligatorio per i parametri **-h** e **-c**.

### Parametri opzionali

**-q** Richiede lo stato del gruppo di ripristino e visualizza l'output.

**-l** Richiede lo stato del gruppo di ripristino ed imposta i seguenti simboli DCL:

Nome del simbolo DCL	Descrizione
MQS\$QMGR_NODE	Impostazione per il nodo OpenVMS che esegue il Queue Manager; se non vi è alcun Queue Manager in esecuzione la stringa è vuota.
MMQS\$AVAILABLE_NODES	Impostazione per l'elenco dei nodi OpenVMS su cui è possibile eseguire il Queue Manager. Tali nodi sono quelli che visualizzano lo stato AVAILABLE e quelli su cui è in esecuzione un failover monitor.
MQS\$MONITOR_NODES	Impostazione per l'elenco dei nodi OpenVMS su cui è in esecuzione un failover monitor.

**-s** Avvia il Queue Manager nel gruppo di ripristino. Se viene specificato il

## failover

parametro -n, il Queue Manager è avviato sul nodo OpenVMS specificato; in caso contrario è avviato sul nodo disponibile con priorità più alta.

- e Arresta il Queue Manager nel gruppo di ripristino.
- f Sposta il Queue Manager in un altro nodo del gruppo di ripristino. Se è specificato il parametro -n, il Queue Manager viene spostato nel nodo specificato; in caso contrario, viene spostato nel nodo disponibile con priorità più alta.
- h Interrompe il failover monitor in esecuzione sul nodo specificato con il parametro -n.
- u Annulla le flag di aggiornamento in elaborazione.
- c Modifica lo stato del gruppo di ripristino. Gli stati modificati sono determinati in base ai tre parametri seguenti. Le modifiche hanno effetto solo se compatibili con lo stato di esecuzione del gruppo di ripristino.
  - cluster started | stopped  
Utilizzato con il parametro -c per modificare lo stato generale del gruppo di ripristino.
  - qmgr available | running | excluded  
Utilizzato con il parametro -c per modificare lo stato del Queue Manager del nodo specificato con il parametro -n.
  - monitor started | stopped | watching  
Utilizzato con il parametro -c per modificare lo stato del failover monitor del nodo specificato con il parametro -n.

## Codici di errore

0	Comando completato normalmente
5	Il Queue Manager è in esecuzione
36	Gli elementi immessi in un comando sono invalidi
326	Queue Manager MQseries non in esecuzione
1925	Non esiste un failover monitor avviato per il Queue Manager
1926	Operazione di aggiornamento del gruppo di ripristino in corso
1937	Nodo sul quale avviare il Queue Manager non disponibile
1939	Arresto forzato del Queue Manager
1940	L'arresto del Queue Manager si è verificato prima del completamento.

### Codici di errore OpenVMS

36	%SYSTEM-F-NOPRIV, privilegio insufficiente o violazione di oggetto protetto
652	%SYSTEM-F-NOSUCHNODE, nodo remoto sconosciuto
660	%SYSTEM-F-REJECT, connessione all'oggetto rete respinta

## Esempi

1. Questo esempio richiede lo stato di un gruppo di ripristino relativo ad un Queue Manager denominato testqm.

```
failover -m "testqm" -q
```

2. Questo esempio avvia il Queue Manager denominato testqm sul nodo batman.

```
failover -m "testqm" -n batman -s
```

3. Questo esempio sposta il Queue Manager denominato testqm al nodo disponibile con la priorità maggiore.

```
failover -m "testqm" -f
```

**failover**

## **Comandi correlati**

**runmqfm**      Avvia un failover monitor

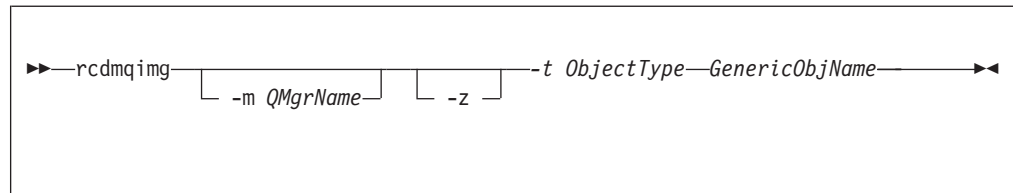
## rcdmqimg (Registra immagine supporto)

### Scopo

Utilizzare il comando **rcdmqimg** per scrivere un'immagine di un oggetto o gruppo di oggetti di MQSeries nel log per l'utilizzo nel ripristino dei supporti. Utilizzare il comando **rcrmqobj** associato per ricreare l'oggetto dall'immagine.

Questo comando viene utilizzato con un Queue Manager attivo. Un'ulteriore attività sul Queue Manager viene registrata in modo che, sebbene l'immagine divenga obsoleta, i record del log riflettono qualsiasi modifica dell'oggetto.

### Sintassi



### Parametri richiesti

#### *GenericObjName*

Specifica il nome dell'oggetto che deve essere registrato. Questo parametro potrebbe avere un asterisco rimanente per indicare che qualsiasi oggetto con un nome corrispondente alla porzione del nome precedente l'asterisco deve essere registrato.

Questo parametro è necessario *a meno che* si stia registrando un oggetto Queue Manager o il file di sincronizzazione canale. Se si specifica un nome oggetto per il file di sincronizzazione canale, questo viene ignorato.

#### **-t** *ObjectType*

Specifica il tipo di oggetti le cui immagini devono essere registrate. Tipi di oggetto validi sono:

<b>prcs</b> o <b>process</b>	Processi
<b>q</b> o <b>queue</b>	Tutti i tipi di coda
<b>ql</b> o <b>qlocal</b>	Code locali
<b>qa</b> o <b>qalias</b>	Code di alias
<b>qr</b> o <b>qremote</b>	Code remote
<b>qm</b> o <b>qmodel</b>	Code modello
<b>qmgr</b>	Oggetto Queue Manager
<b>syncfile</b>	File di sincronizzazione canale
<b>nl</b> o <b>namelist</b>	Namelist
<b>ctlg</b> o <b>catalog</b>	Un catalogo oggetti
<b>*</b> o <b>all</b>	Tutti i precedenti

## rcdmqimg

### Parametri opzionali

**-m** *QMgrName*

Specifica il nome del Queue Manager per il quale si devono registrare le immagini. Se omissso, il comando opera sul Queue Manager predefinito.

**-z** Sopprime i messaggi di errore.

### Codici di errore

0	Operazione eseguita
36	Elemento immesso non valido
40	Queue Manager non disponibile
49	Arresto di Queue Manager
68	Ripristino media non supportato
69	Memoria non disponibile
71	Errore imprevisto
72	Errore relativo al nome del Queue Manager
119	Utente non autorizzato
128	Nessun oggetto elaborato
131	Problema di risorsa
132	Oggetto danneggiato
135	Impossibile registrare oggetto temporaneo

### Esempi

Il seguente comando registra un'immagine dell'oggetto Queue Manager saturn.queue.manager nel log.

```
rcdmqimg -t qmgr -m "saturn.queue.manager"
```

### Comandi correlati

**rcrmqobj** Ricrea un oggetto Queue Manager

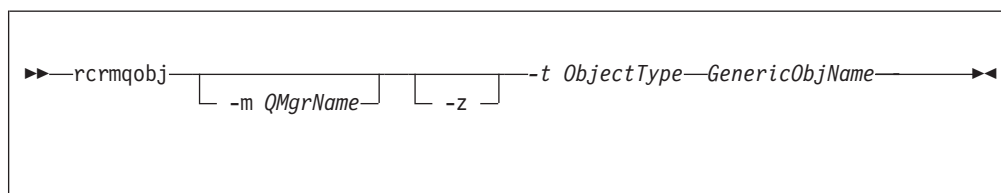
## rcrmqobj (Ricrea oggetto)

### Scopo

Utilizzare il comando **rcrmqobj** per ricreare un oggetto o un gruppo di oggetti, dalle relative immagini contenute nel log. Utilizzare il comando associato, **rcdmqimg**, per registrare le immagini dell'oggetto nel log.

Questo comando deve essere utilizzato su un Queue Manager in esecuzione. Tutte le attività sul Queue Manager dopo che l'immagine è stata memorizzata vengono registrate nel log. Per ricreare un oggetto occorre ripetere il log per ricreare gli eventi che si sono verificati dopo che l'immagine dell'oggetto è stata catturata.

### Sintassi



### Parametri richiesti

#### *GenericObjName*

Specifica il nome dell'oggetto che deve essere ricreato. Questo parametro potrebbe avere un asterisco rimanente per indicare che qualsiasi oggetto con un nome corrispondente alla porzione del nome precedente l'asterisco deve essere ricreato.

Questo parametro è necessario *a meno che* il tipo di oggetto sia il file di sincronizzazione canale; se viene fornito un nome oggetto per questo tipo di oggetto, questo sarà ignorato.

#### **-t** *ObjectType*

Specifica il tipo di oggetti da ricreare. Tipi di oggetto validi sono:

<b>prcs</b> o <b>process</b>	Processi
<b>q</b> o <b>queue</b>	Tutti i tipi di coda
<b>ql</b> o <b>qlocal</b>	Code locali
<b>qa</b> o <b>qalias</b>	Code di alias
<b>qr</b> o <b>qremote</b>	Code remote
<b>qm</b> o <b>qmodel</b>	Code modello
<b>nl</b> o <b>namelist</b>	Namelist
<b>ctlg</b> o <b>catalog</b>	Un catalogo oggetti
<b>*</b> o <b>all</b>	Tutti i precedenti
<b>syncfile</b>	Il file di sincronizzazione canale

**Nota:** L'utilizzo di questo flag determina la rigenerazione del file di sincronizzazione canale per il Queue Manager specificato. Ciò è necessario poiché il file non viene salvato dal comando **rcdmqimg**.

## Parametri opzionali

**-m** *QMgrName*

Specifica il nome del Queue Manager per il quale si devono ricreare gli oggetti.  
Se omissso, il comando opera sul Queue Manager predefinito.

**-z** Sopprime i messaggi di errore.

## Codici di errore

0	Operazione eseguita
36	Elemento immesso non valido
40	Queue Manager non disponibile
49	Arresto Queue Manager
66	Immagine supporto non disponibile
68	Ripristino media non supportato
69	Memoria non disponibile
71	Errore imprevisto
72	Errore relativo al nome del Queue Manager
119	Utente non autorizzato
128	Nessun oggetto elaborato
135	Impossibile ripristinare oggetto temporaneo
136	Oggetto in uso

## Esempi

1. Il seguente comando ricrea tutte le code locali per il Queue Manager predefinito:

```
rcrmqobj -t ql *
```

2. Il seguente comando ricrea tutte le code remote associate al Queue Manager store:

```
rcrmqobj -m "store" -t qr *
```



## Comandi correlati

`rcdmqimg` Registra un oggetto MQSeries nel log

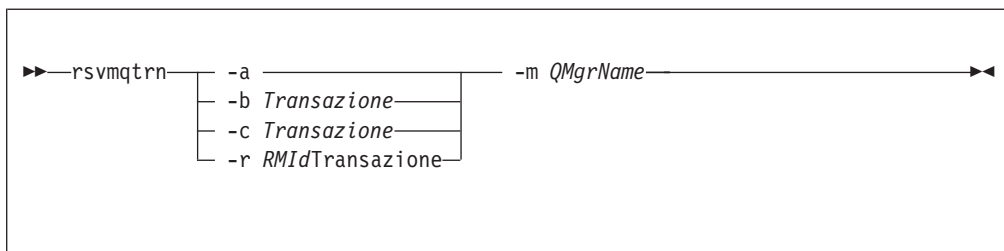
## rsvmqtrn (risolve transazioni MQSeries)

### Scopo

Utilizzare il comando **rsvmqtrn** per eseguire il commit o il back out delle transazioni in sospenso coordinate internamente o esternamente.

Utilizzare questo comando solo quando si è certi che non è possibile risolvere le transazioni mediante i normali protocolli. L'emissione di questo comando potrebbe determinare la perdita di integrità transazione tra i gestori risorse per una transazione distribuita.

### Sintassi



### Parametri richiesti

**-m** *QMgrName*  
 Specifica il nome del Queue Manager. Questo parametro è obbligatorio.

### Parametri opzionali

- a** Specifica il Queue Manager che dovrebbe tentare di risolvere tutte le transazioni in sospenso coordinate internamente (cioè, tutte le unità globali di lavoro).
- b** Specifica che è necessario eseguire il back out della transazione nominata. Questo flag è valido solo per le transazioni coordinate esternamente (cioè, per unità esterne di lavoro).
- c** Specifica che è necessario eseguire il commit della transazione nominata. Questo flag è valido solo per le transazioni coordinate esternamente (cioè, per unità esterne di lavoro).
- r** *RMIId*  
 Identifica il gestore risorse a cui si applica la decisione commit o back out. Questo flag è valido solo per transazioni coordinate internamente e per gestori risorse che non siano più configurati nel file *qm.ini* del Queue Manager. Il risultato sarà congruente con la decisione raggiunta da MQSeries per la transazione.

#### *Transaction*

è il numero di transazione della transazione di cui si sta eseguendo il commit o il back out. Per rilevare il relativo numero di transazione, utilizzare il comando **dspmqrn**. Questo parametro è necessario con i parametri *RMIId* -b, -c e -r.

### Codici di errore

- 0 operazione eseguita
- 32 Non è stato possibile risolvere le transazioni

34	Gestore risorse non riconosciuto
35	Gestore risorse non disponibile permanentemente
36	Elemento immesso non valido
40	Gestore code non disponibile
49	Queue Manager in chiusura
69	Memoria non disponibile
71	Errore imprevisto
72	errore relativo al nome del Queue Manager
85	Transazioni sconosciute

## **Comandi correlati**

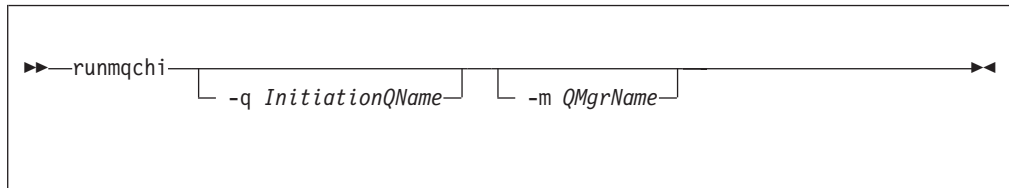
<b>dspmqtrn</b>	Visualizza elenco di transazioni preparate
-----------------	--

## runmqchi (Run channel initiator)

### Scopo

Utilizzare il comando **runmqchi** per eseguire un processo di inizializzazione di un canale. Per ulteriori informazioni relative all'utilizzo di questo comando, consultare il manuale *MQSeries Intercommunication*.

### Sintassi



### Parametri opzionali

**-q** *InitiationQName*

Specifica il nome della coda da inizializzare elaborandola con channel initiator. Se non è specificato, viene utilizzato SYSTEM.CHANNEL.INITQ.

**-m** *QMgrName*

Specifica il nome del Queue Manager sul quale esiste la coda di iniziazione. Se viene omesso il nome, viene utilizzato Queue Manager predefinito.

### Codici di errore

- 0 Comando completato normalmente
- 10 Comando completato con risultato inatteso
- 20 Si è verificato un errore durante l'elaborazione

Se si verificano gli errori riportati dai codici di errore 10 o 20 è necessario rivedere il log di errore Queue Manager associato al canale per i messaggi di errore. Inoltre è necessario rivedere il log di errore del \$SISTEMA, poiché è lì che vengono registrati i problemi che si verificano prima dell'associazione del canale al Queue Manager. Per ulteriori informazioni relative ai log di errore, consultare "Log di errore" a pagina 209.

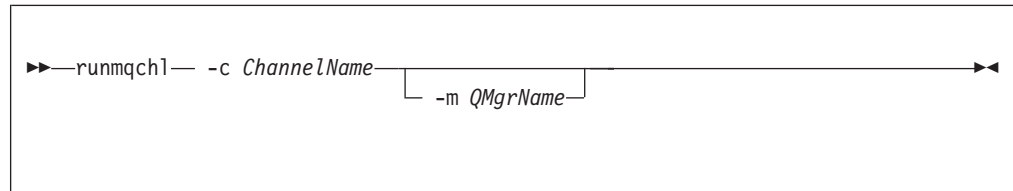
## runmqchl (Run channel)

### Scopo

Utilizzare il comando **runmqchl** per eseguire un Sender (SDR), un Requester (RQSTR).

Il canale esegue in modo sincronico. Per arrestare il canale, immettere il comando MQSC STOP CHANNEL.

### Sintassi



### Parametri richiesti

**-c ChannelName**  
Specifica il nome del canale da eseguire.

### Parametri opzionali

**-m QMgrName**  
Specifica il nome del Queue Manager al quale è associato il canale. Se non viene specificato alcun nome, viene utilizzato il Queue Manager predefinito.

### Codici di errore

**0** Comando completato normalmente  
**10** Comando completato con risultato inatteso  
**20** Si è verificato un errore durante la procedura

Se viene riportato il codice di errore 10 o 20, rivedere il log di errore del Queue Manager associato ai messaggi di errore. Inoltre è necessario rivedere il log degli errori \$SYSTEM poiché è lì che vengono registrati i problemi che si verificano prima dell'associazione del canale al Queue Manager.

## runmqdlq (Run dead-letter queue handler)

### Scopo

Utilizzare il comando **runmqdlq** per avviare il DLQ (Dead-Letter Queue) handler, un'utilità che consente il controllo e la gestione dei messaggi nelle code messaggi non recapitati.

Il DLQ handler può essere utilizzato per eseguire diverse azioni sui messaggi selezionati, specificando, con un gruppo di regole, sia il messaggio da selezionare che l'azione da eseguire.

Il comando **runmqdlq** riceve i suoi input da SYS\$INPUT. Quando il comando viene elaborato, i risultati e un riepilogo vengono inseriti in un rapporto inviato a SYS\$OUTPUT.

Ricevendo SYS\$INPUT dalla tastiera, è possibile immettere in modo interattivo le regole **runmqdlq**.

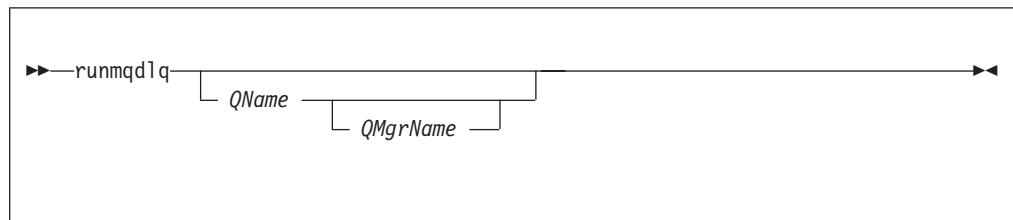
Reindirizzando l'input da un file, è possibile applicare una tabella regole alla coda specificata. La tabella regole deve contenere almeno una regola

Se il DLQ handler è utilizzato in modalità primo piano senza reindirizzare SYS\$INPUT da un file, (la tabella regole) il DLQ handler:

- Legge i propri input dalla tastiera.
- Non avvia l'elaborazione delle code specificate se prima non riceve un carattere end\_of\_file (ctrl-Z).

Per ulteriori informazioni sulle tabelle regole e sulla loro compilazione, consultare "Tabella regole DLQ handler" a pagina 108.

### Sintassi



### Parametri opzionali

Le regole MQSC per le linee di commento e le linee di collegamento vengono applicate anche ai parametri di input DLQ handler.

#### *QName*

Specifica il nome della coda da elaborare.

Se non è specificato alcun nome, viene utilizzata la coda DLQ predefinita per il Queue Manager locale. Utilizzando uno o più spazi vuoti ( ' ') si assegna esplicitamente la coda DLQ del Queue Manager locale.

Un DLQ handler consente di selezionare determinati messaggi presenti nella coda DLQ da destinare ad un'elaborazione particolare. Ad esempio, è possibile reindirizzare i messaggi ad un'altra coda messaggi non recapitati. In seguito ad

una nuova istanza del DLQ handler, i messaggi potranno essere elaborati successivamente in base ad una diversa tabella regole.

***QMgrName***

Specifica il nome del Queue Manager che gestisce la coda da elaborare.

Se non viene specificato alcun nome, viene utilizzato il Queue Manager predefinito. Utilizzando uno o più spazi vuoti ( ' ') si assegna esplicitamente il Queue Manager per questa installazione.

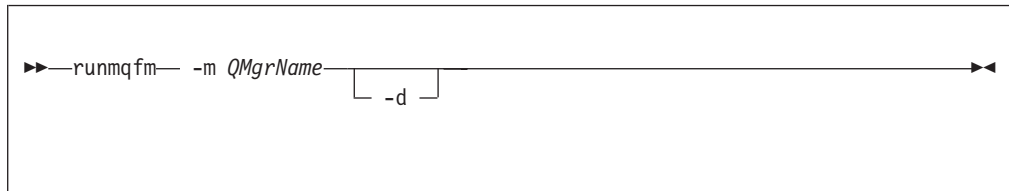
---

## runmqfm (Avvia un monitor di ripristino)

### Scopo

Utilizzare il comando **runmqfm** per avviare il monitor di ripristino su un nodo di OpenVMS. Il monitor di ripristino viene eseguito sul nodo di OpenVMS su cui viene eseguito il comando **runmqfm**.

### Sintassi



### Parametri richiesti

**-m** *QMgrName*

Specifica il nome del Queue Manager per il quale occorre avviare il comando **runmqfm**. La lunghezza massima supportata per il nome *QMgrName* è di 25 caratteri.

### Parametri opzionali

**-d** Specifica che le informazioni di debug aggiuntive devono essere registrate nel file di log.

### Codici di errore

- 0 Comando completato normalmente
- 10 Comando completato con risultato inatteso
- 20 Si è verificato un errore durante l'elaborazione

### Esempi

Il seguente esempio avvia un monitor di ripristino per un Queue Manager denominato `testqm` e scrive informazioni di debug in un file log denominato `test.log`.

```
runmqfm -m "testqm" -d > test.log
```

### Comandi correlati

**ripristino** Gestisci un gruppo di ripristino.

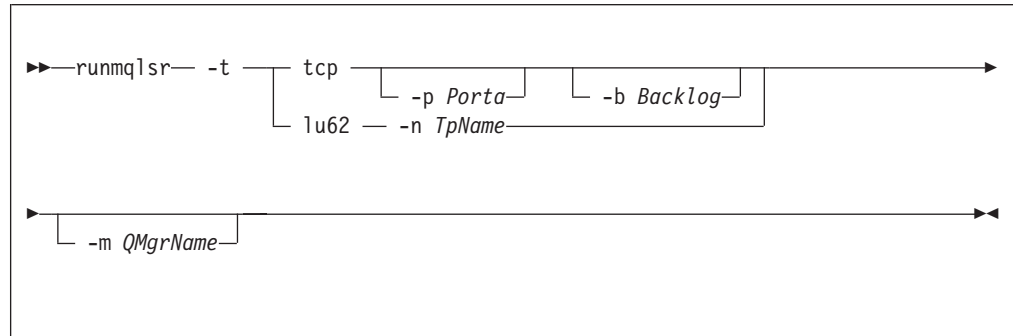


## runmqlsr (Run listener)

### Scopo

Il comando **runmqlsr** (Run listener) esegue un processo listener.

### Sintassi



### Parametri richiesti

- t** Specifica il protocollo di trasmissione da utilizzare:
  - tcp** TCP/IP (Transmission Control Protocol / Internet Protocol)
  - lu62** SNA LU 6.2. Per informazioni aggiornate sull'utilizzo di questo parametro, consultare le note correlate in `sys$help:mqseries0510.release_notes` .

### Parametri opzionali

- p Port**  
Numero di porta per TCP/IP. Questa flag è valida per TCP e UDP. Se non è specificato alcun valore, ne viene acquisito uno dal file di configurazione Queue Manager o da quelli predefiniti dal programma. Il valore predefinito è 1414.
- n TpName**  
LU 6.2 nome del programma di transazione. Questa flag è valida solo per il protocollo di trasmissione LU 6.2. Se non è specificato alcun valore, ne viene acquisito uno dal file di configurazione Queue Manager. Se manca questa impostazione il comando ha esito negativo.
- m QMgrName**  
specifica il nome del Queue Manager. Se non è specificato alcun nome, il comando viene eseguito sul Queue Manager predefinito.
- b Backlog**  
Specifica il numero delle richieste di connessione contemporanee supportate dal listener. Consultare "Le voci LU62 e TCP" a pagina 193 per un elenco di valori predefiniti, e per ulteriori informazioni.

### Codici di errore

- 0 Comando completato normalmente
- 10 Comando completato con risultato inatteso
- 20 Si è verificato un errore durante l'elaborazione

## runmqsr

### Esempi

Questo comando esegue un listener sul Queue Manager predefinito utilizzando il protocollo TCP/IP. Il comando specifica che il listener deve utilizzare la porta 4321.

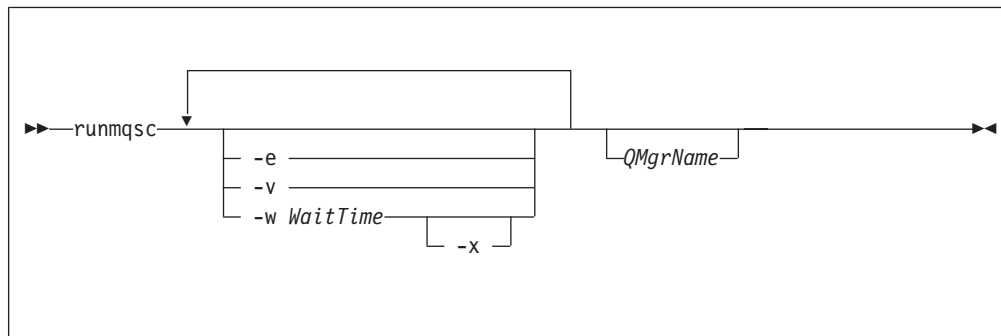
```
runmqsr -t tcp -p 4321
```

## runmqsc (Esegue comandi di MQSeries)

### Scopo

Utilizzare il comando **runmqsc** per emettere comandi MQSC a un Queue Manager. I comandi MQSC consentono di eseguire attività amministrative, come ad esempio definire, alterare o eliminare un oggetto coda locale. I comandi MQSC e la loro sintassi sono descritti nel volume *MQSeries Application Programming Guide*.

### Sintassi



### Descrizione

È possibile richiamare il comando **runmqsc** in tre modi:

#### Verify mode (modo verifica)

I comandi MQSC vengono verificati, ma non effettivamente eseguiti. Viene generato un prospetto di output che indica l'esito positivo o negativo di ciascun comando. Questo modo è disponibile solo su un Queue Manager locale.

#### Direct mode (modo diretto)

I comandi MQSC sono inviati direttamente al Queue Manager locale.

#### Indirect mode (modo indiretto)

I comandi MQSC sono eseguiti su un Queue Manager remoto. Questi comandi sono inseriti nella coda di comandi di un Queue Manager remoto e vengono eseguiti nell'ordine di accodamento. I prospetti dai comandi vengono restituiti al Queue Manager locale.

**Nota:** L'ID utente che esegue il Queue Manager remoto deve esistere localmente e disporre delle autorizzazioni corrette.

Il comando **runmqsc** riceve l'input da SYS\$INPUT. Quando i comandi vengono elaborati, i risultati e il riepilogo vengono inseriti in un prospetto che sarà inviato a SYS\$OUTPUT.

Prelevando SYS\$INPUT dalla tastiera, sarà possibile immettere i comandi MQSC in maniera interattiva.

Reindirizzando l'input da un file sarà possibile eseguire una sequenza di comandi frequentemente utilizzati nel file. È anche possibile reindirizzare il prospetto di output a un file.

## Parametri opzionali

- e Impedisce che il testo d'origine relativo ai comandi MQSC sia copiato in un prospetto. Ciò è utile quando si immettono comandi in maniera interattiva.
- v Specifica il modo di verifica; ciò verifica i comandi specificati senza eseguire le azioni. Questo modo è disponibile solo localmente. I flag -w e -x saranno ignorati se specificati simultaneamente.

### -w *WaitTime*

Specifica il modo indiretto, e cioè che i comandi MQSC dovranno essere eseguiti su un altro Queue Manager. È necessario aver impostato a tale scopo le code di canale e trasmissione necessarie. Consultare "Preparazione di canali e code di trasmissione per la gestione remota" a pagina 72 per ulteriori informazioni

#### *WaitTime*

Specifica il tempo, in secondi, che **runmqsc** attende per le risposte. Qualsiasi risposta ricevuta dopo tale intervallo vengono eliminate, tuttavia i comandi MQSC saranno eseguiti ugualmente. Specificare un intervallo tra 1 e 999 999 secondi.

Ciascun comando viene inviato come PCF Escape alla coda di comando (SYSTEM.ADMIN.COMMAND.QUEUE) del Queue Manager di destinazione.

Le risposte saranno ricevute sulla coda SYSTEM.MQSC.REPLY.QUEUE e il risultato verrà aggiunto al prospetto. Ciò può essere definito come coda locale o coda modello.

Le operazioni in modo indiretto vengono eseguite tramite il Queue Manager predefinito.

Questo flag è ignorato se viene specificato il flag -v.

- x Specifica che il Queue Manager di destinazione è eseguito sotto MVS/ESA. Questo flag si applica solo in modo indiretto. Occorre specificare anche il flag -w. In modo indiretto, i comandi MQSC sono scritti in una forma adeguata alla coda di comandi di MQSeries per MVS/ESA.

### *QMgrName*

Specifica il nome del Queue Manager di destinazione sul quale eseguire i comandi MQSC. Se omissivo, i comandi MQSC saranno eseguiti sul Queue Manager predefinito.

## Codici di errore

- 0 File di comandi MQSC elaborato con esito positivo.
- 10 File di comandi MQSC elaborato con errori—il prospetto contiene i motivi del malfunzionamento dei comandi.
- 20 Errore—file di comandi MQSC non eseguito.

## Esempi

1. Digitare questo comando nel prompt dei comandi di OpenVMS:

```
runmqsc
```

Ora è possibile digitare i comandi MQSC direttamente nel prompt dei comandi di OpenVMS. Non è stato specificato nessun nome di Queue Manager e pertanto i comandi MQSC saranno elaborati sul Queue Manager predefinito.

2. Utilizzare questo comando per specificare che i comandi MQSC sono solo verificati:

```
runmqsc -v BANK < DKA0:[USERS]COMMFILE.IN
```

Ciò verifica il file di comandi MQSC COMMFILE.IN nella directory DKA0:[USERS]. Il nome del Queue Manager è BANK. L'output è visualizzato nella finestra corrente.

3. Questo comando esegue il file di comandi MQSC MQS\_ROOT:[MQM.MQSC]MQSCFILE.IN verso il Queue Manager predefinito.

```
runmqsc < MQS_ROOT:[MQM.MQSC]MQSCFILE.IN > MQS_ROOT:[MQM.MQSC]MQSCFILE.OUT
```

In questo esempio, l'output è diretto al file MQS\_ROOT:[MQM.MQSC]MQSCFILE.OUT.

---

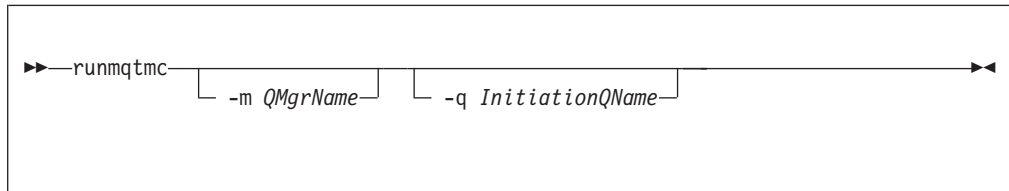
## runmqtmc (Avvia trigger monitor del client)

### Scopo

Utilizzare il comando **runmqtmc** per invocare un trigger monitor per un client. Per ulteriori informazioni sull'utilizzo di trigger monitor, consultare *MQSeries Application Programming Guide*.

**Nota:** Questo comando è disponibile *solo* sui client OpenVMS, OS/2 e AIX.

### Sintassi



### Parametri opzionali

**-m** *QMgrName*

Specifica il nome del Queue Manager su cui opera il trigger monitor del client. Se omissa, il trigger monitor del client opera sul Queue Manager predefinito.

**-q** *InitiationQName*

Specifica il nome della coda di avvio da elaborare. Se omissa, viene utilizzato SYSTEM.DEFAULT.INITIATION.QUEUE.

### Codici di errore

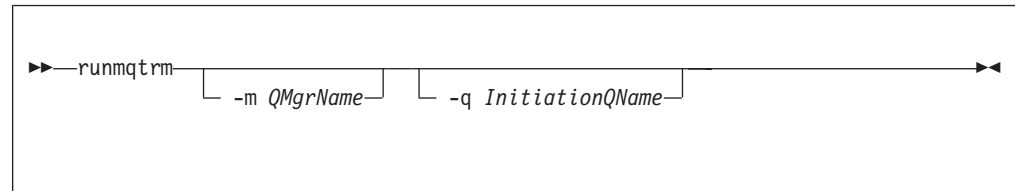
- 0 Non utilizzato. Il trigger monitor del client è progettato per essere eseguito in continuazione e quindi non per essere arrestato. Il valore è riservato.
- 10 Trigger monitor del client interrotto per un errore.
- 20 Errore— trigger monitor del client non eseguito.

## runmqtrm (Avvia un trigger monitor)

### Scopo

Utilizzare il comando **runmqtrm** per invocare un trigger monitor. Per ulteriori informazioni sull'utilizzo di trigger monitor, consultare *MQSeries Application Programming Guide*.

### Sintassi



### Parametri opzionali

**-m** *QMgrName*

Specifica il nome del Queue Manager su cui opera il trigger monitor. Se omissa, il trigger monitor opera sul Queue Manager predefinito.

**-q** *InitiationQName*

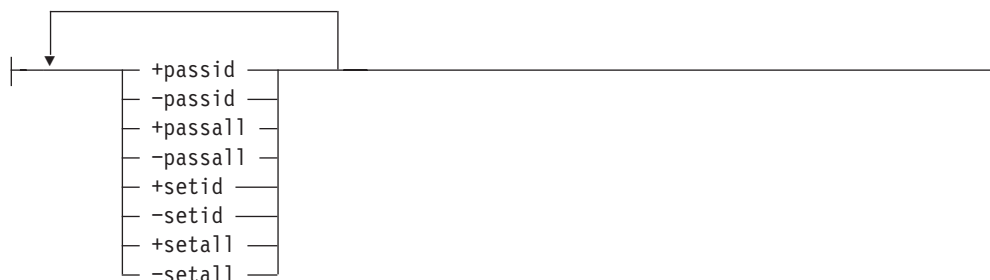
Specifica il nome della coda di avvio da elaborare. Se omissa, viene utilizzato SYSTEM.DEFAULT.INITIATION.QUEUE.

### Codici di errore

- 0 Non utilizzato. Il trigger monitor è progettato per essere eseguito in continuazione e quindi non per essere arrestato. Il valore è riservato.
- 10 Trigger monitor interrotto per un errore.
- 20 Errore— trigger monitor non eseguito.





**Autorizzazioni amministrative:****Autorizzazioni generiche:****Descrizione**

È possibile utilizzare questo comando sia per *impostare* un'autorizzazione, cioè accordare a un gruppo utente o a un principal il permesso di eseguire un'operazione, che per *reimpostare* un'autorizzazione, cioè rimuovere il permesso di eseguire un'operazione. È necessario specificare i gruppi utente e i principal a cui applicare le autorizzazioni, oltre al Queue Manager, al tipo e al nome dell'oggetto. È possibile specificare qualsiasi numero di gruppi e principal in un unico comando.

**Attenzione:** se si specifica un insieme di autorizzazioni per un principal, le stesse saranno concesse a tutti i principal dello stesso gruppo primario.

Le autorizzazioni che è possibile concedere si suddividono nelle seguenti categorie:

- Autorizzazioni di emissione di chiamate MQI
- Autorizzazioni per il contesto MQI

## setmqaut

- Autorizzazioni di emissione di comandi per attività di amministrazione
- Autorizzazioni generiche

Ogni autorizzazione da modificare viene specificata in un elenco di autorizzazioni come parte del comando. Ogni voce dell'elenco è una stringa preceduta da '+' o '-'. Ad esempio, includendo +put nella lista delle autorizzazioni, si conferisce l'autorizzazione a emettere chiamate MQPUT verso una coda. In alternativa, includendo -put nell'elenco di autorizzazioni, si rimuove l'autorizzazione a emettere chiamate MQPUT.

È possibile specificare le autorizzazioni in qualsiasi ordine, a condizione che non entrino in conflitto. Ad esempio, la specificazione di allmqi insieme a set provoca un conflitto.

In un unico comando, è possibile specificare quanti gruppi o autorizzazioni sono necessari.

Se un ID utente è membro di più gruppi, le autorizzazioni applicate sono l'unione delle autorizzazioni di ogni gruppo a cui l'ID utente appartiene.

## Parametri richiesti

**-m** *QMgrName*

Specifica il nome del Queue Manager dell'oggetto per il quale si devono modificare le autorizzazioni. Il nome può contenere fino a 48 caratteri.

**-n** *ObjectName*

Specifica il nome dell'oggetto per il quale si devono modificare le autorizzazioni.

Questo parametro è necessario *a meno che* non sia lo stesso Queue Manager. È necessario specificare un nome di Queue Manager, coda o processo, non utilizzando un nome generico.

**-t** *ObjectType*

Specifica il tipo di oggetto per il quale si devono modificare le autorizzazioni.

I valori possibili sono:

- **q** o **coda**
- **prcs** o **processo**
- **qmgr**

## Parametri opzionali

**-p** *PrincipalName*

Specifica il nome del principal per il quale si devono modificare le autorizzazioni.

È necessario disporre almeno di un principal o un gruppo.

**-g** *GroupName*

Specifica il nome dell'identificativo diritti che rappresenta il gruppo utente le cui autorizzazioni occorre modificare. È possibile specificare più nomi di identificativo diritti, ma ciascuno deve essere preceduto dal flag -g.

**-s** *ServiceComponent*

Questo parametro si applica solo se si utilizzano servizi installabili di autorizzazione, altrimenti viene ignorato.

Se tali servizi sono supportati, questo parametro specifica il nome del servizio di autorizzazione a cui si applicano le autorizzazioni. Questo parametro è facoltativo; se non viene specificato, l'aggiornamento dell'autorizzazione viene eseguito sul primo componente installabile per il servizio.

#### Autorizzazioni

Specifica le autorizzazioni da concedere o rimuovere. Ogni voce dell'elenco è preceduta da un segno '+' che indica l'autorizzazione da concedere, o un segno '-', che indica l'autorizzazione da rimuovere. Ad esempio, per concedere l'autorizzazione per emettere una chiamata MQPUT dal MQI, specificare +put nell'elenco. Per rimuovere tale autorizzazione, specificare -put.

Tabella 17 mostra le autorizzazioni che si possono concedere ai diversi tipi di oggetto.

Tabella 17. Specificazione delle autorizzazioni per i diversi tipi di oggetto

Autorizzazione	Coda	Processo	Qmgr	Namelist
tutto	✓	✓	✓	✓
alladm	✓	✓	✓	✓
allmqi	✓	✓	✓	✓
altusr			✓	
browse	✓			
chg	✓	✓	✓	✓
clr	✓			
connect			✓	
crt	✓	✓	✓	✓
dlt	✓	✓	✓	✓
dsp	✓	✓	✓	✓
get	✓			
inq	✓	✓	✓	✓
passall	✓			
passid	✓			
put	✓			
set	✓	✓	✓	
setall	✓		✓	
setid	✓		✓	

#### Autorizzazioni per le chiamate MQI

**altusr** Utilizza un ID utente alternativo in un messaggio.

Vedere *MQSeries Application Programming Guide* per ulteriori informazioni sugli ID utente alternativi.

#### sfoglia

Richiama un messaggio da una coda emettendo una chiamata MQGET con l'opzione BROWSE.

#### connect

Collega l'applicazione al Queue Manager specificato emettendo una chiamata MQCONN.

**get** Richiama un messaggio da una coda emettendo una chiamata MQGET.

## setmqaut

- inq** Effettua una richiesta su una specifica coda emettendo una chiamata MQINQ.
- put** Invia un messaggio su una specifica coda emettendo una chiamata MQPUT.
- set** Imposta attributi su una coda da MQI emettendo una chiamata MQSET.

**Nota:** Se si apre un coda per più opzioni, occorre essere autorizzati per ognuna di esse.

### Autorizzazioni per il contesto

- passall** Sposta tutto il contesto sulla coda specificata. Tutti i campi del contesto vengono copiati dalla richiesta originale.
- passid** Passa il contesto d'identità sulla coda specificata. Il contesto d'identità è lo stesso della richiesta.
- setall** Imposta tutto il contesto sulla coda specificata. È utilizzato da speciali utilità di sistema.
- setid** Imposta il contesto d'identità sulla coda specificata. È utilizzato da speciali utilità di sistema.

### Autorizzazioni per i comandi

- chg** Modifica gli attributi dell'oggetto specificato.
- clr** Cancella la coda specificata (solo comando PCF Clear queue).
- crt** Crea oggetti del tipo specificato.
- dlt** Cancella l'oggetto specificato.
- dsp** Visualizza gli attributi dell'oggetto specificato.

### Autorizzazioni per operazioni generiche

- tutto** Utilizza tutte le operazioni applicabili all'oggetto.
- alladm** Esegue tutte le operazioni di amministrazione applicabili all'oggetto.
- allmqi** Utilizza tutte le chiamate MQI applicabili all'oggetto.

## Codici di errore

- 0 Comando completato normalmente
- 36 Elemento immesso non valido
- 40 Gestore code non disponibile
- 49 Queue Manager in chiusura
- 69 Memoria non disponibile
- 71 Errore imprevisto
- 72 errore relativo al nome del Queue Manager
- 133 Nome oggetto sconosciuto
- 145 Nome oggetto imprevisto
- 146 Nome oggetto assente
- 147 Tipo oggetto assente
- 148 Tipo oggetto invalido
- 149 Nome entità mancante
- 150 Specificazione di autorizzazione assente
- 151 Specificazione di autorizzazione non valida

## Esempi

1. Questo esempio mostra un comando che specifica che l'oggetto a cui sono state concesse le autorizzazioni è la coda orange.queue sul Queue Manager saturn.queue.manager.

```
setmqaut -m "saturn.queue.manager" -n "orange.queue" -t queue -g "tango" +inq +alladm
```

Le autorizzazioni sono state concesse al gruppo utente tango e l'elenco autorizzazioni associato specifica che il gruppo utente tango:

- Può emettere chiamate MQINQ.
  - Dispone dell'autorizzazione a eseguire tutte le operazioni di amministrazione su quell'oggetto.
2. In questo esempio, l'elenco autorizzazioni specifica che il gruppo utente foxy:
    - Non può emettere alcuna chiamata da MQI alla coda specificata.
    - Dispone dell'autorizzazione a eseguire tutte le operazioni di amministrazione sulla coda specificata.

```
setmqaut -m "saturn.queue.manager" -n "orange.queue" -t queue -g "foxy" -allmqi +alladm
```

3. In questo esempio, l'elenco autorizzazioni specifica che il gruppo utente waltz dispone dell'autorizzazione a creare ed eliminare il Queue Manager saturn.queue.manager:

```
setmqaut -m "saturn.queue.manager" -t qmgr -g "waltz" +crt +dlt
```

**setmqaut**

## **Comandi correlati**

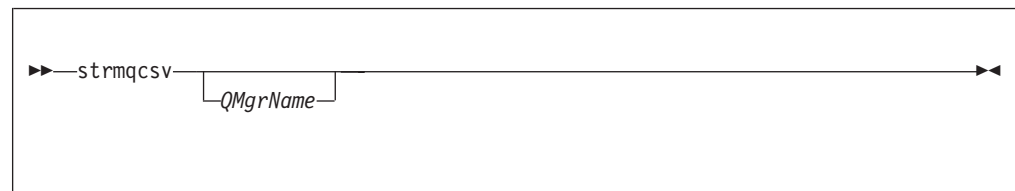
**dspmqaut** Visualizza l'autorizzazione

## strmqcsv (Avvia server di comandi)

### Scopo

Utilizzare il comando **strmqcsv** per avviare il server di comandi per il Queue Manager specificato. Ciò abilita MQSeries all'elaborazione dei comandi inviati alla coda di comando.

### Sintassi



### Parametri opzionali

*QMgrName*

Specifica il nome del Queue Manager per il quale avviare il server di comandi.

### Codici di errore

- 0        Comando completato normalmente
- 10      Comando completato con risultato inatteso
- 20      Si è verificato un errore durante l'elaborazione

### Esempi

Il seguente comando avvia un server di comandi per il Queue Manager earth:

```
strmqcsv "earth"
```

### Comandi correlati

- endmqcsv**      Chiude un server di comandi
- dspmqcsv**     Visualizza lo stato di un server di comandi

---

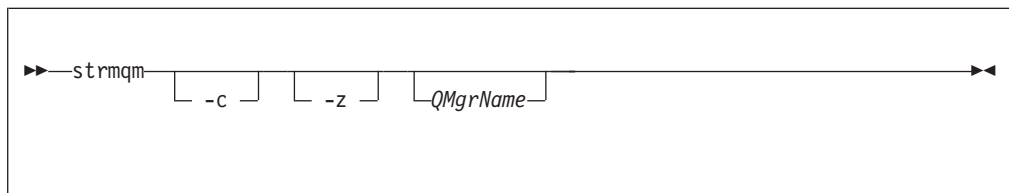
## strmqm (Avvio Queue Manager)

### Scopo

Utilizzare il comando **strmqm** per avviare un Queue Manager locale.

**Nota:** Prima di utilizzare il comando **strmqm** o qualsiasi altro comando di controllo, occorre eseguire `mqs_startup` una volta dall'ultimo riavvio prima che il Queue Manager possa essere riavviato ed eseguito correttamente.

### Sintassi



### Parametri opzionali

**-c** Avvia il Queue Manager, ridefinisce gli oggetti di sistema predefiniti, quindi arresta il Queue Manager (gli oggetti di sistema predefiniti per un Queue Manager vengono creati inizialmente dal comando **crmqm**). Qualsiasi oggetto di sistema predefinito appartenente al Queue Manager viene sostituito se si specifica questo flag.

*QMgrName*

Specifica il nome di un Queue Manager locale da avviare. Se non si specifica un nome, viene avviato il Queue Manager predefinito.

**-z** Sopprime i messaggi di errore.

Questo flag viene utilizzato in MQSeries per eliminare i messaggi di errore indesiderati. Poiché l'utilizzo di questo flag comporta la perdita di dati, si consiglia di non utilizzarlo quando si immettono comandi in una riga comandi.

### Codici di errore

0	Queue Manager avviato
3	Creazione di Queue Manager in corso
5	Esecuzione di Queue Manager
16	Il Queue Manager non esiste
23	Log non disponibile
49	Queue Manager in chiusura
69	Memoria non disponibile
71	Errore imprevisto
72	Errore relativo al nome del gestore code
100	Posizione del log invalida

### Esempi

I seguenti comandi avviano l'account del Queue Manager:



```
strmqm "account"
```

## Comandi correlati

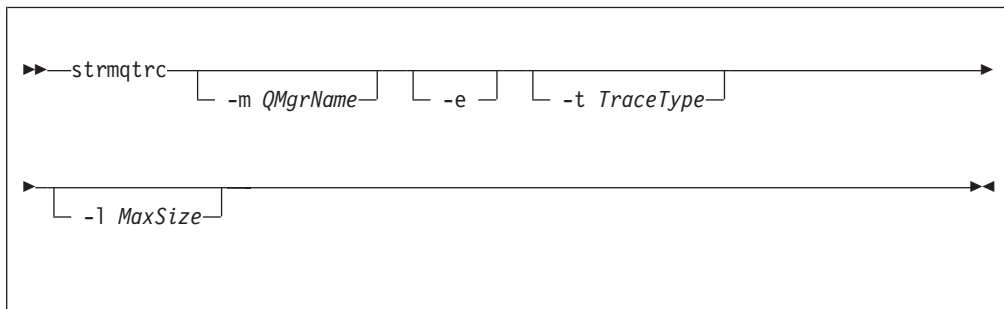
<b>crtmqm</b>	Creare un Queue Manager
<b>dltmqm</b>	Eliminare un Queue Manager
<b>endmqm</b>	Terminare un Queue Manager

## strmqtrc (Avvio traccia di MQSeries)

### Scopo

Utilizzare il comando **strmqtrc** per abilitare la traccia. L'esecuzione di tale comando è possibile sia in caso di traccia abilitata che non abilitata. Se la traccia è già abilitata, le opzioni di traccia vengono infatti modificate in quelle specificate nel più recente richiamo del comando.

### Sintassi



### Descrizione

È possibile richiedere differenti livelli di dettaglio di traccia. Per ogni valore di tracetype di flusso specificato, includendo -t in modo completo, specificando per qualsiasi particolare tipo di traccia sia i parametri -t che il dettaglio -t, sarà generato solo un dettaglio di traccia predefinito per quel tipo di traccia.

Per alcuni esempi di dati di traccia generati da questo comando, vedere "Utilizzo della traccia MQSeries" a pagina 214.

### Parametri opzionali

#### **-m** *QMgrName*

È il nome del Queue Manager da tracciare.

È possibile specificare per lo stesso comando un nome di Queue Manager e il flag -m allo stesso modo del flag -e. Se a una data entità da tracciare si applica più di una specifica di traccia, la traccia effettiva include tutte le opzioni specificate.

L'omissione del flag -m e del nome del Queue Manager provoca un errore, a meno che non si specifichi il flag -e.

- e** Se questo flag è specificato, è necessaria una traccia iniziale. Di conseguenza, è possibile tracciare la creazione o l'avvio di un Queue Manager. Ciò implica la scrittura dell'informazione di traccia prima che i processi riconoscano a quale componente MQSeries appartengono. Qualsiasi processo, appartenente a qualsiasi componente di qualsiasi Queue Manager, traccia la propria elaborazione iniziale se tale flag è specificato. Per impostazione predefinita, se il flag non è specificato, non sarà eseguita la traccia iniziale.

#### **-t** *TraceType*

Definisce quali punti possono essere tracciati durante l'elaborazione.

L'omissione di questo flag comporta l'abilitazione di tutti i punti di traccia e la generazione di una traccia completa.

Alternativamente, è possibile fornire una o più opzioni nel seguente elenco.

**Nota:** Se sono disponibili più tipi di traccia, ciascuna *deve* disporre del proprio flag -t. È possibile specificare qualsiasi quantità di flag -t, a condizione che a ciascuno sia associato un tipo di traccia valido.

Non è un errore specificare lo stesso tipo di traccia su molteplici flag -t.

<b>tutto</b>	Dati di output per ogni punto di traccia nel sistema. Questo è anche il valore predefinito se il flag -t non è specificato.
<b>api</b>	Dati di output per i punti di traccia associati ai componenti MQI e al principale Queue Manager.
<b>comms</b>	Dati di output per i punti di traccia associati ai dati trasmessi sulle reti di comunicazione.
<b>csflows</b>	Dati di output per i punti di traccia associati al flusso di elaborazione nei servizi comuni.
<b>lqmflows</b>	Dati di output per i punti di traccia associati al flusso di elaborazione nel Queue Manager locale.
<b>remoteflows</b>	Dati di output per i punti di traccia associati al flusso di elaborazione nei componenti di comunicazione.
<b>otherflows</b>	Dati di output per i punti di traccia associati al flusso di elaborazione in altri componenti.
<b>csdata</b>	Dati di output per i punti di traccia associati ai buffer di dati interni nei servizi in comune.
<b>lqmdata</b>	Dati di output per i punti di traccia associati ai buffer di dati interni nel Queue Manager locale.
<b>remotedata</b>	Dati di output per i punti di traccia associati ai buffer di dati interni nei componenti di comunicazione.
<b>otherdata</b>	Dati di output per i punti di traccia associati ai buffer di dati interni in altri componenti.
<b>versiondata</b>	Dati di output per i punti di traccia associati alla versione di MQSeries in esecuzione.
<b>commentary</b>	Dati di output per i punti di traccia associati ai commenti nei componenti MQSeries.

#### **-l *MaxSize***

Il valore di *MaxSize* indica la massima dimensione in milioni di byte di un file di traccia (AMQnnnn.TRC). Ad esempio, specificando per *MaxSize* il valore 1, la dimensione della traccia è limitata a un milione di byte.

Quando un file di traccia raggiunge la massima dimensione specificata, viene rinominato da AQnnnn.TRC a AMQnnnn.TRS e viene avviato un nuovo file AMQnnnn.TRC. Tutti i file di traccia vengono riavviati quando è raggiunta la massima dimensione. Verrà cancellata ogni copia preesistente del file AMQnnnn.TRS.

## **Codici di errore**

<b>AMQ7024</b>	Questo messaggio è visualizzato se vengono forniti al comando argomenti non validi.
<b>AMQ8304</b>	Il numero massimo di nove tracce simultanee è già in esecuzione.

## **Esempi**

Questo comando abilita la traccia di dati dai servizi comuni e dal Queue Manager locale, per un Queue Manager denominato QM1.

## strmqtrc

```
strmqtrc -m QM1 -t csdata -t lqmdata
```

### Comandi correlati

<b>dspmqtrc</b>	Visualizzazione di output di traccia formattata
<b>endmqtrc</b>	Fine traccia di MQSeries

---

## Parte 3. Appendici



---

## Appendice A. MQSeries per Compaq OpenVMS al primo sguardo

---

### Numero programma e parte

- 5724-A38 MQSeries per Compaq OpenVMS, Alpha Versione 5 Rilascio 1, numero parte 0790997.

---

### Requisiti hardware

I server di MQSeries possono essere qualsiasi computer Compaq Alpha con uno spazio su disco di sistema di minimo 128 MB.

---

### Requisiti software

I requisiti software sono identici per ambienti di server e client Compaq OpenVMS a meno che altrimenti dichiarato.

I livelli minimi di supporto sono mostrati:

- Compaq OpenVMS Alpha Versione 7.2-1.

---

### Connettività

MQSeries per Compaq OpenVMS supporta i seguenti protocolli di rete e hardware:

Protocolli di rete:

- SNA LU6.2
- TCP/IP
- DECnet Phase V

E qualsiasi hardware di comunicazione che supporti DECnet o TCP/IP, o DIGITAL DECnet/SNA Gateway.

**Per la connettività DECnet:**

- DECnetPLUS Versione 7.1 per OpenVMS Versione 7.2-1

**Per la connettività TCP/IP:**

- DIGITAL TCP/IP Services per OpenVMS AlphaV5.0.a e V5.1
- Process Software's TCPWare V5.4
- Process Software's Multinet V4.3

**Per la connettività SNA:** occorre installare il software e la licenza di SNA APPC LU6.2. Questo dovrà disporre di accesso a un gateway SNA adeguatamente configurato.

- DECnet SNA Gateway ST V1.3
- DECnet SNA LU6.2 API V2.4

---

### Sicurezza

MQSeries per Compaq OpenVMS utilizza le funzioni di sicurezza di Object Authority Manager (OAM) per MQSeries per Compaq OpenVMS.

## Sicurezza

Tutte le risorse di MQSeries sono eseguite con VMS Rights Identifier MQM. Questo identificativo diritti viene creato durante l'installazione di MQSeries ed è necessario concedere l'identificativo diritti con tale attributo di risorse a tutti gli utenti che dovranno controllare le risorse di MQSeries.

---

## Funzioni di manutenzione

Funzioni di MQSeries con:

- L'interfaccia della riga comandi **runmqsc**.
- 

## Compatibilità

MQI per MQSeries per Compaq OpenVMS Alpha, V5.1 è compatibile con le applicazioni esistenti che eseguono la Versione 2.2.1.1.

## Compilatori supportati

È possibile scrivere i programmi mediante C, C++, COBOL o Java.

- I programmi in C possono utilizzare il compilatore DEC C.
  - I programmi in C++ possono utilizzare il compilatore DEC C++.
  - I programmi in COBOL possono utilizzare il compilatore DEC COBOL.
  - I programmi in Java possono utilizzare il compilatore Java.
- 

## Selezione della lingua

Un file di testo di messaggi incluso è codificato nell'insieme di caratteri a 7 bit che è originario del sistema operativo OpenVMS.

---

## Internazionalizzazione

MQSeries per Compaq OpenVMS consente di specificare i CCSID quando viene creata l'istanza del Queue Manager. Il valore predefinito dei CCSID del Queue Manager è 819. MQSeries per Compaq OpenVMS supporta la conversione dell'insieme di caratteri nel CCSID configurato del Queue Manager. Per informazioni sui CCSID che è possibile specificare per un Queue Manager di MQSeries per Compaq OpenVMS, inclusi quelli che forniscono supporto per il carattere dell'euro, consultare il manuale *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa*.



## Appendice B. Impostazioni predefinite del sistema

Quando si crea un Queue Manager utilizzando il comando di controllo **crtmqm**, vengono automaticamente creati gli oggetti del sistema e gli oggetti predefiniti.

- Gli oggetti del sistema sono quelli di cui MQSeries necessita per il funzionamento di un Queue Manager o di un canale.
- Gli oggetti predefiniti definiscono tutti gli attributi di un oggetto. Quando si crea un oggetto, come ad esempio una coda locale, qualunque attributo omissso verrà sostituito dall'oggetto predefinito.

Tabella 18. Oggetti del sistema e oggetti predefiniti per le code

Nome Oggetto	Descrizione
SYSTEM.DEFAULT.ALIAS.QUEUE	Alias di coda predefinito.
SYSTEM.DEFAULT.LOCAL.QUEUE	Coda locale predefinita.
SYSTEM.DEFAULT.MODEL.QUEUE	Modello di coda predefinito.
SYSTEM.DEFAULT.REMOTE.QUEUE	Coda remota predefinito.
SYSTEM.DEAD.LETTER.QUEUE	Esempio di coda messaggi non recapitati DLQ.
SYSTEM.DEFAULT.INITIATION.QUEUE	Coda di avvio predefinita.
SYSTEM.CICS.INITIATION.QUEUE	Coda di avvio CICS <sup>®</sup> predefinita.
SYSTEM.ADMIN.COMMAND.QUEUE	Coda di comando gestione. Utilizzata per comandi MQSC remoti e comandi PCF.
SYSTEM.MQSC.REPLY.QUEUE	Coda di risposta RTQ (Reply-To-Queue) MQSC. Questo è un modello di coda che crea una coda dinamica temporanea per rispondere ai comandi MQSC remoti.
SYSTEM.ADMIN.QMGR.EVENT	Coda evento per eventi Queue Manager.
SYSTEM.ADMIN.PERFM.EVENT	Coda evento per eventi delle prestazioni.
SYSTEM.ADMIN.CHANNEL.EVENT	Coda evento per eventi canale.
SYSTEM.CHANNEL.INITQ	Coda di avvio canale.
SYSTEM.CHANNEL.SYNCQ	La coda che contiene la sincronizzazione dei dati per i canali.
SYSTEM.CLUSTER.COMMAND.QUEUE	La coda utilizzata per trasmettere messaggi all'archivio Queue Manager.
SYSTEM.CLUSTER.REPOSITORY.QUEUE	La coda utilizzata per memorizzare tutte le informazioni dell'archivio di memoria.
SYSTEM.CLUSTER.TRANSMIT.QUEUE	La coda di trasmissione per tutti messaggi verso i cluster.

Tabella 19. Oggetti del sistema e oggetti predefiniti per i canali

Nome Oggetto	Descrizione
SYSTEM.DEF.SENDER	Canale di invio predefinito.
SYSTEM.DEF.SERVER	Canale server predefinito.
SYSTEM.DEF.RECEIVER	Canale di ricezione predefinito.
SYSTEM.DEF.REQUESTER	Canale di richiesta predefinito.

## Impostazioni predefinite

Tabella 19. Oggetti del sistema e oggetti predefiniti per i canali (Continua)

Nome Oggetto	Descrizione
SYSTEM.DEF.SVRCONN	Canale predefinito per connessione server
SYSTEM.DEF.CLNTCONN	Canale predefinito per connessione client.
SYSTEM.AUTO.RECEIVER	Canale dinamico di ricezione.
SYSTEM.AUTO.SVRCONN	Canale dinamico per connessione al server.
SYSTEM.DEF.CLUSRCVR	Canale predefinito di ricezione per il cluster utilizzato per sostituire le impostazioni predefinite relative a qualsiasi attributo non specificato durante la creazione di un canale CLUSRCVR su un Queue Manager in un cluster.
SYSTEM.DEF.CLUSSDR	Canale predefinito di invio per il cluster utilizzato per sostituire le impostazioni predefinite relative a qualsiasi attributo non specificato durante la creazione di un canale CLUSSDR su un Queue Manager nel cluster.

Tabella 20. Oggetti del sistema e oggetti predefiniti per namelist

Nome Oggetto	Descrizione
SYSTEM.DEFAULT.NAMELIST	Namelist predefinito.

Tabella 21. Oggetti del sistema e oggetti predefiniti per processi.

Nome Oggetto	Descrizione
SYSTEM.DEFAULT.PROCESS	Processo predefinito di definizione.

## Appendice C. Struttura di directory

Figura 26 riporta lo schema generale delle directory di dati e di log associate ad un determinato Queue Manager. Le directory illustrate si riferiscono all'installazione predefinita. Se si modifica l'installazione, la posizione delle directory e dei file verrà conseguentemente modificata.

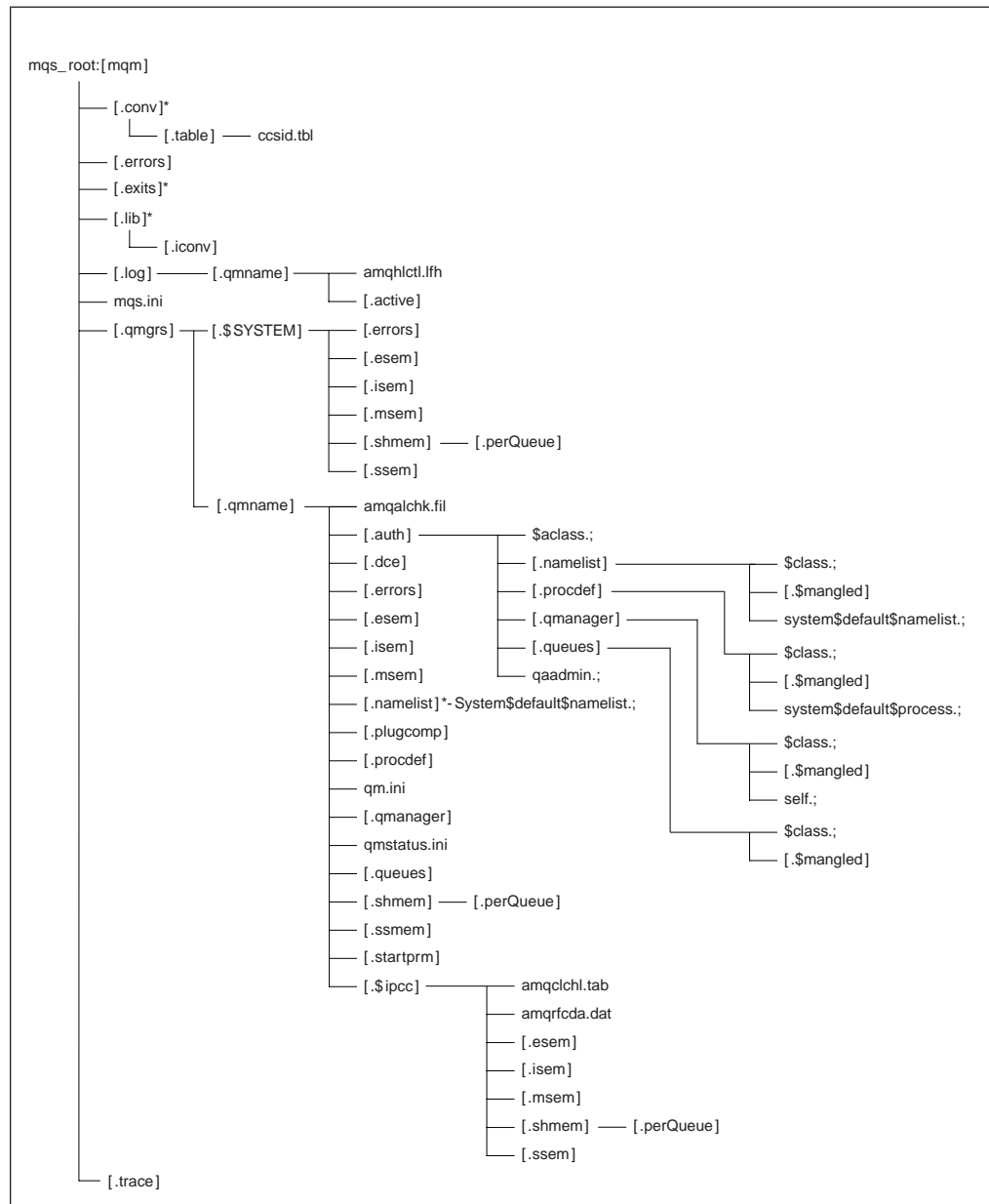


Figura 26. Struttura di directory predefinita dopo l'avvio di un Queue Manager

In Figura 26, lo schema rappresenta un Queue Manager MQSeries che è già stato utilizzato alcune volte. La struttura corrente dipende dalle operazioni relative al Queue Manager compiute.

### Directory e file in MQS\_ROOT:[MQM]

Per impostazione predefinita, le directory e file che seguono sono memorizzati nella directory MQS\_ROOT:[MQM]:

**.conv** Questa directory contiene tutti i file utilizzati per la conversione dei dati.

**.table** Questa directory contiene il file ccsid.tbl.

**.errors** Questa directory contiene i file dei messaggi per l'operatore, dal più recente al più remoto.

AMQERR01.LOG

AMQERR02.LOG

AMQERR03.LOG

**.exits** Una directory vuota che serve a contenere le uscite utente.

**.lib** Questa directory contiene le sottodirectory .iconv. La sottodirectory contiene tutte le tabelle per la conversione dei gruppi di dati.

**.iconv** Una directory che contiene tabelle per la conversione dei gruppi di dati (come 002501B5.TBL a 44B031A8.TBL).

**.log** In seguito all'installazione di MQSeries, alla creazione, all'avvio e ad un breve utilizzo di un Queue Manager, questa directory conterrà le seguenti sottodirectory e i seguenti file.

**amqhlctl.lfh**

File di controllo del log control.

**active** Questa directory contiene i file di log, così numerati:

S0000000.LOG

S0000001.LOG

S0000002.LOG

... e così di seguito.

**mqs.ini**

File di configurazione di MQSeries

**.qmgrs**

Questa directory contiene una sottodirectory `.$system` e una sottodirectory `.qmname` per ogni Queue Manager. La directory `.$system` contiene directory e file utilizzate internamente da MQSeries. Per ulteriori informazioni relative alla sottodirectory `.qmname`, consultare "Directory e file della sottodirectory MQS\_ROOT:[MQM.QMGRS.QMNAME]".

**.trace** Questa directory contiene i file di traccia creati dal comando `strmqtrc`.

---

### Directory e file della sottodirectory MQS\_ROOT:[MQM.QMGRS.QMNAME]

Per impostazione predefinita, le directory e i file riportati di seguito sono memorizzati nella directory MQS\_ROOT:[MQM.QMGRS.QMNAME]. L'estensione .QMNAME è generata per ogni Queue Manager creato ed eseguito sul sistema.

**amqalchk.fil**

File di checkpoint che contengono informazioni relative all'ultimo checkpoint.

**.auth** Questa directory contiene sottodirectory e file associati alle autorizzazioni.

**\$aclass.;**

Questo file contiene le voci di autorizzazione per tutte le classi.

### **.namelist**

Questa directory contiene un file per ogni namelist. Ogni file contiene le voci di autorizzazione per il namelist associato.

#### **\$class.;**

Questo file contiene le voci di autorizzazione per la classe di namelist.

#### **.\$mangled**

Se i nomi dei namelist contengono caratteri non validi per OpenVMS, questi vengono automaticamente convertiti in nomi validi per OpenVMS. I nomi OpenVMS validi sono contenuti in questo file. Consultare "Analisi dei nomi di file MQSeries" a pagina 21.

#### **system\$default\$namelist**

Questo file contiene le voci di autorizzazione per i namelist predefiniti dal sistema.

### **.procdef**

Ogni definizione di processo MQSeries è associato con un file di questa directory.

#### **\$class.;**

Questo file contiene le voci di autorizzazione per la classe della definizione di processo.

#### **.\$mangled**

Se i nomi delle definizioni di processo contengono caratteri non validi per OpenVMS, questi vengono automaticamente convertiti in nomi validi per OpenVMS. I nomi OpenVMS validi sono contenuti in questo file. Consultare "Analisi dei nomi di file MQSeries" a pagina 21.

#### **.system\$default\$process.;**

Questo file contiene le voci di autorizzazione per i processi predefiniti dal sistema.

### **.qmanager**

Questa directory contiene un file per ogni Queue Manager. Ogni file contiene le voci di autorizzazione per i Queue Manager associati.

#### **\$class.;**

Questo file contiene le voci di autorizzazione per la classe di Queue Manager.

#### **.\$mangled**

Se i nomi delle definizioni di Queue Manager contengono caratteri non validi per OpenVMS, questi vengono automaticamente convertiti in nomi validi per OpenVMS. I nomi OpenVMS validi sono contenuti in questo file. Consultare "Analisi dei nomi di file MQSeries" a pagina 21.

#### **self.;**

Questo file contiene le voci di autorizzazione per l'oggetto Queue Manager.

### **.queues**

Questa directory contiene un file per ogni coda. Ogni file contiene le voci di autorizzazione per la coda associata.

## Struttura di directory

### **\$CLASS**

Questo file contiene le voci di autorizzazione per la classe di coda.

### **.\$mangled**

Se i nomi delle code contengono caratteri non validi per OpenVMS, questi vengono automaticamente convertiti in nomi validi per OpenVMS. I nomi OpenVMS validi sono contenuti in questo file. Consultare "Analisi dei nomi di file MQSeries" a pagina 21.

### **File di definizione per la coda**

Ogni file corrisponde ad un oggetto predefinito per il Queue Manager.

```
system$admin$channel$event.;
system$admin$command$queue.;
system$admin$perfm$event.;
system$admin$qmgr$event.;
system$channel$initq.;
system$channel$syncq.;
system$cics$initiation$queue.;
system$cluster$command$queue.;
system$cluster$repository$queue.;
system$cluster$transmit$queue.;
system$dead$letter$queue.;
system$default$alias$queue.;
system$default$initiation$queue.;
system$default$local$queue.;
system$default$model$queue.;
system$default$remote$queue.;
system$mqsc$reply$queue.;
```

### **.qaadmin.;**

File utilizzato internamente per il controllo delle autorizzazioni.

**.dce** Directory vuota riservata all'utilizzo del supporto DCE.

**.errors** Questa directory contiene i file dei messaggi per l'operatore, dal più recente al più remoto.

```
amqerr01.log
amqerr02.log
amqerr03.log
```

**.esem** Directory che contiene file utilizzati internamente.

**.isem** Directory che contiene file utilizzati internamente.

**.msem** Directory che contiene file utilizzati internamente.

### **.namelist**

Questa directory contiene i namelist di ogni Queue Manager.

### **.plugcomp**

Questa directory vuota è riservata all'utilizzo di servizi installabili.

### **.procdef**

Ogni definizione di processo MQSeries è associato ad un file di questa directory. Il nome del file corrisponde al nome della definizione di processo.

**qm.ini** File di configurazione Queue Manager.

### **.qmanager**

Oggetto Queue Manager.

### **qmstatus.ini**

Questo file contiene del testo che descrive lo stato del Queue Manager.

### **.queues**

Ogni coda possiede una directory che contiene un singolo file denominato 'q'.

Il nome del file corrisponde al nome della coda—in base ad alcune regole; consultare “Analisi dei nomi di file MQSeries” a pagina 21.

### **.shmem**

#### **.perQueue**

Directory che contiene file utilizzati internamente.

**.ssem** Directory che contiene file utilizzati internamente.

### **.startprm**

Directory che contiene file temporanei utilizzati internamente.

### **.\$ipcc**

#### **amqclchl.tab**

File tabella di canale client.

#### **amqrfcda.dat**

File tabella di canale.

**.esem** Directory che contiene file utilizzati internamente.

**.isem** Directory che contiene file utilizzati internamente.

**.msem** Directory che contiene file utilizzati internamente.

### **.shmem**

#### **.perQueue**

Directory che contiene file utilizzati internamente.

**.ssem** Directory che contiene file utilizzati internamente.

## Struttura di directory



---

## Appendice D. Confronto tra gruppi di comandi

Le seguenti tabelle riportano il confronto tra le funzioni proprie di ciascun gruppo di comandi di gestione:

- “Comandi per la gestione di Queue Manager”
- “Comandi per la gestione di server di comandi”
- “Comandi per la gestione di code” a pagina 336
- “Comandi per la gestione di processi” a pagina 336
- “Comandi per la gestione di canali” a pagina 337
- “Altri comandi di controllo” a pagina 337

**Nota:** sono riportati solo i comandi MQSC che si applicano a MQSeries per Compaq OpenVMS.

---

### Comandi per la gestione di Queue Manager

Tabella 22. Comandi per la gestione di Queue Manager

PCF	MQSC	Controllo
Change Queue Manager	ALTER QMGR	–
Crea Queue Manager (Create queue manager)*		crtmqm
Elimina Queue Manager (Delete queue manager)*		dltmqm
Inquire Queue Manager	DISPLAY QMGR	–
Arresta Queue Manager (Stop queue manager)*		endmqm
Ping Queue Manager	PING QMGR	–
Avvia Queue Manager (Start queue manager)*		strmqm
<b>Nota:</b> * non disponibile tra i comandi PCF		

---

### Comandi per la gestione di server di comandi

Tabella 23. Comandi per la gestione di server di comandi

Descrizione	Controllo
Visualizza server di comandi (Display command server)	dsdmsvr
Avvia server di comandi (Start command server)	startmqsv
Arresta server di comandi (Stop command server)	endmqsv
<b>Nota:</b> questo gruppo di funzioni è disponibile solo come comando di controllo. Non esistono comandi PCF o MQSC equivalenti.	

## Comandi per la gestione di code

Tabella 24. Comandi per la gestione di code

PCF	MQSC
Change Queue	ALTER QLOCAL ALTER QALIAS ALTER QMODEL ALTER QREMOTE
Clear Queue	CLEAR QUEUE
Copy Queue	DEFINE QLOCAL(x) LIKE(y) DEFINE QALIAS(x) LIKE(y) DEFINE QMODEL(x) LIKE(y) DEFINE QREMOTE(x) LIKE(y)
Create Queue	DEFINE QLOCAL DEFINE QALIAS DEFINE QMODEL DEFINE QREMOTE
Delete Queue	DELETE QLOCAL DELETE QALIAS DELETE QMODEL DELETE QREMOTE
Inquire Queue	DISPLAY QUEUE
Inquire Queue Names	DISPLAY QUEUE
<b>Nota:</b> non esistono comandi di controllo equivalenti.	

## Comandi per la gestione di processi

Tabella 25. Comandi per la gestione di processi

PCF	MQSC
Change Process	ALTER PROCESS
Copy Process	DEFINE PROCESS(x) LIKE(y)
Create Process	DEFINE PROCESS
Delete Process	DELETE PROCESS
Inquire Process	DISPLAY PROCESS
Inquire Process Names	DISPLAY PROCESS
<b>Nota:</b> non esistono comandi di controllo equivalenti.	

## Comandi per la gestione di canali

Tabella 26. Comandi per la gestione di canali

PCF	MQSC	Controllo
Change Channel	ALTER CHANNEL	-
Copy Channel	DEFINE CHANNEL(x) LIKE(y)	-
Create Channel	DEFINE CHANNEL	-
Delete Channel	DELETE CHANNEL	-
Inquire Channel	DISPLAY CHANNEL	-
Inquire Channel Names	DISPLAY CHANNEL	-
Ping Channel	PING CHANNEL	-
Reset Channel	RESET CHANNEL	-
Resolve Channel	RESOLVE CHANNEL	-
Start Channel	START CHANNEL	runmqchl
Start Channel Initiator	START CHINIT	runmqchi
Start Channel Listener	-	runmqslr
Stop Channel	STOP CHANNEL	-

## Altri comandi di controllo

Tabella 27. Altri comandi di controllo

Descrizione	Controllo
Crea un'uscita di conversione MQSeries (Create MQSeries conversion exit)	crmqcvx
Visualizza autorizzazione (Display authority)	dspmqaut
Visualizza i file utilizzati dagli oggetti (Display files used by objects)	dspmqfls
Visualizza l'output di traccia formattato MQSeries (Display MQSeries formatted trace output)	dspmqtrc
Arresta traccia MQSeries (End MQSeries trace)	endmqtrc
Gestione di un gruppo di ripristino (Manage a failover set)	failover
Registra immagine oggetto (Record media image)	rcdmqimg
Rigenera oggetto (Recreate media object)	rcrmqobj
Risolvi transazioni MQSeries (Resolve MQSeries transactions)	rsvmqtrn
Esegui comandi MQSC (Run MQSC commands)	runmqsc
Esegui trigger monitor (Run trigger monitor)	runmqtrm
Esegui trigger monitor client (Run client trigger monitor)	runmqtrmc
Imposta o reimposta l'autorizzazione (Set or reset authority)	setmqaut
Avvia un failover monitor (Start a failover monitor)	runmqfm
Avvia traccia MQSeries (Start MQSeries trace)	strmqtrc
<b>Nota:</b> le funzioni di questo gruppo sono disponibili solo come comandi di controllo. Non esistono comandi PCF diretti o comandi MQSC equivalenti.	

## Confronto tra gruppi di comandi

---

## Appendice E. Esempi di programmi MQI e file MQSC

MQSeries per Compaq OpenVMS fornisce un insieme di brevi esempi di programmi MQI e file di comando MQSC che si possono utilizzare per fare pratica. La descrizione è disponibile nelle seguenti sezioni:

- “Esempi di file di comando MQSC”
- “Esempi di programmi in C e COBOL”
- “Strumenti vari” a pagina 340

---

### Esempi di file di comando MQSC

Tabella 28 elenca gli esempi di file di comando MQSC. Si tratta di semplici file di testo ASCII contenenti comandi MQSC. È possibile invocare il comando **runmqsc** rispetto a ciascun file per creare gli oggetti specificati nel file. Consultare “Esecuzione dei file di comandi MQSC forniti” a pagina 44.

Per impostazione predefinita, tali file sono memorizzati nella directory `MQS_EXAMPLES`:

Tabella 28. File di comando MQSC

Nome file	Scopo
AMQSCOS0.TST	Crea un insieme di oggetti MQI per l'utilizzo con esempi di programmi in C e COBOL.

---

### Esempi di programmi in C e COBOL

Tabella 29 elenca gli esempi di file di origine MQSC. Per impostazione predefinita, i file di origine si trovano nella directory `MQS_EXAMPLES`: e le versioni compilate nella directory `[.BIN]` sotto `MQS_EXAMPLES`. Per ulteriori informazioni sullo scopo e le modalità d'utilizzo dei programmi, consultare la sezione *MQSeries Application Programming Guide*.

Tabella 29. Programmi di esempio - file di origine

C	COBOL	Scopo
AMQSBCG0.C	–	Legge ed emette sia il messaggio di descrizione che i campi del contesto del messaggio di tutti i messaggi su una coda specificata.
AMQSECHA.C	AMQVECHX.COB	Restituisce un messaggio da una coda di messaggi alla coda di risposta RTQ. È possibile eseguirlo come programma applicativo avviato.
AMQSGBR0.C	AMQ0GBR0.COB	Scrive messaggi da una coda a <code>SYS\$OUTPUT</code> lasciando i messaggi in coda. Utilizza <code>MQGET</code> con l'opzione <code>browse</code> .
AMQSGET0.C	AMQ0GET0.COB	Rimuove i messaggi dalla coda nominata (utilizzando <code>MQGET</code> ) e li scrive su <code>SYS\$OUTPUT</code> .
AMQSINQA.C	AMQVINQX.COB	Legge la coda avviata; ciascuna richiesta viene letta come nome di coda; risponde con informazioni su tale coda.
AMQSPUT0.C	AMQ0PUT0.COB	Copia <code>SYS\$INPUT</code> in un messaggio e quindi inserisce questo messaggio in una coda specifica.
AMQSREQ0.C	AMQ0REQ0.COB	Inserisce messaggi di richiesta su una coda specificata e quindi visualizza i messaggi di risposta.

## esempi

Tabella 29. Programmi di esempio - file di origine (Continua)

C	COBOL	Scopo
AMQSSETA.C	AMQVSETX.COB	Inibisce i put su una coda nominata e risponde con una istruzione del risultato. Esegue un'applicazione avviata.
AMQSTRG0.C	–	Un monitor di trigger che legge una coda di avvio e quindi avvia il programma associato a ciascun messaggio di trigger. Fornisce un sottoinsieme della funzione di triggering completa del comando <b>runmqtrm</b> fornito.
AMQSVFCX.C	–	Un esempio di skeleton C di routine di uscita di conversione dei dati.

## Strumenti vari

Questi file strumento sono forniti come supporto per la formattazione e la conversione del codice.

Tabella 30. File vari

Nome file	Posizione	Scopo
AMQTRC.FMT	SYS\$LIBRARY	Definisce i formati di traccia MQSeries.
CCSID.TBL	MQS_ROOT:[MQM.CONV.TABLE]	Modificare questo file per aggiungere valori CSSID recentemente supportati dal sistema MQSeries. Per ulteriori informazioni su CCSID, consultare la documentazione di CDRA (Character Data Representation Architecture).

---

## Appendice F. OpenVMS modelli di gruppi di ripristino di cluster

Questa appendice contiene i seguenti modelli di gruppi di ripristino:

- “Modello di file di configurazione FAILOVER.TEMPLATE”
- “Modello di procedura StartCommand START\_QM.TEMPLATE” a pagina 343
- “Modello di procedura EndCommand END\_QM.TEMPLATE” a pagina 344
- “Modello di procedura TidyCommand TIDY\_QM.TEMPLATE” a pagina 347

---

### Modello di file di configurazione FAILOVER.TEMPLATE

```
#####  
#*                                     *#  
#* Statement:      Licensed Materials - Property of IBM          *#  
#*                                     *#  
#*                33H2205, 5622-908                               *#  
#*                33H2267, 5765-623                               *#  
#*                29H0990, 5697-176                               *#  
#*                (C) Copyright IBM Corp. 2000, 2001            *#  
#*                                     *#  
#####  
#  
# FAILOVER.TEMPLATE  
# Template for creating a FAILOVER.INI configuration file  
# All lines beginning with a '#' are treated as comments  
#  
# OpenVMS Cluster Failover Set Configuration information  
# -----  
#  
# The TCP/IP address used by the OpenVMS Cluster Failover Set  
#  
IpAddress=n.n.n.n  
#  
# The TCP/IP port number used by the MQSeries Queue Manager  
#  
PortNumber=1414  
#  
# The timeout used by the EndCommand command procedure  
#  
TimeOut=30  
#  
# The command procedure used to start the Queue Manager  
#  
StartCommand=@sys$manager:start_qm  
#  
# The command procedure used to end the Queue Manager  
#  
EndCommand=@sys$manager:end_qm  
#  
# The command procedure used to tidy up on a node after a  
# Queue Manager failure but the OpenVMS node did not fail  
#  
TidyCommand=@sys$manager:tidy_qm  
#  
# The directory in which the log files for the start, end and  
# tidy commands are written  
#  
LogDirectory=mqs_root:[mqm.errors]  
#  
# The number of nodes in the OpenVMS Cluster Failover Set. The
```

## FAILOVER.TEMPLATE

```
# number of nodes defined below must agree with this number
#
NodeCount=2
#
# The Name of the OpenVMS node
#
NodeName=BATMAN
#
# The TCP/IP interface name for the node
#
Interface=we0
#
# The priority of the node
#
Priority=1
#
# The Name of the OpenVMS node
#
NodeName=ROBIN
#
# The TCP/IP interface name for the node
#
Interface=we0
#
# The priority of the node
#
Priority=2
```

*Figura 27. File di configurazione modello: failover.template*



## Modello di procedura StartCommand START\_QM.TEMPLATE

```

$ on error then exit
$!*****
$!* Statement:      Licensed Materials - Property of IBM      *
$!*                33H2205, 5622-908                        *
$!*                33H2267, 5765-623                        *
$!*                29H0990, 5697-176                        *
$!*                (C) Copyright IBM Corp. 2000, 2001      *
$!*****
$! Template command procedure used by Failover Sets to start the
$! queue manager
$! Parameters :
$! P1 = Queue Manager Name
$! P2 = Queue Manager Directory Name
$! P3 = TCP/IP address
$! P4 = TCP/IP interface name
$! P5 = Listener port number
$!
$ @sys$startup:mqs_symbols
$ set def mqs_root:[mqm.qmgrs.'p2'.errors]
$ define sys$scratch mqs_root:[mqm.qmgrs.'p2'.errors]
$!
$! Digital TCP/IP Services for OpenVMS commands
$!
$ @sys$startup:tcpip$define_commands
$!
$! Configure the IP address
$!
$ ifconfig 'p4' alias 'p3'
$!
$! TCPware for OpenVMS commands
$!
$! @tcpware:tcpware_commands
$!
$! Configure the IP address
$!
$! netcu add secondary 'p3'
$!
$! MultiNet for OpenVMS commands
$!
$! Configure the IP address
$!
$! define/sys/exec multinet_ip_cluster_aliases "'p3'"
$!
$! Restart the Multinet server
$!
$! @multinet:start_server
$!$! Start the queue manager
$!
$ strmqm 'p1'
$!
$! Start the listener
$!
$! runmqlsr -t tcp -p 'p5' -m 'p1'
$!
$! Insert commands to start any applications
$!
$exit

```

Figura 28. Modello di procedura StartCommand: Start\_QM.template

---

**Modello di procedura EndCommand END\_QM.TEMPLATE**

```

$ on error then exit
$!
$!*****
$!*
$!* Statement:      Licensed Materials - Property of IBM      *
$!*
$!*                33H2205, 5622-908                          *
$!*                33H2267, 5765-623                          *
$!*                29H0990, 5697-176                          *
$!*                (C) Copyright IBM Corp. 2000, 2001        *
$!*
$!*****
$!
$! Template Command procedure used by Failover Sets to end the
$! queue manager
$!
$! Parameters :
$!
$! P1 = Queue Manager Name
$! P2 = Queue Manager Directory Name
$! P3 = TCP/IP address
$! P4 = TCP/IP interface name
$! P5 = Listener port number
$! P6 = End Queue Manager Timeout
$!
$ @sys$startup:mqs_symbols
$ check_qm==$sys$system:mqcheckqm
$ set def mqs_root:[mqm.qmgrs.'p2'.errors]
$ define sys$scratch mqs_root:[mqm.qmgrs.'p2'.errors]
$ SS$ _NORMAL=1
$ SS$ _ABORT=44
$ SS$ _TIMEOUT=556
$!
$! Insert commands to shutdown any applications prior to ending MQSeries
$!
$! Get the timeout period for each operation seconds
$!
$ timeout = 'p6'
$!
$! Initialise the outer loop
$!
$ out_count = 0
$!
$! Initialise the complete flag
$!
$ complete = 0
$!
$ out_next:
$ if (out_count .gt. 2) .or. (complete .eq. 1) then goto out_finish
$!
$ if out_count .eq. 0
$ then
$!
$! End the queue manager gracefully first
$!
$ spawn/nowait $endmqm -i 'p1'
$ else
$ if out_count .eq. 1
$ then
$!
$! End the queue manager abruptly
$!
$ spawn/nowait $endmqm -p 'p1'
$ else
$!

```

```

$! Stop/id the execution controller
$!
$ check_qm -m 'p1'
$ if ( mqs$ec_pid .nes. "" ) then $stop/id='mqs$ec_pid'
$ endif
$ endif
$!
$ in_start:
$!
$! Initialise the outer loop
$!
$ in_count = 0
$!
$ in_next:
$!
$! Inner loop
$!
$ if ( ( in_count .ge. timeout ) .and. ( timeout .ne. 0 ) ) -
    .or. ( complete .eq. 1 ) then goto in_finish
$!
$! Check if the execution controller is still running
$!
$ check_qm -m 'p1'
$ if mqs$ec_pid .eqs. ""
$ then
$!
$! The Execution controller is no longer running so we are finished
$!
$ complete = 1
$ goto in_finish
$ endif
$!
$! Wait a second and go round again
$!
$ wait 00:00:01
$ in_count = in_count + 1
$ goto in_next
$ in_finish:
$!
$! End of the inner loop
$!
$ out_count = out_count + 1
$ goto out_next
$ out_finish:
$!
$! End of the outer loop
$!
$! Digital TCP/IP Services for OpenVMS commands
$!
$ @sys$startup:tcpip$define_commands
$!
$! De-configure the IP address
$!
$ ifconfig 'p4' -alias 'p3'
$!
$! TCPware for OpenVMS commands
$!
$! @tcpware:tcpware_commands
$!
$! De-configure the IP address
$!
$! netcu remove secondary 'p3'
$!
$! MultiNet for OpenVMS commands
$!
$! De-configure the IP address
$!

```

## END\_QM.TEMPLATE

```
#! deass/sys/exec multinet_ip_cluster_aliases
#!
#! Restart the Multinet server
#!
#! @multinet:start_server
#!
#!
#! If the Queue Manager was shutdown successfully set the status
#! to SS$ _NORMAL. If it was necessary to STOP/ID the Execution
#! controller set the status to SS$ _ABORT and if the Execution
#! controller is still running set the status to SS$ _TIMEOUT to
#! indicate an error
#!
$ if ( complete .eq. 1 )
$then
$!
$! End the listener process
$!
$! endmq1sr -m 'p1'
$!
$ if ( out_count .eq. 3 )
$ then
$ exit SS$ _ABORT
$ else
$ exit SS$ _NORMAL
$ endif
$else
$ exit SS$ _TIMEOUT
$endif
```

*Figura 29. Modello di procedura EndCommand: END\_QM.template*

## Modello di procedura TidyCommand TIDY\_QM.TEMPLATE

```

$ on error then exit
$!*****
$!* Statement:      Licensed Materials - Property of IBM      *
$!*
$!*                33H2205, 5622-908                        *
$!*                33H2267, 5765-623                        *
$!*                29H0990, 5697-176                        *
$!*                (C) Copyright IBM Corp. 2000, 2001      *
$!*****
$! Template Command procedure used by Failover Sets to tidy up after
$! a queue manager failure
$!
$! Parameters :
$! P1 = Queue Manager Name
$! P2 = Queue Manager Directory Name
$! P3 = TCP/IP address
$! P4 = TCP/IP interface name
$! P5 = Listener port number
$!
$ @sys$startup:mqs_symbols
$ set def mqs_root:[mqm.qmgrs.'p2'.errors]
$ define sys$scratch mqs_root:[mqm.qmgrs.'p2'.errors]
$!
$! Insert commands to do any tidying up after a queue manager has failed
$!
$! Digital TCP/IP Services for OpenVMS commands
$!
$ @sys$startup:tcpip$define_commands
$!
$! De-configure the IP address
$!
$ ifconfig 'p4' -alias 'p3'
$!
$! TCPware for OpenVMS commands
$!
$! @tcpware:tcpware_commands
$!
$! De-configure the IP address
$!
$! netcu remove secondary 'p3'
$!
$! MultiNet for OpenVMS commands
$!
$! De-configure the IP address
$!
$! deass/sys/exec multinet_ip_cluster_aliases
$!
$! Restart the Multinet server
$!
$! @multinet:start_server
$!
$exit

```

Figura 30. Modello di procedura TidyCommand: TIDY\_QM.template



## Appendice G. Codici supportati da MQSeries per Compaq OpenVMS

MQSeries per Compaq OpenVMS supporta la maggior parte dei codici utilizzati dai locali – i gruppi secondari dell’ambiente utente in cui si definiscono le convenzioni per una determinata cultura – forniti come standard su MQSeries per Compaq OpenVMS.

Se non viene impostato il locale, il CCSID utilizzato è 819 - il gruppo di codici ISO8859-1.

Analizzando la categoria LC\_CTYPE della configurazione locale si ottiene il CCSID (Coded Character Set Identifier) utilizzato in MQSeries per identificare il codice per i dati del messaggio e dell’intestazione del messaggio.

Tabella 31 riporta i CCSID e i locali registrati per i codici utilizzati dal locale.

*Tabella 31. Locali e CCSID*

Locale	Lingua	codici	CCSID
C	Inglese	ISO8859-1	819
CS_CZ_ISO8859-2	Cecoslovacco	ISO8859-2	912
DA_DK_ISO8859-1	Danese	ISO8859-1	819
DE_DE_ISO8859-1	Tedesco	ISO8859-1	819
DE_CH_ISO8859-1	Tedesco - Svizzera	ISO8859-1	819
EL_GR_ISO8859-7	Greco	ISO8859-7	813
EN_GB_ISO8859-1	Inglese - Regno Unito	ISO8859-1	819
EN_US_ISO8859-1	Inglese - Stati Uniti	ISO8859-1	819
ES_ES_ISO8859-1	Spagnolo	ISO8859-1	819
FI_FI_ISO8859-1	Finlandese	ISO8859-1	819
FR_FR_ISO8859-1	Francese - Francia	ISO8859-1	819
FR_BE_ISO8859-1	Francese - Belgio	ISO8859-1	819
FR_CA_ISO8859-1	Francese - Canada	ISO8859-1	819
FR_CH_ISO8859-1	Francese - Svizzera	ISO8859-1	819
HU_HU_ISO8859-2	Ungherese	ISO8859-2	912
IS_IS_ISO8859-1	Islandese	ISO8859-1	819
IT_IT_ISO8859-1	Italiano - Italia	ISO8859-1	819
IW_IL_ISO8859-8	Ebraico	ISO8859-8	916
JA_JP_EUCJP	Giapponese	eucJP	954
JA_JP_SDECKANJI	Giapponese	SDECKANJI	954**
JA_JP_SJIS	Giapponese	SJIS	932
KO_KR_DECKOREAN	Coreano	DECKOREAN	970**
NL_NL_ISO8859-1	Olandese - Paesi Bassi	ISO8859-1	819
NL_BE_ISO8859-1	Olandese - Belgio	ISO8859-1	819

## Codici supportati

Tabella 31. Locali e CCSID (Continua)

Locale	Lingua	codici	CCSID
NO_NO_ISO8859-1	Norvegese	ISO8859-1	819
PL_PL_ISO8859-2	Polacco	ISO8859-2	912
PT_PT_ISO8859-1	Portoghese	ISO8859-1	819
SK_SK_ISO8859-2	Slovacco	ISO8859-2	912
RU_RU_ISO8859-5	Cirillico	ISO8859-5	915
SV_SE_ISO8859-1	Svedese	ISO8859-1	819
TR_TR_ISO8859-9	Turco	ISO8859-9	920
ZH_CN_DECHANZI	Cinese - moderno	DECHANZI	1383**
ZH_HK_DECHANZI	Cinese - moderno	DECHANZI	1383**
ZH_HK_EUCTW	Cinese - Tradizionale	eucTW	964
ZH_HK_EUCTW	Cinese - Tradizionale	eucTW	964
ZH_HK_DECHANYU	Cinese - Tradizionale	DECHANYU	964**
ZH_TW_DECHANYU	Cinese - Tradizionale	DECHANYU	964**
ZH_HK_BIG5	Cinese - Tradizionale	big5	950
ZH_TW_BIG5	Cinese - Tradizionale	big5	950
<b>Nota:</b>			
** Il CCSID utilizzato è quello più prossimo al CCSID IBM registrato.			

Per un elenco più dettagliato relativo a questi locali, consultare il manuale *MQSeries per Tandem Non Stop Kernal, V5.1 Guida operativa*.



---

## Appendice H. Utilità di diagnostica MONMQ

L'utilità MONMQ è uno strumento di assistenza nella diagnosi e risoluzione dei problemi con MQSeries per Compaq OpenVMS. L'utilità MONMQ può essere utilizzata in modo interattivo, dalla riga comandi, oppure dall'interno di uno script DCL.

L'utilità MONMQ è principalmente utilizzata per:

- Gestire la memoria condivisa
- Coadiuvare la raccolta di informazioni sull'utilizzo della risorsa OpenVMS
- Ottenere l'output di traccia da un Queue Manager in esecuzione.

MONMQ dispone di un sistema di guida per quanto riguarda i parametri e può anche eseguire uno script di comandi MONMQ. Quando si avvia MONMQ, viene eseguito uno script predefinito `sys$manager:mqs_trace_startup.mqt` in modo tale da fornire una configurazione iniziale.

```
$monmq
ok - mailbox 0 opened as default

MQT> help
Help can be used to display information about available commands or parameters
Help [ <verb> || <parameter/variable name> || commands || parameters || examples]

Valid trace commands are in the format:
Verb [<parameters>] [<variable = expression>][;][optional second command]
```

---

## Panoramica

La traccia di MQSeries su OpenVMS è implementata utilizzando sezioni globali e caselle di posta. Possono coesistere fino a 10 sezioni di traccia (LU) su qualsiasi nodo in cui è installato MQSeries. Tuttavia, si consiglia fortemente di utilizzare sempre una LU per volta in una sessione di traccia e si sconsiglia inoltre l'apertura contemporanea della stessa LU da parte di più utenti; in caso contrario, i risultati non sono prevedibili.

Ogni sezione condivisa (LU) contiene le definizioni di canale e la definizione della stessa LU. Ciascuna definizione di canale contiene i dettagli del sottoprocesso connesso, lo stack privato e il buffer circolare dei sottoprocessi. Inoltre, la sezione condivisa contiene un insieme di flag utilizzati per la comunicazione interprocesso tra MONMQ e i sottoprocessi connessi.

Ad ogni LU è associata una casella di posta utilizzata per ricevere messaggi di traccia in tempo reale. Per eseguire le funzioni di traccia in tempo reale, un processo di client deve essere avviato tramite il comando **TRACE START**. Questo processo dedicato e separato legge, formatta e visualizza ogni messaggio così come arriva nella casella postale delle LU. Ogni sottoprocesso connesso scrive alla stessa casella di posta offrendo, in tal modo, la possibilità di visualizzare fisicamente l'intercomunicazione tra i processi/sottoprocessi di MQSeries.

Se utilizzata correttamente, MONMQ può fornire un metodo completo per la diagnosi dei problemi, quali i problemi di tempo fra un processo e l'altro, risorse di sistema operativo esaurite o anche problemi di codifica.

## Variabili in MONMQ

La descrizione dei comandi MONMQ è contenuta in questa appendice.

---

## Variabili in MONMQ

In MONMQ, le variabili sono utilizzate da molti comandi. Una variabile utilizza un valore predefinito, impostato tramite il comando **set**, qualora non venga specificato all'interno del comando. Il valore predefinito di una variabile, se utilizzata da un comando diverso da **set**, non viene modificato.

Le variabili possono contenere:

- Interi (decimali o esadecimali).  
Quando richiesto, un valore esadecimale può essere inserito con parte iniziale 0x oppure con lettere dalla a alla f.
- Testo, da includere tra apici.
- Un intervallo, definito da un minimo:massimo.  
Ad esempio, un intervallo può essere utilizzato per applicare un comando a un intervallo di canali.

Ad esempio:

```
MQT> set lu=2
MQT> set pid=0x223
MQT> set pid=2fa
MQT> set buffile="filename.buf"
MQT> set chl=0:20
```

Il corrente valore predefinito per le variabili può essere visualizzato utilizzando il comando di variabili.

```
MQT> variables
defined variables
lu=0:0                nochls=20                buffer=1000
chl=0:20              component=0 (HEX)        line=0
mask=0 (HEX)          pid=0 (HEX)             node=(null)
function=0 (HEX)      div=0                    depth=32
resource=0            wait=1 (BOOL)           timestamp=0 (BOOL)
listfile=(null)       buffile=(null)          step=0 (BOOL)
active=0 (BOOL)       fname=(null)            delay=100
post=0 (BOOL)

defined constants
fent=1 (HEX)          fout=2 (HEX)            ferr=4 (HEX)
fxxx=8 (HEX)          dgn=10 (HEX)           shm=20 (HEX)
spl=40 (HEX)          evt=80 (HEX)           mtx=100 (HEX)
prc=200 (HEX)         msc=400 (HEX)          inf=800 (HEX)
log=2000 (HEX)        shl=4000 (HEX)         memory=3
mutex=4               mailbox=5               nanoseconds=1
microseconds=2        milliseconds=3         seconds=4
```

Si consiglia di impostare i valori predefiniti in modo da semplificare i comandi. Ad esempio, le seguenti sequenze di comando sono funzionalmente identiche:

```
MQT> open lu=0 buffer=1000 nochls=20
MQT> open lu=1 buffer=1000 nochls=20
MQT> show channels lu=0 chl=1:10
MQT> show channels lu=1 chl=1:10
```

oppure

```
MQT> set nochls=20 chl=0:10 buffer=1000 lu=0
MQT> open
MQT> open lu=1
MQT> show channels
MQT> show channels lu=1
```

È possibile ridurre al minimo indispensabile il numero dei caratteri per i comandi MONMQ garantendone l'univocità. È possibile ridurre ulteriormente questa serie di comandi a:

```
MQT> se noc=20 ch=0:10 buffe=1000 lu=0
MQT> op
MQT> op lu=1
MQT> sh ch
MQT> sh ch lu=1
```

MONMQ permette anche l'esecuzione di semplici operazioni aritmetiche con le variabili in modo da rendere possibili comandi quali  $lu=lu+1$ .

È possibile dichiarare nuove variabili tramite il comando **declare**. I parametri sono:

- Nome della variabile
- Tipo della variabile
- Testo guida. In seguito, sarà possibile richiamare il testo guida tramite il comando **help**.

```
MQT> declare ec int "channel number for execution controller"
MQT> set ec = 4
MQT> show channel chl=ec
Chl  Pid  Mailbox  Stack  Active  Post  Time  Mask  Process Name
 4   2c1f  7ee70290  4      0      0    0   ffffffff  AMQZXMA0.EXE
MQT> help ec

VARIABLE ec:
channel number for execution controller

MQT>
```

## Assegnazione dei valori predefiniti

DEFAULT variable=<expression> [variable=expression] ...

Questo comando permette l'assegnazione dei valori predefiniti a tutte le variabili definite in MONMQ; una volta impostato, è possibile omettere il nome predefinito della variabile dalla riga comandi. Ad esempio:

```
MQT>default lu=2 chl=3:6
```

Questo comando imposta il valore predefinito 2 alla variabile  $lu$  e i valori da 3 a 6 compreso alla variabile canale. D'ora in avanti, l'utilizzo di un tipico comando quale **show channels**, comporterà la visualizzazione dei canali da 3 a 6 incluso su  $lu$  2.

## Variabili in MONMQ

I valori predefiniti vengono utilizzati solo nei casi in cui viene omessa la variabile nella riga comandi. Tutti i valori predefiniti sono impostati nel file script di avvio MQS\_TRACE\_STARTUP.MQT. Questo file può essere modificato secondo le proprie necessità.

---

## Aprire o creare una sezione di traccia e la casella postale associata

OPEN [lu=numero] [nochls=numero] [buffer=numero]

Questo comando apre o crea una sezione di traccia e la casella postale associata. Il comando **open** crea la risorsa di base necessaria per la traccia dei processi MQSeries. A ogni LU sono associate una sezione condivisa e una casella postale, utilizzate per comunicare con i processi di MQSeries.

Questo comando prevede tre parametri opzionali. Il primo parametro [LU] è un numero assegnato alla sezione di traccia/casella postale e viene utilizzato come riferimento dalla maggior parte degli altri comandi MONMQ. È possibile creare un massimo di dieci LU su un singolo nodo. Il valore predefinito è zero. L'esistenza del LU specificato comporta la connessione di MONMQ alla sezione di traccia esistente. Qualora non esistano sezioni, ne verrà creata una nuova.

Il secondo parametro [nochls] specifica il numero dei canali di cui disporrà la relativa LU. Ogni canale rappresenta una singola connessione di processo/sottoprocesso MQSeries. Il valore predefinito è 20.

Il terzo parametro [buffer] specifica la massima dimensione del buffer di cronologia traccia per ogni canale. Il valore predefinito è 1000.

Per eseguire altri comandi MONMQ, è necessario avere almeno una LU aperta.

---

## Visualizzazione della definizione di unità logica

SHOW SEGMENT [lu=range]

Questo comando visualizza la definizione di Unità Logica. Di seguito, viene mostrato un esempio di visualizzazione con una breve descrizione a lato di ogni campo dove si immette il comando **show segment lu=0**.

```
Trace LU           : 0           /* The LU number as specified in the OPEN [lu] parameter.
Mailbox name       : MQS_TRC_MBX_0 /* The permanent mailbox name assigned to this LU
Device name        : MBA1065:      /* The device name of the mailbox
Status             : Disabled      /* The current status of the mailbox ie.
Mailbox channel    : 352           /* The mailbox channel number assigned to the MONMQ process
History buffer size: 1000          /* The maximum number of message entries in the history
/* circular buffer (as specified by the OPEN [buffer] parameter)
Threads mapped #   : 1             /* The number of processes/threads mapped to this LUs global
/* section (MONMQ always attached)
Time stamping      : Enabled       /* Global timestamp flag (not yet implemented)
Max channels #     : 20            /* Number of channels defined for this LU as specified by the
/* OPEN [nochls] parameter.
Display depth      : 0             /* The stack display depth. Default (0) is to display all stack entries.
Text filename      :                /* The client text trace file
Binary filename    :                /* The client binary trace file
Last status        : 1             /* Last status of mailbox Qio activity (useful if VMS low on
/* resources and MONMQ fails)
Connection map[0] : 0             /* A bit map of all connected channels (maximum no. of channels is 128)
```

## Chiusura ed eliminazione di una LU

CLOSE [lu=numero]

Questo comando esegue le operazioni inverse di **OPEN** chiudendo ed eliminando la specificata LU. La chiusura di LU consiste in una sequenza controllata che inizia con la segnalazione di disconnessione per ogni processo connesso, prosegue con il ripristino di ogni canale e infine con l'annullamento dell'assegnazione della casella postale della traccia e l'eliminazione della sezione condivisa. Questo comando deve essere eseguito solo quando una sessione di traccia è stata completata.

## Visualizzazione dei dettagli di canale

SHOW CHANNELS [full] [connected] [chl=intervallo]

Questo comando visualizza i dettagli dei canali specificati. Il parametro [connected] provocherà la visualizzazione soltanto dei canali che hanno un sottoprocesso connesso. Ad esempio, il comando `show channels connected` visualizza quanto segue:

Chl	Pid/Tid	Mailbox	Stack	History	RTime	Time	Mask	Process Name
0	00000245/1	800b0330	4	0	0	0	fffffff	AMQZLAA0.EXE
1	00000244/1	800b0200	8	0	0	0	fffffff	RUNMQCHI.EXE
2	00000243/1	800b01e0	10	0	0	0	fffffff	AMQRRMFA.EXE
3	00000242/1	800b01c0	4	0	0	0	fffffff	AMQZLLP0.EXE
4	00000241/1	800b0220	5	0	0	0	fffffff	AMQHASM0.EXE
5	00000240/1	800b03f0	4	0	0	0	fffffff	AMQZXMA0.EXE

Il parametro [full] visualizza la definizione completa dei canali specificati. Ad esempio, il comando `show channels full connected chl=0:3` visualizza:

```

Pid/Tid          : 0000024b/1      /* id Connected threads process id and thread sequence number
Status           : *** Connected *** /* Current status of channel (thread is connected)
Process name     : AMQRRMFA.EXE   /* Process name of connected thread
Assigned LU      : 0              /* This channels associated LU
Channel no.      : 2              /* Allocated channel number within the LU
Mailbox channel  : 800b01e0      /* Connected threads mailbox channel number for the trace mailbox
Current stack depth : 10         /* Threads current stack depth
Circular logging : Disabled       /* History enabled flag
Next log entry   : 0              /* Next history buffer slot number
Realtime tracing : Disabled       /* Real time enable flag (needs client to read messages)
Time stamping    : Disabled       /* Enables timestamping for this threads messages
Trace mask       : ffffffff       /* Hexadecimal format of trace mask for this thread (see show mask command)
Step mode        : Off            /* Not yet implemented
No Wait         : 0n              /* Forces threads qio activity to wait for a resource if not available
Last QIO status  : 0              /* Threads last qio call status
Mapped address   : 9a2000-c9ffff  /* The virtual mapped address range of the LU global section for this thread
  
```

## Visualizzazione della maschera di traccia corrente per un canale

SHOW MASK [chl=range]

Questo comando visualizza la maschera di traccia corrente per un canale. Una riga evidenziata indica che il bit della maschera di traccia è abilitato. Ad esempio, il comando, `show mask chl=1` visualizza:

## SHOW MASK

Trace Mask for Channel 1

<b>Bit 00</b> - (fent) function entry	<b>Function entry messages</b>
<b>Bit 01</b> - (fout) function exit	<b>Function exit messages</b>
Bit 02 - (ferr) function exit with error	Function exit with error return status
Bit 03 - (fxx) missing function exit	Unbalanced function entry/exit message (see note below)
Bit 04 - (dgn) diagnostic messages	Diagnostic messages
Bit 05 - (shm) shared memory	OVMS shared memory messages
<b>Bit 06</b> - (spl) spinlocks	<b>OVMS spinlock messages</b>
<b>Bit 07</b> - (evt) events	<b>OVMS event messages</b>
Bit 08 - (mtx) mutexes	OVMS mutex messages
Bit 09 - (prc) process msgs	OVMS thread messages
Bit 10 - (msc) miscellaneous	OVMS kernal niscellaneous messages
Bit 11 - (inf) informational	Internal data messages as requested by show command
Bit 12 -	Reserved for user defined messages

Questo output mostra che la funzione entry, la funzione exit e i messaggi di evento e di blocchi di spin verranno tracciati per questo sottoprocesso. Tutti gli altri tipi di messaggi saranno bloccati.

---

## Visualizzazione dei contenuti dello stack dei sottoprocessi di destinazione

SHOW STACK [chl=*range*]

Questo comando visualizza i contenuti dello stack dei sottoprocessi di destinazione. Ad esempio, il comando **show stack chl=0:1** visualizza:

```
0001- 00:00:00.00 03 - 01 -->| ExecCtrlrMain
0002 - 12:36:20.18 03 - 02 --->| zcpReceiveOnLink
0003 - 12:36:20.81 03 - 03 ---->| xcsWaitEventSem
0004 - 12:36:20.83 03 - 04 ----->| vms_evt

0001- 00:00:00.00 03 - 01 -->| ExecCtrlrMain
0002 - 12:36:20.18 03 - 02 --->| zcpSendOnLink
0003 - 12:36:20.81 03 - 03 ---->| xcsPostEventSem
0004 - 12:36:20.83 03 - 04 ----->| vms_evt
```

---

## Visualizzazione dei processi attivi relativi a MQSeries e dell'utilizzo della memoria

SHOW PROCESSES

Questo comando visualizza tutti i processi attivi relativi a MQSeries sul corrente nodo e il relativo utilizzo di memoria. Ad esempio, il comando **show process** visualizza:

PID	Proc_Name	Image	Process	WS_Size	WS_Peak	Virt_Peak	Gbl_Pg_Cnt	Prc_Pg_Cnt	Total_Mem
0000023D	BKM3_AG	AMQZLAA0	Agent	23152	16576	203776	3616	12960	16576
0000023C	BKM3_CI	RUNMQCHI	Run Chan Init	8752	6208	180832	1840	4368	6208
0000023B	BKM3_RM	AMQRRMFA	Repository Mgr	11152	8144	185360	2224	5920	8144
0000023A	BKM3_CP	AMQZLLP0	Checkpoint	8752	6384	185952	1920	4464	6384
00000239	BKM3_LG	AMQHASM	Logger	8752	6288	182016	2080	4208	6288
00000238	BKM3_EC	AMQZXMA0	EC	20752	15232	203792	3536	11680	15216
00000128	_FTA4:	MONMQ	MONMQ Utility	8400	8528	198736	2224	3584	5808

---

## Visualizzazione di tutti i messaggi conservati in un canale

SHOW HISTORY [chl=*range*]

## SHOW HISTORY

Questo comando visualizza tutti i messaggi conservati nel buffer di cronologia circolare del canale. Ogni messaggio viene formattato e l'output viene rientrato in relazione alla profondità dello stack in cui è stato generato. Ad esempio, il comando **show history chl=3** visualizza:

```
0215 - 12:35:44.52 03 - 02 ---<| zxcProcessChildren
0216 - 12:35:44.55 03 - 02 --->| zxcStartWLMserver
0217 - 12:35:44.57 03 - 02 ---<| zxcStartWLMserver
0218 - 12:35:44.59 03 - 02 --->| zcpReceiveOnLink
0219 - 12:35:44.61 03 - 03 ---->| xcsRequestMutexSem
0220 - 12:35:44.63 03 - 04 ---->| xllSemReq
0221 - 12:35:44.66 03 - 05 ---->| vms_mtx
0222 - 12:35:44.66 03 - 05 .....| vms_mtx :- Locking BKM3/@ipcc_m_1_10 - timeout: -1
0223 - 12:35:44.70 03 - 06 ---->| vms_get_lock
0224 - 12:35:44.72 03 - 06 ----<| vms_get_lock
0225 - 12:35:44.74 03 - 05 ----<| vms_mtx
0226 - 12:35:44.76 03 - 04 ----<| xllSemReq
0227 - 12:35:44.79 03 - 03 ----<| xcsRequestMutexSem
0228 - 12:35:44.81 03 - 03 ---->| xcsResetEventSem
0229 - 12:35:44.83 03 - 04 ---->| vms_evt
0230 - 12:35:44.83 03 - 04 .....| vms_evt Reset on mailbox BKM3/@ipcc_e_1_2 : tout = -1
0231 - 12:35:44.87 03 - 05 ---->| vms_get_mbx_chan
0232 - 12:35:44.87 03 - 05 .....| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0233 - 12:35:44.87 03 - 05 .....| vms_get_mbx_chan Returning key 1a0
0234 - 12:35:44.94 03 - 05 ----<| vms_get_mbx_chan
0235 - 12:35:44.83 03 - 04 .....| vms_evt rc = 0
0236 - 12:35:44.98 03 - 04 ----<| vms_evt
0237 - 12:35:45.00 03 - 03 ----<| xcsResetEventSem
0238 - 12:35:45.03 03 - 03 ---->| xcsReleaseMutexSem
0239 - 12:35:45.05 03 - 04 ---->| xllSemRel
0240 - 12:35:45.07 03 - 05 ---->| vms_mtx
0241 - 12:35:45.07 03 - 05 .....| vms_mtx :- Unlocking BKM3/@ipcc_m_1_10 - timeout: -1
0242 - 12:35:45.11 03 - 06 ---->| vms_get_lock
0243 - 12:35:45.13 03 - 06 ----<| vms_get_lock
0244 - 12:35:45.16 03 - 05 ----<| vms_mtx
```

Questa visualizzazione mostra il numero di riga nel buffer di cronologia, l'ora di creazione del messaggio, il numero del canale, la profondità dello stack al momento della creazione del messaggio e il nome della funzione. All'apertura della LU, è stato definito il numero massimo di voci di messaggi nella cronologia. Quando il buffer è pieno, MONMQ si sposta alla prima voce e sovrascrive i messaggi iniziando dal primo. La generazione di un FFST durante le funzioni di traccia ne provoca la disattivazione nel punto in cui si è verificato il malfunzionamento per errore di sottoprocesso. In tal modo si previene il riempimento del buffer con messaggi di traccia generati da routine di errore. Di conseguenza, l'ultimo messaggio visualizzato nel buffer di cronologia è quello relativo al punto in cui è stato generato FFST.

---

## Visualizzazione di tutte le sezioni globali relative a MQSeries sul nodo corrente

SHOW GLOBALS

Questo comando visualizza tutte le sezioni globali relative a MQSeries sul nodo corrente.

## SHOW MUTEX

```
MQS1_shm_00000000 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=6128/383
MQS1_shm_01300010 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=1904/119
MQS1_shm_012c000f (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=464/87
MQS1_shm_012c000e (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=4112/514
MQS1_shm_012c000d (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=240/30
MQS1_shm_012c000c (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=528/132
MQS1_shm_012c000b (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=272/51
MQS1_shm_012c000a (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=4112/771
MQS1_shm_012c0009 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=272/51
MQS1_shm_012c0008 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=144/27
MQS1_shm_012c0007 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=528/99
MQS1_shm_012c0006 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=16/2
MQS1_shm_012c0005 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=128/24
MQS1_shm_012c0004 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=1968/492
MQS1_shm_012c0003 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=1904/476
MQS1_shm_012c0002 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=304/76
MQS1_shm_012c0001 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=1904/595
MQS1_shm_01280000 (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=16/6
MQS1_shm_fffffffe (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=16/0
MQS1_shm_fffffff (00000000) WRT DZRO PRM SYS Pgltcnt/Refcnt=144/63
```

---

## Segnalazioni al sottoprocesso di destinazione per l'invio della tabella mutex al processo di traccia del client

SHOW MUTEX [chl=*range*]

Questo comando comunica al sottoprocesso di destinazione di inviare il contenuto della relativa tabella mutex interna al processo di traccia del client. Si noti l'importanza di impostare i corretti bit di maschera della traccia in modo da permettere la visualizzazione di questo tipo di dati informativi al client. Occorre abilitare i bit INF e DGN nella maschera di traccia per questo canale (consultare "Abilitazione o disabilitazione del bit di maschera" a pagina 367). Ad esempio, il comando **show mutex chl=2** visualizza:



Mutex Utilisation for Process AMQZXMA0.EXE - Pid 248 \*\*\*

```

0960 - Lock ID: 0100013c - Name: BKM3/@ipcc_m_1_24
0961 - Lock ID: 020006ca - Name: BKM3/@ipcc_m_1_23
0962 - Lock ID: 0b00061e - Name: BKM3/@ipcc_m_1_22
0963 - Lock ID: 0900068e - Name: BKM3/@ipcc_m_1_21
0964 - Lock ID: 0b00032f - Name: BKM3/@ipcc_m_1_20
0965 - Lock ID: 210006eb - Name: BKM3/@ipcc_m_1_19
0966 - Lock ID: 07000742 - Name: BKM3/@ipcc_m_1_18
0967 - Lock ID: 1e000075 - Name: BKM3/@ipcc_m_1_17
0968 - Lock ID: 0c0004dd - Name: BKM3_m_1_45
0969 - Lock ID: 0d00035a - Name: BKM3/@ipcc_m_1_16
0970 - Lock ID: 190000a1 - Name: BKM3/@ipcc_m_1_15
0971 - Lock ID: 1a0005a3 - Name: BKM3/@ipcc_m_1_14
0972 - Lock ID: 14000628 - Name: BKM3/@ipcc_m_1_13
0973 - Lock ID: 130005f3 - Name: BKM3/@ipcc_m_1_12
0974 - Lock ID: 0f0000dc - Name: BKM3_m_1_43
0975 - Lock ID: 02000095 - Name: BKM3_m_1_42
0976 - Lock ID: 2200053e - Name: BKM3_m_1_41
0977 - Lock ID: 020000fc - Name: BKM3_m_1_40
0978 - Lock ID: 31000113 - Name: BKM3_m_1_39
0979 - Lock ID: 02000555 - Name: BKM3_m_1_38
0980 - Lock ID: 2e000389 - Name: BKM3_m_1_37
0981 - Lock ID: 2300011f - Name: BKM3_m_1_36
0982 - Lock ID: 02000109 - Name: BKM3_m_1_35
0983 - Lock ID: 02000327 - Name: BKM3_m_1_34
0984 - Lock ID: 020004a8 - Name: BKM3_m_1_33
0985 - Lock ID: 02000453 - Name: BKM3_m_1_32
0986 - Lock ID: 260007ad - Name: BKM3_m_1_31
0987 - Lock ID: 0200060c - Name: BKM3_m_1_30

```

I dati mostrano il numero di riga nel file della cronologia, il nome mutex e l'ID del blocco di sistema.

---

## Segnalazioni al sottoprocesso di destinazione per l'invio della tabella degli eventi interni al processo di traccia del client

SHOW EVENTS [chl=*range*]

Questo comando comunica al sottoprocesso di destinazione di inviare il contenuto della relativa tabella degli eventi interni al processo di traccia del client. Si noti l'importanza di impostare i corretti bit della maschera della traccia in modo da permettere la visualizzazione di questo tipo di dati informativi al client. Occorre abilitare i bit INF e DGN nella maschera della traccia per questo canale (consultare "Abilitazione o disabilitazione del bit di maschera" a pagina 367). Ad esempio, il comando **show events chl=2** visualizza:

## SHOW EVENTS

Event Utilisation for Process AMQZXMA0.EXE - Pid 248 \*\*\*

```
1037 - Channel: 000003e0 - Name: BKM3/@ipcc_e_1_19
1038 - Channel: 000003d0 - Name: BKM3/@ipcc_e_1_18
1039 - Channel: 000003c0 - Name: BKM3/@ipcc_e_1_17
1040 - Channel: 000003b0 - Name: BKM3/@ipcc_e_1_14
1041 - Channel: 000003a0 - Name: BKM3/@ipcc_e_1_13
1042 - Channel: 00000390 - Name: BKM3/@ipcc_e_1_12
1043 - Channel: 00000380 - Name: BKM3/@ipcc_e_1_10
1044 - Channel: 00000370 - Name: BKM3/@ipcc_e_1_9
1045 - Channel: 00000360 - Name: BKM3/@ipcc_e_1_8
1046 - Channel: 00000330 - Name: BKM3/@ipcc_e_1_7
1047 - Channel: 00000300 - Name: BKM3/@ipcc_e_1_6
1048 - Channel: 000002f0 - Name: BKM3/@ipcc_e_1_5
1049 - Channel: 000002b0 - Name: BKM3_e_1_11
1050 - Channel: 000002a0 - Name: BKM3_e_1_10
1051 - Channel: 00000290 - Name: BKM3_e_1_9
1052 - Channel: 00000280 - Name: BKM3_e_1_8
1053 - Channel: 00000270 - Name: BKM3_e_1_7
1054 - Channel: 00000260 - Name: BKM3_e_1_6
1055 - Channel: 00000250 - Name: BKM3_e_1_5
1056 - Channel: 00000240 - Name: BKM3_e_1_4
1057 - Channel: 00000230 - Name: BKM3_e_1_3
1058 - Channel: 00000220 - Name: BKM3_e_1_2
1059 - Channel: 00000210 - Name: BKM3_e_1_1
1060 - Channel: 00000200 - Name: BKM3_e_1_0
1061 - Channel: 000001c0 - Name: BKM3/@ipcc_e_1_4
1062 - Channel: 000001b0 - Name: BKM3/@ipcc_e_1_3
1063 - Channel: 000001a0 - Name: BKM3/@ipcc_e_1_2
1064 - Channel: 00000190 - Name: BKM3/@ipcc_e_1_1
1065 - Channel: 00000180 - Name: BKM3/@ipcc_e_1_0
1066 - *** End of data ***
```

I dati mostrano il numero di riga nel file della cronologia, il numero del canale della casella postale e il nome dell'evento.

---

## Segnalazioni al sottoprocesso di destinazione per l'invio della tabella interna della memoria condivisa mappata al processo di traccia del client

SHOW MEMORY [chl=range]

Questo comando comunica al sottoprocesso di destinazione di inviare i contenuti della relativa tabella interna della memoria condivisa mappata al processo di traccia del client. Si noti l'importanza di impostare i corretti bit di maschera della traccia in modo da permettere la visualizzazione di questo tipo di dati informativi al client. Occorre abilitare i bit INF e DGN nella maschera di traccia per questo canale. (consultare "Abilitazione o disabilitazione del bit di maschera" a pagina 367). Ad esempio, il comando **show memory chl=2** visualizza:

\*\*\* Shared Memory Utilisation for Process AMQZXMA0.EXE - pid/tid 248-1 \*\*\*

```
0942 - ShmId: 0248000f - Addr: 011d8000/01211fff - Perm: 950 - Size: 00038530 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/AMQ
0943 - ShmId: 0248000e - Addr: 00fbc000/011bdfff - Perm: 950 - Size: 002005f8 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/AMQ
0944 - ShmId: 0248000d - Addr: 00f1c000/00f39fff - Perm: 950 - Size: 0001d478 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/WLM
0945 - ShmId: 0248000c - Addr: 00cba000/00cfbfff - Perm: 944 - Size: 000405f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/HMEMSET.0
0946 - ShmId: 0248000b - Addr: 00c98000/00cb9fff - Perm: 944 - Size: 000205f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon005.0
0947 - ShmId: 0248000a - Addr: 00a96000/00c97fff - Perm: 944 - Size: 00200584 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon004.0
0948 - ShmId: 02480009 - Addr: 00a74000/00a95fff - Perm: 944 - Size: 000205f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon003.0
0949 - ShmId: 02480008 - Addr: 00a62000/00a73fff - Perm: 944 - Size: 000105f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon002.0
0950 - ShmId: 02480007 - Addr: 00a20000/00a61fff - Perm: 944 - Size: 000405f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon001.0
0951 - ShmId: 02480006 - Addr: 00908000/00909fff - Perm: 950 - Size: 00001664 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/PLU
0952 - ShmId: 02480005 - Addr: 008f8000/00907fff - Perm: 950 - Size: 0000fff8 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/IPC
0953 - ShmId: 02480004 - Addr: 00802000/008f7fff - Perm: 950 - Size: 000f4838 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/IPC
0954 - ShmId: 02480003 - Addr: 00714000/00801fff - Perm: 950 - Size: 000ec718 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/SUB
0955 - ShmId: 02480002 - Addr: 006ee000/00713fff - Perm: 944 - Size: 000253a8 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/zutSESSION
0956 - ShmId: 02480001 - Addr: 00600000/006edfff - Perm: 944 - Size: 000ec710 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/SUBPOOL.0
0957 - ShmId: 01280000 - Addr: 005f8000/005f9fff - Perm: 950 - Size: 00000454 Name: /var/mqm/errors
0958 - *** End of data ***
```

I dati visualizzano il numero di riga nel file della cronologia, l'ID della memoria condivisa, l'intervallo di indirizzi virtuale mappato, i flag utilizzati nel creare/mappare la sezione e il nome MQSeries interno assegnato alla sezione.

---

## Visualizzazione dei componenti attivi MQSeries per nome e ID esadecimali

### SHOW COMPONENTS

Questo comando visualizza tutti i componenti attivi MQSeries per nome e i relativi ID esadecimali associati. Utilizzare tali ID esadecimali in altri comandi show di MONMQ quali show functions e select component. Ad esempio, il comando **show components** visualizza:

```
00000001 - Data hardening
00000002 - Log management
00000003 - Object Catalogue
00000004 - Queue management
00000005 - Transaction Management
00000006 - Mobile Component
00000007 - Mobile Component
00000008 - Communications
0000000a - Object Authority Manager
0000000b - Logger
0000000d - LQM Kernal
0000000f - Administration App
00000010 - Administration App
00000013 - Command Server
00000014 - Remote queue processor
00000015 - XA Transaction Manager
00000016 - Data Conversion
00000017 - Common Services
00000018 - Common Services (overflow)
00000019 - Application Interface
0000001a - IPCC
0000001b - DCE Support
0000001c - Pluggable Services
0000001d - Agent
0000001e - XA Transaction Manager
0000001f - C++ Layer
00000020 - CLI
00000021 - Z Utilities
00000022 - Execution Controller
00000023 - App. Bindings
00000024 - Service Component
00000025 - Publish/Subscribe
00000026 - MMC Snap-in for Admin
00000027 - Web Administration
00000028 - KYG Services
00000029 - OVMS MQ kernel
```

---

## Visualizzazione delle funzioni in un componente specificato

### SHOW FUNCTIONS [comp=*hex*]

Questo comando visualizza tutte le funzioni in un componente specificato. È necessario inserire il componente in valore esadecimale. Utilizzare SHOW COMPONENT per visualizzare tutti i componenti MQSeries attivi. Ad esempio, il comando **show functions component=0x1f** visualizza:

## ONSTARTUP start

```
00000000 - ImqBinary::copyOut
00000001 - ImqBinary::pasteIn
00000002 - ImqCache::operator =
00000003 - ImqCache::moreBytes
00000004 - ImqCache::read
00000005 - ImqCache::resizeBuffer
00000006 - ImqCache::setDataOffset
00000007 - ImqCache::setMessageLength
00000008 - ImqCache::useEmptyBuffer
00000009 - ImqCache::write
0000000a - ImqDeadLetterHeader::pasteIn
0000000b - ImqDistributionList::openInfoPrepare
0000000c - ImqItem::structureIdIs
0000000d - ImqQueueManager::backout
0000000e - ImqQueueManager::begin
0000000f - ImqQueueManager::commit
00000010 - ImqQueueManager::connect
00000011 - ImqQueueManager::disconnect
00000012 - ImqMessageTracker::setAccountingToken
00000013 - ImqMessageTracker::setCorrelationId
00000014 - ImqMessageTracker::setGroupId
00000015 - ImqMessageTracker::setMessageId
00000016 - ImqObject::close
00000017 - ImqObject::closeTemporarily
00000018 - ImqObject::inquire
00000019 - ImqObject::open
0000001a - ImqObject::openFor
.....
```

---

## Attivazione della traccia dal punto di avvio di un processo

ONSTARTUP [start] [lu=*number*] [chl=*range*]

Questo comando permette l'attivazione della traccia dal punto di avvio di un processo. All'esecuzione di tale comando, viene definito un nome logico MQS\_DEF\_TRACE nella tabella dei nomi logici del sistema, con un nome equivalente il cui formato è: *lu channel*. All'avvio di un qualsiasi processo MQSeries, tale nome logico viene controllato nella routine di inizializzazione processi; se questo viene rilevato, il processo si connette alla LU e al numero di canale specificati. Se l'ID di canale è già assegnato, sarà utilizzato il successivo canale disponibile. Questo comando è utile quando si richiede una traccia nelle prime fasi della creazione del processo/sottoprocesso di MQSeries.

---

## Impedire al processo di MQSeries di eseguire la traccia immediatamente all'avvio

ONSTARTUP [stop]

Questo comando annulla l'assegnazione del nome logico MQS\_DEF\_TRACE dalla tabella dei nomi logici del sistema, impedendo così che i processi MQSeries eseguano la traccia immediatamente all'avvio.

---

## Connessione del sottoprocesso di destinazione al canale specificato

CONNECT pid *number* [tid=*number*] [chl=*range*]

Questo comando comunica al sottoprocesso di destinazione di connettersi al canale specificato. Qualora non sia specificato alcun canale, sarà utilizzato il primo disponibile.

---

## Disconnessione del sottoprocesso di destinazione dal canale specificato

DISCONNECT [chl=*range*]

Questo comando comunica al sottoprocesso di destinazione di disconnettersi dal canale specificato.

---

## Visualizzazione del messaggio di traccia in tempo reale scritto alla casella di posta della traccia delle LU

TRACE START [node=*string*]

Questo comando avvia un processo di traccia del client per visualizzare il messaggio di traccia in tempo reale scritto alla casella di posta di traccia delle LU. Il parametro opzionale di nodo crea una finestra sul nodo specificato dirigendovi l'output.

---

## Separazione e termine del processo del client corrente

TRACE STOP

Questo comando determina la separazione del processo del client corrente dalla casella di posta di traccia e il relativo termine. Tutti i sottoprocessi che attualmente scrivono alla casella di posta vengono disabilitati.

MQT> trace stop

```
Circular buffering has been disabled for process 24d thread 1
Circular buffering has been disabled for process 24c thread 1
Circular buffering has been disabled for process 24b thread 1
Circular buffering has been disabled for process 248 thread 1
Disconnecting thread pid : 24d, tid : 1 from channel 0 ..... OK
Disconnecting thread pid : 24c, tid : 1 from channel 1 ..... OK
Disconnecting thread pid : 24b, tid : 1 from channel 2 ..... OK
Disconnecting thread pid : 248, tid : 1 from channel 3 .....OK
```

\*\*\* Trace ended - no processes connected \*\*\*

L'output sopra riportato appare nella finestra del client di traccia.

---

## Specificazione dei dati di traccia

SELECT [component] AND/OR [function] OR [*fname*]

Questo comando permette di specificare fino a otto combinazioni di componente/funzioni da tracciare. Vengono filtrati tutti gli altri dati della traccia. È possibile specificare un nome di funzione o un componente oppure una coppia componente/funzione. Il componente o la funzione selezionati, se validi, vengono scritti nella tabella filtro. Nel caso in cui non venga specificato nulla, per impostazione predefinita vengono tracciate tutte le funzioni componente/funzioni.

Immettendo il comando **SELECT** senza alcun parametro saranno visualizzati i contenuti della tabella filtro. Ogni riga di output presenterà l'indice di tabella, il componente e la funzione in esadecimale e il nome della funzione in testo.

## SELECT

L'immissione di un solo componente comporta la traccia di tutte le funzioni in esso comprese. Ciò viene mostrato come 0xffff a fronte del valore della funzione.

Ad esempio, il comando **SELECT** visualizza:

```
Ch1:0 - Cmp/fnc selection criteria
ALL component/functions
```

Il seguente insieme di comandi,

```
MQT>select fname="kill"
MQT>select comp=0x1f
MQT>select comp=0x20 func=0x3
MQT>select
```

visualizza:

```
Ch1:0 - Cmp/fnc selection criteria
Idx: 0 - Cmp: 00000029 - Fnc: 00000005 - Name - kill
Idx: 1 - Cmp: 00000019 - Fnc: 0000ffff - Name -
Idx: 2 - Cmp: 00000016 - Fnc: 00000003 - Name - vqiAddCacheEntry
```

---

## Rimozione di una singola voce dalla tabella filtri della traccia

DESELECT INDEX=<0:7>

Questo comando rimuove una singola voce dalla tabella filtri della traccia, come specificato dal parametro *index* della tabella. Tutti i componenti/funzioni vengono tracciati quando le otto voci sono vuote. Ad esempio, un comando **select** visualizza le seguenti voci con i loro indici:

```
Ch1:0 - Cmp/fnc selection criteria
Idx: 0 - Cmp: 00000029 - Fnc: 00000005 - Name - kill
Idx: 1 - Cmp: 00000019 - Fnc: 0000ffff - Name -
Idx: 2 - Cmp: 00000016 - Fnc: 00000003 - Name - vqiAddCacheEntry
```

I seguenti comandi **deselect** rimuovono i processi o le funzioni specificati:

```
MQT> dese1 index=0
MQT> dese1 index=2

MQT>select
Ch1:0 - Cmp/fnc selection criteria
Idx: 1 - Cmp: 00000019 - Fnc: 0000ffff - Name -
-----
MQT> deselect index=1
MQT> select

Ch1:0 - Cmp/fnc selection criteria
ALL component/functions
-----
```

---

## Scrittura di messaggi di traccia in un file binario da parte di un processo di client

OPEN BINARY [filename=*string*]

Questo comando apre un file binario di messaggio di traccia e provoca la scrittura in tempo reale di messaggi di traccia da parte del processo del client nello stesso file. Questo file può essere in seguito utilizzato per analizzare le prestazioni delle applicazioni MQSeries. Il nome file predefinito è `mqc_root:[mqm.errors]mqc_buffer_xx.bin` (dove *xx* è il numero di LU).

---

## Chiusura di un file binario di messaggi di traccia

CLOSE BINARY

Questo comando chiude il file binario di traccia di LU specificato.

---

## Scrittura di messaggi di traccia in un file di testo da parte di un processo di client

OPEN TEXT [*filename=string*]

Questo comando apre un file di testo leggibile e provoca la scrittura di messaggi binari di traccia formattati da parte del processo del client nello stesso file. Questo file può essere esaminato in seguito con il semplice utilizzo del comando DCL type o edit. Il nome file predefinito è `mqc_root:[mqm.errors]mqc_buffer_xx.lis` (dove *xx* è il numero di LU). L'utilizzo di questo tipo di file si rivela vantaggioso poiché non richiede un'elaborazione preliminare per essere letto. Tuttavia gli svantaggi sono dovuti a un maggior consumo di spazio disco rispetto a un file binario.

---

## Chiusura di file di testo di messaggi di traccia

CLOSE TEXT

Questo comando chiude il file di testo della traccia di LU specificato.

---

## Messaggi di timestamp

ENABLE TIMESTAMP [*chl=range*]

Questo comando imposta i flag di Timestamp nella tabella definizione del canale. Utilizzare questo comando per costringere i processi MQSeries a inserire la data/ora in ciascun messaggio. Questo flag deve essere impostato durante l'utilizzo di un file di traccia binario per l'analisi delle prestazioni.

---

## Arresto dei messaggi di timestamping

DISABLE TIMESTAMP [*chl=range*]

Questo comando rimuove l'impostazione del flag di Timestamp nella tabella definizione del canale (consultare "Messaggi di timestamp").

---

## Attivazione traccia

ENABLE TRACE [*chl=range*]

Questo comando imposta il flag di traccia RTime nella tabella definizione del canale. Utilizzare questo flag per abilitare o disabilitare l'invio di messaggi di traccia a un client di traccia. Ad esempio, quando un sottoprocesso è connesso ed è

## ENABLE TRACE

presente un client di traccia, TRACE START può essere utilizzato per avviare o chiudere il canale invece di disconnettere il relativo sottoprocesso.

---

## Disattivazione traccia

DISABLE TRACE [chl=*range*]

Questo comando rimuove l'impostazione del flag di traccia RTime nella tabella definizione del canale (consultare "Attivazione traccia" a pagina 365).

---

## Salvataggio cronologia messaggi

ENABLE HISTORY [chl=*range*]

Questo comando imposta il flag History nella tabella definizione del canale per i canali specificati. Questo comando comunica al sottoprocesso connesso di scrivere messaggi di traccia nel buffer circolare della LU. Poiché la scrittura del messaggio viene eseguita dal processo tracciato, non è necessaria l'esistenza di un processo di client. La dimensione del buffer circolare della traccia viene definita durante la creazione della LU tramite il comando **open**. Tale buffer ritorna all'inizio quando viene scritta l'ultima voce. Il campo del record Next Log nella tabella definizione della LU specifica dove verrà scritto nel buffer il record successivo.

---

## Disabilitazione della cronologia messaggi

DISABLE HISTORY [chl=*range*]

Questo comando rimuove l'impostazione del flag History nella tabella definizione del canale per i canali specificati. Consultare "Salvataggio cronologia messaggi".

---

## Eliminazione della cronologia messaggi

DELETE HISTORY [chl=*range*]

Questo comando elimina tutti i messaggi nel buffer di cronologia circolare. Questo comando può essere eseguito anche durante la scrittura nel buffer da parte dei processi, per cui non è necessario disabilitare la cronologia prima di eliminarla.

---

## Impostazione della profondità di cronologia

SET [depth]

Questo comando controlla la massima profondità dello stack da visualizzare nella finestra del client della traccia. I messaggi più profondi di tale valore non verranno visualizzati, tuttavia appariranno nel file binario della traccia e nel buffer di cronologia, se è abilitato. Il valore predefinito zero consente la visualizzazione di tutti i messaggi di qualsiasi profondità dello stack. Si consiglia di impostare un valore molto basso (ad esempio, 1) durante la scrittura di dati di analisi nel file binario. La visualizzazione completa dello stack influirà negativamente sulle prestazioni di un processo del client.

---

## Azzeramento dello stack e dei dati di cronologia per un canale

SET [free] [chl=*range*]



Questo comando azzera il canale specificato. Vengono eliminati tutti i dati esistenti di stack e cronologia e viene annullata l'allocazione del canale, che sarà disponibile per un nuovo utilizzo.

## Abilitazione o disabilitazione del bit di maschera

```
SET [mask=var] [chl=range]
```

Questo comando abilita o disabilita un bit di maschera nel campo bit di maschera dei sottoprocessi connessi. Ogni bit rappresenta un tipo di messaggio generato da un processo MQSeries. È possibile utilizzare questo comando per filtrare il tipo di messaggi che bisogna tracciare. I tipi di messaggio sono i seguenti:

```
MQT>set mask = 0xffffffff chl=1
MQT>show mask chl=1
```

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

Per specificare una combinazione di questi tipi di messaggio delimitare ogni tipo di maschera con un simbolo OR, ad esempio:

```
MQT>set mask =0x0 chl=1 MQT>show mask chl=1
```

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

```
MQT>set mask = mtx | evt | fent chl=1
MQT>show mask chl=1
```

## SET mask

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

Ogni maschera comprende otto tipi di maschera utilizzabili alternativamente per abilitare o disabilitare un particolare tipo di messaggio. Ad esempio, se si è interessati soltanto ai punti della voce funzione, si immetterà il comando **set mask = fent**.

---

## Impostazione di un colore per un canale

SET COLOR [chl=*range*]

Questo comando associa un colore a un canale specificato. Tutte le visualizzazioni relative a questo canale verranno mostrate in questo colore fino a quando il colore viene cambiato oppure il canale azzerato. Questo comando è utile per evidenziare o distinguere i messaggi di sottoprocessi differenti in una singola visualizzazione. Ad esempio, i comandi:

```
MQT> set color=yellow chl=2
MQT> set color=blue chl=0
MQT> sho chan chl=0:3 connected
```

visualizzano:

Chl	Pid/Tid	Mailbox	Stack	History	RTime	Time	Mask	Process Name
0	00000245/1	800b0330	4	0	0	0	fffffff	AMQZLAA0.EXE
1	00000244/1	800b0200	8	0	0	0	fffffff	RUNMQCHI.EXE
2	00000243/1	800b01e0	10	0	0	0	fffffff	AMQRRMFA.EXE
3	00000242/1	800b01c0	4	0	0	0	fffffff	AMQZLLP0.EXE

dove il canale 0 è blu e il canale 2 è giallo.

---

## Reindirizzare l'output a un file

SET OUTPUT [filename=*string*]

Questo comando dirige tutti gli output a un file specificato e disabilita l'output per la visualizzazione. Il comando **output**, quando viene utilizzato come parametro con altri comandi, è valido soltanto per quel comando. Si noti che gli errori continueranno a essere riportati sul dispositivo di visualizzazione e non su file. Solo i dati di traccia validi saranno scritti nel file specificato.

---

## Analisi del file binario di traccia

ANALYSE [component] [function] [unit=*xx*]

Questo comando analizza il contenuto del file binario di traccia utilizzato precedentemente in una sessione di traccia. Sebbene sia possibile specificare il componente o la funzione da analizzare, ciò è efficace solo se il file contiene i dati relativi a tale componente o funzione. Ad esempio, se durante la creazione del file è stata specificata una maschera di traccia o selezionato uno specifico componente, allora nel file binario sono presenti soltanto queste voci, perciò i componenti o le funzioni esterne a tale criterio non possono essere utilizzate nell'analisi.

**Nota:** Se si utilizza il comando completo, verificare che ANALYSE sia scritto con la S. Non confondere questo comando MONMQ con il comando ANALYZE di OpenVMS. I due comandi sono differenti l'uno dall'altro.

Il parametro unit viene utilizzato per specificare l'unità di tempo per l'analisi e può avere uno dei seguenti valori (xx) - secondi, millisecondi, microsecondi, nanosecondi. Il valore predefinito è millisecondi.

Ad esempio, per visualizzare l'output in microsecondi, utilizzare il comando **analyse unit=micro**. Di seguito è mostrato un esempio di output per questo comando:

## ANALYSE

```

=====
                        COMPONENT :- Common Services
=====
Calls  Minimum      Average      Maximum      Total      Function
42     0.00         66.41       311.78      2789.15    xcsRequestMutexSem
42     0.00         96.98       222.64      4072.98    xcsReleaseMutexSem
6      0.00        138.50       286.11       831.00     xcsResetEventSem
5      0.00     10112.60    10159.51    50563.01    xcsWaitEventSem
6      0.00         564.74     1029.23     3388.45    xcsCheckExtendMemory
42     0.00          51.32       266.86     2155.41    xllSemReq
42     0.00          73.05       159.17     3068.16    xllSemRel
=====
                        COMPONENT :- Common Services (overflow)
=====
Calls  Minimum      Average      Maximum      Total      Function
36     0.00         41.15       122.06     1481.35    xcsCheckProcess
6      0.00         37.92       68.35      227.52     xihGetConnSPDetailsFromList
6      0.00         65.26       114.25     391.58     xihHANDLEtoSUBPOOLFn
6      0.00         12.69       23.44     1267.49    xihGetNextSetConnDetailsFromList
6      0.00         12.53       22.46       75.19     xcsRequestThreadMutexSem
6      0.00         12.37       22.46       74.21     xcsReleaseThreadMutexSem
=====
                        COMPONENT :- IPCC
=====
Calls  Minimum      Average      Maximum      Total      Function
5      0.00     10589.97    11029.84    52949.85    xcpReceiveOnLink
=====
                        COMPONENT :- CLI
=====
Calls  Minimum      Average      Maximum      Total      Function
6      0.00          1.95         1.95        11.72      zapInquireStatus
=====
                        COMPONENT :- Execution Controller
=====
Calls  Minimum      Average      Maximum      Total      Function
6      0.00         954.46     3275.18    11726.79    zxcProcessChildren
6      0.00         12.69       22.46       76.17     zxcStartWLMServer
=====
                        COMPONENT :- OVMS MQ kernel
=====
Calls  Minimum      Average      Maximum      Total      Function
36     0.00         15.49       99.60       557.58     kill
6      0.00          0.65         3.91         3.91     vms_mapgbl
84     0.00         11.17       88.16       938.68     vms_get_lock
84     0.00         42.41      221.94     3562.54     vms_mtx
12     0.00         44.51      149.40       534.14     vms_get_mbx_chan
11     0.00     4651.77    10137.05    51169.42    vms_evt
6      0.00          1.63         4.88          9.76     vms_check_health
=====

```

Le colonne sono:

<b>Calls</b>	Il numero di voci nella funzione durante la sessione di traccia.
<b>Minimum</b>	Il tempo più breve trascorso nella funzione.
<b>Average</b>	Il tempo totale trascorso in tutte le chiamate alla funzione diviso per il numero delle chiamate.
<b>Maximum</b>	Il tempo più lungo trascorso nella funzione.
<b>Total</b>	Il tempo totale trascorso in questa funzione per tutte le chiamate.
<b>Function</b>	Il nome della funzione.

**Nota:** L'obiettivo dell'analisi è il contenuto del file di traccia binario. Spetta all'utente definire i limiti dell'analisi aprendo un file di traccia binario e abilitando/disabilitando la traccia al momento opportuno.

---

## Visualizzazione dello stato corrente dei sottoprocessi di MQSeries

FFST [chl=*range*]

Questo comando obbliga il sottoprocesso connesso al canale di destinazione a provocare un FFST. Questo comando NON ha effetto sul percorso di esecuzione dei sottoprocessi di destinazione e consente di ottenere un'istantanea dello stato corrente di qualsiasi sottoprocesso MQSeries. La parte FFST contiene l'utilizzo di risorsa dei sottoprocessi, privilegi e altre informazioni utili di sistema. È indicato in modo evidente nell'intestazione che il FFST è stato creato da MONMQ (vedere sotto) e NON è il risultato di un errore.

Il seguente è un esempio di output:

```
MQSeries First Failure Symptom Report
=====
Date/Time          :- Wednesday November 12  10:59:38 GMT 2000
Host Name          :- CATWMN (Unknown)
PIDS               :- 5697175
LVLS               :- 510
Product Long Name  :- MQSeries for OpenVMS Alpha
Vendor             :- IBM
Probe Id           :- VM026000
Application Name   :- MQM
Component          :- vms_evt
Build Date         :- Oct 22 2000 (Collector)
Userid             :- [400,400] (SYSTEM)
Program Name       :- AMQZXMA0.EXE
Process            :- 00000248
Thread             :- 00000001
QueueManager       :- BKM3
Major Errorcode    :- xecF_E_UNEXPECTED_SYSTEM_RC
Minor Errorcode    :- OK
Probe Type         :- MSGAMQ6119
Probe Severity     :- 2
Probe Description  :- AMQ6119: An internal MQSeries error has occurred
                    (*** FORCED FFST BY USER ***)
Comment1           :- *** FORCED FFST BY USER ***
Comment2           :- -SYSTEM-S-NORMAL, normal successful completion

etc.....
```

---

## Chiusura di traccia e uscita da MONMQ

EXIT

Questo comando esegue un comando CLOSE ed esce da MONMQ.

---

## Abbandono di MONMQ senza chiudere la traccia

QUIT

Questo comando non esegue un comando CLOSE, ma esce da MONMQ. Questo comando è utile se si desidera lasciare la traccia in esecuzione senza chiudere MONMQ. Alla successiva attivazione di MONMQ viene ripresa la precedente sessione di traccia.

## Gestione della memoria condivisa con MONMQ

In circostanze insolite, ad esempio un malfunzionamento del Queue Manager o l'arresto forzato con il comando stop /id di OpenVMS, è possibile che i segmenti della memoria condivisa di MQSeries non siano automaticamente eliminati dal Queue Manager. In tale eventualità, non sarà possibile riavviare il Queue Manager, poiché **strmqm** riporterà che il Queue Manager è già in esecuzione.

MONMQ può elencare la memoria condivisa MQ, ovvero le sezioni globali attualmente esistenti, ed eliminarle.

**Nota:** Assicurarsi che tutti i Queue Manager siano arrestati prima di utilizzare l'utilità MONMQ per eliminare i segmenti di memoria condivisa. L'eliminazione della memoria condivisa di un Queue Manager in esecuzione comporta il malfunzionamento del Queue Manager, probabilmente la corruzione dei file di code.

Il comando MONMQ SHOW PROCESS può essere utilizzato per assicurarsi che non vi siano processi MQ in esecuzione. Nel caso fosse impossibile arrestare con il comando endmq i processi in esecuzione su un Queue Manager che abbia subito un malfunzionamento, sarà possibile utilizzare il comando stop /id=<pid> di OpenVMS DCL.

Controllare che non vi siano processi di MQSeries in esecuzione mediante l'utilizzo del seguente comando:

```
MQT> show process
```

```
MQ Processes
PID      Proc Name      Image      Process      WS Size      WS Peak      Virt Peak      Gbl Pg Cnt Prc Pg Cnt Total Mem
-----
List the shared memory global sections that currently exist

MQT> show globals
MQS1_shm_2695000a (00000000) WRT   DZRO PRM SYS Pgltcnt/Refcnt=4112/514
MQS1_shm_26950009 (00000000) WRT   DZRO PRM SYS Pgltcnt/Refcnt=272/34
MQS1_shm_ffffffff (00000000) WRT   DZRO TMP SYS Pgltcnt/Refcnt=144/63
MQS1_shm_00000000 (00000000) WRT   DZRO TMP SYS Pgltcnt/Refcnt=6304/394
```

Elencare le sezioni globali della memoria condivisa attualmente esistente tramite il comando:

```
MQT> show globals
```

Eliminare queste sezioni globali:

```
MQT> delete
Deleted global section: MQS1_shm_2695000a
Deleted global section: MQS1_shm_26950009
Deleted global section: MQS1_shm_ffffffff
sys$delgbl - unable to delete section MQS1_shm_00000000
```

L'errore di eliminazione della sezione MQS\_shm\_00000000 è previsto, poiché questa sezione è utilizzata da MONMQ. È possibile uscire adesso da MONMQ emettendo il comando:

```
MQT> exit
```

È possibile utilizzare il comando delete dall'interno di uno script se si è certi che tutti i processi del Queue Manager siano stati arrestati:

```
$ monmq delete
Deleted global section: MQS1_shm_2695000a
Deleted global section: MQS1_shm_26950009
Deleted global section: MQS1_shm_ffffffff
sys$delgbl - unable to delete section MQS1_shm_00000000
```

## Script e macro in MONMQ

È possibile eseguire uno script di comandi MONMQ dall'interno di MONMQ o dal prompt dei comandi. Gli script possono essere utili per raggruppare un insieme di dati o per configurare l'ambiente di MONMQ. All'avvio di MONMQ, viene eseguito uno script da SYSS\$MANAGER:MQS\_TRACE\_STARTUP.MQT in modo da configurare le variabili di traccia in MONMQ.

**Nota:** Nel caso in cui lo script non sia presente nella directory corrente, sarà necessario inserire tra apici il nome del percorso completo. Ad esempio:

```
MQT> ! "sys$manager:test.mqt"
```

È inoltre possibile definire una macro per abbreviare attività comuni o ripetitive. Una dichiarazione di macro è composta da tre parti:

1. La prima parte è il nome della macro, che deve essere un nome di comando unico.
2. La seconda parte è il corpo della macro, che può estendersi su più righe e consiste in un elenco di comandi MQSeries. Il corpo della macro è delimitato da { e }.  
Il prompt di MONMQ cambia in **\*\*MACRO>** quando si dichiara il corpo di una macro con più righe. Qualsiasi \$n, dove n è una singola cifra, viene sostituito con il parametro n sulla riga comandi della macro.
3. La terza parte della definizione di macro è una breve guida testuale che viene visualizzata quando si utilizza il comando help <nomemacro>. Il testo della guida deve essere racchiuso tra apici.

Quando si dichiara una macro, occorre considerare i tempi di caricamento. Una macro esegue elaborazioni molto rapidamente, ma alcuni comandi di MONMQ comunicano a processi remoti di eseguire un'attività, la quale deve essere completata prima dell'avvio del successivo comando della macro.

Per tale motivo, è talvolta necessario attendere brevemente. A tale scopo, utilizzare il comando **sleep**, che prevede un parametro di ritardo specificato in decimi di secondo.

È possibile immettere i seguenti comandi per creare una macro che disconnetta un canale, azzeri la maschera della traccia e renda il canale libero.

## Script e macro

**Nota:** È possibile posizionare su una linea più di un comando MONMQ utilizzando il “;” come separatore.

```
MQT> declare tmpchl intrange "variable to hold a chl range temporarily"
MQT> macro remove { set tmpchl = chl ; dis chl= $1 ; sleep delay=5
**MACRO> set mask=0xffffffff chl= $1 ; set free chl = $1
**MACRO> set chl=tmpchl
**MACRO> } "A macro to disconnect and free channels Param: chl number"
MQT> help remove

VERB remove:
A macro to disconnect and free channels Param: chl number

Macro text:
set tmpchl = chl ; dis chl= $1 ; sleep delay=5 ; set mask=0xffffffff chl=$1
; set free chl = $1 ; set chl=tmpchl
MQT>remove 4
ok - process disconnected process 282 from channel 4
```

## Esempio di sessione di traccia

Questa sezione descrive una tipica sessione di traccia illustrando ogni comando MONMQ in sequenza. Questo esempio traccia il controllore di esecuzione di un Queue Manager in esecuzione e il relativo sottoprocesso principale di agenti.

Prima di iniziare la traccia, è necessario che le seguenti condizioni siano soddisfatte:

- Avviare il Queue Manager da tracciare - STRMQM BKM3
- Verificare che sys\$manager:mqs\_trace\_startup.mqt non abbia comandi aggiuntivi diversi da quelli preinstallati.
- Verificare che il MQS\_DEF\_TRACE logico NON sia definito. In caso contrario, eseguire un **ONSTARTUP END** in MONMQ.

Avviare momq.

```
>monmq
```

Viene visualizzato il prompt di MONMQ:

```
MQT>
```

Aprire una singola LU con ID uguale a zero, con dieci canali e un buffer di 100 messaggi (Nota: 100 messaggi è un valore troppo piccolo a scopo di traccia normale. Il valore adeguato è normalmente 1000).

```
MQT> open lu=0 nochls=10 buffer=100
ok - LU:0 opened
```

Visualizzare la definizione di LU1:

```
MQT> show seg lu=1
```



## Esempio di sessione di traccia

```
Trace LU           : 1
Mailbox name      : MQS_TRC_MBX_1
Device name       : MBA431:
Status            : Disabled
Mailbox channel   : 384
History buffer size : 100
Threads mapped #  : 1
Time stamping     : Enabled
Max channels #    : 10
Display depth     : 0
Text filename     :
Binary filename   :
Last status       : 1
Connection map[0] : 0
=====
```

Visualizzare i processi MQSeries:

```
MQT> show process
```

PID	Proc_Name	Image	Process	WS_Size	WS_Peak	Virt_Peak	Gbl_Pg_Cnt	Prc_Pg_Cnt	Total_Mem
2A00023D	BKM1_AG	AMQZLAA0	Agent	23152	16576	203776	3616	12960	16576
2A00023C	BKM1_CI	RUNMQCHI	Run Chan Init	8752	6208	180832	1840	4368	6208
2A00023B	BKM1_RM	AMQRRMFA	Repository Mgr	11152	8144	185360	2224	5920	8144
2A00023A	BKM1_CP	AMQZLLP0	Checkpoint	8752	6384	185952	1920	4464	6384
2A000239	BKM1_LG	AMQHASMx	Logger	8752	6288	182016	2080	4208	6288
2A000238	BKM1_EC	AMQZXMA0	EC	20752	15232	203792	3536	11680	15216
2A000128	_FTA4:	MONMQ	MONMQ Utility	8400	8528	198736	2224	3584	5808
									----- 52112

Identificare il controllore di esecuzione e il processo di agente e collegarli, rispettivamente, ai canali uno e due.

```
MQT>connect pid=0x238 tid=1 lu=1 chl=1
MQT>connect pid=0x23D tid=1 lu=1 chl=2
```

controllo dei dettagli di connessione.

```
MQT>show channel full connected lu=1
```

## Esempio di sessione di traccia

```
Pid/Tid          : 2a000bc/1
Status           : *** Connected ***
Process name     : AMQZXMA0.EXE
Assigned LU      : 1
Channel no.      : 1
Mailbox channel  : 800c03f0
Current stack depth : 4
Circular logging  : Disabled
Next log entry   : 0
Realtime tracing : Disabled
Time stamping    : Disabled
Trace mask       : ffffffff
Step mode        : Off
No Wait          : On
Last QIO status  : 0
Mapped address   : 1242000-126bfff
=====
```

```
Pid/Tid          : 2a000c1/1
Status           : *** Connected ***
Process name     : AMQZLAA0.EXE
Assigned LU      : 1
Channel no.      : 2
Mailbox channel  : 800c03f0
Current stack depth : 4
Circular logging  : Disabled
Next log entry   : 0
Realtime tracing : Disabled
Time stamping    : Disabled
Trace mask       : ffffffff
Step mode        : Off
No Wait          : On
Last QIO status  : 0
Mapped address   : 1376000-139ffff
=====
```

Impostare valori predefiniti per chl, lu e tid per evitare di immetterli ogni volta per i successivi comandi.

```
MQT>default chl=1:2 lu=1 tid=1
```

Impostare colori di canale in modo da operare una distinzione tra messaggi di traccia diversi. Si noti che il parametro chl viene specificato in questi due comandi poiché è stato utilizzato il valore predefinito (1: 2) e i canali sarebbero stati impostati entrambi su giallo e poi ciano.

```
MQT>set chl=1 color=yellow
MQT>set chl=2 color=cyan
```

Visualizzare ora i canali.

```
MQT>show channels
```

Chl	Pid/Tid	Mailbox	Stack	History	RTime	Time	Mask	Process Name
1	2a0000bc/1	800c03f0	4	0	0	0	fffffff	AMQZXMA.EXE
2	2a0000c1/1	800c0360	4	0	0	0	fffffff	AMQZLAA0.EXE

## Esempio di sessione di traccia

Una volta connessi i processi ai canali, è possibile esaminare gli stack.

```
MQT>show stacks
```

```
0001- 00:00:00.00 03 - 01 -->| ExecCtrlrMain
0002 - 00:00:00.00 03 - 02 ---->| zcpReceiveOnLink
0003 - 00:00:00.00 03 - 03 ----->| xcsWaitEventSem
0004 - 00:00:00.00 03 - 04 ----->| vms_evt
```

```
0001- 00:00:00.00 03 - 01 -->| zlaMain
0002 - 00:00:00.00 03 - 02 ---->| zcpReceiveOnLink
0003 - 00:00:00.00 03 - 03 ----->| xcsWaitEventSem
0004 - 00:00:00.00 03 - 04 ----->| vms_evt
```

Abilitando il timestamping per entrambi i canali, è possibile vedere se i loro processi sono bloccati o meno.

```
MQT>enable timestamp
MQT>show stack
```

```
0001- 00:00:00.00 03 - 01 -->| ExecCtrlrMain
0002 - 12:36:20.18 03 - 02 ---->| zcpReceiveOnLink
0003 - 12:36:20.81 03 - 03 ----->| xcsWaitEventSem
0004 - 12:36:20.83 03 - 04 ----->| vms_evt
```

```
0001- 00:00:00.00 03 - 01 -->| zlaMain
0002 - 12:36:20.18 03 - 02 ---->| zcpReceiveOnLink
0003 - 12:36:20.81 03 - 03 ----->| xcsWaitEventSem
0004 - 12:36:20.83 03 - 04 ----->| vms_evt
```

È adesso possibile rilevare la presenza di alcuni messaggi con un valido timestamp. Ciò indica che entrambi i processi sono attivi. In questo caso entrambi i processi sono in un ciclo di evento con un periodo di timeout di 10 secondi. Tale timeout può essere controllato in corrispondenza del timestamp del messaggio tramite l'esecuzione continua di un comando **show stack** fino a quando non si ottiene una modifica dei dati del timestamp.

Abilitando la cronologia è possibile costringere ogni processo a scrivere i relativi messaggi di traccia nel buffer circolare.

```
MQT>enable history
MQT> show history
```

A questo punto si stanno scrivendo tutti i messaggi di traccia nel buffer. È possibile verificare ciò mostrando la maschera della traccia e la tabella del componente/funzione.

```
MQT>show mask
```

## Esempio di sessione di traccia

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

```
Trace Mask for Channel 2
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

```
MQT> select
```

```
Ch1:1 - Cmp/fnc selection criteria
ALL component/functions
```

```
-----
Ch1:2 - Cmp/fnc selection criteria
ALL component/functions
-----
```

## Esempio di sessione di traccia

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

```
Trace Mask for Channel 2
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

Si concentri ora l'attenzione su un particolare tipo di messaggio. Si supponga, ad esempio, di essere interessati solo ai messaggi di diagnostica della memoria condivisa.

```
MQT>set mask=shm
MQT>show mask
```

Adesso entrambi i processi scriveranno nel buffer solo i messaggi di diagnostica della memoria. Cancellare il buffer, aspettare pochi secondi e riesaminare i contenuti del buffer.

```
MQT>clear history

(wait a few seconds)

MQT> show history
```

## Esempio di sessione di traccia

\*\*\* Trace History Ch1:1 \*\*\*

```
0990 - 00:00:00.00 00 - 04 ..... | vms_mapgbl key : ffffffff - addr : 0/0
0991 - 00:00:00.00 00 - 04 ..... | vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0992 - 00:00:00.00 00 - 04 ..... | vms_mapgbl key : ffffffff - addr : 0/0
0993 - 00:00:00.00 00 - 04 ..... | vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0994 - 00:00:00.00 00 - 04 ..... | vms_mapgbl key : ffffffff - addr : 0/0
0995 - 00:00:00.00 00 - 04 ..... | vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0996 - 00:00:00.00 00 - 04 ..... | vms_mapgbl key : ffffffff - addr : 0/0
0997 - 00:00:00.00 00 - 04 ..... | vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0998 - 00:00:00.00 00 - 04 ..... | vms_mapgbl key : ffffffff - addr : 0/0
0999 - 00:00:00.00 00 - 04 ..... | vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
```

\*\*\* End of buffer \*\*\*

\*\*\* Trace History Ch1: \*\*\*

\*\*\* End of buffer \*\*\*

Visualizzare ora anche i messaggi di diagnostica di tipo di evento nell'output della traccia. Attendere alcuni secondi dopo aver impostato la maschera.

```
MQT>set mask=evt | shm
(wait a few seconds)
MQT>show history
```

## Esempio di sessione di traccia

\*\*\* Trace History Ch1:1 \*\*\*

```
0977 - 00:00:00.00 00 - 04 ..... | vms_mapgbl Section MQS1_shm_ffffffff mapped at 1510000-1511fff
0978 - 00:00:00.00 00 - 04 ..... | vms_evt Event BKM3/@ipcc_e_1_2 TIMEOUT
0979 - 00:00:00.00 00 - 04 ..... | vms_evt rc = 1
0980 - 00:00:00.00 00 - 04 ..... | vms_mapgbl key : ffffffff - addr : 0/0
0981 - 00:00:00.00 00 - 04 ..... | vms_mapgbl Section MQS1_shm_ffffffff mapped at 1510000-1511fff
0982 - 00:00:00.00 00 - 04 ..... | vms_evt Reset on mailbox BKM3/@ipcc_e_1_2 : tout = -1
0983 - 00:00:00.00 00 - 05 ..... | vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0984 - 00:00:00.00 00 - 05 ..... | vms_get_mbx_chan Returning key 1a0
0985 - 00:00:00.00 00 - 04 ..... | vms_evt rc = 0
0986 - 00:00:00.00 00 - 04 ..... | vms_evt Wait on mailbox BKM3/@ipcc_e_1_2 : tout = 10000
0987 - 00:00:00.00 00 - 05 ..... | vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0988 - 00:00:00.00 00 - 05 ..... | vms_get_mbx_chan Returning key 1a0
0989 - 00:00:00.00 00 - 04 ..... | vms_evt Event BKM3/@ipcc_e_1_2 TIMEOUT
0990 - 00:00:00.00 00 - 04 ..... | vms_evt rc = 1
0991 - 00:00:00.00 00 - 04 ..... | vms_mapgbl key : ffffffff - addr : 0/0
0992 - 00:00:00.00 00 - 04 ..... | vms_mapgbl Section MQS1_shm_ffffffff mapped at 1510000-1511fff
0993 - 00:00:00.00 00 - 04 ..... | vms_evt Reset on mailbox BKM3/@ipcc_e_1_2 : tout = -1
0994 - 00:00:00.00 00 - 05 ..... | vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0995 - 00:00:00.00 00 - 05 ..... | vms_get_mbx_chan Returning key 1a0
0996 - 00:00:00.00 00 - 04 ..... | vms_evt rc = 0
0997 - 00:00:00.00 00 - 04 ..... | vms_evt Wait on mailbox BKM3/@ipcc_e_1_2 : tout = 10000
0998 - 00:00:00.00 00 - 05 ..... | vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0999 - 00:00:00.00 00 - 05 ..... | vms_get_mbx_chan Returning key 1a0
```

\*\*\* End of buffer \*\*\*

\*\*\* Trace History Ch1:2 \*\*\*

```
0982 - 00:00:00.00 01 - 04 ..... | vms_evt Event BKM3/@ipcc_e_1_7 TIMEOUT
0983 - 00:00:00.00 01 - 04 ..... | vms_evt rc = 1
0984 - 00:00:00.00 01 - 04 ..... | vms_evt Reset on mailbox BKM3/@ipcc_e_1_7 : tout = -1
0985 - 00:00:00.00 01 - 05 ..... | vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_7
0986 - 00:00:00.00 01 - 05 ..... | vms_get_mbx_chan Returning key 1c0
0987 - 00:00:00.00 01 - 04 ..... | vms_evt rc = 0
0988 - 00:00:00.00 01 - 04 ..... | vms_evt Wait on mailbox BKM3/@ipcc_e_1_7 : tout = 10000
0989 - 00:00:00.00 01 - 05 ..... | vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_7
0990 - 00:00:00.00 01 - 05 ..... | vms_get_mbx_chan Returning key 1c0
0991 - 00:00:00.00 01 - 04 ..... | vms_evt Event BKM3/@ipcc_e_1_7 TIMEOUT
0992 - 00:00:00.00 01 - 04 ..... | vms_evt rc = 1
0993 - 00:00:00.00 01 - 04 ..... | vms_evt Reset on mailbox BKM3/@ipcc_e_1_7 : tout = -1
0994 - 00:00:00.00 01 - 05 ..... | vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_7
0995 - 00:00:00.00 01 - 05 ..... | vms_get_mbx_chan Returning key 1c0
0996 - 00:00:00.00 01 - 04 ..... | vms_evt rc = 0
0997 - 00:00:00.00 01 - 04 ..... | vms_evt Wait on mailbox BKM3/@ipcc_e_1_7 : tout = 10000
0998 - 00:00:00.00 01 - 05 ..... | vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_7
0999 - 00:00:00.00 01 - 05 ..... | vms_get_mbx_chan Returning key 1c0
```

\*\*\* End of buffer \*\*\*

```
MQT>disable history
MQT>clear history
```

Concentrare ora la propria attenzione sulla traccia di una specifica funzione. Utilizzando **SHOW COMPONENT** e **SHOW FUNCTION** è possibile identificare la particolare area che si desidera tracciare. In questo esempio verrà tracciata la funzione dei servizi comuni 'xcsRequestMutexSem'. Il componente è 0x17 e il codice della funzione è 0x1b. È possibile impostare ciò in uno dei due modi seguenti:

## Esempio di sessione di traccia

```
MQT>select comp=0x17 func=0x1b
```

oppure

```
MQT>select fname="xcsRequestMutexSem"
```

Se adesso si abilita la cronologia, si verifica l'assenza di qualsiasi output nel buffer. Ciò è dovuto al fatto che occorre azzerare completamente i bit di maschera della traccia.

```
MQT>enable history  
MQT>show history
```

```
*** Trace History Ch1:1 ***
```

```
*** End of buffer ***
```

```
*** Trace History Ch:2 ***
```

```
*** End of buffer ***
```

```
MQT>set mask=0xffffffff  
MQT>show history
```



```
*** Trace History Ch1:1 ***
```

```
0973 - 00:00:00.00 01 - 02 --->| xcsRequestMutexSem
0974 - 00:00:00.00 01 - 03 ---->| x11SemReq
0975 - 00:00:00.00 01 - 04 ---->| vms_mtx
0976 - 00:00:00.00 01 - 04 .....| vms_mtx :- Locking BKM3_m_1_45 - timeout: -1
0977 - 00:00:00.00 01 - 05 ---->| vms_get_lock
0978 - 00:00:00.00 01 - 05 ----<| vms_get_lock
0979 - 00:00:00.00 01 - 04 ----<| vms_mtx
0980 - 00:00:00.00 01 - 03 ----<| x11SemReq
0981 - 00:00:00.00 01 - 02 ---<| xcsRequestMutexSem
0982 - 00:00:00.00 01 - 03 ---->| xcsRequestMutexSem
0983 - 00:00:00.00 01 - 04 ---->| x11SemReq
0984 - 00:00:00.00 01 - 05 ---->| vms_mtx
0985 - 00:00:00.00 01 - 05 .....| vms_mtx :- Locking BKM3_m_1_6 - timeout: -1
0986 - 00:00:00.00 01 - 06 ---->| vms_get_lock
0987 - 00:00:00.00 01 - 06 ----<| vms_get_lock
0988 - 00:00:00.00 01 - 05 ----<| vms_mtx
0989 - 00:00:00.00 01 - 04 ----<| x11SemReq
0990 - 00:00:00.00 01 - 03 ----<| xcsRequestMutexSem
0991 - 00:00:00.00 01 - 03 ---->| xcsRequestMutexSem
0992 - 00:00:00.00 01 - 04 ---->| x11SemReq
0993 - 00:00:00.00 01 - 05 ---->| vms_mtx
0994 - 00:00:00.00 01 - 05 .....| vms_mtx :- Locking BKM3/@ipcc_m_1_18 - timeout: -1
0995 - 00:00:00.00 01 - 06 ---->| vms_get_lock
0996 - 00:00:00.00 01 - 06 ----<| vms_get_lock
0997 - 00:00:00.00 01 - 05 ----<| vms_mtx
0998 - 00:00:00.00 01 - 04 ----<| x11SemReq
0999 - 00:00:00.00 01 - 03 ----<| xcsRequestMutexSem
```

```
*** End of buffer ***
```

È possibile, adesso, notare che solo la funzione specificata e le funzioni child sono tracciate per entrambi i processi. Mediante il comando **select** è possibile tracciare fino a otto componenti e funzioni. Per abilitare la traccia in tempo reale (cioè, quando si verifica) occorre creare un processo di client per visualizzare i messaggi per una specifica LU. È possibile fare ciò eseguendo il comando **TRACE**.

Tale comando avvia un processo di client sul nodo specificato e attende i messaggi di traccia in arrivo. È ancora possibile controllare le sessioni di traccia nelle finestre del client tramite **MONMQ**.

```
MQT>trace start node="mihell"
```

A questo punto, abilitare il processo del client e visualizzare i messaggi come e quando arrivano.

```
MQT>enable trace
```

I sottoprocessi di MQSeries possono essere aggiunti o rimossi a piacimento dall'output della traccia. I sottoprocessi possono restare connessi, ma i loro dati di traccia possono essere disabilitati in modo che la traccia non abbia effetti sfavorevoli sulle prestazioni.

È possibile iniziare la traccia all'avvio di un processo o di un sottoprocesso. Il comando **ONSTARTUP** viene utilizzato per questo obiettivo e risulta in tutti i nuovi processi di MQSeries da tracciare all'avvio.

## Esempio di sessione di traccia

Per chiudere una sessione di traccia, occorre disabilitare tutti i canali attivi e terminare il processo di client. Per eseguire ciò, viene utilizzato il comando **close**.

Per lasciare in esecuzione la traccia, utilizzare **quit** da MONMQ e ripristinare in seguito la traccia.

Si noti che i bit di maschera di traccia e la selezione componente/funzione sono molto diversi. I bit di maschera di traccia controllano l'output dei tipi di messaggio di traccia. Ad esempio, le voci e gli output di traccia sono tipi di messaggio. Se vengono disabilitati, qualsiasi cosa si imposti con il comando **select** non avrà alcun effetto poiché la selezione componente/funzione dipende dall'impostazione dei relativi bit di maschera.

---

## Appendice I. Uscite utente

MQSeries per Compaq OpenVMS supporta sia i programmi di uscita del canale che i programmi di uscita di conversione dati. Per informazioni relative alle uscite di canale, consultare il manuale *MQSeries Intercommunication*. Per informazioni relative alle uscite di conversione dati, consultare *MQSeries Application Programming Guide* e il manuale *MQSeries per Tandem Non Stop Kernal, V5.1 Guida operativa*.

questa appendice fornisce informazioni specifiche per l'utilizzo dei programmi di uscita in MQSeries per Compaq OpenVMS.

---

### Uscite del canale e del carico di lavoro

La richiesta di collegamento ad un versione di sottoprocessi separata di un Uscita non è applicabile in MQSeries per Compaq OpenVMS.

---

### MQSeries Cluster Workload Exit

Durante il collegamento ad un workload exit su OpenVMS, nel file di opzione linker devono essere effettuate le seguenti specifiche:

```
sys$share:mqm/share  
sys$share:mqt1/share  
SYMBOL_VECTOR=(c1w1Function=PROCEDURE,MQStart=PROCEDURE)
```

Per riferirsi all'oggetto uscita è necessario un nome logico con un simbolo esecutivo di sistema. Ad esempio, se il nome dell'uscita è `SYSSHARE:AMQSWLM.EXE`, dovrà essere definito il seguente nome logico:  
`$DEFINE/SYSTEM/EXEC AMQSWLM SYSSHARE:AMQSWLM`

L'estensione del file `.EXE` non deve essere specificata nella definizione del nome logico.

Definire questo nome logico durante l'avvio del sistema in `SY$MANAGER:MQS_SYSTARTUP.COM`.



---

## Appendice J. Applicazioni affidabili

Se in un ambiente stabile le prestazioni sono importanti, si possono definire "affidabili" le applicazioni, i canali e i listener che utilizzano il fastpath binding (il tempo impiegato per elaborare le chiamate MQPUT e MQGET dei messaggi non permanenti può essere ridotto del 400% sui sistemi OpenVMS).

In un'applicazione affidabile, l'applicazione MQSeries e l'agente del Queue Manager locale diventano lo stesso processo. L'applicazione si connette direttamente alle risorse del Queue Manager e diventa completamente una estensione di quest'ultimo. Questa opzione può compromettere l'integrità di un Queue Manager, poiché la relativa memoria non è protetta da sovrascrittura.

In aggiunta, per le applicazioni affidabili, può essere necessario creare determinate risorse come la memoria condivisa. Se occorre accedere a tali risorse da un altro processo di gestione code, queste devono appartenere allo stesso UIC. I processi di Queue Manager sono tutti in esecuzione sotto l'account MQM e, pertanto, occorre eseguire le applicazioni affidabili anche sotto questo account.

Considerare quanto sopra illustrato, prima di utilizzare le applicazioni affidabili.

---

### Applicazioni per l'utente

Non è necessario eseguire la propria applicazione direttamente dall'account MQM. Dopo aver effettuato correttamente la connessione al Queue Manager, MQSeries modificherà automaticamente il profilo di sicurezza del sottoprocesso attivo in modo che quest'ultimo assuma l'identità dell'account MQM. Il sottoprocesso recupera l'identità originale in seguito a una chiamata per disconnettersi dal Queue Manager.

È importante notare che, durante la connessione al Queue Manager, l'applicazione sicura viene eseguita in modo efficace sotto l'account MQM. Se è necessario modificare l'identità del sottoprocesso in un altro UIC durante la connessione a un Queue Manager, occorre assicurarsi di modificarlo di nuovo in MQM prima di effettuare la prossima chiamata MQI.

### Impostazione delle applicazioni affidabili

Per eseguire un'applicazione affidabile su MQSeries per OpenVMS occorre specificare che il tipo di binding nel campo Options della chiamata MQCONNX sia MQCNO\_FASTPATH\_BINDING (utilizzare l'opzione MQCNO\_STANDARD\_BINDING per binding standard). Se non è specificata alcuna opzione (MQCNO\_NONE), sarà utilizzato STANDARD\_BINDING per impostazione predefinita.

Inoltre, è possibile utilizzare il nome logico MQ\_CONNECT\_TYPE per ignorare il tipo di binding specificato nella chiamata MQCONNX. Se si definisce il nome logico, questo dovrà essere caratterizzato dal valore FASTPATH o STANDARD per selezionare il tipo di binding richiesto. Tuttavia, si utilizza il FASTPATH binding solo se l'opzione di connessione è specificata in modo appropriato nella chiamata MQCONNX. Qualora dovessero sorgere problemi con FASTPATH\_BINDING, il nome logico rende possibile l'esecuzione di un'applicazione con STANDARD\_BINDING, eliminando la necessità di ricostruirla.

## Impostazione delle applicazioni affidabili

Riassumendo, per eseguire un'applicazione affidabile, occorre:

- Specificare l'opzione MQCNO\_FASTPATH\_BINDING nella chiamata MQCONNX e definire il nome logico MQ\_CONNECT\_TYPE come FASTPATH

oppure

- Specificare l'opzione MQCNO\_FASTPATH\_BINDING nella chiamata MQCONNX e lasciare non definito il nome logico MQ\_CONNECT\_TYPE.

Per ulteriori informazioni sull'utilizzo delle applicazioni affidabili vedere la sezione *MQSeries Intercommunication*.

---

## Esecuzione di canali e listener come applicazioni affidabili

I programmi di canale avviati utilizzando il comando di **avvio canale runmqsc** vengono eseguiti sotto l'account MQM. I programmi di ricezione del canale vengono avviati anche da richieste TCP in ingresso (o connessioni DECnet) eseguite sotto l'account MQM.

I comandi **runmqchl** e **runmqslr** creano un processo separato eseguito sotto l'account MQM. Una combinazione del nome logico MQ\_CONNECT\_TYPE e MQIBindType nella voce dei canali di un file `qm.ini` del Queue Manager definisce se un canale o un listener deve essere eseguito poiché affidabile.

Per impostare un canale o un listener affidabile, occorre:

- Specificare MQIBindType=FASTPATH nel file `qm.ini` e impostare il nome logico come FASTPATH

oppure

- Specificare MQIBindType=FASTPATH nel file `qm.ini` e lasciare indefinito il nome logico.

## Messaggi non permanenti rapidi

È possibile utilizzare l'attributo NPMSPEED del canale (velocità del messaggio non permanente) per specificare la velocità di invio dei messaggi non permanenti. È possibile specificare normale o rapida. Il valore predefinito è rapida, per cui i messaggi non permanenti su un canale non devono attendere il syncpoint prima di poter essere richiamati. Tali messaggi possono essere richiamati molto più velocemente, ma è possibile perderli in caso di errore di trasmissione o di interruzione del canale durante il loro transito. Per ulteriori informazioni sull'esecuzione di canali e listener come applicazioni affidabili e sui messaggi non permanenti rapidi, consultare il manuale *MQSeries Intercommunication*.

---

## Appendice K. Informazioni ausiliarie

Quest'appendice elenca le informazioni ausiliarie necessarie per impostare MQSeries per Compaq OpenVMS.

Le informazioni contenute in quest'appendice saranno inserite nel manuale identificato, nel relativo aggiornamento successivo.

---

### Guida di programmazione dell'applicazione

Le informazioni sulla programmazione su OpenVMS saranno corrette per annotare che non è possibile effettuare le chiamate di Message Queue Interface dall'interno di una routine AST.

Il motivo per ciò è che MQSeries stesso utilizza le routine AST e queste non possono essere eseguite quando un'altra routine AST è attiva.

### Triggering dell'applicazione

Il file di comando MQTRIGGER.COM è fornito come esempio di un file di comando progettato per assumere i parametri forniti dal monitor trigger di MQSeries (RUNMQTRM) e separare i campi nella struttura MQTMC2.

Il file di comando presume che il primo parametro sia l'immagine, o file di comando, da invocare con i campi selezionati dalla struttura MQTMC2.

MQTRIGGER sposta i seguenti campi dalla struttura MQTMC2 all'immagine invocata o al file di comando:

Parametro	Campo MQTMC2
1	QName
2	ProcessName
3	TriggerData
4	ApplType
5	UserData
6	QMgrName

### Esempi

1. Per attivare l'immagine amqsech:

Il campo ApplcId della definizione di processo di trigger è specificato come segue:

```
APPLICID('@mqs_examples:mqtrigger $mqbin:amqsech')
```

Questo esempio presuppone che la directory logica MQBIN sia stata definita come:

```
SYS$SYSROOT:[SYSHLP.EXAMPLES.MQSERIES.BIN]
```

## Informazioni ausiliarie

2. Per invocare un file di comando, dka200:[user]cmd.com:

Il campo ApplicId della definizione di processo di trigger è specificato come segue:

```
APPLICID('@mqs_examples:mqtrigger @dka200:[user]cmd')
```



---

## Appendice L. Informazioni particolari

Queste informazioni sono state sviluppate per i prodotti e servizi offerti negli Stati Uniti. E' possibile che negli altri paesi l'IBM non offra i prodotti, le funzioni o i servizi illustrati in questo documento. Consultare il rappresentante IBM locale per informazioni sui prodotti e sui servizi disponibili nel proprio paese. Ogni riferimento relativo a prodotti, programmi o servizi IBM non implica che solo quei prodotti, programmi o servizi IBM possano essere utilizzati. In sostituzione a quelli forniti dall'IBM, possono essere usati prodotti, programmi o servizi funzionalmente equivalenti che non comportino la violazione dei diritti di proprietà intellettuale o di altri diritti dell'IBM. E' responsabilità dell'utente valutare e verificare la possibilità di utilizzare altri programmi e/o prodotti, fatta eccezione per quelli espressamente indicati dall'IBM.

L'IBM può avere brevetti o domande di brevetto in corso relativi a quanto trattato nella presente pubblicazione. La fornitura di questa pubblicazione non implica la concessione di alcuna licenza su di essi. Chi desiderasse ricevere informazioni relative alle licenze può rivolgersi per iscritto a:

Director of Commercial Relations  
IBM Europe  
Schoenaicher str. 220  
D-7030 Boeblingen  
Deutschland

**Il seguente paragrafo non è valido per il Regno Unito o per tutti i paesi le cui leggi nazionali siano in contrasto con le disposizioni in esso contenute:**

L'INTERNATIONAL BUSINESS MACHINES CORPORATION FORNISCE QUESTA PUBBLICAZIONE "NELLO STATO IN CUI SI TROVA", SENZA ALCUNA GARANZIA, ESPLICITA O IMPLICITA, IVI INCLUSE EVENTUALI GARANZIE DI COMMERCIALIZZABILITA' ED IDONEITA' AD UNO SCOPO PARTICOLARE. Alcuni stati non consentono la rinuncia a garanzie esplicite o implicite in determinate transazioni; quindi la presente dichiarazione potrebbe non essere a voi applicabile.

Questa pubblicazione potrebbe contenere imprecisioni tecniche o errori tipografici. Le informazioni incluse in questo documento vengono modificate su base periodica; tali modifiche verranno incorporate nelle nuove edizioni della pubblicazione. L'IBM si riserva il diritto di apportare miglioramenti e/o modifiche al prodotto o al programma descritto in questa pubblicazione in qualsiasi momento e senza preavviso.

Tutti i riferimenti a siti Web non dell'IBM contenuti in questo documento sono forniti unicamente a scopo di consultazione. I materiali contenuti in tali siti Web non fanno parte di questo prodotto e l'utente si assume ogni rischio relativo al loro utilizzo.

L'IBM può utilizzare o divulgare le informazioni ricevute dagli utenti secondo le modalità ritenute appropriate, senza alcun obbligo nei loro confronti.

Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire (i) uno scambio di informazioni tra programmi indipendenti ed altri (compreso questo) e (ii) l'uso reciproco di tali informazioni, dovrebbero rivolgersi a:

## Informazioni particolari

IBM United Kingdom Laboratories,  
Mail Point 151,  
Hursley Park,  
Winchester,  
Hampshire,  
England  
SO21 2JN.

Queste informazioni possono essere rese disponibili secondo condizioni contrattuali appropriate, compreso, in alcuni casi, il pagamento di un addebito.

Il programma su licenza descritto in queste informazioni e tutto il materiale su licenza ad esso relativo sono forniti dall'IBM nel rispetto delle condizioni previste dalla licenza d'uso.

Le informazioni relative a prodotti non IBM sono state ottenute dai fornitori di tali prodotti. L'IBM non ha verificato tali prodotti e, pertanto, non può garantirne l'accuratezza delle prestazioni. Eventuali commenti relativi alle prestazioni dei prodotti non IBM devono essere indirizzati ai fornitori di tali prodotti.

### LICENZA RELATIVA AI DIRITTI D'AUTORE:

Queste informazioni contengono programmi applicativi di esempio in lingua originale, che illustrano le tecniche di programmazione su diverse piattaforme operative. Potete copiare, modificare o distribuire questi esempi di programmi sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in conformità alle API (Application Programming Interface) a seconda della piattaforma operativa per cui tali esempi di programmi sono stati scritti. Questi esempi non sono stati testati approfonditamente tenendo conto di tutte le condizioni possibili. La IBM, quindi, non può garantire o assicurare l'affidabilità, la praticità o il funzionamento di questi programmi. Potete copiare, modificare e distribuire questi esempi di programmi sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in modo conforme alle API (Application Programming Interface) della IBM.

---

## Marchi

I seguenti termini sono marchi della IBM Corporation negli Stati Uniti, e/o in altri paesi:

AIX	IBM
MQSeries	AS/400
MVS/ESA	NetView
CICS	OS/2
First Failure Support Technology	VSE/ESA
OS/390	BookManager

UNIX è un marchio concesso su licenza esclusivamente tramite la X/Open Company Limited.

DIGITAL, OpenVMS, Compaq, DecNet e Alpha sono marchi della Compaq Corporation.

Intel è un marchio della Intel Corporation negli Stati Uniti e/o in altri paesi.

## **Informazioni particolari**

Microsoft, Windows e il logo Windows sono marchi della Microsoft Corporation.

MultiNet e TCPware sono marchi della Process Software.

Java e tutti i marchi e i logo basati su Java sono marchi della Sun Microsystems, Inc. negli Stati Uniti e/o in altri paesi.

Nomi ai altri servizi, prodotti o società possono essere marchi di altre società.



---

## Bibliografia

Questa sezione descrive la documentazione disponibile per tutti i prodotti correnti MQSeries.

---

### MQSeries pubblicazioni multiplatforma

Molte di queste pubblicazioni, talvolta definite manuali della "famiglia" MQSeries, riguardano tutti i prodotti MQSeries Livello 2. I prodotti MQSeries Livello 2 più recenti sono:

- MQSeries per AIX, V5.2
- MQSeries per AS/400, V5.2
- MQSeries per AT&T GIS UNIX, V2.2
- MQSeries per Compaq OpenVMS Alpha, V5.1
- MQSeries per Compaq Tru64 UNIX, V5.1
- MQSeries per HP-UX, V5.2
- MQSeries per Linux, V5.2
- MQSeries per OS/2 Warp, V5.1
- MQSeries per OS/390, V5.2
- MQSeries per SINIX e DC/OSx, V2.2
- MQSeries per Sun Solaris, V5.2
- MQSeries per Sun Solaris, Intel Platform Edition, V5.1
- MQSeries per Tandem NonStop Kernel, V2.2.0.1
- MQSeries per VSE/ESA, V2.1.1
- MQSeries per Windows, V2.0
- MQSeries per Windows, V2.1
- MQSeries per Windows NT e Windows 2000, V5.2

Le pubblicazioni multiplatforma MQSeries sono:

- *MQSeries Brochure*, G511-1908
- *An Introduction to Messaging and Queuing*, GC33-0805
- *MQSeries Intercommunication*, SC33-1872
- *MQSeries Queue Manager Clusters*, SC34-5349
- *MQSeries - Client*, GC13-2676
- *MQSeries System Administration*, SC33-1873
- *MQSeries - Guida di riferimento per i comandi*, SC13-2823
- *MQSeries Event Monitoring*, SC34-5760
- *MQSeries Programmable System Management*, SC33-1482
- *MQSeries Administration Interface Programming Guide and Reference*, SC34-5390
- *MQSeries Messages*, GC33-1876
- *MQSeries Application Programming Guide*, SC33-0807

- *MQSeries per Tandem Non Stop Kernel, V5.1 Guida operativa*, SC33-1673
- *MQSeries Programming Interfaces Reference Summary*, SX33-6095
- *MQSeries Using C++*, SC33-1877
- *MQSeries Using Java*, SC34-5456
- *MQSeries Application Messaging Interface*, SC34-5604

---

### Pubblicazioni MQSeries per piattaforme specifiche

Ogni prodotto MQSeries è riportato in almeno una delle seguenti pubblicazioni, oltre che nei manuali della famiglia MQSeries.

#### MQSeries per AIX, V5.2

*MQSeries per AIX, V5.0 Guida rapida*, GC13-2677

#### MQSeries per AS/400, V5.2

*MQSeries for AS/400 V5.1 Quick Beginnings*, GC34-5557

*MQSeries for AS/400 V5.1 System Administration*, SC34-5558

*MQSeries for AS/400 V5.1 Application Programming Reference (RPG)*, SC33-5559

#### MQSeries per AT&T GIS UNIX, V2.2

*MQSeries for AT&T GIS UNIX System Management Guide*, SC33-1642

#### MQSeries per Compaq OpenVMS Alpha, V5.1

*MQSeries per Compaq OpenVMS Alpha, V5.1 Guida operativa*, GC13-2967

*MQSeries per Compaq OpenVMS Alpha, V5.1 Guida alla gestione del sistema*, SC13-2966

#### MQSeries per Compaq Tru64 UNIX, V5.1

*MQSeries per Compaq Tru64 UNIX, V5.1 Quick Beginnings*, GC34-5684

#### MQSeries per HP-UX, V5.2

*MQSeries per HP-UX, V5.0 Guida rapida*, GC13-2679

#### MQSeries per Linux, V5.2

*MQSeries per Linux - Guida operativa*, GC13-2862

## Bibliografia

### MQSeries per OS/2 Warp, V5.1

*MQSeries per OS/2 Warp, V5.0 Guida rapida*, GC13-2678

### MQSeries per OS/390, V5.2

*MQSeries for OS/390 Concepts and Planning Guide*, GC34-5650

*MQSeries for OS/390 System Setup Guide*, SC34-5651

*MQSeries for OS/390 System Administration Guide*, SC34-5652

*MQSeries for OS/390 System Administration Guide*, GC34-5892

*MQSeries for OS/390 Messages and Codes*, GC34-5891

*MQSeries for OS/390 Licensed Program Specifications*, GC34-5893

*MQSeries for OS/390 Program Directory*

### MQSeries link for R/3, Version 1.2

*MQSeries link for R/3 Version 1.2 User's Guide*, GC33-1934

### MQSeries per SINIX e DC/OSx, V2.2

*MQSeries for SINIX and DC/OSx System Management Guide*, GC33-1768

### MQSeries per Sun Solaris, V5.2

*MQSeries per Sun Solaris, V5.0 Guida rapida*, GC13-2680

### MQSeries per Sun Solaris, Intel Platform Edition, V5.1

*MQSeries for Sun Solaris, Intel Platform Edition Quick Beginnings*, GC34-5851

### MQSeries per Tandem NonStop Kernel, V2.2.0.1

*MQSeries for Tandem NonStop Kernel System Management Guide*, GC33-1893

### MQSeries per VSE/ESA, V2.1.1

*MQSeries for VSE/ESA Licensed Program Specifications*, GC34-5365

*MQSeries for VSE/ESA System Management Guide*, GC34-5364

### MQSeries per Windows, V2.0

*MQSeries for Windows V2.0 User's Guide*, GC33-1822

### MQSeries per Windows, V2.1

*MQSeries for Windows V2.1 User's Guide*, GC33-1965

### MQSeries per Windows NT e Windows 2000, V5.2

*MQSeries per Windows NT, V5.0 Guida rapida*, GC13-2818

*MQSeries for Windows NT Using the Component Object Model Interface*, SC34-5387

*MQSeries LotusScript Extension*, SC34-5404

---

## Manuali in formato elettronico

Molti manuali MQSeries sono disponibili sia in formato cartaceo che in formato elettronico.

## Formato HTML

Documentazione MQSeries è fornita in formato HTML con questi prodotti MQSeries:

- MQSeries per AIX, V5.2
- MQSeries per AS/400, V5.2
- MQSeries per Compaq OpenVMS Alpha, V5.1
- MQSeries per Compaq Tru64 UNIX, V5.1
- MQSeries per HP-UX, V5.2
- MQSeries per Linux, V5.2
- MQSeries per OS/2 Warp, V5.1
- MQSeries per OS/390, V5.2
- MQSeries per Sun Solaris, V5.2
- MQSeries per Sun Solaris, Intel Platform Edition, V5.1
- MQSeries per Windows NT e Windows 2000, V5.2 (compiled HTML)
- MQSeries link for R/3, V1.2

I manuali MQSeries in formato HTML sono disponibili anche nel sito Web dei prodotti della famiglia MQSeries all'indirizzo:

<http://www.ibm.com/software/mqseries/>

## Portable Document Format (PDF)

I file PDF possono essere letti e stampati utilizzando Adobe Acrobat Reader.

Se non si possiede Adobe Acrobat Reader o si desidera ottenere informazioni aggiornate sulle piattaforme che supportano Acrobat Reader, visitare il sito Web Adobe Systems Inc. all'indirizzo:

<http://www.adobe.com/>

Versioni PDF relative ai manuali MQSeries sono fornite con i seguenti prodotti MQSeries:

- MQSeries per AIX, V5.2
- MQSeries per AS/400, V5.2
- MQSeries per Compaq OpenVMS Alpha, V5.1
- MQSeries per Compaq Tru64 UNIX, V5.1
- MQSeries per HP-UX, V5.2
- MQSeries per Linux, V5.2

- MQSeries per OS/2 Warp, V5.1
- MQSeries per OS/390, V5.2
- MQSeries per Sun Solaris, V5.2
- MQSeries per Sun Solaris, Intel Platform Edition, V5.1
- MQSeries per Windows NT e Windows 2000, V5.2
- MQSeries link for R/3, V1.2

Versioni PDF di tutti gli attuali manuali MQSeries sono inoltre disponibili nel sito Web dei prodotti della famiglia MQSeries all'indirizzo:

<http://www.ibm.com/software/mqseries/>

## Formato BookManager

La libreria MQSeries è disponibile in formato IBM BookManager in una varietà di librerie in linea, tra cui *Transaction Processing and Data collection kit*, SK2T-0730. Per visualizzare questi manuali in formato IBM BookManager, utilizzare i seguenti programmi su licenza IBM:

BookManager READ/2  
 BookManager READ/6000  
 BookManager READ/DOS  
 BookManager READ/MVS  
 BookManager READ/VM  
 BookManager READ for Windows

## Formato PostScript

La libreria MQSeries viene fornita in formato PostScript (.PS) con molti prodotti MQSeries Versione 2. I manuali in formato PostScript possono essere stampati utilizzando una stampante PostScript e possono essere letti utilizzando un apposito visualizzatore.

## Formato Windows Help

Il testo *MQSeries for Windows User's Guide* viene fornito in formato Windows Help con MQSeries per Windows, Versione 2.0 e MQSeries per Windows, Versione 2.1.

## Informazioni su MQSeries disponibili su Internet

Il sito Web della famiglia dei prodotti MQSeries è all'indirizzo:

<http://www.ibm.com/software/mqseries/>

Seguendo i link contenuti in questo sito Web è possibile:

- Ottenere informazioni aggiornate relative alla famiglia di prodotti MQSeries.

- Accedere ai manuali MQSeries in formato HTML e PDF.
- Scaricare MQSeries SupportPac.

---

## Publicazioni correlate

- *Compaq OpenVMS Performance Management*, Gennaio 1999

Questo manuale fornisce informazioni utili per l'ottimizzazione delle prestazioni sui sistemi OpenVMS.

- *Compaq OpenVMS System Management Utilities 2 volumi*, Gennaio 1999

Questi manuali contengono informazioni di riferimento per le utilità di gestione del sistema con OpenVMS.

- *Character Data Representation Library, Character Data Representation Architecture, Reference and Registry*, SC09-2190-00

Il presente documento contiene una panoramica dell'architettura CDRA (Character Data Representation Architecture), e definisce gli elementi dell'architettura sotto forma di manuale di riferimento.

- *DecNet SNA Gateway for Synchronous Transport Installation (OpenVMS)*, Novembre 1993

Questa guida illustra le modalità di installazione e configurazione di DecNet SNA Gateway.

- *Digital SNA APPC/LU6.2 Programming Interface for OpenVMS*, Maggio 1996

Questa guida illustra le modalità di installazione e configurazione di SNA APPC/LU6.2.

- *Digital TCP/IP Services for OpenVMS Installation and Configuration*, Gennaio 1999

Questa guida contiene istruzioni per l'installazione e la configurazione del TCP/IP digitale.

- *Guidelines for OpenVMS Cluster Configurations*, Gennaio 1999

Questa guida descrive le modalità di ottimizzazione della disponibilità e della scalabilità dei cluster di OpenVMS.

- *Introduction to Compaq Networking and Data Communications*, (Numero di codice Compaq 093148)

Questa guida fornisce una panoramica dei concetti, delle attività, dei prodotti e dei manuali relativi ai servizi di rete e alle comunicazioni di dati Compaq.

## Pubblicazioni correlate



---

## Glossario dei termini e delle abbreviazioni

Questo glossario definisce MQSeries i termini e le abbreviazioni utilizzate nel presente manuale. Se non si riesce a trovare il termine desiderato, consultare l'indice oppure la pubblicazione *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

Questo glossario include i termini e le definizioni tratte dalla pubblicazione *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 dell'ANSI (American National Standards Institute). È possibile ordinare delle copie presso American National Standards Institute, 11 West 42 Street, New York, New York 10036. Le definizioni sono identificate dal simbolo (A) dopo la definizione.

### A

**accodamento.** Vedere *accodamento messaggi*.

**accodamento di messaggi.** Una tecnica di programmazione per cui ciascun programma nell'ambito di un'applicazione comunica con gli altri programmi inserendo messaggi nelle code.

**accodamento remoto.** Nell'accodamento di messaggi, la fornitura di servizi che consentono di inserire messaggi nelle code appartenenti ad altri Queue Manager.

**adattatore.** Un'interfaccia fra MQSeries per OS/390 e TSO, IMS, CICS, o spazi di indirizzi batch. Un adattatore è una struttura di collegamento che consente alle applicazioni di accedere ai servizi MQSeries.

**allied address space.** Vedere *ally*.

**ally.** Uno OS/390 spazio di indirizzi che è connesso a MQSeries per OS/390.

**ambiente.** Vedere *ambiente applicativo*.

**ambiente applicativo.** Le funzioni del software che sono accessibili da un programma applicativo. Sulla piattaforma di OS/390, CICS e IMS sono esempi di ambienti applicativi.

**annulla/ripeti record.** Un record di log utilizzato nel ripristino. La ripetizione del record descrive una modifica da apportare un oggetto MQSeries. L'annullamento descrive in che modo eseguire il backout della modifica se il lavoro non è impegnato.

**APAR.** Authorized Program Analysis Report (prospetto analitico del programma autorizzato).

**applicazione del client.** Un'applicazione, eseguita su una stazione di lavoro e collegata a un client, che consente all'applicazione di accedere ai servizi di accodamento su un server.

**applicazione distribuita.** Nell'accodamento messaggi, un insieme di programmi applicativi che è possibile collegare individualmente a un diverso Queue Manager, ma costituiscono collettivamente un'unica applicazione.

**arresto.** Vedere *arresto immediato*, *arresto preventivo* e *arresto a riposo*.

**arresto a riposo.** In MQSeries, un arresto di un Queue Manager che consente la disconnessione di tutte le applicazioni connesse. Confrontare con *arresto immediato* e *arresto preventivo*. Un tipo di arresto dell'adattatore CICS in cui l'adattatore si disconnette da MQSeries, ma solo dopo che tutte le attività correnti sono state completate. Confrontare con *arresto forzato*.

**arresto controllato.** Vedere *arresto a riposo*.

**arresto forzato.** Un tipo di arresto dell'adattatore CICS in cui l'adattatore si disconnette immediatamente da MQSeries per OS/390, indipendentemente dallo stato di qualsiasi applicazione attiva. Confrontare con *arresto a riposo*.

**arresto immediato.** In MQSeries, un arresto di un Queue Manager che non attende la disconnessione delle applicazioni. È consentito il completamento delle chiamate MQI, ma le nuove chiamate MQI non vengono portate a termine dopo che è stato richiesto un arresto immediato. Confrontare con *arresto a riposo* e *arresto preventivo*.

**arresto preventivo.** In MQSeries, un arresto di un Queue Manager che non attende la disconnessione delle applicazioni, né il completamento delle chiamate MQI. Confrontare con *arresto immediato* e *arresto a riposo*.

**ASID.** Address Space Identifier (identificatore spazio di indirizzi).

**attributo.** Una delle proprietà di un insieme di proprietà che definisce le caratteristiche di un oggetto MQSeries.

**avviso.** Un messaggio inviato a un punto cruciale dei servizi di gestione in una rete allo scopo di identificare un problema o un problema imminente.

## B

**backout.** Un'operazione che inverte tutte le modifiche apportate durante la corrente unità di ripristino o unità di lavoro. Al termine dell'operazione, inizia una nuova unità di ripristino o un'unità di lavoro. Confrontare con *commit*.

**backout a fase singola.** Un metodo secondo il quale non è consentito il completamento di un'azione in corso e tutte le modifiche che fanno parte di tale azione devono essere annullate.

**Basic Mapping Support (BMS).** Un'interfaccia tra CICS e i programmi applicativi che formatta i dati di visualizzazione in entrata e in uscita e indirizza i messaggi in uscita di pagina multipla indipendentemente dai caratteri di controllo utilizzati dai vari terminali.

**BMS.** Basic Mapping Support (supporto mapping di base).

**Bootstrap Data Set (BSDS).** Un insieme di dati VSAM che contiene:

- Un inventario di tutti gli insiemi di dati di log attivi e archiviati noti a MQSeries per OS/390
- Un inventario circolare di tutta l'attività recente di MQSeries per OS/390

BSDS è necessario se occorre riavviare il sottosistema di MQSeries per OS/390.

**browse.** Nell'accodamento di messaggi, per utilizzare la chiamata MQGET al fine di copiare un messaggio senza rimuoverlo dalla coda. Vedere anche *get*.

**BSDS.** Bootstrap Data Set (insieme di dati bootstrap).

**buffer-log di output.** In MQSeries per OS/390, un buffer che mantiene record di log di ripristino prima che questi siano scritti sul log di archivio.

**buffer pool (memoria di transito).** Un'area di memorizzazione principale utilizzata per code, messaggi e definizioni di oggetto di MQSeries per OS/390. Vedere anche la sezione *insieme di pagine*.

## C

**canale.** Vedere *canale messaggi*.

**canale messaggi.** Nell'accodamento messaggi distribuiti, un meccanismo per spostare messaggi da un Queue Manager a un altro. Un canale messaggi comprende due agenti del canale messaggi (un mittente a un'estremità e un destinatario dall'altra) e una connessione. Confrontare con *canale MQI*.

**canale mittente.** Nell'accodamento di messaggi, un canale che inizia il trasferimento, rimuove messaggi da

una coda di trasmissione e li trasferisce tramite una connessione a un canale ricevente o richiedente.

**Canale MQI.** Connette un client di MQSeries a un Queue Manager su un sistema di server e trasferisce solo le chiamate MQI e risponde in maniera bidirezionale. Confrontare con *canale messaggi*.

**canale ricevente.** Nell'accodamento di messaggi, un canale che può essere avviato in maniera remota da un canale mittente. Il canale richiedente accetta messaggi dal canale mittente tramite una connessione e inserisce i messaggi sulla coda locale indicata nel messaggio. Vedere anche *canale del server*.

**canale ricevente.** Nell'accodamento messaggi, un canale che risponde a un canale mittente, prende i messaggi da una connessione e li inserisce in una coda locale.

**canale server.** Nell'accodamento di messaggi, un canale che risponde a un canale richiedente, rimuove messaggi da una coda di trasmissione e li trasferisce tramite una connessione al canale richiedente.

**carattere nullo.** Il carattere rappresentato da X'00'.

**CCF.** Channel Control Function (funzione di controllo canale).

**CCSID.** Coded Character Set Identifier (identificatore insieme di caratteri codificati).

**CDF.** Channel Definition File (file di definizione canale).

**Channel Control Function (CCF).** In MQSeries, un programma che sposta messaggi da una coda di trasmissione a una connessione e da quest'ultima a una coda locale, assieme a un'interfaccia del pannello operatore per consentire l'impostazione e il controllo dei canali.

**Channel Definition File (CDF).** In MQSeries, un file contenente definizioni del canale di comunicazione che associano code di trasmissione a connessioni.

**checkpoint.** Un intervallo durante il quale vengono scritte informazioni significative sul log. Confrontare con *syncpoint*. In MQSeries su sistemi UNIX, il punto in cui un record di dati descritto nel log è identico al record di dati nella coda. I checkpoint vengono generati automaticamente e sono utilizzati durante il processo di riavvio del sistema.

**CI.** Control Interval (intervallo di controllo).

**CL.** Control Language (linguaggio di controllo).

**classe di memoria.** In MQSeries per OS/390, una classe di memoria che definisce l'insieme di pagine che deve conservare i messaggi per una particolare coda. La classe di memoria viene specificata quando la coda è definita.

**client.** Un componente di runtime che fornisce accesso ai servizi di accodamento su un server per applicazioni di utenti locali. Le code utilizzate dalle applicazioni risiedono sul server. Vedere anche la sezione *Client di MQSeries*.

**Client MQSeries.** Parte di un prodotto di MQSeries che può essere installato su un sistema senza installare il Queue Manager completo. Il client di MQSeries accetta chiamate MQI dalle applicazioni e comunica con un Queue Manager su un sistema di server.

**cluster.** Una rete di Queue Manager logicamente associati in qualche modo.

**codà.** Un oggetto MQSeries. Le applicazioni di accodamento messaggi possono inserire e richiamare i messaggi da una codà. Una codà è posseduta e gestita da un Queue Manager. Le code locali possono contenere un elenco di messaggi in attesa di elaborazione. Le code di altri tipi non possono contenere messaggi—puntano ad altre code oppure possono essere utilizzate come modelli per code dinamiche.

**codà di applicazione.** Una codà utilizzata da un'applicazione.

**codà di iniziazione.** Una codà locale sulla quale il Queue Manager inserisce messaggi di trigger.

**codà di messaggi.** Sinonimo di *codà*.

**codà dinamica.** Una codà locale creata quando un programma apre un oggetto modello di codà. Vedere anche *codà dinamica permanente* e *codà dinamica temporanea*.

**codà dinamica permanente.** Una codà dinamica che viene eliminata quando è chiusa solo se l'eliminazione è richiesta espressamente. Le code dinamiche permanenti vengono ripristinate se il Queue Manager subisce un malfunzionamento, in modo che possano contenere messaggi permanenti. Confrontare con *codà dinamica temporanea*.

**codà dinamica temporanea.** Una codà dinamica che viene eliminata quando è chiusa. Le code dinamiche temporanee non sono ripristinate se il Queue Manager subisce un malfunzionamento, in modo che possano contenere solo messaggi non permanenti. Confrontare con *codà dinamica permanente*.

**codà di risposta RTQ (reply-to queue).** Il nome di una codà alla quale il programma che ha emesso la chiamata MQPUT intende inviare un messaggio di risposta o di notifica.

**codà di trasmissione.** Una codà locale in cui i messaggi preparati destinati a un Queue Manager remoto sono temporaneamente memorizzati.

**codà evento.** La codà nella quale il Queue Manager inserisce un messaggio di evento dopo che ha rilevato un evento. Ciascuna categoria di evento (Queue Manager, prestazioni o evento canale) dispone della propria codà di eventi.

**codà locale.** Una codà appartenente al Queue Manager locale. Una codà locale può contenere un elenco di messaggi in attesa di elaborazione. Confrontare con *codà remota*.

**codà messaggi non recapitati.** Vedere *Dead-letter Queue*.

**codà remota.** Una codà appartenente a un gestore codà remota. I programmi possono inserire i messaggi su code remote, ma non possono ricevere messaggi da code remote. Confrontare con *codà locale*.

**codà system.command.input.** Una codà locale in cui i programmi applicativi possono inserire comandi di MQSeries. I comandi vengono recuperati nella codà dal server dei comandi, che li convalida e li trasferisce al processore dei comandi affinché li esegua.

**Coded Character Set Identifier (CCSID).** Il nome di un insieme codificato di caratteri e le relative assegnazioni di punti di codice.

**codice d'errore.** Un codice restituito che descrive il motivo del malfunzionamento o la parziale riuscita di una chiamata MQI.

**codice di completamento.** Un codice restituito che indica in che modo è terminata una chiamata MQI.

**codice di erroreabend (fine anomala).** Un codice esadecimale a 4 byte che identifica in maniera univoca un problema con MQSeries per OS/390. Un elenco completo dei codici di errore di fine anomala di MQSeries per OS/390 e delle relative spiegazioni è contenuto nel manuale *MQSeries for OS/390 Messages and Codes*.

**codici di errore.** Il nome collettivo per i codici di completamento e codici di errore.

**comandi amministratore.** Comandi di MQSeries utilizzati per gestire gli oggetti MQSeries, quali code, processi e namelist.

**comandi di controllo del sistema.** I comandi utilizzati per manipolare entità specifiche della piattaforma, quali pool buffer, classi di memoria e insiemi di pagina.

**Comandi MQSeries (MQSC).** Comandi leggibili dagli utenti, uniformi in tutte le piattaforme, che sono utilizzati per manipolare oggetti MQSeries. Confrontare con *Programmable Command Format (PCF)*.

**comando.** In MQSeries, un'istruzione di amministrazione che può essere eseguita dal Queue Manager.

**comando di controllo.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, un comando che può essere immesso interattivamente dalla riga comandi del sistema operativo. Tale comando richiede solo che il prodotto MQSeries sia installato; non richiede un'utilità o un programma particolare per eseguirlo.

**comando PCF.** Vedere *formato comando programmabile*.

**Command Prefix (CPF).** In MQSeries per OS/390, una stringa di carattere che identifica il Queue Manager a cui sono diretti i comandi di MQSeries per OS/390 e da cui sono ricevuti i messaggi per l'operatore di MQSeries per OS/390.

**commit.** Un'operazione che applica tutte le modifiche apportate durante la corrente unità di ripristino o unità di lavoro. Al termine dell'operazione, inizia una nuova unità di ripristino o un'unità di lavoro. Confrontare con *backout*.

**commit a due fasi.** Un protocollo per la coordinazione delle modifiche a risorse recuperabili quando viene utilizzato più di un gestore risorse da una singola transazione. Confrontare con *commit a singola fase*.

**commit a fase singola.** Un metodo secondo il quale un programma può impegnare aggiornamenti in una coda senza coordinarli con gli aggiornamenti effettuati dal programma sulle risorse controllate da un altro gestore di risorse. Confrontare con *il commit a due fasi*.

**commutazione di attività.** La sovrapposizione di operazioni I/O e l'elaborazione tra diverse attività. In MQSeries per OS/390, il commutatore di attività ottimizza le prestazioni consentendo l'esecuzione di alcune chiamate MQI in sottoattività piuttosto che sotto il TCB CICS principale.

**connect.** Fornisce un handle di connessione del Queue Manager, che un'applicazione utilizza su chiamate MQI successive. La connessione viene eseguita sia dalla chiamata MQCONN o automaticamente dalla chiamata MQOPEN.

**connessione rinviata.** Un evento in sospeso che viene attivato quando un sottosistema CICS tenta di effettuare la connessione a MQSeries per OS/390 prima che MQSeries per OS/390 sia stato avviato.

**consegna sequenziale.** In MQSeries, un metodo di trasmissione di messaggi con un numero di sequenza in modo che il canale ricevente possa ristabilire la sequenza di messaggi memorizzando i messaggi. Ciò è necessario quando occorre consegnare solo una volta i messaggi e nell'ordine corretto.

**contesto.** Informazioni sull'origine di un messaggio.

**Control Interval (CI).** Un'area dalla lunghezza fissa della memoria ad accesso diretto in cui VSAM registra record e crea spazi liberi distribuiti. L'intervallo di

controllo è l'unità di informazione che VSAM trasmette alla o dalla memoria ad accesso diretto.

**Control Language (CL).** In MQSeries per AS/400, un linguaggio che è possibile utilizzare per emettere comandi, nella riga comandi o scrivendo un programma di CL.

**CPF.** Command Prefix.

**cursore browse.** Nell'accodamento di messaggi, un indicatore utilizzato nello sfogliare una coda allo scopo di identificare il successivo messaggio in sequenza.

## D

**DAE.** Analisi ed eliminazione dump.

**DAE (Dump analysis and Elimination).** Un servizio di OS/390 che consente a un'installazione di sopprimere i dump SVC e ABEND SYSUDUMP non necessari, poiché duplicano i dump precedentemente scritti.

**Data Conversion Interface (DCI).** L'interfaccia di MQSeries a cui devono conformarsi i programmi scritti dal cliente (o dal rivenditore) che convertono i dati applicativi tra le codifiche di macchine diverse e i CCSID. Fa parte di MQSeries Framework.

**datagramma.** Il messaggio più semplice supportato da MQSeries. Questo tipo di messaggio non richiede una risposta.

**dati evento.** In un messaggio di evento, la parte di dati del messaggio che contiene informazioni sull'evento (come il nome del Queue Manager e l'applicazione che ha dato origine all'evento). Vedere anche *intestazione evento*.

**DCE.** Distributed Computing Environment.

**DCI.** Interfaccia di conversione di dati.

**Dead-Letter Queue (DLQ).** Una coda alla quale il Queue Manager o l'applicazione invia messaggi che non può consegnare alla rispettiva destinazione corretta.

**definizione locale.** Un oggetto di MQSeries appartenente a un Queue Manager locale.

**definizione locale di una coda remota.** Un oggetto di MQSeries appartenente a un Queue Manager locale. Quest'oggetto definisce gli attributi di una coda che appartenente a un altro Queue Manager. Inoltre, viene utilizzato per l'aliasing del manager di code e per l'aliasing RTQ.

**descrittore messaggio.** Informazioni di controllo che descrivono il formato del messaggio e la presentazione che è eseguita come parte di un messaggio di



MQSeries. Il formato del descrittore messaggio è definito dalla struttura di MQMD.

**descrittore oggetto.** Una struttura di dati che identifica un particolare oggetto di MQSeries. Nel descrittore sono inclusi il nome dell'oggetto e il tipo di oggetto.

**Distributed Computing Environment (DCE).** Middleware che fornisce alcuni servizi fondamentali, semplificando lo sviluppo di applicazioni distribuite. DCE è definito da Open Software Foundation (OSF).

**Distributed Queue Management (DQM).** Nell'accodamento di messaggi, l'impostazione e il controllo dei canali di messaggi a Queue Manager su altri sistemi.

**DLQ.** Coda di messaggi non recapitati.

**DQM.** Gestione di code distribuite.

## E

**Editor del registro di sistema.** In Windows NT, il programma che consente all'utente di modificare il Registro di sistema.

**ESM.** External Security Manager (gestore protezione esterna).

**ESM (External Security Manager).** Un prodotto di protezione invocato da System Authorization Facility di OS/390. RACF è un esempio di un ESM.

**ESTAE.** Extended Specify Task Abnormal Exit.

**ESTAE (Extended Specify Task Abnormal Exit).** Una macro di OS/390 che fornisce capacità di ripristino e controllo alla routine di uscita specificata per l'elaborazione, il rilevamento diabend o la specificazione di un indirizzo per ulteriori tentativi.

**evento.** Vedere *evento canale*, *evento strumentazione*, *evento prestazioni* e *evento Queue Manager*.

**evento canale.** Un evento che indica che un'istanza del canale è divenuta disponibile o non disponibile. Gli eventi di canale sono generati sui Queue Manager su entrambe le estremità del canale.

**evento gestore coda.** Un evento che indica:

- Una condizione di errore che si è verificata in relazione alle risorse utilizzate da un Queue Manager. Ad esempio, una coda non è disponibile.
- Una modifica significativa che si è verificata nel Queue Manager. Ad esempio, un Queue Manager è stato arrestato o avviato.

**evento intervallo di servizio.** Un evento correlato all'intervallo di servizio.

**evento prestazione.** Una categoria di evento indicante che si è verificata una condizione di limite.

**evento sospeso.** Un evento non pianificato che si verifica quale risultato di una richiesta di connessione da un adattatore CICS.

**evento strumentazione.** Una funzione che può essere utilizzata per monitorare il funzionamento dei Queue Manager in una rete di sistemi MQSeries. MQSeries fornisce eventi di strumentazione per il monitoraggio di definizioni di risorse del Queue Manager, condizioni delle prestazioni e condizioni del canale. Gli eventi di strumentazione possono essere utilizzati da un meccanismo di notifica scritta dall'utente in un'applicazione amministrativa che visualizza gli eventi a un operatore di sistema. Essi inoltre consentono alle applicazioni di comportarsi come agenti per altre reti amministrative per monitorare prospetti e creare gli avvisi appropriati.

**evento trigger.** Un evento (quale un messaggio in arrivo in una coda) che consente al Queue Manager di creare un messaggio trigger su una coda di iniziazione.

**Event Viewer.** Uno strumento fornito da Windows NT per esaminare e gestire i file di log.

## F

**FFST.** First Failure Support Technology (tecnologia di assistenza per guasti).

**FFST (First Failure Support Technology).** Utilizzata da MQSeries su sistemi UNIX, MQSeries per OS/2 Warp, MQSeries per Windows NT e Windows 2000 e MQSeries per AS/400 per rilevare e riportare problemi di software.

**FIFO.** First-in-first-out (primo entrato primo uscito).

**file di autorizzazione.** In MQSeries su sistemi UNIX, un file che fornisce definizioni di protezione per un oggetto, una classe di oggetti o tutte le classi di oggetti.

**file di configurazione.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, un file contenente informazioni di configurazione di riferimento, come ad esempio log, comunicazioni o servizi installabili. Sinonimo del file *.ini*. Vedere anche *stanza*.

**file di controllo del log.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, il file contenente le informazioni necessarie a monitorare l'utilizzo dei file di log (ad esempio, la dimensione e la posizione e il nome del successivo file disponibile).

**file di log.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, un file in cui sono registrate tutte le modifiche significative dei dati controllati da un Queue Manager. Se i file di log primari si riempiono, MQSeries alloca file di log secondari.

**file ini.** Vedere *file di configurazione*.

**first-in-first-out (FIFO).** Una tecnica di accodamento in cui l'elemento successivo da recuperare è quello che è rimasto in coda più a lungo. (A)

**Framework.** In MQSeries, una raccolta di interfacce di programmazione che consentono ai clienti o ai rivenditori di scrivere programmi che estendono o sostituiscono alcune funzioni fornite in MQSeries. Le interfacce sono:

- Data Conversion Interface (DCI, interfaccia di conversione dati) di MQSeries
- Message Channel Interface (MCI, interfaccia del canale messaggio) di MQSeries
- Name Service Interface (NSI, interfaccia del servizio nome) di MQSeries
- Security Enabling Interface (SEI, interfaccia di abilitazione protezione) di MQSeries
- Trigger Monitor Interface (TMI, interfaccia di monitoraggio trigger) di MQSeries

**FRR.** Functional Recovery Routine (routine di ripristino funzionale).

**Functional Recovery Routine (FRR).** Una funzione di gestione ripristino/arresto di OS/390 che consente a una routine di ripristino di ottenere il controllo nel caso di un'interruzione di un programma.

## G

**GCPC.** Generalized Command Preprocessor (pre-processor di comandi generalizzati).

**Generalized Command Preprocessor (GCPC).** Un componente di MQSeries per OS/390 che elabora i comandi MQSeries e li esegue.

**Generalized Trace Facility (GTF).** Un programma di servizio OS/390 che registra eventi di sistema significativi, quali le chiamate del supervisore e le operazioni di avvio operazioni di input e di output allo scopo di determinare i problemi.

**gestore coda locale.** Per un programma, un gestore coda che non è l'unico a cui il programma è connesso.

**gestore di risorse.** Un'applicazione, un programma o una transazione che gestisce e controlla l'accesso alle risorse condivise quali buffer di memoria e insiemi di dati. MQSeries, CICS e IMS sono gestori di risorse.

**get.** Nell'accodamento di messaggi, per utilizzare la chiamata MQGET allo scopo di rimuovere un messaggio da una coda. Vedere anche *browse*.

**GTF.** Generalized Trace Facility (funzione traccia generalizzata).

## H

**handle.** Vedere *handle di connessione* e *handle dell'oggetto*.

**handle dell'oggetto.** L'identificatore o token con il quale un programma accede all'oggetto MQSeries con il quale sta lavorando.

**handle di connessione.** L'identificativo o token tramite il quale un programma accede al Queue Manager a cui è connesso.

**handler di coda messaggi non recapitati.** Un'utilità fornita da MQSeries che controlla una coda di messaggi non recapitati (DLQ) ed elabora messaggi in coda in conformità alla tabella delle regole scritte dall'utente.

**Hive del registro di sistema.** In Windows NT, la struttura dei dati memorizzata nel registro di sistema.

## I

**identificatore di spazio indirizzo (ASID).** Un identificatore univoco assegnato dal sistema a uno spazio di indirizzo.

**identificatore transazione.** In CICS, un nome che viene specificato quando la transazione è definita e che viene utilizzato per invocare la transazione.

**ID sessione.** In MQSeries per OS/390, l'ID univoco di CICS che definisce la connessione da utilizzare da un agente di canale messaggi durante il trasferimento da una coda di trasmissione a un collegamento.

**immagine del media.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, la sequenza dei record di log che contengono un'immagine di un oggetto. È possibile creare nuovamente l'oggetto da questa immagine.

**insieme di dati di input di inizializzazione.** Insieme di dati utilizzati da MQSeries per OS/390 al suo avvio.

**insieme di pagina.** Un insieme di dati VSAM utilizzato quando MQSeries per OS/390 trasferisce i dati (ad esempio, code e messaggi) dai buffer nella memoria principale alla memoria permanente (DASD).

**Interactive Problem Control System (IPCS).** Un componente di OS/390 che permette la gestione di problemi in linea, la diagnosi interattiva di problemi, il debug in linea per dump abend residenti sul disco, l'analisi dei problemi e il report dei problemi.

**Interactive System Productivity Facility (ISPF).** Un programma concesso in licenza da IBM che funge da editor a tutto schermo e gestore delle finestre di dialogo. È utilizzato per la scrittura di programmi applicativi e fornisce uno strumento di generazione di

pannelli di schermo e finestre di dialogo interattive tra il programmatore dell'applicazione e l'utente finale.

**interruzione della verifica macchina.** Un'interruzione che si verifica come risultato di un malfunzionamento o un errore della macchina. Un'interruzione della verifica macchina può essere recuperabile via hardware, software o non recuperabile.

**intervallo di servizio.** Un intervallo di tempo, rispetto al quale il tempo trascorso tra put e get e un successivo get è confrontato dal Queue Manager nello stabilire se le condizioni per un evento di intervallo di servizio sono state soddisfatte. L'intervallo di servizio per una coda è specificato da un attributo di coda.

**intestazione evento.** In un messaggio di evento, la parte dei dati di messaggio che identifica il tipo di evento del codice di errore relativo a tale evento.

**IPCS.** Interactive Problem Control System (sistema di controllo interattivo dei problemi).

**ISPF.** Interactive System Productivity Facility (funzione interattiva di produttività del sistema).

## L

**listener.** Nell'accodamento distribuito di MQSeries, un programma che controlla le connessioni di rete in arrivo.

**locale.** Sui sistemi UNIX, un sottoinsieme dell'ambiente dell'utente che definisce convenzioni per una cultura specifica (quali il tempo, la formattazione numerica o monetaria e la classificazione, il confronto o la conversione di caratteri). Il CCSID del Queue Manager deriva dalla locale dell'ID utente che ha creato il Queue Manager.

**log.** In MQSeries, un file che registra il lavoro svolto dai Queue Manager mentre ricevono, trasmettono e consegnano messaggi, per consentirne il ripristino in caso di malfunzionamento.

**log attivo.** Vedere *log di ripristino*.

**log dell'applicazione.** In Windows NT, un log che registra gli eventi significativi dell'applicazione.

**log di archivio.** Vedere *log di ripristino*.

**log di evento.** Vedere *log dell'applicazione*.

**log di ripristino.** In MQSeries per OS/390, insieme di dati contenenti informazioni necessarie per ripristinare messaggi, code e il sottosistema di MQSeries. MQSeries per OS/390 scrive ciascun record in un insieme di dati denominato *log attivo*. Quando il log attivo è pieno, il suo contenuto viene ripartito in un DASD o in un insieme di dati su nastro, denominato *log archivio*. Sinonimo di *log*.

**Logical Unit of Work (LUW).** Vedere *unità di lavoro*.

## M

**MCA.** Message Channel Agent (agente del canale messaggi).

**MCA (Message Channel Agent).** Un programma che trasmette messaggi da una coda di trasmissione a una connessione oppure da una connessione a una coda di destinazione. Vedere anche *interfaccia di accodamento messaggi*.

**MCI.** Message Channel Agent (interfaccia del canale messaggi).

**memorizzazione e invio.** La memorizzazione temporanea di pacchetti, messaggi o frame in una rete di dati prima che siano ritrasmessi verso la loro destinazione.

**Message Channel Interface (MCI).** L'interfaccia di MQSeries a cui devono conformarsi i programmi scritti dal cliente (o dal rivenditore) che trasmettono messaggi tra un Queue Manager di MQSeries e un altro sistema di messaggistica. Fa parte di MQSeries Framework.

**messaggio.** Nelle applicazioni di accodamento messaggi, una comunicazione inviata tra programmi. Vedere anche *messaggio permanente* e *messaggio non permanente*. Nella programmazione del sistema, le informazioni dirette all'operatore del terminale o all'amministratore di sistema.

**messaggio di evento.** Contiene informazioni (quali la categoria dell'evento, il nome dell'applicazione che ha provocato l'evento e le statistiche del Queue Manager) in relazione all'origine di un evento di strumentazione in una rete di sistemi MQSeries.

**messaggio di notifica.** Un tipo di messaggio che fornisce informazioni circa un altro messaggio. Un messaggio di notifica può indicare che un messaggio è stato consegnato, è giunto a destinazione, è scaduto o non è stato elaborato per qualche motivo. Confrontare con *messaggio di risposta* e *messaggio di richiesta*.

**messaggio di richiesta.** Un tipo di messaggio utilizzato per richiedere una risposta da un altro programma. Confrontare con *messaggio di risposta* e *messaggio di notifica*.

**messaggio di risposta.** Un tipo di messaggio utilizzato per le risposte allo scopo di richiedere messaggi. Confrontare con *messaggio di richiesta* e *messaggio di notifica*.

**messaggio non permanente.** Un messaggio che non sopravvive al riavvio del Queue Manager. Confrontare con *messaggio permanente*.

**messaggio permanente.** Un messaggio che sopravvive al riavvio del Queue Manager. Confrontare con *messaggio non permanente*.

**messaggio protetto.** Un messaggio che è scritto nella memoria (disco) ausiliaria in modo da non essere perduto in caso di un malfunzionamento del sistema. Vedere anche *messaggio permanente*.

**messaggio trigger.** Un messaggio contenente informazioni sul programma che un monitor trigger deve avviare.

**messaggistica.** Vedere *messaggistica sincrona* e *messaggistica asincrona*.

**messaggistica asincrona.** Un metodo di comunicazione tra programmi in cui questi inseriscono messaggi in code di messaggi. Con la messaggistica asincrona, il programma di invio procede con la propria elaborazione senza attendere una risposta al messaggio. Confrontare con *messaggistica sincrona*.

**messaggistica indipendente dal tempo.** Vedere *messaggistica asincrona*.

**messaggistica sincrona.** Un metodo di comunicazione tra programmi in cui questi inseriscono messaggi in code di messaggi. Con la messaggistica sincrona, il programma mittente attende una risposta al proprio messaggio prima di riprendere l'elaborazione. Confrontare con *messaggistica asincrona*.

**modalità doppia.** Vedere *registrazione nei log doppia*.

**monitor avviso.** In MQSeries per OS/390, un componente dell'adattatore CICS che gestisce eventi non pianificati che si verificano come risultato di richieste di connessione a MQSeries per OS/390.

**monitor trigger.** Un'applicazione continuamente in esecuzione che serve una o più code di iniziazione. Quando un messaggio trigger giunge su una coda di iniziazione, il monitor trigger recupera il messaggio. Esso utilizza le informazioni nel messaggio trigger per avviare un processo che serve la coda in cui si è verificato un evento trigger.

**MQAI.** MQSeries Administration Interface (interfaccia di amministrazione di MQSeries).

**MQI.** Message Queue Interface (interfaccia di coda messaggi).

**MQI (Message Queue Interface).** L'interfaccia di programmazione fornita dai Queue Manager di MQSeries. Questa interfaccia di programmazione consente ai programmi applicativi di accedere ai servizi di accodamento dei messaggi.

**MQSC.** Comandi di MQSeries.

**MQSeries.** Una famiglia di programmi concessi in licenza da IBM che offre servizi di accodamento messaggi.

**MQSeries Administration Interface (MQAI).** Un'interfaccia di programmazione di MQSeries.

## N

**namelist.** Un oggetto di MQSeries che contiene un elenco di nomi, ad esempio, nomi di code.

**Name Service Interface (NSI).** L'interfaccia di MQSeries a cui devono conformarsi i programmi scritti dal cliente (o dal rivenditore) che risolvono la proprietà del nome della coda. Fa parte di MQSeries Framework.

**New Technology File System (NTFS).** Un file system recuperabile di NT che fornisce protezione per i file.

**notifica di arresto.** Un evento sospeso che viene attivato quando un sottosistema CICS si connette correttamente a MQSeries per OS/390.

**NSI.** Name Service Interface.

**NTFS.** New Technology File System (file system di nuova tecnologia).

**numerazione sequenza messaggi.** Una tecnica di programmazione che assegna numeri univoci ai messaggi durante la trasmissione attraverso una connessione. Ciò consente al processo di ricezione di verificare se tutti i messaggi sono ricevuti, inserirli in una coda secondo l'ordine originale ed eliminare i messaggi duplicati.

## O

**OAM.** Object Authority Manager (gestore di autorizzazione dell'oggetto).

**Object Authority Manager (OAM).** In MQSeries su sistemi UNIX, MQSeries per AS/400 e MQSeries per Windows NT e Windows 2000, il servizio di autorizzazione predefinito per la gestione del comando e dell'oggetto. OAM può essere sostituito da, o eseguito in combinazione con un servizio di protezione fornito dall'utente.

**oggetto.** In MQSeries, un oggetto è un Queue Manager, una coda, una definizione di un processo, un canale, un namelist, o una classe di memoria (solo OS/390).

**oggetto coda dell'alias.** Un oggetto MQSeries, il cui nome è un alias per una coda di base definita al gestore di coda locale. Quando un'applicazione o un gestore di coda utilizzano un alias di coda, il nome alias viene risolto e l'operazione richiesta viene eseguita sulla coda di base associata.



**oggetto coda modello.** Un insieme di attributi di coda che agisce da modello quando un programma crea una coda dinamica.

**oggetto coda remota.** Vedere *definizione locale di una coda remota*.

**oggetto di definizione del processo.** Un oggetto MQSeries che contiene la definizione di un'applicazione MQSeries. Ad esempio, un Queue Manager utilizza la definizione quando opera con messaggi trigger.

**oggetto predefinito.** Una definizione di un oggetto (ad esempio, una coda) con tutti gli attributi definiti. Se un utente definisce un oggetto, ma non specifica tutti i possibili attributi per tale oggetto, il Queue Manager utilizza attributi predefiniti in luogo di quelli che non sono stati specificati.

## P

**parametro di input.** Un parametro di una chiamata MQI in cui si forniscono informazioni durante la chiamata.

**parametro di output.** Un parametro di una chiamata MQI in cui il Queue Manager restituisce le informazioni quando la chiamata è completata o non riuscita.

**parametro input/output.** Un parametro di una chiamata MQI in cui si forniscono informazioni durante la chiamata e in cui il Queue Manager modifica le informazioni quando la chiamata è completata o non riuscita.

**PCF.** Programmable Command Format (formato comando programmabile).

**percolazione.** Nel ripristino degli errori, il proseguimento lungo un percorso di controllo prestabilito da una routine di ripristino a un'altra routine di ripristino di livello superiore.

**percorso di risoluzione.** L'insieme di code che sono aperte quando un'applicazione specifica un alias o una coda remota su immissione in una chiamata MQOPEN.

**piattaforma.** In MQSeries, il sistema operativo sotto il quale il Queue Manager è in esecuzione.

**ping.** Nell'accodamento distribuito, un ausilio diagnostico che utilizza lo scambio di un messaggio di prova per confermare che un canale di messaggi o una connessione TCP/IP siano funzionanti.

**principal.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, un termine utilizzato per un identificatore utente. Utilizzato dal gestore di autorizzazione oggetti per verificare le autorizzazioni di accesso alle risorse del sistema.

**priorità del messaggio.** In MQSeries, un attributo di un messaggio che può influenzare l'ordine di recupero dei messaggi di una coda e l'eventuale generazione di un evento trigger.

**processore comandi.** Il componente di MQSeries che elabora i comandi.

**profilo switch.** In MQSeries per OS/390, un profilo RACF utilizzato quando MQSeries viene avviato o quando viene emesso un comando di protezione aggiornato. Ciascun profilo switch rilevato da MQSeries termina la verifica della risorsa specificata.

**Programmable Command Format (PCF).** Un tipo di messaggio di MQSeries utilizzato da:

- Applicazioni amministrative dell'utente, per inserire comandi PCF nella coda di immissione comandi del sistema di un Queue Manager specifico.
- Applicazioni amministrative dell'utente, per ottenere i risultati di un comando PCF da un gestore di code specifico
- Un Queue Manager, come notifica del verificarsi di un evento.

Confrontare con *MQSC*.

**programma di trasmissione.** Vedere *agente di canale messaggi*.

**Program Temporary Fix (PTF).** Una soluzione o by-pass di un problema diagnosticato dagli ingegneri IBM come risultato di un difetto in un rilascio corrente e non modificato di un programma.

**prospetto analitico del programma autorizzato (APAR).** Un prospetto di un problema provocato da un presunto difetto in una versione corrente e non modificata di un programma.

**protezione del contesto.** In MQSeries, un metodo per consentire la gestione della protezione in modo che i messaggi siano obbligati a recare dettagli delle relative origini nella descrizione del messaggio.

**protezione utente alternativo.** Una funzione di protezione grazie alla quale l'autorizzazione di un ID utente può essere utilizzata da un altro ID utente; ad esempio, per aprire un oggetto MQSeries.

**PTF.** Program Temporary Fix (riparazione temporanea del programma).

**punto di ripristino.** In MQSeries per OS/390, il termine utilizzato per descrivere un insieme di copie di riserva degli insiemi di pagina di MQSeries per OS/390 e i corrispondenti insiemi di dati del log richiesti per ripristinare tali insiemi di pagina. Tali copie di riserva forniscono un punto di riavvio potenziale nel caso di perdita dell'insieme di pagine (ad esempio, un errore di I/O dell'insieme di pagina).

## Q

**Queue Manager.** Un programma di sistema che fornisce i servizi di accodamento alle applicazioni. Fornisce un'API (la MQI) che consente ai programmi di accedere ai messaggi nelle code appartenenti al Queue Manager. Vedere anche *gestore coda locale* e *gestore coda remota*. Un oggetto MQSeries che definisce gli attributi di un particolare Queue Manager.

**Queue Manager locale.** Il Queue Manager a cui è connesso un programma, che fornisce servizi di accodamento dei messaggi al programma. I Queue Manager a cui non è connesso un programma sono denominati *Queue Manager remoto*, anche se sono in esecuzioni sullo stesso sistema del programma.

## R

**RBA.** Relative Byte Address (indirizzo byte relativo).

**Recovery Termination Manager (RTM).** Un programma che gestisce tutte le attività di arresto normale o anomalo trasferendo il controllo a una routine di ripristino associato alla funzione arrestata.

**registrazione nei log circolare.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, il processo che conserva tutti i dati di riavvio in un ring di file di log. La registrazione di log riempie il primo file nel ring e quindi passa al successivo, finché tutti i file sono pieni. A questo punto, la registrazione torna al primo file nel ring e ricomincia, se lo spazio è stato liberato o non è più necessario. La registrazione circolare è utilizzata durante il ripristino di riavvio, utilizzando il log per eseguire il roll back delle transazioni che erano in corso quando il sistema è stato arrestato. Confrontare con *registrazione nei log lineare*.

**registrazione nei log doppia.** Un metodo di registrazione dell'attività di MQSeries per OS/390, in cui ciascuna modifica è registrata in due insiemi di dati, in modo che se è necessario riavviare e un insieme di dati è illeggibile, sia possibile utilizzare l'altro. Confrontare con *registrazione nei log singola*.

**registrazione nei log lineare.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, il processo che conserva i dati di riavvio in un sequenza di file. I nuovi file vengono aggiunti alla sequenza secondo le necessità. Lo spazio in cui i dati vengono scritti non viene riutilizzato finché il Queue Manager non è riavviato. Confrontare con *registrazione nei log circolare*.

**registrazione nei log singola.** Un metodo di registrazione dell'attività di MQSeries per OS/390 in cui ciascuna modifica viene registrata su un unico insieme di dati. Confrontare con *registrazione nei log doppia*.

**Registro di sistema.** In Windows NT, un database protetto che fornisce un'unica fonte per dati di sistema e della configurazione dell'applicazione.

**Relative Byte Address (RBA).** Il trasferimento in byte di un record memorizzato o un intervallo di controllo dall'inizio dello spazio di memoria allocato all'insieme di dati a cui appartiene.

**RESLEVEL.** In MQSeries per OS/390, un'opzione che controlla il numero di ID utente di CICS verificati per la protezione di risorse API in MQSeries per OS/390.

**Resource Recovery Services (RRS).** Una funzione di OS/390 che fornisce supporto syncpoint a 2 fasi tramite gestori di risorse partecipanti.

**responder.** Nell'accodamento distribuito, un programma che risponde alle richieste di connessione alla rete da un altro sistema.

**resynch.** In MQSeries, un'opzione che consente di indirizzare il canale ad avviare e risolvere qualsiasi messaggio di stato in sospenso, ma senza riavviare il trasferimento di messaggi.

**richiamata.** In MQSeries, un canale di messaggi di richiesta inizia un trasferimento da un canale di invio chiamando prima il mittente, quindi chiudendo e restando in attesa di una richiamata.

**ripartizione.** In MQSeries per OS/390, un processo automatico durante il quale il log attivo di un Queue Manager viene trasferito al relativo log di archivio.

**riposo.** In MQSeries, lo stato di un Queue Manager precedente al suo arresto. In tale stato, i programmi i programmi possono terminare l'elaborazione, ma non è possibile avviare nuovi programmi.

**risorsa.** Qualsiasi funzione del sistema di calcolo o sistema operativo richiesto da un lavoro o un'attività. In MQSeries per OS/390, esempi di risorse sono pool buffer, insiemi di pagine, insiemi di pagine del log, code e messaggi.

**rollback.** Sinonimo di *back out*.

**RRS.** Resource Recovery Services (servizi di ripristino risorse).

**RTM.** Recovery Termination Manager (gestore fine ripristino).

## S

**SAF.** System Authorization Facility (sistema di autorizzazione del sistema).

**SDWA.** System Diagnostic Work Area (area di lavoro diagnostica del sistema).

**Security Enabling Interface (SEI).** L'interfaccia di MQSeries a cui devono conformarsi i programmi scritti dal cliente (o dal rivenditore) che verificano le autorizzazioni, forniscono ID o eseguono l'autenticazione. Fa parte di MQSeries Framework.

**segnalazione.** In MQSeries per OS/390 e MQSeries per Windows 2.1, una funzione che consente al sistema operativo di inviare una notifica a un programma quando un messaggio atteso giunge nella coda.

**SEI.** Security Enabling Interface (interfaccia di attivazione protezione).

**server.** (1) In MQSeries, un Queue Manager che fornisce servizi di code alle applicazioni client in esecuzione su una stazione di lavoro remota. (2) Il programma che risponde alle richieste di informazioni nel particolare modello di client/server di flusso di informazioni a due programmi. Vedere anche *client*.

**server di comandi.** Il componente di MQSeries che legge i comandi dalla coda d'immissione dei comandi di sistema, li verifica e passa i comandi validi al processore comandi.

**servizi installabili.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, le funzionalità aggiuntive fornite come componenti indipendenti. L'installazione di ciascun componente è facoltativa: è possibile utilizzare componenti IBM o di terzi. Vedere anche *servizio di autorizzazione*, *servizio nome* e *servizio identificatore utente*.

**servizio di autorizzazione.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, un servizio che fornisce la verifica dell'autorizzazione dei comandi e chiamate MQI per l'identificatore dell'utente associato al comando o alla chiamata.

**servizio nome.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, la funzione che determina a quale Queue Manager appartiene una coda specifica.

**SIT.** System Initialization Table (tabella di inizializzazione del sistema).

**sottosistema.** In OS/390, un gruppo di moduli che fornisce la funzione che dipende da OS/390. Ad esempio, MQSeries per OS/390 è un sottosistema di OS/390.

**spazio indirizzo.** L'area della memorizzazione virtuale disponibile per un particolare lavoro.

**stringa sintomo.** Informazioni diagnostiche visualizzate in un formato strutturato progettato per la ricerca del database di supporto del software IBM.

**Supervisor Call (SVC).** Un'istruzione di OS/390 che interrompe un programma in esecuzione e trasferisce il

controllo al supervisore in modo che possa eseguire il servizio specifico indicato dall'istruzione.

**SVC.** Chiamata supervisore.

**syncpoint.** Un punto intermedio o finale durante l'elaborazione di una transazione al quale le risorse protette sono congruenti. A un syncpoint, le modifiche delle risorse possono essere impegnate con sicurezza, oppure è possibile eseguire il backout al precedente syncpoint.

**SYS1.LOGREC.** Un ausilio di servizio contenente informazioni sugli errori di programma e di hardware.

**System Authorization Facility (SAF).** Una funzione di OS/390 tramite la quale MQSeries per OS/390 comunica con un gestore di protezione esterno quale RACF.

**System Diagnostic Work Area (SDWA).** Dati registrati in una voce SYS1.LOGREC, che descrive un errore di programma o di hardware.

**System Initialization Table (SIT).** Una tabella contenente parametri utilizzati da CICS all'avvio.

## T

**tabella regole.** Un file di controllo contenente una o più regole che l'handler della coda messaggi non recapitati applica alla DLQ.

**TACL.** Tandem Advanced Command Language.

**TCB.** Task Control Block (blocco di controllo delle attività).

**TCB (Task Control Block).** Un blocco di controllo di OS/390 utilizzato per comunicare informazioni sulle attività nell'ambito di un spazio indirizzi che sono connesse a un sottosistema di OS/390 quali MQSeries per OS/390 o CICS.

**thlqual.** Target library high-level qualifier (qualificatore di alto livello della libreria di destinazione).

**thlqual (target library high-level qualifier).** Qualificatore di alto livello per i nomi degli insiemi di dati di destinazione di OS/390.

**thread.** In MQSeries, il livello inferiore dell'esecuzione parallela disponibile su una piattaforma di sistema operativo.

**tipo canale di connessione del server.** Il tipo di definizione canale MQI associata al server che esegue un Queue Manager. Vedere anche *tipo di canale di connessione del client*.

**tipo di canale di connessione del client.** Il tipo di definizione del canale MQI associato con un client MQSeries. Vedere anche la sezione *tipo di canale di connessione del server*.

**TMI.** Trigger Monitor Interface (interfaccia monitor del trigger).

**traccia.** In MQSeries, una funzione per la registrazione dell'attività di MQSeries. Le destinazioni per le voci di traccia possono includere GTF e la funzione di gestione del sistema (SMF). Vedere anche *traccia globale* e *traccia prestazioni*.

**traccia di prestazione.** Un'opzione di traccia di MQSeries in cui i dati di traccia devono essere utilizzati per l'analisi e l'ottimizzazione delle prestazioni.

**traccia globale.** Un'opzione di traccia di MQSeries per OS/390 in cui i dati della traccia provengono dall'intero sottosistema di MQSeries per OS/390.

**tranid.** Vedere *identificatore transazione*.

**trasformazione del nome.** In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, un processo interno che modifica il nome di un Queue Manager in modo da renderlo univoco e valido per il sistema utilizzato. Esternamente, il nome del Queue Manager rimane invariato.

**trasmissione ciclica del numero sequenziale.** In MQSeries, un metodo che garantisce che entrambe le estremità di una connessione reimpostino contemporaneamente i numeri di sequenza dei messaggi correnti. La trasmissione di messaggi con un numero di sequenza garantisce che il canale ricevente possa ristabilire la sequenza di messaggio quando memorizza i messaggi.

**triggering.** In MQSeries, una funzione che consente a un Queue Manager di avviare un'applicazione automaticamente quando sono soddisfatte condizioni predefinite su una coda.

**Trigger Monitor Interface (TMI).** L'interfaccia di MQSeries a cui devono conformarsi i programmi di monitor trigger scritti dal cliente (o dal rivenditore). Fa parte di MQSeries Framework.

## U

**UIS.** User Identifier Service (servizio identificazione utente).

**unità di lavoro.** Una sequenza recuperabile di operazioni eseguite da un'applicazione tra due punti di consistenza. Un'unità di lavoro inizia all'avvio di una transazione o al termine di un syncpoint richiesto

dall'utente. Essa termina sia al syncpoint richiesto dall'utente o al termine di una transazione. Confrontare con *unità di ripristino*.

**unità di ripristino.** Una sequenza recuperabile di operazioni nell'ambito di un singolo gestore di risorse. Confrontare con *unità di lavoro*.

**unità di ripristino in sospeso.** In MQSeries, lo stato di un'unità di ripristino per cui è stato richiesto un syncpoint che non è stato ancora confermato.

**User Identifier Service (UIS).** In MQSeries per OS/2 Warp, la funzione che consente alle applicazioni MQI per associare un ID utente, diverso dall'ID utente predefinito, con messaggi di MQSeries.

**utilità.** In MQSeries, un insieme di programmi incluso che fornisce all'operatore di sistema o all'amministratore di sistema funzioni aggiuntive oltre a quelle fornite dai comandi di MQSeries. Alcune utilità invocano più di una funzione.

**verifiche di autorizzazione.** Verifiche di protezione che vengono eseguite quando un utente prova a emettere comandi amministrativi rispetto a un oggetto, ad esempio per aprire una coda o per connettersi a un Queue Manager.

**voce.** Un gruppo di righe in un file di configurazione che assegna un valore a un parametro che modifica il comportamento di un Queue Manager, di un client o di un canale. In MQSeries su sistemi UNIX, MQSeries per OS/2 Warp e MQSeries per Windows NT e Windows 2000, un file di configurazione (.ini) che può contenere un numero di voci.

# Indice analitico

## Caratteri speciali

- , requisiti
- hardware 325
- software 325

## A

- abilitazione
  - eventi 124
  - sicurezza 89
- accesso non autorizzato, protezione da 85
- accodamento di messaggi 5
- accodamento distribuito
  - code messaggi non recapitati 12
  - code undelivered-message 12
  - output non corretto 209
- accodamento remoto 69
- algoritmi di richiamo per i messaggi 7
- alias
  - code di risposta RTQ 81
  - Queue Manager 80
- alias di coda di risposta RTQ 81
- alias di code 56
  - descrizione 11
- ambiente operativo specificato 325
- amqsdli, il modello di DLQ handler 108
- annullamento di una coda locale 50
- applicazione
  - affidabile 387
  - ambiente client-server 15
  - connessione al Queue Manager locale 77
  - considerazioni sulla progettazione 205
  - dati 6
  - indipendente dal tempo 5
  - supporto gestione MQI 37
- Applicazione
  - errori di programmazione, esempi di 202
- applicazione affidabile 387
  - binding 387
- applicazioni indipendenti dal tempo 5
- arresto traccia MQSeries 286
- attributi
  - attributo ALL 49
  - code 9
  - confronto tra MQSC e PCF 21
  - modifica 42, 50
  - predefiniti 49
  - Queue Manager
    - modifica 42
    - visualizzazione 40
  - visualizzazione Queue Manager 40
- Attributo AdoptNewMCA 191
- attributo LIKE 49
- attributo ListenerBacklog 194
- attributo MaxMsgLength 8

- attributo REPLACE, comandi DEFINE 43
- autorizzazione
  - comandi 91
  - comando dspmqaut 92
  - comando set/reset 310
  - comando setmqaut 92
  - contestuale 94
  - elenchi 89
  - gestione 99
  - identificativi diritti 88
  - MQI 96
  - servizi installabili 91
  - utente alternativo 93
- autorizzazione contestuale 94
- autorizzazione per utente alternativo 93
- autorizzazioni diritti di elaborazione 87
- autorizzazioni gruppo primario 87
- autorizzazioni MQOPEN 96
- autorizzazioni MQPUT 96
- avvio
  - canali 74
  - Queue Manager in un gruppo di ripristino 235
  - un Queue Manager 32

## B

- bilanciamento del carico di lavoro
  - reindirizzamento di chiamata MQI 56
  - utilizzo di cluster 7
- binary
  - aprire un file binario di traccia 364
  - chiusura di file binario di traccia 365
- binding
  - per applicazioni affidabili 387
- BookManager 397

## C

- canale
  - accodamento remoto 69
  - affidabile 388
  - autorizzazioni di comando
    - escape 100
  - avvio 74
  - comandi 95
  - comando run initiator 298
  - connettere il sottoprocesso di destinazione a 362
  - definizione 73
  - definizione automatica di 75
  - definizione di canale mittente 69
  - definizione di canale ricevente 69
  - definizione tra Queue Manager 11
  - descrizione 13, 69
  - disconnessione sottoprocesso di destinazione da 363
  - eventi 123
  - fastpath 192

- canale (*Continua*)
  - gestione remota 72
  - mostra cronologia dei messaggi 356
  - requisiti di sicurezza del comando 95
  - show mask (MONMQ) 355
  - sicurezza 95
  - visualizzare i dettagli di canale (MONMQ) 355
  - voce Channels, qm.ini 191
- canale di connessione al server, definizione automatica di 75
- canale fastpath 192
- canale ricevente, definizione automatica di 75
- capacità attuale della coda 49
- capacità della coda
  - attuale 49
  - determinazione 49
- CCSID 349
  - conversione dei dati 81
  - riavvio di Queue Manager 83
  - supportato da MQSeries per Compaq OpenVMS 326
- ccsid.tbl 81
- channel
  - comando run 299
- chiusura
  - Queue Manager 33
    - ad attività completate 33
    - controllata 33, 34
    - immediata 33
    - preventiva 34
  - chiusura ad attività completate, Queue Manager 33
  - chiusura controllata 33
  - chiusura di un Queue Manager 34
  - chiusura preventiva del Queue Manager 34
- client 15
  - applicazioni di collegamento 15
  - comando avvia trigger monitor 308
  - creazione di canali per 15
  - determinazione dei problemi 220
  - messaggi di errore per DOS e Windows 221
- cluster
  - accodamento remoto 69
  - attributi voce ExitProperties 183
  - descrizione di 70
  - di Queue Manager 7
  - OpenVMS
    - differenza rispetto al cluster di Queue Manager 227
    - gruppo di ripristino 228
    - installazione di MQSeries 227
  - Queue Manager
    - coda di trasmissione 12
    - descrizione 14
    - differenza dal cluster OpenVMS 14
    - utilizzo di namelist 14



- cluster (*Continua*)
    - workload exit 17, 385
  - cluster alias service 241
  - cluster di Queue Manager
    - Consultare anche cluster 14
  - cluster OpenVMS 227
  - cluster workload exit 385
  - coda comandi 12
  - coda di destinazione
    - definizione 78
    - messaggi non recapitati alla 107
    - nome del Queue Manager
      - proprietario 78
  - coda di evento 13
  - coda di iniziazione
    - definizione 60
    - descrizione 11
  - coda di trasmissione
    - cluster 12
    - creazione 79
    - definizione 73
    - definizione tra Queue Manager 11
    - descrizione 12, 70
    - gestione remota 72
    - predefinita 12, 31
    - predefinite 80
    - specifica del nome 79
  - coda di trasmissione predefinita 80
  - coda messaggi non recapitati 107
  - codici di errore
    - comando rsvmqtrn 296
    - comando runmqfm 302
    - comando runmqm 308
    - comando setmqaut 314
    - comando strmqcsv 317
    - comando strmqm 318
  - code
    - alias 11
    - alias, utilizzo di 56
    - applicazione, definizione per triggering 59
    - attributi 9
      - modifica 50
    - autorizzazioni a 93
    - comandi 12
    - condivise operazioni di configurazione 177
    - condivise relative a diversi Queue Manager 177
    - definizione 9
    - descrizione 6
    - di risposta 13, 80
    - dinamiche 7
    - distribuito, output non corretto da 209
    - esame 52
    - esempio
      - definizione 58
      - utilizzo di 57
    - evento 13
    - iniziazione
      - definizione 60
      - messaggi trigger 11
    - locale 10
      - annullamento 50
      - copia 49
      - definizione 47
  - code (*Continua*)
    - locale 10 (*Continua*)
      - eliminazione 52
    - locali 9
      - predefinite 14
    - malfunzionamento 204
    - messaggi non recapitati 12
      - specifica 30
    - modello 11
    - notificazione di evento 124
    - oggetti
      - alias 11
      - locale 10
      - modello 11
      - remoti 11
    - per applicazioni MQSeries 37
    - predefinite 7
    - remote 9, 11
      - alias Queue Manager 80
      - creazione 77
      - funzionante con 80
    - temporanee 7
    - trasmissione 12
      - creazione 79
      - definizione 73
      - gestione remota 72
      - predefinita 31
      - predefinite 80
    - trasmissione cluster 12
    - undelivered-message
      - specifica 30
      - utilizzo di 47
  - code condivise
    - operazioni di configurazione 177
  - code di alias
    - autorizzazioni a 93
  - code di esempio
    - definizione 58
    - utilizzo di 57
  - code di risposta 13
  - code di trasmissione del cluster
    - descrizione 12
  - code dinamiche 7
    - autorizzazioni a 93
    - descrizione 7
  - code locali
    - annullamento 50
    - comandi 12
    - copia di definizioni 49
    - definizione 47
    - descrizione 9, 10
    - eliminazione 52
    - iniziazione 11
    - messaggi non recapitati 12
    - trasmissione 12
  - code permanenti 7
  - code predefinite 7
  - code remote
    - autorizzazioni a 93
    - descrizione 9, 11
  - code temporanee 7
  - coded character set
    - identifier 349
  - codice 349
  - codice di errore 9, 107, 112
  - codici di errore
    - comando crtmqcvx 256
- codici di errore (*Continua*)
  - comando crtmqm 260
  - comando dltnqm 263
  - comando dspmqaut 270
  - comando dspmqcsv 272
  - comando dspmqfls 274
  - comando dspmqtrn 277
  - comando endmqcsv 279
  - comando endmqslr 282
  - comando endmqm 284
  - comando endmqtrc 286
  - comando failover 288
  - comando rcdmqimg 292
  - comando rcrmqobj 294
  - comando runmqchi 298
  - comando runmqchl 299
  - comando runmqslr 303
  - comando runmqsc 306
  - comando runmqtrm 309
  - interpretazione dei valori di 255
- codici di errori 200
- codici restituiti
  - comando strmqtrc 321
- comandi
  - avvia monitor di ripristino (runmqfm) 302
  - avvia server di comandi (strmqcsv) 317
  - avvia trigger monitor (runmqtrm) 309
  - avvia trigger monitor del client (runmqtrmc) 308
  - avvio Queue Manager (strmqm) 318
  - avvio traccia di MQSeries (strmqtrc) 320
  - comandi di sicurezza
    - dspmqaut 92
    - setmqaut 89
  - comandi MQSeries (MQSC) 20
  - comando dump log (dmpmqlog) 266
  - comando end listener (endmqslr) 282
  - comando run listener (runmqslr) 303
  - confronto tra gruppi 335
  - controllo 19
  - conversione dei dati (crtmqcvx) 256
  - crea Queue Manager (crtmqm) 258
  - delete Queue Manager (dltnqm) 263
  - display authority (dspmqaut) 268
  - display command server (dspmqcsv) 272
  - display MQSeries files (dspmqfls) 273
  - display MQSeries transactions (dspmqtrn) 277
  - end command server (endmqcsv) 279
  - end MQSeries trace (endmqtrc) 286
  - end queue manager (endmqm) 283
  - esegue comandi di MQSeries (runmqsc) 305
  - file di comandi MQSC
    - input 43
    - notifiche di output 43
  - Gestione di ripristino (failover) 287
  - guida per la sintassi 253
  - imposta/reimposta l'autorizzazione (setmqaut) 310

- comandi (*Continua*)
  - impostare/reimpostare
    - l'autorizzazione (setmqaut) 91
  - MQSC
    - ALTER QLOCAL 50
    - ALTER QREMOTE 79
    - DEFINE CHANNEL 73
    - DEFINE QALIAS 56
    - DEFINE QLOCAL 49
    - DEFINE QLOCAL LIKE 50
    - DEFINE QLOCAL REPLACE 50
    - DEFINE QMODEL 58
    - DEFINE QREMOTE 77
    - DELETE QLOCAL 52
    - DELETE QREMOTE 79
    - DISPLAY QREMOTE 79
  - MQSeries (MQSC)
    - utilizzo 20
    - verifica 44
  - PCF (Programmable Command Format) 20
  - registra immagine supporto (rcdmqimg) 291
  - ricrea oggetto (rcrmqobj) 293
  - risolvere transazione di MQSeries (rsvmqtrn) 296
  - run channel (runmqchl) 299
  - run channel initiator (runmqchi) 298
  - run dead-letter queue handler 300
  - run DLQ handler (runmqdlq) 107
  - runmqsc 39
  - visualizza traccia formattata di MQSeries (dspmqtrc) 275
- comandi dell'operatore, nessuna risposta da 203
- comandi di controllo 19
  - runmqsc 39
  - sensibilità al
    - minuscolo/maiuscolo 23
    - utilizzo 27
- comandi MQSC
  - ALTER QLOCAL 50
  - ALTER QREMOTE 79
  - DEFINE CHANNEL 73
  - DEFINE QALIAS 56
  - DEFINE QLOCAL 49
  - DEFINE QLOCAL LIKE 50
  - DEFINE QLOCAL REPLACE 50
  - DEFINE QMODEL 58
  - DEFINE QREMOTE 77
  - DELETE QLOCAL 52
  - DELETE QREMOTE 79
  - DISPLAY QREMOTE 79
  - immissione interattiva 39
  - lunghezza massima della linea 43
  - sensibilità al
    - minuscolo/maiuscolo 25
    - utilizzo 20
- comandi PCF
  - attributi in MQSC e PCF 64
  - automazione delle attività di gestione utilizzando PCF 63
  - escape PCF 64
  - MQAI, utilizzo per semplificare l'impiego di 64
- comandi per MQSeries 19
- comando analyse trace (MONMQ) 368
- comando close binary (MONMQ) 365
- comando close LU (MONMQ) 355
- comando close text (MONMQ) 365
- comando connect (MONMQ) 362
- comando crtmqcvx
  - codici di errore 256
  - esempi 256
  - parametri 256
- comando crtmqmq 258
  - codici di errore 260
  - comandi correlati 262
  - esempi 260
  - parametri 258
- comando default variable (MONMQ) 353
- comando delete history (MONMQ) 366
- comando deselect index (MONMQ) 364
- comando di avvio Queue Manager 318
- comando di avvio traccia di MQSeries 320
- comando disable history (MONMQ) 366
- comando disable timestamp (MONMQ) 365
- comando disable tracing (MONMQ) 366
- comando disconnect (MONMQ) 363
- comando dlmqmq 263
  - codici di errore 263
  - comandi correlati 265
  - esempi 263
  - parametri 263
- comando dspmqaut 268
  - codici di errore 270
  - comandi correlati 271
  - esempi 271
  - parametri 268
  - utilizzo 89, 91
- comando dspmqcsv 272
  - codici di errore 272
  - comandi correlati 272
  - esempi 272
  - parametri 272
- comando dspmqfls 273
  - codici di errore 274
  - esempi 274
  - parametri 273
- comando dspmqtrc 275
  - comandi correlati 276
  - esempi 275
  - parametri 275
- comando dspmqtrn 277
  - codici di errore 277
  - comandi correlati 278
  - parametri 277
- comando enable history (MONMQ) 366
- comando enable timestamp (MONMQ) 365
- comando enable tracing (MONMQ) 365
- comando endmqcsv 279
  - codici di errore 279
  - comandi correlati 281
  - esempi 279
  - parametri 279
- comando endmqslr (end listener)
  - codici di errore 282
  - formato 282
  - funzione 282
  - parameters 282
- comando endmqmq 33, 283
  - codici di errore 284
  - comandi correlati 285
  - esempi 284
  - parametri 283
- comando endmqtrc 286
  - codici di errore 286
  - comandi correlati 286
  - esempi 286
  - parametri 286
- comando exit (MONMQ) 371
- comando failover 229, 233, 287
  - codici di errore 288
  - comandi correlati 290
  - esempi 288
  - parametri 287
- comando FFST (MONMQ) 371
- comando onstartup start (MONMQ) 362
- comando onstartup stop (MONMQ) 362
- comando open (MONMQ) 354
- comando open binary (MONMQ) 364
- comando open text (MONMQ) 365
- comando quit (MONMQ) 371
- comando rcdmqimg 291
  - codici di errore 292
  - comandi correlati 292
  - esempi 292
  - parametri 291
- comando rcrmqobj 293
  - codici di errore 294
  - comandi correlati 295
  - esempi 294
  - parametri 293
- comando rsvmqtrn 296
  - codici di errore 296
  - comandi correlati 297
  - parametri 296
- comando runmqchi 298
  - codici di errore 298
  - parametri 298
- comando runmqchl 299
  - codici di errore 299
  - parametri 299
- comando runmqdlq 107
- comando runmqfm 229, 302
  - codici di errore 302
  - comandi correlati 302
  - esempi 302
  - parameters 302
  - parametri 287
- comando runmqslr (run listener)
  - codici di errore 303
  - esempio 304
  - formato 303
  - funzione 303
  - parametri 303
- comando runmqsc 305
  - codici di errore 306
  - esempi 306
  - parametri 306
  - reindirizzamento di input e output 39
- comando runmqtrc 308
  - codici di errore 308
  - parametri 308
- comando runmqtrm 309
  - codici di errore 309
  - parametri 309

- comando select (MONMQ) 363
  - comando set color (MONMQ) 368
  - comando set depth (MONMQ) 366
  - comando set free (MONMQ) 366
  - comando set mask (MONMQ) 367
  - comando set output (MONMQ) 368
  - comando setmqaut 310
    - codici di errore 314
    - comandi correlati 316
    - esempi 315
    - parameters 312
    - servizi installabili 91
    - utilizzo 89, 91
  - comando show channels (MONMQ) 355
  - comando show components (MONMQ) 361
  - comando show events (MONMQ) 359
  - comando show functions (MONMQ) 361
  - comando show globals (MONMQ) 357
  - comando show history (MONMQ) 356
  - comando show mask (MONMQ) 355
  - comando show memory (MONMQ) 360
  - comando show mutex (MONMQ) 358
  - comando show processes (MONMQ) 356
  - comando show segment (MONMQ) 354
  - comando show stack (MONMQ) 356
  - comando strmqcsv 317
    - codici di errore 317
    - comandi correlati 317
    - esempi 317
    - parametri 317
  - comando strmqm 318
    - codici di errore 318
    - comandi correlati 319
    - esempi 318
    - parametri 318
  - comando strmqtrc 320
    - codici restituiti 321
    - comandi correlati 322
    - esempi 321
    - parametri 320
  - comando trace start (MONMQ) 363
  - comando trace stop (MONMQ) 363
  - command server
    - end comando 279
  - componente
    - show functions (MONMQ) 361
  - componente installabile
    - authority manager (OAM) 86
    - Name Service 177
  - condivisione di code 177
  - configurazione
    - AllQueueManagers voce, mqs.ini 181
    - database, qm.ini 190
    - esempio di file qm.ini 195
    - failover.ini 232
    - file di configurazione Queue Manager, qm.ini 181
    - gestori di database 129
    - gruppo di ripristino cluster OpenVMS 232
    - implementazione delle modifiche 180
    - log 188
    - modifica 179
    - mqs.ini, descrizione di 180
  - configurazione (*Continua*)
    - Oracle 131
    - priorità 180
    - Voce Channels, qm.ini 191
    - voce ClientExitPath, mqs.ini 182
    - Voce DefaultQueueManager, mqs.ini 183
    - voce Exitpath, qm.ini 194
    - Voce ExitProperties, mqs.ini 183
    - voce Log, qm.ini 188
    - voce LogDefaults, mqs.ini 184
    - voce LU62, qm.ini 193
    - voce QueueManager, mqs.ini 186
    - voce Service, qm.ini 186
    - voce ServiceComponent, qm.ini 187
    - voce TCP, qm.ini 193
    - voce XAResourceManager, qm.ini 190
  - considerazioni sulle prestazioni
    - CorrelId 206
    - durante l'utilizzo di traccia 214
    - lunghezza del messaggio 206
    - messaggi di lunghezza variabile 206
    - MsgId 206
    - permanenza dei messaggi 206
    - syncpoint 206
    - vantaggi di MQPUT1 207
  - Controllo dei Queue Manager 122
  - convenzioni di denominazione, supporto lingua nazionale 251
  - conversione dei dati 81
    - Attributo ConvEBCDICNewline, voce AllQueueManagers 181
    - comando crtmqcvx 256
    - conversione del carattere EBCDIC NL in ASCII 181
    - conversione di dati predefinita 82
    - conversione di formati di messaggio definiti dall'utente 83
  - conversione del carattere EBCDIC NL in ASCII 181
  - conversione di dati predefinita 82
  - CorrelId, considerazioni sulle prestazioni durante l'utilizzo 206
  - Costanti MQZAO e autorizzazioni 96
  - creazione
    - code 9
    - comando crtmqm 258
    - definizioni di processo 60
    - oggetti 8
    - Queue Manager 28, 31
  - cronologia
    - dei messaggi nel canale 356
- D**
- Database relazionali XA compatibili 128
  - DCE
    - condivisione di code 177
    - configurazione 178
    - insieme 177
    - sicurezza 18
  - dead-letter header, MQDLH 107
  - definizione automatica dei canali 75
  - definizione di canale mittente 69
  - definizione di canale ricevente 69
  - definizione dinamica dei canali 75
- E**
- definizioni di processo
    - creazione 60
    - descrizione 13
    - visualizzazione 61
  - determinazione dei problemi 199
    - client 220
    - controlli principali 199, 202
    - controlli ulteriori 203
    - errori di programmazione 202
    - file di configurazione 214
    - nessuna risposta dai comandi 203
    - output non corretto
      - con accodamento distribuito 209
      - messaggi assenti nelle code 207
      - messaggi che contengono informazioni inattese 208
    - tracce 214
  - diagrammi railroad, come leggere 252
  - Digital TCP/IP Services per OpenVMS 234
  - dimensione del messaggio, diminuzione 50
  - directory
    - autorizzazione 102
    - Queue Manager 93
  - disabilitazione di OAM (object authority manager) 89
  - disabilitazione eventi 124
  - DLQ (dead-letter queue)
    - descrizione 12
    - handler, consultare anche DLQ handler 107
    - run dead-letter queue handler (runmqdlq) 300
    - specifica 30
  - DLQ handler
    - modello, amqsdq 108
    - richiamo 107
    - tabella regole 108
  - dump
    - copia formattata del log di sistema (comando dmpmqlog) 266
    - scaricamento dei record dei log (comando dmpmqlog) 158
    - scaricamento del contenuto di un log di ripristino 158
  - dump log
    - formato 266
    - funzione 266
    - parametri 266



- esecuzione del debug (*Continua*)
  - errori di programmazione frequenti 202
- esempi
  - comando crtmqcvx 256
  - comando crtmqm 260
  - comando dlrmqm 263
  - comando dspmqaut 271
  - comando dspmqcsv 272
  - comando dspmqfls 274
  - comando dspmqtrc 275
  - comando endmqcsv 279
  - comando endmqm 284
  - comando endmqtrc 286
  - comando failover 288
  - comando rcdmqimg 292
  - comando rcrmqobj 294
  - comando runmqfm 302
  - comando runmqslr 304
  - comando runmqsc 306
  - comando setmqaut 315
  - comando strmqcsv 317
  - comando strmqm 318
  - comando strmqtrc 321
  - creazione di una coda di trasmissione 79
  - errori di programmazione 202
  - file mqs.ini, MQSeries per Compaq OpenVMS 194
  - file qm.ini 195
  - log degli errori 211
- esempio
  - dati traccia 215
  - file MQSC 339
  - programmi, utilizzo 339
- eventi
  - canale 123
  - code 124
  - show events (MONMQ) 359
  - strumentazione
    - abilitazione e disabilitazione 124
    - cosa sono 121
    - descrizione 121
    - messaggio 125
    - perché usarli 122
    - tipi di 123
  - tipi di 123
  - trigger 124
- eventi prestazioni 123
- evento di strumentazione
  - abilitazione 124
  - descrizione 121
  - messaggi 125
  - perché usarli 122
  - tipi di 123

## F

- failback
  - descrizione 229
- failover
  - descrizione 229
- failover monitor
  - arresto 239
  - avvio 235
  - descrizione 229
  - watcher 229

- failover monitor supervisore
  - arresto 239
- failover.template 242
- fastpath binding 387
- feedback da comandi MQSC 40
- FFST
  - FFST (MONMQ) 371
- FFST, esame 215
- file
  - analisi dei nomi 21
  - autorizzazione
    - autorizzazioni per 105
    - classe 104
    - comprendere 102
    - contenuto 103
    - gestione 105
    - percorsi 102
    - tutte le classi 104
  - configurazione
    - nella determinazione dei problemi 214
  - configurazione Queue Manager 181
  - controllo dei log 144
  - MQSeries configurazione 180
- file di autorizzazione
  - autorizzazione per 105
  - classe 104
  - comprendere 102
  - contenuto 103
  - directory 102
  - gestione 105
  - percorsi 102
  - tutte le classi 104
- file di comandi 43
- file di configurazione
  - failover.template 341
  - gruppo di ripristino cluster OpenVMS 232
  - modifica 179
  - MQSeries (mqs.ini)
    - AllQueueManagers voce 181
    - contenuto 180
    - percorso 45
    - Voce ClientExitPath 182
    - voce DefaultQueueManager 183
    - voce ExitProperties 183
    - Voce LogDefaults 184
    - Voce QueueManager 186
- queue manager (qm.ini)
  - contenuto 195
  - voce Channels 191
  - Voce ExitPath 194
  - Voce Log 188
  - Voce Service Component 187
  - Voce XARsourceManager 190
  - Voci LU62 e TCP 193
- Queue Manager (qm.ini)
  - contenuto 181
  - disabilitazione di OAM (object authority manager) 89
  - Voce Service 186
- file di configurazione failover.ini 242
  - modifica 232
- file di configurazione qm.ini
  - definizione di 181
  - LU62 voce 193
  - modifica 179

- file di configurazione qm.ini (*Continua*)
  - priorità 180
  - voce Channels 191
  - voce Exitpath 194
  - voce Log 188
  - voce Service 186
  - voce ServiceComponent 187
  - voce TCP 193
  - voce XARsourceManager 190
- file di log 210
- file MODPARAMS.DAT 223
- formati messaggio definiti dall'utente 83
- formato dei log 144
- Formato PostScript 397
- funzioni
  - visualizzazione nel componente 361

## G

- gestione
  - autorizzazioni 99
  - gruppi di comandi 19
    - comandi di controllo 19
    - comandi MQSeries (MQSC) 20
    - Comandi PCF (Programmable Command Format) 20
  - locali 37
  - remota 71
    - canali 72
    - code di trasmissione 72
  - remoti
    - oggetti 69
- gestione accesso 87
- gestione degli oggetti per 59
- gestione della memoria condivisa 372
- gestione di file di log 151
- Gestione gruppi di ripristino 287
- gestione locale 37
- gestione remota
  - di oggetti 69
  - problemi iniziali 77
  - server di comando 65
- gestori di database
  - comando dspmqtrn, controllo delle unità di lavoro in sospenso 137
  - comando rsvmqtrn, esplicita risincronizzazione di unità di elaborazione 139
  - configurazione 129
  - connessioni al 129
  - coordinazione 128
  - definizione dei gestori di database in qm.ini 130
  - istanze del gestore di database, rimozione 141
  - limitazioni, supporto di coordinazione di database 129
  - modifica delle informazioni di configurazione 140
  - switch load file, creazione 130
  - unità di elaborazione sospese 137
- glossario 399
- gruppi di ripristino
  - descrizione 228
  - gestione 235
- gruppo comandi
  - confronto 335

- gruppo comandi (*Continua*)
  - gestione 19
- gruppo di ripristino
  - arresto di un Queue Manager in 236
  - avvio di un Queue Manager in 235
  - configurazione 230
  - esempio di configurazione 242
  - fasi precedenti la configurazione 230
  - file di configurazione 229
  - modifica dello stato 239
  - procedure di comando 233
  - risoluzione dei problemi 241
  - spostamento di un Queue Manager in 236
  - utilizzo di ICC (Intra Cluster Communication) 240
  - utilizzo di MultiNet per OpenVMS 241
  - visualizzazione stato 236
- gruppo di ripristino cluster OpenVMS 230
- guida per la sintassi 253

## H

- history
  - abilita cronologia (MONMQ) 366
  - azzerata per il canale (MONMQ) 366
  - disabilita cronologia (MONMQ) 366
  - elimina cronologia (MONMQ) 366
  - imposta cronologia (MONMQ) 366
- HTML (Hypertext Markup Language) 396
- Hypertext Markup Language (HTML) 396

## I

- ICC (Intra Cluster Communication) 240
- ID esadecimali
  - visualizzazione per componenti 361
- ID utente
  - appartenente al gruppo nobody 88
  - autorizzazione 85, 92
  - OpenVMS utente connesso 92
  - per l'autorizzazione 92
  - principal 87
- identificativi di diritti, per l'autorizzazione 87
- identificativi di diritti primario, per l'autorizzazione 87
- identificativo diritti
  - MQM 85
  - predefinito, nobody 88
  - predefinito per l'autorizzazione 88
- identificativo diritti di OpenVMS
  - MQM 85
- immagini del supporto
  - descrizione 153
  - record 154
  - ripristino 154
- immagini supporto
  - comando record 291
- immissione di comandi MQSeries 38
- informazioni ausiliarie 389
- insieme, DCE e code 177

- insiemi di caratteri codificati 81

## L

- listener
  - affidabile 388
  - comando arrestalister (endmqlsr) 282
  - utilizzo del comando run listener (runmqlsr) 303
- locale 349
- log
  - configurazione 188
  - errore 209
  - errore, esempio di 211
  - file
    - controllo 144
    - posizione 152
    - riutilizzo 146
  - formato 144
  - gestione 150
  - output dal comando dmpmqlog 159
  - parametri 31
  - queue manager 143
  - scaricamento dei record dei log (comando dmpmqlog) 158
  - scaricamento del contenuto di 158
  - utilizzo per il ripristino 152
  - voce Log, qm.ini 188
- log degli errori
  - esempio 211
- log di errore
  - errori verificatisi prima dell'installazione 210
  - panoramica di 209
- lunghezza massima della linea di comandi MQSC 43

## M

- macro
  - MONMQ 373
- manuali in formato elettronico 396
- mask
  - imposta maschera (MONMQ) 367
- massima
  - capacità di coda 9
  - dimensione del messaggio 6, 8
  - dimensione di una coda 6
  - lunghezza del nome per oggetti MQSeries 8
  - numero di messaggi 9
- massimo
  - numero di messaggi 7
- MAXMSG 8
- MCA (Message Channel Agent)
  - Attributo AdoptNewMCA 191
  - canale in stato RETRY 194
  - descrizione 70
- memoria globale
  - controllo di utilizzo dei parametri sysgen 223
- messaggi di errore 40
- Messaggi di errore per client DOS 221
- Messaggi di errore per client Windows 221
- messaggi per l'operatore 211
- messaggio
  - accodamento 5
  - algoritmi di richiamo 7
  - assenti nelle code 207
  - che contiene informazioni inattese 208
  - considerazioni sulle prestazioni
    - lunghezza di 206
    - permanente 206
  - descrittore 6, 63
  - descrizione 6
  - dimensione massima di 6
  - errori relativi a client DOS e Windows 221
  - lunghezza variabile 206
  - non recapitato 213
  - operatore 211
  - per eventi di strumentazione 125
  - ricerca di un particolare 206
  - velocità del messaggio non permanente 388
- messaggio non permanente 388
- modalità accodamento, di runmqsc 75
- modello code
  - descrizione 11
- modello di gruppi di ripristino 341
- modifica degli attributi di coda 50
- modifica degli attributi Queue Manager 42
- MQAI
  - descrizione di 64
- MQDATA 221
- MQDLH, dead-letter header 107
- MQI
  - autorizzazioni 96
  - chiamata Queue Manager 9
  - descrizione 5
  - supporto gestione locale 37
- MQI (Message Queue Interface) 5
- MQM
  - ID utente 92
  - identificativo diritti 85
- MQPUT e MQPUT1, considerazioni sulle prestazioni 207
- mqs.ini
  - definizione di 179
  - modifica 179
  - percorso per 45
  - priorità 180
  - voce AllQueueManagers 181
  - voce ClientExitPath 182
  - voce DefaultQueueManager 183
  - voce ExitProperties 183
  - voce LogDefaults 184
  - voce QueueManager 186
- MQSC
  - attributi 21
  - comandi 20
  - come immettere i comandi 38
  - emissione remota 75
  - escape PCF 21
  - esecuzione dei comandi da file di testo 42
  - file di comandi
    - esecuzione 44
    - input 43

MQSC (*Continua*)  
 notifiche di output 43  
 file di esempio 339  
 immissione interattiva di comandi 39  
 lunghezza massima della linea 43  
 problemi  
 locali 45  
 remoto 77  
 reindirizzamento di input e output 39  
 requisiti di sicurezza sui canali 95  
 risposte dai comandi scadute 76  
 sensibilità al  
 minuscolo/maiuscolo 25  
 termine di input interattivo 40  
 utilizzo dei comandi 42  
 verifica dei comandi 44  
 MQSC interattivi  
 feedback dei 40  
 termine 40  
 utilizzo 39  
 MQSeries  
 identificativo diritti, MQM 85  
 panoramica di OpenVMS 325  
 MQSPREFIX, variabile di ambiente 181  
 MsgId, considerazioni sulle prestazioni  
 durante l'utilizzo 206  
 MultiNet per OpenVMS 241  
 configurazione dei file modello 234

## N

Name Service 177  
 namelist  
 descrizione 14  
 utilizzato da cluster di Queue  
 Manager 14  
 NL carattere, EBCDIC conversione in  
 ASCII 181  
 nobody, identificativo diritti  
 predefinito 88  
 nome logico, disabilitazione  
 sicurezza 89  
 nome logico del sistema operativo,  
 disabilitazione della sicurezza 89  
 nome logico MQSNOAUT 89  
 nomi  
 consentiti per gli oggetti 251  
 numero massimo di caratteri 8  
 oggetti 8  
 notificazione di eventi 124

## O

OAM 86  
 OAM (object authority manager)  
 comando dspmqaut 92  
 comando setmqaut 89, 91  
 come funziona 87  
 disabilitazione 89  
 identificativo diritti predefinito 88  
 operazioni riservate 92  
 principal 87  
 object authority manager 86  
 oggetti  
 accesso a 85  
 attributi 8

oggetti (*Continua*)  
 coda 10  
 comando recreate 293  
 convenzioni di denominazione 251  
 definizione di processo 13  
 file rappresentante un oggetto 251  
 gestione 8  
 gestione remota 69  
 immagine del supporto 153  
 impostazione predefinita 14, 327  
 namelist 14  
 nomi 8, 38  
 per triggering 59  
 predefiniti  
 attributi 49  
 Queue Manager  
 Chiamata MQI 9  
 ripristino 154  
 tipi 7  
 oggetto  
 trasformazione del nome 23  
 OpenVMS  
 hardware richiesto 325  
 panoramica di 325  
 software richiesto 325  
 OpenVMS identificativo diritti  
 predefinito, nobody 88  
 OpenVMSID utente connesso 92  
 oracle  
 configurazione 131  
 impostazioni della variabile di  
 ambiente, controllo 131  
 livelli minimi di supporto 131  
 ORACLE\_HOME, variabile di  
 ambiente 131  
 ORACLE\_SID, variabile  
 d'ambiente 131  
 parametri di configurazione,  
 modifica 136  
 supporto XA Oracle, abilitazione 131  
 switch load file, creazione 132  
 Voce XAResourceManager, aggiunta a  
 qm.ini 134  
 ottimizzazione delle prestazioni 223  
 Ottimizzazione delle prestazioni 223  
 output  
 reindirizzare 368  
 output non corretto 207

## P

panoramica di MQSeries per Compaq  
 OpenVMS 325  
 parametri sysgen 223  
 parola chiave APPLIDAT, tabella  
 regole 111  
 parola chiave APPLNAME, tabella  
 regole 111  
 parola chiave APPLTYPE, tabella  
 regole 111  
 parola chiave DESTQ, tabella regole 111  
 parola chiave DESTQM, tabella  
 regole 111  
 parola chiave FEEDBACK, tabella  
 regole 111  
 parola chiave FORMAT, tabella  
 regole 111

parola chiave FWDQ, tabella regole 113  
 parola chiave FWDQM, tabella  
 regole 113  
 parola chiave HEADER, tabella  
 regole 113  
 parola chiave INPUTQ, tabella  
 regole 110  
 parola chiave INPUTQM, tabella  
 regole 110  
 parola chiave MSGTYPE, tabella  
 regole 112  
 parola chiave PERSIST, tabella  
 regole 112  
 parola chiave PUTAUT, tabella  
 regole 113  
 parola chiave REASON, tabella  
 regole 112  
 parola chiave REPLYQM, tabella  
 regole 112  
 parola chiave RETRY, tabella regole 113  
 parola chiave RETRYINT, tabella  
 regole 110  
 parola chiave USERID, tabella  
 regole 112  
 parola chiave WAIT, tabella regole 110  
 parole chiave dello schema di confronto,  
 tabella regole 111  
 parole chiave di azione, tabella  
 regole 112  
 parole chiave REPLYQ, tabella  
 regole 112  
 PCF (Programmable Command Format)  
 attributi 21  
 autorizzazioni 96  
 comandi, descrizione di 20  
 escape PCF 21  
 gestione con 20  
 requisiti di sicurezza 95  
 PDF (Portable Document Format) 396  
 Portable Document Format (PDF) 396  
 predefinita  
 coda di trasmissione 31  
 predefiniti  
 attributi di oggetti 49  
 oggetti 14  
 Queue Manager  
 comandi elaborati 39  
 modifica 41  
 predefinito  
 identificativo diritti per  
 l'autorizzazione 88  
 oggetti del sistema 327  
 Queue Manager 29  
 eliminazione involontaria 259  
 modifica 32  
 modifica accidentale 33  
 principal  
 contiene più di un identificativo di  
 diritti 87  
 gestione accesso a 87  
 problemi  
 eseguendo i comandi MQSC 45  
 risoluzione 152  
 utilizzando MQSC in locale 45  
 utilizzo remoto di MQSC 77  
 Procedura End command 233

- Procedura EndCommand
  - modello 344
- Procedura Start command 233
- Procedura StartCommand
  - modello 343
- Procedura Tidy command 233
- Procedura TidyCommand
  - modello 347
- procedure di comando 233
  - esempi 244
  - modifica 244
- processo
  - punto di avvio traccia 362
- programmi, esempi forniti 339
- pubblicazioni correlate 397
- Pubblicazioni MQSeries 395

## Q

- queue manager
  - endmqm comando 283
  - log 143
  - registrazione di immagini del supporto 153
  - registrazione nei log circolare, riavviare il ripristino 146
  - registrazione nei log lineare 145
  - server di comando 65
- Queue Manager
  - alias che utilizza la coda remota 80
  - arresto 33
  - arresto in un gruppo di ripristino 236
  - autorizzazioni 93
  - avvio 32
  - avvio in un gruppo di ripristino 235
  - chiusura
    - ad attività controllata 33
    - controllata 33
  - chiusura immediata 33
  - chiusura preventiva 34
  - cluster 12, 14
  - comando crtmqm 258
  - comando dltmqm 263
  - controllo 122
  - conversione di messaggi 81
  - creazione 28, 31
  - descrizione 9
  - directory 93
  - directory di autorizzazione 102
  - eliminazione 34
  - eventi 123
  - file di configurazione
    - specifica 31
  - file qm.ini 181
  - gestione locale 37
  - gestione remota 69
  - nome univoco 29
  - numeri di 29
  - OAM (object authority manager)
    - disabilitazione 89
  - object authority manager
    - descrizione 86
  - oggetti
    - chiamata MQI 9
    - predefinito 29
    - eliminazione involontaria 259

- Queue Manager (*Continua*)
  - predefinito 29 (*Continua*)
    - modifica 32
    - modifica accidentale 33
  - riavvio 34
  - riavvio dopo la modifica del CCSID 83
  - scarica una copia formattata del log di sistema (comando dmpmqlog) 266
  - scaricamento del contenuto di un log di ripristino 158
  - specifiche di runmqsc 41
  - Spostamento in un gruppo di ripristino 236
  - su MVS/ESA 76
  - trasformazione del nome 21
- queue manager di destinazione
  - avvio dei canali 74
- Queue Manager di destinazione
  - assegnazione di un alias a 80
  - creazione di canali 73
  - creazione di una coda di trasmissione 73
  - definizione 78
  - estraneo al cluster 12
- Queue Manager MVS/ESA 76
- quote di pool 224

## R

- registrazione nei log
  - checkpoint 146
  - circolare 144
  - lineare 145
  - ripristino di supporti 154
  - tipi di 144
- registrazione nei log circolare 144
- registrazione nei log lineare 145
- reindirizzamento di input e output, con comandi MQSC 39
- remota
  - considerazioni sulla sicurezza 94
  - definizione di coda, creazione 77
  - emissione di comandi MQSC 75
  - gestione 71
  - oggetto coda, funzionante con 80
- remote
  - code
    - come alias di coda di risposta RTQ 81
    - come alias di Queue Manager 80
- remoto
  - accodamento
  - consigli 76
- restrizioni
  - accesso agli oggetti MQM 85
  - nomi degli oggetti 251
  - Supporto di coordinazione di database 129
- riavviare il ripristino, registrazione nei log circolare 146
- riavvio di un Queue Manager 34
- ripristino di immagini del supporto 154
- risorse
  - perché proteggere 85
  - protette 88
- risorse protette 88

- risposte scadute dai comandi MQSC 76
- runmqsc
  - feedback 40
  - immissione di comandi MQSC 38
  - modalità accodamento 75
  - problemi 45
  - specifiche di un Queue Manager 41
  - termine 40
  - utilizzo 42
  - utilizzo interattivo 39
  - verifica 44

## S

- SAFEGUARD 325
- script
  - MONMQ 373
- sensibilità al minuscolo/maiuscolo 23
  - comandi di controllo 23
  - comandi MQSC 25
  - nomi di Queue Manager 23
- server di comandi
  - comando start 317
- server di comando
  - arresto del server di comando 67
  - avvio di un server di comando 65
  - gestione remota 65
  - visualizza comando 272
  - visualizzazione dello stato 65
- servizi installabili
  - disabilitazione di OAM (object authority manager)
    - disabilitazione 89
  - Name Service 177
  - object authority manager 86
- sezioni globali 357
- sicurezza 85
  - abilitazione 89
  - failover monitor del gruppo 240
  - remota 94
  - utilizzo dei comandi 89, 92
- sintassi
  - diagrammi, come leggere 252
  - errore, nei comandi MQSC 40
  - guida 253
- sistema
  - oggetti predefiniti 14, 327
- situazioni di ripristino
  - malfunzionamenti dell'unità disco 157
  - oggetto danneggiato del Queue Manager 158
  - oggetto singolo danneggiato 158
- struttura di directory 329
- supporto euro 326
- supporto lingua nazionale
  - convenzioni di denominazione 251
  - conversione del carattere EBCDIC NL in ASCII 181
- supporto transazionale
  - supporto transazionale 127
- SupportPac 397
- switch load file, creazione 130
- syncpoint, considerazioni sulle prestazioni 206
- SYS\$INPUT, su runmqsc 42
- SYS\$OUTPUT, su runmqsc 42

## T

tabella di memoria 360  
tabella mutex 358  
tabella regole, DLQ handler 108  
  elaborazione di 115  
  esempio 118  
  schemi e azioni, (regole)  
    parola chiave FWDQM 113  
  schemi e azioni (regole) 111  
    parola chiave ACTION 112  
    parola chiave APPLIDAT 111  
    parola chiave APPLNAME 111  
    parola chiave APPLTYPE 111  
    parola chiave DESTQ 111  
    parola chiave DESTQM 111  
    parola chiave FEEDBACK 111  
    parola chiave FORMAT 111  
    parola chiave FWDQ 113  
    parola chiave HEADER 113  
    parola chiave MSGTYPE 112  
    parola chiave PERSIST 112  
    parola chiave PUTAUT 113  
    parola chiave REASON 112  
    parola chiave REPLYQ 112  
    parola chiave REPLYQM 112  
    parola chiave RETRY 113  
    parola chiave USERID 112  
  sintassi 114  
  voce dati di controllo 109  
    parola chiave INPUTQ 110  
    parola chiave INPUTQM 110  
    parola chiave RETRYINT 110  
    parola chiave WAIT 110  
TCP/IP  
  avvio dei canali 74  
  configurato con la procedura  
    StartCommand 234  
  configurazione 193  
  connessione rifiutata 194  
  definito per il processo listener 303  
  Digital TCP/IP Services per  
    OpenVMS 234  
  gestione remota 15  
  impostazione di canali 72  
  operazioni necessarie per l'attività del  
    cluster OpenVMS 230  
  pacchetti supportati 325  
  supporta OpenVMS cluster 228  
termini dei comandi interattivi  
  MQSC 40  
terminologia utilizzata in questo  
  manuale 399  
timestamp  
  disable timestamp (MONMQ) 365  
  enable timestamp (MONMQ) 365  
tipi di eventi 123  
tipi di oggetti 7  
traccia  
  abbandona la traccia MONMQ 371  
  disabilita traccia (MONMQ) 366  
  enable trace (MONMQ) 365  
  esci dalla traccia (MONMQ) 371  
  esempio di sessione MONMQ 374  
  si avvia all'avvio del processo 362  
  specificare componente o  
    funzione 363  
  start trace (MONMQ) 363

traccia (*Continua*)  
  stop trace (MONMQ) 363  
  utilizzo di MONMQ 351  
tracce  
  considerazioni sulle prestazioni 214  
  esempio di dati 215  
transazioni  
  comando display MQSeries 277  
  comando resolve MQSeries 296  
transazioni sospese  
  gestori di database 137  
trigger  
  code evento 124  
  confronto con gli eventi di  
    strumentazione 124  
  messaggi su code di iniziazione 11  
  monitor  
    comando start 309  
    descrizione 12  
    utilizzo di namelist 14  
  monitor guasti  
    comando start 302  
triggering 389  
  coda di applicazione, definizione 59  
  definizione 5  
  gestione degli oggetti per 59

## U

unità di elaborazione globali  
  aggiunta della voce  
    XAResourceManager a qm.ini,  
    Oracle 134  
  definizione di 127  
unità di elaborazione locale  
  definizione di 127  
unità logica  
  close (MONMQ) 355  
  visualizzazione mediante l'utilizzo di  
    show segment (MONMQ) 354  
UOW (unit of work)  
  definizione di 127  
  risincronizzazione esplicita di  
    (comando rsvmqtrn) 139  
  risultati diversi 139  
uscita  
  cluster workload 385  
  cluster workload exit 17  
  uscita del canale 17  
  uscita utente 17  
uscita utente 385  
  cluster workload 17, 385  
  descrizione 17  
  uscita del canale 17  
  uscita di conversione dati 17  
Utilità AUTHORIZE 224  
utilità monmq  
  comandi  
    analizza traccia 368  
    close binary 365  
    close lu 355  
    close text 365  
    connect 362  
    delete history 366  
    deselect index 364  
    disable history 366  
    disable timestamp 365

utilità monmq (*Continua*)

  comandi (*Continua*)  
    disable trace 366  
    disconnect 363  
    enable history 366  
    enable timestamp 365  
    enable trace 365  
    exit 371  
    FFST 371  
    onstartup start 362  
    onstartup stop 362  
    open 354  
    open binary 364  
    open text 365  
    select 363  
    set color 368  
    set depth 366  
    set free 366  
    set mask 367  
    set output 368  
    show channels 355  
    show components 361  
    show functions 361  
    show globals 357  
    show history 356  
    show mask 355  
    show memory 360  
    show mutex 358  
    show process 356  
    show segment 354  
    show stack 356  
    trace start 351, 363  
    trace stop 363  
    variabile predefinita 353  
    visualizzazione di eventi 359  
  gestione della memoria condivisa  
    con 372  
  panoramica 351  
  tracciamento di processi MQSeries  
    esempio di sessione di traccia 374  
    script e macro in MONMQ 373  
    variabili in MONMQ 352

## V

variabili di ambiente  
  MQSPREFIX 181  
  ORACLE\_HOME, Oracle 131  
  ORACLE\_SID, Oracle 131  
verifica dei comandi MQSC 44  
visualizza  
  comando autorizzazione 268  
  comando output di traccia formattata  
    di MQSeries 275  
  comando server di comando 272  
  file MQSeries comando 273  
  ID esadecimali per componenti 361  
  transazioni MQSeries comando 277  
visualizzazione  
  attributi Queue Manager 40  
  definizioni di processo 61  
  processi attivi MQSeries 356  
  stack dei sottoprocessi di  
    destinazione 356  
  stato del server di comando 65  
  tabella di memoria 360  
Voce AllQueueManagers, mqs.ini 181



- voce Channels, qm.ini 191
- voce ClientExitPath, mqs.ini 182
- voce DefaultQueueManager, mqs.ini 183
- voce Exitpath, qm.ini 194
- voce ExitProperties, mqs.ini 183
- voce Log, qm.ini 188
- voce LogDefaults, mqs.ini 184
- voce LU62, qm.ini 193
- voce QueueManager, mqs.ini 186
- voce Service, qm.ini 186
- voce ServiceComponent, qm.ini 187
- voce TCP, qm.ini 193
- voce XAResourceManager, qm.ini 190
- voci
  - AllQueueManagers, mqs.ini 181
  - Channels, qm.ini 191
  - ClientExitPath, mqs.ini 182
  - DefaultQueueManager, mqs.ini 183
  - ExitPath, qm.ini 194
  - ExitProperties, mqs.ini 183
  - Log, qm.ini 188
  - LogDefaults, mqs.ini 184
  - LU62, qm.ini 193
  - QueueManager, mqs.ini 186
  - Service, qm.ini 186
  - ServiceComponent, qm.ini 187
  - TCP, qm.ini 193
  - XAResourceManager, qm.ini 190

## W

- watcher failover monitor
  - descrizione 229
- Windows Help 397

---

# Riservato ai commenti del lettore

MQSeries® per Compaq OpenVMS Alpha®  
Guida alla gestione del sistema  
Versione 5 Rilascio 1

Pubblicazione N. SC13-2966-00

Commenti relativi alla pubblicazione in oggetto potranno contribuire a migliorarla. Sono graditi commenti pertinenti alle informazioni contenute in questo manuale ed al modo in cui esse sono presentate. Si invita il lettore ad usare lo spazio sottostante citando, ove possibile, i riferimenti alla pagina ed al paragrafo.

Si prega di non utilizzare questo foglio per richiedere informazioni tecniche su sistemi, programmi o pubblicazioni e/o per richiedere informazioni di carattere generale.

Per tali esigenze si consiglia di rivolgersi al punto di vendita autorizzato o alla filiale IBM della propria zona oppure di chiamare il "Supporto Clienti" IBM al numero verde 167-017001.

I suggerimenti ed i commenti inviati potranno essere usati liberamente dall'IBM e dalla Selfin e diventeranno proprietà esclusiva delle stesse.

Commenti:

Si ringrazia per la collaborazione.

Per inviare i commenti è possibile utilizzare uno dei seguenti modi.

- Spedire questo modulo all'indirizzo indicato sul retro.
- Inviare un fax al numero: +39-081-660236
- Spedire una nota via email a: [translationassurance@selfin.it](mailto:translationassurance@selfin.it)

Se è gradita una risposta dalla Selfin, si prega di fornire le informazioni che seguono:

---

Nome

---

Indirizzo

---

Società

---

Numero di telefono

---

Indirizzo e-mail

Indicandoci i Suoi dati, Lei avrà l'opportunità di ottenere dal responsabile del Servizio di Translation Assurance della Selfin S.p.A. le risposte ai quesiti o alle richieste di informazioni che vorrà sottoporci. I Suoi dati saranno trattati nel rispetto di quanto stabilito dalla legge 31 dicembre 1996, n.675 sulla "Tutela delle persone e di altri soggetti rispetto al trattamento di dati personali". I Suoi dati non saranno oggetto di comunicazione o di diffusione a terzi; essi saranno utilizzati "una tantum" e saranno conservati per il tempo strettamente necessario al loro utilizzo.

Selfin S.p.A.  
Translation Assurance

Via F. Giordani, 7

80122 NAPOLI







Numero parte: CT8YMIT

Printed in Denmark by IBM Danmark A/S

SC13-2966-00



(1P) P/N: CT8YMIT

