



Sun Java™ System
Message Queue 3.5
관리 설명서

서비스 팩 1

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
U.S.A.

부품 번호: 817-7207

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. 모든 권리는 저작권자의 소유입니다. Sun Microsystems, Inc.는 이 문서에 설명된 제품의 기술 관련 지적 재산권을 소유합니다. 특히 이 지적 재산권에는 <http://www.sun.com/patents>에 나열된 하나 이상의 미국 특허권이 포함될 수 있으며, 미국 및 다른 국가에서 하나 이상의 추가 특허권 또는 출원 중인 특허권이 제한 없이 포함될 수 있습니다.

미국 정부의 권리 - 상용 소프트웨어. 정부 사용자는 Sun Microsystems, Inc. 표준 사용권 계약과 해당 FAR 규정 및 보충 규정을 준수해야 합니다. 본 제품의 사용은 사용권 조항의 적용을 받습니다. 이 배포에는 타사에서 개발한 자료가 포함되어 있을 수 있습니다.

Sun, Sun Microsystems, Sun 로고, Java, Solaris, Sun[tm] ONE, JDK, Java Naming and Directory Interface, Javadoc, JavaMail, JavaHelp, Java Coffee Cup 로고 및 Sun[tm] ONE 로고는 미국 및 다른 국가에서 Sun Microsystems, Inc.의 상표 또는 등록 상표입니다.

모든 SPARC 상표는 사용 허가를 받았으며 미국 및 다른 국가에서 SPARC International, Inc.의 상표 또는 등록 상표입니다. SPARC 상표를 사용하는 제품은 Sun Microsystems, Inc.가 개발한 구조를 기반으로 하고 있습니다.

UNIX는 미국 및 다른 국가에서 X/Open Company, Ltd.를 통해 독점적으로 사용권이 부여되는 등록 상표입니다.

이 제품은 미국 수출 관리법에 의해 규제되며 다른 국가의 수출 또는 수입 관리법의 적용을 받을 수도 있습니다. 이 제품과 정보를 직간접적으로 핵무기, 미사일 또는 생화학 무기에 사용하거나 핵과 관련하여 해상에서 사용하는 것은 엄격하게 금지됩니다. 미국 수출 금지 국가 또는 금지된 개인과 특별히 지정된 국민 목록을 포함하여 미국 수출 금지 목록에 지정된 대상으로의 수출이나 재수출은 엄격하게 금지됩니다.

목차

그림 목차	15
표 목차	17
절차 목록	21
머리말	23
대상	23
구성	24
활자체 규약	25
텍스트 활자체 규약	25
디렉토리 변수 규칙	26
기타 설명서 자원	28
Message Queue 설명서 집합	28
온라인 도움말	29
JavaDoc	29
클라이언트 응용 프로그램의 예	29
JMS (Java Message Service) 사양	29
관련 타사 웹 사이트 참조	30
1장 개요	31
Sun Java System Message Queue 소개	31
제품 관	33
플랫폼관	33
엔터프라이즈관	34
엔터프라이즈 메시징 시스템	34
엔터프라이즈 메시징 시스템의 요구 사항	34
중앙 집중식 메시징과 피어 투 피어(Peer to Peer) 메시징 비교	35

메시징 시스템 개념	36
메시지	36
메시지 서비스 구조	36
메시지 전달 모델	37
JMS 사양	38
JMS 메시지 구조	38
JMS 프로그래밍 모델	38
JMS 관리 대상 객체	40
JMS/J2EE 프로그래밍: Message-Driven Bean	40
Message-Driven Bean	41
J2EE Application Server 지원	42
JMS 메시징 문제	43
JMS 공급자 독립성	43
프로그래밍 도메인	44
클라이언트 식별자	45
안정적인 메시징	46
확인/트랜잭션	46
영구 저장소	48
성능 균형	49
메시지 선택	49
메시지 순서 및 우선 순위	49
2장 Message Queue 메시징 시스템	51
Message Queue 메시지 서버	52
브로커	52
연결 서비스	54
메시지 라우터	58
지속성 관리자	63
보안 관리자	66
모니터링 서비스	70
물리적 대상	76
대기열 대상	77
주제 대상	78
자동 작성(대 관리 작성) 대상	78
임시 대상	81
멀티 브로커 클러스터(엔터프라이즈판)	82
멀티 브로커 구조	82
개발 환경에서의 클러스터 사용	85
클러스터 구성 등록 정보	86

Message Queue 클라이언트 런타임	86
메시지 생성	87
메시지 사용	88
Message Queue 관리 대상 객체	89
연결 팩토리 관리 대상 객체	90
대상 관리 대상 객체	91
클라이언트 시작 시 속성 값 무시	91
3장 Message Queue 관리 작업 및 도구	93
Message Queue 관리 작업	93
개발 환경	93
작업 환경	94
설정 작업	94
작업 환경을 설정하는 방법	94
유지 보수 작업	95
작업 환경을 설정하는 방법	95
Message Queue 관리 도구	97
관리 콘솔	97
명령줄 유틸리티 요약	98
명령줄 구문	99
공통 명령줄 옵션	100
4장 관리 콘솔 자습서	101
준비	102
관리 콘솔 시작	102
관리 콘솔을 시작하는 방법	103
도움말 확인	104
관리 콘솔 도움말 정보를 표시하는 방법	104
브로커 작업	105
브로커 시작	106
브로커를 시작하는 방법	106
브로커 추가	106
관리 콘솔에 브로커를 추가하는 방법	107
관리 비밀번호 변경	108
관리자 비밀번호를 변경하는 방법	108
브로커에 연결	108
브로커에 연결하는 방법	108
연결 서비스 보기	109
사용 가능한 연결 서비스를 보는 방법	109
브로커에 물리적 대상 추가	110
브로커에 대기열 대상을 추가하는 방법	111

물리적 대상 작업	112
물리적 대상의 등록 정보를 보는 방법	112
대상에서 메시지를 제거하는 방법	113
대상을 삭제하는 방법	114
주제 대상에 대한 정보 얻기	114
객체 저장소 작업	115
객체 저장소 추가	115
파일 시스템 객체 저장소를 추가하는 방법	115
객체 저장소 등록 정보 확인	118
객체 저장소 등록 정보를 표시하는 방법	118
객체 저장소에 연결	118
객체 저장소에 연결하는 방법	118
연결 팩토리 관리 객체 추가	118
객체 저장소에 연결 팩토리를 추가하는 방법	119
대상 관리 대상 객체 추가	120
객체 저장소에 대상을 추가하는 방법	121
관리 대상 객체 등록 정보	122
대상 객체의 등록 정보를 확인 또는 업데이트하는 방법	122
콘솔 정보 업데이트	123
샘플 응용 프로그램 실행	123
HelloWorldMessageJNDI 응용 프로그램을 실행하는 방법	124
5장 브로커 시작 및 구성	127
구성 파일	127
인스턴스 구성 파일	128
등록 정보 값 병합	128
등록 정보 이름 지정 구문	129
인스턴스 구성 파일 편집	129
브로커 시작	134
imqbrokerd 명령 구문	135
시작 예	136
기본 브로커 이름 및 구성을 사용하는 브로커 인스턴스를 시작하는 방법	136
엔터프라이즈판 시험 사용권으로 브로커 인스턴스를 시작하는 방법	136
플러그인 지속성을 사용하여 명명된 브로커 인스턴스를 시작하는 방법	136
imqbrokerd 옵션 요약	136
클러스터를 이용한 작업(엔터프라이즈판)	140
클러스터 구성 등록 정보	140
브로커 연결	142
연결 방법	142
브로커를 클러스터에 연결하는 방법	143
브로커간 연결 보안	143
클러스터 내에서 보안 연결을 구성하는 방법	143

클러스터의 브로커 관리	143
클러스터에 브로커 추가	144
기존 클러스터에 새 브로커를 추가하는 방법	144
클러스터에서 브로커 다시 시작	144
이미 기존 클러스터의 구성원인 브로커를 다시 시작하는 방법	144
클러스터에서 브로커 제거	145
기존 클러스터에서 브로커를 제거하는 방법	145
마스터 브로커의 구성 변경 기록 관리	145
구성 변경 기록 백업	146
구성 변경 기록을 백업하는 방법	146
구성 변경 기록 복원	146
문제가 발생한 경우 마스터 브로커를 복원하는 방법	146
로그	147
기본 로그 구성	147
로그 메시지 형식	148
로거 구성 변경	148
브로커의 로거 구성을 변경하는 방법	148
출력 채널 변경	149
로그 파일 롤오버 기준 변경	150
6장 브로커 및 응용 프로그램 관리	151
명령 유틸리티	152
imqcmd 명령 구문	152
imqcmd 하위 명령	152
imqcmd 옵션 요약	154
imqcmd 명령 사용	156
imqcmd 사용 예	157
브로커 관리	157
브로커 정보 표시	159
브로커 등록 정보 업데이트	160
브로커 상태 제어	160
브로커 일시 중지 및 다시 시작	161
브로커 종료 및 다시 시작	161
브로커 메트릭 표시	162
연결 서비스 관리	162
연결 서비스 나열	164
연결 서비스 정보 표시	165
연결 서비스 등록 정보 업데이트	165
연결 서비스 메트릭 표시	166
연결 서비스 일시 중지 및 다시 시작	166
연결 정보 얻기	167

대상 관리	168
대상 만들기	170
대상 나열	173
대상 정보 표시	173
대상 속성 업데이트	174
대상 메트릭 표시	175
대상 일시 중지 및 다시 시작	175
대상 제거	176
대상 완전 삭제	176
대상 압축	176
대상 디스크 사용률 모니터링	177
사용되지 않은 대상 디스크 공간 확보	178
사용되지 않는 대상 디스크 공간을 확보하는 방법	178
영구 가입 관리	179
트랜잭션 관리	180

7장 관리 대상 객체 관리 183

객체 저장소 정보	184
LDAP 서버 객체 저장소	184
파일 시스템 객체 저장소	185
관리 대상 객체	186
연결 팩토리 관리 대상 객체 속성	187
대상 관리 대상 객체 속성	189
객체 관리자 유틸리티(imqobjmgr)	189
imqobjmgr 명령 구문	189
imqobjmgr 하위 명령	190
imqobjmgr 명령 옵션 요약	190
필요한 정보	192
명령 파일 사용	193
관리 대상 객체 추가 및 삭제	195
연결 팩토리 추가	195
주제 또는 대기열 추가	196
관리 대상 객체 삭제	198
정보 얻기	198
관리 대상 객체 나열	198
단일 객체 관련 정보	199
관리 대상 객체 업데이트	200

8장 보안 관리	201
사용자 인증	202
플랫 파일 사용자 저장소 사용	202
사용자 저장소 만들기	203
사용자 관리자 유틸리티(imqusermgr)	203
그룹	205
상태	206
사용자 아이디 및 비밀번호 형식	206
사용자 저장소 채우기 및 관리	207
기본 관리자 비밀번호 변경	208
사용자 저장소에 LDAP 서버 사용	209
LDAP 서버를 사용하도록 구성 파일을 편집하는 방법	209
사용자 권한 부여:	
액세스 제어 등록 정보 파일	212
액세스 제어 등록 정보 파일 작성	213
액세스 규칙 구문	213
권한 계산	215
연결 액세스 제어	216
대상 액세스 제어	216
대상 자동 작성 액세스 제어	217
암호화: SSL 기반 서비스를 사용한 작업(엔터프라이즈판)	218
TCP/IP에서 SSL 기반 서비스 설정	219
SSL 기반 연결 서비스를 설정하는 방법	219
1단계. 자체 서명된 인증서 생성	219
키 쌍을 다시 생성하는 방법	221
2단계. 브로커에서 SSL 기반 서비스 활성화	221
브로커에서 SSL 기반 서비스를 활성화하는 방법	222
3단계. 브로커 시작	222
4단계. SSL기반 클라이언트 구성 및 실행	223
HTTP에서 SSL 서비스 설정	224
Passfile 사용	225
9장 메시지 서비스 분석 및 조정	227
성능 정보	227
성능 조정 프로세스	227
성능 요소	228
벤치마크	229
기본 사용 패턴	230

성능에 영향을 미치는 요소	231
성능에 영향을 미치는 응용 프로그램 설계 요소	232
전달 모드(지속성/비지속성 메시지)	234
트랜잭션 사용	235
확인 모드	236
영구 가입 및 비영구 가입	237
선택기 사용(메시지 필터링)	238
메시지 크기	238
메시지 본문 유형	239
성능에 영향을 미치는 메시지 서비스 요소	240
하드웨어	240
운영 체제	241
Java 가상 머신(JVM)	241
연결	241
메시지 서비스 구조	243
브로커 제한 및 동작	244
데이터 저장소 성능	244
클라이언트 런타임 구성	245
메시지 서버 모니터링	245
모니터링 도구	246
Message Queue 명령 유틸리티(imqcmd)	246
메트릭 하위 명령을 사용하는 방법	248
Message Queue 브로커 로그 파일	251
로그 파일을 사용하여 메트릭 정보를 보고하는 방법	251
메시지 기반 모니터링 API	252
메시지 기반 모니터링을 설정하는 방법	253
올바른 모니터링 도구 선택	255
메트릭 데이터 설명	257
JVM 메트릭	257
브로커 전체 메트릭	258
연결 서비스 메트릭	260
대상 메트릭	261
성능 문제 해결	264
문제: 클라이언트가 연결을 설정할 수 없음	264
증상:	264
가능한 원인:	264
문제: 연결 처리량이 너무 느림	269
증상:	269
가능한 원인:	269
문제: 클라이언트가 메시지 생성자를 만들 수 없음	270
증상:	270
가능한 원인:	271

문제: 메시지 생성이 지연되거나 느림	272
증상:	272
가능한 원인:	272
문제: 메시지가 메시지 서버에서 백로그됨	275
증상:	275
가능한 원인:	275
문제: 메시지 서버 처리량이 일정하지 않음	279
증상:	279
가능한 원인:	279
문제: 메시지가 사용자에게 도달하지 않음	281
증상:	281
가능한 원인:	281
성능 향상을 위한 구성 조정	282
시스템 조정	282
Solaris 조정: CPU 사용률, 페이징/스왑/디스크 입출력	282
Java 가상 머신 조정	283
전송 프로토콜 조정	283
파일 기반 영구 저장소 조정	287
브로커 조정	287
메모리 관리: 로드 하에서 브로커 안정성 증가	287
다중 사용자 대기열 성능	288
클라이언트 런타임 메시지 흐름 조정	289
메시지 흐름 측정	289
메시지 흐름 제한	289
부록 A Message Queue 데이터의 위치	293
Solaris	293
Linux	294
Windows	295
부록 B 플러그인 지속성 설정	297
소개	297
JDBC로 액세스할 수 있는 데이터 저장소 플러그인	298
JDBC로 액세스할 수 있는 데이터 저장소를 플러그인하는 방법	298
JDBC 관련 브로커 구성 등록 정보	299
데이터베이스 관리자 유틸리티(imqdbmgr)	303
imqdbmgr 명령의 구문	303
imqdbmgr 하위 명령	304
imqdbmgr 명령 옵션 요약	305

부록 C HTTP/HTTPS 지원(엔터프라이즈판)	307
HTTP/HTTPS 지원 구조	307
HTTP 지원 활성화	309
HTTP 지원을 활성화하는 방법	309
1단계. Web Server에 HTTP 터널 서블릿 배포	309
Jar 파일로 배포	309
웹 아카이브 파일로 배포	310
2단계. httpjms 연결 서비스 구성	310
httpjms 연결 서비스를 활성화하는 방법	311
3단계. HTTP 연결 구성	312
연결 팩토리 구성	312
단일 서블릿을 사용하여 다중 브로커에 액세스	313
HTTP 프록시 사용	313
예 1: Sun Java System Web Server에 HTTP 터널 서블릿 배포	314
Jar 파일로 배포	314
터널 서블릿을 추가하는 방법	314
터널 서블릿에 대한 가상 경로(서블릿 URL)를 구성하는 방법	315
Web Server 시작 시 터널 서블릿을 로드하는 방법	315
서버 액세스 로그를 비활성화하는 방법	316
WAR 파일로 배포	316
http 터널 서블릿을 WAR 파일로 배포하는 방법	316
예 2: Sun Java System Application Server 7.0에 HTTP 터널 서블릿 배포	317
배포 도구 사용	317
Application Server 7.0 환경에 HTTP 터널 서블릿을 배포하는 방법	317
server.policy 파일 수정	318
Application Server의 server.policy 파일을 수정하는 방법	319
HTTPS 지원 활성화	319
HTTPS 지원을 활성화하는 방법	319
1단계. HTTPS 터널 서블릿에 대해 자체 서명된 인증서 생성	319
2단계. Web Server에 HTTPS 터널 서블릿 배포	320
Jar 파일로 배포	321
웹 아카이브 파일로 배포	321
3단계. httpsjms 연결 서비스 구성	322
httpsjms 연결 서비스를 활성화하는 방법	322
4단계. HTTPS 연결 구성	323
JSSE 구성	324
JSSE를 구성하는 방법	324
루트 인증서 가져오기	324
연결 팩토리 구성	325
단일 서블릿을 사용하여 다중 브로커에 액세스	325
HTTP 프록시 사용	326

예 3: Sun Java System Web Server에 HTTPS 터널 서블릿 배포	326
Jar 파일로 배포	326
터널 서블릿을 추가하는 방법	327
터널 서블릿에 대한 가상 경로(서블릿 URL)를 구성하는 방법	328
Web Server 시작 시 터널 서블릿을 로드하는 방법	328
서버 액세스 로그를 비활성화하는 방법	329
WAR 파일로 배포	329
HTTPS 터널 서블릿 WAR 파일을 수정하는 방법	329
https 터널 서블릿을 WAR 파일로 배포하는 방법	330
예 4: Sun Java System Application Server 7.0에 HTTPS 터널 서블릿 배포	331
배포 도구 사용	331
Application Server 7.0 환경에 HTTPS 터널 서블릿을 배포하는 방법	331
server.policy 파일 수정	332
Application Server의 server.policy 파일을 수정하는 방법	332
부록 D 브로커를 Windows 서비스로 사용	333
브로커를 Windows 서비스로 실행	333
서비스 관리자 유틸리티(imqsvcadmin)	334
imqsvcadmin 명령의 구문	334
imqsvcadmin 하위 명령	334
imqsvcadmin 옵션 요약	335
브로커 서비스 제거	335
브로커 서비스 재구성	336
대체 Java 런타임 사용	336
브로커 서비스 쿼리	336
문제 해결	336
기록된 서비스 오류 이벤트를 보는 방법	336
부록 E 기술 노트	337
시스템 클럭 설정	337
동기화 권장	337
역방향 시스템 클럭 설정 금지	338
OS 정의 파일 설명자 제한	338
지속성 데이터 보안	339
기본 제공 영구 저장소	339
플러그 인 영구 저장소	340

부록 F Message Queue 자원 어댑터	341
부록 G 선택적 JMS 기능의 Message Queue 구현	343
부록 H Message Queue 인터페이스의 안정성	345
용어집	349
색인	353

그림 목차

그림 1-1	중앙 집중식 메시징과 피어 투 피어(Peer to Peer) 메시징 비교	35
그림 1-2	메시지 서비스 구조	37
그림 1-3	JMS 프로그래밍 객체	39
그림 1-4	MDB와의 메시지	42
그림 2-1	Message Queue 시스템 구조	51
그림 2-2	브로커 서비스 구성 요소	53
그림 2-3	연결 서비스 지원	55
그림 2-4	지속성 관리자 지원	64
그림 2-5	보안 관리자 지원	68
그림 2-6	모니터링 서비스 지원	71
그림 2-7	멀티 브로커(클러스터) 구조	83
그림 2-8	메시징 작업	87
그림 2-9	Message Queue 클라이언트 런타임으로의 메시지 전달	88
그림 3-1	로컬 및 원격 관리 유틸리티	98
그림 5-1	브로커 구성 파일	129
그림 9-1	Message Queue 서비스를 통한 메시지 전달	231
그림 9-2	전달 모드의 성능 영향	235
그림 9-3	가입 유형의 성능 영향	237
그림 9-4	메시지 크기의 성능 영향	239
그림 9-5	전송 프로토콜 속도	242
그림 9-6	전송 프로토콜의 성능 영향	243
그림 9-7	1k (1024바이트) 패킷에 대한 inbufsz 변경의 결과	285
그림 9-8	1k (1024바이트) 패킷에 대한 outbufsz 변경의 결과	286
그림 C-1	HTTP/HTTPS 지원 구조	308

표 목차

표 1	설명서 내용	24
표 2	문서 활자체 규약	25
표 3	Message Queue 디렉토리 변수	26
표 4	Message Queue 설명서 집합	28
표 1-1	JMS 프로그래밍 객체	45
표 2-1	주요 브로커 구성 요소 및 기능	53
표 2-2	브로커가 지원하는 연결 서비스	54
표 2-3	연결 서비스 등록 정보	57
표 2-4	메시지 라우터 등록 정보	62
표 2-5	지속성 관리자 등록 정보	66
표 2-6	보안 관리자 등록 정보	69
표 2-7	로깅 범주	72
표 2-8	메트릭 주제 대상	73
표 2-9	모니터링 서비스 등록 정보	74
표 2-10	자동 작성 구성 등록 정보	79
표 2-11	대상 속성	91
표 3-1	공통 Message Queue 명령줄 옵션	100
표 5-1	브로커 인스턴스 구성 등록 정보	130
표 5-2	imqbrokerd 옵션	136
표 5-3	클러스터 구성 등록 정보	140
표 5-4	imqbrokerd 로거 옵션 및 해당 등록 정보	148
표 6-1	imqcmd 하위 명령	152
표 6-2	imqcmd 옵션	154
표 6-3	브로커 관리에 사용되는 imqcmd 하위 명령	158
표 6-4	imqcmd가 업데이트하는 브로커 등록 정보	160
표 6-5	연결 서비스 관리에 사용되는 imqcmd 하위 명령	163
표 6-6	브로커가 지원하는 연결 서비스	164

표 6-7	imqcmd가 업데이트하는 연결 서비스 등록 정보	165
표 6-8	연결 서비스 관리에 사용되는 imqcmd 하위 명령	167
표 6-9	대상 관리에 사용되는 imqcmd 하위 명령	168
표 6-10	대상 속성	171
표 6-11	대상 디스크 사용률 메트릭	177
표 6-12	영구 가입 관리에 사용되는 imqcmd 하위 명령	179
표 6-13	트랜잭션 관리에 사용되는 imqcmd 하위 명령	180
표 7-1	LDAP 객체 저장소 속성	184
표 7-2	파일 시스템 객체 저장소 속성	186
표 7-3	연결 팩토리 관리 대상 객체 속성	187
표 7-4	대상 관리 대상 객체 속성	189
표 7-5	imqobjmgr 하위 명령	190
표 7-6	imqobjmgr 옵션	190
표 7-7	이름 지정 규약 예	196
표 8-1	사용자 저장소 초기 항목	203
표 8-2	imqusermgr 하위 명령	204
표 8-3	imqusermgr 옵션	205
표 8-4	사용자 아이디 및 비밀번호에 유효하지 않은 문자	206
표 8-5	LDAP 관련 등록 정보	210
표 8-6	액세스 규칙의 구문 요소	214
표 8-7	대상 액세스 제어 규칙의 요소	217
표 8-8	키 저장소 등록 정보	220
표 8-9	Passfile의 비밀번호	225
표 9-1	높은 안정성 및 높은 성능 시나리오 비교	233
표 9-2	imqcmd metrics 하위 명령 구문	247
표 9-3	imqcmd metrics 하위 명령 옵션	247
표 9-4	imqcmd query 하위 명령 구문	250
표 9-5	메트릭 주제 대상	253
표 9-6	메트릭 모니터링 도구의 장점 및 단점	256
표 9-7	JVM 메트릭	257
표 9-8	브로커 전체 메트릭	258
표 9-9	연결 서비스 메트릭	260
표 9-10	대상 메트릭	262
표 A-1	Solaris에서 Message Queue 데이터의 위치	293
표 A-2	Linux에서 Message Queue 데이터의 위치	294
표 A-3	Windows에서 Message Queue 데이터의 위치	295
표 B-1	JDBC 관련 등록 정보	300

표 B-2	imqdbmgr 하위 명령	304
표 B-3	imqdbmgr 옵션	305
표 C-1	httpjms 연결 서비스 등록 정보	311
표 C-2	HTTP 터널 서블릿 Jar 파일 배포에 사용되는 서블릿 인수	315
표 C-3	httpsjms 연결 서비스 등록 정보	323
표 C-4	HTTPS 터널 서블릿 Jar 파일 배포에 사용되는 서블릿 인수	327
표 D-1	imqsvcadm 하위 명령	334
표 D-2	imqsvcadm 옵션	335
표 G-1	선택적 JMS 기능	343
표 H-1	Message Queue 인터페이스의 안정성	345
표 H-2	인터페이스 안정성 분류 체계	347

절차 목록

작업 환경을 설정하는 방법	94
작업 환경을 설정하는 방법	95
관리 콘솔을 시작하는 방법	103
관리 콘솔 도움말 정보를 표시하는 방법	104
브로커를 시작하는 방법	106
관리 콘솔에 브로커를 추가하는 방법	107
관리자 비밀번호를 변경하는 방법	108
브로커에 연결하는 방법	108
사용 가능한 연결 서비스를 보는 방법	109
브로커에 대기열 대상을 추가하는 방법	111
물리적 대상의 등록 정보를 보는 방법	112
대상에서 메시지를 제거하는 방법	113
대상을 삭제하는 방법	114
파일 시스템 객체 저장소를 추가하는 방법	115
객체 저장소 등록 정보를 표시하는 방법	118
객체 저장소에 연결하는 방법	118
객체 저장소에 연결 팩토리를 추가하는 방법	119
객체 저장소에 대상을 추가하는 방법	121
대상 객체의 등록 정보를 확인 또는 업데이트하는 방법	122
HelloWorldMessageJNDI 응용 프로그램을 실행하는 방법	124
기본 브로커 이름 및 구성을 사용하는 브로커 인스턴스를 시작하는 방법	136
엔터프라이즈판 시험 사용권으로 브로커 인스턴스를 시작하는 방법	136
플러그인 지속성을 사용하여 명명된 브로커 인스턴스를 시작하는 방법	136
브로커를 클러스터에 연결하는 방법	143
클러스터 내에서 보안 연결을 구성하는 방법	143
기존 클러스터에 새 브로커를 추가하는 방법	144
이미 기존 클러스터의 구성원인 브로커를 다시 시작하는 방법	144

기존 클러스터에서 브로커를 제거하는 방법	145
구성 변경 기록을 백업하는 방법	146
문제가 발생한 경우 마스터 브로커를 복원하는 방법	146
브로커의 로거 구성을 변경하는 방법	148
사용되지 않는 대상 디스크 공간을 확보하는 방법	178
LDAP 서버를 사용하도록 구성 파일을 편집하는 방법	209
SSL 기반 연결 서비스를 설정하는 방법	219
키 쌍을 다시 생성하는 방법	221
브로커에서 SSL 기반 서비스를 활성화하는 방법	222
메트릭 하위 명령을 사용하는 방법	248
로그 파일을 사용하여 메트릭 정보를 보고하는 방법	251
메시지 기반 모니터링을 설정하는 방법	253
JDBC로 액세스할 수 있는 데이터 저장소를 플러그인 하는 방법	298
HTTP 지원을 활성화하는 방법	309
httpjms 연결 서비스를 활성화하는 방법	311
터널 서블릿을 추가하는 방법	314
터널 서블릿에 대한 가상 경로(서블릿 URL)를 구성하는 방법	315
Web Server 시작 시 터널 서블릿을 로드하는 방법	315
서버 액세스 로그를 비활성화하는 방법	316
http 터널 서블릿을 WAR 파일로 배포하는 방법	316
Application Server 7.0 환경에 HTTP 터널 서블릿을 배포하는 방법	317
Application Server의 server.policy 파일을 수정하는 방법	319
HTTPS 지원을 활성화하는 방법	319
httpsjms 연결 서비스를 활성화하는 방법	322
JSSE를 구성하는 방법	324
터널 서블릿을 추가하는 방법	327
터널 서블릿에 대한 가상 경로(서블릿 URL)를 구성하는 방법	328
Web Server 시작 시 터널 서블릿을 로드하는 방법	328
서버 액세스 로그를 비활성화하는 방법	329
HTTPS 터널 서블릿 WAR 파일을 수정하는 방법	329
https 터널 서블릿을 WAR 파일로 배포하는 방법	330
Application Server 7.0 환경에 HTTPS 터널 서블릿을 배포하는 방법	331
Application Server의 server.policy 파일을 수정하는 방법	332
기록된 서비스 오류 이벤트를 보는 방법	336

머리말

이 문서 Sun Java™ System Message Queue 3.5 SP1 *관리 설명서*는 Message Queue 메시징 시스템에서 관리 작업을 수행하는 데 필요한 배경 지식 및 정보를 제공합니다.

머리말에는 다음과 같은 절로 구성되어 있습니다.

- [23페이지의 "대상"](#)
- [24페이지의 "구성"](#)
- [25페이지의 "활자체 규약"](#)
- [28페이지의 "기타 설명서 자원"](#)

대상

이 설명서는 Message Queue 관리 작업을 수행해야 하는 응용 프로그램 개발자와 관리자를 대상으로 합니다.

Message Queue 관리자는 Message Queue 메시징 시스템, 특히 이 시스템의 핵심인 Message Queue 메시지 서버를 설정하고 관리합니다. 이 문서에는 메시징 시스템에 대한 지식이나 이해를 위한 내용이 포함되어 있지 않습니다.

또한 응용 프로그램 개발자가 Message Queue 메시징 시스템의 기능 및 유연성을 십분 활용하기 위해 응용 프로그램을 최적화하는 방법을 보다 자세히 파악하고자 할 때 이 설명서를 사용할 수 있습니다.

구성

이 설명서는 처음부터 끝까지 읽어야 합니다. 다음 표에서는 각 장의 내용을 간단히 설명합니다.

표 1 설명서 내용

장	설명
1장, "개요"	Message Queue 메시징 시스템 및 용어를 상위 개념 수준에서 요약합니다.
2장, "Message Queue 메시징 시스템"	Message Queue 메시징 시스템에 대해, 특히 메시징 서비스를 함께 제공하는 Message Queue 브로커 및 Message Queue 클라이언트 런타임에 대해 중점적으로 설명합니다.
3장, "Message Queue 관리 작업 및 도구"	Message Queue 관리 작업과 도구에 대해 설명하고 관리 용도로 사용하는 명령줄 유틸리티 및 그 공통적인 기능에 대해 소개합니다.
4장, "관리 콘솔 자습서"	Message Queue 메시지 서버의 그래픽 인터페이스인 관리 콘솔을 익힐 수 있는 실습 자습서를 제공합니다.
5장, "브로커 시작 및 구성"	Message Queue 브로커 및 브로커 클러스터를 시작하고 구성하는 방법에 대해 설명합니다.
6장, "브로커 및 응용 프로그램 관리"	Message Queue 브로커 관리 관련 작업(응용 프로그램에 영향을 받지 않음) 및 메시징 응용 프로그램 관리 작업의 수행 방법에 대해 설명합니다.
7장, "관리 대상 객체 관리"	Message Queue 관리 대상 객체 생성 및 관리 관련 작업의 수행 방법에 대해 설명합니다.
8장, "보안 관리"	인증, 권한 부여 및 암호화 관리와 같은 응용 프로그램 관련 보안 작업의 수행 방법에 대해 설명합니다.
9장, "메시지 서비스 분석 및 조정"	메시지 서버 성능을 모니터링하고 분석하는 기술과 메시지 서버를 조정하여 성능을 최적화하는 방법을 설명합니다.
부록 A, "Message Queue 데이터의 위치"	다양한 Message Queue 데이터 범주의 위치에 대해 설명합니다.
부록 B, "플러그 인 지속성 설정"	지속성 기능을 수행하고자 JDBC 호환 데이터베이스를 사용하도록 Message Queue를 설정하는 방법에 대해 설명합니다.
부록 C, "HTTP/HTTPS 지원(엔터프라이즈판)"	메시징 클라이언트와 Message Queue 메시지 서버 사이에 HTTP 연결 서비스를 설정하는 방법에 대해 설명합니다.
부록 D, "브로커를 Windows 서버로 사용"	(Windows 서버로 실행되는) 브로커 설치, 쿼리 및 제거를 위해 Message Queue 서비스 관리 유틸리티(imqsvcadm)를 사용하는 방법에 대해 설명합니다.

표 1 설명서 내용(계속)

장	설명
부록 E, "기술 노트"	이 설명서의 주제와 관련이 있지만 Message Queue 관련 관리에는 포함되지 않은 여러 특별한 기술 노트를 제공합니다.
부록 F, "Message Queue 자원 어댑터"	Message Queue 자원 어댑터를 소개하고, 그 배포 방법과 구성 및 사용 방법을 설명합니다.
부록 G, "선택적 JMS 기능의 Message Queue 구현"	Message Queue 제품이 JMS 사양에 JMS 공급자가 구현할 수 있는 선택 사항으로 나열된 각 항목을 처리하는 방법을 설명합니다.
부록 H, "Message Queue 인터페이스의 안정성"	다양한 Message Queue 인터페이스의 안정성에 대해 설명합니다.
"용어집"	Message Queue 설명서에서 사용하는 용어를 정의합니다.

활자체 규약

이 절에서는 본 설명서에 사용되는 활자체 규약에 관한 정보를 제공합니다.

텍스트 활자체 규약

표 2 문서 활자체 규약

형식	설명
<i>기울임꼴</i>	기울임꼴 텍스트는 위치 표시자를 나타냅니다. 기울임꼴 텍스트로 표시된 부분은 적절한 절 또는 값으로 대체합니다. 기울임꼴 텍스트는 문서 제목, 강조 또는 소개할 단어나 구를 지정할 때도 사용됩니다.
고정 폭	고정 폭 텍스트는 코드의 예, 명령줄에 입력하는 명령, 디렉토리/파일/경로 이름, 오류 메시지 텍스트, 클래스 이름, 메소드 이름(서명의 모든 요소 포함), 패키지 이름, 예약어, URL을 나타냅니다.
[]	대괄호는 명령줄 구문에 선택적으로 사용되는 값을 나타냅니다.
모두 대문자	모두 대문자로 표시된 텍스트는 파일 시스템 유형(GIF, TXT, HTML 등), 환경 변수(IMQ_HOME) 또는 머리글자(Message Queue, JSP)를 나타냅니다.
키+키	동시 키 입력은 더하기 기호와 함께 표시됩니다. Ctrl+A는 두 키를 동시에 누르라는 의미입니다.

표 2 문서 활자체 규약(계속)

형식	설명
키-키	연속적인 키 입력은 하이픈과 함께 표시됩니다. Esc-S는 Esc를 누르고 이를 놓은 다음 S 키를 누르라는 의미입니다.

디렉토리 변수 규칙

Message Queue에서는 세 가지 디렉토리 변수를 사용하며 설정 방법은 플랫폼에 따라 다릅니다. 표 3에서는 이러한 변수를 설명하고 이들이 Solaris™, Windows, Linux 플랫폼에 사용되는 방법을 개괄적으로 설명합니다.

표 3 Message Queue 디렉토리 변수

변수	설명
IMQ_HOME	<p>보통은 다음과 같이 Message Queue 설명서에서 Message Queue 기본 디렉토리(루트 설치 디렉토리)를 참조할 때 사용됩니다.</p> <ul style="list-style-type: none"> • Solaris에는 루트 Message Queue 설치 디렉토리가 없습니다. 따라서 Message Queue 설명서에서 Solaris의 파일 위치를 참조하는 경우에는 IMQ_HOME이 사용되지 않습니다. • Solaris에서 Sun Java System Application Server의 루트 Message Queue 설치 디렉토리는 Application Server 기본 디렉토리 아래에 있는 /imq입니다. • Windows의 경우, 루트 Message Queue 설치 디렉토리는 Message Queue 설치 프로그램에서 설정합니다(기본값은 C:\Program Files\Sun\MessageQueue3). • Windows의 경우 Sun Java System Application Server의 루트 Message Queue 설치 디렉토리는 Application Server 기본 디렉토리 아래의 /imq입니다. • Linux에는 루트 Message Queue 설치 디렉토리가 없습니다. 따라서 Message Queue 설명서에서 Linux의 파일 위치를 참조하는 경우에는 IMQ_HOME이 사용되지 않습니다.

표 3 Message Queue 디렉토리 변수(계속)

변수	설명
IMQ_VARHOME	<p>Message Queue 임시 또는 동적으로 작성된 구성 및 데이터 파일이 저장되는 /var 디렉토리입니다. 모든 디렉토리를 가리키는 환경 변수로 설정할 수 있습니다.</p> <ul style="list-style-type: none"> • Solaris에서 IMQ_VARHOME의 기본값은 /var/imq 디렉토리입니다. • Solaris에서 Sun Java System Application Server 평가판의 IMQ_VARHOME 기본값은 IMQ_HOME/var 디렉토리입니다. • Windows에서 IMQ_VARHOME의 기본값은 IMQ_HOME\var 디렉토리입니다. • Windows에서 Sun Java System Application Server의 IMQ_VARHOME 기본값은 IMQ_HOME\var 디렉토리입니다. • Linux에서 IMQ_VARHOME의 기본값은 /var/opt/imq 디렉토리입니다.
IMQ_JAVAHOME	<p>Message Queue 실행 파일에 필요한 Java™ runtime (JRE)의 위치를 가리키는 환경 변수입니다.</p> <ul style="list-style-type: none"> • Solaris에서 IMQ_JAVAHOME의 기본값은 /usr/j2se/jre 디렉토리이지만, 선택적으로 사용자는 필요한 JRE가 있는 적절한 위치로 값을 설정할 수 있습니다. • Windows에서 IMQ_JAVAHOME의 기본값은 IMQ_HOME\jre이지만, 선택적으로 사용자는 필요한 JRE가 있는 적절한 위치로 값을 설정할 수 있습니다. • Linux에서 Message Queue는 /usr/java/j2sdkVersion 디렉토리 와 /usr/java/j2reVersion 디렉토리를 차례로 조사하여 Java runtime을 찾지만, 선택적으로 사용자는 필요한 JRE가 있는 적절한 IMQ_JAVAHOME 값을 설정할 수 있습니다.

이 설명서에서 IMQ_HOME, IMQ_VARHOME 및 IMQ_JAVAHOME은 플랫폼별 환경 변수 표시나 구문(예: UNIX®의 \$IMQ_HOME) 없이 표시됩니다. 경로 이름에는 일반적으로 UNIX 디렉토리 구분자 표시(/)를 사용합니다.

기타 설명서 자원

Message Queue에는 이 설명서 외에도 추가 설명서 자원이 제공됩니다.

Message Queue 설명서 집합

Message Queue 설명서 집합을 구성하는 문서는 표 4에서 일반적으로 사용되는 순서에 따라 나열되어 있습니다.

표 4 Message Queue 설명서 집합

문서	대상	설명
<i>Message Queue 설치 설명서</i>	개발자와 관리자	Solaris, Linux, Windows 플랫폼에서 Message Queue 소프트웨어를 설치하는 방법을 설명합니다.
<i>Message Queue 릴리스 노트</i>	개발자와 관리자	새로운 기능, 제한, 알려진 버그 및 기술 노트에 관한 설명이 포함되어 있습니다.
<i>Message Queue 관리 설명서</i>	관리자, 개발자에게도 권장	Message Queue 관리 도구를 사용한 관리 작업 수행 시 필요한 배경 및 정보를 제공합니다.
<i>Message Queue Java Client Developer's Guide</i>	개발자	JMS 및 SOAP/JAXM 사양의 Message Queue 구현을 사용하는 Java 클라이언트 프로그램 개발자를 위한 빠른 시작 자습서와 프로그래밍 정보를 제공합니다.
<i>Message Queue C Client Developer's Guide</i>	개발자	Message Queue 메시지 서비스에 대한 C 인터페이스(C-API)를 사용하는 C 클라이언트 프로그램 개발자를 위한 프로그래밍 및 참조 설명서를 제공합니다.

온라인 도움말

Message Queue는 Message Queue 메시지 서비스 관리 작업을 수행하는 명령줄 유틸리티를 포함합니다. 이 유틸리티에 대한 온라인 도움말을 보려면 [100페이지의 "공통 명령줄 옵션"](#)을 참조하십시오.

또한 Message Queue에는 그래픽 사용자 인터페이스(GUI) 관리 도구인 관리 콘솔(imqadmin)이 포함되어 있습니다. 컨텍스트 관련 온라인 도움말은 관리 콘솔에 포함되어 있습니다.

JavaDoc

JavaDoc 형식의 Message Queue Java 클라이언트 API (JMS API 포함) 설명서는 운영 체제에 따라 다른 디렉토리에 제공됩니다(부록 A, "[Message Queue 데이터의 위치](#)" 참조).

이 설명서는 Netscape, Internet Explorer와 같은 모든 HTML 브라우저로 볼 수 있습니다. 표준 JMS API 설명서 및 Message Queue 관리 대상 객체에 대한 Message Queue별 API를 포함하며(*Message Queue Java Client Developer's Guide*의 3장 참조), 이는 메시징 응용 프로그램 개발자에게 도움이 됩니다.

클라이언트 응용 프로그램의 예

샘플 클라이언트 응용 프로그램 코드를 제공하는 여러 응용 프로그램의 예가 운영 체제에 따라 다른 디렉토리에 포함되어 있습니다(부록 A, "[Message Queue 데이터의 위치](#)" 참조).

해당 디렉토리 및 각 하위 디렉토리에 있는 README 파일을 참조하십시오.

JMS (Java Message Service) 사양

다음 위치에서 JMS 사양을 확인할 수 있습니다.

<http://java.sun.com/products/jms/docs.html>

이 사양에는 클라이언트 코드 샘플이 포함되어 있습니다.

관련 타사 웹 사이트 참조

이 문서에 있는 타사 URL에서는 관련 추가 정보를 제공합니다.

주 Sun은 이 문서에 언급된 타사 웹 사이트의 사용 가능성에 대해 책임지지 않습니다. Sun은 그러한 사이트 또는 자원에 있거나 사용 가능한 내용, 광고, 제품 또는 기타 자료에 대하여 보증하지 않으며 책임 또는 의무를 지지 않습니다. Sun은 해당 사이트나 자원을 통해 사용 가능한 내용, 상품 또는 서비스의 사용과 관련해 발생했거나 발생했다고 간주되는 손해나 손실에 대해 책임이나 의무를 지지 않습니다.

개요

이 장에서는 Sun Java™ System Message Queue를 소개하며 관리자와 프로그래머가 관심을 가질 내용으로 구성됩니다.

Sun Java System Message Queue 소개

Message Queue 제품은 분산 응용 프로그램의 안정적인 비동기 메시징을 위한 표준 기반 솔루션입니다. Message Queue는 JMS (Java™ Message Service) 개방형 표준을 구현하는 엔터프라이즈 메시징 시스템이며 JMS 참조 구현 역할을 합니다. 하지만 Message Queue는 엔터프라이즈급 기능을 제공하는 완벽한 기능의 JMS 공급자이기도 합니다.

JMS 사양은 분산 환경에서 Java 언어 응용 프로그램으로 메시지를 작성하고, 전송하고, 수신하며 읽을 수 있는 공통된 방식을 제공하는 메시징 의미와 동작 및 응용 프로그램 프로그래밍 인터페이스(API)의 집합을 설명합니다(38페이지의 "[JMS 프로그래밍 모델](#)" 참조). Java 메시징 응용 프로그램 지원 외에 Message Queue는 Message Queue 서비스에 C 언어 인터페이스(Message Queue C-API)도 제공합니다.

Sun Java System Message Queue 소프트웨어를 사용하면 서로 다른 플랫폼과 운영 체제에서 실행되는 프로세스들이 공통의 Message Queue 메시지 서비스(36페이지의 "[메시지 서비스 구조](#)" 참조)에 연결하여 정보를 보내고 받을 수 있습니다. 응용 프로그램 개발자들은 네트워크를 통한 응용 프로그램의 안정적인 통신 방식과 관련된 하위 수준의 세부 사항보다는 응용 프로그램의 비즈니스 논리에 좀 더 집중할 수 있습니다.

Message Queue에는 JMS 사양의 최소 요구 사항을 능가하는 기능이 있습니다. 다음은 그러한 기능의 예입니다.

중앙 집중식 관리. Message Queue 서비스 관리와 대상, 트랜잭션, 영구 가입, 보안 등과 응용 프로그램 종속 항목의 관리를 위한 명령줄 도구와 GUI 도구를 제공합니다. Message Queue는 Message Queue 서비스의 원격 모니터링도 지원합니다.

확장 가능한 메시지 서비스. 탠덤으로 작동하는 많은 Message Queue 메시지 서버 구성 요소(브로커), 즉 멀티 브로커 클러스터 간에 로드 균형을 조정하여 더 많은 수의 Message Queue 클라이언트(구성 요소 또는 응용 프로그램) 서비스를 제공할 수 있습니다.

클라이언트 연결 페일오버. Message Queue 메시지 서버에 대한 실패한 클라이언트 연결을 자동으로 복원합니다.

조정 가능한 성능. 전달 안정성을 더 낮출 수 있는 경우 Message Queue 서비스의 성능을 높일 수 있습니다.

다중 전송. TCP와 HTTP 등의 다양한 전송 방식을 통해 그리고 보안(SSL) 연결을 사용하여 Message Queue 메시지 서버와 통신이 가능한 Message Queue 클라이언트의 기능을 지원합니다.

JNDI 지원. JNDI (Java Naming and Directory Interface)를 객체 저장소 및 사용자 저장소로 구현하는 파일 기반 및 LDAP 방식을 모두 지원합니다.

SOAP 메시징 지원. JMS 메시징을 통해 SOAP 메시지, 즉 SOAP (Simple Object Access Protocol) 사양에 일치하는 메시지의 생성 및 전달을 지원합니다. SOAP을 사용하면 분산 환경에서 피어 간에 구조화된 XML 데이터를 교환할 수 있습니다. 자세한 내용은 *Message Queue Java Client Developer's Guide*를 참조하십시오.

JMS 호환성에 대한 정보는 [부록 G, "선택적 JMS 기능의 Message Queue 구현"](#)을 참조하십시오.

제품 판

Sun Java System Message Queue는 플랫폼판과 엔터프라이즈판의 두 버전으로 제공되며, 이 버전들은 다음과 같이 기능과 사용권 용량이 서로 다릅니다(Message Queue를 다른 판으로 업그레이드하는 방법은 *Message Queue 설치 설명서* 참조).

플랫폼판

이 버전은 Sun 웹 사이트에서 무료로 다운로드할 수 있으며 Sun Java System Application Server 플랫폼과 함께 제공됩니다. 플랫폼판의 경우 Message Queue 메시지 서비스가 지원하는 클라이언트 연결의 수에 제한이 없습니다. 여기에는 다음과 같은 두 가지 사용권이 제공됩니다.

- **기본 사용권** 이 사용권은 기본적인 JMS 지원(전체 JMS 공급자)을 제공하지만 로드 균형 조정(멀티 브로커 메시지 서비스), HTTP/HTTPS 연결, 보안 연결 서비스, 확장 가능한 연결 기능, 클라이언트 연결 페일오버, 다중 사용자로의 대기열 전달, 원격 메시지 기반 모니터링, C-API 지원 등과 같은 엔터프라이즈 기능은 포함하지 *않습니다*. 사용권은 무기한 사용할 수 있으며 따라서 생산 요구가 적은 환경에서 사용됩니다.
- **90일 시험 엔터프라이즈 사용권** 이 사용권에는 기본 사용권에 포함되지 않은 모든 엔터프라이즈 기능(멀티 브로커 메시지 서비스 지원, HTTP/HTTPS 연결, 보안 연결 서비스, 확장 가능한 연결 기능, 클라이언트 연결 페일오버, 다중 사용자로의 대기열 전달, 원격 메시지 기반 모니터링, C-API 지원 등)이 포함됩니다. 하지만 소프트웨어는 사용권에 90일의 제한을 두기 때문에 해당 제품의 엔터프라이즈판에서 사용 가능한 엔터프라이즈 기능 평가에 적합합니다("엔터프라이즈판" 참조).

주 90일 시험 사용권은 136페이지의 "엔터프라이즈판 시험 사용권으로 브로커 인스턴스를 시작하는 방법"에 설명된 대로 Message Queue 메시지 서버(Message Queue 브로커 인스턴스)를 시작하여 활성화할 수 있습니다.

엔터프라이즈판

이 판은 생산 환경에서 메시징 응용 프로그램을 배포 및 실행할 때 사용됩니다. 여기에는 멀티 브로커 메시지 서비스 지원, HTTP/HTTPS 연결, 보안 연결 서비스, 확장 가능한 연결 기능, 클라이언트 연결 페일오버, 다중 사용자로의 대기열 전달, 원격 메시지 기반 모니터링, C-API 등에 대한 지원이 포함됩니다. 엔터프라이즈판은 메시징 응용 프로그램과 구성 요소의 개발, 디버깅, 로드 테스트에도 사용할 수 있습니다. 엔터프라이즈판에는 멀티 브로커 메시지 서비스의 브로커 수는 제한하지 않지만 사용되는 CPU의 수를 기반으로 하는 무기한 사용권이 있습니다.

엔터프라이즈 메시징 시스템

엔터프라이즈 메시징 시스템에서 독립 분산 응용 프로그램이나 응용 프로그램 구성 요소는 메시지를 통해 상호 작용할 수 있습니다. 동일한 호스트나 네트워크에 있거나 또는 인터넷을 통해 느슨하게 연결되어 있는 이 구성 요소들은 메시징을 사용하여 데이터를 전달하고 각자의 기능을 조정합니다.

엔터프라이즈 메시징 시스템의 요구 사항

일반적으로 엔터프라이즈 응용 프로그램 시스템은 24시간 미션 크리티컬 작업으로 무수히 많은 메시지를 교환하는 많은 수의 분산 구성 요소로 구성됩니다. 그러한 시스템을 지원하기 위해 엔터프라이즈 메시징 시스템은 일반적으로 다음 요구 사항을 만족시켜야 합니다.

안정적인 전달. 구성 요소 간에 전달되는 메시지는 네트워크나 시스템 오류로 인해 손실되지 않아야 합니다. 즉 시스템은 메시지가 성공적으로 전달되도록 보장할 수 있어야 합니다.

비동기식 전달. 많은 수의 구성 요소가 동시에 메시지를 교환하고 고용량의 처리량을 지원하기 위해서는 메시지 발신이 사용자의 즉시 수신 가능 여부에 따라 결정되어서는 안 됩니다. 사용자가 작업 중이거나 오프라인 상태인 경우, 시스템은 사용자가 온라인이 될 때 메시지 발신 및 수신이 가능하도록 해야 합니다. 이를 비동기식 메시지 전달이라 하며, 일반적으로 저장 및 전달(store-and-forward) 메시징이라고 부릅니다.

보안. 메시징 시스템은 사용자 인증, 메시지 및 자원에 대한 인증된 액세스, 회선을 통한 암호화와 같은 기본 보안 기능을 지원해야 합니다.

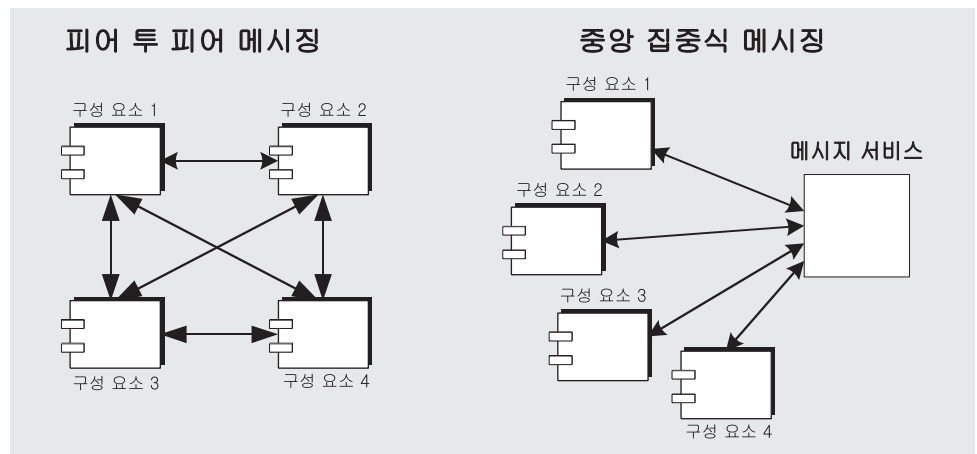
확장성. 메시징 시스템은 성능이나 메시지 처리량이 실제로 저하되지 않으면서 로드 증가(사용자 수 및 메시지 수 증가)를 수용할 수 있어야 합니다. 업무와 응용 프로그램이 늘어나면 이는 매우 중요한 요구 사항이 됩니다.

관리성. 메시징 시스템은 메시지 전달을 모니터링 및 관리하고 시스템 자원을 최적화하는 도구를 제공해야 합니다. 이러한 도구는 안정성, 보안 및 성능의 측정과 관리를 지원 합니다.

중앙 집중식 메시징과 피어 투 피어(Peer to Peer) 메시징 비교

그림 1-1에서 확인할 수 있듯이 기존의 피어 투 피어(Peer to Peer) 메시징 시스템으로는 엔터프라이즈 메시징 시스템의 요구 사항을 만족시키기 어렵습니다.

그림 1-1 중앙 집중식 메시징과 피어 투 피어(Peer to Peer) 메시징 비교



그러한 시스템에서 모든 메시징 구성 요소는 다른 모든 구성 요소와의 연결을 유지 관리 합니다. 이 연결을 통해 빠르고 안정적이며 보안이 유지된 상태로 전달을 할 수 하지만, 안정성과 보안을 지원하는 코드가 각 구성 요소마다 존재해야 합니다. 시스템에 구성 요소가 추가되면서 연결 수는 기하급수적으로 늘어납니다. 그 결과 비동기식 메시지 전달 및 확장성을 구현하기 어려워집니다. 중앙 집중식 관리에도 문제가 있습니다.

그림 1-1에서도 확인할 수 있듯이 권장되는 엔터프라이즈 메시징 방식은 중앙 집중식 메시징 시스템입니다. 이 방식에서 각 메시징 구성 요소는 단일 중앙 메시지 서비스와의 연결을 유지 관리합니다. 이 메시지 서비스는 구성 요소 간의 메시지 라우팅 및 전달을 수행하며, 안정적인 전달 및 보안을 책임집니다.

구성 요소는 잘 정의된 프로그래밍 인터페이스를 통해 메시지 서비스와 상호 작용합니다. 구성 요소가 시스템에 추가되면 일차적으로 연결 수만 증가되므로, 메시지 서비스를 확장하여 시스템을 보다 쉽게 확장할 수 있습니다. 또한 중앙 메시지 서비스는 중앙 집중식 시스템 관리를 지원합니다.

메시징 시스템 개념

엔터프라이즈 메시징 시스템에는 몇 가지 기본 개념이 존재합니다. 아래에서 설명하는 메시지, 메시지 서비스 구조 및 메시지 전달 모델도 그 기본 개념에 포함됩니다.

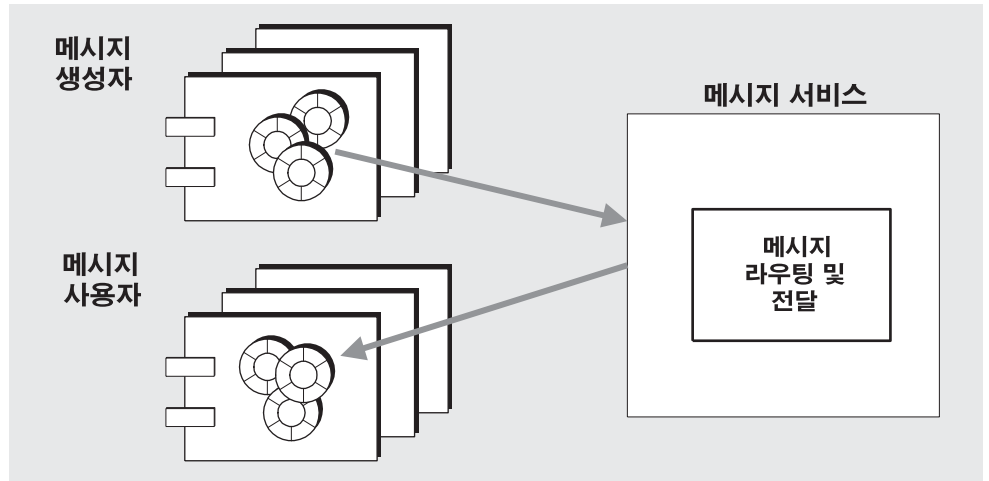
메시지

메시지는 일정 형식의 데이터(메시지 본문) 그리고 대상, 수명 및 그 밖의 메시징 시스템이 결정하는 특성 등과 같이 메시지의 특성이나 등록 정보를 설명하는 메타 데이터(메시지 헤더)로 구성됩니다.

메시지 서비스 구조

메시징 시스템의 기본 구조는 [그림 1-2](#)와 같습니다. 공통 메시지 서비스를 통해 메시지를 교환하는 메시지 생성자와 메시지 사용자로 구성됩니다. 메시지 생성자와 사용자는 그 수의 제한 없이 동일한 메시징 구성 요소(또는 응용 프로그램)에 위치할 수 있습니다. 메시지 생성자가 메시지 서비스에 메시지를 보냅니다. 메시지 서비스는 메시지 라우팅 및 전달 구성 요소를 사용하는 방식으로 해당 메시지에 대해 인터레스트를 등록한 하나 이상의 메시지 사용자에게 메시지를 전달합니다. 메시지 라우팅 및 전달 구성 요소는 관련된 모든 사용자에게 메시지가 전달되도록 보장하는 역할을 합니다.

그림 1-2 메시지 서비스 구조



메시지 전달 모델

생성자와 사용자의 관계는 일대일, 일대다, 다대다 등 여러 종류가 있습니다. 예를 들어 다음과 같이 메시지를 전달할 수 있습니다.

- 단일 생성자가 단일 사용자에게
- 단일 생성자가 여러 사용자에게
- 여러 생성자가 단일 사용자에게
- 여러 생성자가 여러 사용자에게

주로 이 관계들은 *지점간(point-to-point)* 및 *게시/가입* 메시징의 2가지 메시지 전달 모델로 축소됩니다. 지점간 전달 모델은 특정 생성자가 보내고 특정 사용자가 받는 메시지에 중점을 둡니다. 게시/가입 전달 모델은 여러 생성자가 보내고 여러 사용자가 받는 메시지에 중점을 둡니다. 이 메시지 전달 모델은 중복될 수 있습니다.

지금까지 메시징 시스템은 이 2가지 메시지 전달 모델이 다양하게 결합된 형태를 지원했습니다. JMS (Java Message Service) 사양은 Java 프로그래밍용 API를 통한 메시징을 위한 표준 의미를 작성합니다. 이 API는 지점간 및 게시/가입 메시지 전달 모델을 모두 지원합니다(44페이지의 "프로그래밍 도메인" 참조).

JMS 사양

JMS 사양은 메시징에 적용되는 프로그래밍 모델, 메시지 구조, API 등을 비롯한 규칙과 의미의 집합을 규정합니다. *Message Queue*는 JMS 구현을 제공하기 때문에 JMS 개념은 *Message Queue* 메시징 시스템의 작동 방식을 이해하는 토대가 됩니다. 이 소개 부분에서는 이 책의 나머지 장을 이해하는 데 필요한 개념 및 용어를 설명합니다.

JMS 메시지 구조

JMS 메시지는 헤더, 등록 정보 및 본문의 3가지 부분으로 구성됩니다.

헤더. 헤더는 메시지의 JMS 특성, 즉 대상, 지속성 여부, 수명 및 우선 순위를 지정합니다. 이러한 특징은 메시징 시스템의 메시지 전달 방식에 영향을 미칩니다.

등록 정보. 등록 정보(헤더의 확장으로 간주할 수 있음)는 선택 사항이며, 응용 프로그램이 다양한 선택 기준에 따라 메시지를 필터링할 때 사용 가능한 값을 제공합니다. 등록 정보는 선택 사항입니다.

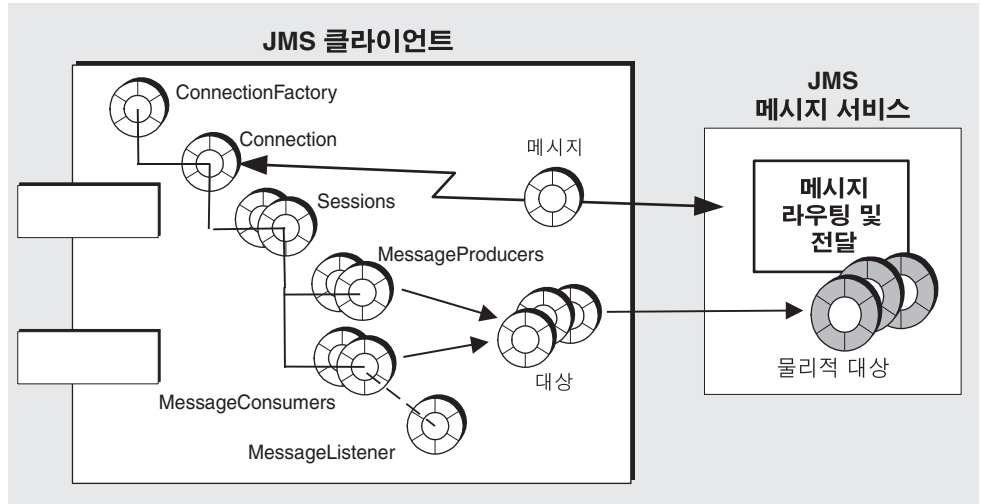
메시지 본문. 메시지 본문에는 교환할 실제 데이터가 포함됩니다. JMS는 6가지 본문 유형을 지원합니다.

JMS 프로그래밍 모델

JMS 프로그래밍 모델에서 JMS 클라이언트(구성 요소 또는 응용 프로그램)는 JMS 메시지 서비스를 통해 메시지를 교환합니다. 메시지 생성자는 메시지 서비스로 메시지를 보내며, 메시지 사용자는 메시지 서비스로부터 메시지를 받습니다. 이러한 메시징 작업은 JMS API (Application Programming Interface)를 구현하는 객체 집합(JMS 공급자가 제공)을 사용하여 수행됩니다.

이 절에서는 JMS API를 구현하고 JMS 클라이언트의 메시지 전달 설정에 사용되는 객체를 소개합니다(자세한 정보는 *Message Queue Java Client Developer's Guide* 참조). [그림 1-3](#)은 메시지 전달 프로그램에 사용되는 JMS 객체들입니다.

그림 1-3 JMS 프로그래밍 객체



JMS 프로그래밍 모델에서 JMS 클라이언트는 ConnectionFactory 객체를 사용하여 메시지 서비스와 메시지를 송수신할 연결을 만듭니다. Connection은 클라이언트와 메시지 서비스 간의 활성 연결입니다. 연결되면 통신 자원 할당 및 클라이언트 인증이 이루어집니다. 이는 비교적 중량급 객체이며, 대부분의 클라이언트는 단일 연결을 사용하여 모든 메시지를 수행합니다.

연결은 세션을 생성할 때 사용합니다. Session은 메시지 생성 및 사용을 위한 단일 스레드 컨텍스트입니다. 메시지를 보내고 받는 메시지 생성자 및 사용자를 생성할 때 사용하며, 전달할 메시지의 일련 순서를 정의합니다. 세션은 여러 확인 옵션이나 트랜잭션을 통해 안정적인 전달을 지원합니다.

클라이언트는 MessageProducer를 사용하여 API에서 대상 아이디 객체로 표시되는 지정된 물리적 대상으로 메시지를 보냅니다. 메시지 생성자는 물리적 대상으로 보낼 모든 메시지에 적용되는 기본 전달 모드(지속성 메시지 및 비지속성 메시지), 우선 순위 및 수명 값을 지정합니다.

그와 비슷하게 클라이언트는 MessageConsumer를 사용하여 API에서 대상 객체로 표시되는 지정된 물리적 대상으로부터 메시지를 받습니다. 메시지 사용자는 메시지 선택기를 사용하여 메시지 서비스에서 선택 기준과 일치하는 메시지 사용자에게 해당 메시지만을 전달하도록 할 수 있습니다.

메시지 사용자는 동기식 또는 비동기식 메시지 사용을 지원할 수 있습니다. 사용자에게 `MessageListener`를 등록하면 비동기식 사용이 이루어집니다. 세션 스레드가 `MessageListener` 객체의 `onMessage()` 메소드를 호출하면 클라이언트가 해당 메시지를 사용합니다.

JMS 관리 대상 객체

JMS 사양은 공급자별 구성 정보를 캡슐화하는 *관리 대상 객체*를 지정하여 공급자 독립 클라이언트를 용이하게 합니다.

38페이지의 "[JMS 프로그래밍 모델](#)"에서 설명하는 객체 중 2가지는 JMS 공급자의 JMS 메시지 서비스 구현 방식에 대한 것입니다. 연결 팩토리 객체는 공급자가 메시지 전달 시 사용하는 기본 프로토콜 및 메커니즘에 따라 다르며, 대상 객체는 공급자가 사용하는 물리적 대상의 특정 이름 지정 규약 및 기능에 따라 다릅니다.

일반적으로 이 공급자별 특성 때문에 JMS 클라이언트 코드는 특정 JMS 구현에 종속됩니다. 그러나 JMS 사양에서는 표준화되고 특정 공급자에 국한되지 않는 방식으로 액세스할 수 있도록 공급자별 구현 및 구성 정보를 연결 팩토리 및 대상 객체로 캡슐화해야 합니다.

관리 대상 객체는 관리자가 작성 및 구성하고 이름 서비스에서 저장하며 클라이언트가 표준 JNDI (Java Naming and Directory Service) 조회 코드를 사용하여 액세스합니다. 이렇게 관리 대상 객체를 사용하면 클라이언트 코드는 공급자 독립성을 갖게 됩니다.

연결 팩토리과 대상이라는 두 가지 관리 객체 유형은 공급자별 정보를 캡슐화하지만 클라이언트 내에서 그 용도는 매우 다릅니다. 연결 팩토리는 메시지 서버에 연결하는 데 사용되고 대상 객체는 물리적 대상을 식별하는 데 사용됩니다.

JMS/J2EE 프로그래밍: Message-Driven Bean

38페이지의 "[JMS 프로그래밍 모델](#)"에서 소개한 일반적인 JMS 클라이언트 프로그래밍 모델 외에도 Java 2 Platform, Enterprise Edition (J2EE 플랫폼) 응용 프로그램 컨텍스트에서 사용하는, 보다 특수화된 JMS 응용 프로그램이 있습니다. 이 특수화된 JMS 클라이언트를 *Message-Driven Bean*이라고 부르며, EJB 2.0 사양 (<http://java.sun.com/products/ejb/docs.html>)에 지정된 EJB (Enterprise JavaBeans) 구성 요소 중 하나입니다.

Message-Driven Bean이 필요한 이유는 다른 EJB 구성 요소(Session Bean과 Entity Bean)들이 동기식 호출만 가능하기 때문입니다. 이 EJB 구성 요소들은 표준 EJB 인터페이스를 통해서만 액세스할 수 있으므로 비동기식으로 메시지를 수신할 메커니즘이 없습니다.

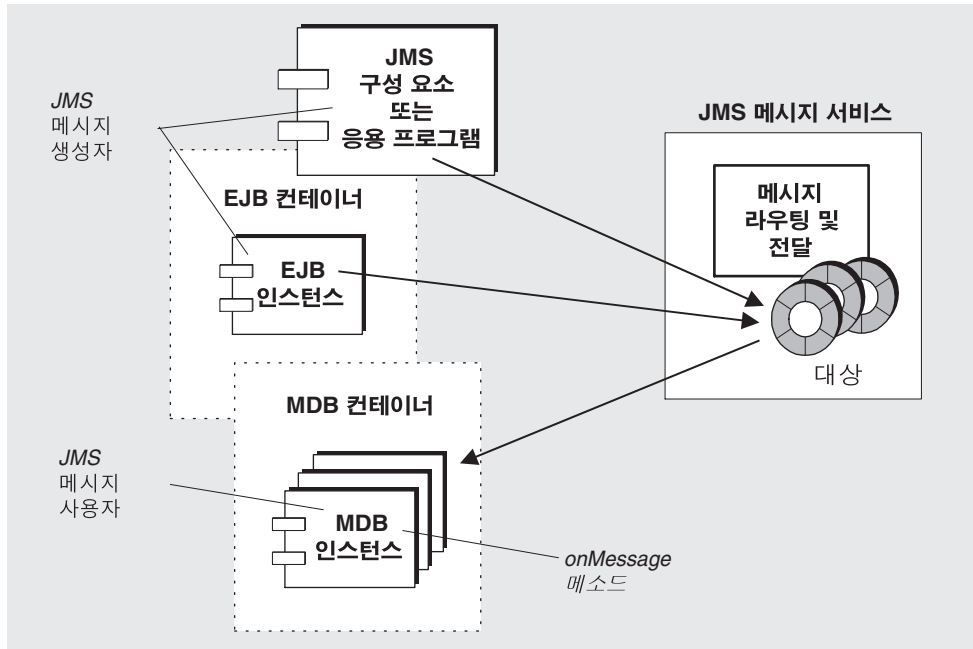
그러나 비동기식 메시징은 많은 엔터프라이즈 응용 프로그램에서 필요합니다. 그러한 응용 프로그램 중 대부분에서는 서버측 구성 요소가 서버 자원을 독점하지 않으면서 상호 통신하고 응답할 수 있어야 합니다. 즉 메시지 생성자와 밀접하게 연결되지 않으면서 메시지 수신 및 사용이 가능한 EJB 구성 요소가 필요합니다. 이 기능은 서버측 구성 요소가 응용 프로그램 이벤트에 응답해야 하는 모든 응용 프로그램에서 필요합니다. 또한 엔터프라이즈 응용 프로그램에서도 로드 증가에 따라 이 기능이 확장되어야 합니다.

Message-Driven Bean

MDB (Message-Driven Bean)는 특정 EJB 컨테이너(지원하는 구성 요소에 대해 분산 서비스를 제공하는 소프트웨어 환경)가 지원하는 특정 EJB 구성 요소입니다.

Message-Driven Bean. MDB는 JMS MessageListener 인터페이스를 구현하는 JMS 메시지 사용자입니다. onMessage 메소드(MDB 개발자가 작성)는 MDB 컨테이너가 메시지를 수신할 때 호출됩니다. 이 onMessage() 메소드는 표준 MessageListener 객체의 onMessage() 메소드처럼 메시지를 사용합니다. 다른 EJB 구성 요소에서처럼 MDB에 대해 메소드를 원격 호출하지 않으므로 MDB와 관련된 홈 또는 원격 인터페이스는 없습니다. MDB는 단일 대상으로부터의 메시지를 사용할 수 있습니다. [그림 1-4](#)에서 확인할 수 있듯이 독립형 JMS 응용 프로그램, JMS 구성 요소, EJB 구성 요소 또는 웹 구성 요소에서 메시지를 생성할 수 있습니다.

그림 1-4 MDB와의 메시지



MDB 컨테이너. MDB 인스턴스를 생성하고 비동기식으로 메시지를 사용하도록 설정하는 역할을 하는 특수화된 EJB 컨테이너가 MDB를 지원합니다. 여기에는 메시지 서비스와의 연결 설정(인증 포함), 지정된 대상과의 세션 풀 생성, 세션 풀 및 관련 MDB 인스턴스 사이에서의 수신 메시지 배포 관리가 포함됩니다. 이 컨테이너는 MDB 인스턴스의 라이프사이클을 제어하므로 MDB 인스턴스 풀이 받는 메시지 로드를 수용할 수 있도록 관리합니다.

MDB와 관련 있는 배포 설명자는 컨테이너가 메시지 사용 설정 시 사용하는 관리 대상 객체, 즉 연결 팩토리와 대상의 JNDI 조회 이름을 지정합니다. 또한 배치 설명자는 배치 도구가 컨테이너 구성 시 사용할 수 있는 다른 정보를 포함할 수 있습니다. 이 컨테이너 각각은 단일 MDB로만 이루어진 인스턴스를 지원합니다.

J2EE Application Server 지원

J2EE 구조(<http://java.sun.com/j2ee/download.html#platformspec>)의 J2EE 플랫폼 사양 참조)에서 EJB 컨테이너는 J2EE Application Server가 호스팅합니다. Application Server는 트랜잭션 관리자, 지속성 관리자, 이름 서비스, JMS 공급자(메시징 및 MDB의 경우) 등과 같은 다양한 컨테이너가 필요로 하는 자원을 제공합니다.

Sun Java System Application Server에서 JMS 메시징 자원은 Sun Java System Message Queue가 제공합니다.

- Sun Java System Application Server 7.0의 경우 Message Queue 메시징 시스템이 Application Server에 원시 JMS 공급자로 통합되어 있습니다.
- Sun J2EE 1.4 Application Server의 경우 Message Queue가 Application Server에 내장 JMS 자원 어댑터로 플러그인되어 있습니다(부록 F, "Message Queue 자원 어댑터" 참조).
- Application Server의 향후 릴리스에서는 Message Queue가 표준 자원 어댑터 배포 및 구성 방법을 사용하는 Application Server에 플러그인됩니다.

JMS 메시징 문제

이 절에서는 Message Queue 메시지 서비스의 관리에 영향을 미치는 다양한 JMS 프로그래밍 문제를 설명합니다. Message Queue 관리자가 필요로 하는 개념 및 용어에 중점을 둡니다.

JMS 공급자 독립성

JMS는 다른 JMS 공급자에게 이식 가능한 클라이언트 응용 프로그램 개발을 지원하도록 관리 대상 객체(40페이지의 "JMS 관리 대상 객체" 참조)의 사용을 지정합니다. 관리 대상 객체를 사용하면 JMS 클라이언트는 공급자별 객체를 조회하고 참조할 때 논리적 이름을 사용할 수 있습니다. 그렇게 되면 클라이언트 코드는 공급자가 사용하는 특정 이름 또는 주소 지정 구문이나 구성 가능한 등록 정보를 알아 둘 필요가 없습니다. 따라서 코드 공급자 독립성을 갖게 됩니다.

관리 대상 객체는 Message Queue 관리자가 작성 및 구성하는 Message Queue 시스템 객체입니다. 이 객체는 JNDI 디렉토리 서비스에 위치하며, JMS 클라이언트는 JNDI 조회를 사용하여 이 객체에 액세스합니다.

또한 Message Queue 관리 대상 객체는 JNDI 디렉토리 서비스에서 조회하기보다는 클라이언트가 인스턴스화할 수 있습니다. 이 경우 응용 프로그램 개발자가 공급자별 API를 사용해야 하는 단점이 있습니다. 또한 이는 Message Queue 관리자가 Message Queue 메시지 서버를 성공적으로 제어하고 관리하는 능력을 저하시킵니다.

관리 대상 객체에 대한 자세한 내용은 [89페이지의 "Message Queue 관리 대상 객체"](#)를 참조하십시오.

프로그래밍 도메인

JMS는 서로 다른 2가지 메시지 전달 모델인 지점간 모델과 게시/가입 모델을 지원합니다.

지점간(대기열 대상). 메시지는 생성자로부터 사용자에게 전달됩니다. 이 전달 모델에서 대상은 *대기열*입니다. 메시지는 먼저 대기열 대상으로 전달된 다음, 대기열의 전달 정책 ([77페이지의 "대기열 대상"](#) 참조)에 따라 대기열로부터 해당 대기열에 등록된 사용자 중 하나에게 한 번에 하나씩 전달됩니다. 생성자 수의 제한 없이 생성자는 대기열 대상에게 메시지를 보낼 수 있으며, 각 메시지는 반드시 *한* 사용자에게만 전달되어 성공적으로 사용됩니다. 대기열 대상에 등록된 사용자가 없는 경우, 대기열은 받은 메시지를 보관했다가 사용자가 대기열에 등록하면 메시지를 전달합니다.

게시/가입(주제 대상). 단일 생성자가 사용자 수의 제한 없이 메시지를 전달합니다. 이 전달 모델에서 대상은 *주제*입니다. 메시지는 먼저 주제 대상으로 전달된 다음, 해당 주제에 *가입한 모든* 활성 사용자에게 전달됩니다. 생성자 수의 제한 없이 생성자는 어떤 주제 대상으로 메시지를 보낼 수 있으며, 각 메시지는 가입한 사용자 수의 제한 없이 전달될 수 있습니다. 또한 주제 대상은 *영구 가입*의 개념을 지원합니다. 영구 가입은 주제 대상에 등록되었지만 메시지가 전달되는 시점에 비활성화될 수 있는 사용자를 의미합니다. 나중에 활성화된 사용자는 메시지를 수신합니다. 주제 대상에 대해 등록된 사용자가 없는 경우, 해당 주제는 비활성 사용자에게 대해 영구 가입이 없는 한 받은 메시지를 보관하지 않습니다.

이 2가지 메시지 전달 모델은 [표 1-1](#)에서 확인할 수 있듯이 각각 다른 프로그래밍 도메인을 나타내고 의미상 약간 차이가 있는 서로 다른 API 객체를 사용하여 처리됩니다.

표 1-1 JMS 프로그래밍 객체

기본 유형 (통합 도메인)	지점간 도메인	게시/가입 도메인
Destination (대기열 또는 주제) ¹	Queue	Topic
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
Connection	QueueConnection	TopicConnection
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver	TopicSubscriber

1. 프로그래밍 방식에 따라 특정 대상 유형을 지정할 수 있습니다.

표 1-1의 첫 번째 열에 표시된 통합 도메인 객체를 사용하여 지점간 및 게시/가입 메시징을 모두 프로그래밍할 수 있습니다. 이 방법이 권장됩니다. 이전 JMS 1.02b 사양에 따라 지점간 도메인 객체를 사용하여 지점간 메시징을 프로그래밍하고 게시/가입 도메인 객체를 사용하여 게시/가입 메시징을 프로그래밍할 수 있습니다.

클라이언트 식별자

JMS 공급자는 JMS 클라이언트와 메시지 서비스 간의 연결에 해당 메시지 서비스가 클라이언트를 대신하여 유지 관리하는 상태 정보를 연관시키는 *클라이언트 식별자* 개념을 지원해야 합니다. 정의에 따라 클라이언트 식별자는 고유하며 한 번에 한 명의 사용자에게만 적용됩니다. 각 영구 가입이 한 명의 사용자에게만 적용되도록 클라이언트 식별자는 영구 가입 이름(44페이지의 "게시/가입(주제 대상)" 참조)과 함께 사용됩니다.

JMS 사양에서는 클라이언트가 API 메소드 호출을 통해 클라이언트 식별자를 설정할 수 있지만, 관리자가 연결 팩토리 관리 대상 객체(40페이지의 "JMS 관리 대상 객체" 참조)를 사용하여 설정하는 것이 좋습니다. 그러나 연결 팩토리에 고정된 경우, 각 사용자는 개별 연결 팩토리마다 고유 아이디를 가져야 합니다.

Message Queue에서는 ConnectionFactory 객체에서 구성 가능한 특수 변수 대체 구문을 사용하여 클라이언트 식별자가 연결 팩토리 및 사용자 고유성을 모두 가질 수 있습니다. 이 방법을 사용할 경우 영구 가입을 작성하는 여러 사용자가 이름 지정 충돌이나 보안 손실에 대한 염려 없이 단일 ConnectionFactory 객체를 사용할 수 있습니다. 따라서 사용자의 영구 가입 시 실수로 지워지거나 다른 사용자의 잘못된 클라이언트 식별자 설정 때문에 사용할 수 없게 되는 상황을 막을 수 있습니다.

이 Message Queue 기능 사용 방법에 대한 자세한 내용은 *Message Queue Java Client Developer's Guide* 중 연결 팩토리 속성 부분을 참조하십시오.

어떤 경우라도 영구 가입을 생성하려면 클라이언트가 JMS API를 사용하여 프로그래밍 방식으로 클라이언트 식별자를 설정하거나 클라이언트가 사용하는 ConnectionFactory 객체에 클라이언트 식별자를 구성해야 합니다.

안정적인 메시징

JMS는 2가지 전달 모드를 정의합니다.

지속성 메시지. 이 메시지는 단 한 차례 전달 및 성공적인 사용이 보장됩니다. 이 메시지에서는 안정성이 중요합니다.

비지속성 메시지. 이 메시지는 최대 한 차례 전달이 보장됩니다. 이러한 메시지의 경우 안정성은 중요한 사항이 아닙니다.

지속성 메시지의 경우 안정성 보장에는 2가지 측면이 있습니다. 한 가지는 메시지 서비스에서 주고 받는 메시지가 성공적으로 전달되도록 하는 것입니다. 다른 하나는 메시지 서비스가 지속성 메시지를 사용자에게 전달하기 전에 그 메시지를 잃지 않도록 하는 것입니다.

확인/트랜잭션

안정적인 메시징은 대상을 오가는 지속성 메시지가 성공적으로 전달될 수 있는지 여부에 따라 결정됩니다. 이는 Message Queue 세션이 지원하는 2가지 일반 메커니즘인 확인 또는 트랜잭션 중 하나를 사용하여 실현할 수 있습니다. 트랜잭션의 경우, 로컬 트랜잭션 또는 분산 트랜잭션 관리자가 제어하는 분산 트랜잭션이 있습니다.

확인(Acknowledgement)

확인을 사용하여 안정적으로 전달되도록 세션을 구성할 수 있습니다.

생성자의 경우, 이는 생성자의 `send()` 메소드가 반환되기 전에 메시지 서비스가 대상에게 지속성 메시지의 전달을 확인하는 것을 의미합니다. 사용자의 경우, 이는 메시지 서비스가 대상으로부터 해당 메시지를 삭제하기 전에 클라이언트가 대상에게서 지속성 메시지의 전달 및 사용을 확인하는 것을 의미합니다.

로컬 트랜잭션

또한 세션을 *트랜잭션*된 것으로 구성할 수 있는데, 이 경우 하나 이상의 메시지 생성 및/또는 사용을 *트랜잭션*이라는 기본 단위로 분류할 수 있습니다. JMS API는 트랜잭션을 시작, 완결 또는 롤백하는 메소드를 제공합니다.

트랜잭션 내부에서 메시지를 생성하거나 사용하면 브로커는 다양한 발신 및 수신을 추적하고, 클라이언트가 트랜잭션을 완결하도록 호출한 경우에만 작업을 완료합니다. 트랜잭션 내부에서 특정 발신 또는 수신 작업이 실패할 경우 예외가 발생합니다. 클라이언트 코드는 예외를 무시하거나 작업을 다시 시도하거나 전체 트랜잭션을 롤백하는 방법으로 예외를 처리할 수 있습니다. 트랜잭션이 완결되면 성공적인 작업이 모두 완료됩니다. 트랜잭션이 롤백되면 성공적인 작업이 모두 취소됩니다.

로컬 트랜잭션의 범위는 항상 단일 세션입니다. 즉 단일 세션 컨텍스트에서 수행되는 하나 이상의 생성자 또는 사용자 작업을 묶어 단일 로컬 트랜잭션으로 분류할 수 있습니다.

트랜잭션 범위가 단일 세션에 국한되므로 메시지 생성과 사용을 모두 총괄하는 중단간 트랜잭션은 만들 수 없습니다. (즉, 대상으로 메시지를 전달하는 것과 나중에 메시지를 클라이언트로 전달하는 것을 하나의 트랜잭션으로 분류할 수 없습니다.)

분산 트랜잭션

또한 Message Queue는 *분산* 트랜잭션을 지원합니다. 즉 메시지 생성 및 사용은 데이터베이스 시스템과 같은 다른 자원 관리자가 관련된 작업들을 포함하는 더 크고 분산된 트랜잭션의 일부가 될 수 있습니다. 분산 트랜잭션의 경우, 분산 트랜잭션 관리자는 JTA (Java Transaction API)인 XA Resource API 사양에 정의된 2단계 완결 프로토콜을 사용하여 여러 자원 관리자(메시지 서비스, 데이터베이스 관리자 등)가 수행하는 작업을 추적하고 관리합니다. Java에서 자원 관리자 및 분산 트랜잭션 관리자 간의 상호 작용은 JTA 사양에서 설명합니다.

분산 트랜잭션 지원은 메시징 클라이언트가 JTA에서 정의된 XA Resource 인터페이스를 통해 분산 트랜잭션에 참여할 수 있음을 의미합니다. 이 인터페이스는 2단계 완결을 구현하는 여러 메소드를 정의합니다. 클라이언트측에서 API 호출이 이루어지는 동안 Message Queue 브로커는 분산 트랜잭션 내부의 다양한 발신 및 수신 작업을 추적하고 트랜잭션 상태를 추적하며 JTS (Java Transaction Service)가 제공하는 분산 트랜잭션 관리자와의 조정을 통해서만 메시징 작업을 완료합니다.

로컬 트랜잭션과 마찬가지로 클라이언트는 예외를 무시하거나 작업을 다시 시도하거나 전체 분산 트랜잭션을 롤백하는 방법으로 예외를 처리할 수 있습니다.

Message Queue는 XA 연결 팩토리를 통해 분산 트랜잭션 지원을 구현합니다. 이를 통해 XA 연결을 생성하고, XA 연결을 통해 XA 세션을 생성할 수 있습니다(38페이지의 "[JMS 프로그래밍 모델](#)" 참조). 또한 분산 트랜잭션을 지원하려면 타사의 JTS나 (JTS를 제공하는) J2EE 호환 Application Server가 필요합니다.

영구 저장소

안정성의 또 다른 중요한 측면은 일단 지속성 메시지가 대상에게 전달되었다면 메시지 서비스는 사용자에게 전달될 때까지 그 메시지를 잃어버리지 않아야 한다는 것입니다. 즉 지속성 메시지가 대상에 전달되면 메시지 서비스는 이를 영구 데이터 저장소에 저장해야 합니다(63페이지의 "[지속성 관리자](#)" 참조). 어떤 이유로 메시지 서비스가 중단되는 경우, 메시지 서비스는 메시지를 복구하여 해당 사용자에게 전달할 수 있습니다. 그 결과 메시지 전달의 오버헤드가 늘어나지만 안정성은 증가합니다.

또한 메시지 서비스는 영구 가입도 저장해야 합니다. 주제 대상의 경우, 메시지 전달을 보장하려면 지속성 메시지만 복구하는 것으로는 충분하지 않기 때문입니다. 또한 메시지 서비스는 어떤 주제의 영구 가입에 대한 정보를 복구해야 합니다. 그렇지 않으면 메시지가 도착했을 시점에 비활성 상태였다가 나중에 활성화되는 가입자에게 메시지를 전달할 수 없습니다.

메시지 전달 보장이 중요한 메시징 응용 프로그램은 메시지가 지속성을 갖도록 지정하고 대기열 대상이나 주제 대상에 대한 영구 가입 중 하나를 사용해야 합니다.

성능 균형

메시지 전달의 안정성이 높아질수록 이를 실현하기 위해 더 많은 오버헤드와 대역폭이 필요합니다. 안정성과 성능 간의 균형은 설계 시 고려해야 할 중요한 사항입니다. 비지속성 메시지를 생성하고 사용하도록 선택함으로써 성능을 극대화할 수 있습니다. 한편 지속성 메시지를 생성 및 사용하고 트랜잭션된 세션을 사용할 경우 안정성을 극대화할 수 있습니다. 이 둘 간에는 Message Queue별 연결 및 확인 등록 정보의 사용을 비롯하여 응용 프로그램의 필요 사항에 따라 다양한 옵션이 존재합니다(Message Queue Java Client Developer's Guide 참조). 이러한 균형에 대해서는 232페이지의 "성능에 영향을 미치는 응용 프로그램 설계 요소"에서 자세히 설명합니다.

메시지 선택

JMS는 메시지 선택기에 설정된 기준을 토대로 메시지 서비스가 메시지 필터링 및 라우팅을 수행할 수 있는 메커니즘을 제공합니다. 생성자 클라이언트는 응용 프로그램별 등록 정보를 메시지에 포함시킬 수 있으며, 사용자 클라이언트는 그러한 등록 정보에 기반한 선택 기준을 사용하여 메시지에 인터레스트를 표시할 수 있습니다. 그 결과 클라이언트 작업이 간소화되고 해당 메시지가 필요하지 않은 클라이언트로 메시지를 전달하는 오버헤드가 없어집니다. 그러나 선택 기준을 처리하는 오버헤드가 메시지 서비스에 추가됩니다. 메시지 선택기 구문과 의미는 JMS 사양에 설명되어 있습니다.

메시지 순서 및 우선 순위

일반적으로 단일 세션을 통해 어떤 대상으로 보내지는 모든 메시지는 발신 순서에 따라 사용자에게 전달됩니다. 그러나 다른 우선 순위가 지정된 경우, 메시징 시스템은 우선 순위가 높은 메시지를 먼저 전달하려고 합니다.

그 외에 클라이언트 응용 프로그램의 메시지 사용 순서와 생성 순서 간의 관계는 대략적인 것에 불과합니다. 대상으로의 메시지 전달과 대상으로부터의 메시지 전달은 메시지 발신 순서, 메시지가 발신된 세션(연결), 메시지의 지속성 여부, 메시지의 수명, 메시지 우선 순위, 대기열 대상의 메시지 전달 정책(77페이지의 "대기열 대상" 참조) 및 메시지 서비스가용성 등 시간에 영향을 미치는 다양한 요소에 따라 결정되기 때문입니다.

Message Queue 메시지 서버가 상호 연결된 여러 브로커를 사용하는 경우(82페이지의 "멀티 브로커 클러스터(엔터프라이즈판)" 참조), 클라이언트의 메시지 사용 순서는 더 복잡해 지는데, 이는 서로 다른 브로커에서 대상으로부터의 메시지 전달 순서가 불확정적이기 때문입니다. 따라서 어떤 브로커가 전달하는 메시지는 먼저 메시지를 수신한 다른 브로커가 전달하는 메시지보다 먼저 처리될 수도 있습니다.

어떤 경우에도 해당 사용자에게 대해 우선 순위가 높은 메시지는 우선 순위가 낮은 메시지보다 먼저 처리됩니다.

Message Queue 메시징 시스템

이 장에서는 [그림 2-1](#)에 나와 있는 시스템의 주요 부분에 중점을 두고 Sun Java™ System Message Queue 메시징 시스템에 대해 소개합니다. 또한 각 부분이 어떻게 상호 작용하면서 안정적인 메시지 전달을 제공하는지 설명합니다.

그림 2-1 Message Queue 시스템 구조

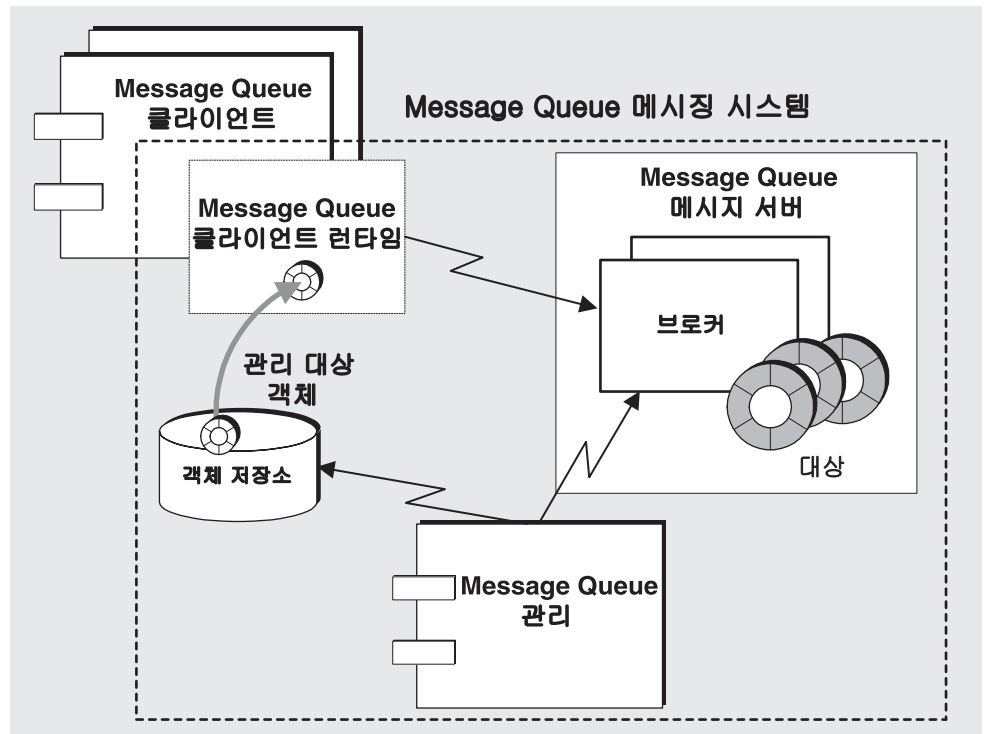


그림 2-1에서 보여주는 Message Queue 메시징 시스템의 주요 부분은 다음과 같습니다.

- Message Queue 메시지 서버
- Message Queue 클라이언트 런타임
- Message Queue 관리 대상 객체
- Message Queue 관리

첫 세 항목에 대해서는 다음 절에서 다룹니다. 마지막 항목은 3장, "[Message Queue 관리 작업 및 도구](#)"에서 소개합니다.

Message Queue 메시지 서버

이 절에서는 51페이지의 [그림 2-1](#)에서 보여주는 Message Queue 메시지 서버의 다양한 부분을 설명합니다. 여기에는 다음 항목이 포함됩니다.

브로커. Message Queue 브로커는 Message Queue 메시징 시스템을 위해 전달 서비스를 제공합니다. 메시지 전달은 연결 서비스, 메시지 라우팅 및 전달, 지속성, 보안 및 로깅을 처리하는 수많은 지원 구성 요소에 따라 달라집니다(자세한 정보는 "[브로커](#)" 참조). 메시지 서버는 하나 이상의 브로커 인스턴스를 사용할 수 있습니다(82페이지의 "[멀티 브로커 클러스터\(엔터프라이즈판\)](#)" 참조).

물리적 대상. 메시지 전달은 생성자 클라이언트에서 브로커가 관리하는 물리적 대상으로 전달 그리고 대상에서 하나 이상의 사용자 클라이언트로 전달의 2단계로 구성되는 과정입니다. 물리적 대상은 브로커의 물리적 메모리 및/또는 영구 저장소에서의 위치를 나타냅니다(자세한 정보는 76페이지의 "[물리적 대상](#)" 참조).

브로커

Message Queue 메시징 시스템의 메시지 전달은 브로커(또는 탠덤으로 작동하는 브로커 인스턴스로 구성된 클러스터)가 수행합니다. 메시지는 생성자 클라이언트에서 대상으로 전달된 다음 대상에서 하나 이상의 사용자 클라이언트로 전달됩니다. 메시지 전달을 수행하려면 브로커가 클라이언트와 통신 채널을 설정하고, 인증 및 권한 부여를 수행하며, 메시지 경로를 올바르게 지정하고, 안정적인 전달을 보장하며, 시스템 성능을 모니터링할 데이터를 제공해야 합니다.

이렇게 복잡한 기능들을 수행하기 위해 브로커는 각각의 전달 과정에서 특별한 역할을 맡는 다양한 내부 구성 요소를 사용합니다. 주요 브로커 구성 요소는 **그림 2-2**에서 확인할 수 있으며, **표 2-1**에서도 간략히 설명합니다. 메시지 라우터 구성 요소가 주요 메시지 라우팅 및 전달 서비스를 수행하고 다른 구성 요소는 메시지 라우터가 종속되는 중요한 지원 서비스를 제공합니다.

그림 2-2 브로커 서비스 구성 요소

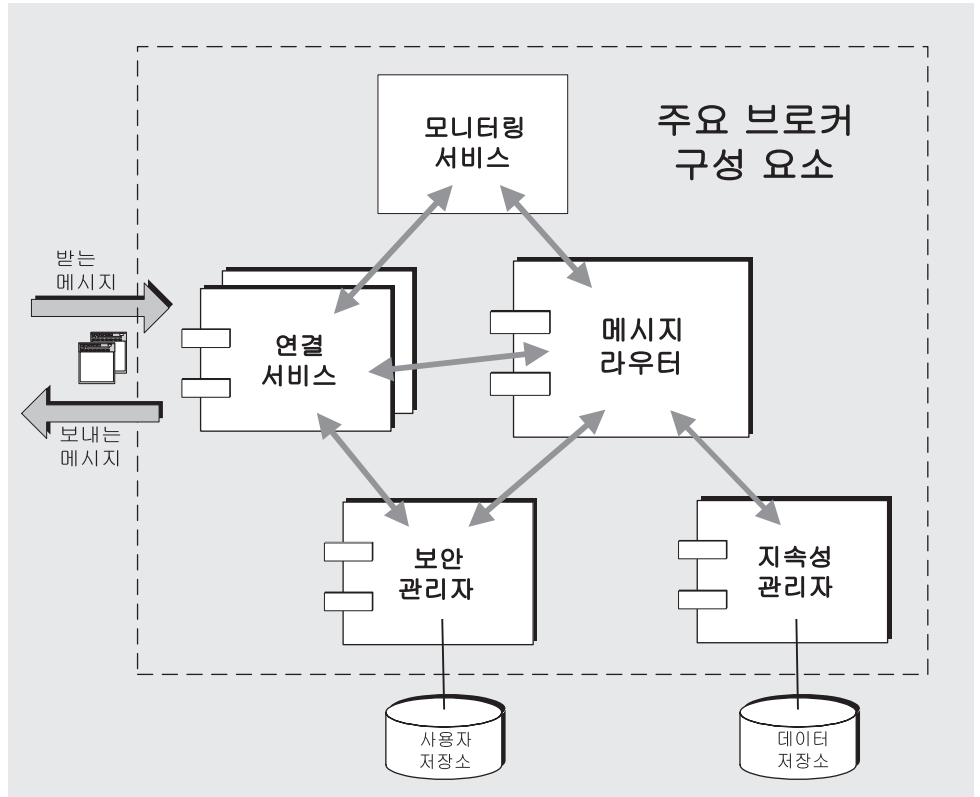


표 2-1 주요 브로커 구성 요소 및 기능

구성 요소	설명/기능
메시지 라우터	메시지의 라우팅 및 전달을 관리합니다. 여기에는 JMS 메시지를 비롯하여 Message Queue 메시징 시스템이 JMS 메시지 전달을 지원하기 위해 사용하는 제어 메시지도 포함됩니다.

표 2-1 주요 브로커 구성 요소 및 기능(계속)

구성 요소	설명/기능
연결 서비스	브로커와 클라이언트 사이의 물리적 연결을 관리하면서 받고 보내는 메시지 전송을 담당합니다.
지속성 관리자	영구 저장소에 대한 데이터 쓰기를 관리하여 시스템 오류로 인해 JMS 메시지 전달 오류가 발생하지 않게 합니다.
보안 관리자	브로커와의 연결을 요청하는 사용자에게 인증 서비스를 제공하고 인증된 사용자에게 권한 부여 서비스(액세스 제어)를 제공합니다.
모니터링 서비스	관리자가 브로커를 모니터링 및 관리할 때 사용할 수 있는 다양한 출력 채널에 기록 가능한 메트릭 및 진단 정보를 생성합니다.

로드 상태, 응용 프로그램 복잡성 등에 따라 브로커 성능을 최적화하도록 이 내부 구성 요소들을 구성할 수 있습니다. 다음 절에서는 다양한 구성 요소가 수행하는 기능 및 그 동작에 영향을 주도록 구성 가능한 등록 정보를 더 자세히 살펴 봅니다.

연결 서비스

Message Queue 브로커는 Message Queue 응용 프로그램 클라이언트와 Message Queue 관리 클라이언트와의 통신을 모두 지원합니다(97페이지의 "Message Queue 관리 도구" 참조). 각 서비스는 서비스 유형 및 프로토콜 유형을 통해 지정됩니다.

서비스 유형. 서비스가 JMS 메시지 전달(NORMAL)을 제공하는지 또는 Message Queue 관리(ADMIN) 서비스를 제공하는지 지정합니다.

프로토콜 유형. 서비스를 지원하는 기본 전송 프로토콜 계층을 지정합니다.

현재 Message Queue 브로커에서 사용 가능한 연결 서비스는 표 2-2에서 확인할 수 있습니다.

표 2-2 브로커가 지원하는 연결 서비스

서비스 이름	서비스 유형	프로토콜 유형
jms	NORMAL	tcp
ssljms (엔터프라이즈판)	NORMAL	tls (SSL 기반 보안)

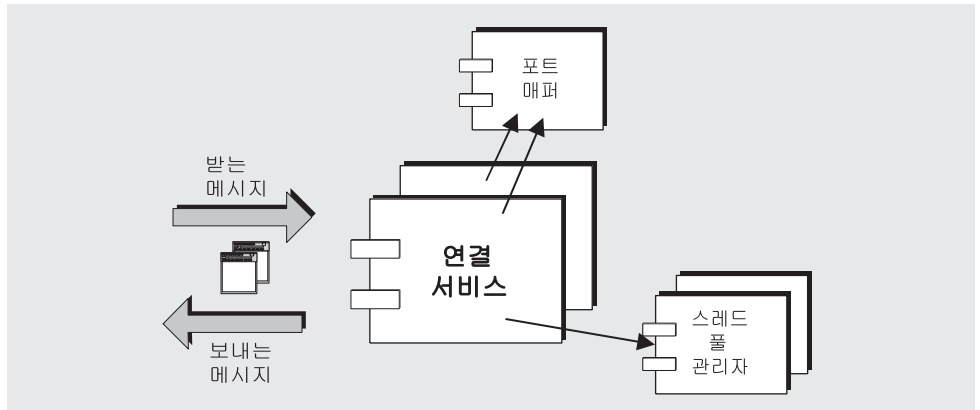
표 2-2 브로커가 지원하는 연결 서비스(계속)

서비스 이름	서비스 유형	프로토콜 유형
httpjms (엔터프라이즈판)	NORMAL	http
httpsjms (엔터프라이즈판)	NORMAL	https (SSL 기반 보안)
admin	ADMIN	tcp
ssladmin (엔터프라이즈판)	ADMIN	tls (SSL 기반 보안)

이 연결 서비스 중 어느 것이라도 또는 전부 실행하도록 브로커를 구성할 수 있습니다. 각 연결 서비스는 특정 포트에서 사용 가능하며, 브로커의 호스트 이름과 포트 번호로 지정됩니다. 포트는 동적으로 할당될 수도 있고 사용자가 연결 서비스에 사용 가능한 포트를 직접 지정할 수도 있습니다.

그림 2-3에 표시된 것처럼 각 서비스는 공통 포트 매핑에 등록되지만 자체 스레드 풀 관리자도 있습니다.

그림 2-3 연결 서비스 지원



포트 매핑

Message Queue는 포트를 여러 연결 서비스에 매핑하는 **포트 매핑**을 제공합니다. 포트 매핑 자체는 표준 포트 번호 7676에 위치합니다. 클라이언트가 브로커와 연결을 설정할 때 이 클라이언트는 먼저 포트 매핑에 접속하여 원하는 연결 서비스의 포트 번호를 요청합니다.

jms, ssljms, admin 및 **ssladmin** 연결 서비스를 구성할 때 *정적* 포트 번호를 할당할 수도 있지만 이런 구성은 특수한 상황(예: 방화벽을 통한 연결)에서만 수행되고 일반적으로는 사용하지 않는 것이 좋습니다. **httpjms** 및 **httpsjms** 서비스는 각각 **부록 C**, **"HTTP/HTTPS 지원(엔터프라이즈판)"**에서 **311페이지의 표 C-1**과 **323페이지의 표 C-3**에서 설명되어 있는 등록 정보를 사용하여 구성합니다.

스레드 풀 관리자

각 연결 서비스는 다중 스레드 방식으로서, 다중 연결을 지원합니다. 이 연결에 필요한 스레드는 *스레드 풀 관리자* 구성 요소가 관리하는 스레드 풀에서 유지 관리됩니다. 스레드 풀 관리자를 구성하여 스레드 풀에서 유지 관리되는 최소 스레드 수와 최대 스레드 수를 설정할 수 있습니다. 연결 시 스레드가 필요하면 스레드 풀에 해당 스레드가 추가됩니다. 최소 수를 초과할 경우, 시스템은 최소 수 임계값에 도달할 때까지 스레드를 종료시켜 여유 스레드를 확보하는 방법으로 메모리 자원을 절약합니다. 새 스레드가 계속 작성될 필요가 없도록 이 값을 충분히 크게 설정할 수 있습니다. 연결 로드가 많은 경우, 스레드 풀의 최대 수에 도달할 때까지 스레드 수가 증가하기도 합니다. 그 후 스레드가 사용 가능해질 때까지 연결은 대기합니다.

스레드 풀의 스레드는 단일 연결 전용으로 사용되거나(*전용 모델*) 필요에 따라 여러 연결에 지정될 수 있습니다(*공유 모델*).

전용 모델. 브로커와의 연결마다 연결에서 받는 메시지 처리 전용과 보내는 메시지 처리 전용의 2개의 전용 스레드가 필요합니다. 따라서 연결 수가 스레드 풀에 있는 최대 스레드 수의 절반으로 제한되지만, 이 방법은 우수한 성능을 제공합니다.

공유 모델(엔터프라이즈판). 메시지를 보내거나 받을 때마다 공유 스레드에서 연결을 처리합니다. 이 모델에서는 각 연결에 전용 스레드가 필요하지 않기 때문에 연결 서비스(및 브로커)가 지원할 수 있는 연결 수가 증가합니다. 그러나 스레드 공유와 관련된 어느 정도의 성능 오버헤드가 있습니다. 스레드 풀 관리자는 연결 활동을 모니터링하고 필요에 따라 스레드에 연결을 지정하는 분산자 스레드 집합을 사용합니다. 이러한 각 분산자 스레드가 모니터링하는 연결 수를 제한하여 이 활동과 관련된 성능 오버헤드를 최소화할 수 있습니다.

보안

각 연결 서비스는 특정 인증 및 권한 부여(액세스 제어) 기능을 지원합니다(**66페이지의 "보안 관리자"** 참조).

연결 서비스 등록 정보

연결 서비스와 관련하여 구성 가능한 등록 정보는 표 2-3에서 확인할 수 있습니다(등록 정보 구성 지침은 5장, "브로커 시작 및 구성" 참조).

표 2-3 연결 서비스 등록 정보

등록 정보 이름	설명
<code>imq.service.activelist</code>	브로커 시작 시 활성화되는 쉘표로 구분된 이름별 연결 서비스 목록입니다. 지원되는 서비스: <code>jms</code> , <code>ssljms</code> , <code>httpjms</code> , <code>httpsjms</code> , <code>admin</code> , <code>ssladmin</code> . 기본값: <code>jms</code> , <code>admin</code>
<code>imq.ping.interval</code>	브로커가 연결을 통해 Message Queue 클라이언트 런타임을 연속적으로 ping하는 시도 사이의 간격입니다. 기본값: 120초
<code>imq.hostname</code>	사용 가능한 호스트가 두 개 이상 있는 경우(예: 컴퓨터에 네트워크 인터페이스 카드가 두 개 이상인 경우) 모든 연결 서비스를 바인딩할 호스트(호스트 이름 또는 IP 주소)를 지정합니다. 기본값: 사용 가능한 모든 IP 주소
<code>imq.portmapper.port</code>	브로커의 기본 포트(포트 매퍼가 위치한 포트)를 지정합니다. 한 호스트에서 둘 이상의 브로커 인스턴스를 실행하는 경우 각 인스턴스는 고유한 포트 매퍼 포트를 지정 받아야 합니다. 기본값: 7676
<code>imq.portmapper.hostname</code>	사용 가능한 호스트가 두 개 이상 있는 경우(예: 컴퓨터에 네트워크 인터페이스 카드가 두 개 이상인 경우) 포트 매퍼를 바인딩할 호스트(호스트 이름 또는 IP 주소)를 지정합니다. 기본값: <code>imq.hostname</code> 의 값을 상속합니다.
<code>imq.portmapper.backlog</code>	포트 매퍼가 요청 거부 전까지 처리할 수 있는 최대 동시 요청 수를 지정합니다. 이 등록 정보는 운영 체제 백로그에 저장되어 포트 매퍼의 처리를 대기할 수 있는 요청 수를 설정합니다. 기본값: 50
<code>imq.service_name.protocol_type¹.port</code>	<code>jms</code> , <code>ssljms</code> , <code>admin</code> 및 <code>ssladmin</code> 서비스에 한해 명명된 연결 서비스의 포트 번호를 지정합니다. 기본값: 0 (포트 매퍼가 동적으로 포트를 할당) httpjms 및 httpsjms 연결 서비스를 구성하려면 부록 C , " HTTP/HTTPS 지원(엔터프라이즈판) "을 참조하십시오.

표 2-3 연결 서비스 등록 정보(계속)

등록 정보 이름	설명
<code>imq.service_name.</code> <code>protocol_type¹.hostname</code>	<code>jms</code> , <code>ssljms</code> , <code>admin</code> 및 <code>ssladmin</code> 서비스에 한해, 사용 가능한 호스트가 두 개 이상 있는 경우(예: 컴퓨터에 네트워크 인터페이스 카드가 두 개 이상인 경우) 명명된 연결 서비스를 바인드할 호스트(호스트 이름 또는 IP 주소)를 지정합니다. 기본값: <code>imq.hostname</code> 의 값을 상속합니다.
<code>imq.service_name.</code> <code>min_threads</code>	이 스레드 수에 도달하면 명명된 연결 서비스가 사용할 수 있도록 스레드 풀에서 스레드가 유지 관리됩니다. 기본값: 연결 서비스에 따라 다릅니다(130페이지의 표 5-1 참조).
<code>imq.service_name.</code> <code>max_threads</code>	이 스레드 수를 초과하면 명명된 연결 서비스가 사용할 수 있는 신규 스레드가 더 이상 스레드 풀에 추가되지 않습니다. 이 수는 0보다 크고 <code>min_threads</code> 값보다 커야 합니다. 기본값: 연결 서비스에 따라 다릅니다(130페이지의 표 5-1 참조).
<code>imq.service_name.</code> <code>threadpool_model</code>	명명된 연결 서비스에 대해 스레드가 연결 전용인지(전용) 또는 필요에 따라 여러 연결에 의해 공유되는지(공유)를 지정합니다. 공유 모델(스레드 풀 관리)은 브로커가 지원하는 연결 수를 늘리지만, <code>jms</code> 및 <code>admin</code> 연결 서비스에 대해서만 구현됩니다. 기본값: 연결 서비스에 따라 다릅니다(130페이지의 표 5-1 참조).
<code>imq.shared.</code> <code>connectionMonitor_limit</code>	공유 스레드 풀 모델에 한해 분산자 스레드가 모니터링할 수 있는 최대 연결 수를 지정합니다(시스템은 모든 연결을 모니터링하기에 충분한 수의 분산자 스레드를 할당). 이 값이 작을수록 시스템은 더 신속하게 스레드에 활성 연결을 지정할 수 있습니다. 값 -1은 제한이 없음을 의미합니다. 기본값: 운영 체제에 따라 다릅니다(130페이지의 표 5-1 참조).

1. `protocol_type`은 표 2-2에 지정되어 있습니다.

메시지 라우터

지원되는 연결 서비스를 사용하여 클라이언트와 브로커 사이에 연결이 설정되면 라우팅 및 메시지 전달을 할 수 있습니다.

기본적인 전달 메커니즘

포괄적으로 말하자면, 브로커가 처리하는 메시지는 생성자 클라이언트가 사용자 클라이언트를 대상으로 보내는 페이로드 메시지인 JMS 메시지 그리고 JMS 메시지 전달을 지원하고자 클라이언트를 오가는 수많은 제어 메시지의 2가지 범주로 구분됩니다.

받는 메시지가 JMS 메시지인 경우 브로커는 대상 유형(대기열 또는 주제)에 따라 사용자 클라이언트로 경로를 지정합니다.

- 대상이 주제라면 JMS 메시지는 해당 주제의 모든 활성 가입자로 향하는 경로를 즉시 지정합니다. 비활성 영구 가입자의 경우, 메시지 라우터는 메시지를 보관했다가 해당 가입자가 활성 상태가 되면 전달합니다.
- 대상이 대기열이라면 JMS 메시지는 해당 대기열로 들어가고, 대기열의 맨 앞에 도달하는 시점에 해당 사용자에게 전달됩니다. 메시지가 대기열의 맨 앞에 도달하는 순서는 도착 순서 및 우선 순위에 따라 결정됩니다.

메시지 라우터가 목표한 모든 사용자에게 메시지를 전달했다면 메시지 라우터는 메모리에서 해당 메시지를 삭제합니다. 지속성 메시지인 경우(46페이지의 "안정적인 메시징" 참조) 브로커의 영구 데이터 저장소에서 해당 메시지를 제거합니다.

안정적인 전달: 확인 및 트랜잭션

지금까지 설명한 전달 메커니즘은 안정적인 전달(46페이지의 "안정적인 메시징" 참조)을 위한 요구 사항이 추가될 경우 더 복잡해집니다. 안정적인 전달에는 브로커를 오가는 메시지 전달이 성공적으로 이루어지게 하고 실제 메시지 전달 이전에 브로커가 메시지나 전달 정보를 잃어버리지 않게 하는 2가지 측면이 있습니다.

브로커를 오가는 메시지가 성공적으로 전달되도록 하기 위해 Message Queue는 확인이라고 하는 다양한 제어 메시지를 사용합니다.

예를 들어, 생성자가 어떤 대상으로 JMS 메시지(제어 메시지와 반대되는 페이로드 메시지)를 보낼 때 브로커는 JMS 메시지 수신을 확인하는 제어 메시지인 브로커 확인을 회신합니다(기본적으로 생성자가 JMS 메시지를 지속성 메시지로 지정한 경우에만 Message Queue가 이 기능을 수행함). 생성자 클라이언트는 브로커 확인을 통해 대상에게 확실히 전달되도록 합니다(87페이지의 "메시지 생성" 참조).

그와 비슷하게 브로커가 사용자에게 JMS 메시지를 전달할 경우, 사용자 클라이언트는 해당 메시지를 수신 및 처리했음을 알리는 확인을 회신합니다. 클라이언트는 세션 객체 작성 시 이 확인을 얼마나 자동으로 또는 얼마나 자주 보낼 것인지 지정하지만, 원칙상 메시지 라우터는 자신이 메시지를 전달했던 각 메시지 사용자로부터 확인을 받지 않은 경우(예: 여러 가입자 각각으로부터 한 주제로) 해당 JMS 메시지를 메모리에서 삭제하지 않습니다.

어떤 주제에 대한 영구 가입자의 경우, 메시지 라우터는 각 JMS 메시지를 대상에 보존하고 각 영구 가입자가 활성 사용자가 되면 메시지를 전달합니다. 메시지 라우터는 클라이언트 확인을 수신하여 이를 기록하고, (그 전까지 JMS 메시지가 만료되지 않는 한) 모든 확인을 수신한 후에야 JMS 메시지를 삭제합니다.

또한 메시지 라우터는 클라이언트에게 브로커 확인을 회신함으로써 클라이언트 확인을 받았음을 알립니다. 사용자 클라이언트는 브로커 확인을 통해 브로커가 JMS 메시지를 2회 이상 전달하지 않도록 합니다(88페이지의 "메시지 사용" 참조). 이는 어떤 이유로 브로커가 클라이언트 확인을 받지 못한 경우에 발생할 수 있습니다.

브로커가 클라이언트 확인을 받지 않고 JMS 메시지를 두 번째 전달하는 경우, 메시지에 재전송 플러그가 표시됩니다. 일반적으로 브로커가 클라이언트 확인을 받기 전에 클라이언트 연결이 닫히고 이어서 새 연결이 열리면 브로커는 JMS 메시지를 재전송합니다. 예를 들어 메시지 확인을 받기 전에 대기열의 메시지 사용자가 오프라인되고 이어서 다른 사용자가 해당 대기열에 등록하면 브로커는 미확인된 메시지를 새 사용자에게 재전송합니다.

위에서 설명한 클라이언트와 브로커 확인 과정은 JMS 메시지 전달이 트랜잭션으로 그룹화된 경우에도 적용됩니다. 이 경우 클라이언트와 브로커 확인은 개별 JMS 메시지 송수신 수준뿐 아니라 트랜잭션 수준에서도 수행됩니다. 트랜잭션이 완결되면 브로커 확인이 자동으로 보내집니다.

브로커는 트랜잭션을 추적하면서 트랜잭션 완결 또는 실패 시 롤백이 가능하게 합니다. 또한 이 트랜잭션 관리는 더 큰 규모의 분산 트랜잭션에 포함되는 로컬 트랜잭션을 지원 합니다(47페이지의 "분산 트랜잭션" 참조). 브로커는 트랜잭션이 완결될 때까지 그 상태를 추적합니다. 브로커가 시작되면 이 브로커는 아직 완결되지 않은 모든 트랜잭션을 검사하며, PREPARED 상태의 트랜잭션을 제외하고 모든 트랜잭션을 롤백하도록 기본 설정되어 있습니다.

안정적인 전달: 지속성

안정적인 전달의 또 다른 측면은 실제로 메시지가 전달될 때까지 브로커가 메시지나 전달 정보를 잃어버리지 않게 하는 것입니다. 일반적으로 메시지는 전달되거나 만료될 때까지 메모리에 남아 있습니다. 그러나 브로커에 오류가 발생할 경우 이 메시지는 손실됩니다.

생성자 클라이언트는 메시지가 지속성을 갖도록 지정할 수 있으며, 이 경우 메시지 라우터는 데이터베이스나 파일 시스템에 메시지를 저장하는 *지속성 관리자*에게 메시지를 전달하여(63페이지의 "지속성 관리자" 참조) 브로커 오류 발생 시 메시지가 복구될 수 있게 합니다.

메모리 자원 및 메시지 흐름 관리

브로커의 성능과 안정성은 사용 가능한 시스템 자원 그리고 메모리와 같은 자원이 얼마나 효율적으로 활용되는가에 따라 달라집니다. 특히 메시지의 생성이 사용보다 훨씬 빠를 경우 메시지 라우터가 넘치게 되어 모든 메모리 자원을 소진할 수 있습니다. 이런 현상을 방지하기 위해 메시지 라우터는 자원이 부족해질 경우 다음과 같은 세 가지 메모리 보호 수준을 사용하여 시스템을 운영합니다.

개별 대상에 대한 메시지 제한. 물리적 대상에서 메시지 수와 메시지가 사용하는 총 메모리를 제한하는 속성을 설정할 수 있고(171페이지의 표 6-10 참조), 이러한 제한에 도달할 경우 메시지 라우터가 수행하는 네 가지 응답을 지정할 수도 있습니다. 네 가지 제한 동작은 다음과 같습니다.

- 메시지 생성자의 속도 줄이기
- 메모리에서 가장 오래된 메시지 삭제
- 메시지 보존 기간을 기준으로 메모리에서 우선 순위가 가장 낮은 메시지 삭제
- 최신 메시지 거부

시스템 전체 메시지 제한. 시스템 전체 메시지 제한은 두 번째 보호 집합을 구성합니다. 총 메시지 수, 모든 메시지가 사용하는 메모리 등과 같이 시스템의 모든 대상에 한꺼번에 적용되는 시스템 전체 제한을 지정할 수 있습니다(62페이지의 표 2-4 참조). 시스템 전체 메시지 제한에 도달하면 메시지 라우터는 새 메시지를 거부합니다.

시스템 메모리 임계값. 시스템 메모리 임계값은 세 번째 보호 집합입니다. 브로커가 메모리 과부하 방지를 위한 조치의 수위를 점점 더 높게 되는 사용 가능한 시스템 메모리의 임계값을 지정할 수 있습니다.

조치는 메모리 자원 상태, 즉 초록(사용 가능한 메모리 충분), 노랑(브로커 메모리 감소 중), 주황(브로커 메모리 부족) 및 빨강(브로커가 사용 가능한 메모리 없음)에 따라 달라집니다. 브로커의 메모리 상태가 초록에서 노랑 및 주황을 거쳐 빨강으로 변하면 브로커는 다음과 같이 점점 더 높은 수준의 조치를 수행합니다.

- 활성 메모리의 메시지를 영구 저장소에 스왑(63페이지의 "지속성 관리자" 참조). 일반적으로 저장되지 않는 비지속성 메시지를 스왑하여 시스템이 메모리를 확보할 수 있도록 합니다.
- 비지속성 메시지 생성자를 억제한 뒤 결국 브로커로 향하는 메시지 흐름을 중지시킴(지속성 메시지 흐름은 브로커가 각 메시지를 확인해야 하기 때문에 자동으로 제한됨)

두 조치 모두 성능을 떨어뜨립니다.

시스템 메모리 임계값에 도달하는 경우는 대상별 메시지 제한과 시스템 전체 메시지 제한을 잘못 설정했기 때문입니다. 임계값만으론 잠재적 메모리 과부하를 제때에 잡을 수 없는 경우도 있습니다. 따라서 이 기능에만 의존하여 메모리 자원을 제어해서는 안 되고 대상을 개별적 및 전체적으로 구성하여 메모리 자원을 최적화하는 것이 좋습니다.

메시지 라우터 등록 정보

메모리 자원 관리를 위한 시스템 전체 제한 및 시스템 메모리 임계값은 표 2-4에 자세히 설명되어 있습니다(등록 정보 설정 지침은 5장, "브로커 시작 및 구성" 참조).

표 2-4 메시지 라우터 등록 정보

등록 정보 이름	설명
imq.message.expiration.interval	만료된 메시지 재생 이용이 발생하는 빈도를 초 단위로 지정합니다. 기본값: 60
imq.system.max_count	브로커가 보관하는 최대 메시지 수를 지정합니다. 추가 메시지가 거부됩니다. 값 -1은 제한이 없음을 의미합니다. 기본값: -1
imq.system.max_size	브로커가 보관하는 최대 전체 메시지 최대 크기(바이트, KB, MB)를 지정합니다. 추가 메시지가 거부됩니다. 값 -1은 제한이 없음을 의미합니다. 기본값: -1
imq.message.max_size	메시지 본문의 최대 허용 크기(바이트, KB, MB 단위)를 지정합니다. 이 크기를 초과하는 메시지는 거부됩니다. 값 -1은 제한이 없음을 의미합니다. 기본값: 70m (MB)

표 2-4 메시지 라우터 등록 정보(계속)

등록 정보 이름	설명
<code>imq.resource_state.threshold</code>	각 메모리 자원 상태가 트리거되는 메모리 사용률을 지정합니다. 자원 상태는 초록, 노랑, 주황 및 빨강으로 표시됩니다. 기본값: 각각 0, 80, 90 및 98
<code>imq.resource_state.count</code>	일괄 처리에서 허용되는 받는 메시지의 최대 수를 지정합니다. 이 제한에 이르면 각 메모리 자원 상태가 트리거됩니다. 이 제한은 시스템 메모리가 점점 부족해지면 메시지 생성자를 억제합니다. 기본값: 각각 5000, 500, 50 및 0
<code>imq.transaction.autorollback</code>	PREPARED 상태에 있는 분산 트랜잭션이 브로커 시작 시 자동으로 롤백되는지 여부를 지정합니다(true/false). false라면 <code>imqcmd</code> 를 사용하여 수동으로 트랜잭션을 완결하거나 롤백해야 합니다(180페이지의 "트랜잭션 관리" 참조). 기본값: false

지속성 관리자

오류 발생 시 브로커를 복구하려면 메시지 전달 작업 상태를 다시 작성해야 합니다. 그러기 위해서는 모든 지속성 메시지와 기본적인 라우팅 및 전달 정보를 데이터 저장소에 저장해야 합니다. *지속성 관리자* 구성 요소는 이 정보의 작성 및 검색을 관리합니다.

오류가 발생한 브로커를 복구하려면 전달되지 못한 메시지를 복원하는 것 이상의 작업이 필요합니다. 또한 브로커는 다음을 수행할 수 있어야 합니다.

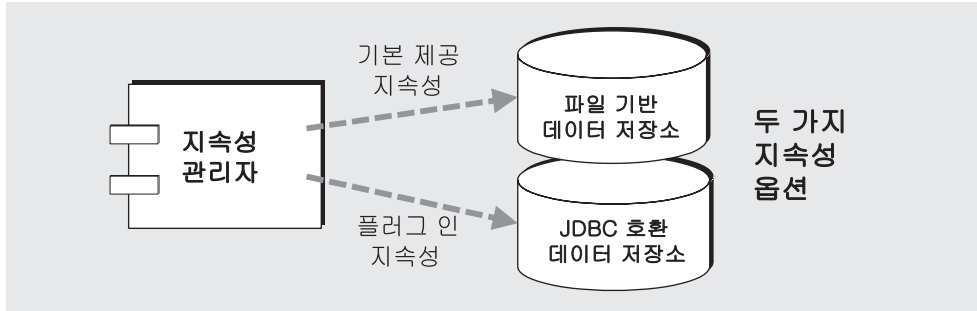
- 대상 다시 작성
- 각 주제의 영구 가입 목록 복원
- 각 메시지의 확인 목록 복원
- 완결된 모든 트랜잭션 상태 복제

지속성 관리자는 이 모든 상태 정보의 저장 및 복원을 관리합니다.

브로커가 다시 시작되면 대상 및 영구 가입을 다시 작성하고 지속성 메시지를 복구하며 모든 트랜잭션의 상태를 복원하고 전달되지 못한 메시지의 라우팅 테이블을 다시 작성합니다. 그런 다음 메시지 전달을 다시 시작할 수 있습니다.

Message Queue는 기본 제공 및 플러그인 지속성 모듈을 모두 지원합니다(그림 2-4 참조). 기본 제공 지속성은 파일 기반 데이터 저장소입니다. 플러그인 지속성은 JDBC™ (Java Database Connectivity) 인터페이스를 사용하며 JDBC 호환 데이터 저장소가 필요합니다. 일반적으로 기본 제공 지속성은 플러그인 지속성보다 더 빠릅니다. 그러나 JDBC 호환 데이터베이스 시스템 사용의 중복 및 관리 기능을 선호하는 사용자도 있습니다.

그림 2-4 지속성 관리자 지원



기본 제공 지속성

기본 Message Queue 영구 저장소 솔루션은 파일 기반 데이터 저장소입니다. 이 방법에서는 개별 파일을 사용하여 메시지, 대상, 영구 가입, 트랜잭션과 같은 지속성 데이터를 저장합니다.

파일 기반 데이터 저장소는 데이터 저장소와 연관된 브로커 인스턴스의 이름(*instanceName*)으로 식별되는 디렉토리에 있습니다(부록 A, "Message Queue 데이터의 위치" 참조).

```
.../instances/instanceName/fs350/
```

메시지가 있는 대상에 따라 지속성 메시지를 디렉토리에 저장하도록 파일 기반 데이터 저장소를 구성합니다. 대부분의 메시지는 가변 크기 레코드로 구성되는 단일 파일에 저장됩니다.

메시지를 추가 및 제거할 때 단편화를 줄이려면 가변 크기 레코드 파일을 압축할 수 있습니다(176페이지의 "대상 압축" 참조). 또한, 기본 제공 지속성 관리자는 크기가 구성 가능한 임계값(`imq.persist.file.message.max_record_size`)을 초과하는 메시지를 가변 크기 레코드 파일이 아니라 각자 해당 파일에 저장합니다. 이러한 개별 파일에서는 파일을 재사용할 수 있도록 파일 풀이 유지 관리됩니다. 메시지 파일이 더 이상 필요하지 않은 경우 해당 파일이 삭제되는 대신 대상 디렉토리의 사용 가능한 파일 풀에 추가되어 새 메시지를 저장하는 데 사용됩니다.

대상 파일 풀의 최대 파일 수(`imq.persist.file.destination.message.filepool.limit`)를 구성할 수 있으며 재사용 태그 표시(잘라내지 않음)와 반대되는 개념으로 파일 풀에서 지울(0으로 잘라냄) 사용 가능한 파일 비율(`imq.persist.file.message.filepool.cleanratio`)을 지정할 수도 있습니다. 지운 파일의 비율이 높을수록 파일 풀 관리에 필요한 디스크 공간은 더 줄어들지만 더 많은 오버헤드가 필요합니다. 또한 종료 시 태그가 표시된 파일을 지울 것인지 여부도 지정할 수 있습니다(`imq.persist.file.message.cleanup` 참조). 파일을 지우면 디스크 공간은 덜 차지하지만, 브로커 종료 속도가 느려집니다.

다른 모든 지속성 데이터(대상, 영구 가입 및 트랜잭션)는 별도의 파일에 저장됩니다. 즉, 모든 대상, 모든 영구 가입 등이 각각 서로 다른 별도의 파일에 저장됩니다.

안정성을 최대화하려면 지속성 작업이 메모리 상태에서 물리적 저장 장치와 동기화되도록 지정할 수 있습니다(`imq.persist.file.sync.enabled`). 이렇게 하면 시스템 충돌로 인한 데이터 손실은 제거할 수 있지만 성능이 떨어집니다.

데이터 저장소에는 중요 정보나 소유 정보를 포함하는 메시지가 있을 수 있기 때문에 `...instances/instanceName/fts350/` 디렉토리를 인증되지 않은 액세스로부터 보호하는 것이 좋습니다. 자세한 지침은 [339페이지의 "지속성 데이터 보안"](#)을 참조하십시오.

플러그인 지속성

JDBC 드라이버를 통해 액세스 가능한 데이터 저장소를 모두 액세스하도록 브로커를 설정할 수 있습니다. 이 경우 여러 JDBC 관련 브로커 구성 등록 정보를 설정하고 데이터베이스 관리자 유틸리티(`imqdbmgr`)를 사용하여 적합한 체계를 갖는 데이터 저장소를 만들어야 합니다. 절차 및 관련 구성 등록 정보는 [부록 B, "플러그인 지속성 설정"](#)을 참조하십시오.

지속성 관리자 등록 정보

지속성 관련 구성 등록 정보는 [66페이지의 표 2-5](#)를 참조하십시오(등록 정보 설정 지침은 [5장, "브로커 시작 및 구성"](#) 참조).

[표 2-5](#)에 있는 등록 정보는 첫 번째만 제외하고 모두 기본 제공 지속성에만 연관됩니다. 플러그인 지속성과 관련된 등록 정보는 [300페이지의 표 B-1](#)에 있습니다.

표 2-5 지속성 관리자 등록 정보

등록 정보 이름	설명
<code>imq.persist.store</code>	브로커가 기본 제공 파일 기반(file) 지속성 또는 플러그인 JDBC 호환(jdbc) 지속성을 사용하고 있는지 지정합니다. 기본값: <code>file</code>
<code>imq.persist.file.sync.enabled</code>	지속성 작업이 메모리 상태에서 물리적 저장 장치와 동기화 될 것인지 여부를 지정합니다. 값이 <code>true</code> 이면 시스템 충돌로 인한 데이터 손실은 줄어들지만 지속성 작업의 성능이 저하됩니다. 기본값: <code>false</code>
<code>imq.persist.file.message.max_record_size</code>	기본 제공 파일 기반 지속성에 대해 개별 파일에 저장되는 것과는 반대로 메시지 저장소 파일에 추가될 메시지의 최대 크기를 지정합니다. 기본값: <code>1m (MB)</code>
<code>imq.persist.file.destination.message.filepool.limit</code>	기본 제공 파일 기반 지속성에 대해 대상 파일 풀에서 재사용 가능한 최대 파일 수를 지정합니다. 이 수가 클수록 브로커는 지속성 데이터를 더 빠르게 처리할 수 있습니다. 이 값을 초과하는 사용 가능한 파일은 삭제됩니다. 이 한도를 초과할 경우 브로커는 필요에 따라 추가 파일을 작성하고 삭제합니다. 기본값: <code>100</code>
<code>imq.persist.file.message.filepool.cleanratio</code>	기본 제공 파일 기반 지속성에 대해 대상 파일 풀에서 <code>clean</code> 상태(0으로 잘라낸 상태)로 유지되는 사용 가능한 파일의 비율을 지정합니다. 이 값이 클수록 작업 중 파일을 지우는 데 필요한 오버헤드가 늘어나지만, 파일 풀에서 필요한 디스크 공간은 줄어듭니다. 기본값: <code>0</code>
<code>imq.persist.file.message.cleanup</code>	기본 제공 파일 기반 지속성에 대해 브로커가 종료 시 대상 파일 저장소에서 사용 가능한 파일을 지울 것인지 여부를 지정합니다. <code>false</code> 값은 브로커 종료 속도를 향상시키지만, 파일 저장소에 더 많은 디스크 공간이 필요합니다. 기본값: <code>false</code>

보안 관리자

Message Queue는 인증 및 권한 부여(액세스 제어) 기능을 제공하며 암호화 기능도 지원합니다.

인증 및 권한 부여 기능은 사용자 저장소(68페이지의 [그림 2-5](#) 참조), 즉 메시징 시스템 사용자에게 대한 정보(예: 아이디, 비밀번호, 그룹 멤버십)를 포함하는 파일, 디렉토리 또는 데이터베이스에 따라 달라집니다. 아이디와 비밀번호는 브로커와의 연결 요청 시 사용자를 인증할 때 사용됩니다. 대상에 대한 메시지 생성/사용과 같은 작업 권한을 부여할 때 아이디와 그룹 멤버십이 액세스 제어 파일과 함께 사용됩니다.

Message Queue 관리자는 Message Queue 제공 사용자 저장소(202페이지의 ["플랫 파일 사용자 저장소 사용"](#) 참조)를 채우거나 기존 LDAP 사용자 저장소를 보안 관리자 구성 요소에 플러그 인합니다(209페이지의 ["사용자 저장소에 LDAP 서버 사용"](#) 참조). 플랫 파일 사용자 저장소는 쉽게 사용할 수 있지만 보안 공격에 취약하므로 평가 및 개발 용도에 **한해** 사용해야 합니다. 반면 LDAP 사용자 저장소는 안전하므로 작업 환경에 가장 적합합니다.

인증

Message Queue 보안은 비밀번호 기반의 인증을 지원합니다. 클라이언트가 브로커와의 연결을 요청할 경우 이 클라이언트는 아이디와 비밀번호를 제출해야 합니다. 보안 관리자는 클라이언트가 제출한 아이디와 비밀번호를 사용자 저장소에 저장된 정보와 비교합니다. 클라이언트가 브로커에게 비밀번호를 전송할 때 이 비밀번호는 기본 64 인코딩이나 메시지 다이제스트(MD5) 중 한 가지를 사용하여 암호화됩니다. 보다 안전한 전송에 대해서는 68페이지의 ["암호화\(엔터프라이즈판\)"](#)를 참조하십시오. 별도로 각 연결 서비스가 사용하는 인코딩 유형을 구성하거나 브로커 전체에 대한 인코딩을 설정할 수 있습니다.

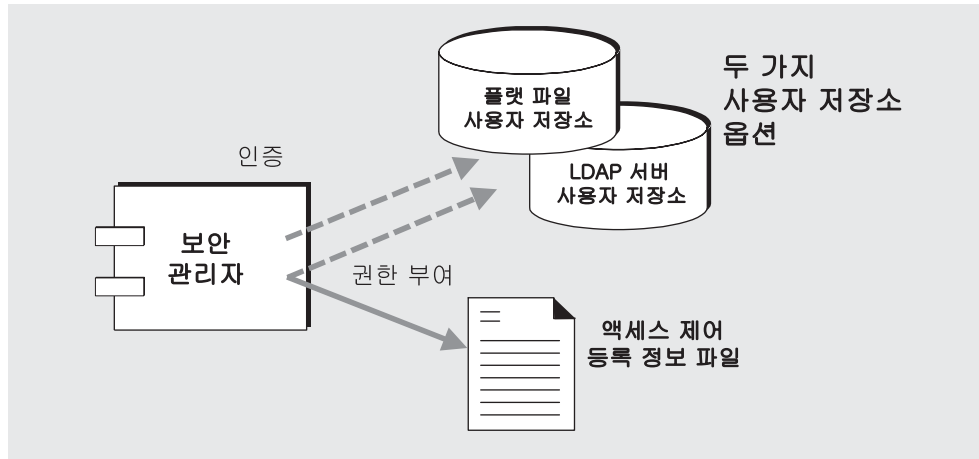
권한 부여

클라이언트 응용 프로그램 사용자가 인증되면 이 사용자는 여러 Message Queue 관련 작업을 수행할 권한을 갖습니다. 보안 관리자는 사용자 기반 및 그룹 기반 액세스 제어를 모두 지원합니다. 사용자 저장소에 있는 아이디 또는 해당 사용자가 속한 그룹에 따라 이 사용자는 특정 Message Queue 작업을 수행할 권한을 갖습니다. 액세스 제어 등록 정보 파일에서 이 액세스 제어를 지정합니다([그림 2-5](#) 참조).

사용자가 어떤 작업을 수행하려 하면 보안 관리자는 (액세스 제어 등록 정보 파일에 있는) 해당 작업 액세스를 위해 지정된 아이디/그룹 멤버십을 (사용자 저장소에 있는) 아이디/그룹 멤버십과 대조 확인합니다. 액세스 제어 등록 정보 파일은 다음 작업에 대한 권한을 지정합니다.

- 브로커와의 연결 설정
- 대상에 액세스: 사용자, 생성자 또는 특정 대상이나 모든 대상에 대한 대기열 브라우저 작성
- 대상 자동 작성

그림 2-5 보안 관리자 지원



기본 액세스 제어 등록 정보 파일이 *admin*이라는 단일 그룹만 명시적으로 참조합니다 (205페이지의 "그룹" 참조). *admin* 그룹의 사용자는 관리 서비스 연결 권한을 갖습니다. 관리 서비스 사용자는 대상 작성, 브로커 모니터링 및 제어와 같은 관리 기능을 수행할 수 있습니다. 다른 그룹으로 정의된 사용자는 기본적으로 관리 서비스 연결 권한을 가질 수 없습니다.

Message Queue 관리자는 사용자 저장소에서 그룹을 정의하고 사용자를 해당 그룹에 연결할 수 있습니다(단, 플랫 파일 사용자 저장소에서는 그룹 기능이 완전히 지원되지 않음). 그런 다음 액세스 제어 등록 정보 파일을 편집하여 메시지 생성 및 사용 또는 대기열 대상에서의 메시지 찾아보기 목적으로 대상에 대한 액세스를 사용자 및 그룹별로 지정할 수 있습니다. 개별 대상이나 모든 대상에 대한 액세스 권한을 특정 사용자나 그룹에게만 부여할 수 있습니다.

또한 브로커가 대상을 자동으로 작성할 수 있도록 구성된 경우(78페이지의 "자동 작성(대 관리 작성) 대상" 참조), 액세스 제어 등록 정보 파일을 편집하여 브로커가 특정 사용자를 위해 대상을 자동으로 작성할 수 있도록 제어할 수 있습니다.

암호화(엔터프라이즈판)

클라이언트와 브로커 사이에 전송되는 메시지를 암호화하려면 Secure Socket Layer (SSL) 표준 기반의 연결 서비스를 사용해야 합니다. SSL은 SSL 사용 가능 브로커와 SSL 사용 가능 클라이언트 사이에 암호화된 연결을 설정함으로써 연결 수준에서의 보안을 제공합니다.

Message Queue SSL 기반 연결 서비스를 사용하려면 키 도구 유틸리티(imqkeytool)를 사용하여 개인 키/공용 키 쌍을 생성합니다. 이 유틸리티는 자체 서명된 인증서에 공용 키를 내장하고 이를 Message Queue 키 저장소에 저장합니다. Message Queue 키 저장소 자체도 비밀번호로 보호됩니다. 이 잠금을 해제하려면 시작할 때 키 저장소 비밀번호를 제공해야 합니다. [218페이지의 "암호화: SSL 기반 서비스를 사용한 작업\(엔터프라이즈판\)"](#)을 참조하십시오.

키 저장소 잠금이 해제되면 브로커는 연결을 요청하는 모든 클라이언트에게 인증서를 전달할 수 있습니다. 그리고 나서 클라이언트는 인증서를 사용하여 브로커에게 보낼 암호화된 연결을 설정합니다.

인증, 권한 부여, 암호화 및 기타 보안이 설정된 통신에 대해 구성 가능한 등록 정보는 [표 2-6](#)에서 확인할 수 있습니다(등록 정보 구성 지침은 [5장, "브로커 시작 및 구성"](#) 참조).

표 2-6 보안 관리자 등록 정보

등록 정보 이름	설명
imq.authentication.type	비밀번호를 기본 64 코딩(basic)으로 전달할지 MD5 다이제스트(digest)로 전달할지 여부를 지정합니다. 브로커가 지원하는 모든 연결 서비스에 대한 인코딩을 설정합니다. 기본값: digest
imq.service_name.authentication.type	비밀번호를 기본 64 코딩(basic)으로 전달할지 MD5 다이제스트(digest)로 전달할지 여부를 지정합니다. 명명된 연결 서비스의 인코딩을 설정하며, 이 값은 브로커 전체에 대한 설정을 무시합니다. 기본값: imq.authentication.type 값을 상속합니다.
imq.authentication.basic.user_repository	(기본 64 코딩에 대해) 인증에 사용할 사용자 저장소 유형을 지정하며, 파일 기반(file) 또는 LDAP (ldap) 중 하나입니다. 추가적인 LDAP 등록 정보에 대해서는 210페이지의 표 8-5 를 참조하십시오. 기본값: file
imq.authentication.client.response.timeout	클라이언트가 브로커로부터의 인증 요청에 응답할 때까지 시스템이 대기할 시간(초)을 지정합니다. 기본값: 180(초)
imq.accesscontrol.file.enabled	브로커가 지원하는 모든 연결 서비스에 대해 액세스 제어(true/false)를 설정합니다. 액세스 제어 등록 정보 파일에 지정된 대로 인증된 사용자가 연결 서비스를 사용하거나 특정 대상에 대해 특정 Message Queue 작업을 수행할 권한이 있는가를 시스템이 확인할 것인지 여부를 표시합니다. 기본값: true

표 2-6 보안 관리자 등록 정보(계속)

등록 정보 이름	설명
imq.service_name. accesscontrol.enabled	명명된 연결 서비스의 액세스 제어를 설정하며(true/false), 이 값은 브로커 전체에 대한 설정을 무시합니다. 액세스 제어 등록 정보 파일에 지정된 대로 인증된 사용자가 명명된 연결 서비스를 사용하거나 특정 대상에 대해 특정 Message Queue 작업을 수행할 권한이 있는가를 시스템이 확인할 것인지 여부를 표시합니다. 기본값: imq.accesscontrol.enabled 값을 상속합니다.
imq.accesscontrol.file. filename	브로커가 지원하는 모든 연결 서비스에 대한 액세스 제어 등록 정보 파일의 이름을 지정합니다. 파일 이름은 액세스 제어 디렉토리에 대한 상대 파일 경로를 지정합니다(부록 A, "Message Queue 데이터의 위치" 참조). 기본값: accesscontrol.properties
imq.service_name. accesscontrol.file. filename	브로커 인스턴스의 명명된 연결 서비스에 대한 액세스 제어 등록 정보 파일의 이름을 지정합니다. 파일 이름은 액세스 제어 디렉토리에 대한 상대 파일 경로를 지정합니다(부록 A, "Message Queue 데이터의 위치" 참조). 기본값: imq.accesscontrol.file.filename 값을 상속합니다.
imq.passfile.enabled	안전한 통신을 위한 사용자 비밀번호(SSL, LDAP, JDBC™)가 passfile에 설정되는지 여부를 지정합니다(true/false). 기본값: false
imq.passfile.dirpath	passfile이 포함된 디렉토리의 경로를 지정합니다(운영 체제에 따라 다름). 기본값: 부록 A, "Message Queue 데이터의 위치"를 참조하십시오.
imq.passfile.name	passfile 이름을 지정합니다. 기본값: passfile
imq.keystore.property_name	SSL 기반 서비스에 대해 SSL 키 저장소와 관련된 보안 등록 정보를 지정합니다. 220페이지의 표 8-8을 참조하십시오.

모니터링 서비스

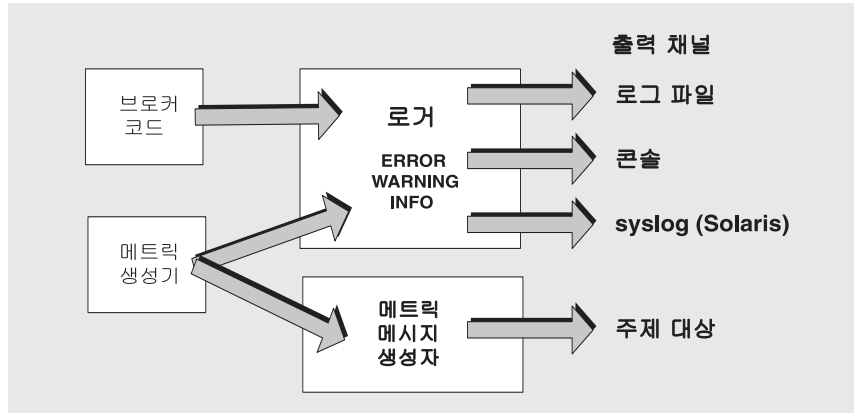
브로커는 로그 작업을 모니터링하고 진단할 다양한 구성 요소를 포함합니다. 다음은 그러한 구성 요소의 예입니다.

- 데이터를 생성하는 구성 요소(이벤트를 기록하는 브로커 코드 및 메트릭 생성자)

- 여러 출력 채널을 통해 정보를 기록하는 로거 구성 요소("로거" 참조)
- 메트릭 정보를 포함하는 JMS 메시지를 JMS 모니터링 클라이언트가 사용할 수 있도록 주제 대상에게 보내는 메시지 생성자

그림 2-6은 그러한 일반 체계를 보여 줍니다.

그림 2-6 모니터링 서비스 지원



메트릭 생성자

메트릭 생성자는 브로커 내부 및 외부로의 메시지 흐름, 브로커 메모리의 메시지 수 및 이 메시지가 사용하는 메모리, 열려 있는 연결 수, 사용 중인 스레드 수 등과 같은 브로커 활동 정보를 제공합니다.

메트릭 데이터 생성을 설정 또는 해제하고 메트릭 보고서 생성 빈도를 지정할 수 있습니다.

로거

Message Queue 로거는 브로커 코드 및 메트릭 생성자가 생성한 정보를 가져와서 표준 출력(콘솔), 로그 파일, syslog 데몬 프로세스(Solaris™ 플랫폼인 경우) 등과 같은 여러 출력 채널에 기록합니다.

로거에서 수집되는 정보 유형과 각 출력 채널에 기록되는 유형을 지정할 수 있습니다.

예를 들어, 가장 심각하고 중요한 정보(오류)에서 중요도가 상대적으로 낮은 정보(메트릭 데이터)까지 로거 수준(로거에서 수집되는 정보 유형)을 지정할 수 있습니다. 표 2-7은 정보 범주를 심각도 내림차순으로 표시합니다.

표 2-7 로깅 범주

범주	설명
ERROR	시스템 오류가 발생할 수 있는 문제에 대한 메시지
WARNING	주의해야 하지만 시스템 오류는 발생하지 않을 경고
INFO	메트릭 및 기타 정보 메시지 보고

로거 수준을 설정하려면 이 범주 중 하나를 지정합니다. 로거는 지정된 범주 및 모든 상위 범주의 데이터를 기록합니다. 예를 들어, **WARNING** 수준의 로깅을 지정한 경우 로거는 경고 정보 및 오류 정보를 기록합니다.

각 출력 채널마다 로거에 대해 설정된 범주 중 어느 것이 해당 채널에 기록될 것인지 지정할 수 있습니다. 예를 들어, 로거 수준이 **INFO**로 설정된 경우 콘솔에는 오류와 경고만, 로그 파일에는 정보(메트릭 데이터)만 기록하게 지정할 수 있습니다(Solaris **syslog** 구성 및 사용에 대한 정보는 **syslog(1M)**, **syslog.conf(4)** 및 **syslog(3C)** 설명서 페이지 참조).

로그 파일의 경우 로그 파일을 닫고 출력을 새 파일로 롤오버하는 지점을 지정할 수 있습니다. 로그 파일이 지정된 크기나 표시 시간에 도달하면 이 파일을 저장하고 새 로그 파일을 작성합니다. 로그 파일은 로그 파일이 연관되어 있는 브로커 인스턴스의 이름 (*instanceName*)으로 식별되는 디렉토리에 기록됩니다(부록 A, "Message Queue 데이터의 위치" 참조).

`.../instances/instanceName/log/`

새 롤오버 로그 파일을 작성하면서 가장 최신의 로그 파일 9개로 구성된 아카이브를 보존합니다.

로거 구성에 대한 자세한 내용은 74페이지의 표 2-9 및 148페이지의 "로거 구성 변경"을 참조하십시오.

메트릭 메시지 생성자(엔터프라이즈판)

메시지 생성자 구성 요소는 메트릭 생성자 구성 요소로부터 일정 간격으로 정보를 받아서 메시지로 기록한 다음 메시지에 포함된 메트릭 정보 유형에 따라 여러 메트릭 주제 대상 중 하나로 보냅니다.

다섯 개의 메트릭 주제 대상이 있으며 표 2-8에 그 이름과 함께 각 대상에 전달되는 메트릭 메시지 유형이 표시되어 있습니다.

표 2-8 메트릭 주제 대상

주제 대상 이름	메트릭 메시지 유형
mq.metrics.broker	브로커 메트릭
mq.metrics.jvm	Java 가상 머신 메트릭
mq.metrics.destination_list	대상 및 대상 유형 목록
mq.metrics.destination.queue. monitoredDestinationName	지정한 이름의 대기열에 대한 대상 메트릭
mq.metrics.destination.topic. monitoredDestinationName	지정한 이름의 주제에 대한 대상 메트릭

이러한 메트릭 주제 대상에 가입한 Message Queue 클라이언트는 대상의 메시지를 사용하고 메시지에 포함된 메트릭 정보를 처리합니다. 예를 들어, 클라이언트가 mq.metrics.broker 대상에 가입하여 브로커의 총 메시지 수와 같은 정보를 받아서 처리할 수 있습니다.

메트릭 메시지 생성자는 메트릭 데이터에 해당하는 이름-값 쌍이 포함된 메시지 (MapMessage 유형)를 만드는 내부 Message Queue 클라이언트입니다. 이러한 메시지는 해당 메트릭 주제 대상에 한 명 이상의 가입자가 있는 경우에만 생성됩니다.

메트릭 메시지 생성자가 생성하는 메시지는 MapMessage 유형이며 포함된 메트릭 유형에 따라 여러 이름/값 쌍으로 구성됩니다. 각 이름/값 쌍은 메트릭 수량과 값에 해당합니다. 예를 들어, 브로커 메트릭 메시지는 브로커를 오가는 메시지 수와 이 메시지의 크기, 현재 메모리에 있는 메시지의 수 및 크기 등 약 12가지의 메트릭 수량 값을 포함합니다. 각 메트릭 메시지 유형에서 보고하는 메트릭 수량에 대한 자세한 내용은 메트릭 메시지를 사용할 Message Queue 클라이언트 작성 방법을 설명하는 *Message Queue Java Client Developer's Guide*를 참조하십시오.

메트릭 메시지 본문에 포함된 메트릭 정보 외에 각 메시지의 헤더에는 메트릭 메시지 유형을 지정하는 등록 정보와 타임스탬프를 기록하는 등록 정보의 두 가지 등록 정보가 있습니다. 이 헤더 등록 정보는 Message Queue 클라이언트 응용 프로그램이 메트릭 메시지에서 데이터를 추출하고 해당 타임스탬프를 기록하는 데 사용될 수 있습니다.

모니터링 서비스 등록 정보

브로커의 정보 생성, 로깅 및 메트릭 메시지 생성을 설정하는 구성 가능한 등록 정보는 표 2-9에서 확인할 수 있습니다(등록 정보 구성 지침은 5장, "브로커 시작 및 구성" 참조).

표 2-9 모니터링 서비스 등록 정보

등록 정보 이름	설명
imq.metrics.enabled	메트릭 정보를 로거에 기록할지 여부를 지정합니다 (true/false). 메트릭 메시지 생성에는 영향을 미치지 않습니다(imq.metrics.topic.enabled 참조). 기본값: true
imq.metrics.interval	메트릭 로깅이 사용 가능한 경우 (imq.metrics.enabled=true) 메트릭 정보가 로거에 기록되는 간격(초)을 지정합니다. 값이 -1이면 기록하지 않습니다. 메트릭 메시지 생성 간격에는 영향을 미치지 않습니다(imq.metrics.topic.interval 참조). 기본값: -1
imq.log.level	로거 수준, 즉 출력 채널에 기록 가능한 출력 범주를 지정합니다. 지정된 범주 및 모든 상위 범주가 함께 포함됩니다. 값은 내림차순으로 ERROR, WARNING, INFO입니다. 기본값: INFO
imq.log.file.output	로그 파일에 기록하는 로깅 정보의 범주를 지정합니다. 사용 가능한 값은 세로 막대()로 구분된 임의의 로깅 범주 집합, ALL 또는 NONE입니다. 기본값: ALL
imq.log.file.dirpath	로그 파일이 포함된 디렉토리의 경로를 지정합니다(운영 체제에 따라 다름). 기본값: 부록 A, "Message Queue 데이터의 위치"를 참조하십시오.
imq.log.file.filename	로그 파일의 이름을 지정합니다. 기본값: log.txt

표 2-9 모니터링 서비스 등록 정보(계속)

등록 정보 이름	설명
imq.log.file.rolloverbytes	새 로그 파일로 출력을 롤오버할 로그 파일 크기(바이트)를 지정합니다. 값 -1은 파일 크기를 기준으로 하는 롤오버가 없음을 의미합니다. 기본값: -1
imq.log.file.rolloversecs	새 로그 파일로 출력을 롤오버할 로그 파일 표시 시간(초)을 지정합니다. 값 -1은 파일 표시 시간을 기준으로 하는 롤오버가 없음을 의미합니다. 기본값: 604800 (1주)
imq.log.console.output	콘솔에 기록하는 로깅 정보의 범주를 지정합니다. 사용 가능한 값은 세로 막대()로 구분된 임의의 로깅 범주 집합, ALL 또는 NONE입니다. 기본값: ERROR WARNING
imq.log.console.stream	콘솔 출력을 stdout (OUT) 또는 stderr (ERR)에 기록할 것인지 지정합니다. 기본값: ERR
imq.log.syslog.facility	(Solaris에만 적용) Message Queue 브로커가 어떤 syslog 기능으로 로깅할 것인지 지정합니다. 값은 syslog(3C) 설명서 페이지에 있는 값을 미리 설정합니다. Message Queue에서 사용 가능한 값은 LOG_USER, LOG_DAEMON 그리고 LOG_LOCAL0부터 LOG_LOCAL7까지입니다. 기본값: LOG_DAEMON
imq.log.syslog.logpid	(Solaris에만 적용) 브로커 프로세스 아이디를 메시지와 함께 로깅할 것인지 여부를 지정합니다(true/false). 기본값: true
imq.log.syslog.logconsole	(Solaris에만 적용) 메시지를 syslog에 보낼 수 없을 경우 시스템 콘솔에 기록할 것인지 여부를 지정합니다(true/false). 기본값: false
imq.log.syslog.identity	(Solaris에만 적용) syslog에 기록되는 모든 메시지의 앞에 추가될 아이디 문자열을 지정합니다. 기본값: 브로커 인스턴스 이름 앞에 imqbrokerd가 붙습니다.
imq.log.syslog.output	(Solaris에만 적용) syslogd(1M)에 기록하는 로깅 정보의 범주를 지정합니다. 사용 가능한 값은 세로 막대()로 구분된 임의의 로깅 범주 집합, ALL 또는 NONE입니다. 기본값: ERROR

표 2-9 모니터링 서비스 등록 정보(계속)

등록 정보 이름	설명
<code>imq.log.timezone</code>	로그 타임스탬프의 표준 시간대를 지정합니다. 식별자는 <code>java.util.TimeZone.getTimeZone()</code> 에 사용되는 식별자와 동일합니다. 예: GMT, 미국/로스앤젤레스, 유럽/로마, 아시아/도쿄. 기본값: 지역 표준 시간대
<code>imq.metrics.topic.enabled</code>	메트릭 메시지 생성이 가능한지 여부를 지정합니다 (<code>true/false</code>). <code>false</code> 인 경우 메트릭 주제 대상에 가입하려고 하면 클라이언트측 예외가 발생합니다. 기본값: <code>true</code>
<code>imq.metrics.topic.interval</code>	메트릭 메시지를 생성하여 메트릭 주제 대상에 보내는 간격(초)을 지정합니다. 기본값: <code>60</code>
<code>imq.metrics.topic.persist</code>	메트릭 메시지가 지속성인지 여부를 지정합니다 (<code>true/false</code>). 기본값: <code>false</code>
<code>imq.metrics.topic.timetolive</code>	메트릭 주제 대상에 보낸 메트릭 메시지의 수명(초)을 지정합니다. 기본값: <code>300</code>

물리적 대상

Message Queue 메시징은 2단계 메시지 전달을 기반으로 합니다. 먼저 생성자 클라이언트에서 브로커의 대상으로 메시지가 전달되고, 두 번째로는 브로커의 대상에서 하나 이상의 사용자 클라이언트로 메시지가 전달됩니다. 대상에는 2가지 유형이 있는데(44페이지의 "프로그래밍 도메인" 참조), 대기열(지점간 전달 모델)과 주제(게시/가입 전달 모델)입니다. 이러한 대상은 브로커의 물리적 메모리에서의 위치를 나타내며, 여기서는 받은 메시지가 사용자 클라이언트에게 라우팅되기 전에 마샬링됩니다.

Message Queue 관리 도구(167페이지의 "연결 정보 얻기" 참조)를 사용하여 물리적 대상을 작성합니다. 또한 78페이지의 "자동 작성(대 관리 작성) 대상"의 설명처럼 자동으로 대상을 작성할 수도 있습니다.

이 절에서는 2가지 물리적 대상 유형, 즉 대기열과 주제의 등록 정보 및 동작에 대해 설명합니다.

대기열 대상

대기열 대상은 대상에서 인터레스트를 등록한 여러 사용자 중 단 한 명에게 메시지가 전달되는 지점간 메시징에서 사용합니다. 메시지 생성자로부터 도착한 메시지는 대기열에 있다가 단일 메시지 사용자에게 전달됩니다.

다중 사용자로의 대기열 전달

대기열 대상의 메시지는 단일 사용자에게만 전달되지만 Message Queue에서는 다중 사용자가 하나의 대기열에 등록할 수 있습니다. 브로커는 받는 메시지를 여러 사용자에게 라우팅하여 사용자 간에 로드 균형을 조정합니다.

다중 사용자로의 대기열 전달 구현에서는 다음 대기열 대상 속성을 기반으로 구성 가능한 로드 균형 조정 방법을 사용합니다.

- **maxNumActiveConsumers:** 로드 균형 조정 대기열 전달에서 활성 상태인 사용자 수(한 명 또는 여러 명)를 지정합니다.
- **maxNumBackupConsumers:** 활성 사용자가 실패할 경우에 활성 사용자를 대신할 수 있는 백업 사용자 수(없음 또는 여러 명)를 지정합니다.

사용자 수가 두 속성의 합계를 초과할 경우 새 사용자가 거부됩니다(Message Queue 플랫폼은 대기열당 최대 3명의 사용자(2명의 활성 사용자와 1명의 백업 사용자)를 지원하고 Message Queue 엔터프라이즈판은 제한이 없음).

로드 균형 조정 메커니즘에서는 여러 사용자의 메시지 사용 속도를 고려합니다. 작동 방식은 다음과 같습니다.

구성 가능한 크기(대기열 대상의 `consumerFlowLimit` 속성)의 일괄 처리에서 대기열 대상의 메시지를 사용 가능한 새 활성 사용자에게 대기열에 등록된 순서에 따라 라우팅합니다. 이 메시지를 전달한 후 사용자가 사용 가능해지면(즉, 사용자가 이전에 전달 받은 메시지의 구성 가능한 비율을 사용하면) 대기열에 도착하는 추가 메시지를 일괄 처리로 사용자에게 라우팅합니다. 각 사용자의 디스패치 속도는 사용자의 현재 용량과 메시지 처리 속도에 따라 다릅니다.

메시지 생성 속도가 느린 경우 브로커가 활성 사용자들에게 메시지를 고르지 않게 디스패치할 수 있습니다. 활성 사용자가 필요 이상으로 많은 경우 메시지를 받지 못하는 사용자도 있을 수 있습니다.

활성 사용자가 실패하면 첫 번째 백업 사용자가 활성화되어 실패한 사용자의 작업을 대신 수행합니다. 대기열 대상에 둘 이상의 활성 사용자가 있는 경우 메시지 사용 순서가 지켜지지 않을 수 있습니다.

브로커 클러스터 환경에서는 다중 사용자로의 전달 시 로컬 사용자를 우선하도록 설정할 수 있습니다. 대기열 대상 속성 `localDeliveryPreferred`를 사용하면 생성자의 홈 브로커, 즉 생성자가 메시지를 보낸 브로커(로컬 브로커)에 사용자가 없는 경우에만 메시지를 원격 사용자에게 전달하도록 지정할 수 있습니다. 그러면 *원격 사용자*의 홈 브로커를 통해 원격 사용자에게 라우팅할 때 처리량이 떨어질 수 있는 상황에서 성능을 향상시킬 수 있습니다. 이 속성을 사용하려면 대상 범위가 로컬 전용 전달로 제한되지 않아야 합니다(171페이지의 표 6-10 참조).

메모리 고려 사항

메시지가 대기열에 오래 머무를 수 있으므로 메모리 자원이 문제가 되기도 합니다. 대기열에 메모리를 너무 많이(메모리는 저활용 상태) 또는 너무 적게(메시지가 거부됨) 할당하기를 원치 않을 것입니다. 유연성을 위해 각 대기열의 로드 수요에 따라 대기열 생성 시 대기열의 메시지 최대 수, 대기 메시지에 할당되는 최대 메모리, 대기 메시지의 최대 크기 등의 물리적 등록 정보를 설정할 수 있습니다(171페이지의 표 6-10 참조).

주제 대상

주제 대상은 대상에서 인터레스트를 등록한 모든 사용자에게 메시지가 전달되는 게시/가입 메시징에서 사용합니다. 생성자로부터 메시지가 도착하면 해당 주제에 가입한 모든 사용자에게 라우팅됩니다. 사용자가 해당 주제에 대한 영구 가입에 등록한 경우, 메시지가 주제에 전달되는 시점에 사용자는 활성 상태가 아니어도 됩니다. 브로커는 이 사용자가 활성화될 때까지 메시지를 보관했다가 전달합니다.

일반적으로 메시지는 주제 대상에 오래 머무르지 않으므로 메모리 자원은 큰 문제가 되지 않는 편입니다. 그러나 대상이 수신하는 메시지에 허용되는 최대 크기를 구성할 수 있습니다(171페이지의 표 6-10 참조).

자동 작성(대 관리 작성) 대상

Message Queue 메시지 서버는 메시징 시스템에서 중앙 허브가 되므로 이 서버의 성능 및 안정성이 성공적인 엔터프라이즈 응용 프로그램에 있어 중요한 요소입니다. 대상은 (처리하는 메시지의 수 및 크기, 등록하는 메시지 사용자의 수 및 지속성에 따라) 많은 자원을 사용할 수 있으므로 메시지 서버 성능 및 안정성을 보장하도록 면밀하게 관리해야 합니다. 따라서 관리자가 응용 프로그램을 대신하여 대상을 만들고 대상을 모니터링하며 필요 시 자원 요구 사항을 재구성하는 것이 표준 방식입니다.

그러나 대상을 동적으로 작성하는 것이 바람직한 경우가 있습니다. 예를 들어 개발 및 테스트 주기에서는 관리자가 개입할 필요 없이 필요에 따라 브로커가 자동으로 대상을 작성하는 것이 나올 수 있습니다.

Message Queue는 이 자동 작성기능을 지원합니다. 자동 작성이 가능한 경우 브로커는 메시지 사용자나 메시지 생성자가 존재하지 않는 대상에 액세스를 시도할 때마다 자동으로 대상을 작성합니다(클라이언트 응용 프로그램의 사용자는 자동 작성 권한이 있어야 합니다. [217페이지의 "대상 자동 작성 액세스 제어"](#) 참조).

그러나 대상이 명시적이 아니라 자동으로 작성될 경우, (동일한 대상 이름을 사용하는) 서로 다른 클라이언트 응용 프로그램이 충돌하거나 (대상 지원에 필요한 자원 때문에) 시스템 성능이 저하될 수 있습니다. 이런 이유로 자동 작성 대상은 더 이상 사용되지 않을 경우, 즉 더 이상 메시지 사용자 클라이언트가 없거나 어떤 메시지도 포함하지 않는 경우 브로커에 의해 자동으로 삭제됩니다. 브로커가 다시 시작되면 지속성 메시지가 있는 경우에만 자동 작성 대상을 다시 작성합니다.

[표 2-10](#)에 있는 등록 정보를 사용하여 자동 작성 기능이 사용 가능/불가능하도록 Message Queue 메시지 서버를 구성할 수 있습니다(등록 정보 구성 지침은 [5장, "브로커 시작 및 구성"](#) 참조).

표 2-10 자동 작성 구성 등록 정보

등록 정보 이름	설명
imq.autocreate.topic	브로커가 주제 대상을 자동 작성할 수 있는지 여부를 지정합니다(true/false). 기본값: true
imq.autocreate.queue	브로커가 대기열 대상을 자동 작성할 수 있는지 여부를 지정합니다(true/false). 기본값: true
imq.autocreate.destination. maxNumMsgs	자동 작성된 대상에 허용되는 사용되지 않은 최대 메시지 수를 지정합니다. 기본값: 100,000

표 2-10 자동 작성 구성 등록 정보(계속)

등록 정보 이름	설명
imq.autocreate.destination. maxTotalMsgBytes	대상의 사용되지 않은 메시지에 허용되는 최대 메모리 합계(바이트)를 지정합니다. 기본값: 10m (MB)
imq.autocreate.destination. limitBehavior	메모리 제한 임계값에 도달할 경우에 브로커가 응답하는 방법을 지정합니다. 값은 다음과 같습니다. FLOW_CONTROL - 생성자 수를 조금씩 줄입니다. REMOVE_OLDEST - 가장 오래된 메시지를 삭제합니다. REMOVE_LOW_PRIORITY - 메시지 보존 기간을 기준으로 우선 순위가 가장 낮은 메시지를 삭제합니다. REJECT_NEWEST - 최신 메시지를 거부합니다. 기본값: REJECT_NEWEST
imq.autocreate.destination. maxBytesPerMsg	자동 작성된 대상에 허용되는 단일 메시지의 최대 크기(바이트)를 지정합니다. 기본값: 10k (10,240)
imq.autocreate.destination. maxNumProducers	대상에 허용되는 최대 생성자 수를 지정합니다. 이 제한에 도달하면 새로운 생성자가 생성되지 않습니다. 기본값: 100
imq.autocreate.destination. isLocalOnly	브로커 클러스터에만 적용됩니다. 대상을 다른 브로커에 복사하지 않도록 지정합니다. 따라서, 메시지 전달이 로컬 사용자(대상이 생성되는 브로커에 연결된 사용자)에게만 제한됩니다. 대상이 생성된 이후에는 이 속성을 업데이트할 수 없습니다. 기본값: false
imq.autocreate.queue. maxNumActiveConsumers	자동 생성된 대기열 대상으로부터의 로드 균형 조정 전달에서 활성 상태가 될 수 있는 최대 사용자 수를 지정합니다. 값 -1은 무제한을 의미합니다. 기본값: 1
imq.autocreate.queue. maxNumBackupConsumers	자동 작성된 대기열 대상으로부터의 로드 균형 조정 전달 중에 오류가 발생할 경우 활성 사용자를 대신할 수 있는 최대 백업 사용자 수를 지정합니다. 값 -1은 무제한을 의미합니다. 기본값: 0

표 2-10 자동 작성 구성 등록 정보(계속)

등록 정보 이름	설명
imq.autocreate.queue. consumerFlowLimit	일괄적으로 사용자에게 전달되는 최대 메시지 수를 지정합니다. 로드 균형 조정된 대기열 전달에서 이 수는 로드 균형 조정을 시작하기 전에 활성 사용자에게 라우팅되는 초기 대기 메시지 수입니다(77페이지의 "다중 사용자로의 대기열 전달" 참조). 이 제한은 해당 연결에서 대상 사용자에게 설정된 낮은 값으로 대체될 수 있습니다(Message Queue Java Client Developer's Guide 의 연결 팩토리 속성 참조). 값 -1은 무제한을 의미합니다. 기본값: 1000
imq.autocreate.topic. consumerFlowLimit	일괄적으로 사용자에게 전달되는 최대 메시지 수를 지정합니다. 값 -1은 무제한을 의미합니다. 기본값: 1,000
imq.autocreate.queue. localDeliveryPreferred	브로커 클러스터의 로드 균형 조정된 대기열 전달에만 적용됩니다. 로컬 브로커에 사용자가 없는 경우에만 원격 사용자에게 메시지를 전달하도록 지정합니다. 자동 작성된 대상을 로컬에만 전달로 제한해서는 안 됩니다(<code>isLocalOnly = false</code>). 기본값: false

임시 대상

임시 대상은 다른 클라이언트에게 보낸 메시지의 응답을 받을 대상이 필요한 클라이언트가 (JMS API를 사용하여) 명시적으로 만들고 삭제합니다. 이러한 대상은 작성 시 해당 연결이 지속되는 동안 브로커에 의해 유지 관리됩니다. 임시 대상은 관리자가 삭제할 수 없습니다. 그리고 사용 중이라면, 즉 활성 메시지 사용자가 있는 경우에는 클라이언트 응용 프로그램에서도 삭제할 수 없습니다. 임시 대상은 (지속성 메시지가 있는) 관리 작성되거나 자동 작성된 대상과 달리 영구 저장되지 않으며 브로커가 다시 시작할 때 다시 작성되지 않지만, Message Queue 관리 도구에 표시될 수 있습니다([168페이지의 표 6-9](#) 참조).

멀티 브로커 클러스터(엔터프라이즈판)

Message Queue 엔터프라이즈판은 상호 연결된 여러 브로커 인스턴스들, 즉 브로커 클러스터를 사용하는 메시지 서버 구현을 지원합니다. 클러스터 지원은 메시지 서버의 확장성을 제공합니다.

브로커에 연결된 클라이언트 수가 늘어나고 전달되는 메시지 수가 늘어나면 브로커는 과일 설명자, 메모리 제한과 같은 자원 제한 사항을 초과하게 됩니다. 늘어나는 로드를 수용하는 한 가지 방법은 Message Queue 메시지 서버에 브로커(브로커 인스턴스)를 추가하여 클라이언트 연결 및 메시지 전달을 여러 브로커에 걸쳐 분산시키는 것입니다.

또한 여러 브로커를 사용하여 네트워크 대역폭을 최적화할 수 있습니다. 예를 들어, 원격 브로커들 사이에는 더 느린 장거리 네트워크 링크를 사용하고 클라이언트를 개별 브로커 인스턴스에 연결할 경우에는 더 빠른 링크를 사용할 수 있습니다.

브로커 클러스터를 사용하는 또 다른 이유들이 있지만(예: 사용자 저장소가 다른 작업 그룹 수용, 방화벽 제한 사항 처리) 페일오버는 해당되지 *않습니다*. Message Queue에서는 클러스터의 다른 브로커를 사용하여 실패한 연결을 다시 설정할 수 있지만 상태 정보는 손실됩니다. 따라서 클러스터의 한 브로커를 실패한 다른 브로커의 자동 백업으로 사용할 수 없습니다.

즉, Message Queue에서는 현재고가용성 메시지 서버를 지원하지 않습니다. 그러나 Sun Cluster 소프트웨어와고가용성 데이터베이스를 사용하여 브로커 페일오버에 대비할 수 있습니다. 멀티 브로커를 사용하도록 메시징 응용 프로그램을 설계하여 사용자 정의 페일오버 솔루션을 구현할 수도 있습니다.

브로커 클러스터 구성 및 관리에 대해서는 [140페이지의 "클러스터를 이용한 작업\(엔터프라이즈판\)"](#)을 참조하십시오.

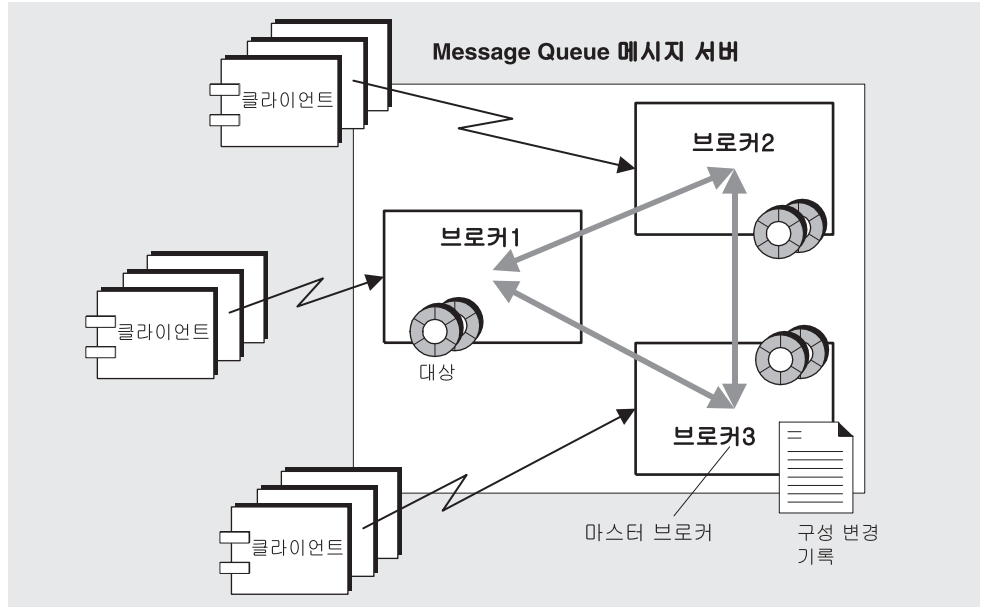
다음 절에서는 Message Queue 브로커 클러스터의 구조 및 내부 기능에 대해 설명합니다.

멀티 브로커 구조

멀티 브로커 메시지 서버에서는 [그림 2-7](#)과 같이 여러 브로커 인스턴스 사이에 클라이언트 연결을 분산시킬 수 있습니다. 클라이언트 관점에서 각 클라이언트는 개별 브로커(자신의 홈브로커)와 연결되며, 홈 브로커가 클러스터에서 유일한 브로커인 것처럼 메시지를 보내고 받습니다. 그러나 메시지 서버 관점에서 이 홈 브로커는 클러스터의 다른 브로커들과 탠덤 방식으로 작동하면서 직접 연결된 메시지 생성자 및 사용자에게 전달 서비스를 제공합니다.

이론적으로는 클러스터 내부의 브로커가 임의의 토폴로지로 연결할 수 있습니다. 그러나 Message Queue는 완전 연결된 클러스터, 즉 **그림 2-7**에서처럼 각 브로커가 클러스터의 다른 모든 브로커와 직접 연결되는 토폴로지만 지원합니다.

그림 2-7 멀티 브로커(클러스터) 구조



메시지 전달

멀티 브로커 구성에서는 각 대상이 클러스터의 모든 브로커에 복제됩니다(몇 가지 예외가 있지만 클러스터 환경의 대상 속성은 일반적으로 대상의 모든 인스턴스에 즉, 개별 대상 인스턴스가 아니라 전체 클러스터에 **집합적으로** 적용됩니다. 또한, `isLocalOnly` 속성이 `true`로 설정된 대상은 클러스터에 복제되지 않습니다. 자세한 내용은 [171페이지의 표 6-10](#)의 대상 속성 설명 참조).

각 브로커는 다른 모든 브로커의 대상에 등록된 메시지 사용자에게 대해 알고 있습니다. 따라서 각 브로커는 자신과 직접 연결된 메시지 생성자로부터 원격 메시지 사용자에게 메시지를 라우팅하고, 원격 생성자로부터 자신과 직접 연결된 사용자에게 메시지를 전달합니다.

클러스터 구성에서 각 메시지 생성자와 직접 연결된 브로커는 해당 사용자가 자신에게 보내는 메시지에 대해 라우팅을 수행합니다. 따라서 지속성 메시지는 그 메시지의 홈 브로커에 의해 저장되고 라우팅됩니다.

클러스터에서 브로커 간의 트래픽을 최소화하기 위해 사용자 연결의 흐름 제어 메커니즘을 통해 대상에서 클라이언트 런타임으로의 메시지 전달을 규제합니다. 이 방법에서는 메시지를 대상 브로커에 연결된 사용자에게 전달해야 하는 경우에만 메시지를 한 브로커에서 다른 브로커로 보내므로 브로커 간의 불필요한 메시지 전달이 방지됩니다. 또한 여러 사용자로의 대기열 전달과 같은 몇 가지 경우에는 로컬 사용자로의 전달이 원격 사용자로의 전달보다 우선 순위를 갖도록 지정하여 브로커간 트래픽을 최소화할 수 있습니다(171 페이지의 표 6-10의 `localDeliveryPreferred` 대기열 대상 속성 참조).

클라이언트와 메시지 서버 간에 암호화된 보안 메시지 전달이 필요한 경우 클러스터에서 브로커 간의 메시지 전달도 보안하도록 클러스터를 구성할 수 있습니다(143 페이지의 "브로커간 연결 보안" 참조).

클러스터 동기화

관리자가 브로커에서 대상을 작성하거나 삭제할 때마다 이 정보는 클러스터의 다른 모든 브로커들에게 자동 전파됩니다. 이와 비슷하게 메시지 사용자가 홈 브로커에 등록되거나 사용자와 홈 브로커 사이의 연결이 (명시적으로 또는 클라이언트나 네트워크 오류로 인해 또는 홈 브로커가 중단되어) 끊길 때마다 이 사용자에 대한 관련 정보가 클러스터 전체에 전파됩니다. 또한 영구 가입에 대한 정보도 클러스터의 모든 브로커에게 전파됩니다.

주 과도한 네트워크 트래픽 및/또는 대용량 메시지는 내부 클러스터 연결에 지장을 줄 수 있습니다. 대기 시간이 길어지면 잠금 프로토콜 시간 초과 오류가 발생하기도 합니다. 그 결과 클라이언트가 영구 가입자나 대기열 메시지 사용자를 작성하려고 할 때 예외가 발생할 수 있습니다. 일반적으로 이러한 문제는 더 빠른 연결을 사용하여 방지할 수 있습니다.

특정 브로커에게 대상과 메시지 사용자에 대한 정보를 전파하려면 일반적으로 공유 자원이 변경될 때 브로커가 온라인 상태가 되어야 합니다. 그러한 변경이 이루어질 때 브로커가 오프라인 상태라면(예: 브로커가 충돌하거나 나중에 다시 시작하거나 클러스터에 새 브로커가 동적으로 추가되는 경우) 어떻게 될까요?

오프라인된 브로커(또는 새로 추가된 브로커)를 수용하기 위해 Message Queue는 클러스터의 모든 지속성 항목에 대한 변경 기록, 즉 작성되거나 삭제된 모든 대상 및 영구 가입에 대한 기록을 관리합니다. 클러스터에 동적으로 브로커가 추가되면 이 브로커는 먼저 이 *구성 변경 기록*에서 대상 및 영구 가입자 정보를 읽습니다. 브로커가 온라인 상태가 되면 다른 브로커들과 현재 활성 상태인 사용자에 대한 정보를 교환합니다. 새 브로커는 이 정보를 사용하여 클러스터에 완전히 통합됩니다.

구성 변경 기록은 클러스터에서 *마스터 브로커*로 지정된 브로커가 관리합니다. 마스터 브로커는 클러스터에 브로커를 동적으로 추가하는 데 핵심적인 역할을 하므로 항상 이 브로커를 먼저 시작해야 합니다. 마스터 브로커가 온라인이 아닌 경우 클러스터의 다른 브로커들은 초기화를 완료할 수 없습니다.

마스터 브로커가 오프라인이 되면 구성 변경 기록은 다른 브로커에서 액세스할 수 없으며, Message Queue는 대상 및 영구 가입이 클러스터에 전파되지 않게 합니다. 이 상태에서 대상이나 영구 가입을 작성하거나 삭제하려고 하면(또는 영구 가입 재활성화와 같은 다양한 관련 작업을 시도할 경우) 예외가 발생합니다.

중요한 응용 프로그램 환경에서는 구성 변경 기록을 정기적으로 백업하여 사고에 의한 기록 손상 및 마스터 브로커 오류를 방지하는 것이 바람직합니다. 구성 변경 기록을 비롯한 백업 파일 작성 방법을 제공하는 `mqbrokerd` 명령의 `-backup` 옵션을 사용하면 됩니다(136페이지의 표 5-2 참조). 나중에 `-restore` 옵션을 사용하여 구성 변경 기록을 복원할 수 있습니다.

필요하다면 마스터 브로커로 제공되는 브로커를 변경할 수 있습니다. 구성 변경 기록을 백업하고 해당 클러스터 구성 등록 정보(140페이지의 표 5-3 참조)를 수정하여 새 마스터 브로커를 지정하고 `-restore` 옵션을 사용하여 새 마스터 브로커를 다시 시작하면 됩니다.

개발 환경에서의 클러스터 사용

클러스터를 테스트 용도로 사용하고 확장성 및 브로커 복구를 심각하게 고려하지 *않아*도 되는 개발 환경에서는 마스터 브로커가 별로 필요하지 않습니다. 마스터 브로커 *없이* 구성된 환경에서 Message Queue는 다른 브로커를 시작하기 위해 마스터 브로커가 반드시 실행 중이어야 할 필요는 없으며, 대상 및 영구 가입을 변경할 수 있고 이 변경 사항은 클러스터에서 실행 중인 다른 모든 브로커에게 전파될 수 있습니다. 그러나 브로커가 오프라인되었다가 나중에 복원될 경우 오프라인 상태였을 때 변경된 사항은 동기화하지 않습니다.

테스트 환경에서는 일반적으로 대상은 자동 작성되고(78페이지의 "자동 작성(대 관리 작성) 대상" 참조), 이 대상에 대한 영구 가입은 테스트 중인 응용 프로그램에서 작성하고 삭제합니다. 대상 및 영구 가입의 변경 사항은 클러스터 전체에 전파됩니다. 그러나 마스터 브로커를 사용하도록 환경을 재구성할 경우, Message Queue는 대상 및 영구 가입을 변경하고 이 변경 사항이 클러스터 전체에 전파되려면 마스터 브로커가 실행 중이어야 합니다.

클러스터 구성 등록 정보

클러스터의 각 브로커는 시작 시 클러스터의 다른 브로커에 대한 정보(호스트 이름 및 포트 번호)를 전달 받아야 합니다. 이 정보는 클러스터의 브로커간 연결을 설정할 때 사용됩니다. 또한 마스터 브로커를 사용할 경우 각 브로커는 마스터 브로커의 호스트 이름 및 포트 번호를 알고 있어야 합니다.

클러스터의 모든 브로커는 공통 클러스터 구성 등록 정보를 사용해야 합니다. 이 등록 정보를 단일 중앙 *클러스터 구성 파일*에 저장하고 각 브로커가 시작할 때마다 이 파일을 참조하게 하면 됩니다.

또한 클러스터 구성 등록 정보를 복제하여 각 브로커에게 개별적으로 제공할 수 있습니다. 그러나 클러스터 구성의 일관성이 손상될 수 있으므로 바람직한 방법은 아닙니다. 클러스터 구성 등록 정보의 복사본을 하나만 유지하면 모든 브로커가 동일한 정보를 참조할 수 있습니다.

클러스터 구성 등록 정보에 대한 자세한 내용은 140페이지의 "클러스터를 이용한 작업(엔터프라이즈판)"을 참조하십시오.

클러스터 구성 파일은 브로커 집합이 공통적으로 사용하는 모든 브로커 구성 등록 정보를 저장할 때 사용할 수 있습니다. 원래 클러스터를 구성하기 위해 만들어졌지만 클러스터의 모든 브로커에 공통적인 다른 브로커 등록 정보를 저장하는 용도로도 사용 가능합니다.

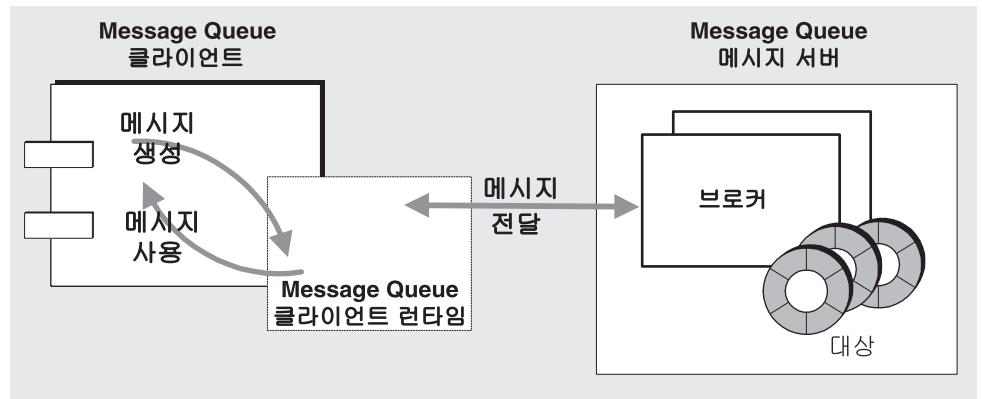
Message Queue 클라이언트 런타임

Message Queue 클라이언트 런타임은 클라이언트 응용 프로그램에 Message Queue 메시지 서비스와의 인터페이스를 제공합니다. 즉, Java 클라이언트 응용 프로그램에 38페이지의 "JMS 프로그래밍 모델"에서 소개한 모든 JMS 프로그래밍 객체를 제공하고 C 클라이언트 응용 프로그램에 해당 C 인터페이스를 제공합니다. Message Queue 클라이언트 런타임은 클라이언트가 대상에게 메시지를 보내고 대상으로부터 메시지를 받는 데 필요한 모든 작업을 지원합니다.

이 절에서는 Message Queue 클라이언트 런타임의 작동 방식을 간략하게 설명합니다. Java 클라이언트 런타임 및 C 클라이언트 런타임의 클라이언트 응용 프로그램 설계 및 성능에 영향을 미치는 요소는 각각 *Message Queue Java Client Developer's Guide* 및 *Message Queue C Client Developer's Guide*에 설명되어 있습니다.

그림 2-8은 메시지 생성 및 사용 시 클라이언트 응용 프로그램과 Message Queue 클라이언트 런타임 사이의 상호 작용 그리고 메시지 전달 시 Message Queue 클라이언트 런타임과 Message Queue 메시지 서버 사이의 상호 작용이 어떻게 이루어지는지 보여 줍니다.

그림 2-8 메시징 작업



메시지 생성

메시지 생성 단계에서는 클라이언트가 메시지를 작성하고 연결을 통해 브로커의 대상으로 전달합니다. MessageProducer 객체의 메시지 전달 모드가 지속성으로 설정된 경우(한 번씩만 전달), 메시지가 대상에 전달되고 브로커의 영구 데이터 저장소에 저장되었음을 브로커가 확인할 때까지 클라이언트 스레드가 차단됩니다. 메시지가 지속성을 갖지 않은 경우 브로커는 브로커 확인 메시지(등록 정보 이름에서 "Ack"라고 표시됨)를 보내지 않으며 클라이언트 스레드는 차단되지 않습니다.

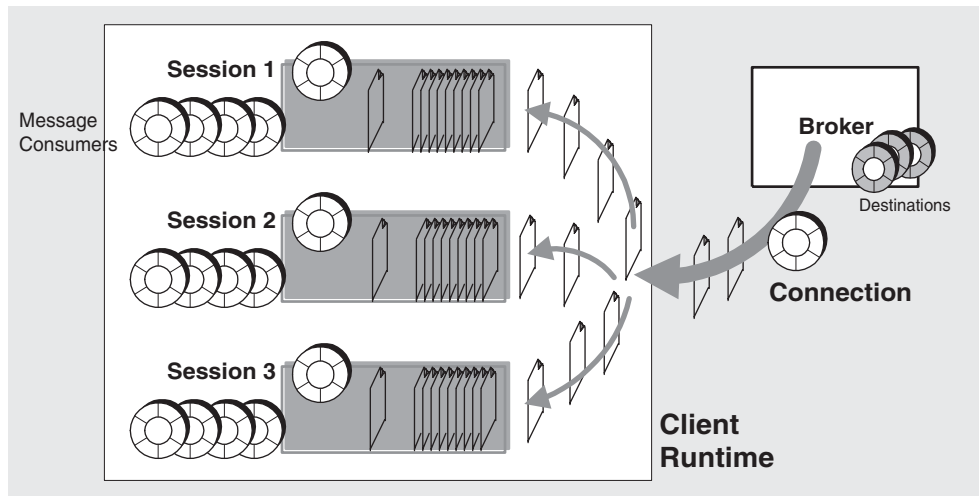
메시지 사용

메시지 사용은 생성보다 더 복잡합니다. 브로커의 대상에 도착한 메시지는 다음 조건에서 Message Queue 클라이언트 런타임과의 연결을 통해 전달됩니다.

- 클라이언트가 특정 대상에 대해 사용자를 설정한 경우
- 사용자의 선택 기준이 있는 경우 이 기준이 특정 대상에 도착한 메시지의 기준과 일치하는 경우
- 연결이 메시지 전달을 시작하도록 통보 받은 경우

연결을 통해 전달된 메시지는 해당 Message Queue 세션으로 배포되고, **그림 2-9**에서처럼 MessageConsumer 객체에 의해 사용될 때까지 대기합니다. 메시지는 각 세션 대기열에서 한 번에 하나씩 전송되며(세션은 단일 스레드로 구성) 동기식(receive 메소드를 호출하는 클라이언트 스레드를 통해) 또는 비동기식(MessageListener 객체의 onMessage 메소드를 호출하는 세션 스레드를 통해)으로 사용됩니다.

그림 2-9 Message Queue 클라이언트 런타임으로의 메시지 전달



브로커가 클라이언트 런타임으로 메시지를 전달할 때 그에 따라 메시지에 표시하지만 실제로 수신 또는 사용되었는지 여부는 알 수 없습니다. 따라서 브로커는 브로커의 대상에서 메시지를 삭제하기 전에 클라이언트가 메시지 수신을 확인할 때까지 기다립니다.

Message Queue 관리 대상 객체

관리 대상 객체를 사용하면 클라이언트 응용 프로그램 코드는 공급자 독립성을 가질 수 있습니다. 이는 클라이언트 응용 프로그램이 공급자와 독립적인 방식으로 사용하는 객체에 공급자별 구현 및 구성 정보를 캡슐화하여 수행합니다. 관리 대상 객체는 관리자가 작성 및 구성하고 이름 서비스에서 저장하며 클라이언트 응용 프로그램이 표준 JNDI 조회 코드를 사용하여 액세스합니다.

Message Queue는 `ConnectionFactory`와 `Destination`의 2가지 관리 대상 객체 유형을 제공합니다. 두 객체 모두 공급자별 정보를 캡슐화하지만, 클라이언트 응용 프로그램 내부에서 그 용도는 매우 다릅니다. `ConnectionFactory` 객체는 메시지 서버와의 연결을 생성할 때 사용하며, `Destination` 객체는 물리적 대상을 식별할 때 사용됩니다.

관리 대상 객체를 사용하면 Message Queue 메시지 서버를 매우 쉽게 제어하고 관리할 수 있습니다.

- 클라이언트 응용 프로그램이 사전 구성된 `ConnectionFactory` 객체에 액세스하도록 요구함으로써 연결의 동작을 제어할 수 있습니다(187페이지의 "연결 팩토리 관리 대상 객체 속성" 참조).
- 클라이언트 응용 프로그램이 기존 물리적 대상과 일치하는 사전 구성된 `Destination` 객체에 액세스하도록 요구함으로써 물리적 대상의 증가를 제어할 수 있습니다(또한 브로커의 자동 작성 기능을 사용 불가능하게 할 수 있습니다. 78페이지의 "자동 작성(대 관리 작성) 대상" 참조).
- 클라이언트 응용 프로그램이 설정한 메시지 헤더 값을 무시함으로써 Message Queue 메시지 서버 자원을 제어할 수 있습니다(187페이지의 "연결 팩토리 관리 대상 객체 속성" 참조).

따라서 이러한 방법을 통해 Message Queue 관리자는 메시지 서버의 구성 세부 정보를 제어함과 동시에 클라이언트 응용 프로그램이 공급자 독립성을 갖도록 할 수 있습니다. 즉 클라이언트 응용 프로그램은 공급자별 구문 및 객체 이름 지정 규약(43페이지의 "JMS 공급자 독립성" 참조) 또는 공급자별 구성 등록 정보를 알 필요가 없습니다.

7장, "관리 대상 객체 관리"에 설명된 것처럼 Message Queue 관리 도구를 사용하여 관리 대상 객체를 만듭니다. 관리 객체를 만들 때 읽기 전용으로, 즉 객체를 만들 때 설정한 Message Queue 특정 구성 값을 클라이언트 응용 프로그램에서 변경하지 못하도록 지정할 수 있습니다. 다시 말해 클라이언트 코드는 읽기 전용 관리 대상 객체에 대해 속성 값을 설정할 수 없으며, 관리자는 91페이지의 "클라이언트 시작 시 속성 값 무시"에서 설명하는 것처럼 응용 프로그램 시작 옵션을 사용하여 이 값을 무시할 수 없습니다.

클라이언트 응용 프로그램에서 `ConnectionFactory`와 `Destination` 관리 대상 객체 모두를 자체적으로 인스턴스화할 수 있지만, 이는 `Message Queue` 관리자가 응용 프로그램에서 필요로 하는 브로커 자원을 제어하고 그 성능을 조정하게 하려는 관리 대상 객체의 기본 목표에 어긋납니다. 또한 관리 대상 객체를 직접 인스턴스화하면 클라이언트 응용 프로그램은 공급자 독립성을 갖기 보다는 공급자별 특성을 갖게 됩니다.

연결 팩토리 관리 대상 객체

`ConnectionFactory` 객체는 클라이언트 응용 프로그램과 `Message Queue` 메시지 서버 사이의 물리적 연결을 설정하는 데 사용됩니다. 또한 연결 및 이 연결을 사용하여 브로커에 액세스하는 클라이언트 런타임의 동작을 지정하는 데에도 사용됩니다.

분산 트랜잭션을 지원하려면(47페이지의 "로컬 트랜잭션" 참조) 분산 트랜잭션을 지원하는 특정 `XAConnectionFactory` 객체를 사용해야 합니다.

`ConnectionFactory` 관리 대상 객체를 만들려면 195페이지의 "연결 팩토리 추가"를 참조하십시오.

`ConnectionFactory` 관리 대상 객체를 구성하여 이 객체가 생성하는 모든 연결에 공통적인 속성 값(등록 정보)을 지정할 수 있습니다. `ConnectionFactory` 및 `XAConnectionFactory` 객체는 동일한 속성 집합을 갖습니다. 이 속성들은 영향을 미치는 동작에 따라 여러 범주로 그룹화됩니다.

- 연결 사양
- 자동 재연결 동작
- 클라이언트 식별
- 메시지 헤더 무시
- 안정성 및 흐름 제어
- 대기열 브라우저 동작
- Application Server 지원
- JMS 정의 등록 정보 지원

각 범주와 관련 속성은 *Message Queue Java Client Developer's Guide*에서 자세히 다룹니다. Message Queue 관리자로서 이 속성 값을 조정해야 하는 경우가 있지만, 일반적으로 클라이언트 응용 프로그램의 성능 조정을 위해 어떤 속성을 조정할 것인가는 응용 프로그램 개발자가 결정합니다. 187페이지의 표 7-3은 이 속성을 알파벳 순서로 요약합니다.

대상 관리 대상 객체

Destination 관리 대상 객체는 브로커상에서 공개적으로 명명된 Destination 객체에 해당되는 물리적 대상(대기열이나 주제)을 의미합니다. 그 2가지 속성을 표 2-11에서 확인할 수 있습니다. Destination 객체를 만들어 클라이언트 응용 프로그램의 MessageConsumer 및/또는 MessageProducer 객체가 그 물리적 대상에 액세스하게 할 수 있습니다.

Destination 관리 대상 객체를 만들려면 196페이지의 "주제 또는 대기열 추가"를 참조하십시오.

표 2-11 대상 속성

속성/등록 정보 이름	설명
imqDestinationName	물리적 대상의 공급자별 이름을 지정합니다. 물리적 대상을 만들 때 이 이름을 지정합니다. 대상 이름은 영숫자(공백 없음)만 포함하고 영문자나 "-" 및 "\$" 문자로 시작할 수 있습니다. "mq" 문자열로는 시작할 수 없습니다. 기본값: Untitled_Destination_Object
imqDestinationDescription	객체 관리에 유용한 정보를 지정합니다. 기본값: 대상 객체 설명

클라이언트 시작 시 속성 값 무시

모든 Java 응용 프로그램처럼 명령줄에서 시스템 등록 정보를 지정하여 메시징 응용 프로그램을 시작할 수 있습니다. 또한 이 메커니즘은 클라이언트 응용 프로그램 코드에서 사용하는 관리 대상 객체의 속성 값을 무시할 때 사용할 수 있습니다. 예를 들어 클라이언트 응용 프로그램 코드 중 JNDI 조회를 통해 액세스하는 관리 대상 객체 구성을 무시할 수 있습니다.

클라이언트 응용 프로그램 시작 시 관리 대상 객체 설정을 무시하려면 다음 명령줄 구문을 사용합니다.

```
java [-Dattribute=value ]... clientAppName
```

여기서 *attribute*는 187페이지의 "연결 팩토리 관리 대상 객체 속성"에서 소개하는 `ConnectionFactory` 관리 객체 대상 속성 중 하나를 의미합니다.

예를 들어 클라이언트 코드에서 액세스하는 `ConnectionFactory` 관리 대상 객체에 지정된 브로커가 아닌 다른 브로커에 클라이언트 응용 프로그램을 연결하려면 명령줄 무시를 통해 다른 브로커의 `imqBrokerHostName` 및 `imqBrokerHostPort`를 설정하여 클라이언트 응용 프로그램을 시작할 수 있습니다.

그러나 관리 대상 객체가 읽기 전용으로 설정된 경우 그 속성 값은 명령줄 무시로 변경할 수 없습니다. 그러한 무시는 적용되지 않습니다.

Message Queue 관리 작업 및 도구

Sun Java™ System Message Queue 관리는 다양한 작업과 해당 작업을 수행하기 위한 다양한 도구로 구성됩니다.

이 장에서는 먼저 관리 작업을 요약 소개하고 명령줄 관리 유틸리티의 공통 기능에 중점을 두어 관리 도구를 설명합니다.

Message Queue 관리 작업

수행해야 할 특정 작업은 개발 환경에 있는지 또는 작업 환경에 있는지에 따라 달라집니다.

개발 환경

개발 환경에서의 작업은 Message Queue 클라이언트 응용 프로그램을 프로그래밍하는데 중점을 둡니다. Message Queue 메시지 서버는 주로 테스트 용도로 필요합니다. 개발 환경에서는 유연성이 강조되며 일반적으로 다음과 같은 방법을 적용합니다.

- 관리는 주로 개발자가 테스트에 사용할 브로커를 시작하는 정도로 최소화합니다.
- 데이터 저장소(기본 제공 파일 기반 지속성), 사용자 저장소(파일 기반 사용자 저장소), 액세스 제어 등록 정보 파일 및 객체 저장소(파일 시스템 저장소)로 구성된 기본 구현을 사용합니다. 이 기본 구현은 일반적으로 개발 테스트에 적합합니다.

- 멀티 브로커 테스트를 수행 중이라면 마스터 브로커는 필요하지 않을 것입니다.
- 일반적으로 대상은 명시적으로 만들지 않고 자동 작성된 대상을 사용합니다.
- 중앙에서 관리하는 객체를 사용하는 대신 클라이언트 코드에서 관리 대상 객체를 인스턴스화할 수 있습니다.

작업 환경

응용 프로그램을 안정적으로 배포하고 실행해야 하는 작업 환경에서는 관리가 훨씬 더 중요합니다. 수행해야 할 관리 작업은 메시징 시스템 및 이 시스템이 지원해야 하는 응용 프로그램의 복잡성에 따라 달라집니다. 그러나 일반적으로 설정 작업과 유지 보수 작업으로 구분할 수 있습니다.

설정 작업

▶ 작업 환경을 설정하는 방법

보통 다음 설정 작업 중 전부는 아니더라도 최소한 일부는 수행해야 합니다.

- **관리자 보안**(관리 도구 사용 보호)
 - admin 연결 서비스가 활성 상태인지 확인합니다(57페이지의 표 2-3 참조).
 - 권한 부여: 특정 개인 또는 admin 그룹에 대해 admin 연결 서비스에 액세스할 수 있게 합니다(216페이지의 "연결 액세스 제어" 참조).
 - 그룹에 대해 권한 부여할 경우 관리자가 admin 그룹에 속해 있는지 확인합니다.
 - 파일 기반 사용자 저장소: 기본 admin 그룹을 가집니다. 관리자가 admin 그룹에 속해 있는지 확인합니다. 또는 기본 admin 사용자를 사용하는 경우 admin 비밀번호를 변경합니다(208페이지의 "기본 관리자 비밀번호 변경" 참조).
 - LDAP 사용자 저장소: 관리자가 admin 그룹에 속하는지 확인합니다.

- **일반 보안(8장, "보안 관리" 참조)**
 - 권한 부여: 파일 기반 사용자 저장소에 항목을 만들거나 브로커가 기존 LDAP 사용자 저장소를 사용하도록 구성합니다.
(최소한 관리 기능은 비밀번호로 보호합니다.)
 - 권한 부여: 액세스 제어 등록 정보 파일에서 액세스 설정을 수정합니다.
 - 암호화: SSL 기반 연결 서비스를 설정합니다.
- **관리 대상 객체(7장, "관리 대상 객체 관리" 참조)**
 - LDAP 객체 저장소를 구성하거나 설정합니다.
 - 연결 팩토리 및 대상 관리 대상 객체를 만듭니다.
- **브로커 클러스터(140페이지의 "클러스터를 이용한 작업(엔터프라이즈판)" 참조)**
 - 중앙 구성 파일을 작성합니다.
 - 마스터 브로커를 사용합니다.
- **지속성:** 브로커가 기본 제공 지속성보다는 플러그인 지속성을 사용하도록 구성합니다 (부록 B, "플러그인 지속성 설정" 참조).
- **메모리 관리:** 메시지에 할당된 메시지 수와 메모리 양이 사용 가능한 브로커 메모리 자원에 적합하도록 대상 속성을 설정합니다(171페이지의 표 6-10 참조).

유지 보수 작업

▶ 작업 환경을 설정하는 방법

또한 작업 환경에서 Message Queue 메시지 서버 자원은 엄격하게 모니터링하고 제어해야 합니다. 응용 프로그램 성능, 신뢰성 및 보안이 매우 중요하며, Message Queue 관리 도구를 사용하여 다음과 같은 다양한 작업을 지속적으로 수행해야 합니다.

- **응용 프로그램 관리**
 - 브로커의 자동 작성 기능을 사용 불가능하게 합니다(79페이지의 표 2-10 참조).
 - 응용 프로그램을 대신하여 물리적 대상을 작성합니다(170페이지의 "대상 만들기" 참조).
 - 대상에 대한 사용자 액세스를 설정합니다(212페이지의 "사용자 권한 부여: 액세스 제어 등록 정보 파일" 참조).

- 대상을 모니터링하고 관리합니다(168페이지의 "대상 관리" 참조).
- 영구 가입을 모니터링하고 관리합니다(179페이지의 "영구 가입 관리" 참조).
- 트랜잭션을 모니터링하고 관리합니다(180페이지의 "트랜잭션 관리" 참조).
- **브로커 관리 및 조정**
 - 브로커 메트릭을 사용하여 브로커를 조정 및 재구성합니다(227페이지의 9장, "메시지 서비스 분석 및 조정" 참조).
 - 브로커 메모리 자원을 관리합니다(227페이지의 9장, "메시지 서비스 분석 및 조정" 참조).
 - 클러스터에 브로커를 추가하여 로드 균형을 조정합니다(140페이지의 "클러스터를 이용한 작업(엔터프라이즈판)" 참조).
 - 오류가 발생한 브로커를 복구합니다(134페이지의 "브로커 시작" 참조).
- **관리 대상 객체 관리**
 - 필요에 따라 연결 팩토리 및 대상 관리 대상 객체를 추가로 작성합니다(195페이지의 "관리 대상 객체 추가 및 삭제" 참조).
 - 연결 팩토리 속성 값을 조정하여 성능 및 처리량을 향상시킵니다(187페이지의 "연결 팩토리 관리 대상 객체 속성" 및 200페이지의 "관리 대상 객체 업데이트" 참조).

Message Queue 관리 도구

Message Queue 관리 도구는 명령줄 유틸리티와 그래픽 사용자 인터페이스(GUI) 관리 콘솔(imqadmin)의 두 가지 범주로 구분됩니다. 콘솔은 명령 유틸리티(imqcmd)와 객체 관리자 유틸리티(imqobjmgr)의 두 가지 명령줄 유틸리티 기능을 결합합니다. 콘솔(및 이 두 가지 명령줄 유틸리티)을 사용하여 브로커를 원격 관리하고 Message Queue 관리 대상 객체를 관리할 수 있습니다. 나머지 명령줄 유틸리티(imqbrokerd, imqusermgr, imqdbmgr 및 imqkeytool)는 98페이지의 [그림 3-1](#)에서처럼 연결된 브로커와 동일한 호스트에서 실행되어야 합니다.

관리 콘솔에 대한 정보는 온라인 도움말을 참조하십시오. 일반적으로 특별한 작업을 수행하는 데 사용하는 명령줄 유틸리티는 "[명령줄 유틸리티 요약](#)"에서 설명합니다.

관리 콘솔

관리 콘솔을 사용하여 다음 작업을 수행할 수 있습니다.

- 브로커에 연결하여 관리
- 브로커에 물리적 대상 만들기 및 관리
- 객체 저장소에 연결
- 객체 저장소에 관리 대상 객체 추가 및 관리

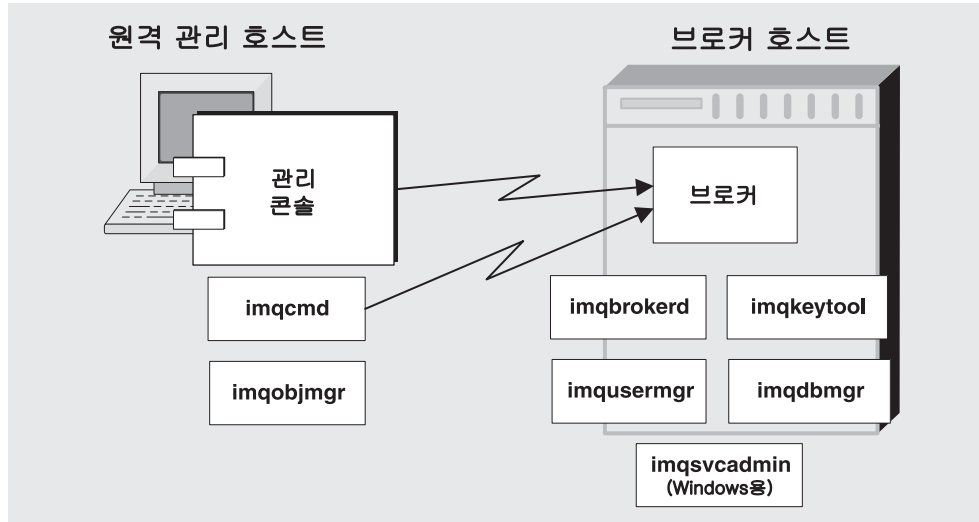
관리 콘솔로 수행할 수 없는 작업이 있으며, 그 대표적인 예로 브로커 시작, 브로커 클러스터 작성, 보다 특별한 브로커 및 물리적 대상 등록 정보 구성, 사용자 데이터베이스 관리 등이 있습니다.

4장, "[관리 콘솔 자습서](#)"에서 제공하는 간단한 실습 자습서를 통해 콘솔 기능을 익히고 이 기능을 사용하여 기본적인 작업을 수행하는 방법을 학습할 수 있습니다.

명령줄 유틸리티 요약

이 절에서는 Message Queue 관리 작업 수행 시 사용하는 명령줄 유틸리티를 소개합니다. Message Queue 유틸리티는 브로커를 시작 및 관리하고 보다 특별한 다른 관리 작업을 수행할 때 사용합니다.

그림 3-1 로컬 및 원격 관리 유틸리티



모든 Message Queue 유틸리티는 명령줄 인터페이스(CLI)에서 액세스할 수 있습니다. 이 장의 뒤에 나오는 절에서 설명하겠지만 유틸리티 명령은 공통의 형식, 구문 규칙 및 옵션을 공유합니다. 명령줄 유틸리티 사용에 대한 자세한 정보는 뒤에 나오는 장을 참조하십시오.

브로커(imqbrokerd). 브로커 유틸리티는 브로커를 시작할 때 사용합니다. imqbrokerd 명령에 옵션을 사용하여 브로커와 클러스터의 연결 여부 및 추가 구성 정보를 지정합니다. imqbrokerd 명령은 [5장, "브로커 시작 및 구성"](#)에 설명되어 있습니다.

명령(imqcmd). 브로커를 시작한 후 명령 유틸리티를 사용하여 물리적 대상을 작성, 업데이트 및 삭제하고, 브로커와 해당 연결 서비스를 제어하고, 브로커의 자원을 관리합니다. imqcmd 명령에 대한 설명은 [6장, "브로커 및 응용 프로그램 관리"](#)를 참조하십시오.

객체 관리자(imqobjmgr). 객체 관리자 유틸리티는 JNDI를 통해 액세스 가능한 객체 저장소에서 관리 객체를 추가, 나열, 업데이트 및 삭제할 때 사용합니다. 관리 객체는 JMS 클라이언트를 공급자별 이름 지정 및 구성 형식과 분리하여 공급자 독립성을 부여할 수 있습니다. imqobjmgr 명령은 [7장, "관리 대상 객체 관리"](#)에 설명되어 있습니다.

사용자 관리자(imqusermgr). 사용자 관리자 유틸리티는 사용자 인증 및 권한 부여에 사용하는 파일 기반 사용자 저장소를 채울 때 사용합니다. imqusermgr 명령은 8장, "보안 관리"에 설명되어 있습니다.

키 도구(imqkeytool). 키 도구 유틸리티는 SSL 인증에 사용하는 자체 서명된 인증서를 생성할 때 사용합니다. imqkeytool 명령은 8장, "보안 관리" 및 부록 C, "HTTP/HTTPS 지원(엔터프라이즈판)"에 설명되어 있습니다.

데이터베이스 관리자(imqdbmgr). 데이터베이스 관리자 유틸리티는 영구 저장소에 사용하는 JDBC 호환 데이터베이스를 작성하고 관리할 때 사용합니다. imqdbmgr 명령은 부록 B, "플러그인 지속성 설정"에 설명되어 있습니다.

서비스 관리자(imqsvcadm). 서비스 관리자 유틸리티는 브로커를 Windows 서비스로 설치, 쿼리 및 제거할 때 사용합니다. imqsvcadm 명령은 부록 D, "브로커를 Windows 서비스로 사용"에 설명되어 있습니다.

명령줄 구문

Message Queue 명령줄 인터페이스 유틸리티는 간단한 셸 명령입니다. 즉 Windows, Linux 또는 Solaris의 명령 입력 셸에서는 유틸리티 이름 자체가 명령이며 하위 명령이나 옵션은 해당 명령에 전달된 인수에 불과합니다. 따라서 이 유틸리티를 시작하거나 종료하는 명령은 없으며, 필요하지도 않습니다.

모든 명령줄 유틸리티는 다음과 같은 명령 구문을 공유합니다.

```
Utility_Name [subcommand] [argument] [[-option_name [-option_argument]]...]
```

*Utility_Name*은 Message Queue 유틸리티의 이름(예: imqcmd, imqobjmgr, imqusermgr 등)을 지정합니다.

기억해 둘 네 가지 중요한 사항이 있습니다.

- 하위 명령(유틸리티가 두 가지 피연산자 유형을 모두 받아들이는 경우 인수도 포함) 다음에 옵션을 지정합니다.
- 인수가 공백을 포함하는 경우 전체 인수를 따옴표로 묶습니다. 일반적으로 속성-값 쌍을 따옴표로 묶는 것이 가장 안전합니다.

- 명령줄에 `-v` (버전)나 `-h/-H` (도움말) 옵션을 지정하면 해당 명령줄의 다른 부분은 실행되지 않습니다. 공통 옵션에 대한 설명은 [100페이지의 표 3-1](#)을 참조하십시오.
- 하위 명령, 인수, 옵션 및 옵션 인수는 공백으로 구분합니다.

다음은 하위 명령줄이 없는 명령줄의 예입니다. 이 명령은 기본 브로커를 시작합니다.

```
imqbrokerd
```

다음 명령은 좀더 복잡합니다. `admin` 관리자 아이디와 `admin` 비밀번호를 사용하여 확인 절차 및 콘솔 출력 없이 `myQueue`라는 이름의 대기열 유형 대상을 제거합니다.

```
imqcmd destroy dst -t q -n myQueue -u admin -p admin -f -s
```

공통 명령줄 옵션

[표 3-1](#)은 모든 Message Queue 관리 유틸리티에서 공통적으로 사용하는 옵션을 설명합니다. 명령줄에서 하위 명령을 지정한 *다음에*이 옵션을 지정해야 한다는 요구 사항을 제외하면 다음 옵션(및 유틸리티에 전달되는 기타 모든 옵션)은 특정 순서대로 입력할 필요는 없습니다.

표 3-1 공통 Message Queue 명령줄 옵션

옵션	설명
<code>-h</code>	지정한 유틸리티의 사용 도움말을 표시합니다.
<code>-H</code>	속성 목록 및 예를 포함하여 확장 사용 도움말을 표시합니다(<code>imqcmd</code> 및 <code>imqobjmgr</code> 만 지원).
<code>-s</code>	자동 모드를 시작합니다. 표시되는 출력이 없습니다. <code>imqbrokerd</code> 의 경우 <code>-silent</code> 로 지정합니다.
<code>-v</code>	버전 정보를 표시합니다.
<code>-f</code>	사용자 확인 프롬프트 없이 해당 작업을 수행합니다.
<code>-pre</code>	(<code>imqobjmgr</code> 에서만 사용) 미리 보기 모드를 시작하며, 사용자는 실제로 명령을 수행하지 않은 상태에서 명령줄 나머지 부분의 실행 결과를 확인할 수 있습니다. 기본 속성 값을 확인할 때 유용합니다.
<code>-javahome path</code>	Java 2와 호환할 수 있는 대체 런타임을 지정하여 사용합니다(기본값은 시스템의 런타임 또는 Message Queue와 함께 제공되는 런타임 사용).

관리 콘솔 자습서

이 자습서는 Message Queue 메시지 서버 관리용 그래픽 인터페이스인 관리 콘솔의 사용에 대해 중점적으로 다룹니다. 이 자습서에서 다음 작업 방법을 학습할 수 있습니다.

- 브로커를 시작하고 콘솔을 사용하여 연결 및 관리
- 브로커에 물리적 대상 만들기
- 객체 저장소를 작성하고 콘솔을 사용하여 연결
- 객체 저장소에 관리 대상 객체 추가

이 자습서에서는 간단한 JMS 호환 응용 프로그램인 HelloWorldMessageJNDI를 실행하는데 필요한 대상 및 관리 대상 객체를 설정합니다. 이 응용 프로그램은 예제 응용 프로그램 /demo 디렉토리의 helloworld 하위 디렉토리에 있습니다(부록 A, “Message Queue 데이터의 위치” 참조). 자습서 마지막 부분에서 이 응용 프로그램을 실행합니다.

이 자습서는 관리 콘솔을 사용하여 기본 관리 작업을 수행하는 방법을 안내하기 위해 작성된 것입니다. 이 자습서를 참조하더라도 *Message Queue Java Client Developer’s Guide*나 이 *관리 설명서*의 나머지 장을 읽어보아야 합니다.

일부 Message Queue 관리 작업은 그래픽 도구를 사용하여 수행할 수 없으며, 다음과 같은 작업은 명령줄 유틸리티를 사용해야 합니다.

- 특정 물리적 대상 등록 정보 구성

일부 물리적 대상 등록 정보는 관리 콘솔을 사용하여 구성할 수 없습니다. 이러한 등록 정보는 168페이지의 “대상 관리” 절에서 설명한 대로 구성할 수 있습니다.

- 브로커 클러스터 만들기
자세한 내용은 [140페이지](#)의 "[클러스터를 이용한 작업\(엔터프라이즈판\)](#)"을 참조하십시오.
- 사용자 데이터베이스 관리
자세한 내용은 [202페이지](#)의 "[사용자 인증](#)"을 참조하십시오.

준비

이 자습서를 시작하기 전에 **Message Queue** 제품을 설치해야 합니다. 자세한 내용은 *Message Queue 설치 설명서*를 참조하십시오. 이 자습서는 **Windows**를 중심으로 구성되어 있고 **Unix®** 사용자를 위한 참고 내용이 추가되어 있습니다.

이 자습서에서 항목1 > 항목2 > 항목3을 선택하는 것은 항목1이라는 폴다운 메뉴에서 항목2를 선택하고 항목2가 제시하는 선택 사항 중 항목3을 선택한다는 의미입니다.

관리 콘솔 시작

관리 콘솔은 다음 작업을 수행할 때 사용하는 그래픽 도구입니다.

- 브로커에 대해 참조를 생성하고 브로커에 연결
- 브로커 관리
- 브로커가 메시지를 전달할 때 사용할 물리적 대상을 브로커에 만들기
- **Message Queue** 관리 대상 객체를 저장하는 객체 저장소에 연결

관리 대상 객체를 사용하여 **JMS** 호환 응용 프로그램의 메시징 요구 사항을 관리할 수 있습니다. 자세한 내용은 [89페이지](#)의 "[Message Queue 관리 대상 객체](#)"를 참조하십시오.

▶ **관리 콘솔을 시작하는 방법**

1. 시작 > 프로그램 > Sun Java System Message Queue 3.5 SP1 > 관리를 선택합니다.

콘솔 창이 표시되려면 몇 초 정도 기다려야 합니다.

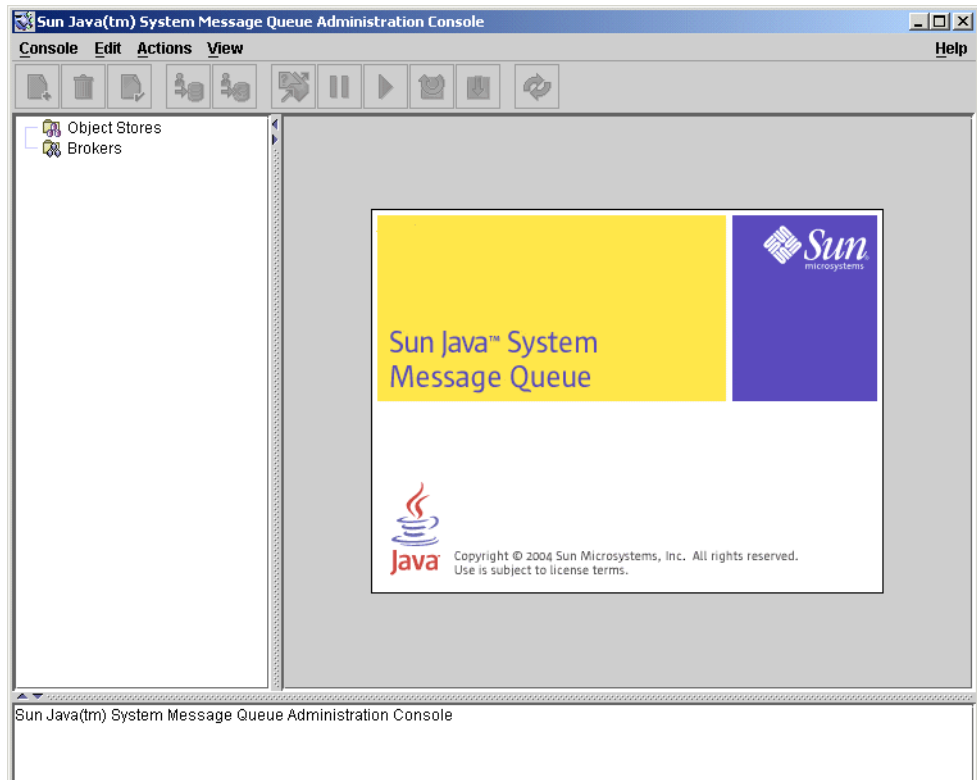
비 Windows 사용자: 명령 프롬프트에 다음 명령을 입력합니다.

```
/usr/bin/imqadmin (Solaris)
```

```
/opt/imq/bin/imqadmin (Linux)
```

2. 몇 초간 콘솔 창을 검사합니다.

콘솔에서 맨 위에 메뉴 모음, 메뉴 모음 바로 아래에 도구 모음, 왼쪽에는 탐색 창, 오른쪽에는 결과 창(현재 Sun Java System Message Queue 제품을 나타내는 그래픽 표시) 그리고 맨 아래에 상태 창이 표시됩니다.



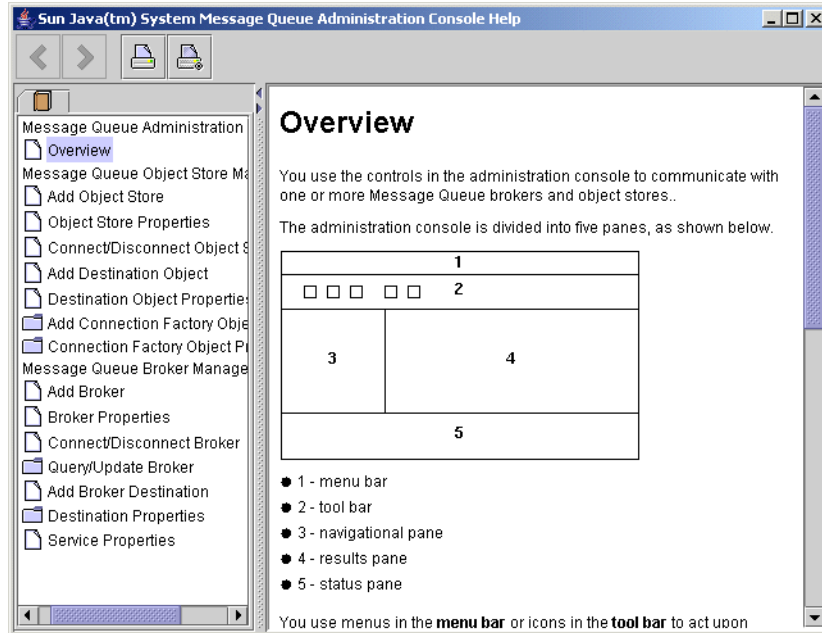
자습서만으로는 완전한 정보를 제공할 수는 없으므로, 우선 관리 콘솔에 대한 도움말 정보를 얻는 방법을 알아 봅니다.

도움말 확인

메뉴 표시줄 맨 오른쪽에서 도움말 메뉴를 찾습니다.

▶ **관리 콘솔 도움말 정보를 표시하는 방법**

1. 도움말 풀다운 메뉴에서 개요를 선택합니다. 도움말 창이 표시됩니다.



도움말 정보의 구성 방식에 유의하십시오. 왼쪽 탐색 창은 목차를 표시하고 오른쪽 결과 창은 탐색 창에서 선택한 항목의 내용을 표시합니다.

도움말 창의 결과 창을 확인합니다. 이 창에는 관리 콘솔의 구조 보기를 표시하고 각 콘솔 창의 사용을 보여 줍니다.

2. 도움말 창의 탐색 창을 확인합니다. 이 창은 개요, 객체 저장소 관리 및 브로커 관리의 3가지 영역의 주제로 구성되어 있습니다. 각 영역은 파일과 폴더로 구성됩니다. 각 폴더는 여러 탭으로 구성된 대화 상자에 대한 도움말을 제공하고, 각 파일은 단순한 대화 상자나 탭에 대한 도움말을 제공합니다.

첫 번째 콘솔 관리 작업인 106페이지의 "브로커 추가"에서는 콘솔을 통해 관리하는 브로커에 대한 참조를 작성합니다. 그러나 시작하기 전에 온라인 도움말에서 정보를 확인하십시오.

3. 도움말 창의 탐색 창에서 브로커 추가 항목을 누릅니다.
결과 창이 변경됩니다. 이제 브로커 추가의 의미 및 브로커 추가 대화 상자의 각 필드 사용법을 설명하는 텍스트가 표시됩니다. 필드 이름은 굵은 체로 표시됩니다.
4. 도움말 텍스트를 읽어 봅니다.
5. 도움말 창을 닫습니다.

브로커 작업

브로커는 **Message Queue** 메시징 시스템을 위해 전달 서비스를 제공합니다. 메시지 전달은 2단계 과정입니다. 먼저 메시지는 브로커에 있는 물리적 대상에게 전달되며, 그런 다음 하나 이상의 사용자 클라이언트에게 전달됩니다.

브로커 작업과 관련된 작업은 다음과 같습니다.

- 브로커 시작 및 구성

Windows에서 시작 > 프로그램 메뉴, 또는 `imqbrokerd` 명령을 사용하여 브로커를 시작할 수 있습니다. `imqbrokerd` 명령을 사용할 경우, 명령줄 옵션을 사용하여 브로커 구성 정보를 지정할 수 있습니다. 프로그램 메뉴를 사용할 경우, 콘솔이나 [5장](#), "[브로커 시작 및 구성](#)"에서 설명하는 다른 방법으로 구성 정보를 지정할 수 있습니다.

주 관리 콘솔을 사용하면 브로커 인스턴스를 시작할 수 없습니다.

- 관리 콘솔을 사용하거나 **Command** 명령줄 유틸리티(`imqcmd`)를 사용하여 브로커 및 그 서비스를 관리합니다.
- 클라이언트 응용 프로그램이 필요로 하는 물리적 대상을 작성합니다.
- 자원 사용을 모니터링하여 처리량 및 신뢰성을 향상시킵니다.

브로커는 응용 프로그램 클라이언트 및 관리 클라이언트와의 통신을 지원합니다. 다양한 연결 서비스를 통해 이를 지원하며, 브로커가 이 서비스 중 일부 또는 전부를 실행하도록 구성할 수 있습니다. 연결 서비스에 대한 자세한 정보는 [54페이지의 "연결 서비스"](#)를 참조하십시오.

브로커 시작

관리 콘솔을 사용하여 브로커를 시작할 수 없습니다. 다음 절차에 설명한 대로 브로커를 시작합니다(5장, "브로커 시작 및 구성" 참조).

▶ 브로커를 시작하는 방법

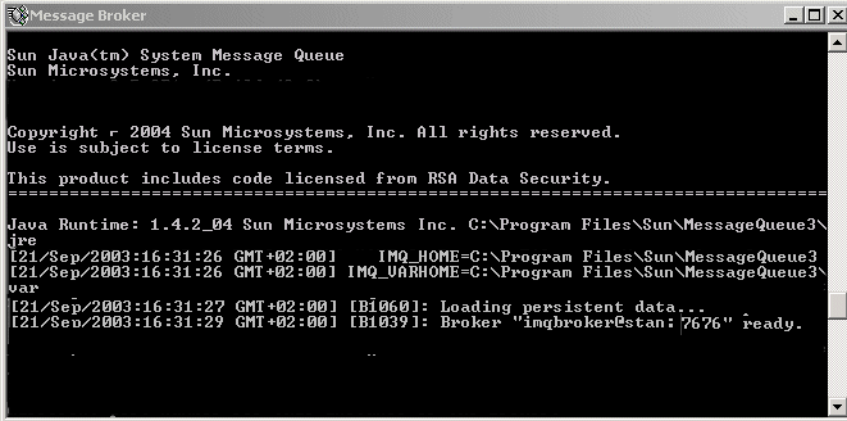
1. 시작 > 프로그램 > Sun Java System Message Queue 3.5 SP1 > 메시지 브로커를 선택합니다.

비 Windows: 다음 명령을 입력하여 브로커를 시작합니다.

`/usr/bin/imqbrokerd (Solaris)`

`/opt/imq/bin/imqbrokerd (Linux)`

명령 프롬프트 창이 표시되고 브로커가 준비 중인 상태로 표시됩니다.



```

Message Broker
Sun Java(tm) System Message Queue
Sun Microsystems, Inc.

Copyright © 2004 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.

This product includes code licensed from RSA Data Security.
=====
Java Runtime: 1.4.2_04 Sun Microsystems Inc. C:\Program Files\Sun\MessageQueue3\
jre
[21/Sep/2003:16:31:26 GMT+02:00] IMQ_HOME=C:\Program Files\Sun\MessageQueue3\
[21/Sep/2003:16:31:26 GMT+02:00] IMQ_VARHOME=C:\Program Files\Sun\MessageQueue3\
var
[21/Sep/2003:16:31:27 GMT+02:00] [B1060]: Loading persistent data...
[21/Sep/2003:16:31:29 GMT+02:00] [B1039]: Broker "imqbroker@stan:7676" ready.
    
```

2. 다시 관리 콘솔 창으로 돌아옵니다. 이제 콘솔에 브로커를 추가하여 연결할 준비가 되었습니다.

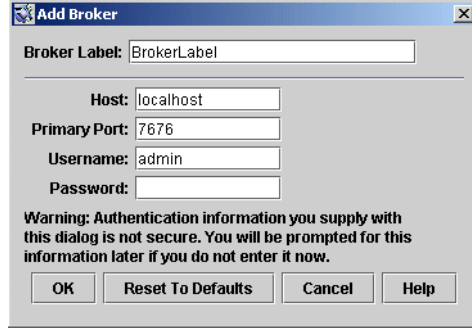
관리 콘솔에서 브로커에 대한 참조를 추가하기 전에 브로커를 시작할 필요는 없지만, 브로커에 연결하려면 먼저 브로커를 시작해야 합니다.

브로커 추가

브로커를 추가하면 관리 콘솔에서 해당 브로커에 대한 참조가 생성됩니다. 브로커를 추가한 후 연결할 수 있습니다.

▶ 관리 콘솔에 브로커를 추가하는 방법

1. 탐색 창에서 브로커를 마우스 오른쪽 버튼으로 누른 다음 브로커 추가를 선택합니다.
2. 브로커 레이블 필드에 MyBroker라고 입력합니다.
관리 콘솔에서 이 브로커를 식별하는 레이블을 제공합니다.

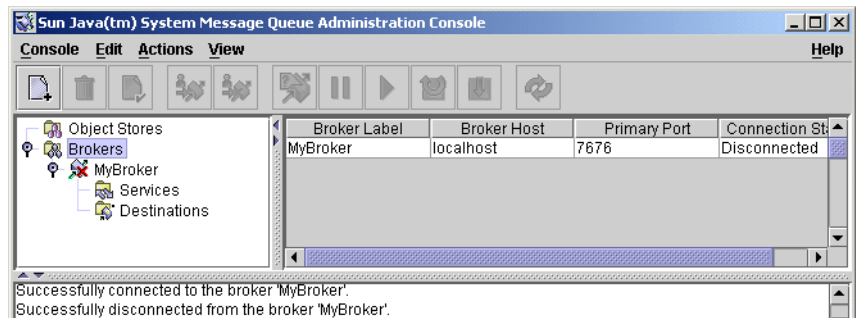


대화 상자에 지정된 기본 호스트 이름(localhost) 및 기본 포트(7676)에 유의합니다. 나중에 클라이언트가 이 브로커에 대해 연결을 설정할 때 사용할 연결 팩토리를 구성할 때 지정해야 하는 값입니다.

비밀번호 필드는 비워 둡니다. 연결할 때 비밀번호를 지정하면 더 안전합니다.

3. 확인을 눌러 브로커를 추가합니다.

탐색 창을 확인합니다. 방금 추가한 브로커가 브로커 아래에 표시됩니다. 브로커 아이콘 위의 빨간색 X 표시는 이 브로커가 현재 콘솔에 연결되어 있지 않다는 의미입니다.



4. MyBroker를 마우스 오른쪽 버튼으로 누르고 팝업 메뉴에서 등록 정보를 선택합니다.
브로커 등록 정보 대화 상자가 표시됩니다. 이 대화 상자에서는 브로커를 추가할 때 지정했던 등록 정보를 업데이트할 수 있습니다.
5. 취소를 눌러 대화 상자를 닫습니다.

관리 비밀번호 변경

브로커를 추가할 때 비밀번호를 지정하지 않았다면 브로커에 연결할 때 비밀번호를 입력 하라는 프롬프트가 표시됩니다. 기본적으로 관리 콘솔은 admin 아이디와 admin 비밀번호를 사용하여 브로커에 연결할 수 있습니다. 보다 안전하게 하기 위해서 연결 전에 기본 관리자 비밀번호(admin)를 변경하는 것이 좋습니다.

▶ 관리자 비밀번호를 변경하는 방법

1. 명령 프롬프트 창을 열거나, 이미 열려 있는 경우에는 앞으로 가져옵니다.
2. 다음과 같이 명령을 입력합니다. abracadabra 대신 자신의 비밀번호를 입력합니다. 그러면 지정한 비밀번호가 admin 기본 비밀번호를 대체합니다.

```
imqusermgr update -u admin -p abracadabra
```

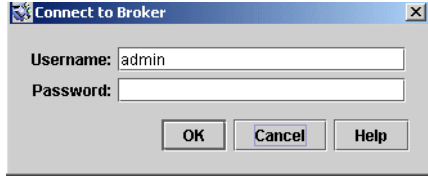
변경 사항은 즉시 적용됩니다. Message Queue 명령줄 유틸리티 중 하나 또는 관리 콘솔을 사용할 때마다 새 비밀번호를 지정해야 합니다.

클라이언트는 관리자와 다른 연결 서비스를 사용하지만, 기본 아이디와 비밀번호가 지정되므로 광범위한 관리 설정 작업 없이 Message Queue를 테스트할 수 있습니다. 기본적으로 클라이언트는 guest 아이디와 guest 비밀번호를 사용하여 브로커에 연결할 수 있습니다. 그러나 가급적 일찍 클라이언트를 위한 안전한 아이디와 비밀번호를 설정해야 합니다. 자세한 정보는 [202페이지의 "사용자 인증"](#)을 참조하십시오.

브로커에 연결

▶ 브로커에 연결하는 방법

1. MyBroker를 마우스 오른쪽 버튼으로 누르고 브로커에 연결을 선택합니다.
아이디와 비밀번호를 지정할 수 있는 대화 상자가 표시됩니다.



- 비밀번호 필드에 admin을 입력하거나, 108페이지의 "관리 비밀번호 변경"에서 비밀번호로 지정한 값을 입력합니다.

admin 아이디와 정확한 비밀번호를 제시하면 관리자 권한으로 브로커에 연결됩니다.

- 확인을 눌러 브로커에 연결합니다.

브로커에 연결한 후, 작업 메뉴에서 브로커에 대한 정보 얻기, 브로커 중지 및 다시 시작, 브로커 종료 및 다시 시작 그리고 브로커와의 연결 해제를 선택할 수 있습니다.

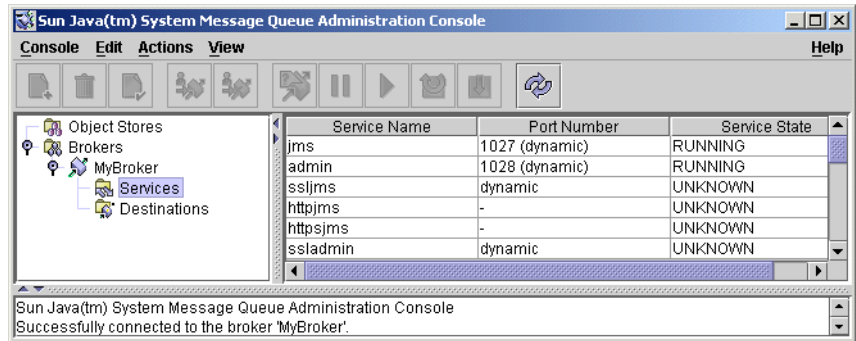
연결 서비스 보기

브로커는 해당 브로커에서 제공하는 연결 서비스와 지원하는 물리적 대상으로 구별됩니다.

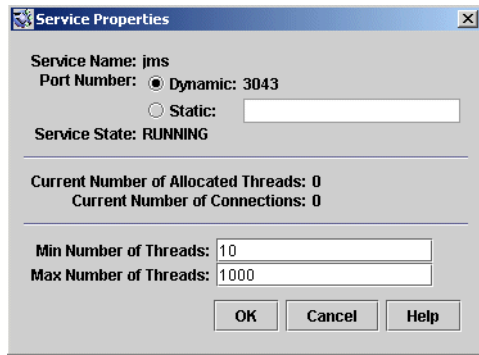
▶ 사용 가능한 연결 서비스를 보는 방법

- 탐색 창에서 서비스를 선택합니다.

사용 가능한 서비스가 결과 창에 표시됩니다. 각 서비스마다 이름, 포트 번호 및 상태가 제시됩니다.



2. 결과 창에서 jms 서비스를 눌러 선택합니다.
3. 작업 풀다운 메뉴를 누르고 강조 표시된 항목을 확인합니다.
jms 서비스를 중지하거나 그 등록 정보를 확인하고 업데이트할 수 있습니다.
4. 작업 메뉴에서 등록 정보를 선택합니다.
서비스 등록 정보 대화 상자를 사용하여 서비스에 정적 포트 번호를 할당하고 이 서비스에 할당된 최소 및 최대 스레드 수를 변경할 수 있습니다.



5. 확인이나 취소를 눌러 등록 정보 대화 상자를 닫습니다.
6. 결과 창에서 관리 서비스를 선택합니다.
7. 작업 풀다운 메뉴를 누릅니다.
이 서비스는 일시 중지할 수 없습니다(일시 중지 항목 사용 불가). 관리 서비스는 관리자가 브로커에 연결하는 링크입니다. 이 링크를 일시 중지하면 관리자는 더 이상 브로커에 액세스할 수 없습니다.
8. 작업 > 등록 정보를 선택하여 관리 서비스의 등록 정보를 확인합니다.
9. 작업을 마쳤으면 확인이나 취소를 누릅니다.

브로커에 물리적 대상 추가

기본적으로 브로커에 대해 대상 자동 작성이 사용 가능하므로 물리적 대상을 동적으로 만들 수 있습니다. 따라서, 개발 환경에서는 클라이언트 코드를 테스트하기 위해 대상을 명시적으로 만들 필요가 없습니다.

그러나 작업 설정에서는 물리적 대상을 명시적으로 만드는 것이 좋습니다. 그렇게 하면 관리자는 브로커에서 사용 중인 대상을 완전히 파악할 수 있습니다.

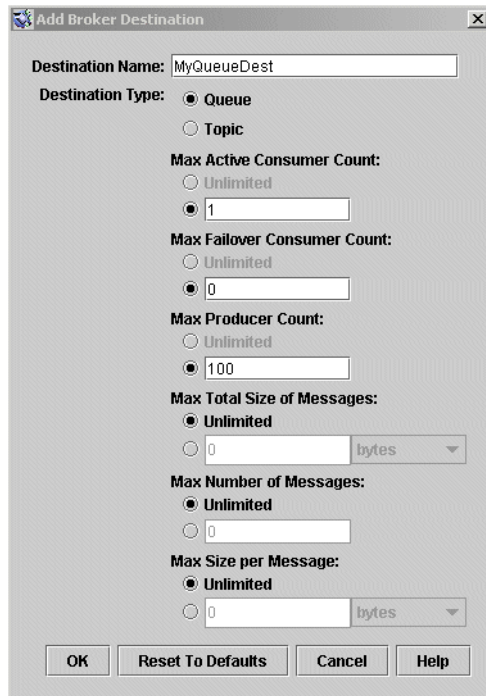
imq.autocreate.topic 또는 imq.autocreate.queue 등록 정보를 설정하여 브로커가 자동 작성된 대상을 추가할 수 있는지 여부를 제어합니다. 자세한 내용은 [78페이지의 "자동 작성\(대 관리 작성\) 대상"](#)을 참조하십시오.

이 절에서 브로커에 물리적 대상을 추가합니다. 관리자는 대상에 지정한 이름을 적어 놓아야 합니다. 나중에 이 물리적 대상에 해당되는 관리 대상 객체를 작성할 때 이 이름이 필요합니다.

➤ **브로커에 대기열 대상을 추가하는 방법**

1. MyBroker의 대상 노드를 마우스 오른쪽 버튼으로 누르고 브로커 대상 추가를 선택합니다.

다음 대화 상자가 표시됩니다.



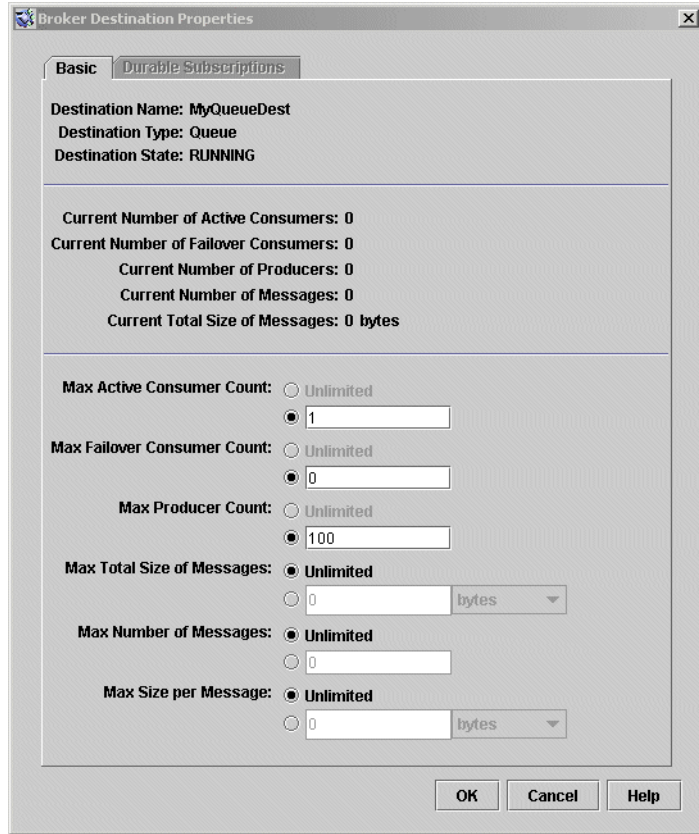
2. 대상 이름 필드에 MyQueueDest를 입력합니다.
3. 대기열 라디오 버튼이 선택되어 있지 않으면 선택합니다.
4. 확인을 눌러 물리적 대상을 추가합니다.

이제 결과 창에 대상이 표시됩니다.

물리적 대상 작업

브로커에 물리적 대상을 추가했다면 아래 절차에 설명된 대로 다음 작업을 수행할 수 있습니다.

- 물리적 대상의 등록 정보 확인 및 업데이트
 - 대상에서 메시지 제거
 - 대상 삭제
- ▶ **물리적 대상의 등록 정보를 보는 방법**
1. MyBroker의 대상 노드를 선택합니다.
 2. 결과 창에서 MyQueueDest를 선택합니다.
 3. 작업 > 등록 정보를 선택합니다.
다음 대화 상자가 표시됩니다.



대화 상자에는 대기열에 대해 현재 상태 정보와 변경 가능한 일부 등록 정보가 표시됩니다.

4. 취소를 눌러 대화 상자를 닫습니다.

▶ 대상에서 메시지를 제거하는 방법

1. 결과 창에서 물리적 대상을 선택합니다.

2. 작업 > 메시지 제거를 선택합니다.

확인 대화 상자가 표시됩니다.

메시지 제거 기능은 메시지를 삭제하고 빈 대상을 남겨 둡니다.

▶ **대상을 삭제하는 방법**

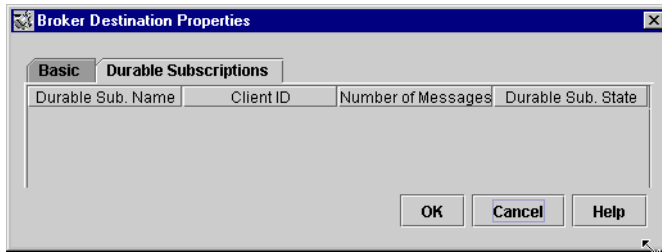
1. 결과 창에서 물리적 대상을 선택합니다.
2. 편집 > 삭제를 선택합니다.
확인 대화 상자가 표시됩니다.

주 MyQueueDest 대기열 대상은 삭제하지 마십시오.

대상 삭제 기능은 대상에서 메시지를 제거하고 대상도 제거합니다.

주제 대상에 대한 정보 얻기

브로커 주제 대상 등록 정보 대화 상자에는 영구 가입 정보를 나열하는 추가 탭이 있습니다. 대기열 대상의 경우 이 탭이 비활성화됩니다.



이 대화 상자에서는 다음 작업을 할 수 있습니다.

- 영구 가입 제거, 영구 가입과 관련된 모든 메시지 제거
- 영구 가입 제거, 영구 가입과 관련된 모든 메시지 및 해당 영구 가입 제거

객체 저장소 작업

객체 저장소는 LDAP 디렉토리 서버든 파일 시스템 저장소(파일 시스템의 디렉토리)든, 클라이언트 응용 프로그램이 사용하는 객체에 대한 Message Queue 특정 구현 및 구성 정보를 캡슐화하는 Message Queue 관리 대상 객체 저장에 사용합니다.

관리 대상 객체는 클라이언트 코드 내부에서 인스턴스화하고 구성할 수 있지만, 관리자가 이 객체를 작성 및 구성하고 이를 클라이언트 응용 프로그램이 JNDI를 통해 액세스하는 객체 저장소에 저장하는 것이 바람직합니다. 이 경우 클라이언트 코드는 공급자 독립성을 가질 수 있습니다.

관리 대상 객체에 대한 자세한 내용은 89페이지의 "[Message Queue 관리 대상 객체](#)"를 참조하십시오.

관리 콘솔을 사용하여 객체 저장소를 작성할 수 없습니다. 다음 절에 설명된 대로 미리 작성해두어야 합니다.

객체 저장소 추가

객체 저장소를 추가하면 관리 콘솔의 기존 객체 저장소에 대해 참조가 생성됩니다. 이 참조는 콘솔을 종료하고 다시 시작하는 경우에도 유지됩니다.

▶ 파일 시스템 객체 저장소를 추가하는 방법

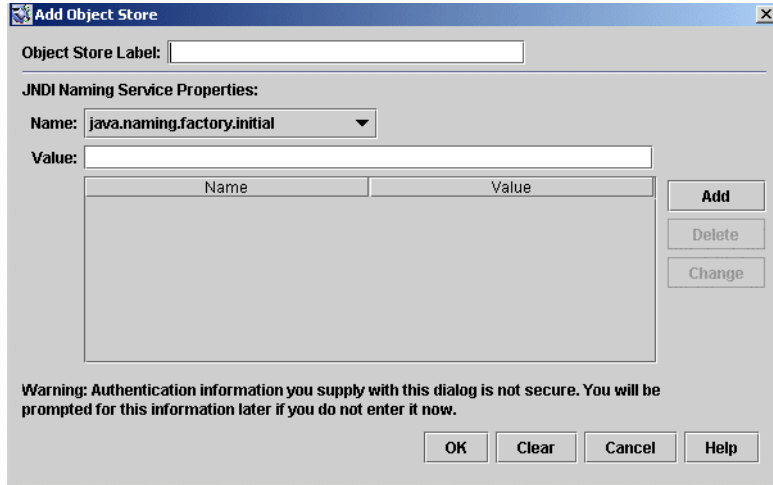
1. 아직 C 드라이브에 Temp라는 이름의 폴더가 없다면 지금 만듭니다.

이 자습서에서 사용하는 샘플 응용 프로그램에서는 객체 저장소가 C 드라이브의 Temp 폴더라고 가정합니다. 일반적으로 파일 시스템 객체 저장소는 어떤 드라이브의 어떤 디렉토리도 가능합니다.

비 Windows: 이미 존재하는 /tmp 디렉토리를 사용할 수 있습니다.

2. 객체 저장소를 마우스 오른쪽 버튼으로 눌러 객체 저장소 추가를 선택합니다.

다음 대화 상자가 표시됩니다.

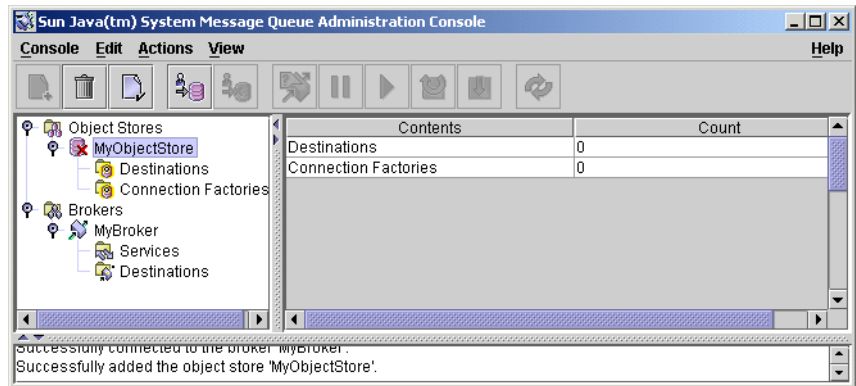


3. 객체 저장소 레이블이라는 이름의 필드에 MyObjectStore라고 입력합니다.
 이는 관리 콘솔에서 객체 저장소를 표시할 레이블을 제공합니다.
 다음 단계에서는 JNDI 이름/값 쌍을 입력해야 합니다. 이 쌍은 JMS 호환 응용 프로그램에서 관리 대상 객체를 조회할 때 사용됩니다.
4. 이름 드롭다운 목록에서 java.naming.factory.initial을 선택합니다.
 이 등록 정보에서는 어떤 JNDI 서비스 공급자를 사용할 것인지 지정할 수 있습니다. 파일 시스템 서비스 제공자나 LDAP 서비스 제공자를 예로 들 수 있습니다.
5. 값 필드에 다음을 입력합니다.

```
com.sun.jndi.fscontext.RefFSContextFactory
```

 이는 파일 시스템 저장소를 사용한다는 의미입니다(LDAP 저장소의 경우 com.sun.jndi.ldap.LdapCtxFactory를 지정).
 작업 환경에서는 LDAP 디렉토리 서버를 객체 저장소로 사용할 것입니다. 서버 설정 및 JNDI 조회 수행에 대한 정보는 [184페이지의 "LDAP 서버 객체 저장소"](#)를 참조하십시오.
6. 추가 버튼을 누릅니다.
 등록 정보 및 그 값은 이제 등록 정보 요약 창에 표시됩니다.

7. 이름 드롭다운 목록에서 `java.naming.provider.url`을 선택합니다.
이 등록 정보에서 객체 저장소의 정확한 위치를 지정할 수 있습니다. 파일 시스템 유형의 객체 저장소의 경우 이것이 기존 디렉토리의 이름이 됩니다.
8. 값 필드에 다음을 입력합니다.
`file:///C:/Temp`
(Solaris 및 Linux의 경우 `file:///tmp`)
9. 추가 버튼을 누릅니다.
두 등록 정보와 그 값이 이제 등록 정보 요약 창에 표시됩니다. LDAP 서버를 사용하는 경우, 인증 정보를 지정해야 하는 경우가 있습니다. 파일 시스템 저장소에 대해서는 지정할 필요가 없습니다.
10. 확인을 눌러 객체 저장소를 추가합니다.
11. 탐색 창에서 `MyObjectStore` 노드가 선택되지 않았다면 지금 선택합니다.
이제 관리 콘솔은 다음과 같이 표시됩니다.



객체 저장소는 탐색 창에, 그 내용, 대상 및 연결 팩토리는 결과 창에 표시됩니다. 아직 객체 저장소에 관리 대상 객체를 추가하지 않았으며, 이는 결과 창의 수 열에서 확인할 수 있습니다.

탐색 창의 객체 저장소 아이콘에 빨간색 X가 표시됩니다. 이 표시가 나타나면 연결이 끊겼다는 의미입니다. 객체 저장소를 사용하려면 먼저 연결해야 합니다.

객체 저장소 등록 정보 확인

관리 콘솔이 객체 저장소에 연결되지 않은 상태에서 객체 저장소의 일부 등록 정보를 확인하고 변경할 수 있습니다.

▶ 객체 저장소 등록 정보를 표시하는 방법

1. 탐색 창에서 MyObjectStore를 마우스 오른쪽 버튼으로 누릅니다.
2. 팝업 메뉴에서 등록 정보를 선택합니다.
객체 저장소를 추가할 때 지정한 모든 등록 정보를 보여주는 대화 상자가 표시됩니다. 이 등록 정보를 변경하고 확인을 누르면 기존 정보를 업데이트합니다.
3. 확인 또는 취소를 눌러 대화 상자를 닫습니다.

객체 저장소에 연결

객체 저장소에 객체를 추가하려면 먼저 객체 저장소에 연결해야 합니다.

▶ 객체 저장소에 연결하는 방법

1. 탐색 창에서 MyObjectStore를 마우스 오른쪽 버튼으로 누릅니다.
2. 팝업 메뉴에서 객체 저장소에 연결을 선택합니다.
객체 저장소 아이콘에 더 이상 X 표시가 없습니다. 이제 객체 저장소에 객체, 연결 팩토리 및 대상을 추가할 수 있습니다.

연결 팩토리 관리 객체 추가

관리 콘솔을 사용하여 연결 팩토리를 작성하고 구성할 수 있습니다. 연결 팩토리는 클라이언트 코드가 브로커에 연결할 때 사용됩니다. 연결 팩토리 구성을 통해 이 연결 팩토리를 사용하여 작성되는 연결의 동작을 제어할 수 있습니다.

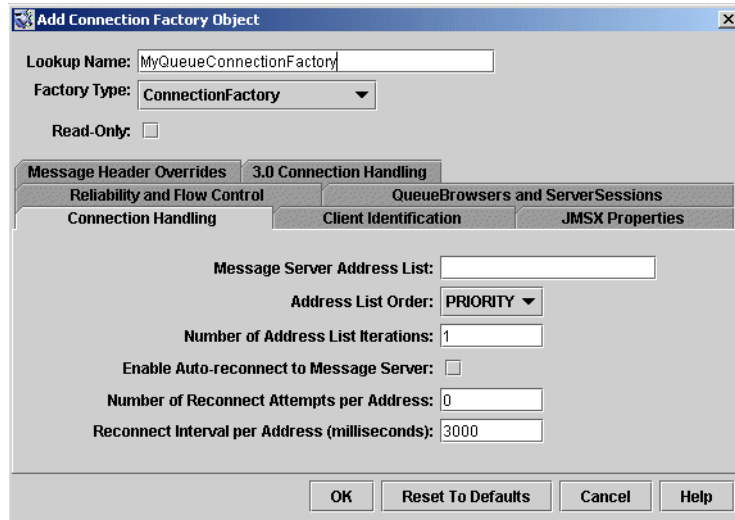
연결 팩토리 구성에 대한 정보는 온라인 도움말 및 *Message Queue Java Client Developer's Guide*를 참조하십시오.

주 관리 콘솔은 Message Queue 관리 대상 객체만 나열하고 표시합니다. 객체 저장소에 추가할 관리 대상 객체와 동일한 조회 이름을 가진 비 Message Queue 객체를 포함해야 하는 경우에 이 객체를 추가하려고 하면 오류 메시지가 표시됩니다.

▶ 객체 저장소에 연결 팩토리를 추가하는 방법

1. 아직 MyObjectStore에 연결되지 않았다면 지금 연결합니다(118페이지의 "객체 저장소에 연결" 참조).
2. 연결 팩토리 노드를 마우스 오른쪽 버튼으로 누르고 연결 팩토리 객체 추가를 선택합니다.

연결 팩토리 객체 추가 대화 상자가 표시됩니다.



3. 조회 이름 필드에 "MyQueueConnectionFactory"라는 이름을 입력합니다.
이 이름은 HelloWorldMessageJNDI.java의 다음 행에서 알 수 있듯이 클라이언트 코드가 연결 팩토리를 조회할 때 사용하는 이름입니다.

```
qcf= (javax.jms.QueueConnectionFactory)
      ctx.lookup("MyQueueConnectionFactory")
```
4. 폴다운 메뉴에서 QueueConnectionFactory를 선택하여 연결 팩토리 유형을 지정합니다.

5. 연결 처리 탭을 누릅니다.
6. 메시지 서버 주소 목록 필드에는 일반적으로 클라이언트가 연결할 브로커의 주소를 입력합니다. 다음은 이 필드의 예입니다.

```
mq://localhost:7676/jms
```

기본적으로 연결 팩토리가 로컬 호스트의 포트 7676에서 실행 중인 브로커에 연결하도록 구성되므로 (이 자습서 예에서 사용하는 구성) 값을 입력할 필요가 없습니다.
7. 이 대화 상자에 있는 탭을 차례로 눌러 연결 팩토리에 대해 구성 가능한 정보 종류를 확인합니다. 연결 팩토리 객체 추가 대화 상자의 오른쪽 아래에 있는 도움말 버튼을 사용하여 각 탭에 대한 정보를 얻을 수 있습니다. 지금은 기본값을 바꾸지 마십시오.
8. 확인을 눌러 대기열 연결 팩토리를 만듭니다.
9. 결과 창을 확인합니다. 새로 만든 연결 팩토리의 조회 이름 및 유형이 표시됩니다.

대상 관리 대상 객체 추가

대상 관리 대상 객체는 브로커의 물리적 대상과 관련됩니다. 즉 이 대상을 가리키기 때문에 공급자별로 사용하는 대상 이름 지정 및 구성 방식과는 상관 없이, 클라이언트가 물리적 대상을 조회하고 찾을 수 있습니다.

클라이언트는 메시지를 보낼 때 관리 대상 객체를 조회(또는 인스턴스화)하고 이를 JMS API의 `send()` 메소드에서 참조합니다. 그런 다음 브로커는 관리 대상 객체에 해당되는 물리적 대상에게 메시지를 전달합니다.

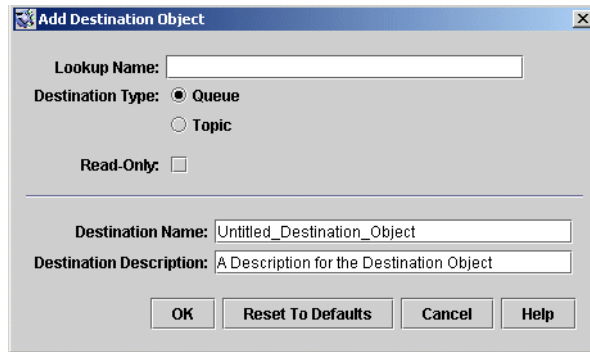
- 관리 대상 객체에 해당되는 물리적 대상이 작성되어 있다면, 브로커는 이 물리적 대상에게 메시지를 전달합니다.
- 물리적 대상을 만들지 않았고 물리적 대상의 자동 작성 기능이 사용 가능한 상태라면 브로커 스스로 물리적 대상을 작성하고 그 대상에게 메시지를 전달합니다.
- 물리적 대상을 만들지 않았고 물리적 대상의 자동 작성 기능도 *사용 불가* 상태라면 브로커는 물리적 대상을 생성할 수도, 메시지를 전달할 수도 없습니다.

자습서의 다음 부분에서는 이미 추가해 놓은 물리적 대상에 해당되는 관리 대상 객체를 추가합니다.

▶ 객체 저장소에 대상을 추가하는 방법

1. 탐색 창의 대상 노드(MyObjectStore 노드 아래)를 마우스 오른쪽 버튼으로 누릅니다.
2. 대상 객체 추가를 선택합니다.

관리 콘솔은 객체에 대한 정보를 지정할 때 사용하는 대상 객체 추가 대화 상자를 표시합니다.



3. 조회 이름 필드에 "MyQueue"라고 입력합니다.

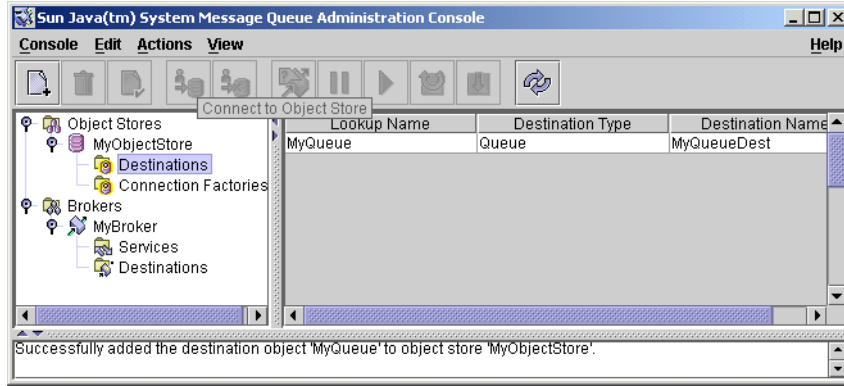
이 조회 이름은 JNDI 조회 호출을 사용하여 객체를 찾을 때 사용합니다. 이 샘플 응용 프로그램에서는 다음과 같이 호출합니다.

```
queue=(javax.jms.Queue)ctx.lookup("MyQueue");
```

4. 대상 유형의 대기열 라디오 버튼을 선택합니다.
5. 대상 이름 필드에 MyQueueDest를 입력합니다.

이것은 브로커에 물리적 대상을 추가했을 때 지정한 이름입니다([110페이지의 "브로커에 물리적 대상 추가" 참조](#)).

6. 확인을 누릅니다.
7. 탐색 창에서 대상을 선택하고 방금 추가한 대기열 대상 관리 대상 객체에 대한 정보가 결과 창에 어떻게 표시되는지 확인합니다.

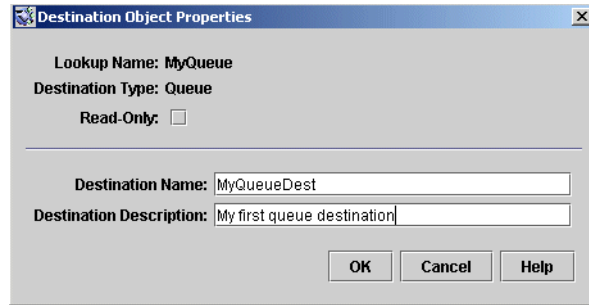


관리 대상 객체 등록 정보

관리 대상 객체의 등록 정보를 확인하거나 업데이트하려면 탐색 창에서 대상이나 연결 팩토리를 선택하고 결과 창에서 특정 객체를 선택한 다음 작업 > 등록 정보를 선택해야 합니다.

▶ 대상 객체의 등록 정보를 확인 또는 업데이트하는 방법

1. 탐색 창에서 MyObjectStore의 대상 노드를 선택합니다.
2. 결과 창에서 MyQueue를 선택합니다.
3. 작업 > 등록 정보를 선택하여 대상 객체 등록 정보 대화 상자를 확인합니다.
 변경 가능한 값은 대상 이름과 설명뿐입니다. 조회 이름을 변경하려면 객체를 삭제한 다음 원하는 조회 이름으로 새 대기열 관리 대상 객체를 추가해야 합니다.
4. 취소를 눌러 대화 상자를 닫습니다.



콘솔 정보 업데이트

객체 저장소나 브로커로 작업하는 모든 경우에 보기 > 새로 고침을 선택하면 어떤 요소나 요소 그룹의 시각적 표시를 업데이트할 수 있습니다.

샘플 응용 프로그램 실행

이 자습서에서 사용할 수 있도록 샘플 응용 프로그램 HelloWorldMessageJNDI가 제공됩니다(위치는 아래의 1단계 참조). 이 응용 프로그램은 이 자습서의 지금까지 만든 물리적 대상 및 관리 대상 객체, 즉 MyQueueDest라는 대기열 물리적 대상, JNDI 조회 이름이 각각 MyQueueConnectionFactory 및 MyQueue인 대기열 연결 팩토리 관리 대상 객체와 대기열 관리 객체를 사용합니다.

이 코드는 간단한 대기열 발신기 및 수신기를 만들고 "Hello World" 메시지를 보내고 받습니다.

▶ **HelloWorldMessageJNDI 응용 프로그램을 실행하는 방법**

1. HelloWorldmessageJNDI 응용 프로그램을 포함하는 디렉토리가 현재 디렉토리가 되게 합니다. 예를 들면 다음과 같습니다.

```
cd IMQ_HOME\demo\helloworld\helloworldmessagejndi (Windows)
```

```
cd /usr/demo/imq/helloworld/helloworldmessagejndi (Solaris)
```

```
cd /opt/imq/demo/helloworld/helloworldmessagejndi (Linux)
```

HelloWorldMessageJNDI.class 파일이 있는지 확인합니다(응용 프로그램을 변경할 경우, *Message Queue C Client Developer's Guide*의 Quick Start Tutorial에서 소개하는 클라이언트 응용 프로그램 컴파일 지침을 사용하여 다시 컴파일해야 함).

2. CLASSPATH 변수가 HelloWorldMessageJNDI.class 파일을 비롯하여 Message Queue 제품에 포함된 jms.jar, imq.jar 및 fscontext.jar 파일이 들어 있는 현재 디렉토리를 포함하도록 설정합니다. CLASSPATH 설정 지침은 *Message Queue Java Client Developer's Guide*를 참조하십시오.

JNDI jar 파일(jndi.jar)은 JDK 1.4와 함께 제공됩니다. 이 JDK를 사용하는 경우 jndi.jar 파일을 CLASSPATH 설정에 추가할 필요가 없습니다. 이전 버전 JDK를 사용하는 경우 jndi.jar을 CLASSPATH에 포함시켜야 합니다. 자세한 내용은 *Message Queue Java Client Developer's Guide*를 참조하십시오.

3. 응용 프로그램을 실행하기 전에 HelloWorldMessageJNDI.java 소스 파일을 열고 확인합니다. 소스는 간단하지만 확실하게 기록되어 있으며, 이 자습서에서 만든 관리 대상 객체 및 대상을 어떻게 사용하는지에 대해 명확하게 설명합니다.
4. 아래 명령 중 하나를 실행하여 HelloWorldMessageJNDI 응용 프로그램을 실행합니다.

```
java HelloWorldMessageJNDI (Windows)
```

```
% java HelloWorldMessageJNDI file:///tmp (Solaris 및 Linux)
```

응용 프로그램이 성공적으로 실행되면 다음과 같이 출력됩니다.

```
java HelloWorldMessageJNDI
Using file:///C:/Temp for Context.PROVIDER_URL

Looking up Queue Connection Factory object with lookup name:
MyQueueConnectionFactory
Queue Connection Factory object found.
Looking up Queue object with lookup name: MyQueue
Queue object found.

Creating connection to broker.
Connection to broker created.

Publishing a message to Queue: MyQueueDest
Received the following message: Hello World
```

샘플 응용 프로그램 실행

브로커 시작 및 구성

Sun Java™ System Message Queue를 설치한 후 `imqbrokerd` 명령을 사용하여 브로커를 시작합니다. 브로커 인스턴스의 구성은 일련의 구성 파일 및 `imqbrokerd` 명령과 함께 전달되는 옵션에 따라 결정되며, 옵션은 구성 파일의 해당 등록 정보를 대체합니다.

이 장에서는 `imqbrokerd` 명령의 구문, 그리고 명령줄 옵션 및 구성 파일을 사용하여 브로커 인스턴스를 구성하는 방법에 대해 설명합니다. 또한 다음을 수행하는 방법도 설명합니다.

- 브로커 인스턴스 구성 파일 편집
- 브로커 클러스터를 이용한 작업
- 브로커의 로깅 제어

브로커를 Windows 서비스로 시작하고 사용하는 방법에 관한 설명은 [333페이지의 "브로커를 Windows 서비스로 사용"](#)을 참조하십시오.

구성 파일

브로커를 구성할 때 사용되는 설치한 브로커 구성 파일 템플릿은 [부록 A, "Message Queue 데이터의 위치"](#)에 나와 있는 것처럼 운영 체제에 따라 다른 디렉토리에 있습니다.

이 디렉토리에는 다음과 같은 파일이 저장됩니다.

- 시작할 때 로드되는 **기본 구성 파일**. 이 파일은 `default.properties`이며 편집할 수 없습니다. 기본 설정을 지정하고 변경할 등록 정보의 정확한 이름을 찾으려면 이 파일을 읽습니다.

- Message Queue를 설치할 때 지정한 모든 등록 정보를 포함하는 **설치 구성 파일**. 이 파일은 `install.properties`이며 설치하고 나면 편집할 수 없습니다.

인스턴스 구성 파일

브로커를 처음 실행하면 브로커 인스턴스의 구성 등록 정보를 지정할 때 사용할 수 있는 인스턴스 구성 파일이 만들어집니다. 인스턴스 구성 파일은 구성 파일이 연결되어 있는 브로커 인스턴스의 이름(*instanceName*)으로 식별되는 디렉토리에 저장됩니다(**부록 A**, "**Message Queue 데이터의 위치**" 참조).

```
.../instances/instanceName/props/config.properties
```

주 `.../instances/instanceName` 디렉토리 및 인스턴스 구성 파일은 해당 브로커 인스턴스를 만든 사람이 소유합니다. 이후에는 항상 해당 사용자가 브로커 인스턴스를 시작해야 합니다.

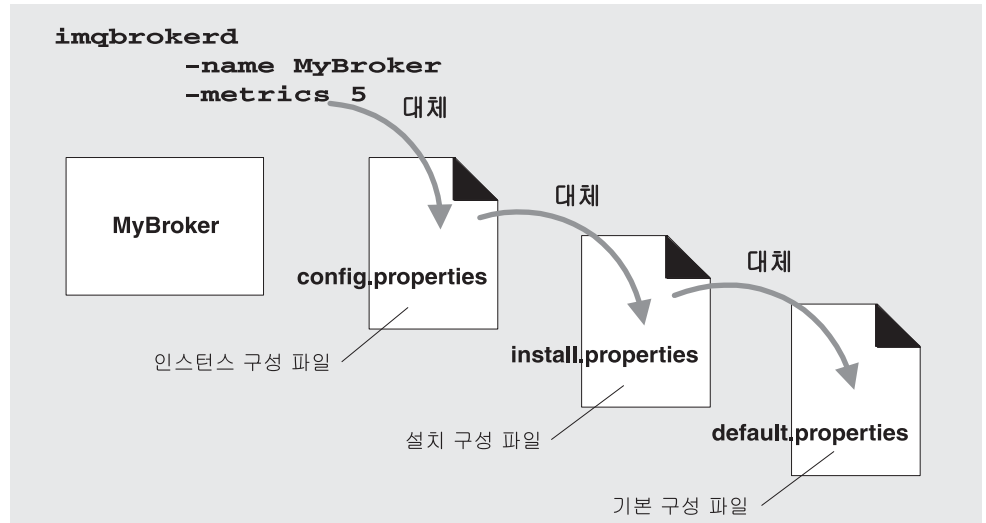
인스턴스 구성 파일은 브로커 인스턴스에 의해 유지 관리됩니다. 관리 도구를 사용하여 구성을 변경하면 인스턴스 구성 파일이 수정됩니다. 인스턴스 구성 파일을 직접 편집하여 구성을 변경할 수도 있습니다(**129페이지의 "인스턴스 구성 파일 편집"** 참조). 그렇게 하려면 `.../instances/instanceName` 디렉토리의 소유자이거나 루트로 로그인하여 디렉토리에 관한 권한을 변경해야 합니다.

클러스터에서 브로커 인스턴스를 연결하는 경우에는(**82페이지의 "멀티 브로커 클러스터 (엔터프라이즈판)"** 참조) *클러스터 구성 파일*을 사용하여 클러스터 구성 정보를 지정해야 할 수도 있습니다. 자세한 내용은 **140페이지의 "클러스터 구성 등록 정보"**을 참조하십시오.

등록 정보 값 병합

시작할 때 시스템에서는 다른 구성 파일에 있는 등록 정보 값을 병합합니다. 설치 및 인스턴스 구성 파일에 설정된 값을 사용하여 기본 구성 파일에 지정된 값을 대체합니다. `mqbrokerd` 명령 옵션을 사용하면 결과 값을 대체할 수 있습니다. 이 체계가 **그림 5-1**에 표시되어 있습니다.

그림 5-1 브로커 구성 파일



등록 정보 이름 지정 구문

구성 파일의 모든 Message Queue 등록 정보 정의에는 다음과 같은 이름 지정 구문이 사용됩니다.

```
propertyName=value [, value1]...
```

예를 들어, 다음 항목은 브로커가 추가 메시지를 거부하기 전까지 메모리 및 영구 저장소에 최대 50,000개의 메시지를 저장하도록 지정합니다.

```
imq.system.max_count=50000
```

다음 항목은 매일(86400초) 새 로그 파일을 작성하도록 지정합니다.

```
imq.log.file.rolloversecs=86400
```

130페이지의 표 5-1에는 브로커 구성 등록 정보(및 기본값)가 알파벳순으로 나와 있습니다.

인스턴스 구성 파일 편집

브로커 인스턴스를 처음으로 실행하면 `config.properties` 파일이 자동으로 작성됩니다. 이 인스턴스 구성 파일을 편집하여 해당 브로커 인스턴스의 동작과 자원을 사용자 정의할 수 있습니다.

브로커 인스턴스는 시작 시에만 config.properties 파일을 읽습니다.
config.properties 파일을 영구적으로 변경하려면 다음 중 하나를 수행합니다.

- 관리 도구를 사용합니다. imqcmd를 사용해서 설정할 수 있는 등록 정보에 대한 내용은 [160페이지의 표 6-4](#)를 참조하십시오.
- 브로커 인스턴스가 종료된 사이에 config.properties 파일을 편집한 후 인스턴스를 다시 시작합니다(Solaris와 Linux 플랫폼에서는 브로커 인스턴스를 처음으로 시작한 사용자만 config.properties 파일을 편집할 권한을 갖음).

[표 5-1](#)에는 브로커 인스턴스 구성 등록 정보(및 기본값)가 알파벳순으로 나와 있습니다.
각 등록 정보의 의미와 사용에 대한 자세한 내용은 지정된 상호 참조 부분을 보십시오.

표 5-1 브로커 인스턴스 구성 등록 정보

등록 정보 이름	유형	기본값	참조
imq.accesscontrol.enabled	부울	true	69페이지의 표 2-6
imq.accesscontrol.file.filename	문자열	accesscontrol.properties	69페이지의 표 2-6
imq.authentication.user_repository	문자열	file	69페이지의 표 2-6
imq.authentication.client.response.timeout	정수 (초)	180	69페이지의 표 2-6
imq.authentication.type	문자열	digest	69페이지의 표 2-6
imq.autocreate.destination.isLocalOnly	부울	false	79페이지의 표 2-10
imq.autocreate.destination.limitBehavior	문자열	REJECT_NEWEST	79페이지의 표 2-10
imq.autocreate.destination.maxBytesPerMsg	바이트 문자열 ¹	10k	79페이지의 표 2-10
imq.autocreate.destination.maxNumMsgs	정수	100,000	79페이지의 표 2-10
imq.autocreate.destination.maxNumProducers	정수	100	79페이지의 표 2-10
imq.autocreate.destination.maxTotalMsgBytes	바이트 문자열 ¹	10m	79페이지의 표 2-10
imq.autocreate.queue	부울	true	79페이지의 표 2-10

¹ 바이트 문자열로 입력하는 값은 바이트, KB, MB로 나타낼 수 있습니다. 예를 들어, 1000은 1000바이트를, 7500b는 7500바이트를, 77k는 77KB (77 x 1024 = 78848바이트)를, 17m은 17MB (17 x 1024 x 1024 = 17825792바이트)를 나타냅니다.

표 5-1 브로커 인스턴스 구성 등록 정보(계속)

등록 정보 이름	유형	기본값	참조
imq.autocreate.queue.consumerFlowLimit	정수	1000	79페이지의 표 2-10
imq.autocreate.queue.localDeliveryPreferred	부울	false	79페이지의 표 2-10
imq.autocreate.queue.maxNumActiveConsumers	정수	1	79페이지의 표 2-10
imq.autocreate.queue.maxNumBackupConsumers	정수	0	79페이지의 표 2-10
imq.autocreate.topic	부울	true	79페이지의 표 2-10
imq.autocreate.topic.consumerFlowLimit	정수	1,000	79페이지의 표 2-10
imq.cluster. <i>property_name</i>			140페이지의 표 5-3
imq.hostname	문자열	사용 가능한 모든 IP 주소	57페이지의 표 2-3
imq.httpjms.http. <i>property_name</i>			311페이지의 표 C-1
imq.httpsjms.https. <i>property_name</i>			323페이지의 표 C-3
imq.keystore. <i>property_name</i>			220페이지의 표 8-8
imq.log.console.output	문자열	ERROR WARNING	74페이지의 표 2-9
imq.log.console.stream	문자열	ERR	74페이지의 표 2-9
imq.log.file.dirpath	문자열	부록 A, "Message Queue 데이터의 위치" 참조	74페이지의 표 2-9
imq.log.file.filename	문자열	log.txt	74페이지의 표 2-9
imq.log.file.output	문자열	ALL	74페이지의 표 2-9
imq.log.file.rolloverbytes	정수 (바이트)	-1 (롤오버 없음)	74페이지의 표 2-9
imq.log.file.rolloversecs	정수 (초)	604800	74페이지의 표 2-9
imq.log.level	문자열	INFO	74페이지의 표 2-9
imq.log.syslog.facility	문자열	LOG_DAEMON	74페이지의 표 2-9
imq.log.syslog.identity	문자열	imqbrokerd_\${imq.instanceName}	74페이지의 표 2-9

1 바이트 문자열로 입력하는 값은 바이트, KB, MB로 나타낼 수 있습니다. 예를 들어, 1000은 1000바이트를, 7500b는 7500바이트를, 77k는 77KB (77 x 1024 = 78848바이트)를, 17m은 17MB (17 x 1024 x 1024 = 17825792바이트)를 나타냅니다.

표 5-1 브로커 인스턴스 구성 등록 정보(계속)

등록 정보 이름	유형	기본값	참조
imq.log.syslog.logconsole	부울	false	74페이지의 표 2-9
imq.log.syslog.logpid	부울	true	74페이지의 표 2-9
imq.log.syslog.output	문자열	ERROR	74페이지의 표 2-9
imq.log.timezone	문자열	지역 표준 시간대	74페이지의 표 2-9
imq.message.expiration.interval	정수 (초)	60	62페이지의 표 2-4
imq.message.max_size	바이트 문자열 ¹	70m	62페이지의 표 2-4
imq.metrics.enabled	부울	true	74페이지의 표 2-9
imq.metrics.interval	정수 (초)	-1 (해당 없음)	74페이지의 표 2-9
imq.metrics.topic.enabled	부울	true	74페이지의 표 2-9
imq.metrics.topic.interval	정수 (초)	60	74페이지의 표 2-9
imq.metrics.topic.persist	부울	false	74페이지의 표 2-9
imq.metrics.topic.timetolive	정수 (초)	300	74페이지의 표 2-9
imq.passfile.dirpath	문자열	부록 A, "Message Queue 데이터의 위치" 참조	69페이지의 표 2-6
imq.passfile.enabled	부울	false	69페이지의 표 2-6
imq.passfile.name	문자열	passfile	69페이지의 표 2-6
imq.persist.file.destination.message.filepool.limit	정수	100	66페이지의 표 2-5
imq.persist.file.message.cleanup	부울	false	66페이지의 표 2-5
imq.persist.file.message.filepool.cleanratio	정수	0	66페이지의 표 2-5
imq.persist.file.message.max_record_size	바이트 문자열 ¹	1m	66페이지의 표 2-5
imq.persist.file.sync.enabled	부울	false	66페이지의 표 2-5

¹ 바이트 문자열로 입력하는 값은 바이트, KB, MB로 나타낼 수 있습니다. 예를 들어, 1000은 1000바이트를, 7500b는 7500바이트를, 77k는 77KB (77 x 1024 = 78848바이트)를, 17m은 17MB (17 x 1024 x 1024 = 17825792바이트)를 나타냅니다.

표 5-1 브로커 인스턴스 구성 등록 정보(계속)

등록 정보 이름	유형	기본값	참조
<code>imq.persist.jdbc.property_name</code>			300페이지의 표 B-1
<code>imq.persist.store</code>	문자열	file	66페이지의 표 2-5
<code>imq.ping.interval</code>	정수	120	57페이지의 표 2-3
<code>imq.portmapper.backlog</code>	정수	50	57페이지의 표 2-3
<code>imq.portmapper.hostname</code>	문자열	<code>imq.hostname</code> 에서 상속됨	57페이지의 표 2-3
<code>imq.portmapper.port</code>	정수	7676	57페이지의 표 2-3
<code>imq.resource_state.count</code>	정수 (퍼센트)	5000 (녹색) 500 (노랑) 50(주황) 0 (빨강)	62페이지의 표 2-4
<code>imq.resource_state.threshold</code>	정수 (퍼센트)	0 (녹색) 80 (노랑) 90(주황) 98 (빨강)	62페이지의 표 2-4
<code>imq.service.activelist</code>	목록	jms, admin	57페이지의 표 2-3
<code>imq.service_name.accesscontrol.enabled</code>	부울	시스템 차원 등록 정보에서 값 상속	69페이지의 표 2-6
<code>imq.service_name.accesscontrol.file.filename</code>	문자열	시스템 차원 등록 정보에서 값 상속	69페이지의 표 2-6
<code>imq.service_name.authentication.type</code>	문자열	시스템 차원 등록 정보에서 값 상속	69페이지의 표 2-6
<code>imq.service_name.max_threads</code>	정수	1000 (jms) 500 (ssljms) 500 (httpjms) 500 (httpsjms) 10 (admin) 10 (ssladmin)	57페이지의 표 2-3
<code>imq.service_name.min_threads</code>	정수	10 (jms) 10 (ssljms) 10 (httpjms) 10 (httpsjms) 4 (admin) 4 (ssladmin)	57페이지의 표 2-3
<code>imq.service_name.protocol_type.hostname</code>	문자열	<code>imq.hostname</code> 에서 상속됨	57페이지의 표 2-3

1 *바이트 문자열*로 입력하는 값은 바이트, KB, MB로 나타낼 수 있습니다. 예를 들어, 1000은 1000바이트를, 7500b는 7500바이트를, 77k는 77KB (77 x 1024 = 78848바이트)를, 17m은 17MB (17 x 1024 x 1024 = 17825792바이트)를 나타냅니다.

표 5-1 브로커 인스턴스 구성 등록 정보(계속)

등록 정보 이름	유형	기본값	참조
<code>imq.service_name.protocol_type.port</code>	정수	0 (동적으로 할당됨)	57페이지의 표 2-3
<code>imq.service_name.threadpool_model</code>	문자열	dedicated (jms) dedicated (ssljms) dedicated (httpjms) dedicated (httpsjms) dedicated (admin) dedicated (ssladmin)	57페이지의 표 2-3
<code>imq.shared.connectionMonitor_limit</code>	정수	512 (Solaris & Linux) 64 (Windows)	57페이지의 표 2-3
<code>imq.system.max_count</code>	정수, 0 (제한 없음)	-1	62페이지의 표 2-4
<code>imq.system.max_size</code>	바이트 문자열 ¹ , 0 (제한 없음)	-1	62페이지의 표 2-4
<code>imq.transaction.autorollback</code>	부울	false	62페이지의 표 2-4
<code>imq.user_repository.ldap.property_name</code>			210페이지의 표 8-5

¹ 바이트 문자열로 입력하는 값은 바이트, KB, MB로 나타낼 수 있습니다. 예를 들어, 1000은 1000바이트를, 7500b는 7500바이트를, 77k는 77KB (77 x 1024 = 78848바이트)를, 17m은 17MB (17 x 1024 x 1024 = 17825792바이트)를 나타냅니다.

브로커 시작

브로커 인스턴스를 시작하려면 `imqbrokerd` 명령을 사용합니다.

주 관리 콘솔(`imqadmin`)이나 명령 유틸리티(`imqcmd`)를 사용해서는 브로커 인스턴스를 시작할 수 없습니다. 이 Message Queue 관리 도구를 사용하려면 브로커 인스턴스가 이미 실행 중이어야 합니다.

등록 정보 값을 하나 이상 대체하려면 유효한 `imqbrokerd` 명령줄 옵션을 지정합니다. 명령줄 옵션은 브로커 구성 파일의 값을 대체하지만 현재 브로커 세션에만 적용됩니다. 명령줄 옵션은 인스턴스 구성 파일에 기록되지 않습니다.

imqbrokerd 명령 구문

imqbrokerd 명령의 구문은 다음과 같습니다(옵션 및 인수는 공백으로 구분).

```
imqbrokerd [[ -Dproperty=value]...]
[ -backup fileName]
[ -cluster "[broker1] [[,broker2]...]"]
[ -dbuser userName] [ -dbpassword password]
[ -force]
[ -h|-help]
[ -javahome path]
[ -ldappassword password]
[ -license licenseName]
[ -loglevel level]
[ -metrics interval]
[ -name instanceName]
[ -password keypassword] [ -passfile fileName]
[ -port number]
[ -remove instance]
[ -reset data]
[ -restore fileName]
[ -shared]
[ -silent|-s] [ -tty]
[ -upgrade-store-nobackup]
[ -version]
[ -vmargs arg1 [[arg2]...]
```

주 Solaris의 경우에는 /etc/imq/imqborkerd.conf 구성 파일의 RESTART 등록 정보를 YES로 설정하여 비정상적으로 종료된 브로커를 자동으로 시작하도록 구성할 수 있습니다.

주 Solaris와 Linux 플랫폼에서는 구성 정보와 지속성 데이터를 포함하는 디렉토리의 권한이 브로커 인스턴스를 처음으로 시작하는 사용자의 umask에 따라 달라집니다. 따라서 브로커 인스턴스가 제대로 작동하려면 원래 사용자가 계속 시작해야 합니다.

시작 예

다음에서는 `imqbrokerd` 명령을 사용하는 예를 보여 줍니다. `imqbrokerd` 명령줄 옵션에 대한 자세한 내용은 [136페이지의 표 5-2](#)를 참조하십시오.

- ▶ **기본 브로커 이름 및 구성을 사용하는 브로커 인스턴스를 시작하는 방법**
다음 명령을 사용합니다.

```
imqbrokerd
```

그러면 포트 7676에 포트 매핑이 있는 로컬 시스템에서 브로커의 기본 인스턴스(이름 `imqbroker`)를 시작합니다.

- ▶ **엔터프라이즈판 시험 사용권으로 브로커 인스턴스를 시작하는 방법**

플랫폼판 사용권은 있지만 90일 동안 엔터프라이즈판 기능을 사용해 보려는 경우 다음과 같이 `-license` 명령줄 옵션을 사용하고 사용할 사용권으로 "try"를 전달하여 엔터프라이즈판 시험 사용권을 활성화할 수 있습니다.

```
imqbrokerd -license try
```

이 옵션은 브로커 인스턴스를 시작할 때마다 사용해야 하며, 그렇지 않으면 기본값이 기본 플랫폼판 사용권으로 돌아갑니다.

- ▶ **플러그인 지속성을 사용하여 명명된 브로커 인스턴스를 시작하는 방법**

플러그인 데이터 저장소([297페이지의 부록 B, "플러그인 지속성 설정"](#) 참조)를 사용하며 사용자 아이디와 비밀번호가 필요한 `myBroker`라는 브로커를 시작하려면 다음 명령을 사용합니다.

```
imqbrokerd -name myBroker -dbuser myName -dbpassword myPassword
```

imqbrokerd 옵션 요약

[표 5-2](#)에는 `imqbrokerd` 명령의 옵션과 각 옵션에 영향을 받는 구성 등록 정보(있는 경우)에 대한 설명이 나와 있습니다.

표 5-2 `imqbrokerd` 옵션

옵션	영향을 받는 등록 정보	설명
<code>-backup fileName</code>	없음	브로커 클러스터에만 적용됩니다. 마스터 브로커의 구성 변경 기록을 지정된 파일에 백업합니다. 146페이지의 "구성 변경 기록 백업" 을 참조하십시오.

표 5-2 imqbrokerd 옵션(계속)

옵션	영향을 받는 등록 정보	설명
-cluster "[broker1] [[broker2]...]" broker는 다음 중 하나입니다. • host[port] • [host]:port	imq.cluster.brokerlist를 연결할 브로커의 목록으로 설정합니다.	브로커 클러스터에만 적용됩니다. 지정된 호스트 및 포트의 모든 브로커에 연결합니다. 이 목록은 imq.cluster.brokerlist 등록 정보의 목록과 병합됩니다. host에 값을 지정하지 않으면 localhost가 사용됩니다. port에 값을 지정하지 않으면 7676이 사용됩니다. 이 옵션을 사용해서 여러 브로커에 연결하는 방법에 대한 자세한 내용은 140페이지의 "클러스터를 이용한 작업(엔터프라이즈판)"을 참조하십시오.
-dbpassword password	imq.persist.jdbc.password를 지정한 비밀번호로 설정합니다.	플러그인 JDBC 호환 데이터 저장소의 비밀번호를 지정합니다. 부록 B, "플러그인 지속성 설정" 을 참조하십시오.
-dbuser userName	imq.persist.jdbc.user를 지정한 사용자 아이디로 설정합니다.	플러그인 JDBC 호환 데이터베이스의 사용자 아이디를 지정합니다. 부록 B, "플러그인 지속성 설정" 을 참조하십시오.
-Dproperty=value	시스템 등록 정보를 설정합니다. 인스턴스 구성 파일의 해당 등록 정보 값을 대체합니다.	지정한 등록 정보를 지정한 값으로 설정합니다. 브로커 구성 등록 정보는 130페이지의 표 5-1을 참조하십시오. 경고: D 옵션으로 설정한 등록 정보의 맞춤법과 형식에 주의하십시오. 잘못된 값을 전달한 경우 시스템에서는 경고를 하지 않으며 Message Queue에서는 해당 값을 설정할 수 없습니다.
-force	없음	사용자의 확인 없이 작업을 수행합니다. 이 옵션은 일반적으로 확인이 필요한 -remove instance 및 -upgrade-store-nobackup 옵션에만 적용됩니다.
-h -help	없음	도움말을 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
-javahome path	없음	대체 Java 2 호환 JDK의 경로를 지정합니다. 기본값은 번들형 런타임을 사용하는 것입니다.
-ldappassword password	imq.user_repository.ldap.password를 지정한 비밀번호로 설정합니다.	LDAP 사용자 저장소에 액세스할 때 필요한 비밀번호를 지정합니다. 209페이지의 "사용자 저장소에 LDAP 서버 사용"을 참조하십시오.

표 5-2 imqbrokerd 옵션(계속)

옵션	영향을 받는 등록 정보	설명
-license <i>[licenseName]</i>	없음	로드 사용권이 Message Queue 제품의 기본값과 다른 경우 해당 사용권을 지정합니다. 사용권 이름을 지정하지 않으면 시스템에 설치된 모든 사용권이 표시됩니다. <i>licenseName</i> 값은 설치된 Message Queue 버전에 따라 pe (플랫폼판-기본 기능), try (플랫폼판 -90일 시험 엔터프라이즈 기능), un1 (엔터프라이즈 판)입니다. 33페이지 의 "제품 판"을 참조하십시오.
-loglevel <i>level</i>	imq.broker.log.level을 지정한 수준으로 설정합니다.	로깅 수준을 NONE, ERROR, WARNING 또는 INFO 중 하나로 지정합니다. 기본값은 INFO입니다. 자세한 내용은 71페이지 의 "로거"를 참조하십시오.
-metrics <i>interval</i>	imq.metrics.interval을 지정한 초 수로 설정합니다.	지정된 간격(초)으로 브로커 메트릭이 로거에 기록되도록 지정합니다.
-name <i>instanceName</i>	imq.instancename을 지정한 이름으로 설정합니다.	이 브로커의 인스턴스 이름을 지정하고 해당 인스턴스 구성 파일을 사용합니다. 브로커 이름을 지정하지 않으면 인스턴스 이름은 imqbroker로 설정됩니다. 주: 같은 호스트에서 두 개 이상의 브로커 인스턴스를 실행하는 경우에는 각각에 고유한 이름이 있어야 합니다.
-passfile <i>fileName</i>	imq.passfile.enabled를 true로 설정합니다. imq.passfile.dirpath를 파일이 포함된 경로로 설정합니다. imq.passfile.name을 파일 이름으로 설정합니다.	SSL 키 저장소, LDAP 사용자 저장소 또는 JDBC 호환 데이터베이스의 비밀번호를 읽을 파일의 이름을 지정합니다. 자세한 내용은 225페이지 의 "Passfile 사용"을 참조하십시오.
-password <i>keypassword</i>	imq.keystore.password를 지정한 비밀번호로 설정합니다.	SSL 인증 키 저장소의 비밀번호를 지정합니다. 자세한 내용은 66페이지 의 "보안 관리자"를 참조하십시오.
-port <i>number</i>	imq.portmapper.port를 지정한 번호로 설정합니다.	브로커의 포트 매퍼 포트 번호를 지정합니다. 이 값은 기본적으로 7676으로 설정됩니다. 같은 서버에서 두 개의 브로커 인스턴스를 실행하려면 각 브로커 포트 매퍼에 서로 다른 포트 번호가 있어야 합니다. Message Queue 클라이언트는 이 포트 번호를 사용하여 브로커 인스턴스에 연결합니다.
-remove instance	없음	브로커 인스턴스를 제거합니다. 인스턴스 구성 파일, 로그 파일, 영구 저장소, 인스턴스와 관련된 기타 파일 및 디렉토리를 삭제합니다. -force 옵션을 함께 지정한 경우가 아니면 사용자의 확인이 필요합니다.

표 5-2 imqbrokerd 옵션(계속)

옵션	영향을 받는 등록 정보	설명
-reset store messages durables props	없음	주어진 인수에 따라 데이터 저장소(또는 데이터 저장소의 일부) 또는 브로커 인스턴스의 등록 정보를 재설정합니다. 데이터 저장소를 재설정하면 지속성 메시지, 지속성 가입, 트랜잭션 정보 등의 모든 지속성 데이터가 지워집니다. 그러면 브로커 인스턴스를 초기 상태로 시작할 수 있습니다. 지속성 메시지만 지우거나 영구 가입만 지울 수도 있습니다. (이후에 다시 시작할 때 영구 저장소가 재설정되지 않게 하려면 -reset 옵션을 사용하지 않고 브로커 인스턴스를 다시 시작합니다.) 자세한 내용은 63페이지의 "지속성 관리자"를 참조하십시오. 브로커의 등록 정보를 재설정하면 기존 인스턴스 구성 파일(config.properties)이 빈 파일로 교체됩니다. 모든 등록 정보에 기본값이 설정됩니다.
-restore fileName	없음	브로커 클러스터에만 적용됩니다. 마스터 브로커의 구성 변경 레코드를 지정한 백업 파일로 교체합니다. 이 파일은 -backup 옵션을 사용해서 미리 작성해 두어야 합니다. 146페이지의 "구성 변경 기록 복원"을 참조하십시오.
-shared	imq.jms.threadpool_model을 shared로 설정합니다.	연결 간에 스레드를 공유하여 브로커 인스턴스가 지원하는 연결의 수를 늘리기 위해 jms 연결 서비스가 공유 스레드 풀 모델을 사용해서 구현되도록 지정합니다. 자세한 내용은 54페이지의 "연결 서비스"를 참조하십시오.
-silent -s	imq.log.console.output을 NONE으로 설정합니다.	콘솔에 대한 로깅을 끕니다.
-tty	imq.log.console.output을 ALL로 설정합니다.	모든 메시지가 콘솔에 표시되도록 설정합니다. 기본적으로 WARNING 및 ERROR 수준 메시지만 표시됩니다.
-upgrade-store-nobackup	없음	호환되지 않는 버전에서 Message Queue 3.5 또는 Message Queue 3.5 SPx로 업그레이드할 때 이전 데이터 저장소를 자동으로 제거하도록 지정합니다. 자세한 내용은 Message Queue 설치 설명서를 참조하십시오.
-version	없음	설치된 제품의 버전 번호를 표시합니다.

표 5-2 imqbrokerd 옵션(계속)

옵션	영향을 받는 등록 정보	설명
-vmargs <i>arg1</i> [[<i>arg2</i>]...]	없음	Java VM으로 전달할 인수를 지정합니다. 인수는 공백으로 구분합니다. 인수를 두 개 이상 전달하거나 인수에 공백이 포함된 경우에는 따옴표를 사용합니다. 예를 들면 다음과 같습니다. imqbrokerd -tty -vmargs "-Xmx128m -Xincgc"

클러스터를 이용한 작업(엔터프라이즈판)

이 절에서는 멀티 브로커 클러스터를 구성할 때 사용하는 등록 정보, 브로커를 연결하는 두 가지 방법, 클러스터 관리 방법 등에 대해 설명합니다. 클러스터에 대한 소개는 [82페이지](#)의 "멀티 브로커 클러스터(엔터프라이즈판)"를 참조하십시오.

클러스터를 사용하여 작업할 때에는 클러스터에 있는 모든 브로커의 호스트 사이에서 클럭이 동기화되어 있는지 확인합니다([337페이지](#)의 "시스템 클럭 설정" 참조).

클러스터 구성 등록 정보

브로커를 클러스터에 연결할 때에는 연결된 모든 브로커가 일련의 클러스터 구성 등록 정보로 지정되어야 합니다. 다음 등록 정보는 클러스터의 브로커 참가에 대해 설명합니다. [표 5-3](#)에는 클러스터와 관련된 구성 등록 정보가 요약되어 있습니다. 별표(*)가 표시된 등록 정보는 클러스터의 모든 브로커에 대해 같은 값을 지정해야 합니다.

표 5-3 클러스터 구성 등록 정보

등록 정보 이름	설명
imq.cluster.brokerlist*	클러스터의 모든 브로커를 지정합니다. <i>host:port</i> 항목이 쉼표로 구분된 목록으로 구성되며, 여기서 <i>host</i> 는 각 브로커의 호스트 이름, <i>port</i> 는 포트 매핑의 포트 번호입니다. 예를 들면 다음과 같습니다. host1:3000, host2:8000, ctrhost
imq.cluster.masterbroker*	클러스터 중에서 상태 변경을 추적하는 마스터 브로커(있는 경우)를 지정합니다. 등록 정보는 <i>host:port</i> 로 구성되며, 여기서 <i>host</i> 는 마스터 브로커의 호스트 이름, <i>port</i> 는 포트 매핑의 포트 번호입니다. 작업 환경에 대해 이 등록 정보를 설정합니다. 예: ctrhost:7676

표 5-3 클러스터 구성 등록 정보(계속)

등록 정보 이름	설명
<code>imq.cluster.url*</code>	클러스터 구성 파일의 위치를 지정합니다. 브로커가 개별적으로 구성되지 않고 단일 중앙 클러스터 구성 파일을 참조하는 경우에 사용합니다. URL 문자열로 구성됩니다. Web Server에서 관리할 경우 일반 <code>http:URL</code> 을 사용하여 액세스할 수 있습니다. 공유 드라이브에 있는 경우 <code>file:URL</code> 을 사용하여 액세스할 수 있습니다. 예: <code>http://webserver/imq/cluster.properties</code> <code>file:/net/mfssserver/imq/cluster.properties</code>
<code>imq.cluster.port</code>	클러스터에 있는 각 브로커는 클러스터 연결 서비스를 위한 포트 번호를 지정하는 데 사용할 수 있습니다. 클러스터 연결 서비스는 클러스터 내의 브로커간 내부 통신에 사용됩니다. 기본값: 0 (포트가 동적으로 할당됨)
<code>imq.cluster.hostname</code>	클러스터에 있는 각 브로커는 사용 가능한 호스트가 두 개 이상 있는 경우(예를 들어 컴퓨터에 네트워크 인터페이스 카드가 둘 이상인 경우) 클러스터 연결 서비스를 바인드할 호스트(호스트 이름 또는 IP 주소)를 지정하는 데 사용할 수 있습니다. 클러스터 연결 서비스는 클러스터 내의 브로커간 내부 통신에 사용됩니다. 기본값: <code>imq.hostname</code> 값 상속(57페이지의 표 2-3 참조)
<code>imq.cluster.transport*</code>	클러스터 연결 서비스가 클러스터 내의 브로커간 내부 통신에 사용하는 네트워크 전송을 지정합니다. 브로커 간에 암호화된 보안 메시지를 전달하려면 클러스터 내의 모든 브로커에 대해 이 등록 정보를 <code>ssl</code> 로 설정합니다. 기본값: <code>tcp</code>

두 방법 중 하나를 사용해서 클러스터 등록 정보를 설정할 수 있습니다.

- 클러스터와 관련된 구성 등록 정보는 각 브로커의 인스턴스 구성 파일(또는 각 브로커를 시작하는 명령줄)에서 설정할 수 있습니다. 예를 들어, 브로커 A (`host1`, 포트 7676), 브로커 B (`host2`, 포트 5000), 브로커 C (`ctrlhost`, 포트 7676)를 연결하는 경우 브로커 A, B, C의 인스턴스 구성 파일에서는 다음 등록 정보를 설정해야 합니다.

```
imq.cluster.brokerlist=host1, host2:5000, ctrlhost
```

클러스터 구성을 변경하기로 한 경우, 이 방법을 사용하려면 모든 브로커에서 클러스터 관련 등록 정보를 업데이트해야 합니다.

- 클러스터 구성 등록 정보를 중앙의 한 클러스터 구성 파일에서 설정합니다. 이러한 등록 정보에는 연결할 브로커 목록(`imq.cluster.brokerlist`) 및 클러스터 연결 서비스에 사용할 네트워크 전송(`imq.cluster.transport`)이 포함되며 선택적으로 마스터 브로커의 주소(`imq.cluster.masterbroker`)가 포함될 수 있습니다.

이 방법을 사용하는 경우에는 클러스터 구성 파일의 위치를 나타내는 `imq.cluster.url` 등록 정보도 설정해야 합니다(클러스터의 모든 브로커에서 설정). 유지 관리가 쉽다는 점에서 볼 때 이 방법은 클러스터 구성에 사용하는 것이 좋습니다.

다음 코드 샘플은 클러스터 구성 파일의 내용을 나타냅니다. `host1`과 `ctrlhost`는 모두 기본 포트에서 실행됩니다. 다음 등록 정보는 `host1`, `host2` 및 `ctrlhost`가 클러스터에서 연결되고 `ctrlhost`가 마스터 브로커임을 지정합니다.

```
imq.cluster.brokerlist=host1,host2:5000,ctrlhost
imq.cluster.masterbroker=ctrlhost
```

이 클러스터에 연결된 각 브로커의 인스턴스 구성 파일에는 다음과 같이 클러스터 구성 파일의 `url`이 포함되어야 합니다.

```
imq.cluster.url=file:/home/cluster.properties
```

브로커 연결

이 절에서는 브로커를 클러스터에 연결하는 방법과 클러스터에서 브로커 간에 암호화된 보안 메시지 전달을 위해 클러스터를 구성하는 방법을 설명합니다.

연결 방법

일반적으로 클러스터 구성 파일을 사용하거나 사용하지 않는 두 가지 방법으로 브로커를 클러스터에 연결합니다.

사용하는 방법에 관계 없이 시작하는 각 브로커는 5초마다 다른 브로커와의 연결을 시도합니다. 마스터 브로커가 시작되면 이 시도가 성공한 것입니다. 클러스터의 브로커가 마스터 브로커보다 먼저 시작되면 일시 중지 상태로 있으면서 클라이언트 연결을 거부합니다. 마스터 브로커가 시작되면 일시 중지된 브로커의 모든 기능을 자동으로 사용할 수 있게 됩니다.

방법 1: 클러스터 구성 파일 없이 연결

▶ 브로커를 클러스터에 연결하는 방법

1. 브로커를 시작하는 `imqbrokerd` 명령에 `-cluster` 옵션을 사용하고 연결할 브로커의 전체 목록을 `-cluster` 옵션의 인수로 지정합니다.
2. 브로커를 시작할 때 클러스터에 연결할 각 브로커에 대해 이 작업을 수행합니다.

예를 들어, 다음 명령은 새 브로커를 시작하고 `host1`의 기본 포트에서 실행 중인 브로커와 `host2`의 포트 7677에서 실행 중인 브로커, `localhost`의 포트 7678에서 실행 중인 브로커에 연결합니다.

```
imqbrokerd -cluster host1,host2:7677,:7678
```

방법 2: 클러스터 구성 파일을 사용하여 연결

연결할 브로커의 목록과 선택적인 마스터 브로커의 주소를 지정하는 클러스터 구성 파일을 만들 수도 있습니다. 이러한 클러스터 정의 방법은 작업 시스템에 보다 적합합니다. 이 방법을 사용하는 경우 클러스터의 각 브로커는 `imq.cluster.url` 등록 정보의 값이 클러스터 구성 파일을 가리키도록 설정해야 합니다.

브로커간 연결 보안

클러스터에서 브로커 간에 암호화된 보안 메시지 전달이 필요한 경우 다음과 같이 SSL 기반 전송 프로토콜을 사용하도록 클러스터 연결 서비스를 구성해야 합니다.

▶ 클러스터 내에서 보안 연결을 구성하는 방법

1. 클러스터 내의 각 브로커에 대해 SSL 기반 연결 서비스를 설정합니다.
[219페이지의 "TCP/IP에서 SSL 기반 서비스 설정"](#)의 지침을 참조하십시오.
2. `imq.cluster.transport` 클러스터 구성 등록 정보를 `ssl`로 설정합니다.
클러스터 구성 파일을 사용하지 않는 경우 클러스터의 각 브로커에 대해 이 등록 정보를 설정해야 합니다.

클러스터의 브로커 관리

브로커 클러스터를 설정하고 나면 새 브로커를 추가하거나, 이미 클러스터에 속해 있는 브로커를 다시 시작하거나, 클러스터에서 브로커를 제거해야 할 수 있습니다.

클러스터에 브로커 추가

▶ 기존 클러스터에 새 브로커를 추가하는 방법

- 클러스터 구성 파일을 사용하는 경우
 - a. 클러스터 구성 파일의 `imq.cluster.brokerlist` 등록 정보에 새 브로커를 추가합니다.
 - b. 클러스터의 모든 브로커에 대해 다음 명령을 실행합니다.

```
imqcmd reload cls
```

그러면 모든 브로커에서 `imq.cluster.brokerlist` 등록 정보를 다시 로드하고 클러스터에 있는 브로커의 모든 지속성 정보를 최신 상태로 유지할 수 있습니다.
 - c. 새 브로커를 시작할 때 명령줄에서 `-D` 옵션을 사용하여 `imq.cluster.url` 등록 정보를 지정합니다.

그러면 브로커가 클러스터 구성 파일을 가리킵니다.
- 클러스터 구성 파일을 사용하지 않는 경우 새 브로커를 시작할 때 명령줄에서 `-D` 옵션을 사용하여 `imq.cluster.brokerlist`, `imq.cluster.transport` (보안 클러스터 연결 서비스를 사용하는 경우) 및 `imq.cluster.masterbroker` (필요한 경우) 등록 정보를 지정합니다.

클러스터에서 브로커 다시 시작

클러스터의 브로커가 어떤 이유로 충돌하거나 종료된 경우 브로커를 해당 클러스터의 구성원으로 다시 시작해야 합니다.

▶ 이미 기존 클러스터의 구성원인 브로커를 다시 시작하는 방법

- 클러스터 구성 파일을 사용하여 클러스터를 정의하지 않은 경우에는 브로커를 다시 시작할 때 명령줄에서 `-D` 옵션을 사용하여 `imq.cluster.brokerlist` (그리고 필요한 경우 `imq.cluster.masterbroker`) 등록 정보를 지정합니다. 클러스터에 마스터 브로커가 포함되어 있지 않은 경우에는 `-cluster` 옵션을 사용하면 브로커를 다시 시작할 때 클러스터에 있는 브로커 목록을 지정할 수 있습니다.
- 클러스터 구성 파일을 사용해서 클러스터를 정의하는 경우에는 명령줄에서 브로커를 시작할 때 사용되는 `-D` 옵션을 사용하여 `imq.cluster.url` 등록 정보를 지정합니다.

클러스터에서 브로커 제거

▶ 기존 클러스터에서 브로커를 제거하는 방법

- 다음 명령줄을 사용해서 브로커 A, B, C를 모두 시작한 경우, A를 다시 시작하는 것만으로는 클러스터에서 제거할 수 없습니다.

```
imgbrokerd -cluster A,B,C
```

대신 다음 명령줄을 사용하여 다른 모든 브로커를 다시 시작해야 합니다.

```
imgbrokerd -cluster B,C
```

그 후에 `-cluster` 옵션을 지정하지 않고 브로커 A를 시작해야 합니다.

- 클러스터 구성 파일을 사용해서 브로커 목록을 지정한 경우에는 다음을 수행해야 합니다.
 - a. 구성 파일에서 브로커에 관한 내용을 제거합니다.
 - b. 제거할 브로커의 `img.cluster.url` 등록 정보를 변경 또는 제거하여 더 이상 일반 등록 정보를 사용하지 않게 합니다.
 - c. `imgcmd reload cls` 명령을 사용해서 모든 브로커가 클러스터 구성을 다시 로드하고 클러스터를 재구성하게 합니다.

마스터 브로커의 구성 변경 기록 관리

각 클러스터는 클러스터에서 지속성 상태의 모든 변경 사항을 추적하는 마스터 브로커 하나를 가질 수 있습니다. 이러한 상태에는 영구 가입 및 관리자가 만든 물리적 대상에 대한 정보가 포함됩니다. 모든 브로커는 시작할 때 마스터 브로커를 참조(즉, 마스터 브로커의 구성 변경 기록 참조)하여 해당 지속성 객체에 대한 정보를 동기화합니다. 따라서, 마스터 브로커에 오류가 발생하면 이러한 동기화가 불가능해 집니다. 즉, 마스터 브로커에 오류가 발생하면 물리적 대상 또는 영구 가입을 만들거나 삭제할 수 없습니다.

마스터 구성 변경 기록은 중요한 정보를 포함하고 있기 때문에 정기적으로 백업하여 오류가 발생할 경우에 복원하는 것이 중요합니다.

다음 절에서는 구성 변경 기록을 백업하고 복원하는 방법을 설명합니다.

구성 변경 기록 백업

▶ 구성 변경 기록을 백업하는 방법

imqbrokerd 명령에 `-backup` 옵션을 사용합니다. 예를 들면 다음과 같습니다.

```
imqbrokerd -backup mybackuplog
```

이 작업은 적절한 시기에 수행해야 합니다. 너무 오래된 백업 파일을 복원하면 백업을 마지막으로 수행한 후에 작성된 모든 물리적 대상 또는 영구 가입에 대한 변경 정보가 손실됩니다.

구성 변경 기록 복원

▶ 문제가 발생한 경우 마스터 브로커를 복원하는 방법

1. 클러스터에 있는 모든 브로커를 종료합니다.
2. 다음 명령을 사용해서 마스터 브로커의 구성 변경 기록을 복원합니다.

```
imqbrokerd -restore mybackuplog
```

3. 마스터 브로커에 새 이름 또는 포트 번호를 지정하는 경우에는 클러스터 구성 파일을 업데이트하여 마스터 브로커가 클러스터에 속하도록 하고 새 이름을 지정해야 합니다 (imq.cluster.masterbroker 등록 정보 사용).
4. 모든 브로커를 다시 시작합니다.

브로커를 복원하면 브로커의 구성 변경 레코드에 오래된 데이터가 다시 로드되는 것을 피할 수 없습니다. 하지만 앞 절의 설명과 같이 정기적으로 백업을 하면 이 문제를 최소화할 수 있습니다.

마스터 브로커에서는 지속성 객체의 전체 변경 사항을 추적하기 때문에 시간이 지나면 데이터베이스의 크기가 상당히 커질 수 있습니다. 백업 및 복원 작업을 수행하면 이러한 데이터베이스를 압축하고 최적화할 수 있는 이점이 있습니다.

로깅

이 절에서는 브로커의 기본 로깅 정보를 설명하며 구성을 변경하여 로그 정보를 대체 출력 채널로 리디렉션하고 로그 파일 롤오버 조건을 변경하는 방법을 설명합니다. 로깅에 대한 소개는 [71페이지](#)의 "로거"를 참조하십시오. 로깅을 사용하여 브로커 메트릭을 보고 하는 방법은 [246페이지](#)의 "모니터링 도구"를 참조하십시오.

기본 로깅 구성

브로커 시작 시 로그 파일이 연결된 브로커 인스턴스의 이름(*instanceName*)으로 식별되는 디렉토리에 있는 로그 파일 집합에 로그 출력을 저장하도록 자동으로 구성됩니다([부록 A, "Message Queue 데이터의 위치"](#) 참조).

```
.../instances/instanceName/log/
```

로그 파일은 단순 텍스트 파일입니다. 이름은 다음과 같으며 이 순서대로 지정됩니다.

```
log.txt
log_1.txt
log_2.txt
...
log_9.txt
```

기본적으로 로그 파일은 한 주에 한 번씩 롤오버되며, 시스템에서는 아홉 개의 백업 파일을 보존합니다.

- 로그 파일을 보존하는 디렉토리를 변경하려면 `img.log.file.dirpath` 등록 정보를 원하는 경로로 설정합니다.
- 로그 파일의 기본 이름을 `log`가 아닌 다른 이름으로 변경하려면 `img.log.file.filename` 등록 정보를 설정합니다.

브로커는 `ERROR`, `WARNING`, `INFO` 등 세 가지 로그 범주를 지원합니다([72페이지](#)의 [표 2-7](#) 참조). 로깅 수준을 설정하면 해당 수준 이상의 메시지를 수집합니다. 기본 로그 수준은 `INFO`입니다. 즉, `ERROR`, `WARNING` 및 `INFO` 메시지가 기본적으로 기록됩니다.

로그 메시지 형식

기록된 메시지는 타임스탬프(타임스탬프 표준 시간대를 변경하려면 [74페이지의 표 2-9](#) 참조), 메시지 코드 및 메시지 자체로 구성됩니다. 정보의 양은 설정한 로그 수준에 따라 달라집니다. 다음은 INFO 메시지의 예입니다.

```
[13/Sep/2000:16:13:36 PDT] B1004 Starting the broker service using tcp [
25374,100] with min threads 50 and max threads of 500
```

로거 구성 변경

모든 로거 등록 정보에 대한 설명은 [74페이지의 표 2-9](#)에 나와 있습니다.

▶ 브로커의 로거 구성을 변경하는 방법

1. 로그 수준을 설정합니다.
2. 로깅 범주 하나 이상에 해당하는 출력 채널(파일, 콘솔, 또는 둘 다)을 설정합니다.
3. 출력을 파일에 기록하는 경우에는 파일의 롤오버 기준을 구성합니다.

이 단계들은 로거 등록 정보를 설정하여 완료합니다. 이 작업은 두 방법 중 한 가지를 사용하여 수행할 수 있습니다.

- 브로커를 시작하기 전에 브로커의 `config.properties` 파일에 있는 로거 등록 정보를 변경 또는 추가합니다.
- 브로커를 시작하는 `imqbrokerd` 명령에서 로거 명령줄 옵션을 지정합니다. 브로커 옵션 `-D`를 사용해서 로거 등록 정보(또는 모든 브로커 등록 정보)를 변경할 수도 있습니다.

명령줄에 전달되는 옵션은 브로커 인스턴스 구성 파일에서 지정한 등록 정보를 대체합니다. [표 5-4](#)에는 로깅에 영향을 주는 `imqbrokerd` 옵션이 나와 있습니다.

표 5-4 `imqbrokerd` 로거 옵션 및 해당 등록 정보

imqbrokerd 옵션	설명
<code>-metrics interval</code>	메트릭 정보가 로거에 기록되는 간격(초)을 지정합니다.
<code>-loglevel level</code>	로그 수준을 ERROR, WARNING, INFO 중 하나로 설정합니다.

표 5-4 imqbrokerd 로거 옵션 및 해당 등록 정보(계속)

imqbrokerd 옵션	설명
-silent	콘솔에 대한 로깅을 끕니다.
-tty	모든 메시지를 콘솔로 보냅니다. 기본적으로 WARNING 및 ERROR 수준 메시지만 표시됩니다.

다음 절에서는 기본 구성을 변경하여 다음을 수행하는 방법을 설명합니다.

- 출력 채널(로그 메시지 대상) 변경
- 롤오버 기준 변경

출력 채널 변경

기본적으로 오류 및 경고 메시지는 로그 파일에 기록될 뿐 아니라 터미널에도 표시됩니다. (Solaris의 경우에는 오류 메시지가 시스템의 syslog 데몬에도 기록됩니다.)

로그 메시지의 출력 채널은 다음과 같은 방법으로 변경할 수 있습니다.

- 모든 로그 범주(주어진 수준에서)의 출력이 화면에 표시되게 하려면 imqbrokerd 명령에 -tty 옵션을 사용합니다.
- 로그 출력이 화면에 표시되지 않게 하려면 imqbrokerd 명령에 -silent 옵션을 사용합니다.
- 로그 파일에 기록할 로깅 정보의 범주를 지정하려면 imq.log.file.output 등록 정보를 사용합니다. 예를 들면 다음과 같습니다.

```
imq.log.file.output=ERROR
```

- 콘솔에 기록할 로깅 정보의 범주를 지정하려면 imq.log.console.output 등록 정보를 사용합니다. 예를 들면 다음과 같습니다.

```
imq.log.console.output=INFO
```

- Solaris의 경우 Solaris syslog에 기록할 로깅 정보의 범주를 지정하려면 imq.log.syslog.output 등록 정보를 사용합니다. 예를 들면 다음과 같습니다.

```
imq.log.syslog.output=NONE
```

주 로거 출력 채널을 변경하기 전에 출력 채널에 매핑할 정보를 지원하는 수준으로 로깅을 설정해야 합니다. 예를 들어, 로그 수준을 ERROR로 설정하고 `imq.log.console.output` 등록 정보를 WARNING으로 설정한 경우에는 WARNING 메시지의 로깅을 활성화하지 않았기 때문에 메시지가 기록되지 않습니다.

로그 파일 롤오버 기준 변경

로그 파일의 롤오버 기준에는 시간과 크기의 두 가지가 있습니다. 기본값은 시간 기준을 사용하고 7일마다 파일을 롤오버하는 것입니다.

- 시간 간격을 변경하려면 `imq.log.file.rolloversecs` 등록 정보를 변경해야 합니다. 예를 들어, 다음과 같은 등록 정보 정의를 사용하면 시간 간격을 10일로 변경할 수 있습니다.

```
imq.log.file.rolloversecs=864000
```

- 롤오버 기준을 파일 크기로 변경하려면 `imq.log.file.rolloverbytes` 등록 정보를 설정해야 합니다. 예를 들어, 다음 정의는 500,000바이트 제한에 도달하면 파일을 롤오버하도록 브로커를 설정합니다.

```
imq.log.file.rolloverbytes=500000
```

시간 및 크기 관련 롤오버 등록 정보를 모두 설정한 경우에는 먼저 도달한 제한에 의해 롤오버가 발생합니다. 앞에서 설명했듯이 브로커는 아홉 개까지의 롤오버 파일을 보존합니다.

브로커 및 응용 프로그램 관리

이 장에서는 브로커 및 제공 서비스 관리에 관련된 작업을 수행하는 방법을 설명합니다. 작업 중 일부는 특정 클라이언트 응용 프로그램에 대해 독립적이며 다음이 포함됩니다.

- 브로커의 상태 제어: 브로커를 일시 중지하고, 다시 시작하고, 종료하고, 처음부터 다시 시작할 수 있습니다.
- 브로커 등록 정보 쿼리 및 업데이트
- 연결 서비스 관리

기타 브로커 작업은 특정 응용 프로그램을 대신하여 수행합니다. 여기에는 물리적 대상 관리, 영구 가입, 트랜잭션이 포함됩니다.

- **Message Queue** 메시지는 브로커 대상을 통해 수신기 또는 가입자에게 라우팅됩니다. 브로커에서 이러한 대상을 만드는 것은 관리자의 책임입니다.
- **Message Queue**는 영구 가입이 있는 클라이언트가 비활성화되어 있는 동안에도 영구 가입자의 자원을 할당 및 관리합니다. 영구 가입에 대한 정보를 얻고 영구 가입을 완전 삭제하거나 메시지를 제거하여 **Message Queue** 자원을 확보하려면 **Message Queue** 명령 도구를 사용합니다.
- **Message Queue** 트랜잭션과 분산 트랜잭션은 브로커에 의해 추적됩니다. 오류가 발생하면 트랜잭션을 수동으로 완결 또는 롤백해야 할 수도 있습니다.

이 장에서는 명령 유틸리티(imqcmd)를 사용해서 이 모든 작업을 수행하는 방법을 설명합니다. **Message Queue** 메시지 서버의 그래픽 인터페이스인 관리 콘솔을 사용하면 이와 같은 여러 작업을 수행할 수 있습니다. 자세한 내용은 4장, "관리 콘솔 자습서"를 참조하십시오.

명령 유틸리티

명령 유틸리티를 사용하면 브로커와 브로커가 제공하는 서비스를 관리할 수 있습니다. 이 절에서는 기본 `imqcmd` 명령 구문을 설명하고, 하위 명령 목록을 제공하고, `imqcmd` 옵션을 요약합니다. 다음 절에서는 이런 명령을 사용하여 특정 작업을 수행하는 방법에 대해 설명합니다.

imqcmd 명령 구문

`imqcmd` 명령의 일반 구문은 다음과 같습니다.

```
imqcmd subcommand argument [options]
imqcmd -h|H
imqcmd -v
```

`-v`, `-h` 또는 `-H` 옵션을 지정하는 경우 명령줄에 지정된 하위 명령이 실행되지 않습니다. 예를 들어, 다음 명령을 입력하면 버전 정보는 표시되지만 `restart` 하위 명령은 실행되지 않습니다.

```
imqcmd restart bkr -v
```

imqcmd 하위 명령

명령 유틸리티(`imqcmd`)는 표 6-1에 나열된 하위 명령을 포함합니다. 하위 명령은 이 장의 작업 관련 절에 자세히 설명되어 있습니다.

표 6-1 `imqcmd` 하위 명령

하위 명령 및 인수	설명
<code>commit txn</code>	트랜잭션을 완결합니다.
<code>compact dst</code>	기본 제공 파일 기반 데이터 저장소에서 하나 이상의 대상을 압축합니다.
<code>create dst</code>	대상을 만듭니다.
<code>destroy dst</code>	대상을 완전 삭제합니다.
<code>destroy dur</code>	영구 가입을 완전 삭제합니다.
<code>list cxn</code>	브로커에 대한 연결을 나열합니다.
<code>list dst</code>	브로커의 대상을 나열합니다.
<code>list dur</code>	주제의 영구 가입을 나열합니다.

표 6-1 imqcmd 하위 명령(계속)

하위 명령 및 인수	설명
list svc	브로커의 서비스를 나열합니다.
list txn	브로커의 트랜잭션을 나열합니다.
metrics bkr	브로커 메트릭을 표시합니다.
metrics dst	대상 메트릭을 표시합니다.
metrics svc	서비스 메트릭을 표시합니다.
pause bkr	브로커의 모든 서비스를 일시 중지합니다.
pause dst	브로커에서 하나 이상의 대상을 일시 중지합니다.
pause svc	브로커에서 단일 서비스를 일시 중지합니다.
purge dst	대상을 완전 삭제하지 않고 대상에 있는 모든 메시지를 제거합니다.
purge dur	영구 가입을 완전 삭제하지 않고 영구 가입의 모든 메시지를 제거합니다.
query bkr	브로커의 정보를 쿼리 및 표시합니다.
query cxn	연결의 정보를 쿼리 및 표시합니다.
query dst	대상의 정보를 쿼리 및 표시합니다.
query svc	서비스의 정보를 쿼리 및 표시합니다.
query txn	트랜잭션의 정보를 쿼리 및 표시합니다.
reload cls	브로커 클러스터 구성을 다시 로드합니다.
restart bkr	현재 실행 중인 브로커 인스턴스를 다시 시작합니다. 새 브로커 인스턴스를 시작하는 데 사용할 수는 없습니다.
resume bkr	브로커의 모든 서비스를 다시 시작합니다.
resume dst	브로커에서 일시 중지된 하나 이상의 대상을 다시 시작합니다.
resume svc	한 서비스를 다시 시작합니다.
rollback txn	트랜잭션을 롤백합니다.
shutdown bkr	브로커 인스턴스를 종료합니다. imqbrokerd 명령을 사용해서 나중에 시작할 수 있지만, imqcmd의 restart bkr 하위 명령으로 시작할 수는 없습니다.
update bkr	브로커의 속성을 업데이트합니다.
update dst	대상의 속성을 업데이트합니다.
update svc	서비스의 속성을 업데이트합니다.

imqcmd 옵션 요약

표 6-2에 imqcmd 명령의 옵션이 나열되어 있습니다. 사용 설명은 다음의 작업 기반 절을 참조하십시오.

표 6-2 imqcmd 옵션

옵션	설명
-b <i>hostName:port</i>	브로커의 호스트 이름과 해당 포트 번호를 지정합니다. 기본값은 localhost:7676입니다. 포트만 지정하려면: -b :7878 이름만 지정하려면: -b somehost
-c <i>clientID</i>	주제에 영구 가입의 아이디를 지정합니다. 179페이지의 "영구 가입 관리"를 참조하십시오.
-d <i>destinationName</i>	주제의 이름을 지정합니다. list dur 및 destroy dur 하위 명령에 사용됩니다. 179페이지의 "영구 가입 관리"를 참조하십시오.
-f	사용자의 확인 없이 작업을 수행합니다.
-h	사용 도움말을 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
-H	사용 도움말, 속성 목록 및 예를 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
-int <i>interval</i>	metrics bkr, metrics dst 및 metrics svc 하위 명령이 메트릭 출력을 표시하는 간격(초)을 지정합니다.
-javahome <i>path</i>	Java 2와 호환할 수 있는 대체 런타임을 지정하여 사용합니다(기본값은 시스템의 런타임 또는 Message Queue와 함께 제공되는 런타임 사용).
-m <i>metricType</i>	표시할 메트릭 정보의 유형을 지정합니다. 이 옵션은 metrics dst, metrics svc 또는 metrics bkr 하위 명령에 사용됩니다. <i>metricType</i> 값은 메트릭이 대상, 서비스 또는 브로커 중 어느 것에 대해 생성되었는지에 따라 다릅니다.
-msp <i>numSamples</i>	metrics bkr, metrics dst 및 metrics svc 하위 명령이 메트릭 출력에 표시하는 메트릭 샘플 수를 지정합니다.
-n <i>argumentName</i>	하위 명령 인수의 이름을 지정합니다. 하위 명령에 따라 서비스나 물리적 대상, 영구 가입의 이름일 수도 있고 연결 아이디나 트랜잭션 아이디일 수도 있습니다.

표 6-2 imqcmd 옵션(계속)

옵션	설명
-o <i>attribute=value</i>	속성의 값을 지정합니다. 하위 명령 인수에 따라 브로커(157페이지의 "브로커 관리" 참조), 서비스(162페이지의 "연결 서비스 관리" 참조) 또는 대상(168페이지의 "대상 관리" 참조)의 속성이 될 수 있습니다.
-p <i>password</i>	자신(관리자)의 비밀번호를 지정합니다. 이 값을 생략하면 값을 묻는 메시지가 표시됩니다.
-pst <i>pauseType</i>	대상을 일시 중지할 때 생성자나 사용자 또는 둘 다를 일시 중지할지 여부를 지정합니다. 168페이지의 "대상 관리"를 참조하십시오.
-rtm <i>timeout</i>	imqcmd 하위 명령의 초기(다시 시도) 시간 초과 기간(초)을 지정합니다. 시간 초과는 imqcmd 하위 명령이 브로커에 요청한 후 기다리는 시간입니다. 이후 하위 명령이 다시 시도할 때마다 초기 시간 초과 기간의 배수인 시간 초과 값을 사용합니다. 기본값: 10
-rtr <i>numRetries</i>	imqcmd 하위 명령이 처음으로 시간 초과된 후에 다시 시도하는 횟수를 지정합니다. 기본값: 5
-s	비대화형 모드입니다. 출력이 표시되지 않습니다.
-secure	ssladmin 연결 서비스를 사용해서 브로커에 대한 보안 관리 연결을 지정합니다(223페이지의 "4단계. SSL기반 클라이언트 구성 및 실행" 참조).
-svn <i>serviceName</i>	연결을 나열할 서비스를 지정합니다. 167페이지의 "연결 정보 얻기"를 참조하십시오.
-t <i>destType</i>	대상의 유형을 지정합니다. t (주제) 또는 q (대기열). 168페이지의 "대상 관리"를 참조하십시오.
-tmp	임시 대상을 표시합니다. 168페이지의 표 6-9를 참조하십시오.
-u <i>userName</i>	자신(관리자)의 이름을 지정합니다. 이 값을 생략하면 값을 묻는 메시지가 표시됩니다.
-v	버전 정보를 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.

imqcmd 하위 명령을 실행할 때마다 호스트 이름 및 포트 번호(-b), 사용자 아이디(-u), 비밀번호(-p) 및 보안 연결(-secure) 옵션을 지정해야 합니다. 호스트 이름과 포트 번호를 지정하지 않으면 기본값이 사용됩니다. 사용자 아이디 및 비밀번호 정보를 지정하지 않으면 입력 프롬프트가 표시됩니다. -secure를 지정하지 않으면 연결에 보안이 적용되지 않습니다.

주 -secure 옵션을 사용하려면 먼저 [219페이지](#)의 "TCP/IP에서 SSL 기반 서비스 설정"에 설명된 것처럼 대상 브로커 인스턴스에서 ssladmin 서비스를 설정한 다음 활성화해야 합니다.

imqcmd 명령 사용

imqcmd 명령을 사용해서 브로커를 관리하려면 다음을 수행해야 합니다.

- imqbrokerd 명령을 사용해서 브로커를 시작합니다.
 - [134페이지](#)의 "브로커 시작"을 참조하십시오. 명령 유틸리티는 이미 실행 중인 브로커의 관리에만 사용할 수 있으며 이 유틸리티를 사용해서 브로커를 시작할 수는 없습니다.
- 브로커가 로컬 호스트의 포트 7676에서 실행 중인 경우가 아니면 -b 옵션을 사용해서 대상 브로커를 지정합니다.
- 적절한 관리자 사용자 아이디와 비밀번호를 지정합니다. 이를 지정하지 않으면 입력 프롬프트가 표시됩니다. 어떤 방법을 사용해도, imqcmd를 사용해서 수행하는 모든 작업은 사용자 저장소에 대해 인증됩니다. 자세한 내용은 [202페이지](#)의 "사용자 인증"을 참조하십시오.

Message Queue를 설치하면 기본 플랫폼 파일 사용자 저장소가 설치됩니다. 저장소에는 admin 사용자와 guest 사용자 각각에 대해 하나씩 두 개의 항목이 있습니다. 이러한 항목을 통해 추가 작업을 하지 않고도 브로커 인스턴스에 연결할 수 있습니다. 예를 들어, Message Queue를 테스트만 하는 경우에는 기본 사용자 아이디와 비밀번호(admin/admin)를 사용하여 imqcmd 유틸리티를 실행할 수 있습니다.

작업 시스템을 설정하는 경우에는 추가 작업을 통해 관리 사용자를 인증하고 권한을 부여해야 합니다([8장](#), "보안 관리" 참조). 특히, Message Queue 사용자 저장소에 항목을 만들어야 합니다([202페이지](#)의 "플랫폼 파일 사용자 저장소 사용" 참조). 사용자 저장소에 LDAP 디렉토리 서버를 사용할 수도 있습니다([209페이지](#)의 "사용자 저장소에 LDAP 서버 사용" 참조).

imqcmd 사용 예

아래에서는 imqcmd 명령 사용의 예를 보여 줍니다.

- 다음 명령은 localhost의 포트 7676에서 실행 중인 브로커의 등록 정보를 나열합니다.


```
imqcmd query bkr -u admin -p admin
```
- 다음 명령은 myserver의 포트 1564에서 실행 중인 브로커의 등록 정보를 나열합니다. 사용자 아이디는 alladin, 사용자 비밀번호는 abracadabra입니다.


```
imqcmd query bkr -b myserver:1564 -u alladin -p abracadabra
```

사용자 아이디 alladin이 admin 그룹에 할당되어 있으면 지정한 브로커에 admin 클라이언트로 연결할 수 있습니다.
- 다음 명령은 localhost의 포트 7676에서 실행 중인 브로커의 등록 정보를 나열합니다. 이 명령의 초기 시간 초과는 20초로, 시간 초과 이후의 재시도 횟수는 7로 설정합니다.


```
imqcmd query bkr -u admin -p admin -rtm 20 -rtr 7
```

브로커 관리

명령 유틸리티의 하위 명령을 사용하여 다음과 같은 브로커 관리 작업을 수행할 수 있습니다.

- [브로커 정보 표시](#)
- [브로커 등록 정보 업데이트](#)
- [브로커 메트릭 표시](#)
- [브로커 상태 제어](#)

브로커의 연결 서비스를 관리하려면 [162페이지](#)의 "[연결 서비스 관리](#)"를 참조하십시오. 브로커 대상을 관리하려면 [168페이지](#)의 "[대상 관리](#)"를 참조하십시오.

[표 6-3](#)에서는 브로커 관리에 사용되는 imqcmd 하위 명령을 나열합니다. 호스트 이름 또는 포트가 지정되어 있지 않으면 기본값(localhost:7676)인 것으로 가정합니다.

표 6-3 브로커 관리에 사용되는 `imgcmd` 하위 명령

하위 명령 구문	설명
<code>metrics bkr [-b <i>hostName:port</i>]</code> <code> [-m <i>metricType</i>]</code> <code> [-int <i>interval</i>]</code> <code> [-msp <i>numSamples</i>]</code>	<p>기본 브로커 또는 지정한 호스트 및 포트의 브로커 메트릭을 표시합니다.</p> <p><code>-m</code> 옵션을 사용하여 표시할 메트릭 유형을 지정합니다.</p> <p>tt1 브로커에 유입 및 유출되는 메시지와 패킷의 메트릭을 표시합니다(기본 메트릭 유형).</p> <p>rts 브로커에 유입 및 유출되는 메시지와 패킷의 메트릭을 초당 속도로 표시합니다.</p> <p>cxm 연결, 가상 메모리 힙 및 스레드를 표시합니다.</p> <p><code>-int</code> 옵션을 사용하여 메트릭 표시 간격(초)을 지정합니다. 기본값은 5초입니다.</p> <p><code>-msp</code> 옵션을 사용하여 출력에 표시되는 샘플 수를 지정합니다. 기본값은 무제한 수(무한)입니다.</p>
<code>pause bkr [-b <i>hostName:port</i>]</code>	<p>기본 브로커 또는 지정한 호스트 및 포트의 브로커를 일시 중지합니다. 161페이지의 "브로커 일시 중지 및 다시 시작"을 참조하십시오.</p>
<code>query bkr -b <i>hostName:port</i></code>	<p>기본 브로커 또는 지정한 호스트 및 포트의 브로커에 해당하는 현재 등록 정보 설정을 나열합니다. 지정한 브로커에 연결된 실행 중인 브로커(브로커가 여러 개인 클러스터에서)의 목록도 표시합니다.</p>
<code>reload cls</code>	<p>브로커 클러스터에만 적용됩니다. 클러스터에 있는 모든 브로커에서 <code>img.cluster.brokerlist</code> 등록 정보를 다시 로드하고 클러스터 정보를 업데이트하도록 합니다. 자세한 내용은 144페이지의 "클러스터에 브로커 추가"를 참조하십시오.</p>
<code>restart bkr [-b <i>hostName:port</i>]</code>	<p>기본 브로커 또는 지정한 호스트 및 포트의 브로커를 종료한 후 처음부터 다시 시작합니다.</p> <p>이 명령은 브로커를 처음 시작했을 때 지정한 옵션을 사용해서 브로커를 다시 시작합니다. 다른 옵션을 적용하려면 브로커를 종료한 후 원하는 옵션을 지정하여 다시 시작해야 합니다.</p>
<code>resume bkr [-b <i>hostName:port</i>]</code>	<p>기본 브로커 또는 지정한 호스트 및 포트의 브로커를 다시 시작합니다.</p>
<code>shutdown bkr [-b <i>hostName:port</i>]</code>	<p>기본 브로커 또는 지정한 호스트 및 포트의 브로커를 종료합니다.</p>

표 6-3 브로커 관리에 사용되는 imqcmd 하위 명령(계속)

하위 명령 구문	설명
<pre>update bkr [-b hostName:port] -o attribute=value [-o attribute=value1]...</pre>	기본 브로커 또는 지정한 호스트 및 포트의 브로커에 해당하는 지정된 속성을 변경합니다.

localhost의 포트 7676에서 실행 중인 브로커가 대상이 아닌 경우에는 표 6-3에 나열된 하위 명령을 사용할 때 브로커 호스트 이름과 포트 번호를 지정해야 합니다.

브로커 정보 표시

단일 브로커에 대한 정보를 쿼리 및 표시하려면 query bkr 하위 명령을 사용합니다. 예를 들면 다음과 같습니다.

```
imqcmd query bkr -u admin -p admin
```

다음과 같은 출력이 표시됩니다.

Version	3.5 SP1
Instance Name	imqbroker
Primary Port	7676
Current Number of Messages in System	0
Current Total Message Bytes in System	0
Max Number of Messages in System	unlimited (-1)
Max Total Message Bytes in System	unlimited (-1)
Max Message Size	70m
Auto Create Queues	true
Auto Create Topics	true
Auto Created Queue Max Number of Active Consumers	1
Auto Created Queue Max Number of Backup Consumers	0
Cluster Broker List (active)	
Cluster Broker List (configured)	
Cluster Master Broker	
Cluster URL	
Log Level	INFO
Log Rollover Interval (seconds)	604800
Log Rollover Size (bytes)	unlimited (-1)

브로커 등록 정보 업데이트

update bkr 하위 명령을 사용하면 표 6-4에 나열된 브로커 등록 정보를 업데이트할 수 있습니다. 브로커에서 업데이트된 사항은 브로커의 인스턴스 구성 파일에 자동으로 기록됩니다.

표 6-4 imqcmd가 업데이트하는 브로커 등록 정보

등록 정보	참조
imq.autocreate.queue	79페이지의 표 2-10
imq.autocreate.topic	79페이지의 표 2-10
imq.autocreate.queue.maxNumActiveConsumers	79페이지의 표 2-10
imq.autocreate.queue.maxNumBackupConsumers	79페이지의 표 2-10
imq.cluster.url	140페이지의 표 5-3
imq.log.level	74페이지의 표 2-9
imq.log.file.rolloversecs	74페이지의 표 2-9
imq.log.file.rolloverbytes	74페이지의 표 2-9
imq.system.max_count	62페이지의 표 2-4
imq.system.max_size	62페이지의 표 2-4
imq.message.max_size	62페이지의 표 2-4
imq.portmapper.port	57페이지의 표 2-3

예를 들어, 다음 명령은 대기열 대상의 자동 작성을 해제합니다.

```
imqcmd update bkr -o "imq.autocreate.queue=false"
-u admin -p admin
```

브로커 상태 제어

브로커를 시작한 후에는 다음 imqcmd 하위 명령을 사용하여 브로커의 상태를 제어할 수 있습니다.

브로커 일시 중지 및 다시 시작

- **브로커 일시 중지.** 브로커를 일시 중지하면 브로커의 연결 서비스 스레드가 지연되고 브로커에서는 연결 포트의 수신을 중지합니다. 따라서 브로커가 더 이상 새로운 연결을 받아들이거나, 메시지를 수신하거나, 메시지를 디스패치할 수 없습니다.

그러나 브로커를 일시 중지하더라도 관리 연결 서비스는 지연되지 않으므로 브로커에 대한 메시지 흐름을 규제하는 데 필요한 관리 작업은 수행할 수 있습니다. 예를 들어, 특정 대상으로 메시지가 폭주하는 경우에는 브로커를 일시 중지한 후 메시지 소스 추적, 대상 크기 제한, 대상 완전 삭제 등의 작업 중 하나를 수행하여 문제 해결을 도울 수 있습니다.

또한, 브로커를 일시 중지해도 클러스터 연결 서비스는 지연되지 않습니다. 그러나 클러스터 내의 메시지 전달은 클러스터의 여러 브로커에서 수행하는 전달 기능에 따라 다릅니다.

다음 명령은 myhost의 포트 1588에서 실행 중인 브로커를 일시 중지합니다.

```
imqcmd pause bkr -b myhost:1588 -u admin -p admin
```

또한 개별 연결 서비스를 일시 중지할 수도 있고(166페이지의 "연결 서비스 일시 중지 및 다시 시작" 참조) 개별 대상을 일시 중지할 수도 있습니다(175페이지의 "대상 일시 중지 및 다시 시작" 참조).

- **브로커 다시 시작.** 브로커를 다시 시작하면 브로커의 서비스 스레드가 다시 활성화되고 브로커가 다시 포트 수신을 시작합니다. 다음 명령은 localhost의 포트 7676에서 실행 중인 브로커를 다시 시작합니다.

```
imqcmd resume bkr -u admin -p admin
```

브로커 종료 및 다시 시작

- **브로커 종료.** 브로커를 종료하면 브로커 프로세스가 종료됩니다. 이것은 유예 기간이 있는 종료 방법으로, 브로커는 새 연결과 메시지의 수신을 멈추고, 기존 메시지의 전달을 완료한 다음 브로커 프로세스를 종료합니다. 다음 명령은 ctrlsrv의 포트 1572에서 실행 중인 브로커를 종료합니다.

```
imqcmd shutdown bkr -b ctrlsrv:1572 -u admin -p admin
```

- **브로커를 처음부터 다시 시작.** 브로커를 종료한 후 처음부터 다시 시작합니다. 다음 명령은 localhost의 포트 7676에서 실행 중인 브로커를 처음부터 다시 시작합니다.

```
imqcmd restart bkr -u admin -p admin
```

브로커 메트릭 표시

브로커에 대한 메트릭 정보를 표시하려면 `metrics bkr` 하위 명령을 사용합니다. 예를 들어, 다음 명령은 브로커에 메시지가 유입 및 유출되는 속도를 10초 간격으로 표시합니다.

```
imqcmd metrics bkr -m rts -int 10 -u admin -p admin
```

다음과 같은 출력이 표시됩니다.

Msgs/sec		Msg Bytes/sec		Pkts/sec		Pkt Bytes/sec	
In	Out	In	Out	In	Out	In	Out
0	0	27	56	0	0	38	66
10	0	7365	56	10	10	7457	1132
0	0	27	56	0	0	38	73
0	10	27	7402	10	20	1400	8459
0	0	27	56	0	0	38	73

`imqcmd`를 사용하여 브로커 메트릭을 보고하는 방법에 대한 자세한 내용은 [246페이지](#)의 "모니터링 도구"를 참조하십시오.

연결 서비스 관리

명령 유틸리티의 하위 명령을 사용하여 다음과 같은 연결 서비스 관리 작업을 수행할 수 있습니다.

- [연결 서비스 나열](#)
- [연결 서비스 정보 표시](#)
- [연결 서비스 등록 정보 업데이트](#)
- [연결 서비스 메트릭 표시](#)
- [연결 서비스 일시 중지 및 다시 시작](#)

Message Queue 연결 서비스에 대한 개요는 [54페이지](#)의 "연결 서비스"를 참조하십시오.

[표 6-5](#)에 연결 서비스 관리에 사용되는 `imqcmd` 하위 명령이 나열되어 있습니다. 호스트 이름 또는 포트가 지정되어 있지 않으면 기본값(`localhost:7676`)인 것으로 가정합니다.

표 6-5 연결 서비스 관리에 사용되는 `imqcmd` 하위 명령

하위 명령 구문	설명
<code>list svc [-b <i>hostName:port</i>]</code>	기본 브로커 또는 지정한 호스트 및 포트의 브로커에 있는 모든 연결 서비스를 나열합니다.
<code>metrics svc -n <i>serviceName</i></code> <code>[-b <i>hostName:port</i>]</code> <code>[-m <i>metricType</i>]</code> <code>[-int <i>interval</i>]</code> <code>[-msp <i>numSamples</i>]</code>	기본 브로커 또는 지정한 호스트 및 포트의 브로커에서 지정된 서비스의 메트릭을 표시합니다. -m 옵션을 사용하여 표시할 메트릭 유형을 지정합니다. tt1 브로커에 유입 및 유출되는 메시지와 패킷의 메트릭을 지정된 서비스 방법으로 표시합니다(기본 메트릭 유형). rts 브로커에 메시지와 패킷이 유입 및 유출되는 초당 속도에 대한 메트릭을 지정된 서비스 방법으로 표시합니다. cxn 연결, 가상 메모리 힙 및 스레드를 표시합니다. -int 옵션을 사용하여 메트릭 표시 간격(초)을 지정합니다. 기본값은 5초입니다. -msp 옵션을 사용하여 출력에 표시되는 샘플 수를 지정합니다. 기본값은 무제한 수(무한)입니다.
<code>pause svc -n <i>serviceName</i></code> <code>[-b <i>hostName:port</i>]</code>	기본 브로커 또는 지정한 호스트 및 포트의 브로커에서 실행 중인, 지정된 서비스를 일시 중지합니다. 관리 서비스는 일시 중지할 수 없습니다.
<code>query svc -n <i>serviceName</i></code> <code>[-b <i>hostName:port</i>]</code>	기본 브로커 또는 지정한 호스트 및 포트의 서비스에서 실행 중인, 지정된 서비스에 대한 정보를 표시합니다.
<code>resume svc -n <i>serviceName</i></code> <code>[-b <i>hostName:port</i>]</code>	기본 브로커 또는 지정한 호스트 및 포트에서 실행 중인, 지정된 서비스를 다시 시작합니다.
<code>update svc -n <i>serviceName</i></code> <code>[-b <i>hostName:port</i>]</code> <code>-o <i>attribute=value</i></code> <code>[-o <i>attribute=value1</i>]...</code>	기본 브로커 또는 지정한 호스트 및 포트의 브로커에서 실행 중인, 지정된 서비스의 지정된 속성을 업데이트합니다. 서비스 속성에 대한 설명은 165페이지의 표 6-7 을 참조하십시오.

브로커는 응용 프로그램 클라이언트 및 관리 클라이언트 모두와의 연결을 지원합니다. 현재 Message Queue 브로커에서 사용할 수 있는 연결 서비스가 표 6-6에 표시되어 있습니다. 서비스 이름 옆에 있는 값은 `-n` 옵션의 서비스 이름을 지정할 때 사용되는 값입니다. 표에 있는 것과 같이 각 서비스는 NORMAL (응용 프로그램 클라이언트) 또는 ADMIN (관리 클라이언트) 중에서 사용하는 서비스 유형과 기본 전송 계층에 따라 지정됩니다.

표 6-6 브로커가 지원하는 연결 서비스

서비스 이름	서비스 유형	프로토콜 유형
jms	NORMAL	tcp
ssljms (엔터프라이즈판)	NORMAL	tls (SSL 기반 보안)
httpjms (엔터프라이즈판)	NORMAL	http
httpsjms (엔터프라이즈판)	NORMAL	https (SSL 기반 보안)
admin	ADMIN	tcp
ssladmin (엔터프라이즈판)	ADMIN	tls (SSL 기반 보안)

연결 서비스 나열

브로커에서 사용할 수 있는 연결 서비스를 나열하려면 다음과 같은 명령을 사용합니다.

```
imqcmd list svc [-b hostName:portNumber] -u admin -p admin
```

예를 들어, 다음 명령은 myServer 호스트의 포트 6565에서 실행 중인 브로커에서 사용할 수 있는 서비스를 나열합니다.

```
imqcmd list svc -b MyServer:6565 -u admin -p admin
```

다음 명령은 localhost의 포트 7676에서 실행 중인 브로커의 모든 서비스를 나열합니다.

```
imqcmd list svc -u admin -p admin
```

이 명령은 다음과 같은 정보를 출력합니다.

```

-----
Service Name      Port Number      Service State
-----
admin             41844 (dynamic)  RUNNING
httpjms          -                UNKNOWN
httpsjms         -                UNKNOWN
    
```

jms	41843 (dynamic)	RUNNING
ssladmin	dynamic	UNKNOWN
ssljms	dynamic	UNKNOWN

연결 서비스 정보 표시

단일 서비스에 대한 정보를 쿼리 및 표시하려면 `query` 하위 명령을 사용합니다. 예를 들면 다음과 같습니다.

```
imqcmd query svc -n jms -u admin -p admin
```

다음과 같은 출력이 표시됩니다.

Service Name	jms
Service State	RUNNING
Port Number	60920 (dynamic)
Current Number of Allocated Threads	0
Current Number of Connections	0
Min Number of Threads	10
Max Number of Threads	1000

연결 서비스 등록 정보 업데이트

`update` 하위 명령을 사용하여 표 6-7에 나열된 서비스 등록 정보 중 하나 이상을 변경할 수 있습니다.

표 6-7 imqcmd가 업데이트하는 연결 서비스 등록 정보

등록 정보	설명
port	업데이트할 서비스에 할당된 포트입니다(httpjms 또는 httpsjms에는 적용되지 않음). 값이 0인 경우 포트 매퍼가 포트를 동적으로 할당합니다.
minThreads	서비스에 할당된 최소 스레드 수입니다.
maxThreads	서비스에 할당된 최대 스레드 수입니다.

다음 명령은 `jms` 서비스에 할당된 최소 스레드 수를 20으로 변경합니다.

```
imqcmd update svc -n jms -o "minThreads=20"
```

연결 서비스 메트릭 표시

단일 서비스에 대한 메트릭 정보를 표시하려면 `metrics` 하위 명령을 사용합니다. 예를 들어, 다음 명령은 `jms` 연결 서비스에서 처리된 메시지 및 패킷의 누적 총 수를 구합니다.

```
imqcmd metrics svc -n jms -m ttl -u admin -p admin
```

다음과 같은 출력이 표시됩니다.

Msgs		Msg Bytes		Pkts		Pkt Bytes	
In	Out	In	Out	In	Out	In	Out

164	100	120704	73600	282	383	135967	102127
657	100	483552	73600	775	876	498815	149948

`imqcmd`를 사용하여 연결 서비스 메트릭을 보고하는 방법에 대한 자세한 내용은 [246페이지](#)의 "모니터링 도구"를 참조하십시오.

연결 서비스 일시 중지 및 다시 시작

관리 서비스(일시 중지 불가)를 제외한 다른 서비스를 일시 중지하려면 다음과 같은 명령을 사용합니다.

```
imqcmd pause svc -n serviceName -u admin -p admin
```

서비스를 일시 중지하면 다음과 같이 됩니다.

- 브로커는 일시 중지된 서비스에서 새 클라이언트 연결 수신을 멈춥니다. `Message Queue` 클라이언트가 새 연결을 열려고 하면 예외가 발생합니다.
- 일시 중지된 서비스의 기존 연결은 모두 그대로 유지되지만 브로커는 서비스가 다시 시작될 때까지 이러한 연결의 모든 메시지 처리를 지연합니다(예를 들어, 클라이언트가 메시지를 보내려고 하면 서비스가 다시 시작될 때까지 `send()` 메소드가 차단됨).

- 브로커가 이미 수신한 메시지의 메시지 전달 상태는 그대로 유지됩니다(예를 들어, 트랜잭션이 중단되지 않고 서비스가 다시 시작되면 메시지 전달이 재개됨).

서비스를 다시 시작하려면 다음과 같은 명령을 사용합니다.

```
imqcmd resume svc -n serviceName -u admin -p admin
```

연결 정보 얻기

명령 유틸리티의 하위 명령을 사용하여 연결 정보를 나열하고 가져올 수 있습니다.

표 6-8에 연결에 적용되는 imqcmd 하위 명령이 나열되어 있습니다. 호스트 이름 또는 포트가 지정되어 있지 않으면 localhost, 7676인 것으로 가정합니다.

표 6-8 연결 서비스 관리에 사용되는 imqcmd 하위 명령

하위 명령 구문	설명
<code>list cxn [-svn serviceName] [-b hostName:port]</code>	기본 브로커 또는 지정한 호스트 및 포트의 브로커에서 지정된 서비스 이름의 연결을 모두 나열합니다. 서비스 이름을 지정하지 않는 경우 모든 연결이 나열됩니다.
<code>query cxn -n connectionID [-b hostName:port]</code>	기본 브로커 또는 지정한 호스트 및 포트의 브로커에서 지정된 연결에 대한 정보를 표시합니다.

단일 연결 서비스에 대한 정보를 쿼리 및 표시하려면 `query` 하위 명령을 사용합니다. 예를 들면 다음과 같습니다.

```
imqcmd query cxn -n 421085509902214374 -u admin -p admin
```

다음과 같은 출력이 표시됩니다.

```

Connection ID      421085509902214374
User               guest
Service           jms
Producers         0
Consumers         1
Host              111.22.333.444
Port              60953
Client ID
Client Platform

```

대상 관리

모든 Message Queue 메시지는 특정 브로커에 만들어지는 대기열 및 주제 대상을 통해 해당 사용자 클라이언트로 라우팅됩니다.

명령 유틸리티의 하위 명령을 사용하여 다음과 같은 대상 관리 작업을 수행할 수 있습니다.

- 대상 만들기
- 대상 나열
- 대상 정보 표시
- 대상 속성 업데이트
- 대상 메트릭 표시
- 대상 일시 중지 및 다시 시작
- 대상 제거
- 대상 완전 삭제
- 대상 압축

대상에 대한 소개는 [76페이지의 "물리적 대상"](#)을 참조하십시오.

[표 6-9](#)에서는 imqcmd 대상 하위 명령을 요약합니다. 기본 브로커(localhost:7676)가 아닌 경우 브로커의 호스트 이름과 포트를 지정해야 합니다.

표 6-9 대상 관리에 사용되는 imqcmd 하위 명령

하위 명령 구문	설명
<code>compact dst [-t <i>destType</i> -n <i>destName</i>]</code>	기본 제공 파일 기반 데이터 저장소에서 지정된 유형과 이름의 대상을 압축합니다. 대상 유형과 이름을 지정하지 않으면 모든 대상이 압축됩니다. 대상을 압축하려면 먼저 일시 중지해야 합니다.
<code>create dst -t <i>destType</i> -n <i>destName</i> [-o <i>attribute=value</i>] [-o <i>attribute=value1</i>]...</code>	지정한 유형의 대상을 지정한 이름과 지정한 속성을 사용하여 만듭니다. 대상 이름은 영숫자(공백 없음)만 포함하고 영문자나 "_" 및 "\$" 문자로 시작할 수 있습니다. "mq" 문자열로는 시작할 수 없습니다.
<code>destroy dst -t <i>destType</i> -n <i>destName</i></code>	지정한 유형과 이름의 대상을 완전 삭제합니다.

표 6-9 대상 관리에 사용되는 imqcmd 하위 명령(계속)

하위 명령 구문	설명
list dst [-t <i>destType</i>] [-tmp]	지정된 유형의 대상을 모두 나열합니다. 임시 대상을 함께 나열할 수도 있습니다(81페이지의 "임시 대상" 참조). 유형 인수는 다음과 같은 두 가지 값을 가질 수 있습니다. <i>destType</i> = q (대기열) <i>destType</i> = t (주제) 유형을 지정하지 않으면 모든 유형의 대상이 모두 나열됩니다.
metrics dst -t <i>destType</i> -n <i>destName</i> [-m <i>metricType</i>] [-int <i>interval</i>] [-msp <i>numSamples</i>]	지정한 유형과 이름의 대상에 대한 메트릭 정보를 표시합니다. -m 옵션을 사용하여 표시할 메트릭 유형을 지정합니다. ttl 대상에 유입 및 유출되고 메모리에 있는 메시지와 패킷의 메트릭을 표시합니다(기본 메트릭 유형). rts 대상에 유입 및 유출되는 메시지와 패킷의 메트릭을 초당 속도로 나타내고 기타 속도 정보를 표시합니다. con 사용자 관련 메트릭을 표시합니다. dsk 디스크 사용 메트릭을 표시합니다. -int 옵션을 사용하여 메트릭 표시 간격(초)을 지정합니다. 기본값은 5초입니다. -msp 옵션을 사용하여 출력에 표시되는 샘플 수를 지정합니다. 기본값은 무제한 수(무한)입니다.
pause dst [-t <i>destType</i> -n <i>destName</i>] [-pst <i>pauseType</i>]	지정된 유형과 이름의 대상에서 사용자에게 메시지 전달(-pst CONSUMERS), 생성자로부터 메시지 전달(-pst PRODUCERS) 또는 두 가지 모두(-pst ALL)를 일시 중지합니다. 대상 유형과 이름을 지정하지 않으면 모든 대상이 일시 중지됩니다. 기본값은 ALL입니다.
purge dst -t <i>destType</i> -n <i>destName</i>	지정한 유형과 이름의 대상에서 메시지를 제거합니다.
query dst -t <i>destType</i> -n <i>destName</i>	지정한 유형과 이름의 대상에 대한 정보를 나열합니다.
resume dst [-t <i>destType</i> -n <i>destName</i>]	지정된 유형과 이름의 일시 중지된 대상에서 메시지 전달을 다시 시작합니다. 대상 유형과 이름을 지정하지 않으면 모든 대상이 다시 시작됩니다.

표 6-9 대상 관리에 사용되는 imqcmd 하위 명령(계속)

하위 명령 구문	설명
<pre>update dst -t <i>destType</i> -n <i>destName</i> -o <i>attribute=value</i> [-o <i>attribute=value1</i>]...</pre>	<p>지정한 대상에서 지정한 속성의 값을 업데이트합니다.</p> <p>속성 이름에는 표 6-10에 설명된 속성을 사용할 수 있습니다.</p>

대상 만들기

대상을 만들 때 다음을 지정해야 합니다.

- 대상 유형: 주제 또는 대기열
- 대상 이름: 영숫자(공백 없음)만 포함하고 영문자나 "_" 및 "\$" 문자로 시작할 수 있습니다. "mq" 문자열로는 시작할 수 없습니다.
- 대상 속성에서 기본값이 아닌 값

많은 대상 속성이 브로커 메모리 자원 및 메시지 흐름 관리에 사용됩니다. 예를 들어, 대상에 허용되는 최대 생성자 수 또는 최대 메시지 수(또는 크기)를 지정할 수 있습니다. 이러한 제한은 브로커 구성 등록 정보를 사용하여 브로커 전체에 설정할 수 있는 제한과 비슷합니다([61페이지](#)의 "메모리 자원 및 메시지 흐름 관리" 참조). 또한 이러한 제한에 도달할 때 브로커가 응답하는 방법도 지정할 수 있습니다.

대기열 대상에만 적용되는 대상 속성도 있습니다. 이러한 속성은 다중 사용자로의 로드 균형 조정 메시지 전달 시 활성화 및 백업 사용자 수를 지정하는 데 사용됩니다([77페이지](#)의 "대기열 대상" 참조).

[표 6-10](#)에서는 각 대상 유형에 적용되는 속성을 설명합니다. 대상을 만들거나 업데이트할 때 속성 값을 설정할 수 있습니다. 자동 작성 대상의 경우 브로커의 인스턴스 구성 파일에 기본 등록 정보 값을 설정합니다([127페이지](#)의 "구성 파일" 참조).

표 6-10 대상 속성

대상 유형	속성	기본값	설명
대기열 및 주제	maxNumMsgs ¹	-1 (제한 없음)	대상에 허용되는 사용되지 않은 최대 메시지 수를 지정합니다.
대기열 및 주제	maxTotalMsgBytes ¹	-1 (제한 없음)	대상에서 사용되지 않은 메시지가 사용할 수 있는 전체 메모리 양의 최대값(바이트)을 지정합니다.
대기열 및 주제	limitBehavior	REJECT_NEWEST	메모리 제한 임계값에 도달할 때 브로커가 응답하는 방법을 지정합니다. 값은 다음과 같습니다. FLOW_CONTROL - 생성자를 느리게 합니다. REMOVE_OLDEST - 가장 오래된 메시지를 삭제합니다. REMOVE_LOW_PRIORITY - 메시지의 보존 기간을 기준으로 우선 순위가 가장 낮은 메시지를 삭제합니다(생성자 클라이언트는 메시지 삭제에 대한 알림 메시지를 받지 않음). REJECT_NEWEST - 최신 메시지를 거부합니다(생성자 클라이언트는 지속성 메시지가 거부되는 경우 예외를 받지만 비지속성 메시지가 거부되는 경우 알림 메시지를 받지 않음).
대기열 및 주제	maxBytesPerMsg	-1 (제한 없음)	대상에 허용되는 단일 메시지의 최대 크기(바이트)를 지정합니다(생성자 클라이언트는 지속성 메시지가 거부되는 경우 예외를 받지만 비지속성 메시지가 거부되는 경우 알림 메시지를 받지 않음).
대기열 및 주제	maxNumProducers ¹	-1 (제한 없음)	대상에 허용되는 최대 생성자 수를 지정합니다. 이 제한에 도달하면 새로운 생성자가 만들어지지 않습니다.
대기열만 해당	maxNumActiveConsumers	1	대기열 대상으로부터의 로드 균형 조정 전달에서 활성 상태가 될 수 있는 최대 사용자 수를 지정합니다. 값 -1은 제한이 없음을 의미합니다(플랫폼판은 이 값을 2로 제한).
대기열만 해당	maxNumBackupConsumers	0	대기열 대상으로부터의 로드 균형 조정 전달 중에 오류가 발생할 경우 활성 사용자를 대신할 수 있는 백업 사용자의 최대 수를 지정합니다. 값 -1은 제한이 없음을 의미합니다.

1. 클러스터 환경에서 이 등록 정보는 클러스터의 모든 인스턴스에 한꺼번에 적용되는 것이 아니라 클러스터에 있는 대상의 각 인스턴스에 적용됩니다.

표 6-10 대상 속성(계속)

대상 유형	속성	기본값	설명
대기열 및 주제	consumerFlowLimit	주제: 1000 대기열: 1000	단일 일괄 처리로 사용자에게 전달되는 최대 메시지 수를 지정합니다. 로드 균형 조정 대기열 전달에서 이 값은 로드 균형 조정이 시작되기 전에 활성 사용자에게 라우팅된 대기 메시지의 초기 수입니다(77페이지의 "다중 사용자로의 대기열 전달" 참조). 이 제한은 해당 연결에서 대상 사용자에게 지정된 낮은 값으로 대체될 수 있습니다(Message Queue Java Client Developer's Guide의 연결 팩토리 속성 참조). 값 -1은 제한이 없음을 의미합니다.
대기열만 해당	localDeliveryPreferred	false	브로커 클러스터의 로드 균형 조정 대기열 전달에만 적용됩니다. 로컬 브로커에 사용자가 없는 경우에만 원격 사용자에게 메시지를 전달하도록 지정합니다. 대상을 로컬에만 전달로 제한해서는 안 됩니다(isLocalOnly = false).
대기열 및 주제	isLocalOnly	false	브로커 클러스터에만 적용됩니다. 대상이 다른 브로커에 복제되지 않도록 지정합니다. 따라서, 메시지가 로컬 사용자(대상을 만든 브로커에 연결된 사용자)에게만 전달되도록 지정합니다. 대상이 만들어진 이후에는 이 속성을 업데이트할 수 없습니다.

1. 클러스터 환경에서 이 등록 정보는 클러스터의 모든 인스턴스에 한꺼번에 적용되는 것이 아니라 클러스터에 있는 대상의 각 인스턴스에 적용됩니다.

- 대기열 대상을 만들려면 다음과 같이 명령을 입력합니다.

```
imqcmd create dst -n myQueue -t q -o "maxNumActiveConsumers=5"
```

대상 이름은 영숫자(공백 없음)만 포함하고 영문자 또는 "_" 및 "\$" 문자로 시작할 수 있습니다. 매트릭 주제 대상에 예약된 "mq" 문자열로는 시작할 수 없습니다(73페이지의 표 2-8 참조).

- 주제 대상을 만들려면 다음과 같이 명령을 입력합니다.

```
imqcmd create dst -n myTopic -t t -o "maxBytesPerMsg=5000"
```

대상 나열

대상의 현재 속성 값, 대상과 연관된 생성자 또는 사용자 수 및 메시징 메트릭(대상의 메시지 수 및 크기 등)에 대한 정보를 얻을 수 있습니다.

정보를 얻을 대상을 찾으려면 먼저 `list dst` 하위 명령을 사용하여 특정 브로커의 모든 대상을 나열할 수 있습니다. 예를 들어, `myHost`의 포트 `4545`에서 실행 중인 브로커의 모든 대상을 나열하려면 다음 명령을 입력합니다.

```
imqcmd list dst -b myHost:4545
```

`list dst` 하위 명령에서는 나열할 대상의 유형을 지정하거나 임시 대상을 포함하도록 지정할 수도 있습니다(`-tmp` 옵션 사용). 임시 대상은 일반적으로 다른 클라이언트에 보낸 메시지에 대한 응답을 수신하기 위해 클라이언트가 만듭니다([81페이지의 "임시 대상" 참조](#)).

대상 정보 표시

대상의 현재 속성 값에 대한 정보를 얻으려면 다음 명령에서처럼 `query dst` 하위 명령을 사용합니다.

```
imqcmd query dst -t q -n XQueue -u admin -p admin
```

다음과 같은 출력이 표시됩니다.

```
-----
Destination Name      Destination Type
-----
XQueue                Queue

On the broker specified by:

-----
Host                  Primary Port
-----
localhost             7676

Destination Name      XQueue
Destination Type      Queue
Destination State     RUNNING
Created Administratively true

Current Number of Messages      0
Current Total Message Bytes     0
Current Number of Producers     0
Current Number of Active Consumers 0
```

```

Current Number of Backup Consumers    0

Max Number of Messages               unlimited (-1)
Max Total Message Bytes              unlimited (-1)
Max Bytes per Message                unlimited (-1)
Max Number of Producers               100
Max Number of Active Consumers        1
Max Number of Backup Consumers        0

Limit Behavior                       REJECT_NEWEST
Consumer Flow Limit                  100
Is Local Destination                 false
Local Delivery is Preferred          false

```

출력에는 해당 대상에 연관되어 있는 생성자와 사용자 수도 표시됩니다. 대기열 대상의 경우 활성 사용자와 백업 사용자도 모두 포함됩니다.

update dst 하위 명령을 사용하여 하나 이상의 속성 값을 변경할 수 있습니다([174페이지](#)의 "대상 속성 업데이트" 참조).

대상 속성 업데이트

update dst 하위 명령에 업데이트할 속성을 지정하는 -o 옵션을 사용하여 대상의 속성을 변경할 수 있습니다. 업데이트할 속성이 두 개 이상인 경우에는 -o 옵션을 두 번 이상 사용할 수 있습니다. 예를 들어, 다음 명령은 maxBytesPerMsg 속성을 1000으로 변경하고 MaxNumMsgs 속성을 2000으로 변경합니다.

```

imqcmd update dst -t q -n myQueue -o "axBytesPerMsg=1000"
-o "maxNumMsgs=2000" -u admin -p admin

```

업데이트할 수 있는 속성의 목록을 보려면 [171페이지](#)의 표 6-10을 참조하십시오.

update dst 하위 명령을 사용해서 대상의 유형을 업데이트하거나 isLocalOnly 속성을 업데이트할 수 없습니다.

대상 메트릭 표시

대상에 대한 메시지 메트릭 정보를 얻으려면 다음 명령에서처럼 `metrics dst` 하위 명령을 사용합니다.

```
imqcmd metrics dst -t q -n XQueue -m ttl -u admin -p admin
```

다음과 같은 출력이 표시됩니다.

Msgs		Msg Bytes		Msg Count			Total Msg Bytes (k)			Largest
In	Out	In	Out	Current	Peak	Avg	Current	Peak	Avg	Msg (k)
200	200	147200	147200	0	200	0	0	143	71	0
300	200	220800	147200	100	200	10	71	143	64	0
300	300	220800	220800	0	200	0	0	143	59	0

`imqcmd`를 사용하여 대상 메트릭을 보고하는 방법에 대한 자세한 내용은 [246페이지의 "모니터링 도구"](#)를 참조하십시오.

대상 일시 중지 및 다시 시작

대상을 일시 중지하여 생성자에서 대상으로, 대상에서 사용자로 또는 둘 다의 메시지 전달을 제어할 수 있습니다. 특히, 대상으로의 메시지 흐름을 일시 중지하면 메시지 생성이 사용보다 훨씬 빠를 때 대상에서 메시지가 넘치는 것을 방지할 수 있습니다.

대상으로부터 또는 대상으로의 메시지 전달을 일시 중지하려면 다음 명령에서처럼 `pause dst` 하위 명령을 사용합니다.

```
imqcmd pause dst -n myQueue -t q -pst PRODUCERS -u admin -p admin
```

```
imqcmd pause dst -n myTopic -t t -pst CONSUMERS -u admin -p admin
```

대상을 일시 중지한 경우 전달을 다시 시작하려면 다음 명령을 입력합니다.

```
imqcmd resume dst -n myQueue -t q
```

다중 브로커 클러스터에서 대상 인스턴스는 클러스터의 각 브로커에 있습니다. 이러한 대상을 개별적으로 일시 중지해야 합니다.

대상 제거

현재 대상의 대기열에 들어 있는 모든 메시지를 제거할 수 있습니다. 대상을 제거하는 것은 물리적 대상의 대기열에 들어 있는 모든 메시지를 삭제하는 것을 의미합니다. 대상에 누적된 메시지가 시스템의 자원을 너무 많이 차지하는 경우에 메시지를 제거할 수 있습니다. 이런 상황은 대기열에 등록된 사용자 클라이언트가 없는 상태에서 너무 많은 메시지를 받으면 발생할 수 있습니다. 주제의 비활성 영구 가입자가 활성화되지 않는 경우에도 발생할 수 있습니다. 두 경우 모두 불필요한 메시지가 보존됩니다.

대상에서 메시지를 제거하려면 다음 명령에서처럼 `purge dst` 하위 명령을 사용합니다.

```
imqcmd purge dst -n myQueue -t q -u admin -p admin
```

```
imqcmd purge dst -n myTopic -t t -u admin -p admin
```

브로커를 종료했다가 다시 시작할 때 오래된 메시지가 전달되지 않게 하려면 `-reset messages` 옵션을 사용하여 오래된 메시지를 제거합니다. 예를 들면 다음과 같습니다.

```
imqbrokerd -reset messages -u admin -p admin
```

그러면 브로커를 다시 시작한 후에 대상을 제거할 필요가 없습니다.

다중 브로커 클러스터에서 대상 인스턴스는 클러스터의 각 브로커에 있습니다. 이러한 대상을 개별적으로 제거해야 합니다.

대상 완전 삭제

대상을 완전히 삭제하려면 다음 명령에서처럼 `destroy dst` 하위 명령을 사용합니다.

```
imqcmd destroy dst -t q -n myQueue -u admin -p admin
```

대상을 완전 삭제하면 대상에 있는 모든 메시지가 제거되고 대상이 브로커에서 제거됩니다. 이 작업은 다시 되돌릴 수 없습니다.

대상 압축

플러그인 JDBC 호환 데이터 저장소와 반대되는 기본 제공 파일 기반 데이터 저장소를 메시지의 영구 저장소로 사용하는 경우 디스크 사용률을 모니터링하고 필요한 경우 디스크를 압축할 수 있습니다.

파일 기반 메시지 저장소는 메시지가 보관될 대상에 따라 다른 디렉토리에 저장되도록 구성됩니다. 각 대상 디렉토리에서 대부분의 메시지는 가변 크기 레코드로 구성되는 단일 파일인 가변 크기 레코드 파일에 저장됩니다(단편화를 줄이기 위해 크기가 구성 가능한 임계값을 초과하는 메시지는 자체의 개별 파일에 저장). 가변 크기 레코드 파일에서 다양한 크기의 메시지가 지속되다가 제거될 때 파일에서 사용 가능한 레코드가 다시 사용되지 않는 공간이 생길 수 있습니다.

사용되지 않은 사용 가능한 레코드를 관리하려면 명령 유틸리티의 하위 명령을 사용하여 대상별 디스크 사용률을 모니터하고 사용률이 떨어지면 사용 가능한 디스크 공간을 재생 이용합니다.

대상 디스크 사용률 모니터링

대상 디스크 사용률을 모니터하려면 다음 `imqcmd` 하위 명령을 사용합니다.

```
imqcmd metrics dst -t q -n myQueue -m dsk -u admin -p admin
```

다음과 같은 출력이 표시됩니다.

Reserved	Used	Utilization Ratio
806400	804096	99
1793024	1793024	100
2544640	2518272	98

하위 명령 출력에서 각 열의 의미는 다음과 같습니다.

표 6-11 대상 디스크 사용률 메트릭

메트릭	설명
예약됨	활성 메시지가 보관된 레코드와 재사용 대기 중인 사용 가능한 레코드를 포함한 모든 레코드가 사용하는 디스크 공간(바이트)
사용됨	활성 메시지가 보관된 레코드에서 사용하는 디스크 공간(바이트)
사용률	예약된 디스크 공간에서 사용되는 디스크 공간의 비율. 비율이 높을수록 활성 메시지를 보관하는 데 사용되고 있는 디스크 공간이 많은 것입니다.

사용되지 않은 대상 디스크 공간 확보

특정 대상을 사용하는 메시징 응용 프로그램의 특성에 따라 디스크 사용률 패턴이 다릅니다. 대상에 유입 및 유출되는 메시지의 상대적 흐름과 상대적 메시지 크기에 따라 예약된 디스크 공간이 점점 더 커질 수 있습니다.

메시지 생성 속도가 메시지 사용 속도보다 큰 경우 사용 가능한 레코드를 다시 사용하고 사용률을 높은 수준으로 유지해야 합니다. 그러나 메시지 생성 속도가 메시지 사용 속도보다 작거나 비슷한 경우 사용률이 낮아도 됩니다.

일반적으로 예약된 디스크 공간은 안정적으로 유지하고 사용률은 높게 유지해야 합니다. 통상 시스템이 예약된 디스크 공간이 매우 일정하게 유지되고 사용률이 높은(75% 이상) 안정적인 상태에 도달하는 경우 사용되지 않는 디스크 공간을 확보할 필요가 없습니다. 시스템이 안정적인 상태에 도달하고 사용률이 낮은(50% 이하) 경우 디스크를 압축하여 사용 가능한 레코드가 사용 중인 디스크 공간을 확보할 수 있습니다.

예약된 디스크 공간이 점점 증가하는 경우 대상 메모리 제한 등록 정보와 제한 동작(171 페이지의 표 6-10 참조)을 설정하여 대상의 메모리 관리를 다시 구성해야 합니다.

▶ 사용되지 않는 대상 디스크 공간을 확보하는 방법

1. 대상을 일시 중지합니다.

```
imqcmd pause dst -t q -n myQueue -u admin -p admin
```

2. 디스크를 압축합니다.

```
imqcmd compact dst -t q -n myQueue -u admin -p admin
```

3. 대상을 다시 시작합니다.

```
imqcmd resume dst -t q -n myQueue -u admin -p admin
```

대상 유형과 이름을 지정하지 않으면 이 작업이 모든 대상에 대해 수행됩니다.

영구 가입 관리

imqcmd 하위 명령을 사용해서 브로커의 영구 가입을 관리할 수 있습니다. 영구 가입은 클라이언트에 영구로 등록된 주제에 가입하는 것입니다. 여기에는 고유한 아이디가 있으며, 사용자가 비활성 상태인 동안에도 브로커에서 가입에 해당하는 메시지를 보존해야 합니다. 보통 브로커는 메시지가 만료될 때 영구 가입자에 대해 보존된 메시지만 삭제할 수 있습니다.

표 6-12에서는 imqcmd 영구 가입 하위 명령을 요약합니다. 기본(localhost:7676) 브로커가 아닌 경우 브로커의 호스트 이름과 포트를 지정해야 합니다.

표 6-12 영구 가입 관리에 사용되는 imqcmd 하위 명령

하위 명령	설명
<code>list dur -d destName</code>	지정한 대상의 모든 영구 가입을 나열합니다.
<code>destroy dur -n subscrName -c client_id</code>	지정한 클라이언트 식별자에 해당하는 지정된 영구 가입을 완전 삭제합니다(45페이지의 "클라이언트 식별자" 참조).
<code>purge dur -n subscrName -c client_id</code>	지정한 클라이언트 식별자에 해당하는 지정된 영구 가입에 대한 모든 메시지를 제거합니다(45페이지의 "클라이언트 식별자" 참조).

예를 들어, 다음 명령은 SPQuotes 주제의 모든 영구 가입을 나열합니다.

```
imqcmd list dur -d SPQuotes
```

list dur 하위 명령은 주제의 각 영구 가입에 대해 영구 가입의 이름과 사용자의 클라이언트 아이디, 이 주제의 대기열에 들어 있는 메시지의 수, 영구 가입 상태(활성/비활성)를 반환합니다. 예를 들면 다음과 같습니다.

Name	Client ID	Number of Messages	Durable Sub State
myDurable	myClientID	1	INACTIVE

list dur 하위 명령에서 반환된 정보를 사용하여 완전 삭제하거나 메시지를 제거할 영구 가입을 확인할 수 있습니다. 가입 이름과 클라이언트 아이디를 사용하여 가입을 확인합니다. 예를 들면 다음과 같습니다.

```
imqcmd destroy dur -n myDurable -c myClientID
```

트랜잭션 관리

클라이언트 응용 프로그램에서 시작하는 모든 트랜잭션은 브로커에서 추적됩니다. 이는 단순한 Message Queue 트랜잭션일 수도 있고 XA 자원 관리자가 관리하는 분산 트랜잭션일 수도 있습니다(47페이지의 "로컬 트랜잭션" 참조). 모든 트랜잭션에는 브로커의 트랜잭션을 고유하게 나타내는 64비트 숫자, 즉 Message Queue 트랜잭션 아이디가 있습니다. 분산 트랜잭션에도 분산 트랜잭션 관리자가 할당하는 최대 128바이트의 분산 트랜잭션 아이디(XID)가 있습니다. Message Queue에서는 Message Queue 트랜잭션 아이디와 XID 사이의 연결을 보존합니다.

분산 트랜잭션에 오류가 발생하면 그 트랜잭션은 PREPARED 상태로 남아 완결되지 않을 수도 있습니다. 따라서 관리자는 준비된 상태로 남아 있는 트랜잭션을 모니터링하고 롤백 또는 완결해야 합니다.

표 6-13에서는 imqcmd 트랜잭션 하위 명령을 요약합니다. 기본(localhost:7676) 브로커가 아닌 경우 브로커의 호스트 이름과 포트를 지정해야 합니다.

표 6-13 트랜잭션 관리에 사용되는 imqcmd 하위 명령

하위 명령	설명
list txn	브로커에서 추적하는 모든 트랜잭션을 나열합니다.
query txn -n <i>transaction_id</i>	지정한 트랜잭션에 대한 정보를 나열합니다.
commit txn -n <i>transaction_id</i>	지정한 트랜잭션을 완결합니다.
rollback txn -n <i>transaction_id</i>	지정한 트랜잭션을 롤백합니다.

예를 들어, 다음 명령은 브로커에 있는 모든 트랜잭션을 나열합니다.

```
imqcmd list txn
```

list 하위 명령은 각 트랜잭션에 대해 트랜잭션 아이디, 상태, 사용자 아이디, 메시지 또는 확인 응답의 수, 작성 시간을 반환합니다. 예를 들면 다음과 같습니다.

Transaction ID	State	User name	# Msgs/ # Acks	Creation time
64248349708800	PREPARED	guest	4/0	1/30/02 10:08:31 AM
64248371287808	PREPARED	guest	0/4	1/30/02 10:09:55 AM

명령은 브로커에 있는 로컬 및 분산 트랜잭션을 모두 표시합니다. PREPARED 상태인 트랜잭션만을 완결 또는 롤백할 수 있습니다. 트랜잭션이 오류로 인해 준비 상태에 있고 분산 트랜잭션 관리자에서 완결을 진행하고 있지 않다는 것이 확인된 경우에만 완결 또는 롤백할 수 있습니다.

예를 들어, 브로커의 자동 롤백 등록 정보가 **false**로 설정되어 있으면(62페이지의 표 2-4 참조) 브로커를 시작할 때 PREPARED 상태인 트랜잭션을 수동으로 완결 또는 롤백해야 합니다.

list 하위 명령도 트랜잭션에서 생성된 메시지의 수와 트랜잭션에서 확인 응답된 (#Msgs/#Acks) 메시지의 수를 표시합니다. 트랜잭션이 완결될 때까지는 이런 메시지가 전달되지 않으며 확인 응답이 처리되지도 않습니다.

query 하위 명령을 사용하면 같은 정보와 클라이언트 아이디, 연결 아이디, 분산 트랜잭션 아이디(XID) 등의 여러 추가 값을 볼 수 있습니다. 예를 들면 다음과 같습니다.

```
imqcmd query txn -n 64248349708800
```

다음과 같은 출력이 표시됩니다.

Client ID	
Connection	guest@192.18.116.219:62209->jms:62195
Creation time	1/30/02 10:08:31 AM
Number of acknowledgements	0
Number of messages	4
State	PREPARED
Transaction ID	64248349708800
User name	guest
XID	
	6469706F6C7369646577696E6465723130313234313431313030373230

commit 및 rollback 하위 명령은 분산 트랜잭션을 완결 또는 롤백할 때 사용할 수 있습니다. 앞에서 설명한 것과 같이, PREPARED 상태에 있는 트랜잭션만 완결 또는 롤백할 수 있습니다. 예를 들면 다음과 같습니다.

```
imqcmd commit txn -n 64248349708800
```

브로커를 시작할 때 PREPARED 상태에 있는 트랜잭션을 자동으로 롤백하도록 브로커를 구성할 수도 있습니다. 자세한 내용은 [62페이지의 표 2-4](#)의 `imq.transaction.autorollback` 등록 정보를 참조하십시오.

관리 대상 객체 관리

관리 대상 객체를 사용하면 다른 JMS 공급자에 이식할 수 있는 클라이언트 응용 프로그램을 개발할 수 있습니다. *관리 대상 객체*는 공급자별 구성 및 이름 지정 정보를 캡슐화하는 객체입니다. 이 객체는 보통 Message Queue 관리자가 만들며, 클라이언트 응용 프로그램에서 브로커에 연결을 설정하고 이어서 물리적 대상과 메시지를 주고 받을 때 사용합니다.

관리 대상 객체에 대한 개요는 89페이지의 "[Message Queue 관리 대상 객체](#)"를 참조하십시오.

Message Queue에는 관리 대상 객체를 만들고 관리할 때 사용되는 두 개의 관리 도구가 있습니다. 하나는 명령줄 객체 관리자 유틸리티(imqobjmgr)이고 다른 하나는 GUI 관리 콘솔입니다. 이 도구를 사용하면 다음을 수행할 수 있습니다.

- 객체 저장소에서 관리 대상 객체를 추가 또는 삭제합니다.
- 기존 관리 대상 객체를 나열합니다.
- 관리 객체 관련 정보를 쿼리 및 표시합니다.
- 객체 저장소에 있는 기존 관리 대상 객체를 수정합니다.

이 장에서는 객체 관리자 유틸리티(imqobjmgr)를 사용해서 이런 작업을 수행하는 방법을 설명합니다. 이 작업을 수행하려면 사용할 객체 저장소와 만들 관리 대상 객체의 속성을 이해하고 있어야 하므로 이 장에서 imqobjmgr을 사용하여 관리 대상 객체를 관리하는 방법을 설명하기 전에 이 두 항목에 대한 배경 정보를 제공합니다.

관리 콘솔 사용에 대한 자세한 내용은 4장, "[관리 콘솔 자습서](#)"를 참조하십시오.

객체 저장소 정보

관리 대상 객체는 클라이언트 응용 프로그램에서 JNDI 조회를 통해 액세스할 수 있도록 미리 만들어진 객체 저장소에 있습니다. 사용할 수 있는 객체 저장소의 유형에는 표준 LDAP 디렉토리 서버 또는 파일 시스템 객체 저장소의 두 가지가 있습니다.

LDAP 서버 객체 저장소

LDAP 서버는 작업 메시징 시스템에 권장되는 객체 저장소입니다. LDAP 구현은 여러 공급업체에서 제공하며 분산 시스템에서 사용할 수 있도록 디자인되어 있습니다. LDAP 서버는 작업 환경에 유용한 보안 기능도 제공합니다.

Message Queue 관리 도구는 LDAP 서버의 객체 저장소를 관리할 수 있습니다. 그러나 먼저 LDAP 서버 설명서에 나와 있는 대로 java 객체를 저장하고 JNDI 조회를 수행하도록 LDAP 서버를 구성해야 합니다.

LDAP 서버를 객체 저장소로 사용하는 경우 표 7-1에 나와 있는 속성들을 지정해야 합니다. 이 속성들은 다음 범주로 구분됩니다.

- **초기 컨텍스트:** LDAP 서버 객체 저장소의 이 속성은 고정되어 있습니다.
- **위치:** LDAP 서버에 설정된 대로 관리 대상 객체를 저장할 URL 및 디렉토리 경로를 지정합니다. 특히 지정된 경로가 존재하는지 확인해야 합니다.
- **보안 정보:** LDAP 공급자에 따라 다릅니다. 보안 정보가 모든 작업에서 필요한지 아니면 저장된 데이터를 변경하는 작업에서만 필요한지 확인하려면 LDAP 구현에 제공된 설명서를 참조하십시오.

표 7-1 LDAP 객체 저장소 속성

속성	설명
java.naming.factory.initial	LDAP 서버에서 JNDI 조회의 초기 컨텍스트. com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url	LDAP 서버 URL 및 디렉토리 경로 정보. 예를 들면 다음과 같습니다. ldap://mydomain.com:389/ou=mqobjs, o=myapp 여기서 관리 대상 객체는 /myapp/mqobjs 디렉토리에 저장됩니다.

표 7-1 LDAP 객체 저장소 속성(계속)

속성	설명
java.naming.security.principal	LDAP 서버에 대해 호출자를 인증할 때 사용하는 principal의 아이디. 그 형식은 인증 방법에 따라 달라지며, 예를 들면 다음과 같습니다. uid=fooUser, ou=People, o=mcq 이 등록 정보를 지정하지 않으면 LDAP 서비스 공급자가 동작을 결정합니다.
java.naming.security.credentials	LDAP 서버에 대해 호출자를 인증할 때 사용하는 principal의 자격 증명. 이 등록 정보 값은 인증 방법에 따라 다르며 해시 비밀번호, 단순 텍스트 비밀번호, 키, 인증서 등이 될 수 있습니다. 예를 들면 다음과 같습니다. fooPasswd 이 등록 정보를 지정하지 않으면 LDAP 서비스 공급자가 동작을 결정합니다.
java.naming.security.authentication	사용할 보안 수준입니다. 값은 none, simple, strong의 세 키워드 중 하나입니다. 예를 들어, simple를 지정하면 누락된 principal 또는 자격 증명 값을 묻는 프롬프트가 나타납니다. 그러면 아이디 정보를 좀더 안전하게 제공할 수 있습니다. 이 등록 정보를 지정하지 않으면 LDAP 서비스 공급자가 동작을 결정합니다.

파일 시스템 객체 저장소

Message Queue에서는 파일 시스템 객체 저장소 구현도 지원합니다. 파일 시스템 객체 저장소는 완전히 테스트되지 않았기 때문에 작업 시스템에는 권장되지 않지만, 개발 환경에서는 매우 사용하기가 쉽다는 장점이 있습니다. LDAP 서버를 설정할 필요 없이 로컬 파일 시스템에 디렉토리를 만들기만 하면 됩니다.

그러나 클라이언트가 여러 컴퓨터 노드에 배포된 경우 이 클라이언트들이 객체 저장소가 위치한 디렉토리에 액세스할 수 없으면 파일 시스템 저장소를 중앙 집중식 객체 저장소로 사용할 수 없습니다. 또한 해당 디렉토리에 액세스할 수 있는 모든 사용자는 Message Queue 관리 도구를 사용하여 관리 대상 객체를 만들고 관리할 수 있습니다.

파일 시스템 객체 저장소를 사용하는 경우 표 7-2에 나와 있는 속성들을 지정해야 합니다. 이 속성들은 다음 범주로 구분됩니다.

- **초기 컨텍스트:** 파일 시스템 객체 저장소의 이 속성 값은 고정되어 있습니다.
- **위치:** 이 속성 값은 관리 대상 객체를 저장할 디렉토리 경로를 지정합니다. 해당 디렉토리가 존재해야 하며 Message Queue 관리 도구 사용자와 이 저장소에 액세스할 클라이언트 응용 프로그램 사용자에게 적절한 액세스 권한이 있어야 합니다.

표 7-2 파일 시스템 객체 저장소 속성

속성	설명
java.naming.factory.initial	파일 시스템 객체 저장소에서 JNDI 조회의 초기 컨텍스트: com.sun.jndi.fscontext.RefFSContextFactory
java.naming.provider.url	디렉토리 경로 정보. 예를 들면 다음과 같습니다. file:///C:/myapp/mqobjs

관리 대상 객체

관리 객체에 대한 개요는 [89페이지](#)의 "[Message Queue 관리 대상 객체](#)"를 참조하십시오.

Message Queue 관리 대상 객체에는 기본적으로 연결 팩토리와 대상의 두 가지 종류가 있습니다. *연결 팩토리* 관리 대상 객체는 클라이언트 응용 프로그램에서 브로커에 대한 연결을 만들 때 사용됩니다. *대상* 관리 대상 객체는 클라이언트 응용 프로그램에서 생성자가 메시지를 보내거나 사용자가 메시지를 받는 대상을 나타낼 때 사용됩니다. (SOAP 메시징에서는 특수한 SOAP 중점 관리 대상 객체를 사용합니다. 자세한 내용은 *Message Queue Java Client Developer's Guide*를 참조하십시오.)

메시지 전달 모델(지점간 또는 게시/가입)에 따라 특정 유형의 연결 팩토리 및 대상을 사용할 수 있습니다. 예를 들어, 지점간 프로그래밍에서는 대기열 연결 팩토리와 대기열 대상을 사용할 수 있습니다. 마찬가지로 게시 및 가입 프로그래밍에서는 주제 연결 팩토리와 주제 대상을 사용할 수 있습니다. 불특정 연결 팩토리와 대상 관리 대상 객체 유형도 사용할 수 있으며, 분산 트랜잭션을 지원하는 연결 팩토리 유형도 마찬가지입니다(지원되는 모든 유형은 [45페이지](#)의 표 1-1 참조).

관리 대상 객체의 속성은 속성-값 쌍을 사용해서 지정합니다. 다음 절에서는 이런 속성에 대해 설명합니다.

연결 팩토리 관리 대상 객체 속성

연결 팩토리(및 XA 연결 팩토리) 관리 대상 객체에는 표 7-3에 나열된 것과 같은 속성이 있습니다. 여기에서 주로 고려할 속성은 클라이언트가 연결을 설정할 브로커를 지정할 때 사용하는 `imqAddressList`입니다. 195페이지의 "연결 팩토리 추가" 절에서는 객체 저장소에 연결 팩토리 관리 대상 객체를 추가할 때 속성을 지정하는 방법을 설명합니다.

연결 팩토리 속성과 그 사용 방법에 대한 자세한 설명은 *Message Queue Java Client Developer's Guide*와 JavaDoc API 설명서에서 `Message Queue` 클래스 `com.sun.messaging.ConnectionConfiguration` 부분을 참조하십시오.

표 7-3 연결 팩토리 관리 대상 객체 속성

속성/등록 정보 이름	유형	기본값
<code>imqAckOnAcknowledge</code>	문자열	없음
<code>imqAckOnProduce</code>	문자열	없음
<code>imqAckTimeout</code>	문자열	0밀리초
<code>imqAddressList</code>	문자열	없음
<code>imqAddressListIterations</code>	정수	1
<code>imqAddressListBehavior</code>	문자열	PRIORITY
<code>imqBrokerHostName (Message Queue 3.0)</code>	문자열	localhost
<code>imqBrokerHostPort (Message Queue 3.0)</code>	정수	7676
<code>imqBrokerServicePort (Message Queue 3.0)</code>	정수	0
<code>imqConfiguredClientID</code>	문자열	없음
<code>imqConnectionFlowCount</code>	정수	100
<code>imqConnectionFlowLimit</code>	정수	1000
<code>imqConnectionFlowLimitEnabled</code>	부울	false
<code>imqConnectionType (Message Queue 3.0)</code>	문자열	TCP

표 7-3 연결 팩토리 관리 대상 객체 속성(계속)

속성/등록 정보 이름	유형	기본값
imqConnectionURL (Message Queue 3.0)	문자열	http://localhost/imq/ tunnel
imqConsumerFlowLimit	정수	1000
imqConsumerFlowThreshold	정수	50
imqDefaultPassword	문자열	guest
imqDefaultUsername	문자열	guest
imqDisableSetClientID	부울	false
imqJMSDeliveryMode	정수	2 (지속성)
imqJMSExpiration	Long	0 (만료되지 않음)
imqJMSPriority	정수	4 (일반)
imqLoadMaxToServerSession	부울	true
imqOverrideJMSDeliveryMode	부울	false
imqOverrideJMSExpiration	부울	false
imqOverrideJMSHeadersTo TemporaryDestinations	부울	false
imqOverrideJMSPriority	부울	false
imqQueueBrowserMaxMessages PerRetrieve	정수	1000
imqQueueBrowserRetrieveTimeout	Long	60,000 (밀리초)
imqReconnectAttempts	정수	0
imqReconnectEnabled	부울	false
imqReconnectInterval	Long	3000 (밀리초)
imqSetJMSXAppID	부울	false
imqSetJMSXConsumerTXID	부울	false
imqSetJMSXProducerTXID	부울	false
imqSetJMSXRcvTimestamp	부울	false
imqSetJMSXUserID	부울	false
imqSSLIsHostTrusted (Message Queue 3.0)	부울	true

대상 관리 대상 객체 속성

물리적 주제 또는 대기열 대상을 나타내는 대상 관리 대상 객체는 표 7-4에 나열된 것과 같은 속성을 갖습니다. 196페이지의 "주제 또는 대기열 추가" 절에서는 객체 저장소에 대상 관리 대상 객체를 추가할 때 이런 속성을 지정하는 방법을 설명합니다.

여기에서 주로 고려해야 할 속성은 `imqDestinationName`입니다. 이 속성은 주제 또는 대기열 관리 대상 객체에 해당하는 물리적 대상에 지정하는 이름입니다. 대상에 설명을 입력해 두면 여러 응용 프로그램을 지원하기 위해 만드는 다른 대상과 구분하기가 쉽습니다.

자세한 내용은 JavaDoc API 설명서에서 `Message Queue` 클래스 `com.sun.messaging.DestinationConfiguration` 부분을 참조하십시오.

표 7-4 대상 관리 대상 객체 속성

속성/등록 정보 이름	유형	기본값
<code>imqDestinationDescription</code>	문자열	대상 객체의 설명
<code>imqDestinationName</code>	문자열 ¹	<code>Untitled_Destination_Object</code>

1. 대상 이름은 영문자와 숫자만 포함할 수 있으며(공백 없음) 영문자 또는 "_" 문자나 "\$" 문자로 시작해야 합니다.

객체 관리자 유틸리티(imqobjmgr)

객체 관리자 유틸리티를 사용하면 `Message Queue` 관리 대상 객체를 만들고 관리할 수 있습니다. 이 절에서는 기본 `imqobjmgr` 명령 구문을 설명하고, 하위 명령 목록을 제공하고, `imqobjmgr` 명령 옵션을 요약합니다. 다음 절에서는 `imqobjmgr` 하위 명령을 사용하여 특정 작업을 수행하는 방법에 대해 설명합니다.

imqobjmgr 명령 구문

`imqobjmgr` 명령의 일반 구문은 다음과 같습니다.

```
imqobjmgr subcommand [options]
imqobjmgr -h|H
imqobjmgr -v
```

`-v`, `-h` 또는 `-H` 옵션을 지정하는 경우 명령줄에 지정된 하위 명령이 실행되지 않습니다. 예를 들어, 다음 명령을 입력하면 버전 정보는 표시되지만 `list` 하위 명령은 실행되지 않습니다.

```
imqobjmgr list -v
```

imqobjmgr 하위 명령

객체 관리자 유틸리티(imqobjmgr)에는 표 7-5에 나열된 하위 명령이 포함되어 있습니다.

표 7-5 imqobjmgr 하위 명령

하위 명령	설명
<code>add</code>	객체 저장소에 관리 대상 객체를 추가합니다.
<code>delete</code>	객체 저장소에서 관리 대상 객체를 삭제합니다.
<code>list</code>	객체 저장소에 있는 관리 대상 객체를 나열합니다.
<code>query</code>	지정한 관리 대상 객체에 대한 정보를 표시합니다.
<code>update</code>	객체 저장소에 있는 기존 관리 대상 객체를 수정합니다.

imqobjmgr 명령 옵션 요약

표 7-6에는 imqobjmgr 명령의 옵션이 나열되어 있습니다. 사용 설명은 다음에서 해당 작업 기반 절을 참조하십시오.

표 7-6 imqobjmgr 옵션

옵션	설명
<code>-f</code>	사용자의 확인 없이 작업을 수행합니다.
<code>-h</code>	사용 도움말을 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
<code>-H</code>	사용 도움말, 속성 목록 및 예를 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
<code>-i fileName</code>	하위 명령 절의 일부 또는 전부를 포함하는 명령 파일 이름을 지정하며, 객체 유형, 조회 이름, 객체 속성, 객체 저장소 속성 또는 기타 옵션을 지정합니다. 보통은 객체 저장소 속성과 같은 반복 정보에 사용됩니다.

표 7-6 imqobjmgr 옵션(계속)

옵션	설명
-j <i>attribute=value</i>	JNDI 객체 저장소를 확인하고 액세스할 때 필요한 속성을 지정합니다. 184페이지의 "LDAP 서버 객체 저장소" 및 185페이지의 "파일 시스템 객체 저장소" 를 참조하십시오.
-javahome <i>path</i>	Java 2와 호환할 수 있는 대체 런타임을 지정하여 사용합니다(기본값은 시스템의 런타임이나 Message Queue 와 함께 제공되는 런타임 사용).
-l <i>lookupName</i>	관리 객체의 JNDI 조회 이름을 지정합니다. 이 이름은 관리 대상 객체 저장소의 컨텍스트에서 고유해야 합니다.
-o <i>attribute=value</i>	관리 객체의 속성을 지정합니다. 187페이지의 "연결 팩토리 관리 대상 객체 속성" 및 189페이지의 "대상 관리 대상 객체 속성" 을 참조하십시오.
-pre	미리 보기 모드입니다. 명령을 수행하지 않고 수행 결과를 확인합니다.
-r <i>read-only_state</i>	관리 대상 객체가 읽기 전용인지 여부를 지정합니다. true 값은 관리 대상 객체가 읽기 전용임을 나타냅니다. 클라이언트는 읽기 전용 관리 대상객체의 속성을 수정할 수 없습니다. 읽기 전용 상태는 기본적으로 false로 설정됩니다.
-s	비대화형 모드입니다. 출력이 표시되지 않습니다.
-t <i>objectType</i>	Message Queue 관리 대상 객체의 유형을 지정합니다. q = 대기열 t = 주제 cf = 연결 팩토리 qf = 대기열 연결 팩토리 tf = 주제 연결 팩토리 xcf = XA 연결 팩토리(분산 트랜잭션) xqf = XA 대기열 연결 팩토리(분산 트랜잭션) xtf = XA 주제 연결 팩토리(분산 트랜잭션) e = SOAP 종점 ¹
-v	버전 정보를 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.

1. 이 관리 대상 객체 유형은 SOAP 메시지 지원에 사용됩니다(*Message Queue Java Client Developer's Guide* 참조).

다음 절에서는 imqobjmgr 하위 명령을 사용해서 작업할 때 제공해야 하는 정보에 대해 설명합니다.

필요한 정보

관리 대상 객체와 관련된 대부분의 작업을 수행할 때 imqobjmgr 하위 명령의 옵션으로 다음 정보를 지정해야 합니다.

- **관리 대상 객체 유형:**

허용되는 유형은 표 7-6에 나와 있습니다.

- **관리 대상 객체의 JNDI 조회 이름:**

클라이언트 코드에서 객체 저장소에 있는 관리 대상 객체를 참조할 때(JNDI 사용) 사용되는 논리적 이름입니다.

- **관리 대상 객체 속성(특히 add 및 update 하위 명령에서 필요):**

- 대상의 경우: 브로커에 있는 물리적 대상의 이름. imqcmd create dst 하위 명령의 -n 옵션으로 지정했던 이름입니다. 이름을 지정하지 않으면 기본 이름인 Untitled_Destination_Object가 사용됩니다.

- 연결 팩토리의 경우: 가장 많이 사용하는 속성은 클라이언트가 연결을 시도할 하나 이상의 메시지 서버 주소를 지정하는 주소 목록(imqAddressList). 이 정보를 지정하지 않으면 로컬 호스트와 기본 포트 번호(7676)를 사용합니다. 즉 클라이언트는 로컬 호스트의 포트 7676에 있는 브로커에 연결을 시도합니다. [195페이지의 "연결 팩토리 추가"](#) 절에서는 객체 속성을 지정하는 방법에 대해 설명합니다.

추가 속성에 대한 내용은 [187페이지의 "연결 팩토리 관리 대상 객체 속성"](#)을 참조하십시오.

- **객체 저장소 속성:**

이 정보는 파일 시스템 저장소 또는 LDAP 서버 중 어느 것을 사용하는지에 따라 달라지지만 다음 속성을 포함해야 합니다.

- JNDI 구현 유형(initial context 속성). 예를 들어, 파일 시스템 또는 LDAP입니다.
- 객체 저장소의 관리 대상 객체 위치(공급자 URL 속성), 즉 "폴더"
- 객체 저장소 액세스에 필요한 사용자 아이디, 비밀번호, 권한 부여 유형(있는 경우)

객체 저장소 속성에 대한 자세한 내용은 [184페이지의 "LDAP 서버 객체 저장소"](#) 및 [185페이지의 "파일 시스템 객체 저장소"](#)를 참조하십시오.

명령 파일 사용

imqobjmgr 명령을 사용하면 imqobjmgr 하위 명령 절의 일부 또는 전부에 Java 등록 정보 파일 구문을 사용하는 명령 파일의 이름을 지정할 수 있습니다.

객체 관리자 유틸리티(imqobjmgr)에 명령 파일을 사용하면 여러 imqobjmgr 실행에서 동일하게 사용되면서 입력할 내용이 많은 객체 저장소 속성을 지정할 때 유용합니다. 명령 파일을 사용하면 명령줄에 허용된 최대 문자 수를 초과하는 상황을 방지할 수도 있습니다.

imqobjmgr 명령 파일의 일반 구문은 다음과 같습니다(버전 등록 정보는 Message Queue 제품이 아니라 명령 파일의 버전을 나타내며(명령줄 옵션이 아님), 그 값을 2.0으로 설정해야 함).

```
version=2.0
cmdtype=[ add | delete | list | query | update ]
obj.type=[ q | t | qf | tf | cf | xqf | xtf | xcf | e ]
obj.lookupName=lookup Name
obj.attrs.objAttrName1=value1
obj.attrs.objAttrName2=value2
obj.attrs.objAttrNameN=valueN
...
objstore.attrs.objStoreAttrName1=value1
objstore.attrs.objStoreAttrName2=value2
objstore.attrs.objStoreAttrNameN=valueN
...
```

아래에서는 imqobjmgr 명령을 예로 들어 명령 파일의 사용 방법을 설명합니다.

```
imqobjmgr add
-t qf
-l "cn=myQCF"
-o "imqAddressList=mq://foo:777/jms"
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=imq"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=imq"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

이 명령은 다음과 같은 내용을 가진 MyCmdFile과 같은 파일에 캡슐화할 수 있습니다.

```
version=2.0
cmdtype=add
obj.type=qf
obj.lookupName=cn=myQCF
obj.attrs.imqAddressList=mq://foo:777/jms
objstore.attrs.java.naming.factory.initial=\
    com.sun.jndi.ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=\
    ldap://mydomain.com:389/o=imq
objstore.attrs.java.naming.security.principal=\
    uid=fooUser, ou=People, o=imq
objstore.attrs.java.naming.security.credentials=fooPasswd
objstore.attrs.java.naming.security.authentication=simple
```

그리고 -i 옵션을 사용하면 이 파일을 객체 관리자 유틸리티(imqobjmgr)로 전달할 수 있습니다.

```
imqobjmgr -i MyCmdFile
```

명령 파일에 몇 가지 옵션을 지정하고 명령줄을 사용해서 또 다른 옵션을 지정할 수도 있습니다. 이 경우 하위 명령 절 중 유틸리티가 실행될 때마다 동일하게 유지되는 부분을 지정하는 데 명령 파일을 사용할 수 있습니다. 예를 들어, 다음 명령은 관리 대상 객체의 저장 위치를 제외하고 연결 팩토리 관리 대상 객체를 추가할 때 필요한 모든 옵션을 지정합니다.

```
imqobjmgr add
-t qf
-l "cn=myQCF"
-o "imqAddressList=mq://foo:777/jms"
-i MyCmdFile
```

이 경우 MyCmdFile 파일에는 다음과 같은 정의가 포함됩니다.

```
version=2.0
objstore.attrs.java.naming.factory.initial=\
    com.sun.jndi.ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=\
```

```

ldap://mydomain.com:389/o=img
objstore.attrs.java.naming.security.principal=\
uid=fooUser, ou=People, o=img
objstore.attrs.java.naming.security.credentials=fooPasswd
objstore.attrs.java.naming.security.authentication=simple

```

다음 위치에서 명령 파일의 예를 추가로 확인할 수 있습니다.

IMQ_HOME/demo/imqobjmgr

관리 대상 객체 추가 및 삭제

이 절에서는 연결 팩토리 및 주제 또는 대기열 대상의 관리 대상 객체를 객체 저장소에 추가하는 방법을 설명합니다.

주 객체 관리자 유틸리티(imqobjmgr)는 Message Queue 관리 대상 객체만을 나열하고 표시합니다. 객체 저장소에 추가할 관리 대상 객체와 동일한 조회 이름을 가진 비 Message Queue 객체를 포함해야 하는 경우에 이 객체를 추가하려고 하면 오류 메시지가 표시됩니다.

연결 팩토리 추가

클라이언트 응용 프로그램에서 브로커에 대한 연결을 설정할 수 있게 하려면 클라이언트 응용 프로그램에서 원하는 연결 유형(주제 연결 팩토리 또는 대기열 연결 팩토리)을 나타내는 관리 대상 객체를 추가합니다.

대기열 연결 팩토리를 추가하려면 다음과 같은 명령을 사용합니다.

```

imqobjmgr add
-t qf
-l "cn=myQCF"
-o "imqAddressList=mq://myHost:7272/jms"
-j "java.naming.factoryinitial=
com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=ldap://mydomain.com:389/o=img"
-j "java.naming.security.principal=

```

```
uid=fooUser, ou=People, o=img"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

앞의 명령은 조회 이름이 cn=myQCF이고, myHost에서 실행되며 포트 7272를 수신하는 브로커에 연결되는 관리 대상 객체를 만듭니다. 관리 대상 객체는 LDAP 서버에 저장됩니다. imqobjmgr 명령의 인수로 명령 파일을 지정해도 같은 결과를 얻을 수 있습니다. 자세한 내용은 193페이지의 "명령 파일 사용"을 참조하십시오.

주 이름 지정 규약: LDAP 서버를 사용하여 관리 대상 객체를 저장할 경우 위에서처럼 접두어 "cn="이 있는 조회 이름을 지정해야 합니다 (cn=myQCF). -l 옵션을 사용하여 조회 이름을 지정합니다. 파일 시스템 객체 저장소를 사용하는 경우 cn 접두어를 사용할 필요가 없습니다. 그러나 "/"가 포함된 조회 이름은 사용하지 마십시오. 표 7-7을 참조하십시오.

표 7-7 이름 지정 규약 예

객체 저장소 유형	올바른 이름	잘못된 이름
LDAP 서버	cn=myQCF	myQCF
파일 시스템	myTopic	myObjects/myTopic

주제 또는 대기열 추가

클라이언트 응용 프로그램에서 브로커의 물리적 대상에 액세스할 수 있게 하려면 이 대상에 해당하는 관리 대상 객체를 객체 저장소에 추가합니다.

객체 저장소에 해당 관리 대상 객체를 추가하기 전에 물리적 대상을 먼저 만드는 것이 좋습니다. 명령 유틸리티(imqcmd)를 사용하여 브로커에 객체 저장소의 대상 관리 대상 객체에 해당하는 물리적 위치를 만듭니다. 물리적 대상을 만드는 자세한 내용은 167페이지의 "연결 정보 얻기"를 참조하십시오.

다음 명령은 조회 이름이 myTopic이고 물리적 대상 이름이 TestTopic인 주제 대상에 해당하는 관리 대상 객체를 추가합니다. 관리 객체는 LDAP 서버에 저장됩니다.

```

imqobjmgr add
-t t
-l "cn=myTopic"
-o "imqDestinationName=TestTopic"
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=imq"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=imq"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"

```

같은 명령이지만 관리 대상 객체만 Solaris 파일 시스템에 저장됩니다.

```

imqobjmgr add
-t t
-l "cn=myTopic"
-o "imqDestinationName=TestTopic"
-j "java.naming.factory.initial=
    com.sun.jndi.fscontext.RefFSContextFactory"
-j "java.naming.provider.url=
    file:///home/foo/imq_admin_objects"

```

예를 들어, LDAP 서버 사례에서 명령 파일 MyCmdFile을 사용하여 하위 명령 절을 지정할 수 있습니다. 파일에는 다음과 같은 텍스트가 포함됩니다.

```

version=2.0
cmdtype=add
obj.type=t
obj.lookupName=cn=myTopic
obj.attrs.imqDestinationName=TestTopic
objstore.attrs.java.naming.factory.initial=
    com.sun.jndi.fscontext.RefFSContextFactory
objstore.attrs.java.naming.provider.url=
    file:///home/foo/imq_admin_objects
objstore.attrs.java.naming.security.principal=
    uid=fooUser, ou=People, o=imq
objstore.attrs.java.naming.security.credentials=fooPasswd
objstore.attrs.java.naming.security.authentication=simple

```

-i 옵션을 사용해서 해당 파일을 imqobjmgr 명령에 전달합니다.

```
imqobjmgr -i MyCmdFile
```

주 LDAP 서버를 사용하여 관리 대상 객체를 저장할 경우 위 예에서처럼 접두어 "cn="이 있는 조회 이름을 지정해야 합니다. -l 옵션을 사용하여 조회 이름을 지정합니다. 파일 시스템 객체 저장소를 사용하는 경우 이 접두어를 사용하지 않아도 됩니다.

-t 옵션에 q를 지정한다는 점만 제외하면 대기열 객체를 추가하는 것도 이와 같습니다.

관리 대상 객체 삭제

관리 대상 객체를 삭제하려면 delete 하위 명령을 사용합니다. 객체의 조회 이름과 유형, 위치를 지정해야 합니다.

다음 명령은 조회 이름이 cn=myTopic이고 LDAP 서버에 저장되는 주제의 관리 대상 객체를 삭제합니다.

```
imqobjmgr delete
-t t
-l "cn=myTopic"
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=imq"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=imq"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

정보 얻기

객체 저장소에 있는 관리 대상 객체를 나열하고 개별 객체에 대한 정보를 표시하려면 list와 query 하위 명령을 사용합니다.

관리 대상 객체 나열

모든 관리 대상 객체의 목록을 보거나 특정 유형에 해당하는 모든 관리 대상 객체의 목록을 보려면 list 하위 명령을 사용합니다. 다음 샘플 코드에서는 관리 객체가 LDAP 서버에 저장되어 있다고 가정합니다.

다음 명령은 모든 객체를 나열합니다.

```
imgobjmgr list
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=img"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=img"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

다음 명령은 queue 유형의 모든 객체를 나열합니다.

```
imgobjmgr list
-t q
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=img"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=img"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

단일 객체 관련 정보

특정 관리 대상 객체에 대한 정보를 얻으려면 query 하위 명령을 사용합니다. 객체의 조회 이름과 관리 대상 객체를 포함하는 객체 저장소의 속성(초기 컨텍스트, 위치 등)을 지정해야 합니다.

다음 예에서는 query 하위 명령을 사용하여 조회 이름이 cn=myTopic인 객체에 관련된 정보를 표시합니다.

```
imgobjmgr query
-l "cn=myTopic"
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=img"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=img"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

관리 대상 객체 업데이트

관리 대상 객체의 속성을 수정하려면 `update` 명령을 사용합니다. 조회 이름과 객체 위치를 지정해야 합니다. 속성 값을 수정하려면 `-o` 옵션을 사용합니다.

이 명령은 주제 연결 팩토리를 나타내는 관리 대상 객체의 속성을 변경합니다.

```
imqobjmgr update
-t tf
-l "cn=MyTCF"
-o imqReconnectAttempts=3
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=imq"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=imq"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```


보안 관리

이 장에서는 인증, 권한 부여, 암호화 등 보안과 관련된 작업을 수행하는 방법에 대해 설명합니다.

사용자 인증. 사용자 저장소에 사용자 목록, 해당 그룹, 비밀번호를 관리해야 합니다. 브로커 인스턴스마다 다른 사용자 저장소를 사용할 수 있습니다. 이 장의 첫 부분에서는 저장소를 만들고, 채우고, 관리하는 방법에 대해 설명합니다. Message Queue 보안에 대한 소개는 [66페이지의 "보안 관리자"](#)를 참조하십시오.

사용자 권한 부여: 액세스 제어 등록 정보 파일. 브로커 작업에 대한 각 사용자의 액세스를 사용자 아이디 또는 그룹 멤버십에 매핑하는 액세스 제어 등록 정보 파일을 편집해야 합니다. 브로커 인스턴스마다 다른 액세스 제어 등록 정보 파일을 사용할 수 있습니다. 이 장의 두 번째 부분에서는 이 등록 정보 파일을 사용자 정의하는 방법을 설명합니다.

암호화: SSL 기반 서비스를 사용한 작업(엔터프라이즈판). Secure Socket Layer (SSL) 표준 기반의 연결 서비스를 사용하면 클라이언트와 브로커 사이에 전달되는 메시지를 암호화할 수 있습니다. Message Queue에서 암호화를 처리하는 방법에 대한 소개는 [68페이지의 "암호화\(엔터프라이즈판\)"](#)를 참조하십시오. 이 장의 마지막 부분에서는 SSL 기반 연결 서비스를 설정하는 방법에 대해 설명하고 SSL 사용에 대한 추가 정보를 제공합니다.

브로커에서 SSL 키 저장소, LDAP 사용자 저장소, 또는 JDBC 호환 영구 저장소에 대한 액세스 보안을 설정할 때 비밀번호가 필요한 경우에는 다음과 같은 세 가지 방법으로 비밀번호를 입력할 수 있습니다.

- 브로커가 시작할 때 시스템에서 프롬프트를 표시하게 합니다.
- 브로커를 시작할 때 명령줄 옵션으로 비밀번호를 전달합니다([134페이지의 "브로커 시작"](#) 및 [136페이지의 표 5-2](#) 참조).
- 시스템에서 브로커를 시작할 때 액세스하는 passfile에 비밀번호를 저장합니다([225페이지의 "Passfile 사용"](#) 참조).

사용자 인증

사용자가 브로커에 연결하려고 하면 브로커는 제공되는 아이디와 비밀번호를 검사하여 사용자를 인증한 후 각 브로커가 참조하는 브로커별 사용자 저장소의 정보와 일치하면 연결을 허용합니다. 이 저장소의 유형에는 두 가지가 있습니다.

- Message Queue와 함께 제공되는 플랫폼 파일 저장소

이 유형의 사용자 저장소는 사용하기가 매우 쉽습니다. 하지만 보안 공격에 취약하기 때문에 평가 및 개발 목적에 **만** 사용해야 합니다. 사용자 관리자 유틸리티 (`imqusermgr`)를 사용하면 저장소를 채우고 관리할 수 있습니다. 인증을 사용하려면 각 사용자의 아이디, 비밀번호, 사용자 그룹 이름으로 사용자 저장소를 채웁니다.

사용자 저장소의 설정 및 관리에 대한 자세한 내용은 ["플랫폼 파일 사용자 저장소 사용"](#)을 참조하십시오.

- LDAP 서버

LDAP v2 또는 v3 프로토콜을 사용하는 기존 또는 새 LDAP 디렉토리 서버가 될 수 있습니다. 플랫폼 파일 저장소만큼 사용이 쉽지는 않지만 안전하고 확장 가능하기 때문에 작업 환경에 더 적합합니다.

LDAP 사용자 저장소를 사용하는 경우에는 LDAP 공급업체에서 제공하는 도구를 사용하여 사용자 저장소를 채우고 관리해야 합니다. 자세한 내용은 [209페이지의 "사용자 저장소에 LDAP 서버 사용"](#)을 참조하십시오.

플랫폼 파일 사용자 저장소 사용

Message Queue에는 플랫폼 파일 사용자 저장소와 플랫폼 파일 사용자 저장소를 채우고 관리할 때 사용할 수 있는 명령줄 도구인 Message Queue 사용자 관리자(`imqusermgr`)가 제공됩니다. 다음 절에서는 플랫폼 파일 사용자 저장소에 대해 설명하고 Message Queue 사용자 관리자 유틸리티(`imqusermgr`)를 사용하여 해당 저장소를 채우고 관리하는 방법을 설명합니다.

사용자 저장소 만들기

플랫 파일 사용자 저장소는 인스턴스별로 고유합니다. 기본 사용자 저장소(passwd)가 시작하는 각 브로커 인스턴스에 대해 만들어집니다. 이 사용자 저장소는 저장소와 연관된 브로커 인스턴스의 이름(instanceName)으로 식별되는 디렉토리에 저장됩니다(부록 A, "Message Queue 데이터의 위치" 참조).

```
.../instances/instanceName/etc/passwd
```

저장소는 다음 표 8-1과 같이 두 개의 항목(행)으로 만들어집니다.

표 8-1 사용자 저장소 초기 항목

사용자 아이디	비밀번호	그룹	상태
admin	admin	admin	active
guest	guest	anonymous	active

초기 항목을 사용하면 관리자가 개입하지 않고도 설치된 Message Queue 브로커를 즉시 사용할 수 있습니다. 즉, Message Queue 브로커를 사용하기 위해 초기 사용자/비밀번호 설정을 할 필요가 없습니다.

초기 guest 사용자 항목을 사용하면 클라이언트에서 기본 guest 사용자 아이디와 비밀번호를 사용하여 브로커 인스턴스에 연결할 수 있습니다(예를 들어 테스트 목적으로).

초기 admin 사용자 항목을 사용하면 imqcmd 명령을 사용하여 기본 admin 사용자 아이디와 비밀번호로 브로커 인스턴스를 관리할 수 있습니다. 이 초기 항목을 업데이트하여 비밀번호를 변경하는 것이 좋습니다(208페이지의 "기본 관리자 비밀번호 변경" 참조).

다음 절에서는 플랫 파일 사용자 저장소를 채우고 관리하는 방법을 설명합니다.

사용자 관리자 유틸리티(imqusermgr)

사용자 관리자 유틸리티(imqusermgr)를 사용하면 플랫 파일 사용자 저장소를 편집하거나 채울 수 있습니다.

이 절에서는 기본 imqusermgr 명령 구문을 설명하고, 하위 명령 목록을 제공하고, imqusermgr 명령 옵션을 요약합니다. 다음 절에서는 imqobjmgr 하위 명령을 사용하여 특정 작업을 수행하는 방법에 대해 설명합니다.

imqusermgr를 사용하기 전에 주의해야 할 사항은 다음과 같습니다.

- 브로커별 사용자 저장소가 없는 경우 해당 브로커 인스턴스를 시작하여 만들어야 합니다.
- imqusermgr 명령은 브로커가 설치되는 호스트에서 실행해야 합니다.
- 저장소에 쓸 수 있는 적절한 권한이 있어야 합니다. Solaris 및 Linux의 경우 해당 브로커 인스턴스를 처음으로 만든 사용자나 루트 사용자여야 합니다.

imqusermgr 명령 구문

imqusermgr 명령의 일반 구문은 다음과 같습니다.

```
imqusermgr subcommand [options]
imqusermgr -h
imqusermgr -v
```

-v, 또는 -h 옵션을 지정하는 경우 명령줄에 지정된 하위 명령이 실행되지 않습니다. 예를 들어, 다음 명령을 입력하면 버전 정보는 표시되지만 list 하위 명령은 실행되지 않습니다.

```
imqusermgr list -v
```

imqusermgr 하위 명령

표 8-2에는 imqusermgr 하위 명령이 나열되어 있습니다.

표 8-2 imqusermgr 하위 명령

하위 명령	설명
add [-i instanceName] -u userName -p passwd [-g group] [-s]	지정한(또는 기본) 브로커 인스턴스 저장소에 사용자와 관련 비밀번호를 추가하고 선택적으로 사용자 그룹을 지정합니다.
delete [-i instanceName] -u userName [-s] [-f]	지정한(또는 기본) 브로커 인스턴스 저장소에서 지정한 사용자를 삭제합니다.
list [-i instanceName] [-u userName]	지정한(또는 기본) 브로커 인스턴스 저장소의 지정한 사용자 또는 모든 사용자에 대한 정보를 표시합니다.
update [-i instanceName] -u userName -p passwd [-a state] [-s] [-f]	지정한(또는 기본) 브로커 인스턴스 저장소에 있는 지정한 사용자의 비밀번호 및/또는 상태를 업데이트합니다.
update [-i instanceName] -u userName -a state [-p passwd] [-s] [-f]	

주 다음 절의 예에서는 기본 브로커 인스턴스인 경우를 가정합니다.

imqusermgr 명령 옵션 요약

표 8-3에는 imqusermgr 명령의 옵션이 나열되어 있습니다.

표 8-3 imqusermgr 옵션

옵션	설명
-a <i>active_state</i>	사용자 상태의 활성화 여부를 지정합니다(true/false). true 값은 활성화 상태를 나타내며 기본값입니다.
-f	사용자의 확인 없이 작업을 수행합니다.
-h	사용 도움말을 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
-i <i>instanceName</i>	명령이 적용될 브로커 인스턴스 사용자 저장소를 지정합니다. 지정하지 않으면 기본 <i>instanceName</i> , imqbroker가 적용됩니다.
-p <i>passwd</i>	사용자의 비밀번호를 지정합니다.
-g <i>group</i>	사용자 그룹을 지정합니다. 유효한 값에는 admin, user, anonymous가 있습니다.
-s	자동 모드를 설정합니다.
-u <i>userName</i>	사용자 아이디를 지정합니다.
-v	버전 정보를 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.

그룹

브로커 인스턴스의 사용자 저장소에 사용자 항목을 추가할 때, 사용자에게 사전 정의된 admin, user, anonymous의 세 가지 그룹 옵션 중 하나를 지정할 수 있습니다. 그룹을 지정하지 않으면 기본 그룹인 user가 할당됩니다.

- **admin 그룹.** 브로커 관리자에 사용됩니다. 이 그룹에 할당된 사용자는 기본적으로 브로커를 구성하고 관리할 수 있습니다. admin 그룹에 사용자를 두 명 이상 할당할 수도 있습니다.

- **user 그룹.** 관리자가 아닌 일반 Message Queue 클라이언트 사용자에게 사용됩니다. 대부분의 클라이언트 사용자는 user 그룹에서 인증된 브로커에 액세스합니다. 기본적으로 이 그룹의 응용 프로그램 클라이언트 사용자는 모든 주제와 대기열에서 메시지를 생성 및 사용하거나 모든 대기열의 메시지를 찾아볼 수 있습니다.
- **anonymous 그룹.** 브로커에 알려진 사용자 아이디를 사용하지 않는(클라이언트 응용 프로그램에서 사용하는 실제 사용자의 아이디를 알지 못하기 때문일 수 있음) Message Queue 클라이언트에 사용됩니다. 이 그룹은 대부분의 FTP 서버에 있는 anonymous 계정과 유사합니다. anonymous 그룹에 한 번에 한 사용자만 할당할 수 있습니다. 액세스 제어를 통해 이 그룹의 액세스 권한을 user 그룹에 비해 제한하거나 배포할 때 이 그룹에서 사용자를 제거해야 합니다.

사용자의 그룹을 변경하려면, 사용자 항목을 삭제한 후 그 사용자에게 다른 항목을 추가하여 새 그룹을 지정해야 합니다.

그룹의 구성원이 수행할 수 있는 작업을 정의하는 액세스 규칙을 지정할 수 있습니다. 자세한 내용은 [212페이지의 "사용자 권한 부여: 액세스 제어 등록 정보 파일"](#)을 참조하십시오.

상태

사용자를 저장소에 추가하면 그 사용자는 기본적으로 활성 상태가 됩니다. 사용자를 비활성 상태로 만들려면 update 명령을 사용해야 합니다. 예를 들어, 다음 명령은 사용자 JoeD를 비활성 상태로 만듭니다.

```
imqusermgr update -u JoeD -a false
```

비활성 상태인 사용자의 항목은 저장소에 보존되지만, 비활성 상태인 사용자가 새 연결을 열 수는 없습니다. 사용자가 비활성 상태일 때 같은 이름을 가진 다른 사용자를 추가하면 작업이 실패합니다. 비활성 사용자 항목을 삭제하거나, 새 사용자의 이름을 변경하거나, 새 사용자에게 다른 이름을 사용해야 합니다. 그러면 중복되는 사용자 아이디를 추가하지 않게 됩니다.

사용자 아이디 및 비밀번호 형식

사용자 아이디와 비밀번호는 다음과 같은 지침을 따라야 합니다.

- 사용자 아이디는 [표 8-4](#)에 있는 문자를 포함할 수 없습니다.

표 8-4 사용자 아이디 및 비밀번호에 유효하지 않은 문자

문자	설명
*	별표
,	쉼표

표 8-4 사용자 아이디 및 비밀번호에 유효하지 않은 문자(계속)

문자	설명
:	콜론

- 사용자 아이디 및 비밀번호에 줄바꿈이나 캐리지 리턴 문자가 포함되지 않아야 합니다.
- 이름 또는 비밀번호에 공백이 포함된 경우에는 이름 또는 비밀번호 전체를 따옴표로 묶어야 합니다.
- 이름 또는 비밀번호는 최소 한 문자 이상이어야 합니다.
- 명령 셸에서 명령줄에 입력할 수 있는 문자 수가 제한되어 있는 것 외에는 비밀번호 또는 사용자 아이디에 적용되는 길이 제한은 없습니다.

사용자 저장소 채우기 및 관리

저장소에 사용자를 추가하려면 `add` 하위 명령을 사용합니다. 예를 들어, 다음 명령은 기본 브로커 인스턴스 사용자 저장소에 비밀번호가 `sesame`인 사용자 `Katharine`을 추가합니다.

```
imqusermgr add -u Katharine -p sesame -g user
```

저장소에서 사용자를 삭제하려면 `delete` 하위 명령을 사용합니다. 예를 들어, 다음 명령은 `Bob`이라는 사용자를 삭제합니다.

```
imqusermgr delete -u Bob
```

사용자의 비밀번호 또는 상태를 변경하려면 `update` 하위 명령을 사용합니다. 예를 들어, 다음 명령은 `Katharine`의 비밀번호를 `alladin`으로 변경합니다.

```
imqusermgr update -u Katharine -p alladin
```

하나 이상의 사용자에 대한 정보를 나열하려면 `list` 명령을 사용합니다. 다음 명령은 `isa`라는 사용자에 대한 정보를 표시합니다.

```
imqusermgr list -u isa
```

```
% imqusermgr list -u isa

User repository for broker instance: imqbroker
-----
User Name      Group      Active State
-----
isa            admin      true
```

다음 명령은 모든 사용자에 대한 정보를 나열합니다.

```
imqusermgr list
```

```
% imqusermgr list

User repository for broker instance: imqbroker
-----
User Name      Group      Active State
-----
admin          admin      true
guest          anonymous  true
isa            admin      true
testuser1      user       true
testuser2      user       true
testuser3      user       true
testuser4      user       false
testuser5      user       false
```

기본 관리자 비밀번호 변경

보안을 위해 admin의 기본 비밀번호를 혼자만 아는 값으로 변경해야 합니다. 그러려면 imqusermgr 도구를 사용해야 합니다.

다음 명령은 mybroker 브로커 인스턴스의 기본 비밀번호를 grandpoobah로 변경합니다.

```
imqusermgr update -i mybroker -u admin -p grandpoobah
```


브로커 인스턴스가 실행 중일 때 명령줄 도구 중 하나를 실행하여 변경 사항의 적용 여부를 빠르게 확인할 수 있습니다. 예를 들어, 다음 명령이 작동해야 합니다.

```
imqcmd list svc -i mybroker -u admin -p grandpoobah
```

이전 비밀번호 사용이 실패해야 합니다.

비밀번호를 변경한 후에 관리 콘솔을 포함한 Message Queue 관리 도구를 사용하려면 새 비밀번호를 제공해야 합니다.

사용자 저장소에 LDAP 서버 사용

사용자 저장소에 LDAP 서버를 사용하려면 인스턴스 구성 파일에서 특정 브로커 등록 정보를 설정해야 합니다. 이 등록 정보를 사용하면 사용자가 브로커 인스턴스에 연결하거나 특정 메시징 작업을 수행하려 할 때 브로커 인스턴스가 LDAP 서버로부터 사용자 및 그룹에 대한 정보를 쿼리할 수 있습니다. 인스턴스 구성 파일(`config.properties`)은 구성 파일이 연관된 브로커 인스턴스의 이름(`instanceName`)으로 식별되는 디렉토리에 있습니다([부록 A, "Message Queue 데이터의 위치"](#) 참조).

```
.../instances/instanceName/props/config.properties
```

▶ LDAP 서버를 사용하도록 구성 파일을 편집하는 방법

1. 다음 등록 정보를 설정하여 LDAP 사용자 저장소를 사용하고 있음을 지정합니다.

```
imq.authentication.basic.user_repository=ldap
```

2. `imq.authentication.type` 등록 정보를 설정하여 클라이언트에서 브로커로 전달되는 비밀번호를 base64 인코딩(basic) 또는 MD5 다이제스트(digest) 중 어느 쪽으로 할 것인지 결정합니다. 사용자 저장소에 LDAP 디렉토리를 사용하는 경우에는 인증 유형을 basic으로 설정해야 합니다. 예를 들면 다음과 같습니다.

```
imq.authentication.type=basic
```

- LDAP 액세스를 제어하는 브로커 등록 정보도 설정해야 합니다. 브로커의 인스턴스 구성 파일에 저장되는 이러한 등록 정보에 대한 설명은 [표 8-5](#)에 나와 있습니다. Message Queue에서는 JNDI API를 사용하여 LDAP 디렉토리 서버와 통신합니다. 이러한 등록 정보에 사용되는 구문과 용어에 대한 자세한 내용은 JNDI 설명서를 참조하십시오. Message Queue에서는 Sun JNDI LDAP 공급자를 사용하며 단순 인증을 사용합니다.

Message Queue는 LDAP 인증 페일오버를 지원하므로 인증을 시도할 LDAP 디렉토리 서버 목록을 지정할 수 있습니다([표 8-5](#)의 `imq.user.repos.ldap.server` 등록 정보 참조).

표 8-5 LDAP 관련 등록 정보

등록 정보	설명
<code>imq.user_repository.ldap.server</code>	LDAP 서버의 <i>host:port</i> 입니다. <i>host</i> 는 디렉토리 서버를 실행 중인 호스트의 정규화된 DNS 이름을 지정하고 <i>port</i> 는 디렉토리 서버에서 통신에 사용하는 포트 번호를 지정합니다. 페일오버 서버 목록을 지정하려면 다음 구문을 사용합니다. <i>host1:port1 ldap://host2:port2 ldap://host3:port3...</i> 여기서 목록의 항목을 공백으로 구분합니다. 첫 번째 항목 다음의 각 페일오버 서버 주소는 ldap로 시작합니다.
<code>imq.user_repository.ldap.principal</code>	브로커에서 검색할 디렉토리 서버에 바인드할 때 사용하는 고유 이름입니다. 디렉토리 서버에서 익명 검색을 허용하는 경우에는 이 등록 정보에 값을 할당할 필요가 없습니다.
<code>imq.user_repository.ldap.password</code>	브로커에서 사용하는 고유 이름에 연결된 비밀번호입니다. 이 등록 정보는 <code>passfile</code> 에만 지정할 수 있습니다(225페이지 의 " Passfile 사용 " 참조). 여러 방법으로 비밀번호를 제공할 수 있습니다. 가장 안전한 방법은 브로커가 비밀번호를 묻는 프롬프트를 표시하게 하는 것입니다. <code>passfile</code> 을 사용하고 <code>passfile</code> 을 읽기 방지하는 방법은 덜 안전합니다. <code>imqbrokerd -ldappassword</code> 명령줄 옵션을 사용하여 비밀번호를 지정하는 방법이 가장 안전하지 않습니다. 디렉토리 서버가 익명 검색을 허용하는 경우에는 비밀번호가 필요 없습니다.
<code>imq.user_repository.ldap.base</code>	사용자 항목에 사용되는 디렉토리 기반입니다.
<code>imq.user_repository.ldap.uidattr</code>	사용자를 고유하게 식별하는 값을 가진 공급자별 속성 식별자입니다. <code>uid</code> , <code>cn</code> 등이 있습니다.

표 8-5 LDAP 관련 등록 정보(계속)

등록 정보	설명
imq.user_repository. ldap.usrfilter	JNDI 검색 필터(논리식으로 표현된 검색 쿼리)입니다. 사용자에 대해 검색 필터를 지정하면 브로커에서 검색 범위를 좁혀 효율을 높일 수 있습니다. 자세한 내용은 http://java.sun.com/products/jndi/tutorial 에 있는 JNDI 자습서를 참조하십시오. 이 등록 정보는 설정하지 않아도 됩니다.
imq.user_repository. ldap.grpsearch	그룹 검색의 사용 여부를 지정하는 부울 값입니다. 사용자를 그룹에 연결할 수 있는지 확인하려면 LDAP 공급자가 제공하는 설명서를 참조하십시오. Message Queue에서는 중첩 그룹이 지원되지 않습니다. 기본값: false
imq.user_repository. ldap.grpbase	그룹 항목에 사용되는 디렉토리 기반입니다.
imq.user_repository. ldap.gidattr	그룹 이름을 값으로 가진 공급자별 속성 식별자입니다.
imq.user_repository. ldap.memattr	그룹 구성원의 고유 이름을 값으로 가진 그룹 항목 내의 속성 식별자입니다.
imq.user_repository. ldap.grpfilter	JNDI 검색 필터(논리식으로 표현된 검색 쿼리)입니다. 그룹에 대해 검색 필터를 지정하면 브로커에서 검색 범위를 좁혀 효율을 높일 수 있습니다. 자세한 내용은 다음 위치의 JNDI 자습서를 참조하십시오. http://java.sun.com/products/jndi/tutorial 이 등록 정보는 설정하지 않아도 됩니다.
imq.user_repository. ldap.timeout	검색의 시간 제한(초)을 지정하는 정수입니다. 기본적으로 180초로 설정됩니다.
imq.user_repository. ldap.ssl.enabled	브로커가 LDAP 서버와 통신할 때 SSL 프로토콜을 사용해야 하는지 여부를 지정하는 부울 값입니다. 기본적으로 이 값은 false로 설정됩니다.

샘플(기본) LDAP 사용자 저장소의 관련 등록 정보 설정을 보려면 브로커의 default.properties 파일을 참조하십시오.

4. 필요한 경우에는 액세스 제어 등록 정보 파일에서 사용자/그룹과 규칙을 편집해야 합니다. 액세스 제어 등록 정보 파일 사용에 대한 자세한 내용은 [212페이지의 "사용자 권한 부여: 액세스 제어 등록 정보 파일"](#)을 참조하십시오.
5. 연결 인증 및 그룹 검색 중 브로커가 SSL을 통해 LDAP 디렉토리 서버와 통신하도록 하려면 LDAP 서버에서 SSL을 활성화한 후 브로커 구성 파일에서 다음 등록 정보를 설정해야 합니다.
 - LDAP 서버가 SSL 통신에 사용하는 포트를 지정합니다. 예를 들면 다음과 같습니다.

```
imq.user_repository.ldap.server=myhost:7878
```
 - 브로커 등록 정보 `imq.user_repository.ldap.ssl.enabled`를 `true`로 설정합니다.

사용자 권한 부여: 액세스 제어 등록 정보 파일

사용자는 브로커 인스턴스에 연결한 후 메시지를 생성하거나, 대상에서 메시지를 사용하거나, 대기열 대상에서 메시지를 찾아볼 수 있습니다. 사용자가 이러한 작업을 시도하면 브로커는 브로커별 *액세스 제어 등록 정보 파일*(ACL 파일)을 검토하여 해당 사용자가 작업을 수행할 권한이 있는지 확인합니다. ACL 파일에는 특정 사용자(또는 사용자 그룹)가 어떤 작업을 수행할 수 있는지를 지정하는 규칙이 포함되어 있습니다. 기본적으로 인증된 모든 사용자는 모든 대상에서 메시지를 생성하고 사용할 수 있습니다. ACL 파일을 편집하여 이런 작업을 특정 사용자 및 그룹으로 제한할 수 있습니다.

ACL 파일은 사용자 정보가 플랫폼 파일 사용자 저장소([202페이지의 "플랫폼 파일 사용자 저장소 사용"](#) 참조)에 있는지 LDAP 사용자 저장소([209페이지의 "사용자 저장소에 LDAP 서버 사용"](#) 참조)에 있는지와 관계 없이 사용됩니다.

액세스 제어 등록 정보 파일 작성

ACL 파일은 인스턴스에 고유합니다. 기본 파일(`accesscontrol.properties`)이 시작하는 각 브로커 인스턴스에 대해 만들어집니다. 이 ACL 등록 정보 파일은 ACL 파일이 연결된 브로커 인스턴스의 이름(`instanceName`)으로 식별되는 디렉토리에 저장됩니다(부록 A, "Message Queue 데이터의 위치" 참조).

```
.../instances/instanceName/etc/accesscontrol.properties
```

ACL 파일은 Java 등록 정보 파일과 같은 형식으로 구성됩니다. 먼저 파일 버전을 정의하는 것으로 시작하여 다음 세 개 섹션에서 액세스 제어 규칙을 지정합니다.

- 연결 액세스 제어
- 대상 액세스 제어
- 대상 자동 작성 액세스 제어

version 등록 정보는 ACL 등록 정보 파일의 버전을 정의하며 이 항목은 변경할 수 없습니다.

```
version=JMQFileAccessControlModel/100
```

액세스 규칙의 기본 구문, 권한을 계산하는 방법 및 ACL 파일에서 액세스 제어를 지정하는 세 개 섹션에 대한 설명이 아래에 나와 있습니다.

액세스 규칙 구문

ACL 등록 정보 파일에서 액세스 제어는 대상 및 연결 서비스와 같이 보호된 자원에 대해 특정 사용자 또는 그룹이 갖는 액세스를 정의합니다. 액세스 제어는 규칙 또는 규칙 집합으로 나타내며 각 규칙은 Java 등록 정보로 표시됩니다.

이 규칙의 기본 구문은 다음과 같습니다.

```
resourceType.resourceVariant.operation.access.principalType = principals
```

표 8-6에서는 구문 규칙의 요소에 대해 설명합니다.

표 8-6 액세스 규칙의 구문 요소

요소	설명
<i>resourceType</i>	connection, queue 또는 topic 중 하나입니다.
<i>resourceVariant</i>	<i>resourceType</i> 에서 지정한 유형의 인스턴스입니다. 예를 들면 <code>myQueue</code> 가 있습니다. 와일드카드 문자(*)를 사용하여 모든 연결 서비스 유형 또는 모든 대상을 나타낼 수 있습니다.
<i>operation</i>	값은 표현할 액세스 규칙의 종류에 따라 달라집니다.
<i>access</i>	allow 또는 deny 중 하나입니다.
<i>principalType</i>	user 또는 group 중 하나입니다. 자세한 내용은 205페이지의 "그룹"을 참조하십시오.
<i>principals</i>	규칙의 왼쪽에 지정한 액세스를 갖는 사람입니다. <i>principalType</i> 이 user이면 개별 사용자 또는 사용자 목록(쉼표로 구분)이 될 수 있고, <i>principalType</i> 이 group이면 한 그룹 또는 그룹 목록(쉼표로 구분)이 될 수 있습니다. 와일드카드 문자(*)를 사용하여 모든 사용자 또는 모든 그룹을 나타낼 수 있습니다.

다음은 액세스 규칙의 몇 가지 예입니다.

- 다음 규칙은 모든 사용자가 q1이라는 이름의 대기열에 메시지를 보낼 수 있다는 것을 나타냅니다.
`queue.q1.produce.allow.user=*`
- 다음 규칙은 모든 사용자가 모든 대기열에 메시지를 보낼 수 있다는 것을 나타냅니다.
`queue.*.produce.allow.user=*`

주 ASCII가 아닌 사용자, 그룹 또는 대상 이름을 지정하려면 유니코드 제어 (\uXXXX) 표기를 사용해야 합니다. ACL 파일을 편집한 후 ASCII 인코딩이 아닌 이름으로 저장한 경우에는 Java native2ascii 도구를 사용해서 해당 파일을 ASCII로 변환할 수 있습니다. 자세한 내용은 <http://java.sun.com/j2se/1.4/docs/guide/intl/faq.html>을 참조하십시오.

권한 계산

일련의 규칙으로 나타내는 권한을 계산할 때에는 다음과 같은 원리가 적용됩니다.

- 구체적인 액세스 규칙은 일반적인 액세스 규칙을 대체합니다. 다음 두 규칙을 적용하고 나면 모두가 모든 대기열에 전송할 수 있지만 Bob은 tq1에 전송할 수 없습니다.

```
queue.*.produce.allow.user=*
```

```
queue.tq1.produce.deny.user=Bob
```

- 명시된 *principal*로 지정한 액세스는 * *principal*로 지정한 액세스를 대체합니다. 다음 규칙은 tq1에서 메시지를 생성할 권한을 Bob에게는 거부하지만 다른 모든 사람에게 허용합니다.

```
queue.tq1.produce.allow.user=*
```

```
queue.tq1.produce.deny.user=Bob
```

- 사용자에 대한 * *principal* 규칙은 그룹에 대한 해당 * *principal*을 대체합니다. 예를 들어, 다음 두 규칙은 인증된 모든 사용자가 tq1에 메시지를 보낼 수 있도록 허용합니다.

```
queue.tq1.produce.allow.user=*
```

```
queue.tq1.produce.deny.group=*
```

- 사용자에게 부여된 액세스 권한은 사용자 그룹에 부여된 액세스 권한을 대체합니다. 다음 예에서 Bob이 User의 구성원이라면 Bob은 tq1에 대해 메시지를 생성할 권한이 거부되지만, User의 다른 모든 구성원은 메시지를 생성할 수 있습니다.

```
queue.tq1.produce.allow.group=User
```

```
queue.tq1.produce.deny.user=Bob
```

- 액세스 규칙을 통해 명확하게 부여되지 않은 모든 액세스 권한은 거부됩니다. 예를 들어, ACL 파일에 액세스 규칙이 없으면 모든 사용자의 모든 작업이 거부됩니다.

- 같은 사용자 또는 그룹에 권한을 거부하고 허용하면 설정이 서로 상쇄됩니다. 예를 들어, 다음과 같은 두 규칙이 있으면 Bob은 q1을 찾아볼 수 없습니다.

```
queue.q1.browse.allow.user=Bob
```

```
queue.q1.browse.deny.user=Bob
```

다음과 같은 두 규칙이 있으면 User 그룹은 q5에서 메시지를 사용할 수 없습니다.

```
queue.q5.consume.allow.group=User
```

```
queue.q5.consume.deny.group=User
```

- 왼쪽의 같은 규칙이 여러 개 있는 경우에는 마지막 항목만 적용됩니다.

연결 액세스 제어

ACL 등록 정보 파일의 연결 액세스 제어 섹션에는 브로커의 연결 서비스에 대한 액세스 제어 규칙이 있습니다. 연결 액세스 제어 규칙의 구문은 다음과 같습니다.

```
connection.resourceVariant.access.principalType = principals
```

*resourceVariant*에 정의할 수 있는 값은 NORMAL과 ADMIN 두 가지입니다. 기본적으로 NORMAL 유형에는 모든 사용자가 액세스할 수 있지만 ADMIN 유형의 연결 서비스에는 그룹이 admin인 사용자만 액세스할 수 있습니다.

연결 액세스 제어 규칙을 편집하여 사용자의 연결 액세스 권한을 제한할 수 있습니다. 예를 들어, 다음 규칙에서는 NORMAL에 대한 Bob의 액세스를 거부하지만 다른 모든 사람은 허용합니다.

```
connection.NORMAL.deny.user=Bob
```

```
connection.NORMAL.allow.user=*
```

별표(*) 문자를 사용해서 인증된 모든 사용자 또는 그룹을 지정할 수 있습니다.

서비스 유형을 직접 만들 수는 없으며 NORMAL과 ADMIN 상수에서 지정하는 사전 정의된 유형만 사용할 수 있습니다.

대상 액세스 제어

액세스 제어 등록 정보 파일의 대상 액세스 제어 섹션에는 대상 기반의 액세스 제어 규칙이 있습니다. 이 규칙은 누가(사용자/그룹) 어디에서(대상) 무엇을(작업) 할 수 있는지 결정합니다. 이 규칙으로 규제되는 액세스 유형에는 대기열로 메시지 보내기, 주제에 메시지 게시, 대기열에서 메시지 받기, 주제에 가입, 대기열에서 메시지 찾아보기가 포함됩니다.

기본적으로 모든 사용자 또는 그룹은 모든 대상에 대해 모든 유형의 액세스를 갖습니다. 특정 대상 액세스 규칙을 추가하거나 기본 규칙을 편집할 수 있습니다. 이 절의 나머지 부분에서는 규칙을 직접 작성하기 위해 알아야 하는 대상 액세스 규칙의 구문에 대해 설명합니다.

대상 규칙의 구문은 다음과 같습니다.

```
resourceType.resourceVariant.operation.access.principalType = principals
```

표 8-7에서는 이러한 요소에 대해 설명합니다.

표 8-7 대상 액세스 제어 규칙의 요소

구성 요소	설명
<i>resourceType</i>	queue 또는 topic 중 하나여야 합니다.
<i>resourceVariant</i>	대상 이름 또는 모든 대기열이나 모든 주제를 나타내는 모든 대상(*)입니다.
<i>operation</i>	produce, consume 또는 browse 중 하나여야 합니다.
<i>access</i>	allow 또는 deny 중 하나여야 합니다.
<i>principalType</i>	user 또는 group 중 하나여야 합니다.

하나 이상의 사용자 또는 하나 이상의 그룹에 액세스를 허용할 수 있습니다.

다음 예에서는 여러 종류의 대상 액세스 제어 규칙을 보여 줍니다.

- 모든 사용자가 모든 대기열 대상에 메시지를 보낼 수 있도록 허용합니다.
queue.*.produce.allow.user=*
- user 그룹의 어떤 구성원도 Admissions 주제에 가입하지 못하게 합니다.
topic.Admissions.consume.deny.group=user

대상 자동 작성 액세스 제어

ACL 등록 정보 파일의 마지막 섹션에는 브로커에서 대상을 자동 작성하는 사용자 또는 그룹을 지정하는 액세스 규칙이 있습니다.

사용자가 아직 존재하지 않는 대상에서 공급자 또는 메시지 사용자를 만든 경우, 브로커의 자동 작성 등록 정보가 활성화되어 있고 물리적 대상이 아직 존재하지 않으면 브로커가 해당 대상을 만듭니다.

기본적으로 모든 사용자 또는 그룹은 브로커가 대상을 자동 작성하도록 할 수 있습니다. 이 권한은 다음과 같은 규칙으로 지정합니다.

```
queue.create.allow.user=*
topic.create.allow.user=*
```

ACL 파일을 편집하여 이 유형의 액세스를 제한할 수도 있습니다.

대상 자동 작성 액세스 규칙의 일반 구문은 다음과 같습니다.

```
resourceType.create.access.principalType = principals
```

여기서 *resourceType*은 queue 또는 topic입니다.

예를 들어, 다음 규칙은 브로커가 Snoopy를 제외한 모든 사람에 대해 주제 대상을 자동 작성하도록 허용합니다.

```
topic.create.allow.user=*
topic.create.deny.user=Snoopy
```

대상 자동 작성 규칙의 효과와 대상 액세스 규칙의 효과 사이에 모순이 없도록 해야 합니다. 예를 들어, 1) 어떤 사용자도 대상으로 메시지를 보낼 수 없도록 대상 액세스 규칙을 변경했는데 2) 대상의 자동 작성은 허용했다면, 브로커에서는 대상이 없는 경우 대상을 *만들기는 하지만* 메시지를 그 대상으로 전달하지는 *않습니다*.

암호화: SSL 기반 서비스를 사용한 작업(엔터프라이즈판)

Message Queue 엔터프라이즈판은 Secure Socket Layer (SSL) 표준에 따라 TCP/IP (ssljms 및 ssladmin)와 HTTP (httpsjms)를 통해 연결 서비스를 지원합니다. 이러한 SSL 기반 연결 서비스를 사용하면 클라이언트와 브로커 사이에서 보내는 메시지를 암호화할 수 있습니다. 현재 Message Queue 릴리스는 자체 서명된 서버 인증서를 기반으로 한 SSL 암호화를 지원합니다.

SSL 기반 연결 서비스를 사용하려면 키 도구 유틸리티(imqkeytool)를 사용해서 개인 키/공용 키 쌍을 생성해야 합니다. 이 유틸리티는 브로커에 대해 연결을 요청하는 모든 클라이언트로 전달되는, 자체 서명된 인증서에 공용 키를 포함합니다. 그러면 클라이언트에서 이 인증서를 사용하여 암호화된 연결을 설정합니다.

Message Queue의 SSL 기반 연결 서비스가 개념은 비슷하지만, 설정 방법에는 몇 가지 차이점이 있습니다. 따라서 TCP/IP와 HTTP에서의 보안 연결에 대한 내용은 뒤의 절에서 별도로 설명합니다.

TCP/IP에서 SSL 기반 서비스 설정

TCP/IP에서 직접 보안 연결을 제공하는 SSL 기반 연결 서비스에는 세 가지가 있습니다.

ssljms. 이 연결 서비스는 클라이언트와 브로커 간에 암호화된 보안 연결을 통해 메시지를 전달할 때 사용됩니다.

ssladmin. 이 연결 서비스는 명령줄 관리 도구인 Message Queue 명령 유틸리티(imqcmd)와 브로커 간에 암호화된 보안 연결을 만들 때 사용됩니다. 관리 콘솔(imqadmin)에서는 보안 연결이 지원되지 않습니다.

클러스터. 이 연결 서비스는 클러스터의 브로커 간에 암호화된 보안 연결을 통해 메시지를 전달할 때 사용됩니다(143페이지의 "[브로커간 연결 보안](#)" 참조).

▶ SSL 기반 연결 서비스를 설정하는 방법

1. 자체 서명된 인증서를 생성합니다.
2. 브로커에서 ssljms 연결 서비스, ssladmin 연결 서비스 또는 클러스터 연결 서비스를 활성화합니다.
3. 브로커를 시작합니다.
4. 클라이언트를 구성하고 실행합니다(ssljms 연결 서비스에만 적용).

ssljms와 ssladmin 연결 서비스를 설정하는 절차는 4단계의 클라이언트 구성 및 실행을 제외하면 서로 같습니다.

각 단계에 대해서는 뒤에서 자세히 설명합니다.

1단계. 자체 서명된 인증서 생성

Message Queue의 SSL 지원은 클라이언트가 알려지고 신뢰할 수 있는 서버와 통신한다는 가정 하에 전송 데이터를 보호하기 위한 것입니다. 따라서 자체 서명된 인증서만을 사용해서 SSL이 구현됩니다.

imqkeytool 명령을 실행하여 브로커에 자체 서명된 인증서를 생성합니다. ssljms 연결 서비스, ssladmin 연결 서비스 또는 클러스터 연결 서비스에 같은 인증서를 사용할 수 있습니다. 명령 프롬프트에서 다음을 입력합니다.

```
imqkeytool -broker
```

유틸리티는 필요한 정보를 묻는 메시지를 표시합니다(Unix 시스템에서 키 저장소를 만들 권한을 가지려면 `imqkeytool`을 슈퍼유저(`root`)로 실행해야 할 수 있음).

`imqkeytool`은 키 저장소 비밀번호, 일부 조직 정보 및 사용자의 확인을 묻는 메시지를 차례로 표시합니다. 확인이 끝나면 키 쌍을 생성하는 동안 일시 중지됩니다. 그런 다음 특정 키 쌍을 잠금 비밀번호(키 비밀번호)를 묻습니다. 이 프롬프트에 대한 응답으로 **Return** 키를 눌러야 합니다. 그러면 키 비밀번호가 키 저장소 비밀번호와 동일하게 지정됩니다.

주 입력한 비밀번호를 기억해야 합니다. 나중에 브로커를 시작할 때 이 비밀번호를 제공해야 브로커가 키 저장소를 열 수 있습니다. 키 저장소 비밀번호를 `passfile`에 저장할 수도 있습니다(225페이지의 "[Passfile 사용](#)" 참조).

`imqkeytool`을 실행하면 `JDK keytool` 유틸리티가 실행되어 자체 서명된 인증서를 생성하고 생성된 인증서를 **부록 A, "Message Queue 데이터의 위치"**와 같이 운영 체제에 따라 다른 디렉토리에 있는 `Message Queue`의 키 저장소에 넣습니다.

키 저장소의 형식은 `JDK1.2 keytool` 유틸리티에서 지원하는 것과 같습니다.

`Message Queue` 키 저장소의 구성 가능한 등록 정보는 **표 8-8**에 나와 있습니다(등록 정보 구성 지침은 **5장, "브로커 시작 및 구성"** 참조).

표 8-8 키 저장소 등록 정보

등록 정보 이름	설명
<code>imq.keystore.file.dirpath</code>	SSL 기반 서비스의 경우: 키 저장소 파일이 있는 디렉토리의 경로를 지정합니다. 기본값: 부록 A, "Message Queue 데이터의 위치" 참조
<code>imq.keystore.file.name</code>	SSL 기반 서비스의 경우: 키 저장소 파일의 이름을 지정합니다. 기본값: <code>keystore</code>

표 8-8 키 저장소 등록 정보(계속)

등록 정보 이름	설명
<code>imq.keystore.password</code>	<p>SSL 기반 서비스의 경우: 키 저장소 비밀번호를 지정합니다. 이 등록 정보는 <code>passfile</code>에만 지정될 수 있습니다(225페이지의 "Passfile 사용" 참조).</p> <p>다양한 방법으로 비밀번호를 제공할 수 있습니다. 가장 보안된 방법은 브로커가 사용자에게 비밀번호를 묻게 하는 것입니다. <code>passfile</code>을 사용하고 <code>passfile</code>을 읽기 보호하는 방법은 보안 수준이 낮습니다. 보안 수준이 가장 낮은 방법은 <code>imqbrokerd -ldappassword</code> 명령줄 옵션을 사용하여 비밀번호를 지정하는 것입니다.</p>

문제를 해결하기 위해 키 쌍을 다시 생성해야 할 수도 있습니다. 예를 들면 다음과 같습니다.

- 키 저장소 비밀번호를 잊은 경우
- 브로커를 시작할 때 SSL 기반 서비스를 초기화할 수 없고 다음과 같은 예외가 발생하는 경우
`java.security.UnrecoverableKeyException: Cannot recover key.`

이 예외는 [219페이지의 "1단계. 자체 서명된 인증서 생성"](#)에서 자체 서명된 인증서를 생성할 때 지정한 키 저장소 비밀번호와 입력한 키 저장소 비밀번호가 다른 경우에 발생할 수 있습니다.

▶ 키 쌍을 다시 생성하는 방법

1. 부록 A, "[Message Queue 데이터의 위치](#)"에 나와 있는 위치에 있는 브로커의 키 저장소를 제거합니다.
2. `imqkeytool`을 다시 실행하여 [219페이지의 "1단계. 자체 서명된 인증서 생성"](#)의 설명과 같이 키 쌍을 생성합니다.

2단계. 브로커에서 SSL 기반 서비스 활성화

브로커에서 SSL 기반 서비스를 활성화하려면 `imq.service.activelist` 등록 정보에 `ssljms` 또는 `ssladmin`을 추가해야 합니다.

주 SSL 기반 클러스터 연결 서비스는 `imq.service.activelist` 등록 정보 대신 `imq.cluster.transport` 등록 정보를 사용하여 활성화합니다. [143페이지의 "브로커간 연결 보안"](#)을 참조하십시오.

▶ 브로커에서 SSL 기반 서비스를 활성화하는 방법

1. 브로커의 인스턴스 구성 파일을 엽니다.

인스턴스 구성 파일은 구성 파일이 연관된 브로커 인스턴스의 이름(*instanceName*)으로 식별되는 디렉토리에 있습니다(부록 A, "Message Queue 데이터의 위치" 참조).

```
.../instances/instanceName/props/config.properties
```

2. `imq.service.activelist` 등록 정보에 항목을 추가하고(항목이 아직 없는 경우) 목록에 SSL 기반 서비스를 포함시킵니다.

기본적으로 이 등록 정보에는 `jms` 및 `admin` 연결 서비스가 포함됩니다. 활성화할 서비스에 따라 `ssljms` 및/또는 `ssladmin` 연결 서비스를 추가해야 합니다.

```
imq.service.activelist=jms,admin,ssljms,ssladmin
```

3단계. 브로커 시작

브로커를 시작하며 키 저장소 비밀번호를 제공합니다. 비밀번호는 다음 중 한 가지 방법으로 제공할 수 있습니다.

- 브로커를 시작할 때 비밀번호를 묻는 프롬프트가 표시되게 합니다.

```
imqbrokerd
```

```
Please enter Keystore password: mypassword
```

- `imqbrokerd` 명령에 `-password` 옵션을 사용합니다.

```
imqbrokerd -password mypassword
```

- 브로커를 시작할 때 액세스하는 `passfile` 파일(225페이지의 "Passfile 사용" 참조)에 비밀번호를 입력합니다. 먼저 다음과 같은 브로커 구성 등록 정보를 설정해야 합니다(129페이지의 "인스턴스 구성 파일 편집" 참조).

```
imq.passfile.enabled=true
```

이 등록 정보가 설정되면 두 가지 중 한 가지 방법으로 `passfile`에 액세스할 수 있습니다.

- `passfile`의 위치를 `imqbrokerd` 명령에 전달합니다.

```
imqbrokerd -passfile /tmp/mypassfile
```

- `-passfile` 옵션 없이 브로커를 시작하되, 다음의 두 브로커 구성 등록 정보를 사용해서 `passfile`의 위치를 지정합니다.

```
imq.passfile.dirpath=/tmp
```

```
imq.passfile.name=mypassfile
```

passfile 관련 브로커 등록 정보의 목록을 보려면 [69페이지의 표 2-6](#)을 참조하십시오.

SSL을 사용하는 브로커나 클라이언트를 시작할 때 몇 초간 CPU 사이클을 과도하게 사용하는 경우가 있을 수 있습니다. 이것은 Message Queue에서 SSL 구현에 JSSE (Java Secure Socket Extension)를 사용하기 때문입니다. JSSE에서는 `java.security.SecureRandom()`을 사용하여 난수를 생성합니다. 이 방법은 초기 난수 시드를 만드는 동안 상당한 시간을 소비하기 때문에 CPU 사용량이 많아지게 됩니다. 시드를 만들고 나면 CPU 수준이 정상으로 돌아옵니다.

4단계. SSL기반 클라이언트 구성 및 실행

마지막으로 보안 연결 서비스를 사용하도록 클라이언트를 구성해야 합니다. 클라이언트의 종류에는 사용하는 연결 서비스에 따라 `ssljms`를 사용하는 응용 프로그램 클라이언트와 `ssladmin`을 사용하는 Message Queue 관리 클라이언트(예: `imqcmd`)의 두 가지가 있습니다. 다음 절에서 이 두 가지에 대해 별도로 설명합니다.

응용 프로그램 클라이언트

클라이언트의 클래스 경로에 필요한 Secure Socket Extension (JSSE) jar 파일이 있는지 확인하고 `ssljms` 연결 서비스를 사용하도록 지정해야 합니다.

1. 클라이언트에서 (JSSE 및 JNDI 지원이 기본적으로 제공되는) J2SDK1.4를 사용하지 않는 경우에는 클라이언트의 클래스 경로에 다음 jar 파일이 있는지 확인합니다.

```
jsse.jar, jnet.jar, jcert.jar, jndi.jar
```

2. 클라이언트의 클래스 경로에 다음 Message Queue jar 파일이 있는지 확인합니다.

```
imq.jar, jms.jar
```

3. 클라이언트를 시작하고 브로커의 `ssljms` 서비스에 연결합니다. 이 작업을 수행하는 방법 중 하나는 다음과 같은 명령을 입력하는 것입니다.

```
java -DimqConnectionType=TLS clientAppName
```

`imqConnectionType`을 설정하면 연결에서 SSL을 사용하게 됩니다.

클라이언트 응용프로그램에서의 `ssljms` 연결 서비스 사용에 대한 자세한 내용은 *Message Queue Java Client Developer's Guide*의 관리 대상 객체 사용에 대한 장을 참조하십시오.

관리 클라이언트(imqcmd)

`imqcmd`를 사용할 때 `-secure` 옵션을 포함하면 보안 관리 연결을 설정할 수 있습니다([154페이지의 표 6-2](#) 참조). 예를 들면 다음과 같습니다.

```
imqcmd list svc -b hostName:port -u adminName -p adminPassword -secure
```

여기서 *adminName*과 *adminPassword*는 Message Queue 사용자 저장소의 유효한 항목입니다(플랫 파일 저장소를 사용하는 경우 [208페이지](#)의 "기본 관리자 비밀번호 변경" 참조).

연결 서비스를 나열하는 것은 `ssladmin` 서비스가 실행 중이며 다음 출력과 같이 보안 관리 연결을 성공적으로 설정할 수 있다는 것을 보여주는 방법입니다.

```
Listing all the services on the broker specified by:

Host                Primary Port
localhost           7676

Service Name      Port Number      Service State
admin              33984 (dynamic)  RUNNING
httpjms            -                 UNKNOWN
httpsjms           -                 UNKNOWN
jms                33983 (dynamic)  RUNNING
ssladmin           35988 (dynamic)  RUNNING
ssljms             dynamic           UNKNOWN

Successfully listed services.
```

HTTP에서 SSL 서비스 설정

이 SSL 기반 연결 서비스(`httpsjms`)에서 클라이언트와 브로커는 HTTPS 터널 서블릿을 통해 보안 연결을 설정합니다. HTTPS 지원의 구조와 구현에 대한 설명은 [307페이지](#)의 **부록 C**, "HTTP/HTTPS 지원(엔터프라이즈판)"을 참조하십시오.

Passfile 사용

필요한 비밀번호를 묻는 프롬프트를 표시하거나 `imqbrokerd` 명령에 해당 비밀번호를 제공하지 않고 브로커를 시작하려면 필요한 비밀번호를 *passfile*에 넣습니다. 그런 다음 브로커를 시작할 때 `-passfile` 옵션을 사용하여 이 *passfile*을 지정할 수 있습니다.

```
imqbrokerd -passfile myPassfile
```

*passfile*은 비밀번호를 포함하는 단순 텍스트 파일입니다. 이 파일은 암호화되지 않기 때문에 비밀번호를 수동으로 제공하는 것보다 덜 안전합니다. 하지만 파일에 대한 권한 설정을 통해 해당 파일을 볼 수 있는 권한을 제한할 수 있습니다. *passfile*의 권한은 브로커를 시작하는 사용자가 이 파일을 읽을 수 있도록 설정해야 합니다.

*passfile*에는 표 8-9에 있는 것과 같은 비밀번호를 포함할 수 있습니다.

표 8-9 Passfile의 비밀번호

비밀번호	설명
<code>imq.keystore.password</code>	SSL 기반 서비스의 키 저장소 비밀번호를 지정합니다(220 페이지의 표 8-8 참조).
<code>imq.user_repository.ldap.password</code>	구성된 LDAP 사용자 저장소에 바인드할 수 있도록 브로커에 할당된 고유 이름에 연결되는 비밀번호를 지정합니다(210 페이지의 표 8-5 참조).
<code>imq.persist.jdbc.password</code>	필요한 경우 데이터베이스 연결을 여는 비밀번호를 지정합니다(300페이지의 표 B-1 참조).

부록 A, "Message Queue 데이터의 위치"에 표시된 것처럼 운영 체제에 따라 다른 위치에 샘플 *passfile*이 있습니다.

Passfile 사용

메시지 서비스 분석 및 조정

이 장에서는 메시징 응용 프로그램의 성능을 최적화하기 위해 Message Queue 서비스를 분석하고 조정하는 방법에 대한 여러 항목을 다룹니다. 이 장은 다음 항목으로 구성되어 있습니다.

- 성능 정보
- 성능에 영향을 미치는 요소
- 메시지 서버 모니터링
- 성능 문제 해결
- 성능 향상을 위한 구성 조정

성능 정보

성능 조정 프로세스

메시징 응용 프로그램의 성능은 응용 프로그램과 Message Queue 서비스 사이의 상호 작용에 따라 달라집니다. 따라서 성능을 최대화하려면 응용 프로그램 개발자와 관리자가 함께 노력해야 합니다.

성능 최적화 프로세스는 응용 프로그램 설계에서 시작되어 응용 프로그램 배포 이후의 메시지 서비스 조정에 이르기까지 계속됩니다. 성능 조정 프로세스에는 다음 단계가 포함됩니다.

- 응용 프로그램의 성능 요구 사항 정의

- 성능에 영향을 미치는 요소(특히 안정성과 성능 간의 균형)를 고려하여 응용 프로그램 설계
- 기본 성능 측정 설정
- 성능 최적화를 위해 메시지 서비스 조정 또는 재구성

위에서 설명한 프로세스는 자주 반복됩니다. 응용 프로그램 배포 중에 Message Queue 관리자는 메시지 서버가 응용 프로그램의 일반 성능 요구 사항에 적합한지 평가합니다. 벤치마크 테스트가 이러한 요구 사항을 충족시키는 경우 관리자는 이 장에서 설명한 대로 시스템을 조정할 수 있습니다. 하지만 벤치마크 테스트가 성능 요구 사항을 충족시키지 못하는 경우 응용 프로그램을 재설계하거나 배포 구조를 수정해야 합니다.

성능 요소

일반적으로 성능은 메시지 서비스가 생성자의 메시지를 사용자에게 전달하는 속도와 효율성에 대한 측정입니다. 하지만 사용자의 필요에 따라 중요할 수 있는 여러 다른 성능 요소가 있습니다.

연결 로드. 메시지 생성자나 메시지 사용자 수 또는 시스템이 지원할 수 있는 동시 연결 수입니다.

메시지 처리량. 메시징 시스템을 통해 전달될 수 있는 초당 메시지 수 또는 메시지 바이트입니다.

대기 시간. 메시지 생성자로부터 메시지 사용자에게 특정 메시지를 전달하는 데 걸리는 시간입니다.

안정성. 메시지 서비스의 전체적인 가용성 또는 로드량이 많거나 장애가 발생하는 경우 메시지 서비스 성능이 유연하게 감소되는 정도입니다.

효율성. 메시지 전달의 효율성, 즉 사용된 계산 자원을 기준으로 메시지 처리량을 측정한 것입니다.

이러한 성능의 여러 요소들은 일반적으로 상호 연관됩니다. 메시지 처리량이 높으면 메시지가 메시지 서버에서 백로그될 가능성이 적어지므로 대기 시간이 낮아집니다(단일 메시지가 매우 빠르게 전달될 수 있음). 하지만 대기 시간은 통신 연결 속도, 메시지 서버 처리 속도, 클라이언트 처리 속도 등 많은 요소에 따라 달라질 수 있습니다.

어느 경우든 여러 다른 성능 요소가 있습니다. 어떤 성능 요소가 가장 중요한지는 일반적으로 특정 응용 프로그램의 요구 사항에 따라 달라집니다.

벤치마크

벤치마크는 메시징 응용 프로그램을 위한 테스트 프로그램을 만들고 이 테스트 프로그램의 메시지 처리량이나 기타 성능 요소를 측정하는 프로세스입니다.

예를 들어, 일정 수의 생성자 클라이언트가 일정 수의 연결, 세션 및 메시지 생성자를 사용하여 메시징 응용 프로그램 설계에 따라 일정 수의 대기열이나 주제에 표준 크기의 지속성 또는 비지속성 메시지를 지정된 속도로 보내는 테스트 프로그램을 만들 수 있습니다. 마찬가지로 테스트 프로그램에는 일정 수의 사용자 클라이언트가 포함되며, 이 사용자 클라이언트는 일정 수의 연결, 세션 및 특정 확인 모드로 테스트 프로그램의 대상에서 메시지를 사용하는 (특정 유형의) 메시지 사용자를 사용합니다.

표준 테스트 프로그램을 사용하면 메시지 생성에서 사용까지 걸리는 시간이나 평균 메시지 처리 속도를 측정할 수 있으며 시스템을 모니터링하여 연결 스레드 사용, 메시지 저장소 데이터, 메시지 흐름 데이터 및 기타 관련 메트릭을 살펴볼 수 있습니다. 그런 다음 성능에 부정적인 영향을 미칠 때까지 메시지 생성 속도, 메시지 생성자 수 또는 기타 변수를 증가시켜볼 수 있습니다. 달성할 수 있는 최대 처리량이 메시지 서비스 구성에 대한 벤치마크입니다.

이 벤치마크를 사용하여 테스트 프로그램의 일부 특성을 수정할 수 있습니다. 성능에 영향을 미칠 수 있는 모든 요소([232페이지의 "성능에 영향을 미치는 응용 프로그램 설계 요소"](#) 참조)를 조심스럽게 제어하여 이러한 요소들 중 일부를 변경했을 때 벤치마크에 미치는 영향을 살펴볼 수 있습니다. 예를 들어, 연결 수나 메시지 크기를 5배나 10배 증가시켜 성능에 미치는 영향을 살펴볼 수 있습니다.

이와 반대로 응용 프로그램 기반 요소를 일정하게 유지하면서 제어 가능한 방식으로 브로커 구성을 변경하고(예: 연결 등록 정보, 스레드 풀 등록 정보, JVM 메모리 제한, 제한 동작, 기본 제공 지속성 대 플러그 인 지속성 등을 변경) 이러한 변경이 성능에 미치는 영향을 살펴볼 수 있습니다.

이러한 응용 프로그램 벤치마크는 메시지 서비스를 조정하여 배포된 응용 프로그램의 성능을 높이고자 할 때 유용한 정보를 제공합니다. 벤치마크를 사용하면 한 가지 변경 사항이나 일련의 변경 사항이 미치는 영향을 좀 더 정확하게 예상할 수 있습니다.

일반적으로 벤치마크는 제어된 테스트 환경에서 메시지 서비스가 안정될 정도의 충분한 기간 동안 실행해야 합니다. (Java 코드를 기계 코드로 변환하는 JIT (Just-In-Time) 컴파일 에 의해 시작 시 성능에 부정적인 영향을 미칩니다.)

기본 사용 패턴

일단 메시징 응용 프로그램이 배포되어 실행 중이면 기본 사용 패턴을 설정하는 것이 중요합니다. 최대 수요가 발생하는 시기를 알아 해당 수요를 수량화할 수 있습니다. 예를 들어, 수요는 일반적으로 최종 사용자 수, 활동 수준, 시간 또는 이러한 모든 것에 의해 변동됩니다.

기본 사용 패턴을 설정하려면 메시지 서버를 오랫동안 모니터링하여 연결 수, 브로커(또는 특정 대상)에 저장되어 있는 메시지 수, 브로커(또는 특정 대상)에 들어오고 나가는 메시지 흐름, 활성 사용자의 수 등의 데이터를 확인해야 합니다. 메트릭 데이터에서 제공하는 평균 값과 최대 값을 사용할 수도 있습니다.

이러한 기본 메트릭을 설계 예측과 비교하여 확인하는 것이 중요합니다. 이렇게 함으로써 클라이언트 코드가 제대로 작동하는지 확인할 수 있습니다. 예를 들어, 연결이 열려 있는 상태로 남아 있지 않은지 또는 사용된 메시지가 확인되지 않은 상태로 남아 있지 않은지 확인할 수 있습니다. 이러한 코딩 오류는 메시지 서버 자원을 사용하므로 성능에 상당한 영향을 미칠 수 있습니다.

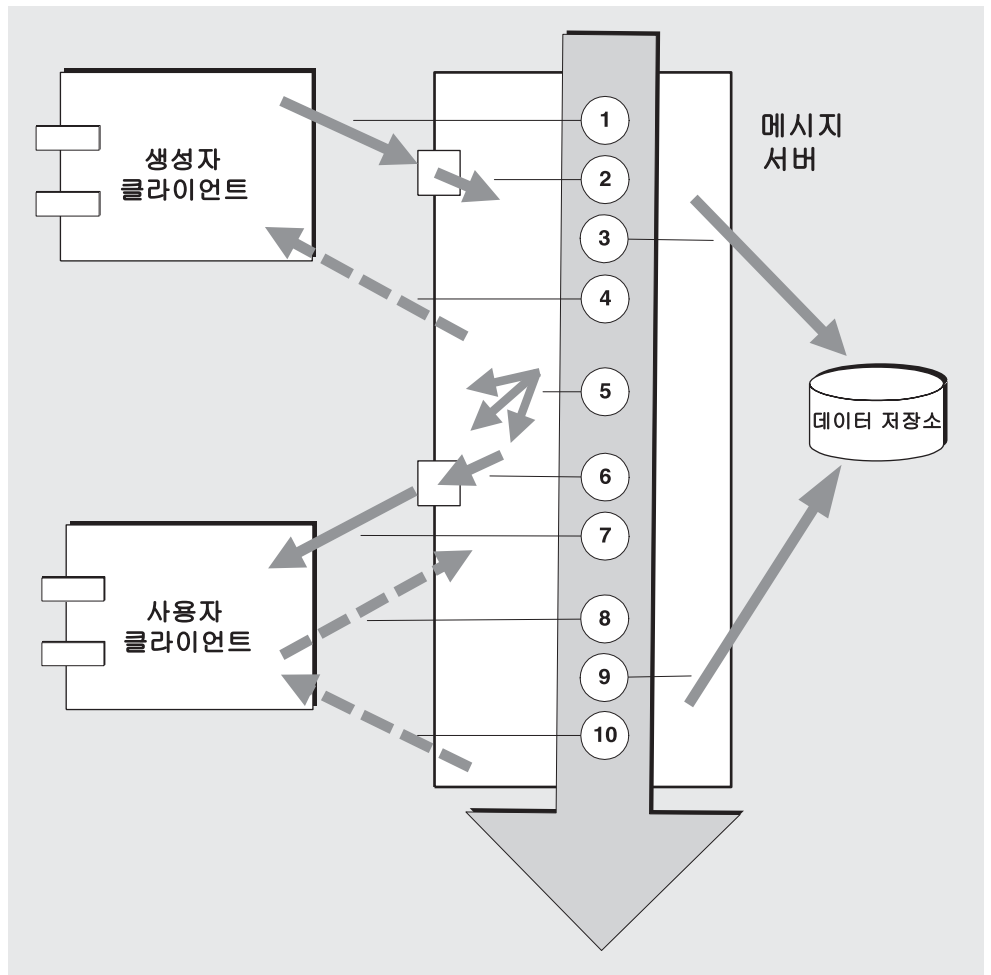
기본 사용 패턴을 통해 최적 성능을 위해 시스템을 조정하는 방법을 결정할 수 있습니다. 예를 들어, 한 대상이 다른 대상에 비해 훨씬 더 많이 사용되는 경우 해당 대상에 다른 대상보다 더 높은 메시지 메모리 제한을 설정하거나 상황에 맞게 제한 동작을 조정할 수 있습니다. 필요한 연결 수가 최대 스레드 풀 크기에서 허용하는 연결 수보다 훨씬 더 큰 경우 스레드 풀 크기를 늘리거나 공유 스레드 모델을 적용할 수 있습니다. 최대 메시지 흐름이 평균 흐름보다 훨씬 큰 경우 메모리가 부족할 때 사용하는 제한 동작에 영향을 미칠 수 있습니다.

일반적으로 사용 패턴에 대해 더 많이 알수록 이러한 패턴에 맞춰 시스템을 조정하고 향후 요구에 더 잘 대비할 수 있습니다.

성능에 영향을 미치는 요소

두 가지 주요 성능 표시기인 메시지 대기 시간과 메시지 처리량은 일반 메시지가 메시지 전달 프로세스의 여러 단계를 완료하는 데 걸리는 시간에 따라 달라집니다. 아래에서 안정적으로 전달되는 지속성 메시지인 경우에 대해 이러한 단계를 보여 줍니다. 그 단계는 다음 그림과 같습니다.

그림 9-1 Message Queue 서비스를 통한 메시지 전달



1. 메시지가 생성자 클라이언트에서 메시지 서버로 전달됩니다.
2. 메시지 서버가 메시지를 읽습니다.
3. 안정성을 위해 메시지가 영구 저장소에 저장됩니다.
4. 안정성을 위해 메시지 서버가 메시지 수신을 확인합니다.
5. 메시지 서버가 메시지를 라우팅합니다.
6. 메시지 서버가 메시지를 씁니다.
7. 메시지가 메시지 서버에서 사용자 클라이언트로 전달됩니다.
8. 안정성을 위해 사용자 클라이언트가 메시지 수신을 확인합니다.
9. 안정성을 위해 메시지 서버가 클라이언트 확인을 처리합니다.
10. 메시지 서버가 클라이언트 확인이 처리되었는지 확인합니다.

이러한 단계는 순차적이므로 어느 단계든 생성자 클라이언트에서 사용자 클라이언트로 메시지를 전달할 때 병목 현상을 일으킬 수 있습니다. 이러한 단계들은 대부분 네트워크 대역폭, 컴퓨터 처리 속도, 메시지 서버 구조 등 메시징 시스템의 물리적 특성에 영향을 받습니다. 하지만 일부 단계는 메시징 응용 프로그램의 특성과 메시징 응용 프로그램이 요구하는 안정성 수준에 따라서도 달라집니다.

다음 하위 절에서는 응용 프로그램 설계 요소와 메시징 시스템 요소가 성능에 미치는 영향을 설명합니다. 응용 프로그램 설계와 메시징 시스템 요소가 메시지 전달에서 긴밀하게 상호 작용하지만 각 범주에 대해 별도로 고려합니다.

성능에 영향을 미치는 응용 프로그램 설계 요소

응용 프로그램 설계 결정은 전체 메시징 성능에 상당한 영향을 미칠 수 있습니다.

성능에 영향을 미치는 가장 중요한 요소는 메시지 전달의 안전성에 영향을 미치는 요소입니다. 이러한 요소는 다음과 같습니다.

- 전달 모드(지속성/비지속성 메시지)
- 트랜잭션 사용
- 확인 모드
- 영구 가입 및 비영구 가입

성능에 영향을 미치는 다른 응용 프로그램 설계 요소는 다음과 같습니다.

- 선택기 사용(메시지 필터링)
- 메시지 크기
- 메시지 본문 유형

다음 절에서는 이러한 요소 각각이 메시징 성능에 미치는 영향을 설명합니다. 일반적으로 성능과 안정성은 동시에 얻을 수 없습니다. 즉, 안정성을 높이는 요소가 성능을 저하시키는 경향이 있습니다.

다음 표는 여러 응용 프로그램 설계 요소가 일반적으로 메시징 성능에 어떤 영향을 미치는지 보여 줍니다. 이 표에서는 두 가지 시나리오(높은 안정성/낮은 성능 시나리오와 높은 성능/낮은 안정성 시나리오)와 각각을 특징 지우는 응용 프로그램 설계 요소의 선택을 보여 줍니다. 이러한 양극단 사이에는 안정성과 성능 모두에 영향을 미치는 여러 선택 사항과 상충이 있습니다.

표 9-1 높은 안정성 및 높은 성능 시나리오 비교

응용 프로그램 설계 요소	높은 안정성 낮은 성능 시나리오	높은 성능 낮은 안정성 시나리오
전달 모드	지속성 메시지	비지속성 메시지
트랜잭션 사용	트랜잭션된 세션	트랜잭션 없음
확인 모드	AUTO_ACKNOWLEDGE 또는 CLIENT_ACKNOWLEDGE	DUPS_OK_ACKNOWLEDGE
영구/비영구 가입	영구 가입	비영구 가입
선택기 사용	메시지 필터링	메시지 필터링 없음
메시지 크기	작은 메시지	큰 메시지
메시지 본문 유형	복합 본문 유형	단순 본문 유형

주 다음 그래프의 성능 데이터는 1002Mhz CPU가 두 개 있는 Solaris 8 시스템에서 파일 기반 지속성을 사용하여 생성되었습니다. 성능 테스트는 JIT (Just-In-Time) 컴파일러가 시스템을 최적화하고 지속성 데이터베이스를 사용할 수 있도록 먼저 Message Queue 브로커를 준비했습니다.

브로커가 준비되면 30초 동안 단일 생성자와 단일 사용자를 만들고 메시지를 생성했습니다. 사용자가 생성된 모든 메시지를 받는 데 필요한 시간을 기록했고 처리 속도(초당 메시지)를 계산했습니다. 표 9-1에 표시된 응용 프로그램 설계 요소의 다른 조합에 대해 이 시나리오를 반복했습니다.

전달 모드(지속성/비지속성 메시지)

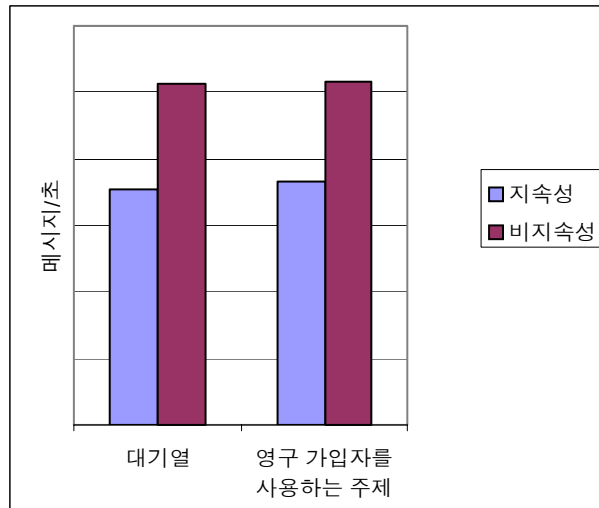
46페이지의 "안정적인 메시징"에서 설명한 대로 지속성 메시지는 메시지 서버에 오류가 발생하는 경우 메시지 전달을 보장합니다. 브로커는 의도된 모든 사용자가 메시지 사용을 확인할 때까지 메시지를 영구 저장소에 저장합니다.

다음과 같은 이유로 브로커의 지속성 메시지 처리가 비지속성 메시지 처리보다 느립니다.

- 브로커는 브로커에 오류가 발생하더라도 지속성 메시지가 손실되지 않도록 안정적으로 메시지를 저장해야 합니다.
- 브로커는 수신되는 각 지속성 메시지의 수신을 확인해야 합니다. 메시지를 생성하는 메소드가 예외 없이 반환되면 브로커로의 전달이 보장됩니다.
- 클라이언트 확인 모드에 따라 브로커는 지속성 메시지를 사용자 클라이언트가 확인했는지 확인해야 할 수 있습니다.

지속성 및 비지속성 모드간 성능 차이가 상당할 수 있습니다. 그림 9-2는 두 가지 안정적 전달 사례(대기열과 영구 가입을 사용하는 주제 양쪽에 전달되는 10k 크기의 메시지)에서 지속성 메시지와 비지속성 메시지에 대한 처리량을 비교합니다. 두 가지 경우 모두 AUTO_ACKNOWLEDGE 확인 모드를 사용합니다.

그림 9-2 전달 모드의 성능 영향



트랜잭션 사용

트랜잭션은 트랜잭션된 세션에서 생성된 모든 메시지와 트랜잭션된 세션에서 사용된 모든 메시지가 하나의 단위로 처리되거나 처리되지 않도록(롤백되도록) 보장합니다.

Message Queue는 로컬 트랜잭션과 분산 트랜잭션을 모두 지원합니다(자세한 내용은 각각 "로컬 트랜잭션" 및 47페이지의 "분산 트랜잭션" 참조).

트랜잭션된 세션에서의 메시지 생성이나 확인은 다음과 같은 이유 때문에 트랜잭션되지 않은 세션보다 느립니다.

- 생성된 각 메시지와 함께 추가 정보를 저장해야 합니다.
- 가입이 없는 주제 대상에 전달되는 지속성 메시지는 일반적으로 삭제되지만 트랜잭션이 시작될 때 가입에 대한 정보를 사용할 수 없는 경우와 같이 일반적으로는 그럴 수 없는 데도 트랜잭션의 메시지가 저장되는 경우가 있습니다.
- 트랜잭션이 완료될 때 트랜잭션 내의 메시지 사용과 확인에 대한 정보를 저장하고 처리해야 합니다.

확인 모드

JMS 메시지 전달의 안정성을 보장하는 한 가지 방법은 Message Queue 메시지 서버가 클라이언트에 전달한 메시지의 사용을 클라이언트가 확인하는 것입니다(59페이지의 "안정적인 전달: 확인 및 트랜잭션" 참조).

클라이언트의 메시지 확인 없이 세션이 닫히거나 확인이 처리되기 전에 메시지 서버에 오류가 발생하는 경우 브로커는 해당 메시지를 재전송하여 JMSRedelivered 플래그를 설정합니다.

트랜잭션되지 않은 세션의 경우 클라이언트는 각각 고유한 성능 특성을 가지는 다음과 같은 세 가지 확인 모드 중 하나를 선택할 수 있습니다.

- **AUTO_ACKNOWLEDGE.** 사용자가 메시지를 처리하면 시스템이 자동으로 메시지를 확인합니다. 이 모드는 공급자 오류 후 최대 한 개의 재전송 메시지를 보장합니다.
- **CLIENT_ACKNOWLEDGE.** 응용 프로그램이 메시지가 확인되는 시점을 제어합니다. 이전 확인 이후 해당 세션에서 처리된 모든 메시지가 확인됩니다. 일련의 확인을 처리하는 동안 메시지 서버가 실패하는 경우 해당 그룹에서 하나 이상의 메시지가 재전송될 수 있습니다.
- **DUPS_OK_ACKNOWLEDGE.** 이 모드는 시스템에게 메시지를 느리게 확인하도록 명령합니다. 공급자 오류 후 여러 메시지가 재전송될 수 있습니다.

(CLIENT_ACKNOWLEDGE 모드를 사용하는 것은 처리 중에 공급자 오류가 발생하는 경우 모든 확인이 함께 처리되도록 보장하지 않는다는 점을 제외하고는 트랜잭션 사용과 유사합니다.)

성능은 다음과 같은 이유로 확인 모드의 영향을 받습니다.

- **AUTO_ACKNOWLEDGE**와 **CLIENT_ACKNOWLEDGE** 모드에서는 브로커와 클라이언트 사이에 추가 제어 메시지가 필요합니다. 추가 제어 메시지는 처리 오버헤드를 추가하고 JMS 페이로드 메시지를 방해할 수 있으므로 처리가 지연됩니다.
- **AUTO_ACKNOWLEDGE**와 **CLIENT_ACKNOWLEDGE** 모드에서는 클라이언트가 추가 메시지를 사용할 수 있으려면 브로커가 클라이언트 확인을 처리했다고 확인할 때까지 대기해야 합니다. (이와 같은 브로커 확인은 브로커가 실수로 이 메시지를 재전송하지 않도록 보장합니다.)
- 사용자가 받은 모든 지속성 메시지에 대한 확인 정보로 Message Queue 영구 저장소를 업데이트해야 하므로 성능이 감소됩니다.

영구 가입 및 비영구 가입

주제 대상의 가입자는 44페이지의 "게시/가입(주제 대상)"에 설명되어 있는 대로 영구 가입자와 비영구 가입자의 두 가지 범주로 구분됩니다.

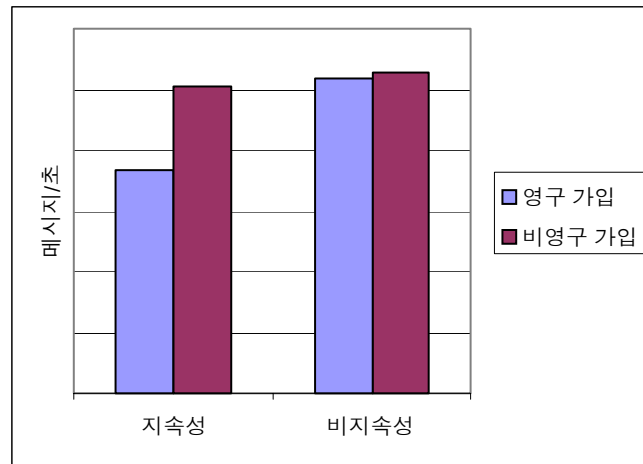
영구 가입은 다음과 같은 이유 때문에 안정성은 높이지만 처리량은 떨어뜨립니다.

- Message Queue 메시지 서버는 메시지 서버에 오류가 발생하더라도 복구 후 목록을 사용할 수 있도록 각 영구 가입에 할당된 메시지 목록을 영구 저장해야 합니다.
- 메시지 서버에 오류가 발생하더라도 복구 후 해당 사용자가 활성화되면 메시지를 계속 전달할 수 있도록 영구 가입의 지속성 메시지가 영구적으로 저장됩니다. 반면 비영구 가입의 지속성 메시지는 영구 저장되지 않습니다. 메시지 서버에 오류가 발생하는 경우 해당 사용자 연결이 끊기며 메시지가 전달되지 않습니다.

그림 9-3은 지속성 및 비지속성 10k 크기 메시지의 두 가지 경우에서 영구 및 비영구 가입을 사용하는 주제 대상의 처리량을 비교합니다. 두 가지 경우 모두 AUTO_ACKNOWLEDGE 확인 모드를 사용합니다.

그림 9-3에서 지속성 메시지인 경우에만 영구 가입 사용의 성능 영향이 확인된 것을 알 수 있으며 그 이유는 위에서 설명한 대로 지속성 메시지가 영구 가입에 대해서만 영구 저장되기 때문입니다.

그림 9-3 가입 유형의 성능 영향



선택기 사용(메시지 필터링)

응용 프로그램 개발자가 특정 사용자들을 메시지 집합의 대상으로 지정할 수 있습니다. 이는 메시지 집합마다 고유 대상을 지정하거나, 단일 대상을 사용하여 각 사용자에 대해 하나 이상의 선택기를 등록하여 수행할 수 있습니다.

선택기는 해당 문자열과 일치하는 등록 정보 값(38페이지의 "JMS 메시지 구조" 참조)을 갖는 메시지만 특정 사용자에게 전달되도록 요청하는 문자열입니다. 예를 들어 선택기 `NumberOfOrders > 1`은 `NumberOfOrders` 등록 정보 값이 2이상인 메시지만 전달합니다.

선택기를 사용하여 사용자를 등록하면 각 메시지를 처리하는 데 추가 처리가 필요하므로 여러 대상을 사용하는 것에 비해 성능이 떨어집니다. 선택기를 사용하는 경우 선택기가 이후의 메시지와 일치될 수 있도록 구문 분석되어야 합니다. 또한 각 메시지가 라우팅될 때 각 메시지의 메시지 등록 정보를 검색하고 선택기와 비교해야 합니다. 그러나 선택기를 사용하면 메시징 응용 프로그램의 융통성이 증가합니다.

메시지 크기

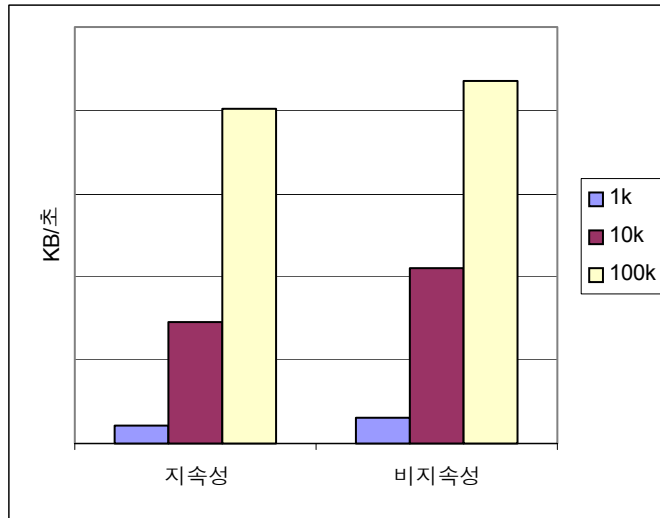
생성자 클라이언트에서 브로커로 그리고 브로커에서 사용자 클라이언트로 더 많은 데이터가 전달되어야 하고 지속성 메시지의 경우 더 큰 메시지를 저장해야 하므로 메시지 크기는 성능에 영향을 미칩니다.

그러나 작은 메시지들을 단일 메시지로 일괄 처리하면 개별 메시지의 라우팅과 처리를 최소화하여 전체적 성능 향상을 제공할 수 있습니다. 이 경우 개별 메시지의 상태에 대한 정보는 손실됩니다.

그림 9-4는 지속성 및 비지속성 메시지의 두 가지 경우에서 1k, 10k 및 100k 크기의 메시지에 대한 처리량(초당 KB)을 비교합니다. 모든 경우에서 메시지를 대기열 대상으로 보내며 `AUTO_ACKNOWLEDGE` 확인 모드를 사용합니다.

그림 9-4는 두 경우 모두에서 작은 메시지보다 큰 메시지를 전달할 때 오버헤드가 적다는 것을 보여 줍니다. 또한 크기가 1k 및 10k인 메시지에서는 비지속성 메시지가 지속성 메시지보다 약 50%의 성능 향상을 보였지만 100k 크기의 메시지에서는 아마도 네트워크 대역폭 때문에 해당 경우의 메시지 처리량에 병목 현상이 발생하여 이러한 성능 향상이 유지되지 않았음을 알 수 있습니다.

그림 9-4 메시지 크기의 성능 영향



메시지 본문 유형

JMS는 복잡성 순서에 따라 아래에 대략적으로 표시된 다섯 개의 메시지 본문 유형을 지원합니다.

- **BytesMessage:** 응용 프로그램에서 지정하는 형식의 바이트 집합을 포함합니다.
- **TextMessage:** 단순한 `java.lang.String`입니다.
- **StreamMessage:** Java 프리미티브 값의 스트림을 포함합니다.
- **MapMessage:** 일련의 이름-값 쌍을 포함합니다.
- **ObjectMessage:** Java 일련화 객체를 포함합니다.

일반적으로 메시지 유형은 응용 프로그램의 필요에 따라 제어되지만 좀 더 복잡한 유형 (**MapMessage** 및 **ObjectMessage**)은 성능 저하(데이터 일련화 및 일련화 해제로 인한 저하)를 수반합니다. 성능 저하는 데이터의 단순성이나 복잡성에 따라 달라집니다.

성능에 영향을 미치는 메시지 서비스 요소

메시징 응용 프로그램의 성능은 응용 프로그램 설계뿐만 아니라 메시지 라우팅 및 전달을 수행하는 메시지 서비스의 영향도 받습니다.

다음 절에서는 성능에 영향을 미칠 수 있는 여러 메시지 서비스 요소를 설명합니다. 이러한 요소의 영향을 이해하는 것은 메시지 서비스 크기를 지정하고 배포된 응용 프로그램에서 발생할 수 있는 성능 병목 현상을 진단하고 해결하는 데 중요합니다.

Message Queue 서비스에서 성능에 영향을 미치는 가장 중요한 요소는 다음과 같습니다.

- 하드웨어
- 운영 체제
- Java 가상 머신(JVM)
- 연결
- 브로커 제한 및 동작
- 메시지 서비스 구조
- 데이터 저장소 성능
- 클라이언트 런타임 구성

아래의 절에서는 이러한 각 요소가 메시징 성능에 미치는 영향을 설명합니다.

하드웨어

Message Queue 메시지 서버나 클라이언트 응용 프로그램 모두 CPU 처리 속도와 사용 가능한 메모리가 메시지 서비스 성능의 주요 결정 요소입니다. 처리량을 높이면 많은 소프트웨어 제한 사항을 제거할 수 있고 메모리를 추가하면 처리 속도와 용량을 모두 늘릴 수 있습니다. 그러나 하드웨어 업그레이드만으로 병목 현상을 극복하는 것은 일반적으로 비용이 높습니다.

운영 체제

하드웨어 플랫폼이 같은 경우라도 각기 다른 운영 체제의 효율성으로 인해 성능은 다양할 수 있습니다. 예를 들어, 운영 체제에서 사용하는 스레드 모델은 메시지 서버가 지원할 수 있는 동시 연결 수에 중요한 영향을 미칠 수 있습니다. 일반적으로 모든 하드웨어가 동일한 경우 Solaris가 Linux보다 빠르며 Linux가 Windows보다 빠릅니다.

Java 가상 머신(JVM)

메시지 서버는 호스트 JVM에서 실행되고 지원되는 Java 프로세스입니다. 결과적으로 JVM 처리는 메시지 서버가 얼마나 빠르고 효율적으로 메시지를 라우팅하고 전달할 수 있는지 결정하는 중요 요소입니다.

특히 JVM의 메모리 자원 관리는 중요할 수 있습니다. 메모리 로드 증가를 수용하기 위해 충분한 메모리를 JVM에 할당해야 합니다. 또한 JVM은 주기적으로 사용되지 않은 메모리를 재생 이용하며 이러한 메모리 재생 이용은 메시지 처리를 지연시킬 수 있습니다. JVM 메모리 힙이 클수록 메모리 재생 이용 중에 경험할 수 있는 잠재적 지연은 더 길어집니다.

연결

클라이언트와 브로커간 연결의 수와 속도는 메시지 서버가 처리할 수 있는 메시지 수와 메시지 전달 속도에 영향을 미칠 수 있습니다.

메시지 서버 연결 제한

메시지 서버에 대한 모든 액세스는 연결을 통해서 이루어집니다. 동시 연결 수에 대한 제한은 메시지 서버를 동시에 사용할 수 있는 생성자나 사용자 클라이언트 수에 영향을 미칠 수 있습니다.

일반적으로 메시지 서버에 대한 연결 수는 사용 가능한 스레드 수에 의해 제한됩니다. Message Queue는 전용 스레드 모델이나 공유 스레드 모델 중 하나를 지원하도록 구성할 수 있는 스레드 풀 관리자를 사용합니다(56페이지의 "스레드 풀 관리자" 참조). 전용 스레드 모델은 각 연결이 전용 스레드를 가지므로 아주 빠르지만 연결 수가 사용 가능한 스레드 수에 의해 제한됩니다(연결당 하나의 입력 스레드와 하나의 출력 스레드 필요). 공유 스레드 모델은 연결 수에 대한 제한이 없지만 많은 연결 사이에서 스레드를 공유하며 이러한 연결이 사용 중인 경우에는 상당한 오버헤드와 처리량 지연이 발생합니다.

전송 프로토콜

Message Queue 소프트웨어를 사용하면 클라이언트가 다양한 저급 전송 프로토콜을 사용하여 메시지 서버와 통신할 수 있습니다. Message Queue는 55페이지의 "연결 서비스 지원"에 표시된 연결 서비스와 해당 프로토콜을 지원합니다. 프로토콜은 응용 프로그램 요구 사항(암호화, 방화벽을 통한 액세스 가능)을 기반으로 선택하지만 이 선택은 전체 성능에 영향을 미칩니다.

그림 9-5 전송 프로토콜 속도



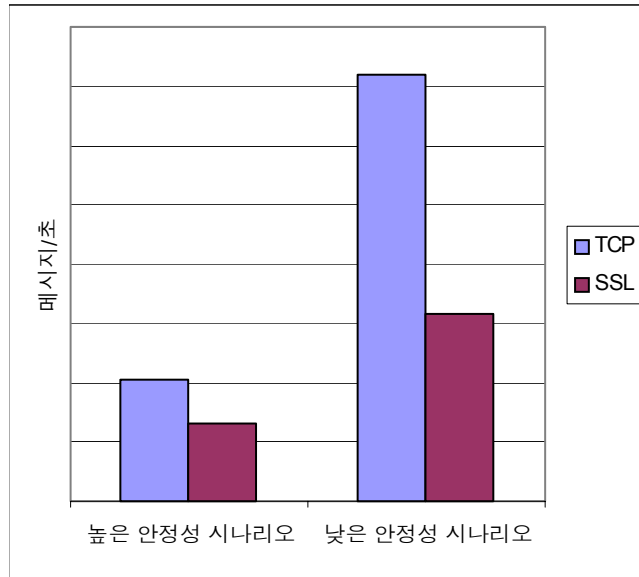
그림 9-5는 다양한 프로토콜 기술의 성능 특성을 반영합니다.

- TCP는 브로커와 통신할 수 있는 가장 빠른 방법을 제공합니다.
- SSL은 메시지를 보내고 받을 때 TCP보다 50-70퍼센트 더 느립니다(지속성 메시지의 경우 50퍼센트, 비지속성 메시지의 경우 70퍼센트에 근접). 또한 클라이언트와 브로커(또는 HTTPS의 경우 Web Server)가 전송할 데이터를 암호화할 때 사용될 개인 키를 설정해야 하므로 SSL을 사용하면 초기 연결 설정이 더 느려집니다(몇 초 걸릴 수 있음). 각 저급 TCP 패킷을 암호화하고 해독하는 데 필요한 추가 처리에 의해 성능이 저하됩니다.

그림 9-6은 높은 안정성 시나리오(영구 가입이 있고 AUTO_ACKNOWLEDGE 확인 모드를 사용하는 주제 대상에 1k의 지속성 메시지를 보냄)와 높은 성능 시나리오(영구 가입이 없고 DUPES_OK_ACKNOWLEDGE 확인 모드를 사용하는 주제 대상에 1k의 비지속성 메시지를 보냄)의 두 가지 경우에 대해 TCP와 SSL의 처리량을 비교합니다.

그림 9-6은 프로토콜이 높은 안정성 사례에서 영향을 덜 미침을 보여 줍니다. 이는 아마도 높은 안정성 사례에서 필요한 지속성 오버헤드가 처리량을 제한하는 데 프로토콜 속도보다 더 중요한 요소이기 때문일 것입니다.

그림 9-6 전송 프로토콜의 성능 영향



- HTTP는 TCP나 SSL보다 느립니다. HTTP는 Web Server에서 실행되는 서블릿을 클라이언트와 브로커 사이의 프록시로 사용합니다. 성능 오버헤드는 HTTP 요청의 패킷 캡슐화와 메시지가 브로커에 도달하기 위해 두 개의 홉(클라이언트에서 서블릿으로, 서블릿에서 브로커로)을 통과해야 한다는 점과 관련됩니다.
- 클라이언트와 서블릿 사이 그리고 서블릿과 브로커 사이에서 패킷을 암호화하는 데 필요한 추가 오버헤드로 인해 HTTPS는 HTTP보다 더 느립니다.

메시지 서비스 구조

Message Queue 메시지 서버는 단일 브로커로 구현되거나 다중 상호 연결 브로커 인스턴스, 즉 브로커 클러스터로 구현될 수 있습니다.

브로커에 연결된 클라이언트 수가 늘어나고 전달되는 메시지 수가 늘어나면 파일 설명자, 스레드, 메모리 제한 같은 브로커의 자원 제한 사항이 초과하게 됩니다. 늘어나는 로드를 수용하는 한 가지 방법은 Message Queue 메시지 서버에 브로커 인스턴스를 추가하여 클라이언트 연결과 메시지 라우팅 및 전달을 여러 브로커에 걸쳐 분산시키는 것입니다.

일반적으로 이러한 확장은 클라이언트, 특히 메시지 생성자 클라이언트가 클러스터 전체에 균등하게 분산되어 있는 경우에 가장 잘 작동합니다. 클러스터에 있는 브로커 사이의 메시지 전달과 관련된 오버헤드로 인해 제한된 연결 수나 메시지 전달 비율을 갖는 클러스터는 단일 브로커보다 더 낮은 성능을 보일 수 있습니다.

또한 브로커 클러스터를 사용하여 네트워크 대역폭을 최적화할 수 있습니다. 예를 들어, 클러스터 내의 원격 브로커들 사이에는 느린 장거리 네트워크 링크를 사용하고 클라이언트와 해당 브로커 인스턴스의 연결에는 고속 링크를 사용할 수 있습니다.

클러스터에 대한 자세한 내용은 [82페이지의 "멀티 브로커 클러스터\(엔터프라이즈판\)"](#) 및 [140페이지의 "클러스터를 이용한 작업\(엔터프라이즈판\)"](#)을 참조하십시오.

브로커 제한 및 동작

메시지 서버가 처리해야 하는 메시지 처리량은 메시지 서버가 지원하는 메시징 응용 프로그램의 사용 패턴에 달려 있습니다. 하지만 메시지 서버는 메모리, CPU 사이클 등 자원이 제한되어 있습니다. 따라서 메시지 서버가 넘치게 되어 응답하지 않거나 불안정하게 될 수 있습니다.

Message Queue 메시지 서버에는 메모리 자원을 관리하고 브로커의 메모리 부족을 방지하기 위해 기본적으로 제공되는 메커니즘이 있습니다. 이러한 메커니즘은 브로커나 개별 대상이 보유할 수 있는 메시지 수나 메시지 바이트에 대한 구성 가능한 제한 및 대상 제한에 이르면 시작될 수 있는 동작 집합을 포함합니다([61페이지의 "메모리 자원 및 메시지 흐름 관리"](#) 참조).

이러한 구성 가능한 메커니즘은 세밀하게 모니터링하고 조정하면 메시지 유입과 유출의 균형을 유지하여 시스템 과부하가 발생할 수 없도록 하는 데 사용할 수 있습니다. 이 메커니즘은 오버헤드를 사용하고 메시지 처리량을 제한할 순 있지만 운영 무결성을 유지합니다.

데이터 저장소 성능

Message Queue는 기본 제공 및 플러그인 지속성 모두를 지원합니다([63페이지의 "지속성 관리자"](#) 참조). 기본 제공 지속성은 파일 기반 데이터 저장소입니다. 플러그인 지속성은 JDBC™ (Java Database Connectivity) 인터페이스를 사용하며 JDBC 호환 데이터 저장소가 필요합니다.

기본 제공 지속성이 플러그인 지속성보다 상당히 빠르지만 JDBC 호환 데이터베이스 시스템은 응용 프로그램에 필요한 중복, 보안 및 관리 기능을 제공할 수 있습니다.

기본 제공 지속성의 경우 지속성 작업이 메모리 상태에서 데이터 저장소와 동기화되도록 지정하여 안정성을 최대화할 수 있습니다. 이렇게 하면 시스템 중단으로 인한 데이터 손실을 방지할 수 있지만 성능은 저하됩니다.

클라이언트 런타임 구성

Message Queue 클라이언트 런타임은 클라이언트 응용 프로그램에 Message Queue 메시지 서비스에 대한 인터페이스를 제공합니다. 클라이언트가 대상에게 메시지를 보내고 이러한 대상으로부터 메시지를 받는 데 필요한 모든 작업을 지원합니다. 클라이언트 런타임은 구성 가능하므로(연결 팩토리 속성 값을 설정하여) 일반적으로 성능과 메시지 처리량을 향상시킬 수 있는 등록 정보와 동작을 설정할 수 있습니다.

예를 들어 Message Queue 클라이언트 런타임은 다음과 같은 구성 가능한 동작을 지원합니다.

- JMS 메시지와 Message Queue 제어 메시지가 동일한 연결을 통해 흐르기 때문에 발생하는 혼잡을 막는 데 도움을 주는 연결 흐름 측정(imqConnectionFlowCount)
- 클라이언트 런타임 연결을 통해 전달될 수 있는 메시지 수와 사용 대기 중인 메시지 수를 제한하여 클라이언트 자원 제한을 피하는 데 도움을 주는 연결 흐름 제한(imqConnectionFlowLimit)
- 다중 사용자 대기열 전달의 경우 사용자 간의 로드 균형 조정을 향상시킬 수 있고(한 사용자에게 보내는 메시지 수의 불균형 방지), 연결된 한 사용자가 다른 연결된 사용자에게 과도하게 메시지를 보내는 것을 방지하는 데 도움을 주는 사용자 흐름 제한(imqConsumerFlowLimit). 이 등록 정보는 클라이언트 런타임 연결을 통해 전달될 수 있고 사용 대기 중인 사용자당 메시지 수를 제한합니다. 또한 이 등록 정보는 대기열 대상 등록 정보(consumerFlowLimit)로 구성될 수 있습니다.

이러한 동작들과 그 구성에 사용되는 속성에 대한 자세한 내용은 [289페이지의 "클라이언트 런타임 메시지 흐름 조정"](#)을 참조하십시오.

메시지 서버 모니터링

성능을 모니터링하는 데 사용할 수 있는 메트릭 정보를 제공하도록 Message Queue 서버를 구성할 수 있습니다. 이 절에서는 메시지 서버를 모니터링하는 데 사용할 수 있는 여러 도구와 이러한 도구를 사용하여 얻을 수 있는 메트릭 데이터에 대해 설명합니다.

메트릭 데이터를 사용하여 성능 문제를 해결하거나 메시지 서버 성능을 분석하고 조정하는 방법에 대한 자세한 내용은 [264페이지의 "성능 문제 해결"](#)을 참조하십시오.

모니터링 도구

다음 도구를 사용하여 메트릭 정보를 얻을 수 있습니다.

- [Message Queue 명령 유틸리티\(imqcmd\)](#)
- [Message Queue 브로커 로그 파일](#)
- [메시지 기반 모니터링 API](#)

다음 절에서는 이러한 각 도구를 사용하여 메트릭 정보를 얻는 방법을 설명합니다. 다른 도구간 비교는 [255페이지의 "올바른 모니터링 도구 선택"](#)을 참조하십시오.

Message Queue 명령 유틸리티(imqcmd)

명령 유틸리티(imqcmd)는 Message Queue의 기본 명령줄 관리 도구이며 물리적 대상, 영구 가입, 트랜잭션 등의 응용 프로그램 관련 자원뿐 아니라 브로커와 해당 연결 서비스를 관리할 수 있게 해 줍니다. imqcmd 명령에 대한 설명은 [6장, "브로커 및 응용 프로그램 관리"](#)를 참조하십시오.

imqcmd 명령의 기능 중 하나는 브로커 전체, 개별 연결 서비스 및 개별 대상에 대한 메트릭 정보를 얻을 수 있는 기능입니다. 메트릭 데이터를 얻으려면 일반적으로 imqcmd의 metrics 하위 명령을 사용합니다. 메트릭 데이터는 지정한 간격이나 지정한 횟수에 콘솔 화면에 기록됩니다.

또한 query 하위 명령([250페이지의 "imqcmd query"](#) 참조)을 사용하여 더욱 제한된 하위 집합의 메트릭 데이터를 얻을 수 있습니다.

imqcmd metrics

imqcmd metrics의 구문과 옵션은 각각 [표 9-2](#)와 [표 9-3](#)에 나타나 있습니다.

표 9-2 imqcmd metrics 하위 명령 구문

하위 명령 구문	제공되는 메트릭 데이터
<pre>metrics bkr [-b <i>hostName:port</i>] [-m <i>metricType</i>] [-int <i>interval</i>] [-msp <i>numSamples</i>] [-u <i>userName</i>] [-p <i>password</i>]</pre> <p>또는</p> <pre>metrics svc -n <i>serviceName</i> [-b <i>hostName:port</i>] [-m <i>metricType</i>] [-int <i>interval</i>] [-msp <i>numSamples</i>] [-u <i>userName</i>] [-p <i>password</i>]</pre> <p>또는</p> <pre>metrics dst -t <i>destType</i> -n <i>destName</i> [-b <i>hostName:port</i>] [-m <i>metricType</i>] [-int <i>interval</i>] [-msp <i>numSamples</i>] [-u <i>userName</i>] [-p <i>password</i>]</pre>	<p>기본 브로커 또는 지정한 호스트 및 포트의 브로커 메트릭을 표시합니다.</p> <p>기본 브로커 또는 지정한 호스트 및 포트의 브로커에서 지정된 서비스의 메트릭을 표시합니다.</p> <p>지정한 유형과 이름의 대상에 대한 메트릭 정보를 표시합니다.</p>

표 9-3 imqcmd metrics 하위 명령 옵션

하위 명령 옵션	설명
-b <i>hostName:port</i>	메트릭 데이터를 보고하는 브로커의 호스트 이름과 포트를 지정합니다. 기본값은 localhost:7676입니다.
-int <i>interval</i>	메트릭을 표시할 간격(초)을 지정합니다. 기본값은 5초입니다.

표 9-3 imqcmd metrics 하위 명령 옵션

하위 명령 옵션	설명
-m <i>metricType</i>	표시할 메트릭의 유형을 지정합니다. ttl 브로커에 유입되고 유출되는 메시지와패킷에 대한 메트릭을 표시합니다(기본 메트릭 유형). rts 브로커에 유입되고 유출되는 메시지와패킷의 흐름 속도에 대한 메트릭을 표시합니다(초당). cxm 연결, 가상 메모리 힙 및 스레드를 표시합니다(브로커 및 연결 서비스에만 해당). oon 사용자 관련 메트릭을 표시합니다(대상에만 해당). disk 디스크 사용 메트릭을 표시합니다(대상에만 해당).
-msp <i>numSamples</i>	출력에 표시되는 샘플 수를 지정합니다. 기본값은 무제한 수입니다(무한).
-n <i>destName</i>	메트릭 데이터가 보고되는 대상(있는 경우)의 대상 이름입니다. 기본값이 없습니다.
-n <i>serviceName</i>	메트릭 데이터가 보고되는 연결 서비스(있는 경우)를 지정합니다. 기본값이 없습니다.
-t <i>destTyp</i>	메트릭 데이터가 보고되는 대상(있는 경우)의 유형(대기열 또는 주제)을 지정합니다. 기본값이 없습니다.
-u <i>userName</i>	자신(관리자)의 이름을 지정합니다. 이 값을 생략하면 값을 묻는 메시지가 표시됩니다.
-p <i>password</i>	자신(관리자)의 비밀번호를 지정합니다. 이 값을 생략하면 값을 묻는 메시지가 표시됩니다.

절차: 메트릭 하위 명령을 사용하여 메트릭 데이터 표시

이 절에서는 metrics 하위 명령을 사용하여 메트릭 정보를 보고하는 절차를 설명합니다.

➤ **메트릭 하위 명령을 사용하는 방법**

1. 메트릭 정보가 필요한 브로커를 시작합니다.
 134페이지의 "브로커 시작"을 참조하십시오.
2. 표 9-2와 표 9-3에 표시된 대로 적절한 imqcmd metrics 하위 명령과 옵션을 실행합니다.

메트릭 출력: imqcmd metrics

이 절에서는 브로커 전체, 연결 서비스 및 대상 메트릭에 대한 `metrics` 하위 명령의 출력 예를 보여 줍니다.

브로커 전체 메트릭. 메시지와 패킷이 브로커에 유입 및 유출되는 속도를 10초 간격으로 구하려면 다음과 같이 `metrics bkr` 하위 명령을 사용합니다.

```
imqcmd metrics bkr -m rts -int 10 -u admin -p admin
```

이 명령은 다음과 유사한 출력을 생성합니다(258페이지의 표 9-8의 데이터 설명 참조).

Msgs/sec		Msg Bytes/sec		Pkts/sec		Pkt Bytes/sec	
In	Out	In	Out	In	Out	In	Out
0	0	27	56	0	0	38	66
10	0	7365	56	10	10	7457	1132
0	0	27	56	0	0	38	73
0	10	27	7402	10	20	1400	8459
0	0	27	56	0	0	38	73

연결 서비스 메트릭. `jms` 연결 서비스가 처리한 메시지와 패킷의 누적 총 수를 구하려면 다음과 같이 `metrics svc` 하위 명령을 사용합니다.

```
imqcmd metrics svc -n jms -m ttl -u admin -p admin
```

이 명령은 다음과 유사한 출력을 생성합니다(260페이지의 표 9-9의 데이터 설명 참조).

Msgs		Msg Bytes		Pkts		Pkt Bytes	
In	Out	In	Out	In	Out	In	Out
164	100	120704	73600	282	383	135967	102127
657	100	483552	73600	775	876	498815	149948

대상 메트릭. 대상에 대한 메트릭 정보를 얻으려면 다음과 같이 `metrics dst` 하위 명령을 사용하십시오.

```
imqcmd metrics dst -t q -n XQueue -m ttl -u admin -p admin
```

이 명령은 다음과 유사한 출력을 생성합니다(262페이지의 표 9-10의 데이터 설명 참조).

Msgs		Msg Bytes		Msg Count			Total Msg Bytes (k)			Largest
In	Out	In	Out	Current	Peak	Avg	Current	Peak	Avg	Msg (k)
200	200	147200	147200	0	200	0	0	143	71	0
300	200	220800	147200	100	200	10	71	143	64	0
300	300	220800	220800	0	200	0	0	143	59	0

대상의 사용자에 대한 정보를 얻으려면 다음과 같은 `metrics dst` 하위 명령을 사용합니다.

```
imqcmd metrics dst -t q -n SimpleQueue -m con -u admin -p admin
```

이 명령은 다음과 유사한 출력을 생성합니다(262페이지의 표 9-10의 데이터 설명 참조).

Active Consumers			Backup Consumers			Msg Count		
Current	Peak	Avg	Current	Peak	Avg	Current	Peak	Avg
1	1	0	0	0	0	944	1000	525

imqcmd query

`imqcmd query`의 구문과 옵션은 이 명령이 제공하는 메트릭 데이터에 대한 설명과 함께 표 9-4에 나와 있습니다.

표 9-4 imqcmd query 하위 명령 구문

하위 명령 구문	제공되는 메트릭 데이터
<pre>query bkr [-b hostName:port] [-int interval] [-msp numSamples]</pre>	브로커 메모리와 영구 저장소에 저장되어 있는 현재의 메시지 수와 메시지 바이트에 대한 정보(159페이지의 "브로커 정보 표시" 참조)
또는	

표 9-4 imqcmd query 하위 명령 구문

하위 명령 구문	제공되는 메트릭 데이터
<pre>metrics svc -n serviceName [-b hostName:port] [-int interval] [-msp numSamples]</pre>	지정한 연결 서비스에 대한 현재의 할당된 스레드 수와 연결 수에 대한 정보(165페이지 의 "연결 서비스 정보 표시" 참조)
또는	
<pre>metrics dst -t destType -n destName [-b hostName:port] [-int interval] [-msp numSamples]</pre>	지정한 대상의 메모리와 영구 저장소에 저장되어 있는 생성자, 활성 및 백업 사용자, 메시지 및 메시지 바이트의 현재 수에 대한 정보(173페이지 의 "대상 정보 표시" 참조)

주 imqcmd query에서 제공하는 메트릭 데이터는 제한되어 있기 때문에 이 도구는 **257페이지**의 "메트릭 데이터 설명" 절에 제시된 표에 설명되어 있지 않습니다.

Message Queue 브로커 로그 파일

Message Queue 로거는 브로커 코드, 디버거, 메트릭 생성기에서 생성한 정보를 가져와서 이 정보를 표준 출력(콘솔), 로그 파일, syslog 데몬 프로세스(Solaris™ 플랫폼인 경우) 등과 같은 여러 출력 채널에 기록합니다. 로거에 대해서는 **71페이지**의 "로거"에서 설명합니다.

로거에서 수집된 정보의 유형과 각 출력 채널에 기록된 유형을 지정할 수 있습니다. 특히 메트릭 정보가 로그 파일에 기록되도록 지정할 수 있습니다.

절차: 브로커 로그 파일을 사용하여 메트릭 데이터 보고

이 절에서는 브로커 로그 파일을 사용하여 메트릭 정보를 보고하는 절차를 설명합니다. 로거 구성에 대한 자세한 내용은 **147페이지**의 "로깅"을 참조하십시오.

▶ 로그 파일을 사용하여 메트릭 정보를 보고하는 방법

1. 브로커의 메트릭 생성 기능을 구성합니다.
 - a. imq.metrics.enabled=true인지 확인합니다.

로깅을 위한 메트릭 생성은 기본적으로 설정되어 있습니다.

- b. 메트릭 생성 간격을 원하는 시간(초)으로 설정합니다.

`imq.metrics.interval=interval`

이 값은 `config.properties` 파일에서 설정하거나 브로커를 시작할 때 `-metrics interval` 명령줄 옵션을 사용하여 설정할 수 있습니다.

- 2. 로거가 메트릭 정보를 수집하는지 확인합니다.

`imq.log.level=INFO`

이것이 기본값입니다. 이 값은 `config.properties` 파일에서 설정하거나 브로커를 시작할 때 `-loglevel level` 명령줄 옵션을 사용하여 설정할 수 있습니다.

- 3. 로거가 메트릭 정보를 로그 파일에 기록하도록 설정되어 있는지 확인합니다.

`imq.log.file.output=INFO`

이것이 기본값입니다. 이 값은 `config.properties` 파일에서 설정할 수 있습니다.

- 4. 브로커를 시작합니다.

메트릭 출력: 로그 파일

다음에는 로그 파일로의 샘플 브로커 메트릭 출력이 나와 있습니다(표 9-7 및 258페이지의 표 9-8의 메트릭 데이터 설명 참조).

```
[21/Jul/2003:11:21:18 PDT]
Connections: 0   JVM Heap: 8323072 bytes (7226576 free) Threads: 0 (14-1010)
  In: 0 msgs (0bytes) 0 pkts (0 bytes)
  Out: 0 msgs (0bytes) 0 pkts (0 bytes)
Rate In: 0 msgs/sec (0 bytes/sec) 0 pkts/sec (0 bytes/sec)
Rate Out: 0 msgs/sec (0 bytes/sec) 0 pkts/sec (0 bytes/sec)
```

메시지 기반 모니터링 API

Message Queue는 브로커가 메트릭 데이터를 JMS 메시지에 기록한 다음 메시지에 포함된 메트릭 정보 유형에 따라 여러 메트릭 주제 대상 중 하나에 보낼 수 있는 메트릭 모니터링 기능을 제공합니다.

메트릭 주제 대상에 가입하고 이러한 대상에서 메시지를 사용하며 메시지에 포함된 메트릭 정보를 처리하는 클라이언트 응용 프로그램을 작성하여 이러한 메트릭 정보에 액세스할 수 있습니다. 일반 체계가 73페이지의 "메트릭 메시지 생성자(엔터프라이즈판)"에 설명되어 있습니다.

다섯 개의 메트릭 주제 대상이 있으며 표 9-5에 그 이름과 함께 각 대상에 전달되는 메트릭 메시지 유형이 표시되어 있습니다.

표 9-5 메트릭 주제 대상

주제 이름	메트릭 메시지 유형
mq.metrics.broker	브로커 메트릭
mq.metrics.jvm	Java 가상 머신 메트릭
mq.metrics.destination_list	대상 및 해당 유형 목록
mq.metrics.destination.queue. <i>monitoredDestinationName</i>	지정된 이름의 대기열에 대한 대상 메트릭
mq.metrics.destination.topic. <i>monitoredDestinationName</i>	지정된 이름의 주제에 대한 대상 메트릭

절차: 메시지 기반 모니터링 설정

이 절에서는 메시지 기반 모니터링 기능을 사용하여 메트릭 정보를 수집하는 절차를 설명합니다. 이 절차는 클라이언트 개발과 관리 작업을 모두 포함합니다.

▶ 메시지 기반 모니터링을 설정하는 방법

1. 메트릭 모니터링 클라이언트를 작성합니다.

메트릭 주제 대상에 가입하고 메트릭 메시지를 사용하며 이러한 메시지에서 메트릭 데이터를 추출하는 클라이언트를 프로그래밍하는 데 대한 지침은 *Message Queue Java Client Developer's Guide*를 참조하십시오.

2. `config.properties` 파일에서 브로커 등록 정보 값을 설정하여 브로커의 메트릭 메시지 생성자를 구성합니다.

- a. 메트릭 메시지 생성을 활성화합니다.

`imq.metrics.topic.enabled=true`를 설정합니다.

기본값은 true입니다.

- b. 메트릭 메시지가 생성되는 간격(초)을 설정합니다.

`imq.metrics.topic.interval=interval`을 설정합니다.

기본값은 60초입니다.

- c. 메트릭 메시지가 지속되는지(즉, 브로커에 오류가 발생해도 메시지가 보존되는지) 여부를 지정합니다.

`mq.metrics.topic.persist`를 설정합니다.

기본값은 `false`입니다.

- d. 메트릭 메시지가 삭제되기 전까지 해당 대상에 남아 있는 기간을 지정합니다.

`mq.metrics.topic.timetolive`를 지정합니다.

기본값은 300초입니다.

- 3. 메트릭 주제 대상에 대한 액세스 제어가 필요한 경우 설정합니다.

아래의 "[보안 및 액세스 고려 사항](#)"에 있는 설명을 참조하십시오.

- 4. 메트릭 모니터링 클라이언트를 시작합니다.

사용자가 메트릭 주제에 가입하면 메트릭 주제 대상이 자동으로 만들어집니다. 메트릭 주제가 만들어지면 브로커 메트릭 메시지 생성자가 메트릭 메시지를 메트릭 주제로 보내기 시작합니다.

보안 및 액세스 고려 사항

메트릭 주제 대상에 대한 액세스를 제한하는 이유는 두 가지입니다.

- 메트릭 데이터에는 브로커와 그 자원에 대한 중요한 정보가 포함되어 있을 수 있습니다.
- 메트릭 주제 대상에 대한 가입 수가 과도한 경우 브로커 오버헤드가 늘어나고 성능에 부정적인 영향을 미칠 수 있습니다.

이러한 사항을 고려할 때 메트릭 주제 대상에 대한 액세스를 제한하는 것이 좋습니다.

모니터링 클라이언트는 다른 클라이언트와 같은 인증 및 권한 부여 제어를 받습니다.

Message Queue 사용자 저장소에서 유지 관리되는 사용자만 브로커에 연결할 수 있습니다.

212페이지의 "[사용자 권한 부여: 액세스 제어 등록 정보 파일](#)"에 설명되어 있는 액세스 제어 등록 정보 파일을 통해 특정 메트릭 주제 대상에 대한 액세스를 제한하여 추가 보호를 제공할 수 있습니다.

예를 들어 `accesscontrol.properties` 파일의 다음 항목은 `user1`과 `user2`를 제외한 모든 사람에게 대해 `mq.metrics.broker` 메트릭 주제에 대한 액세스를 거부합니다.

```
topic.mq.metrics.broker.consume.deny.user=*
topic.mq.metrics.broker.consume.allow.user=user1,user2
```

다음 항목은 사용자 **user3**만 주제 **t1**을 모니터링할 수 있도록 합니다.

```
topic.mq.metrics.destination.topic.t1.consume.deny.user=*
topic.mq.metrics.destination.topic.t1.consume.allow.user=user3
```

메트릭 데이터의 중요도에 따라 암호화된 연결을 사용하여 메트릭 모니터링 클라이언트를 브로커에 연결할 수도 있습니다. 암호화된 연결 사용에 대한 자세한 내용은 [218페이지](#)의 "[암호화: SSL 기반 서비스를 사용한 작업\(엔터프라이즈판\)](#)"을 참조하십시오.

메트릭 출력: 메트릭 메시지

메시지 기반 모니터링 API를 사용하여 얻는 메트릭 데이터 출력은 사용자가 어떤 메트릭 모니터링 클라이언트를 작성하는지에 따라 다릅니다. 단지 브로커의 메트릭 생성기에서 어떤 데이터를 제공하는지에 따라 제한을 받습니다. 이 데이터의 전체 목록을 보려면 [257페이지](#)의 "[메트릭 데이터 설명](#)"을 참조하십시오.

올바른 모니터링 도구 선택

앞 절에서 설명한 모니터링 도구마다 장점과 단점이 있습니다.

예를 들어 `imqcmd metrics` 명령을 사용하면 언제라도 요구에 맞는 정보를 빠르게 샘플링할 수 있지만 기록 정보를 살펴거나 데이터를 프로그래밍 방식으로 조작하기가 다소 어렵습니다.

반면 로그 파일은 장기간의 메트릭 데이터 기록을 제공하지만 로그 파일의 정보를 의미 있는 정보로 구문 분석하기가 어렵습니다.

메시지 기반 모니터링 API를 사용하는 경우 필요한 정보 추출 및 처리, 데이터를 프로그래밍 방식으로 조작 또는 형식 지정, 그래프 표현, 경고 보내기 등을 쉽게 수행할 수 있지만 데이터를 캡처하고 분석하기 위한 사용자 정의 응용 프로그램을 작성해야 합니다.

또한 이러한 각 도구는 브로커가 생성한 메트릭 정보 중 약간씩 다른 하위 집합을 수집합니다. 각 모니터링 도구가 수집하는 메트릭 데이터에 대한 자세한 내용은 [257페이지의 "메트릭 데이터 설명"](#)을 참조하십시오.

표 9-6에서는 다른 도구 간의 장점과 단점을 제시하여 비교합니다.

표 9-6 메트릭 모니터링 도구의 장점 및 단점

메트릭 모니터링 도구	장점	단점
imqcmd metrics	원격 모니터링 스팟 체크에 적합 명령 옵션에 설정된 보고 간격을 실행 중에 변경 가능 원하는 특정 데이터를 선택하기 쉬움 보기 쉬운 테이블 형식으로 데이터 제시	하나의 명령으로 모든 데이터를 얻을 수 없음 데이터를 프로그래밍 방식으로 분석하기 어려움 기록 레코드를 작성하지 않음 기록 추세를 보기 어려움
로그 파일	정기적인 샘플링 기록 레코드 작성	브로커 등록 정보를 구성해야 하며 적용하려면 브로커를 종료하고 다시 시작해야 함 로컬 모니터링 전용 데이터 형식이 읽거나 구문 분석하기가 아주 어려우며 구문 분석 도구 없음 보고 간격을 실행 중에 변경할 수 없으며 모든 메트릭 데이터도 마찬가지임 데이터 선택에 융통성이 없음 브로커 메트릭 전용이며 대상 및 연결 서비스 메트릭은 포함되어 있지 않음 간격을 너무 짧게 설정하면 성능에 악영향을 줄 수 있음
메시지 기반 모니터링 API	원격 모니터링 원하는 특정 데이터를 선택하기 쉬움 데이터를 전자적으로 분석하고 모든 형식으로 표시할 수 있음	브로커 등록 정보를 구성해야 하며 적용하려면 브로커를 종료하고 다시 시작해야 함 사용자 고유의 메트릭 모니터링 클라이언트를 작성해야 함 보고 간격을 실행 중에 변경할 수 없으며 모든 메트릭 데이터도 마찬가지임

메트릭 데이터 설명

브로커가 보고하는 메트릭 정보는 다음과 같은 범주로 그룹화할 수 있습니다.

- **JVM (Java 가상 머신) 메트릭.** JVM 힙 크기에 대한 정보
- **브로커 전체 메트릭.** 메시지 수 및 바이트 수(해당 수 및 속도)라는 측면에서 브로커에 저장되어 있는 메시지 및 브로커에 유입 및 유출되는 메시지 흐름에 대한 정보. 이 범주에는 메모리 사용에 대한 정보도 포함됩니다.
- **연결 서비스 메트릭.** 연결 및 연결 스레드 자원에 대한 정보 및 특정 연결 서비스의 메시지 흐름에 대한 정보
- **대상 메트릭.** 특정 대상에 유입하고 유출하는 메시지에 대한 정보, 대상 사용자에 대한 정보, 메모리 및 디스크 공간 사용에 대한 정보

다음 절에서는 이러한 각 범주에서 사용할 수 있는 메트릭 데이터를 설명합니다. 다음 표에서 설명하는 모니터링 도구에 대한 자세한 내용은 [246페이지의 "모니터링 도구"](#)를 참조하십시오.

JVM 메트릭

[표 9-7](#)에서는 브로커가 브로커 프로세스 JVM 힙에 대해 생성하는 메트릭 데이터를 나열하고 설명하며 메트릭 모니터링 도구에 따라 어떤 데이터를 얻을 수 있는지 보여 줍니다.

표 9-7 JVM 메트릭

메트릭 개수	설명	imqcmd metrics bkr (metricType)	로그 파일	메트릭 메시지 (metrics topic) ²
JVM 힙: 사용 가능한 메모리	JVM 힙에서 사용할 수 있는 사용 가능한 메모리 양	예 (cxn)	예	예 (...jvm)
JVM 힙: 전체 메모리	현재 JVM 힙 크기	예 (cxn)	예	예 (...jvm)
JVM 힙: 최대 메모리	JVM 힙 크기가 늘어날 수 있는 최대값	아니요	예 ¹	예 (...jvm)

1. 브로커 시작 시에만 표시됩니다.
2. 메트릭 주제 대상 이름에 대해서는 [253페이지의 표 9-5](#)를 참조하십시오.

브로커 전체 메트릭

표 9-8에서는 브로커 전체 메트릭 정보에 대해 브로커가 보고하는 데이터를 나열하고 설명합니다. 또한 메트릭 모니터링 도구에 따라 어떤 데이터를 얻을 수 있는지 보여 줍니다.

표 9-8 브로커 전체 메트릭

메트릭 개수	설명	imqcmd metrics bkr (metricType)	로그 파일	메트릭 메시지 (metrics topic) ¹
연결 데이터				
Num connections	브로커에 대해 현재 열려 있는 연결 수	예 (cxn)	예	예 (...broker)
Num threads	현재 사용 중인 스레드 수	예 (cxn)	예	아니요
Min threads	이 스레드 수에 도달하면 스레드 풀에서 연결 서비스용으로 유지 관리되는 스레드 수	예 (cxn)	예	아니요
Max threads	이 스레드 수를 초과하면 스레드 풀에 연결 서비스용으로 새 스레드가 더 이상 추가되지 않는 스레드 수	예 (cxn)	예	아니요
저장된 메시지 데이터				
Num messages	브로커 메모리와 영구 저장소에 현재 저장되어 있는 JMS 메시지 수	아니요 query bkr 사용	아니요	예 (...broker)
Total message bytes	브로커 메모리와 영구 저장소에 현재 저장되어 있는 JMS 메시지 바이트 수	아니요 query bkr 사용	아니요	예 (...broker)
메시지 흐름 데이터				
Num messages in	브로커가 마지막으로 시작된 이후 브로커에 유입된 JMS 메시지 수	예 (ttl)	예	예 (...broker)
Message bytes in	브로커가 마지막으로 시작된 이후 브로커에 유입된 JMS 메시지 바이트 수	예 (ttl)	예	예 (...broker)
Num packets in	브로커가 마지막으로 시작된 이후 브로커에 유입된 패킷 수(JMS 메시지와 제어 메시지 모두 포함)	예 (ttl)	예	예 (...broker)
Packet bytes in	브로커가 마지막으로 시작된 이후 브로커에 유입된 패킷 바이트 수(JMS 메시지와 제어 메시지 모두 포함)	예 (ttl)	예	예 (...broker)

표 9-8 브로커 전체 메트릭(계속)

메트릭 개수	설명	imqcmd metrics bkr (metricType)	로그 파일	메트릭 메시지 (metrics topic) ¹
Num messages out	브로커가 마지막으로 시작된 이후 브로커에서 유출된 JMS 메시지 수	예 (ttl)	예	예 (...broker)
Message bytes out	브로커가 마지막으로 시작된 이후 브로커에서 유출된 JMS 메시지 바이트 수	예 (ttl)	예	예 (...broker)
Num packets out	브로커가 마지막으로 시작된 이후 브로커에서 유출된 패킷 수(JMS 메시지와 제어 메시지 모두 포함)	예 (ttl)	예	예 (...broker)
Packet bytes out	브로커가 마지막으로 시작된 이후 브로커에서 유출된 패킷 바이트 수(JMS 메시지와 제어 메시지 모두 포함)	예 (ttl)	예	예 (...broker)
Rate messages in	브로커로 JMS 메시지가 유입되는 현재 속도	예 (rts)	예	아니요
Rate message bytes in	브로커로 JMS 메시지 바이트가 유입되는 현재 속도	예 (rts)	예	아니요
Rate packets in	브로커로 패킷이 유입되는 현재 속도(JMS 메시지와 제어 메시지 모두 포함)	예 (rts)	예	아니요
Rate packet bytes in	브로커로 패킷 바이트가 유입되는 현재 속도(JMS 메시지와 제어 메시지 모두 포함)	예 (rts)	예	아니요
Rate messages out	브로커에서 JMS 메시지가 유출되는 현재 속도	예 (rts)	예	아니요
Rate message bytes out	브로커에서 JMS 메시지 바이트가 유출되는 현재 속도	예 (rts)	예	아니요
Rate packets out	브로커에서 패키지가 유출되는 현재 속도(JMS 메시지와 제어 메시지 모두 포함)	예 (rts)	예	아니요
Rate packet bytes out	브로커에서 패키지 바이트가 유출되는 현재 속도(JMS 메시지와 제어 메시지 모두 포함)	예 (rts)	예	아니요
대상 데이터				
Num destinations	브로커에 있는 실제 대상 수	아니요	아니요	예 (...broker)

1. 메트릭 주제 대상 이름에 대해서는 253페이지의 표 9-5를 참조하십시오.

연결 서비스 메트릭

표 9-9에서는 개별 연결 서비스에 대해 브로커가 보고하는 메트릭 데이터를 나열하고 설명합니다. 또한 메트릭 모니터링 도구에 따라 어떤 데이터를 얻을 수 있는지 보여 줍니다.

표 9-9 연결 서비스 메트릭

메트릭 개수	설명	imqcmd metrics svc (metricType)	로그 파일	메트릭 메시지 (metrics topic)
연결 데이터				
Num connections	현재 열려 있는 연결의 수	예 (cxn) 또는 query svc	아니요	아니요
Num threads	현재 사용 중인 스레드 수(모든 연결 서비스에서 합산)	예 (cxn) 또는 query svc	아니요	아니요
Min threads	이 스레드 수에 도달하면 스레드 풀에서 연결 서비스용으로 유지 관리되는 스레드 수(전체 연결 서비스에서 합산)	예 (cxn)	아니요	아니요
Max threads	이 스레드 수를 초과하면 스레드 풀에 연결 서비스용으로 새 스레드가 더 이상 추가되지 않는 스레드 수(전체 연결 서비스에서 합산)	예 (cxn)	아니요	아니요
메시지 흐름 데이터				
Num messages in	브로커가 마지막으로 시작된 이후 연결 서비스에 유입된 JMS 메시지의 수	예 (ttl)	아니요	아니요
Message bytes in	브로커가 마지막으로 시작된 이후 연결 서비스에 유입된 JMS 메시지 바이트 수	예 (ttl)	아니요	아니요
Num packets in	브로커가 마지막으로 시작된 이후 연결 서비스에 유입된 패킷 수(JMS 메시지와 제어 메시지 모두 포함)	예 (ttl)	아니요	아니요
Packet bytes in	브로커가 마지막으로 시작된 이후 연결 서비스에 유입된 패킷 바이트 수(JMS 메시지와 제어 메시지 모두 포함)	예 (ttl)	아니요	아니요
Num messages out	브로커가 마지막으로 시작된 이후 연결 서비스에서 유출된 JMS 메시지 수	예 (ttl)	아니요	아니요

표 9-9 연결 서비스 메트릭(계속)

메트릭 개수	설명	mqcmd metrics svc (metricType)	로그 파일	메트릭 메시지 (metrics topic)
Message bytes out	브로커가 마지막으로 시작된 이후 연결 서비스에서 유출된 JMS 메시지 바이트 수	예 (ttl)	아니요	아니요
Num packets out	브로커가 마지막으로 시작된 이후 연결 서비스에서 유출된 패킷 수(JMS 메시지와 제어 메시지 모두 포함)	예 (ttl)	아니요	아니요
Packet bytes out	브로커가 마지막으로 시작된 이후 연결 서비스에서 유출된 패킷 바이트 수(JMS 메시지와 제어 메시지 모두 포함)	예 (ttl)	아니요	아니요
Rate messages in	연결 서비스를 통해 JMS 메시지가 브로커로 유입되는 현재 속도	예 (rts)	아니요	아니요
Rate message bytes in	연결 서비스로 JMS 메시지 바이트가 유입되는 현재 속도	예 (rts)	아니요	아니요
Rate packets in	연결 서비스로 패킷이 유입되는 현재 속도(JMS 메시지와 제어 메시지 모두 포함)	예 (rts)	아니요	아니요
Rate packet bytes in	연결 서비스로 패킷 바이트가 유입되는 현재 속도(JMS 메시지와 제어 메시지 모두 포함)	예 (rts)	아니요	아니요
Rate messages out	연결 서비스에서 JMS 메시지가 유출되는 현재 속도	예 (rts)	아니요	아니요
Rate message bytes out	연결 서비스에서 JMS 메시지 바이트가 유출되는 현재 속도	예 (rts)	아니요	아니요
Rate packets out	연결 서비스에서 패킷이 유출되는 현재 속도(JMS 메시지와 제어 메시지 모두 포함)	예 (rts)	아니요	아니요
Rate packet bytes out	연결 서비스에서 패킷 바이트가 유출되는 현재 속도(JMS 메시지와 제어 메시지 모두 포함)	예 (rts)	아니요	아니요

대상 메트릭

표 9-9에서는 개별 대상에 대해 브로커가 보고하는 메트릭 데이터를 나열하고 설명합니다. 또한 메트릭 모니터링 도구에 따라 어떤 데이터를 얻을 수 있는지 보여 줍니다.

표 9-10 대상 메트릭

메트릭 개수	설명	imqcmd metrics dst (metricType)	로그 파일	메트릭 메시지 (metrics topic) ¹
사용자 데이터				
Num active consumers	현재 활성 사용자의 수	예 (con)	아니요	예 (...destName)
Avg num active consumers	브로커가 마지막으로 시작된 이후 평균 활성 사용자 수	예 (con)	아니요	예 (...destName)
Peak num active consumers	브로커가 마지막으로 시작된 이후 최대 활성 사용자 수	예 (con)	아니요	예 (...destName)
Num backup consumers	현재 백업 사용자 수(대기열에만 적용)	예 (con)	아니요	예 (...destName)
Avg num backup consumers	브로커가 마지막으로 시작된 이후 평균 백업 사용자 수(대기열에만 적용)	예 (con)	아니요	예 (...destName)
Peak num backup consumers	브로커가 마지막으로 시작된 이후 최대 백업 사용자 수(대기열에만 적용)	예 (con)	아니요	예 (...destName)
저장된 메시지 데이터				
Num messages	대상 메모리와 영구 저장소에 현재 저장되어 있는 JMS 메시지 수	예 (con) (ttl) (rts) 또는 query dst	아니요	예 (...destName)
Avg num messages	브로커가 마지막으로 시작된 이후 대상 메모리와 영구 저장소에 저장되어 있는 평균 JMS 메시지 수	예 (con) (ttl) (rts)	아니요	예 (...destName)
Peak num messages	브로커가 마지막으로 시작된 이후 대상 메모리와 영구 저장소에 저장되어 있는 최대 JMS 메시지 수	예 (con) (ttl) (rts)	아니요	예 (...destName)
Total message bytes	대상 메모리와 영구 저장소에 현재 저장되어 있는 JMS 메시지 바이트 수	예 (ttl) (rts) 또는 query dst	아니요	예 (...destName)
Avg total message bytes	브로커가 마지막으로 시작된 이후 대상 메모리와 영구 저장소에 저장되어 있는 평균 JMS 메시지 바이트 수	예 (ttl) (rts)	아니요	예 (...destName)

표 9-10 대상 메트릭(계속)

메트릭 개수	설명	imqcmd metrics dst (metricType)	로그 파일	메트릭 메시지 (metrics topic) ¹
Peak total message bytes	브로커가 마지막으로 시작된 이후 대상 메모리와 영구 저장소에 저장되어 있는 최대 JMS 메시지 바이트 수	예 (ttl) (rts)	아니요	예 (...destName)
Peak message bytes	브로커가 마지막으로 시작된 이후 대상에서 수신한 단일 메시지에 있는 최대 JMS 메시지 바이트 수	예 (ttl) (rts)	아니요	예 (...destName)
메시지 흐름 데이터				
Num messages in	브로커가 마지막으로 시작된 이후 이 대상에 유입된 JMS 메시지 수	예 (ttl)	아니요	예 (...destName)
Msg bytes in	브로커가 마지막으로 시작된 이후 이 대상에 유입된 JMS 메시지 바이트 수	예 (ttl)	아니요	예 (...destName)
Num messages out	브로커가 마지막으로 시작된 이후 이 대상에서 유출된 JMS 메시지 수	예 (ttl)	아니요	예 (...destName)
Msg bytes out	브로커가 마지막으로 시작된 이후 이 대상에서 유출된 JMS 메시지 바이트 수	예 (ttl)	아니요	예 (...destName)
Rate num messages in	JMS 메시지가 대상에 유입되는 현재 속도	예 (rts)	아니요	아니요
Rate num messages out	JMS 메시지가 대상에서 유출되는 현재 속도	예 (rts)	아니요	아니요
Rate msg bytes in	JMS 메시지 바이트가 대상에 유입되는 현재 속도	예 (rts)	아니요	아니요
Rate Msg bytes out	JMS 메시지가 대상에서 유출되는 현재 속도	예 (rts)	아니요	아니요
디스크 사용률 데이터				
Disk reserved	대상 파일 기반 저장소에서 모든 메시지 레코드(활성 및 사용 가능)가 사용하는 디스크 공간(바이트)	예 (disk)	아니요	예 (...destName)
Disk used	대상 파일 기반 저장소에서 활성 메시지 레코드가 사용하는 디스크 공간(바이트)	예 (disk)	아니요	예 (...destName)

표 9-10 대상 메트릭(계속)

메트릭 개수	설명	mqcmd metrics dst (metricType)	로그 파일	메트릭 메시지 (metrics topic) ¹
Disk utilization ratio	예약된 디스크 공간에 대한 사용 중인 디스크 공간의 비율. 비율이 높을수록 더 많은 디스크 공간이 활성 메시지를 보관하는 데 사용되고 있는 것입니다.	예 (disk)	아니요	예 (...destName)#2 004-05-16#

1. 메트릭 주제 대상 이름에 대해서는 [253페이지의 표 9-5](#)를 참조하십시오.

성능 문제 해결

응용 프로그램을 지원하기 위해 Message Queue 서비스를 사용할 때 여러 성능 문제가 발생할 수 있습니다. 이러한 문제는 다음과 같습니다.

- 문제: 클라이언트가 연결을 설정할 수 없음
- 문제: 연결 처리량이 너무 느림
- 문제: 클라이언트가 메시지 생성자를 만들 수 없음
- 문제: 메시지 생성이 지연되거나 느림
- 문제: 메시지가 메시지 서버에서 백로그됨
- 문제: 메시지 서버 처리량이 일정하지 않음
- 문제: 메시지가 사용자에게 도달하지 않음

아래에서 이러한 각 문제 대해 가능한 원인 및 해결책과 함께 설명합니다.

문제: 클라이언트가 연결을 설정할 수 없음

증상:

- 클라이언트가 새 연결을 설정할 수 없습니다.
- 클라이언트가 실패한 연결을 자동으로 다시 연결할 수 없습니다.

가능한 원인:

- 클라이언트 응용 프로그램이 연결을 닫을 수 없어 연결 수가 자원 제한을 초과합니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

브로커에 대한 연결을 모두 나열합니다.

```
imqcmd list cxn
```

출력에 모든 연결과 각 연결이 설정된 호스트가 나열되고 특정 클라이언트에 비정상적인 수의 연결이 열려 있는 것으로 표시됩니다.

문제를 해결하려면 다음 작업을 수행합니다.

문제가 있는 클라이언트를 다시 작성하여 사용되지 않는 연결을 닫습니다.

- 브로커가 실행 중이지 않거나 네트워크 연결에 문제가 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

- 브로커의 기본 포트(예: 기본값 7676)로 텔넷하고 브로커가 포트 매핑 출력으로 응답하는지 확인합니다.
- 브로커 프로세스가 호스트에서 실행 중인지 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

- 브로커를 시작합니다.
- 네트워크 연결 문제를 해결합니다.
- 연결 서비스가 비활성 상태이거나 일시 중지되어 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

모든 연결 서비스의 상태를 확인합니다.

```
imqcmd list svc
```

연결 서비스 상태가 unknown 또는 paused로 표시되어 있는 경우 클라이언트가 해당 서비스를 사용하여 연결을 설정할 수 없습니다.

문제를 해결하려면 다음 작업을 수행합니다.

- 연결 서비스의 상태가 unknown으로 표시되어 있는 경우 활성 서비스 목록(imq.service.active)에서 빠진 것입니다. SSL 기반 서비스의 경우 서비스가 잘못 구성되어 있을 수도 있습니다. 이로 인해 브로커가 브로커 로그에 ERROR [B3009]: Unable to start service ssljms: [B4001]: Unable to open protocol tls for ssljms service... 라는 항목과 이 예외의 근본 원인에 대한 설명을 포함시키게 됩니다.

SSL 서비스를 올바르게 구성하려면 [219페이지의 "TCP/IP에서 SSL 기반 서비스 설정"](#)을 참조하십시오.

- 연결 서비스의 상태가 `paused`로 표시되어 있는 경우 서비스를 다시 시작합니다 (166페이지의 "연결 서비스 일시 중지 및 다시 시작" 참조).

- 필요한 연결 수에 비해 사용 가능한 스레드 수가 너무 적습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

브로커 로그에서 `WARNING [B3004]: No threads are available to process a new connection on service ... Closing the new connection.` 항목이 있는지 확인합니다.

또한 연결 서비스의 연결 수와 현재 사용 중인 스레드 수를 확인합니다.

```
imqcmd query svc -n serviceName
또는
imqcmd metrics svc -n serviceName -m cxn
```

각 연결에는 받는 메시지와 보내는 메시지에 하나씩 두 개의 스레드가 필요합니다 (56페이지의 "스레드 풀 관리자" 참조).

문제를 해결하려면 다음 작업을 수행합니다.

- 전용 스레드 풀 모델을 사용하고 있는 경우(`imq.service_name.threadpool_model=dedicated`), 최대 연결 수는 스레드 풀의 최대 스레드 수의 반입니다. 따라서 연결 수를 늘리려면 스레드 풀의 크기를 늘리거나 (`imq.service_name.max_threads`) 공유 스레드 풀 모델로 전환합니다.
- 공유 스레드 풀 모델을 사용 중인 경우(`imq.service_name.threadpool_model=shared`), 최대 연결 수는 연결 모니터 제한 (`imq.service_name.connectionMonitor_limit`)과 최대 스레드 수 (`imq.service_name.max_threads`)의 두 가지 등록 정보를 곱한 수의 반입니다. 따라서 연결 수를 늘리려면 스레드 풀의 크기를 늘리거나 연결 모니터 제한을 늘립니다.
- 결국 지원 가능한 연결 수(또는 연결의 처리량)가 입출력 제한에 도달합니다. 그런 경우 다중 브로커 클러스터(140페이지의 "클러스터를 이용한 작업(엔터프라이즈판)" 참조)를 사용하여 클러스터 내의 브로커 인스턴스로 연결을 분산합니다.
- Solaris나 Linux 플랫폼에서 필요한 연결 수에 비해 파일 설명자가 너무 적습니다 (338페이지의 "OS 정의 파일 설명자 제한" 참조).

문제의 원인을 확인하려면 다음 작업을 수행합니다.

브로커 로그에 Too many open files와 유사한 항목이 있는지 검사합니다.

문제를 해결하려면 다음 작업을 수행합니다.

ulimit 설명서 페이지에 설명되어 있는 대로 파일 설명자 제한을 늘립니다.

- TCP 백로그가 동시에 설정할 수 있는 새 연결 요청 수를 제한합니다.

TCP 백로그는 포트 매퍼가 추가 요청을 거부하기 전에 시스템 백로그 (img.portmapper.backlog)에 저장할 수 있는 동시 연결 요청 수를 제한합니다 (Windows 플랫폼의 경우 하드 코드된 백로그 제한이 있으며 Windows 데스크탑의 경우 5이고 Windows 서버의 경우 200임).

백로그 제한으로 인한 요청 거부는 비정상적으로 많은 동시 연결 요청 수로 인해 발생하는 일시적인 현상입니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

브로커 로그에서 거의 같은 시간에 받아들여지는 연결 요청이 있는 반면 거부되는 연결 요청도 있는지 확인합니다. 거부된 연결 요청은 java.net.ConnectException: Connection refused를 반환합니다.

문제를 해결하려면 다음 작업을 수행합니다.

다음 방법을 사용하여 TCP 백로그 제한을 해결할 수 있습니다.

- 클라이언트가 시도했던 연결을 잠시 후에 다시 시도하도록 프로그래밍합니다(이 문제가 본래 일시적이기 때문에 이렇게 하면 대개 제대로 작동됨).
- img.portmapper.backlog 값을 늘립니다.
- 클라이언트가 너무 자주 연결을 닫은 다음 열고 있지는 않은지 확인합니다.
- 운영 체제가 동시 연결 수를 제한합니다.

Windows 운영 체제 사용권에서는 지원되는 동시 원격 연결 수를 제한합니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

연결에 사용할 수 있는 스레드가 충분한지 확인하고(imqcmd query svc 사용) Windows 사용권 계약 조항을 확인합니다. 로컬 클라이언트에서는 연결할 수 있지만 원격 클라이언트에서는 연결할 수 없는 경우 운영 체제 제한이 문제의 원인일 수 있습니다.

문제를 해결하려면 다음 작업을 수행합니다.

- 더 많은 연결을 허용하도록 Windows 사용권을 업그레이드합니다.
- 다중 브로커 클러스터를 설정하여 연결을 여러 브로커 인스턴스에 분산합니다.
- 사용자의 인증 또는 권한 부여가 실패합니다.

사용자 저장소에 사용자에 대한 항목이 없거나 사용자가 연결 서비스에 대한 액세스 권한이 없기 때문에 잘못된 비밀번호로 인해 인증이 실패할 수 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

브로커 로그의 항목에 Forbidden 오류 메시지가 있는지 확인합니다. 이 메시지는 인증 오류를 나타낼 뿐 그 원인은 나타내지 않습니다.

- 파일 기반 사용자 저장소를 사용하는 경우 다음 명령을 입력합니다.

```
imqusermgr list -i instanceName -u userName
```

출력에 사용자가 표시되는 경우 잘못된 비밀번호가 제출된 것일 수 있습니다. 출력에 **Error [B3048]: User does not exist in the password file**이 표시되는 경우 사용자 저장소에 항목이 없는 것입니다.

- LDAP 서버 사용자 저장소를 사용 중인 경우 적절한 도구를 사용하여 사용자 항목이 있는지 확인합니다.
- 액세스 제어 등록 정보에 연결 서비스 액세스에 대한 제한이 있는지 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

- 사용자 저장소에 사용자 항목이 없는 경우 사용자 저장소에 사용자를 추가합니다 (207페이지의 "사용자 저장소 채우기 및 관리" 참조).
- 잘못된 비밀번호가 사용된 경우 올바른 비밀번호를 제공합니다.
- 액세스 제어 등록 정보가 잘못 설정된 경우 액세스 제어 등록 정보 파일을 편집하여 연결 서비스 권한을 부여합니다(216페이지의 "연결 액세스 제어" 참조).

문제: 연결 처리량이 너무 느림

증상:

- 메시지 처리량이 기대에 미치지 못합니다.
- 브로커에 대해 지원되는 연결 수가 [264페이지의 "문제: 클라이언트가 연결을 설정할 수 없음"](#)에 설명되어 있는 대로 제한되는 것이 아니라 메시지 입출력 속도에 의해 제한됩니다.

가능한 원인:

- 네트워크 연결 또는 WAN이 너무 느립니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

네트워크를 핑하여 핑이 반환되는 데 걸리는 시간을 확인한 다음 네트워크 관리자에게 문의합니다. 또는 로컬 클라이언트를 사용하여 메시지를 보내고 받은 다음 이 전달 시간을 네트워크 링크를 사용하는 원격 클라이언트의 전달 시간과 비교할 수 있습니다.

문제를 해결하려면 다음 작업을 수행합니다.

연결이 너무 느린 경우 네트워크 링크를 업그레이드합니다.

- 연결 서비스 프로토콜이 기본적으로 TCP에 비해 느립니다. 예를 들어 SSL 기반 또는 HTTP 기반 프로토콜이 TCP보다 느립니다([242페이지의 그림 9-5](#) 참조).

문제의 원인을 확인하려면 다음 작업을 수행합니다.

SSL 기반 또는 HTTP 기반 프로토콜을 사용하는 경우 TCP를 사용해보고 전달 시간을 비교합니다.

문제를 해결하려면 다음 작업을 수행합니다.

일반적으로 응용 프로그램 요구 사항에 따라 사용되는 프로토콜이 지정되므로 [283페이지의 "전송 프로토콜 조정"](#)에 설명되어 있는 대로 프로토콜을 조정해보는 것 이외에 할 수 있는 작업이 거의 없습니다.

- 연결 서비스 프로토콜이 최적으로 조정되어 있지 않습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

프로토콜을 조정해보고 차이가 있는지 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

283페이지의 "전송 프로토콜 조정"에 설명되어 있는 대로 프로토콜을 조정해봅니다.

- 메시지가 너무 커서 너무 많은 대역폭을 사용합니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

작은 크기의 메시지를 사용하여 벤치마크를 실행해 봅니다.

문제를 해결하려면 다음 작업을 수행합니다.

- `java.util.zip`을 사용하여 메시지 본문을 압축합니다.
- 메시지를 보낼 데이터의 알림으로 사용하고 데이터는 다른 프로토콜을 사용하여 이동합니다.
- 느린 연결 처리량으로 보이는 것이 실제로는 메시지 전달 프로세스의 어떤 단계에 병목 현상이 있는 것입니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

위의 항목 중 어떤 것도 느린 연결 처리량으로 보이는 문제의 원인이 아닌 것 같으면 231페이지의 **그림 9-1**을 참조하여 다른 병목 현상이 있을 수 있는지 확인하고 다음 문제와 관련된 증상이 있는지 확인합니다.

- 272페이지의 "문제: 메시지 생성이 지연되거나 느림"
- 275페이지의 "문제: 메시지가 메시지 서버에서 백로그됨"
- 279페이지의 "문제: 메시지 서버 처리량이 일정하지 않음"

문제를 해결하려면 다음 작업을 수행합니다.

위의 문제 해결 절에 제공되어 있는 문제 해결 지침을 따릅니다.

문제: 클라이언트가 메시지 생성자를 만들 수 없음

증상:

- 대상에 대해 메시지 생성자를 만들 수 없기 때문에 클라이언트에서 예외가 발생합니다.

가능한 원인:

- 대상이 제한된 생성자 수만 허용하도록 구성되었습니다.

대상에 메시지가 누적되는 것을 방지하는 방법 중 하나는 대상에서 지원할 수 있는 생성자의 수(maxNumProducers)를 제한하는 것입니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

대상을 확인합니다(173페이지의 "대상 정보 표시" 참조).

```
imqcmd query dst
```

출력에 현재 생성자 수와 maxNumProducers 값이 표시됩니다. 두 값이 같은 경우 생성자의 수가 구성된 제한에 도달한 것입니다. 새 생성자가 브로커에서 거부되는 경우 브로커는 ResourceAllocationException [C4088]: A JMS destination limit was reached를 반환하고 브로커 로그에 [B4183]: Producer can not be added to destination이라는 항목을 만듭니다.

문제를 해결하려면 다음 작업을 수행합니다.

maxNumProducers 속성 값을 늘립니다(174페이지의 "대상 속성 업데이트" 참조).

- 액세스 제어 등록 정보 파일의 설정으로 인해 사용자가 메시지 생성자를 만들 수 있는 권한이 없습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

새 생성자가 브로커에서 거부되는 경우 브로커는 JMSSecurityException [C4076]: Client does not have permission to create producer on destination을 반환하고 브로커 로그에 [B2041]: Producer on destination denied 및 [B4051]: Forbidden guest라는 항목을 만듭니다.

문제를 해결하려면 다음 작업을 수행합니다.

사용자가 메시지를 생성할 수 있도록 액세스 제어 등록 정보를 변경합니다(216페이지의 "대상 액세스 제어" 참조).

문제: 메시지 생성이 지연되거나 느림

증상:

- 지속성 메시지를 보낼 때 `send()` 메소드가 반환되지 않고 클라이언트가 차단됩니다.
- 지속성 메시지를 보낼 때 클라이언트에 예외가 발생합니다.
- 생성자 클라이언트가 느려집니다.

가능한 원인:

- 메시지 서버가 백로그되고(메시지가 브로커 메모리에 누적) 느린 메시지 생성자로 응답했습니다.

대상 메모리의 메시지 수와 메시지 바이트 수가 구성된 제한에 도달하면 브로커가 지정된 제한 동작에 따라 메모리 자원을 절약하려고 합니다. 다음 제한 동작은 메시지 생성자를 느리게 만듭니다.

- `FLOW_CONTROL`: 브로커가 지속성 메시지의 수신을 바로 확인하지 않습니다. 따라서 생성자 클라이언트가 차단됩니다.
- `REJECT_NEWEST`: 브로커가 새 메시지를 거부합니다. 거부된 각 지속성 메시지에 대해 예외가 발생합니다.

마찬가지로 모든 대상의 브로커 전체 메모리에서 메시지 수나 메시지 바이트 수가 구성된 제한에 도달하면 브로커가 최신 메시지를 거부하여 메모리 자원을 절약하려고 합니다.

또한 대상이나 브로커 전체 제한이 제대로 설정되어 있지 않아 시스템 메모리 제한에 도달하면 브로커는 메시지 생성자를 억제하는 등 메모리 오버로드를 막기 위해 점점 더 중대한 조치를 취합니다.

이러한 메커니즘에 대한 자세한 내용은 [61페이지의 "메모리 자원 및 메시지 흐름 관리"](#)를 참조하십시오.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

구성된 메시지 제한으로 인해 메시지가 브로커에서 거부되면 브로커는 `JMSEException [C4036]: A server error occurred`를 반환하고 브로커 로그에 `WARNING [B2011]: Storing of JMS message from IMQconn failed`라는 항목을 만든 다음 제한에 도달했음을 나타내는 메시지를 표시합니다.

- 메시지 제한이 대상에 있는 경우 브로커는 `[B4120]: Can not store message on destination destName because capacity of maxNumMsgs would be exceeded`와 같은 항목을 만듭니다.
- 메시지 제한이 브로커 전체에 있는 경우 브로커는 `[B4024]: The Maximum Number of messages currently in the system has been exceeded, rejecting message`와 같은 항목을 만듭니다.

좀 더 일반적으로는 대상과 브로커를 쿼리하여 각각에 구성된 메시지 제한 설정을 검사하고 적절한 `imqcmd` 명령을 사용하여 대상이나 브로커 전체에서 현재 메시지 수나 현재 메시지 바이트 수를 모니터링하여(각각 [262페이지의 표 9-10](#) 및 [258페이지의 표 9-8](#) 참조) 거부가 발생하기 전에 메시지 제한 조건을 확인할 수 있습니다.

문제를 해결하려면 다음 작업을 수행합니다.

메시지가 백로그되어 생성자가 느려지는 것을 해결하는 데에는 다음과 같은 몇 가지 방법이 있습니다.

- 메모리 자원을 초과하지 않도록 주의하면서 대상(또는 브로커 전체)에 대한 메시지 제한을 수정합니다. 일반적으로는 브로커 전체 메시지 제한에 도달하는 일이 없도록 대상별로 메모리를 관리할 수 있습니다. 자세한 내용은 [287페이지의 "브로커 조정"](#)을 참조하십시오.
- 메시지 제한에 도달하면 메시지 생성을 느리게 하기보다 메모리에서 메시지를 버리도록 대상의 제한 동작을 변경합니다. 예를 들어, 메모리에 누적되는 메시지를 삭제하는 `REMOVE_OLDEST` 및 `REMOVE_LOW_PRIORITY` 제한 동작을 지정할 수 있습니다([171페이지의 표 6-10](#) 참조).
- 브로커가 지속성 메시지를 데이터 저장소에 저장할 수 없습니다.

브로커가 데이터 저장소에 액세스할 수 없거나 지속성 메시지를 데이터 저장소에 기록할 수 없는 경우 생성자 클라이언트가 차단됩니다. 위에 설명되어 있는 대상 또는 브로커 전체 메시지 제한에 도달한 경우에도 이러한 상태가 발생할 수 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

브로커가 데이터 저장소에 기록할 수 없는 경우 브로커는 브로커 로그에 [B2011]: Storing of JMS message from connectionID failed... 또는 [B4004]: Failed to persist message messageID... 항목 중 하나를 만듭니다

문제를 해결하려면 다음 작업을 수행합니다.

- 기본 제공 지속성의 경우 파일 기반 데이터 저장소의 디스크 공간을 늘립니다.
- JDBC 호환 데이터 저장소의 경우 플러그인 지속성이 제대로 구성되어 있는지 확인합니다(부록 B, "플러그인 지속성 설정" 참조). 제대로 구성되어 있는 경우 데이터베이스 관리자에게 문의하여 다른 데이터베이스 문제를 해결합니다.
- 브로커 확인 시간 초과가 너무 짧습니다.

느린 연결 또는 높은 CPU 사용률이나 부족한 메모리 자원 때문에 무능력해진 메시지 서버로 인해 브로커가 지속성 메시지의 수신을 확인하는 데 연결 팩토리의 `imqAckTimeout` 속성 값에서 허용하는 것보다 더 많은 시간이 필요할 수 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

`imqAckTimeout` 값이 초과되는 경우 브로커는 `JMSEException [C4000]: Packet acknowledge failed`를 반환합니다.

문제를 해결하려면 다음 작업을 수행합니다.

`imqAckTimeout` 연결 팩토리 속성 값을 변경합니다(187페이지의 "연결 팩토리 관리 대상 객체 속성" 참조).

- 생성자 클라이언트에서 JVM 제한이 발생했습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

- 클라이언트 응용 프로그램이 메모리 부족 오류를 수신하는지 확인합니다.
- `freeMemory()`, `MaxMemory()`, `totalMemory()` 같은 런타임 메소드를 사용하여 JVM 힙에서 사용 가능한 메모리를 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

JVM을 조정합니다(283페이지의 "Java 가상 머신 조정" 참조).

문제: 메시지가 메시지 서버에서 백로그됨

증상:

- 브로커(또는 특정 대상)의 메시지나 메시지 바이트 수가 시간에 따라 꾸준히 증가합니다.

메시지가 누적되고 있는지 확인하려면 브로커의 메시지나 메시지 바이트 수가 시간에 따라 어떻게 변하는지 확인하고 구성된 제한과 비교합니다. 먼저 구성된 제한을 확인합니다.

```
imqcmd query bkr
```

(주: imqcmd metrics bkr 하위 명령은 이 정보를 표시하지 않습니다.)

그런 다음 각 대상에서 메시지 누적을 확인합니다.

```
imqcmd query dst -t destType -n destName
```

또는

```
imqcmd metrics dst -t destType -n destName -m ttl
```

메시지가 대상이나 브로커 전체에 구성된 제한을 초과했는지 확인하려면 브로커 로그에서 WARNING [B2011]: Storing of JMS message from..failed라는 항목이 있는지 확인합니다. 이 항목 다음에는 초과된 제한에 대해 설명하는 다른 항목이 표시됩니다.

- 메시지 생성이 지연되거나 생성된 메시지가 브로커에서 거부됩니다.
- 메시지가 사용자에게 도달하기까지 비정상적으로 오래 걸립니다.

가능한 원인:

- 클라이언트 코드에 결함이 있습니다. 사용자가 메시지를 확인하지 않습니다.

메시지는 메시지를 받은 모든 사용자가 확인할 때까지 대상에 보관됩니다. 따라서 클라이언트가 사용된 메시지를 확인하지 않는 경우 메시지는 삭제되지 않고 대상에 누적됩니다.

예를 들어, 클라이언트 코드에 다음 결함이 있을 수 있습니다.

- CLIENT_ACKNOWLEDGEMENT 또는 트랜잭션된 세션을 사용하는 사용자가 Session.acknowledge() 또는 Session.commit()을 정기적으로 호출하고 있지 않을 수 있습니다.

- AUTO_ACKNOWLEDGE 세션을 사용하는 사용자가 몇 가지 이유로 중지되어 있을 수 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

메시지 서버가 사용 중이지 않은 경우, 즉 대상에 메시지가 유입 및 유출되는 속도가 낮은 경우 메시지가 확인되지 않아 누적될 수 있습니다.

메시지가 브로커에 유입 및 유출되는 속도를 확인합니다.

```
imqcmd metrics bkr -m rts
```

그런 다음 개별 대상 각각에 대한 흐름 속도를 확인합니다.

```
imqcmd metrics bkr -t destType -n destName -m rts
```

또한 클라이언트 코드를 확인하여 메시지가 제대로 확인되고 있는지 확인합니다.

- 주제 대상에 비활성 영구 가입이 있습니다.

영구 가입이 비활성 상태인 경우 해당 사용자가 활성화되어 메시지를 사용할 수 있을 때까지 대상에 메시지가 저장됩니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

각 주제 대상에서 영구 가입 상태를 확인합니다.

```
imqcmd list dur -d destName
```

문제를 해결하려면 다음 작업을 수행합니다.

다음 작업 중 하나를 수행할 수 있습니다.

- 문제가 있는 영구 가입의 모든 메시지를 제거합니다(179페이지의 "영구 가입 관리" 참조).
- 주제에 대해 메시지 제한 및 제한 동작 속성을 지정합니다(171페이지의 표 6-10 참조). 예를 들어, 메모리에 누적되는 메시지를 삭제하는 REMOVE_OLDEST 및 REMOVE_LOW_PRIORITY 제한 동작을 지정할 수 있습니다.
- 해당 대상에서 모든 메시지를 제거합니다(176페이지의 "대상 제거" 참조).
- 메시지가 메모리에 남아 있을 수 있는 시간을 제한합니다. 생성자 클라이언트를 다시 작성하여 각 메시지의 수명 값을 설정할 수 있습니다. `imqOverrideJMSEExpiration` 및 `imqJMSEExpiration` 연결 팩토리 속성을 설정하여 연결을 공유하는 모든 생성자에 대해 이러한 설정을 대체할 수 있습니다 (187페이지의 표 7-3 참조).

- 대기열의 메시지를 사용할 수 있는 사용자 수가 너무 적습니다.

메시지를 전달할 수 있는 활성 사용자의 수가 너무 적은 경우 메시지가 누적될 때 대기열 대상이 백로그될 수 있습니다. 이런 상태는 다음과 같은 이유 중 하나 때문에 발생할 수 있습니다.

- 대상에 활성 사용자가 수가 너무 적습니다.
- 사용자 클라이언트가 연결을 설정하지 못했습니다.
- 대기열의 메시지에 일치하는 선택기를 사용하는 활성 사용자가 없습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

사용자가 대기열의 메시지를 사용할 수 없는 원인을 확인하려면 대상에서 활성 사용자의 수를 확인합니다.

```
imqcmd metrics dst -n destName -t q -m con
```

문제를 해결하려면 다음 작업을 수행합니다.

사용자가 대기열의 메시지를 사용할 수 없는 원인에 따라 다음 작업 중 하나를 수행할 수 있습니다.

- 추가 사용자 클라이언트를 시작하여 대기열에 더 많은 활성 사용자를 만듭니다.
- `imq.consumerFlowLimit` 브로커 등록 정보를 조정하여 여러 사용자에 대한 대기열 전달을 최적화합니다(288페이지의 "다중 사용자 대기열 성능" 참조).
- 대기열에 대해 메시지 제한 및 제한 동작 속성을 지정합니다(171페이지의 표 6-10 참조). 예를 들어, 메모리에 누적되는 메시지를 삭제하는 `REMOVE_OLDEST` 및 `REMOVE_LOW_PRIORITY` 제한 동작을 지정할 수 있습니다.
- 해당 대상에서 모든 메시지를 제거합니다(176페이지의 "대상 제거" 참조).
- 메시지가 메모리에 남아 있을 수 있는 시간을 제한합니다. 생성자 클라이언트를 다시 작성하여 각 메시지의 수명 값을 설정할 수 있습니다. `imqOverrideJMSEExpiration` 및 `imqJMSEExpiration` 연결 팩토리 속성을 설정하여 연결을 공유하는 모든 생성자에 대해 이러한 설정을 대체할 수 있습니다(187페이지의 표 7-3 참조).
- 메시지 사용자가 너무 느리게 처리하여 메시지 생성자를 따라가지 못합니다.

이 경우 주제 가입자나 대기열 수신자가 메시지를 사용하는 것이 생성자가 메시지를 보내는 것보다 느립니다. 이러한 불균형으로 인해 하나 이상의 대상이 메시지로 백로 그됩니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

메시지가 브로커에 유입 및 유출되는 속도를 확인합니다.

```
imqcmd metrics bkr -m rts
```

그런 다음 개별 대상 각각에 대한 흐름 속도를 확인합니다.

```
imqcmd metrics bkr -t destType -n destName -m rts
```

문제를 해결하려면 다음 작업을 수행합니다.

- 사용자 클라이언트 코드를 최적화합니다.
- 대기열 대상의 경우 활성 사용자의 수를 늘립니다(288페이지의 "다중 사용자 대기열 성능" 참조).
- 클라이언트 확인 처리가 메시지 사용을 느리게 합니다.

두 가지 요소가 클라이언트 확인의 처리에 영향을 미칩니다.

- 상당한 브로커 자원이 클라이언트 확인 처리에 사용될 수 있습니다. 따라서 브로커가 클라이언트 확인을 확인할 때까지 사용자 클라이언트가 차단되는 확인 모드에서는 메시지 사용이 느려질 수 있습니다.
- JMS 페이로드 메시지와 Message Queue 제어 메시지(예: 클라이언트 확인)는 같은 연결을 공유합니다. 따라서 JMS 페이로드 메시지에 의해 제어 메시지가 일시적으로 중단되어 메시지 사용이 느려질 수 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

패킷 흐름을 기준으로 메시지 흐름을 확인합니다. 초당 패킷 수와 메시지 수의 비율이 맞지 않으면 클라이언트 확인에 문제가 있을 수 있습니다.

또는 클라이언트가 JMSException [C4000]: Packet acknowledge failed 메시지를 받았는지 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

- 클라이언트가 사용하는 확인 모드를 수정합니다. 예를 들어 DUPS_OK_ACKNOWLEDGEMENT나 CLIENT_ACKNOWLEDGEMENT로 전환합니다.
- CLIENT_ACKNOWLEDGEMENT 또는 트랜잭션된 세션을 사용 중인 경우 많은 수의 메시지를 하나의 확인으로 그룹화합니다.

- 사용자 및 연결 흐름 제어 매개 변수를 조정합니다(289페이지의 "클라이언트 런타임 메시지 흐름 조정" 참조).
- 브로커가 생성된 메시지를 따라갈 수 없습니다.

이 경우 메시지가 브로커로 유입되는 것이 브로커가 메시지를 라우팅하여 사용자에게 전달하는 것보다 빠릅니다. 브로커의 지체는 CPU, 네트워크 소켓 읽기/쓰기 작업, 디스크 읽기/쓰기 작업, 메모리 페이징, 영구 저장소, JVM 메모리 제한 중 하나나 모두에 대한 제한 때문일 수 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

이 문제를 일으키는 다른 원인은 없는지 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

 - 컴퓨터나 데이터 저장소의 속도를 업그레이드합니다.
 - 브로커 클러스터를 사용하여 로드를 여러 브로커 인스턴스에 분산합니다.

문제: 메시지 서버 처리량이 일정하지 않음

증상:

- 메시지 처리량이 산발적으로 떨어지다가 정상 성능을 회복합니다.

가능한 원인:

- 브로커의 메모리 자원이 매우 적습니다.

대상 및 브로커 제한이 제대로 설정되어 있지 않기 때문에 브로커가 메모리 오버로드를 막기 위해 점점 더 중대한 조치를 취하게 되어 메시지 백로그를 지울 때까지 브로커가 아주 느려질 수 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

브로커 로그에서 부족한 메모리 상태([B1089]: In low memory condition, broker is attempting to free up resources)와 새 메모리 상태 및 사용 중인 전체 메모리 양에 대해 설명하는 항목이 있는지 확인합니다.

또한 JVM 힙에서 사용 가능한 메모리를 확인합니다.

```
imgcmd metrics bkr -m cxn
```

전체 JVM 메모리의 값이 최대 JVM 메모리 값에 근접한 경우 사용 가능한 메모리가 적습니다.

문제를 해결하려면 다음 작업을 수행합니다.

- JVM을 조정합니다(283페이지의 "Java 가상 머신 조정" 참조).
- 시스템 스왑 공간을 늘립니다.
- JVM 메모리 재생 이용(가비지 컬렉션)이 발생합니다.

메모리를 확보하기 위해 메모리 재생 이용이 주기적으로 시스템을 정리합니다. 이럴 경우 모든 스레드가 차단됩니다. 확보할 메모리 양이 크고 JVM 힙 크기가 클수록 메모리 재생 이용으로 인한 지체가 길어집니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

컴퓨터의 CPU 사용을 모니터링합니다. 메모리 재생 이용이 발생하면 CPU 사용이 크게 떨어집니다.

또는 다음 명령줄 옵션을 사용하여 브로커를 시작합니다.

```
-vmargs -verbose:gc
```

표준 출력에 메모리 재생 이용이 발생하는 시간이 표시됩니다.

문제를 해결하려면 다음 작업을 수행합니다.

다중 CPU 컴퓨터인 경우 메모리 재생 이용이 병렬로 발생되도록 설정합니다.

```
-XX:+UseParallelGC=true
```

- JVM이 성능을 높이기 위해 JIT (Just-In-Time) 컴파일러를 사용하고 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

이 문제를 일으키는 다른 원인은 없는지 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

잠시 동안 시스템이 실행되도록 놓아두면 성능이 향상됩니다.

문제: 메시지가 사용자에게 도달하지 않음

증상:

- 생성자가 보낸 메시지를 사용자가 받지 못합니다.

가능한 원인:

- 제한 동작으로 인해 메시지가 브로커에서 삭제됩니다.

대상 메모리의 메시지 수나 메시지 바이트 수가 구성된 제한에 도달하면 브로커는 메모리 자원을 절약하려고 합니다. 이러한 제한에 도달하면 브로커가 수행하는 구성 가능한 동작 중 다음 세 가지 동작으로 인해 메시지가 손실됩니다.

- REMOVE_OLDEST: 가장 오래된 메시지를 삭제
- REMOVE_LOW_PRIORITY: 메시지의 보존 기간을 기준으로 우선 순위가 가장 낮은 메시지 삭제
- REJECT_NEWEST: 새 메시지 거부(거부된 지속성 메시지에 대해 예외 발생)

브로커 메모리의 메시지 수나 메시지 바이트 수가 구성된 제한에 도달하면 브로커는 최신 메시지를 거부하여 메모리 자원을 절약하려고 합니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

브로커 로그에서 WARNING [B2011]: Storing of JMS message from..failed 항목이 있는지 확인합니다. 이 항목 다음에는 초과된 제한에 대해 설명하는 다른 항목이 표시됩니다. 하지만 메시지 삭제를 나타내는 항목은 없습니다.

문제를 해결하려면 다음 작업을 수행합니다.

제한을 변경하거나 동작을 변경합니다.

- 메시지 시간 초과 값이 만료됩니다.

브로커는 시간 초과 값이 만료된 메시지를 삭제합니다. 대상이 충분히 메시지로 백로그된 경우 수명 값이 너무 짧은 메시지는 삭제될 수 있습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

브로커 로그 파일에서 Expiring Messages: Expired *n* messages라는 항목이 있는지 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

대체합니다.

- 다른 컴퓨터 간에 클럭 시간이 동기화되지 않습니다.

클럭이 동기화되어 있지 않은 경우 브로커의 메시지 수명 계산에 오류가 생길 수 있으므로 메시지가 만료 시간을 초과하여 삭제됩니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

모든 컴퓨터의 클럭을 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

클럭을 동기화합니다(337페이지의 "시스템 클럭 설정" 참조).

- 사용자 클라이언트가 연결에서 메시지 전달을 시작하지 못했습니다.

클라이언트 코드가 연결을 설정하고 해당 연결에서 메시지 전달을 시작할 때까지 메시지를 전달할 수 없습니다.

문제의 원인을 확인하려면 다음 작업을 수행합니다.

클라이언트 코드가 연결을 설정하고 메시지 전달을 시작하는지 확인합니다.

문제를 해결하려면 다음 작업을 수행합니다.

연결을 설정하고 메시지 전달을 시작하도록 클라이언트 코드를 다시 작성합니다.

성능 향상을 위한 구성 조정

시스템 조정

다음 절에서는 운영 체제, JVM 및 통신 프로토콜에서 조정할 수 있는 내용을 설명합니다.

Solaris 조정: CPU 사용률, 페이징/스왑/디스크 입출력

운영 체제 조정에 대해서는 시스템 설명서를 참조하십시오.

Java 가상 머신 조정

기본적으로 브로커는 192MB의 JVM 힙 크기를 사용합니다. 이 크기는 상당한 메시지 로드에는 비해 너무 작은 경우가 많으므로 늘리는 것이 좋습니다.

브로커가 Java 객체에서 사용되는 JVM 힙 공간을 소진하는 데 이르면 브로커는 흐름 제어, 메시지 스왑 등 여러 기술을 사용하여 메모리를 확보합니다. 아주 드물게 메모리를 확보하고 메시지 유입을 줄이기 위해 클라이언트 연결을 닫는 경우도 있습니다. 따라서, 이러한 경우가 발생하지 않도록 최대 JVM 힙 공간을 충분하게 설정하는 것이 좋습니다.

하지만, 최대 Java 힙 공간을 너무 높게 설정하면 브로커가 전체 시스템 메모리가 부족해 질 때까지 Java 힙 공간을 계속해서 증가시킬 수 있습니다. 그렇게 되면 성능이 감소하거나, 예상치 않은 브로커 충돌이 발생하거나, 시스템에서 실행 중인 다른 응용 프로그램 및 서비스의 동작에 영향을 미칠 수 있습니다. 일반적으로 운영 체제 및 시스템에서 실행할 다른 응용 프로그램이 충분한 물리적 메모리를 사용할 수 있도록 해야 합니다.

일반적으로 정상 시스템 메모리와 최대 시스템 메모리를 평가하여 시스템 메모리 문제를 일으키지 않고 우수한 성능을 제공할 수 있도록 Java 힙 크기를 구성하는 것이 좋습니다.

브로커의 최소 및 최대 힙 크기를 변경하려면 브로커를 시작할 때 `-vmargs` 명령줄 옵션을 사용합니다. 예를 들면 다음과 같습니다.

```
/usr/bin/imqbrokerd -vmargs "-Xms256m -Xmx1024m"
```

이 명령은 시작 Java 힙 크기를 256MB로 설정하고 최대 Java 힙 크기를 1GB로 설정합니다.

- Solaris에서 `/etc/rc` (즉, `/etc/init.d/imq`)를 통해 브로커를 시작하는 경우 `/etc/imq/imqbrokerd.conf` 파일에 브로커 명령줄 인수를 지정합니다. 자세한 내용은 해당 파일의 주석을 참조하십시오.
- Windows에서 브로커를 Windows 서비스로 시작하는 경우 `imqsvcadm install` 명령에 `-vmargs` 옵션을 사용하여 JVM 인수를 지정합니다. [334페이지의 "서비스 관리자 유틸리티\(imqsvcadm\)"](#)를 참조하십시오.

어떤 경우든 브로커 로그 파일을 확인하거나

`imqcmd metrics bkr -m cxn` 명령을 사용하여 설정을 확인합니다.

전송 프로토콜 조정

응용 프로그램 요구에 맞는 프로토콜을 선택했으면 선택한 프로토콜을 기반으로 추가 조정하여 성능을 향상시킬 수 있습니다.

다음 세 가지 브로커 등록 정보를 사용하여 프로토콜의 성능을 수정할 수 있습니다.

- `imq.protocol protocol_type nodelay`
- `imq.protocol protocol_type inbufsz`
- `imq.protocol protocol_type outbufsz`

TCP 및 SSL 프로토콜의 경우 이러한 등록 정보는 클라이언트와 브로커 사이의 메시지 전달 속도에 영향을 미칩니다. HTTP 및 HTTPS 프로토콜의 경우 이 등록 정보는 Web Server에서 실행 중인 Message Queue 터널 서블릿과 브로커 사이의 메시지 전달 속도에 영향을 미칩니다. HTTP/HTTPS 프로토콜의 경우 성능에 영향을 미칠 수 있는 추가 등록 정보가 있습니다(286페이지의 "[HTTP/HTTPS 조정](#)" 참조).

프로토콜 조정 등록 정보에 대해서는 다음 절에서 설명합니다.

nodelay

`nodelay` 등록 정보는 지정된 프로토콜의 Nagle 알고리즘(TCP/IP에서 TCP_NODELAY 소켓 수준 옵션 값)에 영향을 미칩니다. Nagle 알고리즘은 WAN (Wide-Area Network) 같은 느린 연결을 사용하는 시스템에서 TCP 성능을 향상시키는 데 사용됩니다.

이 알고리즘이 사용되면 TCP는 여러 개의 작은 데이터 청크가 원격 시스템으로 보내지지 않도록 데이터를 큰 패킷으로 묶으려고 합니다. 소켓에 기록된 데이터가 필요한 버퍼 크기를 채우지 못하는 경우 프로토콜은 버퍼가 채워지거나 특정 지연 시간이 경과될 때까지 패킷 전송을 지연합니다. 버퍼가 채워지거나 시간 초과가 발생하면 패킷을 보냅니다.

대부분의 메시징 응용 프로그램에서 패킷을 지연 없이 보내는 경우(Nagle 알고리즘을 사용하지 않는 경우) 최고의 성능을 냅니다. 이는 클라이언트와 브로커 사이의 상호 작용이 대부분 클라이언트가 데이터 패킷을 브로커에게 보내고 응답을 기다리는 요청/응답 상호 작용이기 때문입니다. 예를 들어, 일반적인 상호 작용에는 다음이 포함됩니다.

- 연결 만들기
- 생성자 또는 사용자 만들기
- 지속성 메시지 보내기(브로커가 메시지 수신 확인)
- `AUTO_ACKNOWLEDGE` 또는 `CLIENT_ACKNOWLEDGE` 세션에서 클라이언트 확인 보내기(브로커가 확인 처리 확인)

이러한 상호 작용에서 대부분의 패킷은 버퍼 크기보다 작습니다. 이는 Nagle 알고리즘이 사용되는 경우 브로커가 사용자에게 응답을 보내기 전에 몇 밀리초를 지연함을 의미합니다.

하지만 Nagle 알고리즘은 연결이 느리고 브로커 응답이 필요하지 않은 경우 성능을 향상시킬 수 있습니다. 이런 것으로는 클라이언트가 비지속성 메시지를 보내는 경우나 클라이언트 확인을 브로커에서 확인하지 않는 경우(DUPS_OK_ACKNOWLEDGE 세션)를 들 수 있습니다.

inbufsz/outbufsz

inbufsz 등록 정보는 소켓에서 들어오는 데이터를 읽는 입력 스트림의 버퍼 크기를 설정합니다. 마찬가지로 outbufsz는 데이터를 소켓에 기록하기 위해 브로커가 사용하는 출력 스트림의 버퍼 크기를 설정합니다.

일반적으로 두 매개 변수 모두 보내거나 받는 평균 패킷보다 약간 큰 값으로 설정해야 합니다. 경험에 따르면 이 등록 정보 값을 평균 패킷에 1k를 더한 크기(가장 가까운 k 값으로 반올림)로 설정하는 것이 좋습니다.

예를 들어, 브로커가 본문 크기가 1k인 패킷을 받는 경우 패킷의 전체 크기(메시지 본문 + 헤더 + 등록 정보)는 약 1200바이트입니다. inbufsz가 2k (2048바이트)인 경우 적당한 성능을 제공합니다.

inbufsz나 outbufsz를 이 크기보다 더 키우면 성능은 약간 늘어날 수 있지만 각 연결에 필요한 메모리가 늘어나게 됩니다.

그림 9-6은 1k 패킷에 대해 inbufsz를 변경할 때의 결과를 보여 줍니다.

그림 9-7 1k (1024바이트) 패킷에 대한 inbufsz 변경의 결과

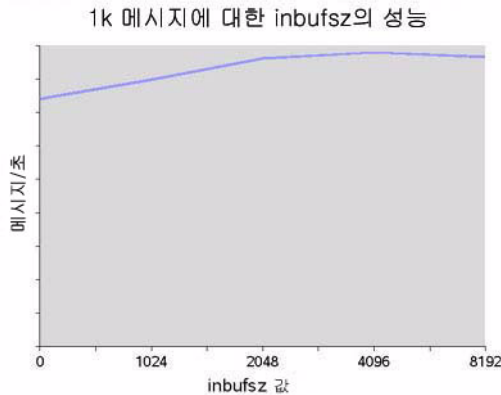
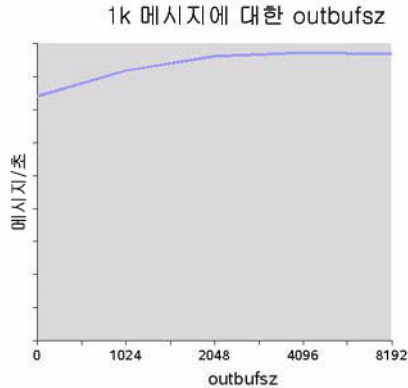


그림 9-8은 1k 패킷에 대해 outbufsz를 변경할 때의 결과를 보여 줍니다.

그림 9-8 1k(1024바이트) 패킷에 대한 outbufsz 변경의 결과



HTTP/HTTPS 조정

앞의 두 절에서 설명한 일반 등록 정보 이외에도 HTTP/HTTPS 성능은 클라이언트가 Message Queue 터널 서블릿을 호스트하는 Web Server에 대해 HTTP 요청을 얼마나 빨리 보낼 수 있는지에 따라 제한됩니다.

단일 소켓에서 다중 요청을 처리하도록 Web Server를 최적화해야 할 수 있습니다. JDK 버전 1.4 이상에서는 Web Server에 대한 HTTP 연결이 유지되어(Web Server 소켓이 열린 상태로 유지) 다중 HTTP 요청을 처리할 때 Web Server에서 사용되는 자원을 최소화합니다. JDK 버전 1.4를 사용하는 클라이언트 응용 프로그램의 성능이 이전 JDK 릴리스로 실행 중인 같은 응용 프로그램보다 느린 경우 성능을 향상시키기 위해 Web Server의 연결 유지 구성 매개 변수를 조정해야 할 수 있습니다.

이러한 Web Server 조정과 더불어 클라이언트가 Web Server를 풀하는 간격도 조정할 수 있습니다. HTTP는 요청 기반 프로토콜입니다. 따라서 HTTP 기반 프로토콜을 사용하는 클라이언트는 Web Server에서 메시지가 대기 중인지 주기적으로 확인해야 합니다.

imq.httpjms.http.pullPeriod 브로커 등록 정보(및 해당

imq.httpsjms.https.pullPeriod 등록 정보)는 Message Queue 클라이언트 런타임이 Web Server를 풀하는 간격을 지정합니다.

pullPeriod 값이 -1(기본값)인 경우 클라이언트 런타임은 이전 요청이 반환되자마자 서버를 풀하여 개별 클라이언트의 성능을 최대화합니다. 따라서 각 클라이언트 연결이 Web Server의 요청 스레드를 독점하여 Web Server 자원이 고갈될 수 있습니다.

pullPeriod 값이 양수인 경우 클라이언트 런타임은 주기적으로 Web Server에 요청을 보내 보류 중인 데이터가 있는지 확인합니다. 이 경우 클라이언트가 Web Server의 요청 스레드를 독점하지 않습니다. 따라서 많은 수의 클라이언트가 Web Server를 사용 중인 경우 pullPeriod를 양수로 설정하면 Web Server 자원을 절약할 수 있습니다.

파일 기반 영구 저장소 조정

파일 기반 영구 저장소 조정에 대한 자세한 내용은 [64페이지의 "기본 제공 지속성"](#)을 참조하십시오.

브로커 조정

다음 절에서는 브로커 등록 정보를 조정하여 성능을 향상시키는 방법에 대해 설명합니다.

메모리 관리: 로드 하에서 브로커 안정성 증가

대상별 또는 시스템 전체 수준(모든 대상을 묶어서)에서 메모리 관리를 구성할 수 있습니다.

대상 제한 사용

대상 제한에 대한 자세한 내용은 [168페이지의 "대상 관리"](#)를 참조하십시오.

시스템 전체 제한 사용

메시지 생성자가 메시지 사용자보다 넘치는 경향이 있는 경우 메시지가 브로커에 누적될 수 있습니다. 브로커에는 메모리가 부족한 경우 생성자를 억제하고 활성 메모리로부터 메시지를 스왑하는 메커니즘이 포함되어 있지만([61페이지의 "메모리 자원 및 메시지 흐름 관리"](#) 참조) 브로커가 보관할 수 있는 전체 메시지(및 메시지 바이트) 수에 대해 엄격한 제한을 설정하는 것이 좋습니다.

`imq.system.max_count` 및 `imq.system.max_size` 브로커 등록 정보를 설정하여 이러한 제한을 제어합니다. 브로커 등록 정보 설정에 대한 자세한 내용은 [129페이지의 "인스턴스 구성 파일 편집"](#) 또는 [136페이지의 "imqbrokerd 옵션 요약"](#)을 참조하십시오.

예를 들면 다음과 같습니다.

```
imq.system.max_count=5000
```

위에서 정의한 값은 브로커가 전달되지 않은/확인되지 않은 메시지를 5000개까지만 보관함을 의미합니다. 추가로 보내는 메시지는 브로커에서 거부됩니다. 메시지가 지속성인 경우 생성자가 메시지를 보내려고 하면 예외가 발생합니다. 메시지가 비지속성인 경우 브로커는 메시지를 자동으로 삭제합니다.

비지속성 메시지가 지속성 메시지처럼 예외를 반환하게 하려면 클라이언트가 사용하는 연결 팩토리 객체에 대해 다음 등록 정보를 설정합니다.

```
imqAckOnProduce = true
```

위의 설정은 비지속성 메시지를 브로커로 보낼 때의 성능을 저하시킬 수 있지만(클라이언트가 다음 메시지를 보내기 전에 응답을 기다림) 브로커로의 메시지 유입은 일반적으로 시스템 병목 현상이 아니므로 종종 이 설정이 허용되기도 합니다.

메시지를 보낼 때 예외가 반환되면 클라이언트는 잠시 동안 일시 중지하고 다시 보내 보아야 합니다.

다중 사용자 대기열 성능

다중 대기열 사용자가 대기열 대상에서 메시지를 처리하는 효율성은 구성 가능한 대기열 대상 속성, 즉 활성 사용자의 수(maxNumActiveConsumers)와 단일 일괄 처리로 사용자에게 전달할 수 있는 최대 메시지 수(consumerFlowLimit)에 따라 달라집니다. 이 속성들에 대해서는 [171페이지의 표 6-10](#)에서 설명합니다.

최적의 메시지 처리량을 달성하려면 대기열의 메시지 생성 속도에 부합하는 충분한 활성 사용자의 수가 있어야 하며 대기열의 메시지를 그 사용 속도를 최대화할 수 있는 방식으로 라우팅한 다음 활성 사용자에게 전달해야 합니다. 여러 사용자 간에 메시지 전달의 균형을 조정하기 위한 일반 메커니즘에 대해서는 [77페이지의 "다중 사용자로의 대기열 전달"](#)에서 설명합니다.

메시지가 대기열에 누적되고 있는 경우 활성 사용자 수가 메시지 로드를 처리하기에 충분하지 않을 수 있습니다. 또는 사용자에서 메시지 정체를 일으키는 일괄 처리 크기로 메시지가 사용자에게 전달되고 있을 수 있습니다. 예를 들어 일괄 처리 크기(consumerFlowLimit)가 너무 큰 경우 한 사용자가 대기열의 모든 메시지를 받는 동안 다른 활성 사용자는 메시지를 전혀 받지 못할 수 있습니다. 사용자가 아주 빠른 경우 이것은 문제가 되지 않을 수 있습니다.

하지만 사용자가 비교적 느린 경우 메시지를 사용자에게 균등하게 분산시켜야 하므로 일괄 처리 크기가 작은 게 좋습니다. 일괄 처리 크기가 작을수록 메시지를 사용자에게 전달하는 데 더 많은 오버헤드가 필요합니다. 그럼에도 불구하고 느린 사용자의 경우 작은 일괄 처리 크기를 사용하는 것이 결과적으로 성능 향상이 있습니다.

클라이언트 런타임 메시지 흐름 조정

이 절에서는 성능에 영향을 미치는 흐름 제어 동작을 설명합니다(245페이지의 "[클라이언트 런타임 구성](#)" 참조). 이러한 동작은 연결 팩토리 관리 대상 객체의 속성으로 구성됩니다. 연결 팩토리 속성 설정에 대한 자세한 내용은 7장, "[관리 대상 객체 관리](#)"를 참조하십시오.

메시지 흐름 측정

클라이언트가 보내고 받는 메시지(JMS 메시지)와 Message Queue 제어 메시지는 같은 클라이언트-브로커 연결을 통해 전달됩니다. 제어 메시지가 JMS 메시지 전달로 인해 일시 중단되면 브로커 확인처럼 제어 메시지 전달이 지연될 수 있습니다. 이러한 유형의 혼잡을 막기 위해 Message Queue는 연결에서 JMS 메시지의 흐름을 측정합니다.

JMS 메시지는 `imqConnectionFactoryCount` 등록 정보에서 지정한 대로 설정된 수만 전달되도록 일괄 처리됩니다. 일괄 처리가 전달되면 JMS 메시지 전달이 일시 중지되고 보류 중인 제어 메시지가 전달됩니다. JMS 메시지의 다른 일괄 처리가 전달된 다음 대기열에 있던 제어 메시지가 전달되는 식으로 이러한 주기가 반복됩니다.

클라이언트가 브로커에 많은 응답을 요구하는 작업을 수행 중인 경우(예: 클라이언트가 `CLIENT_ACKNOWLEDGE` 또는 `AUTO_ACKNOWLEDGE` 모드, 지속성 메시지, 트랜잭션, 대기열 브라우저를 사용하고 있거나 클라이언트가 사용자를 추가 또는 제거하고 있는 경우) `imqConnectionFactoryCount` 값을 낮게 유지해야 합니다. 반면 클라이언트에 연결에서 `DUPS_OK_ACKNOWLEDGE` 모드를 사용하는 단순 사용자만 있는 경우 성능을 저하시키지 않고 `imqConnectionFactoryCount`를 늘릴 수 있습니다.

메시지 흐름 제한

메모리와 같은 로컬 자원 제한이 발생하기 전에 Message Queue 클라이언트 런타임이 처리할 수 있는 JMS 메시지 수에 대한 제한이 있습니다. 이 제한에 가까워지면 성능에 악영향을 줍니다. 따라서 Message Queue에서는 연결을 통해 전달되고 클라이언트 런타임에 버퍼링되어 사용 대기 중일 수 있는 사용자당 메시지(또는 연결당 메시지) 수를 제한할 수 있습니다.

사용자 기반 제한

클라이언트 런타임에 전달된 JMS 메시지 수가 사용자의 `imqConsumerFlowLimit` 값을 초과하면 해당 사용자에 대해 메시지 전달이 중지됩니다. 해당 사용자의 사용되지 않은 메시지 수가 `imqConsumerFlowThreshold`에 설정된 값 이하로 떨어지야만 메시지 전달이 재개됩니다.

다음은 이러한 제한의 사용을 보여 주는 예입니다. 주제 사용자의 기본 설정을 고려해 보십시오.

```
imqConsumerFlowLimit=1000
```

```
imqConsumerFlowThreshold=50
```

사용자가 만들어지면 브로커는 초기 일괄 처리인 1000개의 메시지(있는 경우)를 일시 중지 없이 이 사용자에게 전달합니다. 1000개의 메시지를 보낸 다음 브로커는 클라이언트 런타임이 추가 메시지를 요청할 때까지 전달을 중지합니다. 클라이언트 런타임은 응용 프로그램이 메시지를 처리할 때까지 이 메시지들을 보관합니다. 그런 다음 클라이언트 런타임은 브로커에게 다음 일괄 처리를 보내도록 요청하기 전에 응용 프로그램이 적어도 메시지 버퍼 용량(imqConsumerFlowThreshold)의 50% (즉, 500개의 메시지)를 사용할 수 있도록 합니다.

같은 상황에서 임계값이 10%인 경우 클라이언트 런타임은 다음 일괄 처리를 요청하기 전에 응용 프로그램이 최소 900개의 메시지를 사용할 때까지 기다립니다.

다음 일괄 처리 크기는 다음과 같이 계산됩니다.

imqConsumerFlowLimit - (버퍼에서 현재 보류 중인 메시지 수)

따라서 imqConsumerFlowThreshold가 50%인 경우 다음 일괄 처리 크기는 응용 프로그램의 메시지 처리 속도에 따라 500과 1000 사이에서 변동될 수 있습니다.

imqConsumerFlowThreshold가 너무 높게 설정된 경우(100% 가까이) 브로커는 더 작은 일괄 처리를 보내는 경향이 있으므로 메시지 처리량이 떨어질 수 있습니다. 이 값이 너무 낮게 설정된 경우(0% 가까이) 브로커가 다음 집합을 전달하기 전에 클라이언트가 버퍼링된 나머지 메시지를 모두 처리할 수 있으므로 메시지 처리량이 저하됩니다. 일반적으로 특정 성능이나 안정성 문제가 있는 경우가 아니라면 imqConsumerFlowThreshold 속성의 기본 값을 변경할 필요가 없습니다.

사용자 기반 흐름 제어(특히 imqConsumerFlowLimit)는 클라이언트 런타임의 메모리 관리에 가장 좋은 방법입니다. 일반적으로 클라이언트 응용 프로그램에 따라 연결에서 지원해야 할 사용자 수, 메시지 크기 및 클라이언트 런타임에서 사용할 수 있는 전체 메모리 양에 대해 알고 있습니다.

연결 기반 제한

하지만 일부 클라이언트 응용 프로그램의 경우 최종 사용자의 선택에 따라 사용자 수가 불확실할 수 있습니다. 이러한 경우에도 연결 수준 흐름 제한을 사용하여 메모리를 관리할 수 있습니다.

연결 수준 흐름 제어는 연결의 모든 사용자에게 대해 버퍼링되는 전체 메시지 수를 제한합니다. 이 수가 `imqConnectionFactoryLimit`를 초과하면 전체 수가 연결 제한 아래로 떨어질 때까지 해당 연결을 통한 메시지 전달이 중지됩니다(`imqConnectionFactoryLimitEnabled` 등록 정보를 `true`로 설정하는 경우에만 `imqConnectionFactoryLimit`를 사용할 수 있음).

세션의 대기열에 있는 메시지 수는 각 사용자의 메시지 로드와 세션을 사용하는 메시지 사용자 수의 함수입니다. 클라이언트가 메시지를 생성하거나 사용할 때 지연을 나타내는 경우 일반적으로 응용 프로그램을 재설계하여 메시지 생성자와 사용자를 여러 세션에 분산시키거나 세션을 여러 연결에 분산시킴으로써 성능을 향상시킬 수 있습니다.

성능 향상을 위한 구성 조정

Message Queue 데이터의 위치

Sun Java System Message Queue에서는 많은 데이터 범주를 사용하며, 각 범주는 다음 절에 표시된 것처럼 운영 체제에 따라 다른 위치에 저장됩니다. 다음에 나오는 표에서 *instanceName*은 데이터와 관련된 브로커 인스턴스의 이름을 식별합니다.

Solaris

표 A-1에서는 Solaris 플랫폼에서 Message Queue 데이터의 위치를 보여 줍니다.

주 Solaris에서 Sun Java System Application Server와 함께 제공되는 Message Queue의 데이터 위치는 295페이지의 표 A-3에 나와 있습니다.

표 A-1 Solaris에서 Message Queue 데이터의 위치

데이터 범주	Solaris에서 위치
브로커 인스턴스 구성 등록 정보	<code>/var/imq/instances/<i>instanceName</i>/props/config.properties</code>
브로커 구성 파일 템플릿	<code>/usr/share/lib/imq/props/broker/</code>
영구 저장소(메시지, 대상, 영구 가입, 트랜잭션)	<code>/var/imq/instances/<i>instanceName</i>/fs350/</code> 또는 JDBC 액세스 가능 데이터 저장소
브로커 인스턴스 로그 파일 디렉토리(기본 위치)	<code>/var/imq/instances/<i>instanceName</i>/log/</code>
관리 대상 객체(객체 저장소)	선택한 로컬 디렉토리 또는 LDAP 서버
보안: 사용자 저장소	<code>/var/imq/instances/<i>instanceName</i>/etc/passwd</code> 또는 LDAP 서버

표 A-1 Solaris에서 Message Queue 데이터의 위치(계속)

데이터 범주	Solaris에서 위치
보안: 액세스 제어 파일(기본 위치)	<code>/var/imq/instances/instanceName/etc/accesscontrol.properties</code>
보안: <code>passfile</code> 디렉토리(기본 위치)	<code>/var/imq/instances/instanceName/etc/</code>
보안: <code>passfile</code> 예	<code>/etc/imq/passfile.sample</code>
보안: 브로커 키 저장소 파일 위치	<code>/etc/imq/</code>
JavaDoc API 설명서	<code>/usr/share/javadoc/imq/index.html</code>
응용 프로그램 및 구성 예	<code>/usr/demo/imq/</code>
Java 아카이브(.jar), 웹 아카이브(.war) 및 자원 어댑터 아카이브(.rar) 파일	<code>/usr/share/lib/</code>

Linux

표 A-2에서는 Linux 플랫폼에서 Message Queue 데이터의 위치를 보여 줍니다.

표 A-2 Linux에서 Message Queue 데이터의 위치

데이터 범주	Windows에서 위치
브로커 인스턴스 구성 등록 정보	<code>/var/opt/imq/instances/instanceName/props/config.properties</code>
브로커 구성 파일 템플릿	<code>/opt/imq/lib/props/broker/</code>
영구 저장소(메시지, 대상, 영구 가입, 트랜잭션)	<code>/var/opt/imq/instances/instanceName/fs350/</code> 또는 JDBC 액세스 가능 데이터 저장소
브로커 인스턴스 로그 파일 디렉토리(기본 위치)	<code>/var/opt/imq/instances/instanceName/log/</code>
관리 대상 객체(객체 저장소)	선택한 로컬 디렉토리 또는 LDAP 서버

표 A-2 Linux에서 Message Queue 데이터의 위치(계속)

데이터 범주	Windows에서 위치
보안: 사용자 저장소	<code>/var/opt/imq/instances/instanceName/etc/passwd</code> 또는 LDAP 서버
보안: 액세스 제어 파일(기본 위치)	<code>/var/opt/imq/instances/instanceName/etc/accesscontrol.properties</code>
보안: <code>passfile</code> 디렉토리(기본 위치)	<code>/var/opt/imq/instances/instanceName/etc/</code>
보안: <code>passfile</code> 예	<code>/etc/opt/imq/passfile.sample</code>
보안: 브로커 키 저장소 파일 위치	<code>/etc/opt/imq/</code>
JavaDoc API 설명서	<code>/opt/imq/javadoc/index.html</code>
응용 프로그램 및 구성 예	<code>/opt/imq/demo/</code>
Java 아카이브(.jar), 웹 아카이브(.war) 및 자원 어댑터 아카이브(.rar) 파일	<code>/opt/imq/lib/</code>

Windows

표 A-3에서는 Sun Java System Application Server와 함께 제공되는 일부 Message Queue 설치 및 Windows 플랫폼에서 Message Queue 데이터의 위치를 보여 줍니다. 자세한 내용은 26페이지의 표 3과 `IMQ_HOME` 및 `IMQ_VARHOME`에 대한 정의를 참조하십시오.

표 A-3 Windows에서 Message Queue 데이터의 위치

데이터 범주	Windows에서 위치
브로커 인스턴스 구성 등록 정보	<code>IMQ_VARHOME\instances\instanceName\props\config.properties</code>
브로커 구성 파일 템플릿	<code>IMQ_HOME\lib\props\broker\</code>
영구 저장소(메시지, 대상, 영구 가입, 트랜잭션)	<code>IMQ_VARHOME\instances\instanceName\fs350\</code> 또는 JDBC 액세스 가능 데이터 저장소
브로커 인스턴스 로그 파일 디렉토리(기본 위치)	<code>IMQ_VARHOME\instances\instanceName\log\</code>

표 A-3 Windows에서 Message Queue 데이터의 위치(계속)

데이터 범주	Windows에서 위치
관리 대상 객체 (객체 저장소)	선택한 로컬 디렉토리 또는 LDAP 서버
보안: 사용자 저장소	IMQ_VARHOME\instances\ <i>instanceName</i> \etc\ passwd 또는 LDAP 서버
보안: 액세스 제어 파일(기본)	IMQ_VARHOME\instances\ <i>instanceName</i> \ etc\accesscontrol.properties
보안: passfile 디렉토리 (기본 위치)	IMQ_HOME\etc\ passfile.sample
보안: passfile 예	IMQ_HOME\etc\passfile.sample
보안: 브로커 키 저장소 파일 위치	IMQ_HOME\etc\ brokerkey.properties
JavaDoc API 설명서	IMQ_HOME\javadoc\index.html
응용 프로그램 및 구성 예	IMQ_HOME\demo\ demo.properties
Java 아카이브(.jar), 웹 아카이브 (.war) 및 자원 어댑터 아카이브 (.rar) 파일	IMQ_HOME\lib\ lib.properties

플러그인 지속성 설정

이 부록에서는 플러그인 지속성을 사용해서 JDBC로 액세스할 수 있는 데이터 저장소에 액세스하도록 브로커를 설정하는 방법을 설명합니다.

소개

Message Queue 브로커에는 지속성 정보의 기록과 복원을 관리하는 지속성 관리자 구성 요소가 있습니다(63페이지의 "지속성 관리자" 참조). 지속성 관리자는 기본적으로 기본 제공되는 파일 기반 데이터 저장소에 액세스하도록 구성되어 있지만 JDBC 호환 드라이버를 통해 액세스할 수 있는 데이터 저장소를 플러그인으로 사용하도록 재구성할 수도 있습니다.

플러그인 지속성을 사용하도록 브로커를 구성하려면 브로커 인스턴스 구성 파일에 JDBC 관련 등록 정보를 설정해야 합니다. Message Queue 지속성 작업을 수행할 수 있는 적절한 데이터베이스 스키마도 만들어야 합니다. Message Queue에는 JDBC 드라이버와 브로커 구성 등록 정보를 사용해서 플러그인 데이터베이스를 만들고 관리하는 데이터베이스 관리자 유틸리티(imgdbmgr)가 제공됩니다.

이 부록의 절차에서는 Java 2 Platform 엔터프라이즈판(J2EE) SDK와 함께 제공되는 PointBase DBMS를 예로 사용하여 설명합니다. 버전 1.4는 java.sun.com에서 다운로드할 수 있습니다. 예에서는 클라이언트/서버 버전 대신 PointBase 내장 버전을 사용합니다. 절차에서는 PointBase 예의 경로 이름과 등록 정보 이름을 사용하여 지침을 소개합니다. 이러한 지침은 "Example:"로 표시됩니다.

Oracle 및 PointBase의 구성 예는 부록 A, "Message Queue 데이터의 위치"에 표시된 예 위치에서 찾을 수 있습니다. 또한 PointBase 내장 버전, PointBase 서버 버전, Oracle 및 Cloudscape의 예는 인스턴스 구성 파일에서 주석 처리된 값으로 제공됩니다.

JDBC로 액세스할 수 있는 데이터 저장소 플러그인

JDBC로 액세스할 수 있는 데이터 저장소를 플러그인하는 작업은 단 몇 단계로 이루어집니다.

▶ JDBC로 액세스할 수 있는 데이터 저장소를 플러그인하는 방법

1. 브로커의 구성 파일에서 JDBC 관련 등록 정보를 설정합니다.

300페이지의 표 B-1에 설명된 등록 정보를 참조하십시오.

2. 다음 경로에 위치한 JDBC 드라이버 jar 파일의 사본 또는 심볼릭 링크를 넣습니다.

`/usr/share/lib/imq/imq/ext/ (Solaris)`

`/opt/imq/lib/ext/ (Linux)`

`IMQ_VARHOME\lib\ext (Windows)`

복사 예(Solaris):

```
% cp j2eeSDK_install_directory/pointbase/lib/pointbase.jar
/usr/share/lib/imq/ext
```

심볼릭 링크 예(Solaris):

```
% ln -s j2eeSDK_install_directory/lib/pointbase/pointbase.jar
/usr/share/lib/imq/ext
```

3. Message Queue 지속성에 필요한 데이터베이스 스키마를 만듭니다.

`imqdbmgr create all` 명령(내장 데이터베이스의 경우) 또는 `imqdbmgr create tbl` 명령(외부 데이터베이스의 경우)을 사용합니다. 303페이지의 "데이터베이스 관리자 유틸리티(imqdbmgr)"를 참조하십시오.

예:

- a. `imqdbmgr`가 위치한 디렉토리로 변경합니다.

`cd /usr/bin (Solaris)`

`cd /opt/imq/bin (Linux)`

`cd IMQ_HOME/bin (Windows)`

- b. `imqdbmgr` 명령을 입력합니다.

`imqdbmgr create all`

주 내장 데이터베이스를 사용하는 경우에는 데이터베이스를 다음 디렉토리에 만드는 것이 좋습니다.

```
.../instances/instanceName/dbstore/datababseName
```

내장 데이터베이스가 사용자 아이디와 비밀번호로 보호되지 않는 경우에는 파일 시스템 권한으로 보호합니다. 브로커에서 데이터베이스를 읽고 쓸 수 있게 하려면 브로커를 실행하는 사용자가 `imqdbmgr` 명령을 사용해서 내장 데이터베이스를 만든 사용자와 같아야 합니다 (303페이지의 "데이터베이스 관리자 유틸리티(`imqdbmgr`)" 참조).

JDBC 관련 브로커 구성 등록 정보

브로커의 인스턴스 구성 파일은 구성 파일과 연관된 브로커 인스턴스 이름(`instanceName`)으로 식별되는 디렉토리에 있습니다(부록 A, "Message Queue 데이터의 위치" 참조).

```
.../instances/instanceName/props/config.properties
```

파일이 아직 없는 경우에는 Message Queue에서 파일을 만들 수 있도록 `-name instanceName` 옵션을 사용해서 브로커를 시작해야 합니다.

표 B-1에는 JDBC로 액세스할 수 있는 데이터 저장소를 플러그인할 때 설정해야 하는 구성 등록 정보가 있습니다. 플러그인 지속성을 사용하는 각 브로커 인스턴스의 구성 파일(`config.properties`)에서 이 등록 정보를 설정합니다.

인스턴스 구성 등록 정보를 사용하면 Message Queue 데이터베이스 스키마를 만드는 SQL 코드를 사용자 정의할 수 있습니다. 각 데이터베이스 테이블을 만드는 SQL 코드를 지정하는 구성 가능한 등록 정보가 있습니다. 이 등록 정보에서 플러그인된 데이터베이스가 사용하는 데이터 유형을 올바르게 지정해야 합니다.

데이터베이스 공급업체 간에 정확한 SQL 구문에 대해 호환되지 않는 점이 있으므로 데이터베이스 공급업체의 해당 설명서를 확인하여 표 B-1의 등록 정보를 적절하게 조정해야 합니다. 예를 들어, PointBase 데이터베이스의 경우 `IMQMSG35` 테이블에서 `MSG` 열(`imq.persist.jdbc.table.IMQMSG35` 등록 정보 참조)의 최대 허용 길이를 조정해야 하는 경우가 있습니다.

표 B-1에서는 PointBase DBMS 예에서 지정하게 될 값을 소개합니다.

표 B-1 JDBC 관련 등록 정보

등록 정보 이름	설명
imq.persist.store	파일 기반 또는 JDBC 기반 데이터 저장소를 지정합니다. <i>예:</i> jdbc
imq.persist.jdbc.brokerid (선택 사항)	같은 데이터베이스를 영구 데이터 저장소로 사용하고 있는 브로커 인스턴스가 두 개 이상인 경우 데이터베이스 테이블 이름에 추가하여 고유한 이름으로 만들어 주는 브로커 인스턴스 식별자를 지정합니다. (보통 한 브로커 인스턴스에 대해서만 데이터를 저장하는 내장 데이터베이스에는 필요 없습니다.) 식별자는 데이터베이스에 허용되는 최대 테이블 이름 길이에서 12를 뺀 길이를 넘지 않는 영문자 및 숫자 문자열이어야 합니다. <i>예:</i> PointBase 내장 버전에는 필요 없음
imq.persist.jdbc.driver	데이터베이스에 연결할 JDBC 드라이버의 Java 클래스 이름을 지정합니다. <i>예:</i> com.pointbase.jdbc.jdbcUniversalDriver
imq.persist.jdbc.opendburl	기존 데이터베이스에 대한 연결을 여는 데이터베이스 URL 을 지정합니다. <i>예:</i> jdbc:pointbase:embedded:dbName; database.home= ../instances/instanceName/dbstore
imq.persist.jdbc.createdburl (선택 사항)	데이터베이스를 만들기 위해 연결을 여는 데이터베이스 URL 을 지정합니다. (imqdbmgr을 사용해서 데이터베이스를 만드는 경우에만 지정합니다.) <i>예:</i> jdbc:pointbase:embedded:dbName;new, database.home= ../instances/instanceName/dbstore
imq.persist.jdbc.closedburl (선택 사항)	브로커를 종료할 때 현재 데이터베이스 연결을 종료하는 데이터베이스 URL 을 지정합니다. <i>예:</i> PointBase에는 필요 없음
imq.persist.jdbc.user (선택 사항)	필요한 경우 데이터베이스 연결을 열 때 사용되는 사용자 아이디를 지정합니다. 보안상의 이유로 명령줄 옵션을 사용해서 값을 지정할 수도 있습니다. imqbrokerd -dbuser 및 imqdbmgr -u

표 B-1 JDBC 관련 등록 정보(계속)

등록 정보 이름	설명
imq.persist.jdbc.needpassword (선택 사항)	<p>브로커에 액세스할 때 데이터베이스에 비밀번호가 필요한지 여부를 지정합니다. true 값은 비밀번호가 필요하다는 것을 의미합니다. 비밀번호는 다음과 같은 명령줄 옵션을 사용해서 지정할 수 있습니다.</p> <pre>imqbrokerd -dbpassword imqdbmgr -p</pre> <p>명령줄 옵션과 passfile (225페이지의 "Passfile 사용" 참조) 중 어느 것으로도 비밀번호를 제공하지 않은 경우 브로커는 비밀번호를 묻는 프롬프트를 표시합니다.</p>
imq.persist.jdbc.password (선택 사항)	<p>필요한 경우 데이터베이스 연결을 열 때 사용되는 비밀번호를 지정합니다. 이 등록 정보는 passfile에서만 지정할 수 있습니다(225페이지의 "Passfile 사용" 참조).</p> <p>여러 방법으로 비밀번호를 제공할 수 있습니다. 가장 안전한 방법은 브로커가 비밀번호를 묻는 프롬프트를 표시하게 하는 것입니다. passfile을 사용하고 passfile을 읽기 방지하는 방법은 덜 안전합니다. 다음 명령줄 옵션을 사용하여 비밀번호를 지정하는 방법이 가장 안전하지 않습니다.</p> <pre>imqbrokerd -dbpassword imqdbmgr -p</pre>
imq.persist.jdbc.table.IMQSV35	<p>버전 테이블을 만들 때 사용하는 SQL 명령</p> <p><i>off:</i></p> <pre>CREATE TABLE \${name} (STOREVERSION INTEGER NOT NULL, BROKERID VARCHAR(100))</pre>
imq.persist.jdbc.table.IMQCCREC35	<p>구성 변경 레코드 테이블을 만들 때 사용하는 SQL 명령</p> <p><i>off:</i></p> <pre>CREATE TABLE \${name} (RECORDTIME BIGINT NOT NULL, RECORD BLOB(10k))</pre>
imq.persist.jdbc.table.IMQDEST35	<p>대상 테이블을 만들 때 사용하는 SQL 명령</p> <p><i>off:</i></p> <pre>CREATE TABLE \${name} (DID VARCHAR(100) NOT NULL, DEST BLOB(10k), primary key(DID))</pre>
imq.persist.jdbc.table.IMQINT3	<p>인터레스트 테이블을 만들 때 사용하는 SQL 명령</p> <p><i>off:</i></p> <pre>CREATE TABLE \${name} (CUID BIGINT NOT NULL, INTEREST BLOB(10k), primary key(CUID))</pre>

표 B-1 JDBC 관련 등록 정보(계속)

등록 정보 이름	설명
imq.persist.jdbc.table.IMQMSG35	<p>메시지 테이블을 만들 때 사용하는 SQL 명령</p> <p><i>off:</i></p> <pre>CREATE TABLE \${name} (MID VARCHAR(100) NOT NULL, DID VARCHAR(100), MSGSIZE BIGINT, MSG BLOB(1m), primary key(MID))</pre> <p>MSG 열의 기본 최대 길이는 1MB (1m)입니다. 이보다 큰 메시지가 있을 것으로 예상되면 이 길이를 알맞게 설정합니다. 이미 만든 테이블이 있는 경우 변경 사항을 적용하려면 테이블을 다시 만들어야 합니다.</p>
imq.persist.jdbc.table.IMQPROPS35	<p>등록 정보 테이블을 만들 때 사용하는 SQL 명령</p> <p><i>off:</i></p> <pre>CREATE TABLE \${name} (PROPNAME VARCHAR(100) NOT NULL, PROPVALUE BLOB(10k), primary key(PROPNAME))</pre>
imq.persist.jdbc.table.IMQILIST35	<p>인터레스트 상태 테이블을 만들 때 사용하는 SQL 명령</p> <p><i>off:</i></p> <pre>CREATE TABLE \${name} (MID VARCHAR(100) NOT NULL, CUID BIGINT, DID VARCHAR(100), STATE INTEGER, primary key(MID, CUID))</pre>
imq.persist.jdbc.table.IMQTXN35	<p>트랜잭션 테이블을 만들 때 사용하는 SQL 명령</p> <p><i>off:</i></p> <pre>CREATE TABLE \${name} (TUID BIGINT NOT NULL, STATE INTEGER, TSTATEOBJ BLOB(10K), primary key(TUID))</pre>
imq.persist.jdbc.table.IMQTACK35	<p>트랜잭션 확인 테이블을 만들 때 사용하는 SQL 명령</p> <p><i>off:</i></p> <pre>CREATE TABLE \${name} (TUID BIGINT NOT NULL, TXNACK BLOB(10k))</pre>

브로커 구성 등록 정보와 마찬가지로, -D 명령줄 옵션을 사용해서 값을 설정할 수 있습니다. 데이터베이스에 데이터베이스 특정 등록 정보를 설정해야 하는 경우에도 브로커(imqbrokerd) 또는 데이터베이스 관리자 유틸리티(imqdbmgr)를 시작할 때 -D 명령줄 옵션을 사용하여 설정할 수 있습니다.

예:

PointBase 내장 데이터베이스의 예에서는 데이터베이스 연결 URL에 데이터베이스 절대 경로를 지정하는 대신(표 B-1 에 참조), `-D` 명령줄 옵션을 사용하여 PointBase 시스템 디렉토리를 정의할 수 있습니다.

```
-Ddatabase.home=IMQ_VARHOME/instances/instanceName/dbstore
```

이 경우 데이터베이스를 만들고 여는 URL을 다음과 같이 간단하게 지정할 수 있습니다.

```
imq.persist.jdbc.createdburl=jdbc:pointbase:embedded:dbName;new
```

그리고

```
imq.persist.jdbc.opendburl=jdbc:pointbase:embedded:dbName
```

이 각각 적용됩니다.

데이터베이스 관리자 유틸리티(imqdbmgr)

Message Queue에는 지속성에 필요한 스키마를 설정하는 데이터베이스 관리자 유틸리티(imqdbmgr)가 포함되어 있습니다. 이 유틸리티는 테이블이 손상되거나 데이터 저장소로 다른 데이터베이스를 사용하는 경우에 Message Queue 데이터베이스 테이블을 삭제하는 용도로도 사용할 수 있습니다.

이 절에서는 기본 imqdbmgr 명령 구문을 설명하고, 하위 명령 목록을 제공하고, imqdbmgr 명령 옵션을 요약합니다.

imqdbmgr 명령의 구문

imqdbmgr 명령의 일반 구문은 다음과 같습니다.

```
imqdbmgr subcommand argument [options]
imqdbmgr -h|-help
imqdbmgr -v|-version
```

`-v`, 또는 `-h` 옵션을 지정하는 경우 명령줄에 지정된 하위 명령이 실행되지 않습니다. 예를 들어, 다음 명령을 입력하면 버전 정보는 표시되지만 `create` 하위 명령은 실행되지 않습니다.

```
imqdbmgr create all -v
```

imqdbmgr 하위 명령

데이터베이스 관리자 유틸리티(imqdbmgr)에는 표 B-2에 나열된 하위 명령이 포함되어 있습니다.

표 B-2 imqdbmgr 하위 명령

하위 명령 및 인수	설명
create all	새 데이터베이스와 Message Queue 영구 저장소 스키마를 만듭니다. 이 명령은 내장 데이터베이스 시스템에 사용되며, 사용하는 경우에는 <code>imq.persist.jdbc.createdburl</code> 등록 정보를 지정해야 합니다.
create tbl	기존 데이터베이스 시스템에 Message Queue 영구 저장소 스키마를 만듭니다. 이 명령은 외부 데이터베이스 시스템에 사용됩니다.
delete tbl	현재 영구 저장소 데이터베이스에서 기존 Message Queue 데이터베이스 테이블을 삭제합니다.
delete oldtbl	이전 버전의 영구 저장소 데이터베이스에서 Message Queue 데이터베이스 테이블을 모두 삭제합니다. 영구 저장소가 자동으로 현재 버전의 Message Queue 로 이전된 이후에 사용합니다.
recreate tbl	현재 영구 저장소 데이터베이스에서 기존 Message Queue 데이터베이스 테이블을 삭제한 후 Message Queue 영구 저장소 스키마를 다시 만듭니다.
reset lck	영구 저장소 데이터베이스를 다른 프로세스에서 사용할 수 있도록 잠금을 재설정합니다.

imqdbmgr 명령 옵션 요약

표 B-3에는 imqdbmgr 명령의 옵션이 나열되어 있습니다.

표 B-3 imqdbmgr 옵션

옵션	설명
-D <i>property=value</i>	지정한 등록 정보를 지정한 값으로 설정합니다.
-b <i>instanceName</i>	브로커 인스턴스 이름을 지정하고 해당 인스턴스 구성 파일을 사용합니다.
-h	사용 도움말을 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
-p <i>password</i>	데이터베이스 비밀번호를 지정합니다.
-u <i>name</i>	데이터베이스 사용자 아이디를 지정합니다.
-v	버전 정보를 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.

데이터베이스 관리자 유틸리티(imqdbmgr)

HTTP/HTTPS 지원(엔터프라이즈판)

Message Queue 엔터프라이즈판(33페이지의 "제품 판" 참조)에는 HTTP 및 HTTPS 연결 지원이 포함되어 있습니다(HTTPS는 Secure Socket Layer (SSL) 전송 연결상의 HTTP임). 이 지원을 사용하면 클라이언트 응용 프로그램에서 직접적인 TCP 연결 대신 HTTP 프로토콜을 사용하여 브로커와 통신할 수 있습니다. 이 부록에서는 이 지원의 활성화하는 데 사용되는 구조를 설명하고 클라이언트에서 Message Queue 메시징에 HTTP 기반 연결을 사용할 수 있도록 하는 데 필요한 설정 작업에 대해 설명합니다.

주 HTTP/HTTPS 지원은 Java 클라이언트에 대해 사용할 수 있지만 C 클라이언트에는 사용할 수 없습니다.

HTTP/HTTPS 지원 구조

Message Queue 메시징은 HTTP/HTTPS 연결 위에서 실행할 수 있습니다.

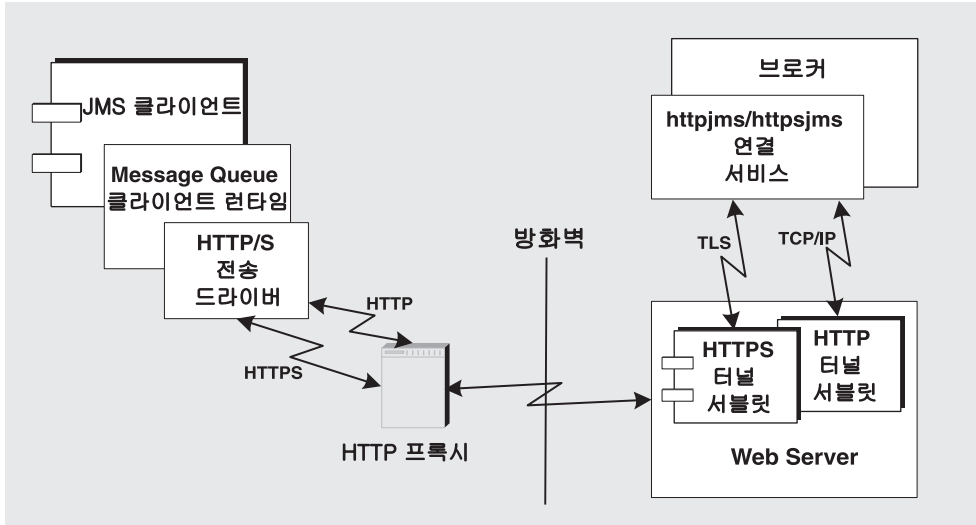
HTTP/HTTPS 연결은 보통 방화벽을 통해서도 허용되기 때문에 이렇게 하면 방화벽으로 클라이언트 응용 프로그램과 브로커를 분리할 수 있습니다.

308페이지의 [그림 C-1](#)에서는 HTTP/HTTPS 지원과 관련된 주요 구성 요소를 보여 줍니다.

- 클라이언트쪽에서는 HTTP 또는 HTTPS 전송 드라이버가 Message Queue 메시지를 HTTP 요청으로 캡슐화하고 요청이 정확한 순서로 Web Server에 전송되는지 확인합니다.
- 클라이언트는 필요한 경우 HTTP 프록시 서버를 사용해서 브로커와 통신할 수 있습니다. 프록시의 주소는 클라이언트를 시작할 때 명령줄 옵션을 사용해서 지정합니다. 자세한 내용은 [313페이지의 "HTTP 프록시 사용"](#)을 참조하십시오.

- HTTP 또는 HTTPS 터널 서블릿(모두 Message Queue와 함께 제공)이 Web Server에 로드되고 JMS 메시지를 브로커로 전달하기 전에 클라이언트 HTTP 요청으로부터 추출하는 데 사용됩니다. HTTP/HTTPS 터널 서블릿은 클라이언트에서 이루어진 HTTP 요청에 대한 응답으로 브로커 메시지를 클라이언트에 다시 보내는 작업도 수행합니다. 한 HTTP/HTTPS 터널 서블릿을 사용해서 여러 브로커에 액세스할 수 있습니다.

그림 C-1 HTTP/HTTPS 지원 구조



- 브로커쪽에서는 httpjms 또는 httpsjms 연결 서비스가 해당 터널 서블릿에서 들어오는 메시지를 분해하고 다중화를 해제합니다.
- Web Server가 오류로 다시 시작되면 모든 연결이 복원되며 클라이언트에는 아무 영향도 주지 않습니다. 브로커가 오류로 다시 시작되면 예외가 발생하며 클라이언트는 연결을 다시 설정해야 합니다. 드물기는 하지만, Web Server와 브로커에 모두 오류가 발생하고 브로커가 다시 시작되지 않은 경우에는 Web Server가 클라이언트 연결을 복원한 후 클라이언트에 알리지 않고 브로커 연결을 계속 기다릴 수 있습니다. 이런 상황을 방지하려면 항상 브로커를 다시 시작해야 합니다.

그림 C-1에서 알 수 있듯이, HTTP와 HTTPS가 지원하는 구조는 서로 매우 비슷합니다. 가장 큰 차이는 HTTPS (httpsjms 연결 서비스)의 경우 터널 서블릿이 클라이언트 응용 프로그램과 브로커 모두에 대해 보안 연결을 갖는다는 점입니다.

브로커에 대한 보안 연결은 Message Queue의 HTTPS 터널 서블릿에 해당하는 SSL 사용 터널 서블릿을 통해 제공되며 연결을 요청하는 모든 브로커에 자체 서명된 인증서를 전달합니다. 인증서는 브로커가 HTTPS 터널 서블릿에 대해 암호화된 연결을 설정할 때 사용됩니다. 이 연결이 설정되고 나면 클라이언트 응용 프로그램과 Web Server에서 클라이언트 응용 프로그램과 터널 서블릿 사이의 보안 연결을 처리할 수 있습니다.

HTTP 지원 활성화

다음 절에서는 HTTP 지원을 활성화하기 위해 수행하는 단계를 설명합니다.

▶ HTTP 지원을 활성화하는 방법

1. HTTP 터널 서블릿을 Web Server에 배포합니다.
2. 브로커의 httpjms 연결 서비스를 구성하고 브로커를 시작합니다.
3. HTTP 연결을 구성합니다.

1단계. Web Server에 HTTP 터널 서블릿 배포

Web Server에 HTTP 터널 서블릿을 배포하는 일반적인 방법에는 두 가지가 있습니다.

- jar 파일로 배포 - Servlet 2.1 이전 버전을 지원하는 Web Server용
- 웹 아카이브(WAR) 파일로 배포 - Servlet 2.2 이상 버전을 지원하는 Web Server용

Jar 파일로 배포

Message Queue 터널 서블릿 배포는 호스트 Web Server에 액세스할 수 있는 적절한 jar 파일을 생성하고 Web Server 시작 시 서블릿을 로드하도록 구성하며 서블릿 URL의 컨텍스트 루트 부분을 지정하는 작업으로 구성됩니다.

터널 서블릿 jar 파일(imqservlet.jar)은 HTTP 터널 서블릿에서 필요한 모든 클래스를 포함하며, 운영 체제에 따라 다른 디렉토리에 있습니다([부록 A, "Message Queue 데이터의 위치"](#) 참조).

서블릿 2.x를 지원하는 모든 Web Server를 사용하여 이 서블릿을 로드할 수 있습니다. 서블릿 클래스 이름은 다음과 같습니다.

```
com.sun.messaging.jmq.transport.  
httpunnel.servlet.HttpTunnelServlet
```

Web Server에서 `imqServlet.jar` 파일을 볼 수 있어야 합니다. Web Server와 브로커를 서로 다른 호스트에서 실행하려면 `imqServlet.jar` 파일의 복사본을 Web Server에서 액세스할 수 있는 위치에 저장해야 합니다.

또한 Web Server 시작 시 이 서블릿을 로드하도록 구성해야 하며, 서블릿 URL의 컨텍스트 루트 부분을 지정해야 하는 경우도 있습니다(314페이지의 "[예 1: Sun Java System Web Server에 HTTP 터널 서블릿 배포](#)" 참조).

또한 성능 향상을 위해 Web Server의 액세스 로깅 기능은 사용하지 않는 것이 좋습니다.

웹 아카이브 파일로 배포

HTTP 터널 서블릿을 WAR 파일로 배포하는 작업은 Web Server에서 제공하는 메커니즘을 사용하여 수행합니다. HTTP 터널 서블릿 WAR 파일(`imqhttp.war`)은 `.jar`, `.war` 및 `.rar` 파일이 포함된 디렉토리에 있으며, 이 디렉토리는 운영 체제에 따라 다릅니다(부록 A, "[Message Queue 데이터의 위치](#)" 참조).

WAR 파일에는 Web Server에서 서블릿을 로드하여 실행하는 데 필요한 기본 구성 정보가 들어 있는 배포 설명자가 포함되어 있습니다. Web Server에 따라 서블릿 URL의 컨텍스트 루트 부분을 지정해야 하는 경우도 있습니다(317페이지의 "[예 2: Sun Java System Application Server 7.0에 HTTP 터널 서블릿 배포](#)" 참조).

2단계. httpjms 연결 서비스 구성

기본적으로 브로커에 대해 HTTP 지원이 활성화되어 있지 않으므로 `httpjms` 연결 서비스가 활성화되도록 브로커를 재구성해야 합니다. 재구성한 경우 134페이지의 "[브로커 시작](#)"에 설명된 대로 브로커를 시작할 수 있습니다.

▶ **httpjms 연결 서비스를 활성화하는 방법**

1. 브로커의 인스턴스 구성 파일을 엽니다.

인스턴스 구성 파일은 해당 구성 파일이 연결되어 있는 브로커 인스턴스의 이름 (*instanceName*)으로 식별되는 디렉토리에 저장됩니다(부록 A, "Message Queue 데이터의 위치" 참조).

```
.../instances/instanceName/props/config.properties
```

2. `imq.service.activelist` 등록 정보에 `httpjms` 값을 추가합니다.

```
imq.service.activelist=jms,admin,httpjms
```

브로커는 시작할 때 호스트 시스템에서 실행 중인 Web Server와 HTTP 터널 서버를 찾습니다. 그러나 원격 터널 서버에 액세스하기 위해 `servletHost` 및 `servletPort` 연결 서비스 등록 정보를 재구성할 수 있습니다.

또한 성능 향상을 위해 `pullPeriod` 등록 정보를 재구성할 수도 있습니다. `httpjms` 연결 서비스 구성 등록 정보에 대한 자세한 설명은 311페이지의 표 C-1을 참조하십시오.

표 C-1 httpjms 연결 서비스 등록 정보

등록 정보 이름	설명
<code>imq.httpjms.http.servletHost</code>	필요한 경우 이 값을 변경하여 HTTP 터널 서버를 실행하는 호스트의 이름(호스트 이름 또는 IP 주소)을 지정합니다(원격 호스트이거나 로컬 호스트의 특정 호스트 이름일 수 있음). 기본값: localhost
<code>imq.httpjms.http.servletPort</code>	이 값을 변경하여 브로커가 HTTP 터널 서버에 액세스하는 데 사용하는 포트 번호를 지정합니다(Web Server에서 기본 포트를 변경한 경우 이 등록 정보를 적절히 변경해야 함). 기본값: 7675
<code>imq.httpjms.http.pullPeriod</code>	클라이언트 런타임이 브로커에서 메시지를 가져오기 위해 만든 HTTP 요청 사이의 간격(초)을 지정합니다(이 등록 정보는 브로커에서 설정되고 클라이언트 런타임에 전파됨). 값이 0 또는 음수인 경우 클라이언트는 하나의 HTTP 요청을 항상 보류 상태로 두고 가능한 빨리 메시지를 가져오도록 준비합니다. 클라이언트 수가 많은 경우 Web Server 자원을 많이 사용하여 서버가 응답하지 않을 수 있습니다. 그런 경우 <code>pullPeriod</code> 등록 정보를 양수(초)로 설정해야 합니다. 이 등록 정보는 후속 가져오기 요청을 만들기 전에 클라이언트의 HTTP 전송 드라이버가 대기하는 시간을 설정합니다. 값을 양수로 설정하면 클라이언트가 응답 시간 동안 대기하는 대신 Web Server 자원이 절약됩니다. 기본값: -1

표 C-1 httpjms 연결 서비스 등록 정보(계속)

등록 정보 이름	설명
imq.httpjms.http.connectionTimeout	클라이언트 런타임이 HTTP 터널 서버릿의 응답을 기다리는 시간(초)을 지정합니다. 이 시간이 초과되면 예외가 발생합니다(이 등록 정보는 브로커에서 설정되고 클라이언트 런타임에 전파됨). 이 등록 정보는 브로커가 HTTP 터널 서버릿과 통신한 후 연결을 해제할 때까지 기다리는 시간도 지정합니다. 이 경우에는 브로커와 터널 서버릿이 HTTP 서버릿에 액세스 중인 클라이언트가 비정상적으로 종료했는지 여부를 알 수 없으므로 시간 초과가 필요합니다. 기본값: 60

3단계. HTTP 연결 구성

클라이언트 응용 프로그램에서는 제대로 구성된 연결 팩토리 관리 대상 객체를 사용해서 브로커에 대한 HTTP 연결을 설정해야 합니다. 이 절에서는 HTTP 연결 구성 문제에 대해 논의합니다.

연결 팩토리 구성

HTTP 지원을 활성화하려면 연결 팩토리의 `imqAddressList` 속성을 HTTP 터널 서버릿 URL로 설정해야 합니다. HTTP 터널 서버릿 URL의 일반 구문은 다음과 같습니다.

```
http://hostName:port/contextRoot/tunnel
```

여기서 `hostName:port`는 HTTP 터널 서버릿을 호스트하는 Web Server의 이름과 포트이며, `contextRoot`는 Web Server에서 터널 서버릿을 배포할 때 설정된 경로입니다.

일반적인 연결 팩토리 속성, 특히 `imqAddressList` 속성에 대한 자세한 내용은 *Message Queue Java Client Developer's Guide*를 참조하십시오.

다음 방법 중 하나를 사용하여 연결 팩토리 속성을 설정할 수 있습니다.

- 연결 팩토리 관리 대상 객체를 만드는 `imqobjmgr` 명령에 `-o` 옵션을 사용하거나(195 페이지의 "연결 팩토리 추가" 참조) 관리 콘솔(`imqadmin`)을 사용하여 연결 팩토리 관리 대상 객체를 만들 때 속성을 설정합니다.
- 클라이언트를 시작하는 명령에 `-D` 옵션을 사용합니다(*Message Queue Java Client Developer's Guide* 참조).

- 클라이언트 코드에서 프로그래밍 방식으로 연결 팩토리를 만든 후 API 호출을 사용하여 해당 연결 팩토리의 속성을 설정합니다(*Message Queue Java Client Developer's Guide* 참조).

단일 서블릿을 사용하여 다중 브로커에 액세스

다중 브로커를 실행할 경우 다중 Web Server 및 서블릿 인스턴스를 구성하지 않아도 됩니다. 동시에 실행 중인 브로커 간에 단일 Web Server와 HTTP 터널 서블릿을 공유할 수 있습니다. 다중 브로커 인스턴스가 단일 터널 서블릿을 공유하는 경우 `imqAddressList` 연결 팩토리 속성을 다음과 같이 구성해야 합니다.

```
http://hostName:port/contextRoot/tunnel?ServerName=bkrHostName:instanceName
```

여기서 `bkrHostName`은 브로커 인스턴스 호스트 이름이고 `instanceName`은 클라이언트가 액세스할 특정 브로커 인스턴스의 이름입니다.

`bkrHostName`과 `instanceName`에 정확한 문자열을 입력했는지 확인하려면 브라우저에서 서블릿 URL에 액세스하여 HTTP 터널 서블릿의 상태 보고서를 생성합니다. 보고서에는 서블릿이 액세스하는 모든 브로커가 나열됩니다.

```
HTTP tunnel servlet ready.
Servlet Start Time : Thu May 30 01:08:18 PDT 2002
Accepting TCP connections from brokers on port : 7675
Total available brokers = 2
Broker List :
    jpgserv:broker2
    cochin:broker1
```

HTTP 프록시 사용

HTTP 프록시를 사용해서 HTTP 터널 서블릿에 액세스하는 경우:

- `http.proxyHost` 시스템 등록 정보를 프록시 서버 호스트 이름으로 설정합니다.
- `http.proxyPort` 시스템 등록 정보를 프록시 서버 포트 번호로 설정합니다.

클라이언트 응용 프로그램을 시작하는 명령에 `-D` 옵션을 사용하여 이러한 등록 정보를 설정할 수 있습니다.

예 1: Sun Java System Web Server에 HTTP 터널 서블릿 배포

이 절에서는 Sun Java System Web Server에서 HTTP 터널 서블릿을 jar 파일과 WAR 파일로 배포하는 방법을 모두 설명합니다. 사용 방법은 Sun Java System Web Server 버전에 따라 다르지만, Servlet 2.2 이상을 지원하지 않으면 WAR 파일 배포를 처리할 수 없습니다.

Jar 파일로 배포

다음은 브라우저 기반 관리 GUI를 사용하여 Sun Java System Web Server 6.1에 배포하는 경우에 대한 지침입니다. 이 절차는 다음과 같은 일반적인 단계로 구성됩니다.

1. 서블릿 추가
2. 서블릿 가상 경로 구성
3. 서블릿 로드
4. 서블릿 액세스 로그 비활성화

이러한 단계는 다음 하위 절에 설명되어 있습니다. 웹 브라우저를 사용하여 서블릿 URL에 액세스함으로써 HTTP 터널 서블릿 배포 성공 여부를 확인할 수 있습니다. 상태 정보를 표시해야 합니다.

서블릿 추가

▶ 터널 서블릿을 추가하는 방법

1. 서블릿 탭을 선택합니다.
2. 서블릿 속성 구성을 선택합니다.
3. 서블릿 이름 필드에 터널 서블릿의 이름을 지정합니다.
4. 서블릿 코드(클래스 이름) 필드를 다음 값으로 설정합니다.

```
com.sun.messaging.jmq.transport.  
httptunnel.servlet.HttpTunnelServlet
```

5. 서블릿 클래스 경로 필드에 imqservlet.jar에 대한 전체 경로를 입력합니다. 예를 들면 다음과 같습니다.

```
/usr/share/lib/imq/imqservlet.jar (Solaris)  
  
/opt/imq/lib/imqservlet.jar (Linux)  
  
IMQ_HOME/lib/imqservlet.jar (Windows)
```

6. 서블릿 인수 필드에 표 C-2와 같은 선택 인수를 입력합니다.

표 C-2 HTTP 터널 서블릿 Jar 파일 배포에 사용되는 서블릿 인수

인수	기본값	참조
<code>servletHost</code>	모든 호스트	311페이지의 표 C-1 참조
<code>servletPort</code>	7675	311페이지의 표 C-1 참조

두 인수를 모두 사용하는 경우에는 인수 사이를 쉼표로 구분합니다.

`servletPort=portnumber, servletHost=...`

`servletHost` 및 `servletPort` 인수는 Web Server와 브로커간 통신에만 적용되며 기본값에 문제가 있는 경우에만 설정됩니다. 그런 경우에도 브로커 구성 등록 정보를 적절하게 설정해야 합니다(311페이지의 표 C-1 참조). 예를 들면 다음과 같습니다.

```
img.httpjms.http.servletPort
```

서블릿 가상 경로(서블릿 URL) 구성

▶ 터널 서블릿에 대한 가상 경로(서블릿 URL)를 구성하는 방법

1. 서블릿 탭을 선택합니다.
2. 서블릿 가상 경로 변환 구성을 선택합니다.
3. 가상 경로 필드를 설정합니다.

가상 경로는 터널 서블릿 URL의 `/contextRoot/tunnel` 부분입니다.

```
http://hostName:port/contextRoot/tunnel
```

예를 들어, `contextRoot`를 `img`로 설정했다면 가상 경로 필드는 다음과 같습니다.

```
/img/tunnel
```

4. 서블릿 이름 필드를 314페이지의 "서블릿 추가"의 단계 3과 동일한 값으로 설정합니다.

서블릿 로드

▶ Web Server 시작 시 터널 서블릿을 로드하는 방법

1. 서블릿 탭을 선택합니다.

2. 전역 속성 구성을 선택합니다.
3. 시작 서블릿 필드에 314페이지의 "서블릿 추가"의 단계 3과 동일한 서블릿 이름 값을 입력합니다.

서버 액세스 로그 비활성화

서버 액세스 로그를 비활성화할 필요는 없지만 그렇게 하면 성능이 향상됩니다.

▶ 서버 액세스 로그를 비활성화하는 방법

1. 상태 탭을 선택합니다.
2. 로그 기본 설정 페이지를 선택합니다.
3. 로그 클라이언트 액세스 제어를 사용하여 로깅을 비활성화합니다.

WAR 파일로 배포

다음은 Sun Java System Web Server 6.0 서비스에 배포하는 경우에 대한 지침입니다. 웹 브라우저를 사용하여 서블릿 URL에 액세스하면 HTTP 터널 서블릿의 배포 성공 여부를 확인할 수 있습니다. 상태 정보를 표시해야 합니다.

▶ http 터널 서블릿을 WAR 파일로 배포하는 방법

1. 브라우저 기반 관리 GUI에서 가상 서버 클래스 탭을 선택한 후 클래스 관리를 선택합니다.
2. 해당 가상 서버 클래스 이름(예: defaultClass)을 선택하고 관리 버튼을 누릅니다.
3. 가상 서버 관리를 선택합니다.
4. 해당 가상 서버 이름을 선택하고 관리 버튼을 누릅니다.
5. 웹 응용 프로그램 탭을 선택합니다.
6. 웹 응용 프로그램 배포를 누릅니다.
7. WAR 파일 위치 및 WAR 파일 경로 필드에서 `imghttp.war` 파일을 가리키는 적절한 값을 선택합니다. 이 파일이 있는 디렉토리는 운영 체제에 따라 다릅니다(부록 A, "Message Queue 데이터의 위치" 참조).

- 응용 프로그램 URI 필드에 경로를 입력합니다.

응용 프로그램 URI 필드 값은 터널 서블릿 URL의 */contextRoot* 부분입니다.

```
http://hostName:port/contextRoot/tunnel
```

예를 들어, *contextRoot*를 *img*로 설정했다면 응용 프로그램 URI 필드는 다음과 같습니다.

```
/img
```

- 서블릿을 배포할 설치 디렉토리 경로(일반적으로 Sun Java System Web Server 설치 루트 아래)를 입력합니다.
- 확인을 누릅니다.
- Web Server 인스턴스를 다시 시작합니다.

이제 서블릿을 다음 주소에서 사용할 수 있습니다.

```
http://hostName:port/contextRoot/tunnel
```

이제 클라이언트에서 이 URL을 통해 HTTP 연결을 사용하는 메시지 서비스에 연결할 수 있습니다.

예 2: Sun Java System Application Server 7.0에 HTTP 터널 서블릿 배포

이 절에서는 Sun Java System Application Server 7.0에서 HTTP 터널 서블릿을 WAR 파일로 배포하는 방법을 설명합니다.

다음 두 단계를 거쳐야 합니다.

- Application Server 7.0 배포 도구를 사용하여 HTTP 터널 서블릿 배포
- Application Server 인스턴스의 *server.policy* 파일 수정

배포 도구 사용

▶ Application Server 7.0 환경에 HTTP 터널 서블릿을 배포하는 방법

- 웹 기반 관리 GUI에서 다음을 선택합니다.

Application Server 인스턴스 > server1 > 응용 프로그램 > 웹 응용 프로그램

- 배포 버튼을 누릅니다.

3. 파일 경로: 텍스트 필드에 HTTP 터널 서블릿 WAR 파일(imqhttp.war)의 위치를 입력합니다.

imqhttp.war 파일의 위치는 운영 체제에 따라 다릅니다(부록 A, "Message Queue 데이터의 위치" 참조).

4. 확인을 누릅니다.
5. 다음 화면에서 컨텍스트 루트 텍스트 필드의 값을 설정합니다.
컨텍스트 루트 필드 값은 터널 서블릿 URL의 `/contextRoot` 부분입니다.

```
http://hostName:port/contextRoot/tunnel
```

예를 들어, 컨텍스트 루트 필드를 다음과 같이 설정할 수 있습니다.

```
/img
```

6. 확인을 누릅니다.
다음 화면에 터널 서블릿이 성공적으로 배포되었고 기본적으로 사용되며 다음 위치(이 예의 경우)에 있다고 표시됩니다.

```
/var/opt/SUNWappserver7/domains/domain1/server1/applications/  
j2ee-modules/imqhttp_1
```

이제 서블릿을 다음 주소에서 사용할 수 있습니다.

```
http://hostName:port/contextRoot/tunnel
```

이제 클라이언트에서 이 URL을 통해 HTTP 연결을 사용하는 메시지 서비스에 연결할 수 있습니다.

server.policy 파일 수정

Application Server 7.0이 시행하는 일련의 기본 보안 정책은 수정하지 않으면 HTTP 터널 서블릿이 Message Queue 브로커에서 연결을 수신할 수 없도록 합니다.

Application Server 인스턴스마다 해당 보안 정책이나 규칙이 포함된 파일이 있습니다. 예를 들어, Solaris의 server1 인스턴스의 경우 이 파일의 위치는 다음과 같습니다.

```
/var/opt/SUNWappserver7/domains/domain1/server1/config/  
server.policy
```

터널 서블릿이 Message Queue 브로커에서 연결을 수신하도록 하려면 이 파일에 항목을 추가해야 합니다.

- ▶ **Application Server의 server.policy 파일을 수정하는 방법**
 1. server.policy 파일을 엽니다.
 2. 다음 항목을 추가합니다.

```
grant codeBase
"file:/var/opt/SUNWappserver7/domains/domain1/server1/
  applications/j2ee-modules/imqhttp_1/-"
{
  permission java.net.SocketPermission "*",
    "connect,accept,resolve";
};
```

HTTPS 지원 활성화

다음 절에서는 HTTPS 지원을 활성화하기 위해 수행하는 단계에 대해 설명합니다. [309페이지의 "HTTP 지원 활성화"](#)와 비슷한 내용에 SSL 인증서 생성 및 액세스에 필요한 단계가 추가되어 있습니다.

- ▶ **HTTPS 지원을 활성화하는 방법**
 1. HTTPS 터널 서블릿에 대해 자체 서명된 인증서를 생성합니다.
 2. HTTPS 터널 서블릿을 Web Server에 배포합니다.
 3. 브로커의 httpsjms 연결 서비스를 구성하고 브로커를 시작합니다.
 4. HTTPS 연결을 구성합니다.

각 단계에 대해서는 뒤에서 자세히 설명합니다.

1단계. HTTPS 터널 서블릿에 대해 자체 서명된 인증서 생성

Message Queue의 SSL 지원은 클라이언트가 알려지고 신뢰할 수 있는 서버와 통신한다는 가정 하에 전송 데이터를 보호하기 위한 것입니다. 따라서 자체 서명된 서버 인증서만 사용하여 SSL을 구현합니다. httpsjms 연결 서비스 구조에서 HTTPS 터널 서블릿은 브로커와 응용 프로그램 클라이언트에 대해 서버 역할을 담당합니다.

imqkeytool 유틸리티를 실행하여 터널 서블릿에 대해 자체 서명된 인증서를 생성합니다. 명령 프롬프트에서 다음을 입력합니다.

```
imqkeytool -servlet keystore_location
```

유틸리티는 필요한 정보를 묻는 메시지를 표시합니다(Unix 시스템에서 키 저장소를 만들 권한을 가지려면 imqkeytool을 슈퍼유저(루트)로 실행해야 할 수도 있음).

imqkeytool은 키 저장소 비밀번호, 일부 조직 정보 및 사용자의 확인을 묻는 메시지를 차례로 표시합니다. 확인이 끝나면 키 쌍을 생성하는 동안 일시 중지됩니다. 그런 다음 특정 키 쌍을 잠금 비밀번호(키 비밀번호)를 묻습니다. 이 프롬프트에 대한 응답으로 **Return** 키를 눌러야 합니다. 그러면 키 비밀번호가 키 저장소 비밀번호와 동일하게 지정됩니다.

주 입력한 비밀번호를 기억해야 합니다. 나중에 이 비밀번호를 제공해야 터널 서블릿이 키 저장소를 열 수 있습니다.

imqkeytool을 실행하면 JDK keytool 유틸리티를 실행하여 자체 서명된 인증서를 생성하고 *keystore_location* 인수에서 지정한 위치의 **Message Queue** 키 저장소 파일에 넣을 수 있습니다(키 저장소의 형식은 JDK1.2 keytool에서 지원하는 것과 같음).

주 HTTPS 터널 서블릿에서 키 저장소를 볼 수 있어야 합니다. *keystore_location*에 생성된 키 저장소를 HTTPS 터널 서블릿에서 액세스할 수 있는 위치로 이동/복사합니다([320페이지의 "2단계. Web Server에 HTTPS 터널 서블릿 배포"](#) 참조).

2단계. Web Server에 HTTPS 터널 서블릿 배포

Web Server에 HTTPS 터널 서블릿을 배포하는 일반적인 방법에는 두 가지가 있습니다.

- jar 파일로 배포 - Servlet 2.1 이전 버전을 지원하는 Web Server용
- 웹 아카이브(WAR) 파일로 배포 - Servlet 2.2 이상 버전을 지원하는 Web Server용

두 경우 모두 클라이언트와 브로커 사이에서 중단간 보안 통신이 이루어질 수 있도록 Web Server에 암호화가 활성화되었는지 확인해야 합니다.

Jar 파일로 배포

Message Queue 터널 서블릿 배포는 호스트 Web Server에 액세스할 수 있는 적절한 jar 파일을 생성하고 Web Server 시작 시 서블릿을 로드하도록 구성하며 서블릿 URL의 컨텍스트 루트 부분을 지정하는 작업으로 구성됩니다.

터널 서블릿 jar 파일(imqServlet.jar)은 HTTPS 터널 서블릿에서 필요한 모든 클래스를 포함하며, 운영 체제에 따라 다른 디렉토리에 있습니다(부록 A, "Message Queue 데이터의 위치" 참조).

서블릿 2.x를 지원하는 모든 Web Server를 사용하여 이 서블릿을 로드할 수 있습니다. 서블릿 클래스 이름은 다음과 같습니다.

```
com.sun.messaging.jmq.transport.  
httpunnel.servlet.HttpsTunnelServlet
```

Web Server에서 imqServlet.jar 파일을 볼 수 있어야 합니다. Web Server와 브로커를 서로 다른 호스트에서 실행하려면 imqServlet.jar 파일의 복사본을 Web Server에서 액세스할 수 있는 위치에 저장해야 합니다.

또한 Web Server 시작 시 이 서블릿을 로드하도록 구성해야 하며, 서블릿 URL의 컨텍스트 루트 부분을 지정해야 하는 경우도 있습니다(326페이지의 "예 3: Sun Java System Web Server에 HTTPS 터널 서블릿 배포" 참조).

JSSE jar 파일이 Web Server에서 서블릿을 실행하는 클래스 경로에 있는지 확인합니다. 수행 방법은 Web Server 설명서를 확인하십시오.

Web Server 구성에서 중요한 부분 중 하나는 HTTPS 터널 서블릿에서 브로커와 보안 연결을 설정할 때 사용할 자체 서명된 인증서의 위치와 비밀번호를 지정하는 것입니다. 319페이지의 "1단계. HTTPS 터널 서블릿에 대해 자체 서명된 인증서 생성"에서 만든 키 저장소를 HTTPS 터널 서블릿에서 액세스할 수 있는 위치에 두어야 합니다.

또한 성능 향상을 위해 Web Server의 액세스 로깅 기능은 사용하지 않는 것이 좋습니다.

웹 아카이브 파일로 배포

HTTPS 터널 서블릿을 WAR 파일로 배포하는 작업은 Web Server에서 제공하는 배포 메커니즘을 사용하여 수행합니다. HTTPS 터널 서블릿 WAR 파일(imqhttps.war)은 운영 체제에 따라 다른 디렉토리에 있습니다(부록 A, "Message Queue 데이터의 위치" 참조).

WAR 파일에는 Web Server에서 서블릿을 로드하여 실행하는 데 필요한 기본 구성 정보가 들어 있는 배포 설명자가 포함되어 있습니다. Web Server에 따라 서블릿 URL의 컨텍스트 루트 부분을 지정해야 하는 경우도 있습니다(331페이지의 "예 4: Sun Java System Application Server 7.0에 HTTPS 터널 서블릿 배포" 참조).

하지만 `imghttps.war` 파일의 배포 설명자에서는 터널 서블릿에 필요한 키 저장소 파일이 있는 위치를 알 수 없습니다(319페이지의 "1단계. HTTPS 터널 서블릿에 대해 자체 서명된 인증서 생성" 참조). 따라서 `imghttps.war` 파일을 배포하기 전에 터널 서블릿의 배포 설명자(XML 파일)를 편집하여 키 저장소 위치를 지정해야 합니다.

3단계. httpsjms 연결 서비스 구성

기본적으로 브로커에 대해 HTTPS 지원이 활성화되어 있지 않으므로 `httpsjms` 연결 서비스가 활성화되도록 브로커를 재구성해야 합니다. 재구성한 경우 134페이지의 "브로커 시작"에 설명된 대로 브로커를 시작할 수 있습니다.

▶ httpsjms 연결 서비스를 활성화하는 방법

1. 브로커의 인스턴스 구성 파일을 엽니다.

인스턴스 구성 파일은 해당 구성 파일이 연결되어 있는 브로커 인스턴스의 이름 (*instanceName*)으로 식별되는 디렉토리에 저장됩니다(부록 A, "Message Queue 데이터의 위치" 참조).

```
.../instances/instanceName/props/config.properties
```

2. `img.service.activelist` 등록 정보에 `httpsjms` 값을 추가합니다.

```
img.service.activelist=jms,admin,httpsjms
```

브로커는 시작할 때 호스트 시스템에서 실행 중인 Web Server와 HTTPS 터널 서블릿을 찾습니다. 그러나 원격 터널 서블릿에 액세스하기 위해 `servletHost` 및 `servletPort` 연결 서비스 등록 정보를 재구성할 수 있습니다.

또한 성능 향상을 위해 `pullPeriod` 등록 정보를 재구성할 수도 있습니다. `httpsjms` 연결 서비스 구성 등록 정보에 대한 자세한 설명은 표 C-3을 참조하십시오.

표 C-3 httpsjms 연결 서비스 등록 정보

등록 정보 이름	설명
imq.httpsjms.https servletHost	필요한 경우 이 값을 변경하여 HTTPS 터널 서블릿을 실행하는 호스트의 이름(호스트 이름 또는 IP 주소)을 지정합니다(원격 호스트이거나 로컬 호스트의 특정 호스트 이름일 수 있음). 기본값: localhost
imq.httpsjms.https servletPort	이 값을 변경하여 브로커가 HTTPS 터널 서블릿에 액세스하는 데 사용하는 포트 번호를 지정합니다(Web Server에서 기본 포트를 변경한 경우 이 등록 정보를 적절히 변경해야 함). 기본값: 7674
imq.httpsjms.https pullPeriod	브로커에서 메시지를 가져오기 위해 각 클라이언트가 만든 HTTP 요청 사이의 간격(초)을 지정합니다(이 등록 정보는 브로커에서 설정되고 클라이언트 런타임에 전파됨). 값이 0 또는 음수인 경우 클라이언트는 하나의 HTTP 요청을 항상 보류 상태로 두고 가능한 빨리 메시지를 가져오도록 준비합니다. 클라이언트 수가 많은 경우 Web Server 자원을 많이 사용하여 서버가 응답하지 않을 수 있습니다. 그런 경우 pullPeriod 등록 정보를 양수(초)로 설정해야 합니다. 이 등록 정보는 후속 가져오기 요청을 만들기 전에 클라이언트의 HTTP 전송 드라이버가 대기하는 시간을 설정합니다. 값을 양수로 설정하면 클라이언트가 응답 시간 동안 대기하는 대신 Web Server 자원이 절약됩니다. 기본값: -1
imq.httpsjms.https connectionTimeout	클라이언트 런타임이 HTTPS 터널 서블릿의 응답을 기다리는 시간(초)을 지정합니다. 이 시간이 초과되면 예외가 발생합니다(이 등록 정보는 브로커에서 설정되고 클라이언트 런타임에 전파됨). 이 등록 정보는 브로커가 HTTPS 터널 서블릿과 통신한 후 연결을 해제할 때까지 기다리는 시간도 지정합니다. 이 경우에는 브로커와 터널 서블릿이 HTTPS 서블릿에 액세스 중인 클라이언트가 비정상적으로 종료했는지 여부를 알 수 없으므로 시간 초과가 필요합니다. 기본값: 60

4단계. HTTPS 연결 구성

클라이언트 응용 프로그램에서는 제대로 구성된 연결 팩토리 관리 대상 객체를 사용해서 브로커에 대한 HTTPS 연결을 설정해야 합니다.

하지만 클라이언트는 Java Secure Socket Extension (JSSE)에서 제공하는 SSL 라이브러리에도 액세스해야 하며 루트 인증서도 있어야 합니다. SSL 라이브러리는 JDK 1.4와 함께 제공됩니다. 이전 버전의 JDK가 있는 경우에는 "JSSE 구성"을 참조하십시오. 그렇지 않으면 "루트 인증서 가져오기"로 이동하십시오.

이 문제를 해결하고 나면 계속해서 HTTPS 연결을 구성할 수 있습니다.

JSSE 구성

▶ JSSE를 구성하는 방법

1. JSSE jar 파일을 `JRE_HOME/lib/ext` 디렉토리로 복사합니다.
`jsse.jar, jnet.jar, jcert.jar`
2. JSSE 보안 공급자를 정적으로 추가합니다. 이 작업은
`security.provider.n=com.sun.net.ssl.internal.ssl.Provider`
 를 `JRE_HOME/lib/security/java.security` 파일에 추가하여 수행할 수 있습니다.
 (여기서 `n`은 보안 공급자 패키지에서 다음으로 사용할 수 있는 우선 순위 번호입니다.)
3. JDK1.4를 사용하지 않는 경우에는 클라이언트 응용 프로그램을 시작하는 명령에 `-D`
 옵션을 사용해서 다음 JSSE 등록 정보를 설정해야 합니다.
`java.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol`

루트 인증서 가져오기

Web Server 인증서에 서명한 CA의 루트 인증서가 기본적으로 트러스트 데이터베이스에 있지 않거나 독자적인 Web Server 인증서를 사용하는 경우에는 인증서를 트러스트 데이터베이스에 추가해야 합니다. 여기에 해당되면 다음 지침에 따르고, 그렇지 않으면 "[연결 팩토리 구성](#)"으로 이동합니다.

인증서가 `cert_file`에 저장되어 있고 `trust_store_file`이 키 저장소라면 다음 명령을 실행합니다.

```
JRE_HOME/bin/keytool -import -trustcacerts
-alias alias_for_certificate -file cert_file
-keystore trust_store_file
```

다음 질문에 YES로 응답합니다. Trust this certificate?

클라이언트 응용 프로그램을 시작하는 명령에 `-D` 옵션을 사용해서 다음과 같은 JSSE 등록 정보도 지정해야 합니다.

```
javax.net.ssl.trustStore=trust_store_file
javax.net.ssl.trustStorePassword=trust_store_passwd
```

연결 팩토리 구성

HTTPS 지원을 활성화하려면 연결 팩토리의 `imqAddressList` 속성을 HTTPS 터널 서블릿 URL로 설정해야 합니다. HTTPS 터널 서블릿 URL의 일반 구문은 다음과 같습니다.

```
https://hostName:port/contextRoot/tunnel
```

여기서 `hostName:port`는 HTTPS 터널 서블릿을 호스트하는 Web Server의 이름과 포트이며, `contextRoot`는 Web Server에 터널 서블릿을 배포할 때 설정된 경로입니다.

일반적인 연결 팩토리 속성, 특히 `imqAddressList` 속성에 대한 자세한 내용은 *Message Queue Java Client Developer's Guide*를 참조하십시오.

다음 방법 중 하나를 사용하여 연결 팩토리 속성을 설정할 수 있습니다.

- 연결 팩토리 관리 대상 객체를 만드는 `imqobjmgr` 명령에 `-o` 옵션을 사용하거나(195 페이지의 "연결 팩토리 추가" 참조) 관리 콘솔(`imqadmin`)을 사용하여 연결 팩토리 관리 대상 객체를 만들 때 속성을 설정합니다.
- 클라이언트 응용 프로그램을 시작하는 명령에 `-D` 옵션을 사용합니다(*Message Queue Java Client Developer's Guide* 참조).
- 클라이언트 응용 프로그램 코드에서 프로그래밍 방식으로 연결 팩토리를 만든 후 API 호출을 사용하여 해당 연결 팩토리의 속성을 설정합니다(*Message Queue Java Client Developer's Guide* 참조).

단일 서블릿을 사용하여 다중 브로커에 액세스

다중 브로커를 실행할 경우 다중 Web Server 및 서블릿 인스턴스를 구성하지 않아도 됩니다. 동시에 실행 중인 브로커 간에 단일 Web Server와 HTTPS 터널 서블릿을 공유할 수 있습니다. 다중 브로커 인스턴스가 단일 터널 서블릿을 공유하는 경우 `imqAddressList` 연결 팩토리 속성을 다음과 같이 구성해야 합니다.

```
https://hostName:port/contextRoot/tunnel?ServerName=bkrHostName:instanceName
```

여기서 `bkrHostName`은 브로커 인스턴스 호스트 이름이고 `instanceName`은 클라이언트가 액세스할 특정 브로커 인스턴스의 이름입니다.

`bkrHostName`과 `instanceName`에 정확한 문자열을 입력했는지 확인하려면 브라우저에서 서블릿 URL에 액세스하여 HTTPS 터널 서블릿의 상태 보고서를 생성합니다. 보고서에는 서블릿이 액세스하는 모든 브로커가 나열됩니다.

```

HTTPS tunnel servlet ready.
Servlet Start Time : Thu May 30 01:08:18 PDT 2002
Accepting TCP connections from brokers on port : 7674
Total available brokers = 2
Broker List :
  jpgserv:broker2
  cochin:broker1

```

HTTP 프록시 사용

HTTP 프록시를 사용해서 HTTPS 터널 서블릿에 액세스하는 경우:

- http.proxyHost 시스템 등록 정보를 프록시 서버 호스트 이름으로 설정합니다.
- http.proxyPort 시스템 등록 정보를 프록시 서버 포트 번호로 설정합니다.

클라이언트 응용 프로그램을 시작하는 명령에 `-D` 옵션을 사용하여 이러한 등록 정보를 설정할 수 있습니다.

예 3: Sun Java System Web Server에 HTTPS 터널 서블릿 배포

이 절에서는 Sun Java System Web Server에서 HTTPS 터널 서블릿을 jar 파일과 WAR 파일로 배포하는 방법을 모두 설명합니다. 사용 방법은 Sun Java System Web Server 버전에 따라 다르지만, Servlet 2.2 이상을 지원하지 않는다면 WAR 파일 배포를 처리할 수 없습니다.

Jar 파일로 배포

다음은 브라우저 기반 관리 GUI를 사용하여 Sun Java System Web Server 6.1에 배포하는 경우에 대한 지침입니다. 이 절차는 다음과 같은 일반적인 단계로 구성됩니다.

1. 서블릿 추가
2. 서블릿 가상 경로 구성
3. 서블릿 로드
4. 서블릿 액세스 로그 비활성화

이러한 단계는 다음 하위 절에 설명되어 있습니다. 웹 브라우저를 사용하여 서블릿 URL에 액세스함으로써 HTTPS 터널 서블릿 배포 성공 여부를 확인할 수 있습니다. 상태 정보를 표시해야 합니다.

서블릿 추가

▶ 터널 서블릿을 추가하는 방법

1. 서블릿 탭을 선택합니다.
2. 서블릿 속성 구성을 선택합니다.
3. 서블릿 이름 필드에 터널 서블릿의 이름을 지정합니다.
4. 서블릿 코드(클래스 이름) 필드를 다음 값으로 설정합니다.
`com.sun.messaging.jmq.transport.
 httptunnel.servlet.HttpsTunnelServlet`
5. 서블릿 클래스 경로 필드에 `imqservlet.jar`에 대한 전체 경로를 입력합니다. 예를 들면 다음과 같습니다.
`/usr/share/lib/imq/imqservlet.jar (Solaris)
 /opt/imq/lib/imqservlet.jar (Linux)
 IMQ_HOME/lib/imqservlet.jar (Windows)`
6. 서블릿 인수 필드에 표 C-4와 같은 필수 및 선택 인수를 입력합니다.

표 C-4 HTTPS 터널 서블릿 Jar 파일 배포에 사용되는 서블릿 인수

인수	기본값	필수?	참조
<code>keystoreLocation</code>	없음	예	220페이지의 표 8-8
<code>keystorePassword</code>	없음	예	220페이지의 표 8-8
<code>servletHost</code>	모든 호스트	아니요	323페이지의 표 C-3
<code>servletPort</code>	7674	아니요	323페이지의 표 C-3

인수를 다음 예와 같이 쉼표로 분리합니다.

```
keystoreLocation=keystore_location, keystorePassword=keystore_password,
servletPort=portnumber
```

servletHost 및 servletPort 인수는 Web Server와 브로커간 통신에만 적용되며 기본값에 문제가 있는 경우에만 설정됩니다. 그런 경우에도 브로커 구성 등록 정보를 적절하게 설정해야 합니다(323페이지의 표 C-3 참조). 예를 들면 다음과 같습니다.

```
img.httpsjms.https.servletPort
```

서블릿 가상 경로(서블릿 URL) 구성

▶ 터널 서블릿에 대한 가상 경로(서블릿 URL)를 구성하는 방법

1. 서블릿 탭을 선택합니다.
2. 서블릿 가상 경로 변환 구성을 선택합니다.
3. 가상 경로 필드를 설정합니다.

가상 경로는 터널 서블릿 URL의 */contextRoot/tunnel* 부분입니다.

```
https://hostName:port/contextRoot/tunnel
```

예를 들어, *contextRoot*를 *img*로 설정한 경우 가상 경로 필드는 다음과 같습니다.

```
/img/tunnel
```

4. 서블릿 이름 필드를 327페이지의 "서블릿 추가"의 단계 3과 동일한 값으로 설정합니다.

서블릿 로드

▶ Web Server 시작 시 터널 서블릿을 로드하는 방법

1. 서블릿 탭을 선택합니다.
2. 전역 속성 구성을 선택합니다.
3. 시작 서블릿 필드에 327페이지의 "서블릿 추가"의 단계 3과 동일한 서블릿 이름 값을 입력합니다.

서버 액세스 로그 비활성화

서버 액세스 로그를 비활성화할 필요는 없지만 그렇게 하면 성능이 향상됩니다.

▶ 서버 액세스 로그를 비활성화하는 방법

1. 상태 탭을 선택합니다.
2. 로그 기본 설정 페이지를 선택합니다.
3. 로그 클라이언트 액세스 제어를 사용하여 로깅을 비활성화합니다.

WAR 파일로 배포

다음은 Sun Java System Web Server 6.0 서비스에 배포하는 경우에 대한 지침입니다. 웹 브라우저를 사용하여 서블릿 URL에 액세스하면 HTTPS 터널 서블릿의 배포 성공 여부를 확인할 수 있습니다. 상태 정보를 표시해야 합니다.

HTTPS 터널 서블릿을 배포하기 전에 Web Server의 클래스 경로에 JSSE jar 파일이 포함되어 있는지 확인합니다. 이 작업을 수행하는 가장 간단한 방법은 `jsse.jar`, `jnet.jar` 및 `jcercert.jar` 파일을 `IWS60_TOPDIR/bin/https/jre/lib/ext`로 복사하는 것입니다.

또한, HTTPS 터널 서블릿을 배포하기 전에 키 저장소 파일의 위치를 가리키고 키 저장소 비밀번호를 지정하도록 배포 설명자를 수정해야 합니다.

▶ HTTPS 터널 서블릿 WAR 파일을 수정하는 방법

1. WAR 파일을 임시 디렉토리로 복사합니다.

```
cp /usr/share/lib/imq/imqhttps.war /tmp (Solaris)
```

```
cp /opt/imq/lib/imqhttps.war /tmp (Linux)
```

```
cp IMQ_HOME/lib/imqhttps.war /tmp (Windows)
```

2. 임시 디렉토리를 현재 디렉토리로 만듭니다.

```
$ cd /tmp
```

3. WAR 파일의 내용을 추출합니다.

```
$ jar xvf imqhttps.war
```

4. WAR 파일의 배포 설명자를 나열합니다.

```
$ ls -l WEB-INF/web.xml
```

5. `keystoreLocation`과 `keystorePassword` 인수(필요한 경우 `servletPort` 및 `servletHost` 인수 포함)에 정확한 값을 제공하도록 `web.xml` 파일을 편집합니다.

6. WAR 파일의 내용을 다시 어셈블합니다.

```
$ jar uvf imqhttps.war WEB-INF/web.xml
```

이제 수정된 `imghttps.war` 파일을 사용하여 HTTPS 터널 서블릿을 배포할 수 있습니다. (키 저장소 비밀번호 노출이 우려되는 경우에는 파일 시스템 권한을 사용하여 `imghttps.war` 파일에 대한 액세스를 제한할 수 있습니다.)

▶ HTTPS 터널 서블릿을 WAR 파일로 배포하는 방법

1. 브라우저 기반 관리 GUI에서 가상 서버 클래스 탭을 선택합니다. 클래스 관리를 누릅니다.
2. 해당 가상 서버 클래스 이름(예: `defaultClass`)을 선택하고 관리 버튼을 누릅니다.
3. 가상 서버 관리를 선택합니다.
4. 해당 가상 서버 이름을 선택하고 관리 버튼을 누릅니다.
5. 웹 응용 프로그램 탭을 선택합니다.
6. 웹 응용 프로그램 배포를 누릅니다.
7. WAR 파일 위치와 WAR 파일 경로 필드에서 수정된 `imghttps.war` 파일(329페이지의 "HTTPS 터널 서블릿 WAR 파일을 수정하는 방법" 참조)을 가리키는 적절한 값을 선택합니다.
8. 응용 프로그램 URI 필드에 경로를 입력합니다.

응용 프로그램 URI 필드 값은 터널 서블릿 URL의 `/contextRoot` 부분입니다.

```
https://hostName:port/contextRoot/tunnel
```

예를 들어, `contextRoot`를 `img`로 설정한 경우 응용 프로그램 URI 필드는 다음과 같습니다.

```
/img
```

9. 서블릿을 배포할 설치 디렉토리 경로(일반적으로 Sun Java System Web Server 설치 루트 아래)를 입력합니다.
10. 확인을 누릅니다.
11. Web Server 인스턴스를 다시 시작합니다.

이제 서블릿을 다음 주소에서 사용할 수 있습니다.

```
https://hostName:port/img/tunnel
```

이제 클라이언트에서 이 URL을 통해 보안 HTTPS 연결을 사용하는 메시지 서비스에 연결할 수 있습니다.

예 4: Sun Java System Application Server 7.0에 HTTPS 터널 서블릿 배포

이 절에서는 Sun Java System Application Server 7.0에서 HTTPS 터널 서블릿을 WAR 파일로 배포하는 방법을 설명합니다.

다음 두 단계를 거쳐야 합니다.

- Application Server 7.0 배포 도구를 사용하여 HTTPS 터널 서블릿 배포
- Application Server 인스턴스의 `server.policy` 파일 수정

배포 도구 사용

▶ Application Server 7.0 환경에 HTTPS 터널 서블릿을 배포하는 방법

1. 웹 기반 관리 GUI에서 다음을 선택합니다.

Application Server 인스턴스 > server1 > 응용 프로그램 > 웹 응용 프로그램

2. 배포 버튼을 누릅니다.

3. 파일 경로 텍스트 필드에 HTTPS 터널 서블릿 WAR 파일(`imqhttps.war`)의 위치를 입력합니다.

`imqhttps.war` 파일의 위치는 운영 체제에 따라 다릅니다(부록 A, "Message Queue 데이터의 위치" 참조).

4. 확인을 누릅니다.

5. 다음 화면에서 컨텍스트 루트 텍스트 필드의 값을 설정합니다.

컨텍스트 루트 필드 값은 터널 서블릿 URL의 `/contextRoot` 부분입니다.

```
https://hostName:port/contextRoot/tunnel
```

예를 들어, 컨텍스트 루트 필드를 다음과 같이 설정할 수 있습니다.

```
/imq
```

6. 확인을 누릅니다.

다음 화면에 터널 서블릿이 성공적으로 배포되었고 기본적으로 사용되며 다음 위치(이 예의 경우)에 있다고 표시됩니다.

```
/var/opt/SUNWappserver7/domains/domain1/server1/applications/
j2ee-modules/imqhttps_1
```

이제 서블릿을 다음 주소에서 사용할 수 있습니다.

```
https://hostName:port/contextRoot/tunnel
```

이제 클라이언트에서 이 URL을 통해 HTTPS 연결을 사용하는 메시지 서비스에 연결할 수 있습니다.

server.policy 파일 수정

Application Server 7.0이 시행하는 일련의 기본 보안 정책은 수정하지 않으면 HTTPS 터널 서블릿이 Message Queue 브로커에서 연결을 수신할 수 없도록 합니다.

Application Server 인스턴스마다 해당 보안 정책이나 규칙이 포함된 파일이 있습니다. 예를 들어, Solaris의 server1 인스턴스의 경우 이 파일의 위치는 다음과 같습니다.

```
/var/opt/SUNWappserver7/domains/domain1/server1/config/
server.policy
```

터널 서블릿이 Message Queue 브로커에서 연결을 수신하도록 하려면 이 파일에 항목을 추가해야 합니다.

▶ Application Server의 server.policy 파일을 수정하는 방법

1. server.policy 파일을 엽니다.
2. 다음 항목을 추가합니다.

```
grant codeBase
"file:/var/opt/SUNWappserver7/domains/domain1/server1/
    applications/j2ee-modules/imqhttps_1/-"
{
    permission java.net.SocketPermission "*",
        "connect,accept,resolve";
};
```

브로커를 Windows 서비스로 사용

이 부록에서는 서비스 관리자 유틸리티(imqsvcadmin)를 사용해서 Windows 서비스로 실행되는 브로커를 설치, 쿼리, 제거하는 방법을 설명합니다.

브로커를 Windows 서비스로 실행

Message Queue를 설치할 때 브로커를 Windows 서비스로 설치할 수 있습니다. Message Queue를 설치한 후에 imqsvcadmin을 사용하여 브로커를 Windows 서비스로 설치할 수도 있습니다.

브로커를 Windows 서비스로 설치한다는 것은 시스템을 시작할 때 시작하여 종료할 때 까지 백그라운드로 실행하는 것을 의미합니다. 따라서 추가 인스턴스를 시작하는 경우가 아니면 imqbrokerd 명령을 사용해서 브로커를 시작하지 않습니다. 브로커에 시작 옵션을 전달하려면 imqsvcadmin 명령에 -args 인수를 사용하고(335페이지의 표 D-2 참조) imqbrokerd 명령에 사용하는 것과 같은 옵션을 지정합니다(134페이지의 "브로커 시작" 참조). imqcmd 명령을 사용해서 보통의 경우와 같이 브로커를 제어합니다.

Windows 서비스를 실행할 때, 작업 관리자는 브로커를 두 개의 실행 프로세스로 나열합니다. 첫 번째 프로세스는 Windows 고유의 서비스 래퍼인 imqbrokersvc.exe입니다. 두 번째 프로세스는 실제로 브로커를 실행하는 Java runtime입니다.

한 번에 하나의 브로커만 Windows 서비스로 설치하여 실행할 수 있습니다.

서비스 관리자 유틸리티(imqsvcadmin)

서비스 관리자 유틸리티(imqsvcadmin)를 사용하면 (Windows 서비스로 실행되는) 브로커를 설치, 쿼리, 제거할 수 있습니다. 이 절에서는 imqsvcadmin 명령의 기본 구문을 설명하고, 하위 명령의 목록을 제공하고, imqsvcadmin 명령 옵션을 요약하고, 이 명령을 사용해서 특정 작업을 수행하는 방법을 설명합니다.

imqsvcadmin 명령의 구문

imqsvcadmin 명령의 일반 구문은 다음과 같습니다.

```
imqsvcadmin subcommand [options]
```

```
imqsvcadmin -h
```

-v, -h 또는 -H 옵션을 지정하는 경우 명령줄에 지정된 다른 하위 명령은 실행되지 않습니다. 예를 들어, 다음 명령을 입력하면 도움말 정보는 표시되지만 query 하위 명령은 실행되지 않습니다.

```
imqsvcadmin query -h
```

imqsvcadmin 하위 명령

Message Queue 서비스 관리자 유틸리티(imqsvcadmin)에는 표 D-1에 나열된 하위 명령이 포함됩니다.

표 D-1 imqsvcadmin 하위 명령

하위 명령	설명
install	서비스를 설치하고 시작 옵션을 지정합니다.
query	imqsvcadmin 명령의 시작 옵션을 표시합니다. 여기에는 서비스의 수동 또는 자동 시작, 위치, Java 런타임 위치, 시작할 때 브로커에 전달되는 인수 값이 포함됩니다.
remove	서비스를 제거합니다.

imqsvcadmin 옵션 요약

표 D-2에는 imqsvcadmin 명령의 옵션이 나열되어 있습니다. 사용 설명은 다음에서 해당 작업 기반 절을 참조하십시오.

표 D-2 imqsvcadmin 옵션

옵션	설명
-h	사용 도움말을 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
-javahome <i>path</i>	Java 2와 호환할 수 있는 대체 런타임의 경로로 지정하여 사용합니다(기본값은 시스템의 런타임이나 Message Queue와 함께 제공되는 런타임 사용). 예: imqsvcadmin -install -javahome d:\jdk1.4
-jrehome <i>path</i>	Java 2 호환 JRE의 경로를 지정합니다. 예: imqsvcadmin -install -jrehome d:\jre\1.4
-vmargs <i>arg</i> [[<i>arg</i>]...]	브로커 서비스를 실행하는 Java VM에 전달할 추가 인수를 지정합니다. (이러한 인수는 Windows 서비스 제어판 시작 매개 변수 필드에서 지정할 수도 있습니다.) 예: -vmargs "-Xms16m -Xmx128m"
-args <i>arg</i> [[<i>arg</i>]...]	브로커 서비스에 전달할 추가 명령줄 인수를 지정합니다. imqbrokerd 옵션에 대한 설명은 134페이지의 "브로커 시작"을 참조하십시오. (이러한 인수는 Windows 서비스 제어판 시작 매개 변수 필드에서 지정할 수도 있습니다.) 예를 들면 다음과 같습니다. imqsvcadmin -install -args "-passfile d:\imqpassfile"

-javahome, -vmargs, -args 옵션을 사용해서 지정하는 정보는 Windows 레지스트리의 다음 경로에 있는 JavaHome, JVMArgs, ServiceArgs 키 아래에 저장됩니다.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet
  \Services\IMQ_Broker\Parameters
```

브로커 서비스 제거

브로커 서비스를 제거하기 전에 imqcmd shutdown bkr 명령을 사용해서 브로커를 종료해야 합니다. 그런 다음 imqsvcadmin remove 명령을 사용해서 서비스를 제거한 후 컴퓨터를 다시 시작합니다.

브로커 서비스 재구성

서비스를 재구성하려면 먼저 서비스를 제거한 후 다시 설치하며 `-args` 인수를 사용해서 다른 시작 옵션을 지정합니다.

대체 Java 런타임 사용

`-javahome` 또는 `-jrehome` 옵션을 사용해서 대체 Java 런타임의 위치를 지정할 수 있습니다. Windows 서비스 제어판 시작 매개 변수 필드에서 이러한 옵션을 지정할 수도 있습니다. 시작 매개 변수 필드에서는 백슬래시(\)를 제어 문자로 사용하기 때문에 경로 분리자로 사용하려면 `-javahome d:\jdk1.3`과 같이 두 번 입력해야 합니다.

브로커 서비스 쿼리

브로커 서비스의 시작 옵션을 결정하려면 `imqsvcadmin` 명령에 `-q` 옵션을 사용합니다.

```
imqsvcadmin -query

Service iMQ_Broker is installed.
Display Name: iMQ_Broker
Start Type: Manual
Binary location: C:Program FilesSun Microsystems
                Message Queue 3.5\bin\imqbrokersvc
JavaHome: c:\j2sdk1.4.0
Broker Args: -passfile d:\imqpassfile
```

문제 해결

서비스를 시작하려 할 때 오류가 발생한 경우에는 다음을 수행하여 기록된 오류 이벤트를 확인할 수 있습니다.

- ▶ 기록된 서비스 오류 이벤트를 보는 방법
 1. 이벤트 뷰어를 시작합니다.
 2. 로그 > 응용 프로그램을 확인합니다.
 3. 보기 > 새로 고침을 선택하여 오류 이벤트를 봅니다.

기술 노트

이 부록에서는 다음 내용을 간단하게 다룹니다.

- 시스템 클럭 설정
- OS 정의 파일 설명자 제한
- 지속성 데이터 보안

시스템 클럭 설정

Message Queue 시스템 사용 시 시스템 클럭 동기화에 주의하고 클럭을 역방향으로 설정하지 않아야 합니다.

동기화 권장

Message Queue 시스템과 상호 작용하는 모든 호스트에서 클럭을 동기화하는 것이 좋습니다. 클럭 동기화는 메시지 만료(TimeToLive)를 사용할 경우에 특히 중요합니다. 호스트의 클럭을 동기화하지 않으면 TimeToLive가 예상한 대로 작동하지 않을 수 있습니다 (메시지가 전달되지 않을 수 있음). 브로커를 시작하기 전에 클럭을 동기화해야 합니다.

Solaris. 로컬 호스트에서 rdate 명령을 실행하여 원격 호스트와 동기화할 수 있습니다 (이 명령 실행에는 슈퍼유저(루트) 권한이 필수). 예를 들어, 다음 명령은 로컬 호스트 (Host 2)를 원격 호스트 Host1과 동기화합니다.

```
# rdate Host1
```

Linux. 이 명령은 Solaris에서와 비슷하지만 다음 -s 옵션을 제공해야 합니다.

```
# rdate -s Host1
```

Windows.net 명령을 **time** 하위 명령과 함께 실행하여 로컬 호스트를 원격 호스트와 동기화할 수 있습니다. 예를 들어, 다음 명령은 로컬 호스트(Host 2)를 원격 호스트 Host1과 동기화합니다.

```
net time \\Host1 /set
```

역방향 시스템 클럭 설정 금지

Message Queue 브로커를 실행하는 시스템에서 시스템 클럭을 역방향으로 설정하지 마십시오. Message Queue는 트랜잭션, 영구 가입과 같은 내부 객체를 식별하기 위해 타임스탬프를 사용합니다. 시스템 클럭을 역방향으로 설정하면 이론적으로 중복 내부 식별자가 생성될 수 있습니다. 브로커는 식별자에 임의성을 부여하고 실행 중에 클럭 이동을 감지하는 방법으로 이 문제점을 보완하려고 합니다. 그러나 브로커가 실행되지 않을 때 시스템 클럭이 역방향으로 크게 이동하면 식별자가 중복될 가능성이 약간 있습니다.

브로커가 실행 중인 시스템에서 시스템 클럭을 몇 초 이상 역방향으로 설정해야 하는 경우 트랜잭션 또는 영구 가입이 없을 때 설정하거나 브로커가 실행 중이 아닐 때 설정 작업을 수행한 다음 클럭을 이동한 시간만큼 대기한 후 브로커를 다시 실행하는 것이 좋습니다.

가장 좋은 방법은 브로커를 시작하기 전에 클럭을 동기화한 다음 적절한 기술을 사용하여 클럭이 배포된 이후에 많이 이동되지 않도록 하는 것입니다.

OS 정의 파일 설명자 제한

Solaris 및 Linux 플랫폼의 경우 클라이언트나 브로커가 실행 중인 셸에서 클라이언트가 사용할 가능한 파일 설명자의 수에는 소프트 한계가 있습니다. Message Queue 시스템에서는 각 클라이언트가 만드는 연결 또는 각 브로커가 받아들이는 연결에 이 파일 설명자 중 하나를 사용합니다. 지속성 메시지를 갖는 각 대상 역시 파일 설명자를 사용합니다.

결과적으로 이러한 요소가 연결 수를 제한합니다. 따라서 파일 설명자 제한을 변경하지 않으면 Solaris에서는 256개, Linux에서는 1024개를 초과하는 연결에서 브로커나 클라이언트를 실행할 수 없습니다(지속성을 위해 파일 설명자를 사용하기 때문에 연결 제한은 실제로 더 낮음).

파일 설명자 제한을 변경하려면 ulimit 설명서 페이지를 참조하십시오. 클라이언트 또는 브로커가 실행될 각 셸에서 한계를 변경해야 합니다.

지속성 데이터 보안

브로커는 정보 가운데 일시적으로 저장되는 메시지 파일을 포함할 수 있는 영구 저장소를 사용합니다. 이러한 메시지에는 소유 정보가 포함될 수 있기 때문에 인증되지 않은 액세스로부터 데이터 저장소를 안전하게 보호하는 것이 좋습니다.

브로커는 기본 제공 또는 플러그인 지속성을 사용할 수 있습니다.

기본 제공 영구 저장소

기본 제공 지속성을 사용하는 브로커는 플랫폼에 따라 다른 디렉토리에 있는 플랫폼 파일 데이터 저장소에 지속성 데이터를 작성합니다(부록 A, "Message Queue 데이터의 위치" 참조).

```
.../instances/instanceName/fs350/
```

여기서 *instanceName*은 브로커 인스턴스를 식별하는 이름입니다.

instanceName/filestore/ 디렉토리는 브로커 인스턴스가 처음으로 시작될 때 생성됩니다. 이 디렉토리 보안 절차는 브로커가 실행 중인 운영 체제에 따라 다릅니다.

Solaris 및 Linux. `IMQ_VARHOME/instances/instanceName/filestore/` 디렉토리에 대한 사용 권한은 브로커 인스턴스를 시작한 사용자의 `umask`에 따라 다릅니다. 따라서, `umask`를 적절하게 설정하여 브로커 인스턴스 시작 권한과 지속성 파일 읽기 권한을 제한할 수 있습니다. 또는, 관리자(수퍼유저)가 `IMQ_VARHOME/instances` 디렉토리에 대한 사용 권한을 700으로 설정하여 지속성 데이터를 안전하게 보호할 수 있습니다.

Windows. 사용 중인 Windows 운영 체제에서 제공하는 메커니즘을 사용하여 `IMQ_VARHOME/instances/instanceName/filestore/` 디렉토리에 대한 사용 권한을 설정할 수 있습니다. 여기에는 일반적으로 디렉토리에 대한 등록 정보 대화 상자 열기가 포함됩니다.

플러그인 영구 저장소

플러그인 지속성을 사용하는 브로커는 JDBC 호환 데이터베이스에 지속성 데이터를 작성합니다.

데이터베이스 서버에서 관리하는 데이터베이스(예: Oracle 데이터베이스)의 경우 Message Queue 데이터베이스 테이블(이름이 'IMQ'로 시작하는 테이블)에 액세스할 사용자 아이디 및 비밀번호를 만드는 것이 좋습니다. 데이터베이스에서 개별 테이블을 보호할 수 없는 경우 Message Queue 브로커 전용 데이터베이스를 만듭니다. 사용자 아이디/비밀번호 액세스 작성 방법에 대한 자세한 내용은 데이터베이스 공급업체 설명서를 참조하십시오.

브로커가 데이터베이스 연결을 여는 데 필요한 사용자 아이디와 비밀번호는 브로커 구성 등록 정보에서 제공될 수 있습니다. 브로커 시작 시 사용자 아이디와 비밀번호를 명령줄 옵션으로 제공하는 것이 보다 안전합니다(Message Queue 관리 설명서, 부록 A, "플러그인 지속성 설정" 참조).

데이터베이스의 JDBC 드라이버를 통해 브로커가 직접 액세스하는 내장 데이터베이스(예: Cloudscape 데이터베이스)의 경우 일반적으로 지속성 데이터가 저장될 디렉토리에 대한 파일 사용 권한(위 "기본 제공 영구 저장소" 설명 참조)을 설정하여 보호합니다. 브로커와 imqdbmgr 유틸리티를 모두 사용하여 데이터베이스를 읽고 쓸 수 있게 하려면 해당 유틸리티와 브로커를 동일한 사용자가 실행해야 합니다.

Message Queue 자원 어댑터

Message Queue에는 JMS 자원 어댑터가 포함되어 있습니다.

자원 어댑터는 J2EECA (J2EE Connector Architecture) 1.5 사양에 따라 J2EE 1.4 호환 Application Server에 추가 기능을 플러그 인하는 표준화된 방식입니다. 이 구조를 사용하면 모든 J2EE 1.4 호환 Application Server가 표준화된 방식으로 외부 시스템과 상호 작용할 수 있습니다. 이러한 외부 시스템에는 JMS 공급자와 같은 다양한 메시징 시스템뿐 아니라 다양한 EIS (enterprise information system)도 포함됩니다.

J2EECA 1.5를 통해 가능하게 된 표준화된 상호 작용으로는 다양한 종류의 Message-Driven Bean 컨테이너에 대한 지원을 비롯하여 연결 풀링, 스레드 풀링, 트랜잭션 및 보안 컨텍스트 전파를 들 수 있습니다. 또한 이 사양은 연결 팩토리 및 기타 관리 대상 객체를 만드는 표준화된 방식도 포함하고 있습니다.

JMS 자원 어댑터를 Application Server에 플러그 인하면 Application Server 환경에서 배포되어 실행 중인 J2EE 구성 요소가 JMS 메시지를 교환할 수 있습니다. 이러한 구성 요소에 필요한 JMS 연결 팩토리 및 대상 관리 대상 객체는 J2EE Application Server 관리 도구를 사용하여 만들고 구성할 수 있습니다.

그러나 메시지 서버 및 물리적 대상 관리와 같은 다른 관리 작업은 J2EECA 사양에 포함되어 있지 않으므로 공급자 고유의 도구를 통해서만 수행할 수 있습니다.

Message Queue 자원 어댑터는 Sun J2EE 1.4 Application Server에 내장되어 있습니다. 그러나 Message Queue 자원 어댑터는 그 밖의 J2EE 1.4 Application Server에서 아직 인증되지 않았습니다.

Message Queue 자원 어댑터는 하나의 파일(imqjmsra.rar)로 이루어졌으며, [부록 A, "Message Queue 데이터의 위치"](#)에 표시된 것처럼 운영 체제에 따라 다른 디렉토리에 있습니다. imqjmsra.rar 파일에는 Application Server에서 어댑터를 사용하려면 있어야 하는 jar 파일뿐만 아니라 자원 어댑터 배포 설명자(ra.xml)가 포함되어 있습니다.

모든 J2EE 1.4 호환 Application Server에서 해당 Application Server와 함께 제공되는 자원 어댑터 배포 및 구성 지침에 따라 Message Queue 자원 어댑터를 사용할 수 있습니다. 상용 J2EE 1.4 Application Server가 출시되고 Message Queue 자원 어댑터가 이러한 Application Server에서 인증되면 이 부록에서 관련 배포 및 구성 절차에 대한 구체적인 정보를 제공할 것입니다.

선택적 JMS 기능의 Message Queue 구현

JMS 사양은 선택 사항인 특정 항목을 나타냅니다. 각 JMS 공급자(공급업체)가 해당 항목의 구현 여부를 선택합니다. 이러한 선택 항목을 처리하는 Message Queue 제품은 아래와 같습니다.

표 G-1 선택적 JMS 기능

JMS 사양 섹션	설명 및 Message Queue 처리
3.4.3 JMSMessageID	<p>"메시지 아이디는 메시지 작성 및 메시지의 크기 증가에 어느 정도 영향을 미치기 때문에 응용 프로그램에서 메시지 아이디를 사용하지 않는다는 힌트가 있을 경우 일부 JMS 공급자가 메시지 오버헤드를 최적화할 수 있습니다. JMS Message Producer는 메시지 아이디 비활성화를 위한 힌트를 제공합니다."</p> <p>Message Queue 구현: 제품은 메시지 아이디 생성을 비활성화하지 않습니다(MessageProducer의 모든 setDisableMessageID() 호출이 무시됨). 모든 메시지에는 유효한 MessageID 값이 있습니다.</p>
3.4.12 메시지 헤더 필드 대체	<p>"JMS는 관리자가 이러한 헤더 필드 값을 대체하는 방법에 대해 특별히 정의하지 않습니다. JMS 공급자는 이 관리 옵션을 지원하지 않아도 됩니다."</p> <p>Message Queue 구현: Message Queue 제품은 연결 팩토리 관리 대상 객체 구성을 통해 메시지 헤더 필드 값의 관리 대체를 지원합니다(187페이지의 표 7-3 참조).</p>
3.5.9 JMS 정의 등록 정보	<p>"JMS는 JMS 정의 등록 정보에 대한 'JMSX' 등록 정보 이름 접두어를 예약합니다." "별도로 명시되지 않은 경우 이러한 등록 정보 지원은 선택 사항입니다."</p> <p>Message Queue 구현: JMS 1.1 사양에 정의된 JMSX 등록 정보는 Message Queue 제품에서 지원합니다(187페이지의 표 7-3 참조).</p>

표 G-1 선택적 JMS 기능(계속)

JMS 사양 섹션	설명 및 Message Queue 처리
3.5.10 공급자별 등록 정보	<p>"JMS는 공급자별 등록 정보에 대한 'JMS_<vendor_name>' 등록 정보 이름 접두어를 예약합니다."</p> <p>Message Queue 구현: 공급자별 등록 정보는 공급자 고유 클라이언트에서 JMS 사용을 지원하는 데 필요한 특수 기능을 제공하는 것을 목적으로 합니다. 이러한 기능은 JMS간 메시징에 사용되지 않습니다. Message Queue 3.5 SP1은 공급자별 등록 정보를 사용하지 않습니다.</p>
4.4.8 분산 트랜잭션	<p>"JMS에서는 공급자가 분산 트랜잭션을 지원하지 않아도 됩니다."</p> <p>Message Queue 구현: 분산 트랜잭션은 Message Queue 제품의 본 릴리스에서 지원됩니다(47페이지의 "분산 트랜잭션" 참조).</p>
4.4.9 다중 세션	<p>"PTP <지점간 배포 모델>의 경우 JMS는 동일한 대기열에 대한 동시 QueueReceivers의 의미를 지정하지 않지만, 공급자가 이를 지정하는 것을 금지하지는 않습니다." 자세한 내용은 JMS 사양의 5.8 절을 참조하십시오.</p> <p>Message Queue 구현: Message Queue 구현에서는 다중 사용자로의 대기열 전달을 지원합니다. 자세한 내용은 77페이지의 "다중 사용자로의 대기열 전달"을 참조하십시오.</p>

Message Queue 인터페이스의 안정성

Sun Java System Message Queue는 관리자가 관리 작업을 자동화할 때 사용 가능한 여러 인터페이스를 사용합니다. 표 H-1에서는 안정성, 즉 향후 제품 버전에서 변경되지 않을 확률을 기준으로 이 인터페이스들을 분류합니다. 분류 체계는 347페이지의 표 H-2에서 확인할 수 있습니다.

표 H-1 Message Queue 인터페이스의 안정성

인터페이스	분류
imqbrokerd 명령줄 인터페이스	변화
imqadmin 명령줄 인터페이스	불안정
imqcmd 명령줄 인터페이스	변화
imqdbmgr 명령줄 인터페이스	불안정
imqkeytool 명령줄 인터페이스	변화
imqobjmgr 명령줄 인터페이스	변화
imqusermgr 명령줄 인터페이스	불안정
imqobjmgr 명령 파일	변화
imqbrokerd 명령	안정
imqadmin 명령	불안정
imqcmd 명령	안정
imqdbmgr 명령	불안정
imqkeytool 명령	안정
imqobjmgr 명령	안정
imqusermgr 명령	불안정

표 H-1 Message Queue 인터페이스의 안정성(계속)

인터페이스	분류
JMS API (javax.jms)	표준
JAXM API (javax.xml)	표준
C-API	변화
메시지 기반 모니터링 API	변화
관리 대상 객체 API (com.sun.messaging)	변화
imq.jar 위치 및 이름	안정
jms.jar 위치 및 이름	변화
imqbroker.jar 위치 및 이름	개인
imqutil.jar 위치 및 이름	개인
imqadmin.jar 위치 및 이름	개인
imqservlet.jar 위치 및 이름	변화
imqhttp.war 위치 및 이름	변화
imqhttps.war 위치 및 이름	변화
imqjmsra.rar 위치 및 이름	변화
imqxm.jar 위치 및 이름	변화
jaxm-api.jar 위치 및 이름	변화
saa-api.jar 위치 및 이름	변화
saa-impl.jar 위치 및 이름	변화
activation.jar 위치 및 이름	변화
mail.jar 위치 및 이름	변화
dom4j.jar 위치 및 이름	개인
fscontext.jar 위치 및 이름	불안정
imqbrokerd, imqadmin, imqcmd, imqdbmgr, imqkeytool, imqobjmgr, imqusermgr의 출력	불안정
브로커 로그 파일 위치 및 내용 형식	불안정
passfile	불안정
accesscontrol.properties	불안정

표 H-2 인터페이스 안정성 분류 체계

분류	설명
개인	고객이 직접 사용하지 않음. 임의의 릴리스에서 변경 또는 삭제될 수 있습니다.
변화	고객용. 주 릴리스(예: 3.0, 4.0) 또는 부 릴리스(예: 3.1, 3.2)에 호환되지 않는 변경이 있을 수 있습니다. 신중하고 천천히 변경됩니다. 모든 변경 사항이 호환성을 갖도록 타당한 조치가 이행되지만 보장되지는 않습니다.
안정	고객용. 주 릴리스(예: 3.0, 4.0)에 한해 호환되지 않는 변경이 있을 수 있습니다.
표준	고객용. 이 인터페이스는 공식 표준에서 정의하며 표준 기관이 관리합니다. 이 인터페이스와 호환되지 않는 변경은 거의 없습니다.
불안정	고객용. 주 릴리스(예: 3.0, 4.0) 또는 부 릴리스(예: 3.1, 3.2)에 호환되지 않는 변경이 있을 수 있습니다. 이 인터페이스는 향후 릴리스에서 사실상 그리고 호환되지 않는 방식으로 제거 또는 변경될 수 있음을 고객에게 알립니다. 고객은 불안정한 인터페이스에 대한 명시적인 종속성을 생성하지 않는 것이 바람직합니다.

용어집

이 용어집에서는 Sun Java System Message Queue 사용 중에 접할 수 있는 용어와 개념에 대한 정보를 제공합니다.

게시/가입 전달 모델. 게시자와 가입자는 일반적으로 익명이며 동적으로 주제를 게시하거나 가입합니다. 시스템은 어떤 주제의 여러 게시자로부터 도착한 메시지를 여러 가입자에게 배포합니다.

관리 대상 객체. 사전 구성된 Message Queue 객체로서, 관리자가 작성하고 하나 이상의 JMS 클라이언트가 사용하는 연결 팩토리나 대상

관리 대상 객체를 사용하면 JMS 클라이언트가 공급자 독립성을 갖게 되어 JMS 클라이언트를 공급자 고유 특성과 분리할 수 있습니다. 이 객체는 관리자가 JNDI 이름 공간에 배치하며 JMS 클라이언트가 JNDI 조회를 사용하여 액세스합니다.

구성 파일. 브로커 구성에 사용하는 Message Queue 설정을 포함한 하나 이상의 텍스트 파일. 인스턴스별 또는 클러스터와 관련된 등록 정보로 구성됩니다.

권한 부여. 사용자가 연결 서비스나 대상과 같은 메시지 서비스 자원에 액세스 가능한지 여부를 메시지 서비스가 결정하는 과정

대기열. 관리자가 지점간 전달 모델을 구현하기 위해 생성하는 객체. 메시지를 사용하는 클라이언트가 비활성 상태이더라도 대기열은 항상 메시지 보관이 가능합니다. 대기열은 생성자와 사용자 사이의 중간 저장소 역할을 합니다.

대상. 생성된 메시지가 라우팅 및 이후 사용자로의 전달을 위해 이동하는 Message Queue 메시지 서버상의 물리적 대상. 물리적 대상은 클라이언트가 자신이 누구를 위해 메시지를 생성하며 누구로부터 받은 메시지를 사용하는지 그 대상을 지정할 때 사용하는 관리 대상 객체에 의해 식별 및 캡슐화됩니다.

데이터 저장소. 브로커가 필요로 하는 정보(영구 가입, 대상 관련 데이터, 지속성 메시지, 감사 데이터 등)가 영구적으로 저장되는 데이터베이스

도메인. JMS 클라이언트가 JMS 메시징 작업을 프로그래밍할 때 사용하는 객체 집합. 지점간 전달 모델을 위한 도메인과 게시/가입 전달 모델을 위한 도메인 등 두 가지 유형의 프로그래밍 도메인이 있습니다.

메시지 선택기. 사용자가 JMS 메시지 헤더의 등록 정보 값(선택기)을 기준으로 메시지를 선택하는 방식. 메시지 서비스는 메시지 선택기에 지정된 기준에 따라 메시지 필터링 및 라우팅을 수행합니다.

메시지 서비스. Message Queue 메시지 서버 참조

메시지. JMS 클라이언트가 사용하는 비동기 요청, 보고서 또는 이벤트. 메시지는 헤더(필드 추가 가능)와 본문으로 구성됩니다. 메시지 헤더는 표준 필드 및 선택적 등록 정보를 지정합니다. 메시지 본문은 전송되는 데이터를 포함합니다.

메시징. 엔터프라이즈 응용 프로그램이 사용하는 비동기 요청, 보고서 또는 이벤트 시스템으로서, 느슨하게 연결된 응용 프로그램들이 신뢰성 있고 안전하게 정보를 전송할 수 있게 합니다.

비동기 통신. 메시지 발신자가 발신 메소드가 반환될 때까지 기다릴 필요 없이 다른 작업을 진행할 수 있는 통신 모드

브로커. 메시지 라우팅, 전달, 지속성, 보안 및 로깅을 관리하고 관리자가 성능 및 자원 사용을 모니터링하고 조정할 수 있는 인터페이스를 제공하는 Message Queue 실체

사용. 대상에서 보낸 메시지를 메시지 사용자가 받는 것

사용자. 대상으로부터의 메시지 수신에 사용하는 세션에서 작성한 객체(메시지 사용자). 지점간 전달 모델의 사용자는 수신기 또는 브라우저(대기열 수신기나 대기열 브라우저)이고, 게시/가입 전달 모델의 사용자는 가입자(주제 가입자)입니다.

사용자 그룹. Message Queue 클라이언트 사용자가 연결 및 대상과 같은 Message Queue 메시지 서버 자원의 액세스 권한을 받기 위해 소속되는 그룹

생성. 메시지가 대상에게 전달될 수 있도록 클라이언트 런타임으로 전달하는 것

생성자. 세션에서 생성한 객체(메시지 생성자)로서 대상에게 메시지를 보낼 때 사용됩니다. 지점간 전달 모델에서 생성자는 발신자(대기열 발신자)이며, 게시/가입 전달 모델의 생성자는 게시자(주제 게시자)입니다.

세션. 메시지를 보내고 받는 단일 스레드 컨텍스트. 대기열 세션이거나 주제 세션이 될 수 있습니다.

연결. 1) Message Queue 메시지 서버와의 활성 연결. 대기열 연결이거나 주제 연결이 될 수 있습니다. 2) Message Queue 메시지 서버의 기반이 되는 연결을 사용하여 메시지를 생성하고 사용하는 세션의 팩토리

연결 팩토리. 클라이언트가 Message Queue 메시지 서버와 연결을 생성할 때 사용하는 관리 대상 객체. 대기열 연결 팩토리 객체나 주제 연결 팩토리 객체가 될 수 있습니다.

전달 모드. 메시징의 신뢰성 지표. 메시지가 단 한 차례 전달되어 성공적으로 사용되는지(지속성 전달 모드) 또는 최대 1회 전달되는지(비지속성 전달 모드) 여부

전달 모델. 메시지가 전달되는 모델로서, 지점간 모델 또는 게시/가입 모델이 있습니다. JMS에서는 각각 특정 클라이언트 런타임 객체와 특정 대상 유형(대기열 또는 주제)을 사용하는 별도의 프로그래밍 도메인과 통합 프로그래밍 도메인이 있습니다.

전달 정책. 둘 이상의 메시지 사용자가 등록된 경우 대기열에서 메시지를 라우팅하는 방식에 대한 사양. 단일, 페일오버 및 라운드 로빈 정책이 있습니다.

주제. 관리자가 게시/가입 전달 모델을 구현하기 위해 생성하는 객체. 주제는 자신에게 전달된 메시지의 수집 및 배포를 담당하는 내용 계층상의 노드로 간주할 수 있습니다. 메시지 게시자와 메시지 가입자는 중간에 있는 주제를 통해 구분됩니다.

지점간 전달 모델. 생성자는 특정 대기열로 메시지를 전달하고 사용자는 자신의 메시지를 보관하도록 설정된 대기열로부터 메시지를 가져옵니다. 메시지는 단 한 명의 메시지 사용자에게 전달됩니다.

클라이언트. 다른 클라이언트와 상호 작용하면서 메시지 서비스를 사용하여 메시지를 교환하는 응용 프로그램(또는 소프트웨어 구성 요소)

클라이언트 식별자. 클라이언트를 대신하여 연결 및 그 객체를 Message Queue 메시지 서버가 관리하는 상태와 연관시키는 식별자

클라이언트 런타임. Message Queue 클라이언트 런타임 참조

클러스터. 함께 작동하면서 메시징 서비스를 제공하는, 상호 연결된 둘 이상의 브로커

트랜잭션. 완료하거나 완전히 롤백해야 하는 작업 기본 단위

JMS (Java Message Service). Java 클라이언트가 메시지 서비스 기능에 액세스하는 방식을 정의하는 인터페이스 및 의미의 표준 집합. 이 인터페이스는 Java 프로그램이 메시지를 작성, 발송, 수신 및 조회하는 표준 방식을 제공합니다.

JMS 공급자. 메시징 시스템을 위한 JMS 인터페이스를 구현하고 완벽한 제품이 필요로 하는 관리 및 제어 기능을 추가한 제품

Message Queue 클라이언트 런타임. JMS 클라이언트에게 Message Queue 메시지 서버와의 인터페이스를 제공하는 소프트웨어. 클라이언트 런타임은 클라이언트가 대상에게 메시지를 보내고 대상으로부터 메시지를 받는 데 필요한 모든 작업을 지원합니다.

Message Queue 메시지 서버. JMS 클라이언트와의 연결, 메시지 라우팅 및 전달, 지속성, 보안 및 로깅을 비롯하여 Message Queue 메시징 시스템을 위해 전달 서비스를 제공하는 소프트웨어. 메시지 서버는 JMS 클라이언트가 보내는 메시지를 받고 또한 사용자 클라이언트에게 그 메시지를 전달하는 물리적 대상을 관리합니다.

가

가입

영구, 영구 가입 참조
정보 44

객체 저장소

위치 293, 294, 296
정보 184
파일 시스템 저장소 185
파일 시스템 저장소 속성 186
LDAP 서버 184
LDAP 서버 속성 184

공급자 독립성

관리 객체 90
정보 43

관리

대상 168
브로커 157

관리 객체

객체 저장소, 객체 저장소 참조
공급자 독립성 90
나열 198
대기열, 대기열 참조
대상, 대상 관리 객체 참조
삭제 198
속성 187
업데이트 200
연결 팩토리, 연결 팩토리 관리 객체 참조
유형 40, 89, 186
정보 40, 89
조회 이름 191

쿼리 199

필요한 정보 192

주제, 주제 참조

XA 연결 팩토리, 연결 팩토리 관리 객체 참조

관리 도구

관리 콘솔 97
명령줄 유틸리티 98
정보 97

관리 작업

개발 환경 93
작업 환경 94

구성 변경 기록 85

구성 요소

EJB 40
MDB 41

구성 파일

기본값 128
설치 128
위치 293, 294, 295
인스턴스 128, 141, 160, 293, 294, 295
템플릿 293, 294, 295
템플릿 위치 293, 294, 295
편집 130

권한

계산 215
내장 데이터베이스 299
데이터 저장소 65
사용자 저장소 204
액세스 제어 등록 정보 파일 67, 213
키 저장소 320

다

- admin 서비스 68
- Message Queue 작업 67
- passfile 225
- 권한 부여
 - 관리 212
 - 사용자 그룹 68
 - 정보 67
 - 액세스 제어 파일 참조
- 기본 제공 지속성 64

다

- 다시 시작
 - 대상 175
 - 브로커 158, 160, 161
 - 연결 서비스 163, 166
- 대기열 77
 - 관리 객체 추가 197
 - 로드 균형 조정 전달, 로드 균형 조정 대기열 전달 참조
 - 속성 171
 - 자동 작성 79, 130
- 대기열 대상, 대기열 참조
- 대기열 로드 균형 조정 전달
 - 속성 171
- 대상
 - 관리 168
 - 나열 169, 173
 - 다시 시작 169, 175
 - 대기열, 대기열 참조
 - 만들기 170
 - 메시지 전달 일괄 처리 81, 172
 - 메시지 제거 169, 176
 - 메트릭, 대상 메트릭 참조
 - 물리적 76
 - 설명 52
 - 속성 171
 - 속성 값 173
 - 속성 값 표시 173
 - 속성 업데이트 170
 - 액세스 제어 216
 - 완전 삭제 168, 170, 176
 - 유형 76, 169
 - 일시 중지 169, 175
 - 임시 81, 173
 - 자동 작성 78, 217
 - 정보 173
 - 정보 얻기 169
 - 제한 동작 61, 170, 171
 - 클러스터의 제한된 범위 80, 172
 - 파일 기반 데이터 저장소 압축 168
 - 주제, 주제 참조
- 대상 관리 객체
 - 설명 39
 - 속성 189
 - 정보 91
- 대상 메트릭
 - 메시지 기반 모니터링 사용 253
 - 메트릭 개수 261
 - imqcmd 메트릭 사용 169
 - imqcmd metrics 사용 247, 249
 - imqcmd query 사용 251
- 대상 자동 작성
 - 등록 정보 79
 - 정보 78
- 데이터 저장소
 - 성능 영향 244
 - 위치 293, 294, 295
 - 재설정 139
 - 정보 63
 - 플랫 파일 64
 - JDBC 액세스 가능 65
- 데이터, 메시지 대기열, 위치 293
- 도구, 관리, 관리 도구 참조
- 도메인 44
- 등록 정보
 - 관리 객체, 관리 객체 참조, 속성 대상, 대상 참조, 속성 로거 74
 - 메모리 관리 62, 170
 - 메시지 라우터 62

- 보안 69
- 브로커 모니터링 서비스 74
- 브로커 인스턴스 구성 130
- 브로커, 업데이트 160
- 연결 서비스 57
- 자동 작성 79
- 지속성 66
- 클러스터 구성 140
- 키 저장소 220
- httpjms 연결 서비스 311
- httpsjms 연결 서비스 323
- JDBC 관련 300
- 디렉토리 변수
 - IMQ_HOME 26
 - IMQ_JAVAHOME 27
 - IMQ_VARHOME 27

라

- 라우팅, 메시지 라우터 참조
- 로거
 - 구성 변경 148
 - 로그 메시지 리디렉션 150
 - 롤오버 기준 150
 - 메시지 형식 148
 - 메트릭 정보 74
 - 범주 72
 - 브로커 구성 요소 54
 - 수준 72, 74, 138
 - 정보 71
 - 출력 채널 71, 149, 251
 - 콘솔에 쓰기 75, 139
- 로그 파일
 - 기본 위치 72, 293, 294, 295
 - 롤오버 기준 75
- 로깅, 로거 참조
- 로드 균형 조정 대기열 전달
 - 성능 조정 288
 - 정보 77
- 로드 균형 조정된 대기열 전달

- 속성 80

마

- 마스터 브로커 85, 86
- 메모리 관리
 - 대상 속성 사용 170
 - 브로커용 61
 - 성능 조정 287
- 메시지
 - 구조 38
 - 대기 시간 228
 - 대상 제한 171
 - 대상에서 제거 169
 - 라우팅 및 전달 59
 - 로드 균형 조정 대기열 전달 77
 - 만료 재생 이용 62
 - 메트릭 73
 - 메트릭 메시지, 메트릭 메시지 참조
 - 본문 유형, 성능 239
 - 브로커 제한 62, 134, 160
 - 사용 88
 - 생성 87
 - 설명 38
 - 수신기 40, 88
 - 순서 49
 - 안정적인 전달 46
 - 우선 순위 49
 - 재전송 60
 - 전달 게시/가입 44
 - 전달 모델 37, 44
 - 전달 모드, 전달 모드 참조
 - 제어 59
 - 지속성 46, 61, 63
 - 지점간 전달 44
 - 처리량 성능 228
 - 크기, 성능 238
 - 필터링, 선택기 참조
 - 확인 60
 - 흐름 제어, 메시지 흐름 제어 참조
 - SOAP 32

바

메시지 라우터

등록 정보 62

브로커 구성 요소 53

정보 58

메시지 사용자, 사용자 참조

메시지 생성자, 생성자 참조

메시지 서버

구조 243

멀티 브로커, 브로커 클러스터 참조 82

정보 52

메시지 서비스

성능에 영향을 미치는 요소 240

정보 36

메시지 수신기, 수신기 참조

메시지 전달 모델 37, 44

메시지 흐름 제어

브로커 61, 170

성능 영향 245

성능 조정 289

제한 289

측정 289

메시징 시스템

구조 36

메시지 서비스 36

Message Queue 구조 52

메트릭

데이터, 메트릭 데이터 참조

메시지, 메트릭 메시지 참조

모니터링 도구, 메트릭 모니터링 도구 참조

정보 71

주제 대상 73, 253

메트릭 데이터

대상, 대상 메트릭 참조

메시지 기반 모니터링 API 사용 253

브로커 로그 파일 사용 251

브로커, 브로커 메트릭 참조

설명 목록 257

연결 서비스, 연결 서비스 메트릭 참조

imqcmd metrics 사용 248

메트릭 메시지

내용 73

유형 73, 253

정보 73, 253

메트릭 모니터링 도구

메시지 기반 모니터링 API 252

비교 255

Message Queue 로그 파일 251

Message Queue 명령 유틸리티(imqcmd) 246

명령 옵션 100

명령 파일 193

명령줄 구문 99

명령줄 유틸리티

공통 옵션 100

기본 구문 99

정보 98

imqbrokerd, imqbrokerd 명령 참조

imqcmd, imqcmd 명령 참조

imqdbmgr, imqdbmgr 명령 참조

imqkeytool, imqkeytool 명령 참조

imqobjmgr, imqobjmgr 명령 참조

imqsvcadmin, imqsvcadmin 명령 참조

imqusermgr, imqusermgr 명령 참조

모니터링, 성능 모니터링 참조

문제 해결 264

바

방화벽 56, 307

벤치마크, 성능 229

병목 현상, 성능 232

보안

객체 저장소 184

관리자, 보안 관리자 참조

권한 부여, 권한 부여 참조

암호화, 암호화 참조

인증, 인증 참조

보안 관리자

등록 정보 69

브로커 구성 요소 54

정보 66

분산 트랜잭션

정보 47

XA 자원 관리자 47, 180

XA 연결 팩토리 참조

브로커

관리 157

구성 요소 및 기능 53

구성 파일, 구성 파일 참조

다시 시작 63, 158, 161

대상 등록 정보 자동 작성 79

등록 정보 160

등록 정보 업데이트 160

등록 정보 표시 159

로깅, 로거 참조

마스터 브로커 85

메모리 관리 61, 170, 244

메시지 경로 지정, 메시지 라우터 참조

메시지 용량 62, 134, 160

메시지 흐름 제어, 메시지 흐름 제어 참조

메트릭, 브로커 메트릭 참조

모니터링, 브로커 모니터링 서비스 참조

보안 관리자, 보안 관리자 참조

상태 제어 160

상호 연결, 브로커 클러스터 참조

시작 135

액세스 제어, 권한 부여 참조

여러 브로커의 클러스터, 브로커 클러스터 참조

연결 156

연결 서비스 나열 164

연결 서비스, 연결 서비스 참조

오류 복구 63

인스턴스 구성 등록 정보 130

인스턴스 이름 138

일시 중지 158, 161

정보 52

제한 동작 61, 244

종료 161

지속성 관리자, 지속성 관리자 참조

쿼리 159

클러스터, 브로커 클러스터 참조

함께 연결 142

확인(Ack) 59, 187

HTTP 지원 309

httpjms 연결 서비스 등록 정보 311

HTTPS, 지원 319

httpsjms 연결 서비스 등록 정보 323

JDBC 지원, JDBC 지원 참조

SSL 기반 서비스 시작 222

Windows 서비스, 실행 333

브로커 다시 시작 158, 161

브로커 메트릭

로거 등록 정보 74, 251

메시지 기반 모니터링 사용 253

메트릭 개수 258

메트릭 메시지 73

보고 간격, 로거 138

브로커 로그 파일 사용 252

imqcmd 사용 162, 249, 250

브로커 모니터링 서비스

등록 정보 74

정보 70

브로커 인스턴스, 브로커 참조

브로커 종료 158, 161

브로커 클러스터

개발 전용 환경 85

구성 등록 정보 86, 140

구성 변경 기록 85

구조 82, 243

등록 정보 설정 141

마스터 브로커 85, 86

브로커 다시 시작 144

브로커 연결 142

브로커 추가 143

브로커간 연결 보안 143

사용 이유 82, 243

성능 영향 244

정보 전파 84

지정할 옵션 137

클러스터 구성 파일 86, 141

클릭 동기화 140

비밀번호

이름 지정 규약 206

인코딩 69

JDBC 225

사

LDAP 225
passfile, passfile [참조](#)
SSL 키 저장소 138, 221, 225
비밀번호 파일, passfile [참조](#)
비밀번호, 기본값 188

사

사용권
시작 옵션 138
엔터프라이즈판 시험 사용권으로 시작 136
Message Queue 판 33
사용자 39
사용자 그룹
기본값 68
사전 정의됨 205
정보 67
할당 삭제 206
사용자 아이디
기본값 203
속성 188
형식 206
사용자 저장소
관리 207
등록 정보 69
사용자 그룹 206
사용자 상태 206
위치 293, 295, 296
정보 67
채우기 207
플랫 파일 202
플랫폼 의존성 204
LDAP 서버 210
생성자
대상 제한 80, 171
정보 39
서블릿, 터널, HTTP/HTTPS 터널 서블릿 [참조](#)
서비스 유형
ADMIN 54
NORMAL 54

선택기
메시지 등록 정보 38
성능 영향 238
정보 49
성능
기본 패턴 230
모니터링, 성능 모니터링 [참조](#)
문제 해결 264
벤치마크 229
병목 현상 232
안정성 균형 49, 233
영향을 미치는 요소, 성능 영향 요소 [참조](#)
정보 227
조정, 성능 조정 [참조](#)
최적화, 성능 조정 [참조](#)
측정 228
표시기 228
성능 모니터링
도구, 메트릭 모니터링 도구 [참조](#) 245
메트릭 데이터, 메트릭 데이터 [참조](#)
성능 영향 요소
데이터 저장소 244
메시지 본문 유형 239
메시지 서버 구조 244
메시지 크기 238
메시지 흐름 제어 245
브로커 제한 동작 244
선택기 238
연결 241
영구 가입 237
운영 체제 241
전달 모드 234
전송 프로토콜 242
트랜잭션 235
하드웨어 240
확인 모드 236
JVM 241
성능 조정
브로커 조정 287
시스템 조정 282
클라이언트 런타임 조정 289
프로세스 개요 227

- 세션
 - 설명 39
 - 트랜잭션 46
 - 확인 옵션 46
- 속성
 - 관리 객체 187
 - 대상 171
- 수신기 40, 41
- 스레드 풀 관리자
 - 공유 스레드 56
 - 전용 스레드 56
 - 정보 56
- 시스템 등록 정보, 설정 91
- 시작
 - 브로커 135
 - 클러스터의 브로커 144
 - SSL 기반 연결 서비스 222

아

- 안정적인 전달 46
 - 성능 균형 49, 233
- 암호화
 - 정보 68
 - 키 도구 및 69
 - SSL 기반 서비스 218
- 액세스 규칙 215
- 액세스 제어 파일
 - 버전 213
 - 액세스 규칙 215
 - 용도 212
 - 위치 294, 295, 296
 - 형식 213
- 업데이트
 - 브로커 160
 - 연결 서비스 163, 165, 167
- 엔터프라이즈판 34
- 연결
 - 나열 167
 - 설명 39
- 성능 영향 241
- 영향을 주는 명령 167
- 쿼리 167
- 연결 서비스
 - 다시 시작 163, 166
 - 등록 정보 57, 165
 - 등록 정보 표시 165
 - 메트릭 데이터, 연결 서비스 메트릭 참조
 - 서비스 유형 54
 - 스레드 풀 관리자 56
 - 스레드 할당 165
 - 시작 시 활성화 57
 - 액세스 제어 69
 - 업데이트 163, 165, 167
 - 연결 유형 54
 - 영향을 주는 명령 163
 - 일시 중지 163, 166
 - 정보 54
 - 정적 포트 57
 - 쿼리 163, 167
 - 클러스터 143, 219
 - 포트 매핑, 포트 매핑 참조
 - admin 55, 164
 - HTTP, HTTP 연결 참조
 - httpjms 55, 164
 - HTTPS, HTTPS 연결 참조
 - httpsjms 55, 164
 - jms 54, 164
 - SSL 기반 221
 - ssladmin, ssladmin 연결 서비스 참조
 - ssljms, ssljms 연결 서비스 참조
- 연결 서비스 메트릭
 - 메트릭 개수 260
 - imqcmd 메트릭 사용 166
 - imqcmd metrics 사용 249
 - imqcmd query 사용 251
- 연결 팩토리 관리 객체
 - 무시 91
 - 설명 39
 - 속성 90, 187
 - 정보 90
 - 추가 195

자

- ClientID 및 45
- JNDI 조회 40
- 연결, 브로커에 156
- 영구 가입
 - 관리 179
 - 나열 179
 - 메시지 제거 179
 - 성능 영향 237
 - 완전 삭제 179, 180
 - 정보 44
- ClientID 및 45
- id 154
- 영구 가입자, 영구 가입 참조
- 운영 체제
 - 성능 영향 241
 - Solaris 성능 조정 282
- 응용 프로그램 예 29, 294, 295, 296
- 응용 프로그램, 클라이언트 응용 프로그램 참조
- 이식성, 공급자 독립성 참조
- 인스턴스 구성 파일, 구성 파일 참조
- 인증
 - 관리 202
 - 정보 67
- 인증서 219, 319
- 일시 중지
 - 대상 169, 175
 - 브로커 158, 160, 161
 - 연결 서비스 163, 166
- 임시 대상 81, 173

자

- 자원 어댑터 43
- 자체 서명된 인증서 219, 319
- 재전송 플래그 60
- 전달 게시/가입 44
- 전달 모드
 - 비지속성 46
 - 성능 영향 234

- 지속성 46
- 전달, 안정성 46
- 전송 프로토콜
 - 상대 속도 242
 - 성능 영향 242
 - 성능 조정 283
 - 프로토콜 유형, 프로토콜 유형 참조
- 제거, 대상에서 메시지 176
- 제어 메시지 59
- 제한 동작
 - 대상 61, 170, 171
 - 브로커 61
- 주제
 - 관리 객체 추가 196
 - 물리적 대상 78
 - 속성 171
 - 자동 작성 79, 130
 - 정보 44
- 주제 대상, 주제 참조
- 지속성
 - 기본 제공 64
 - 데이터 저장소, 데이터 저장소 참조
 - 전달 모드, 전달 모드 참조
 - 지속성 관리자, 지속성 관리자 참조
 - 플러그 인, 플러그 인 지속성 참조
 - JDBC, JDBC 지속성 참조
- 지속성 관리자
 - 데이터 저장소, 데이터 저장소 참조
 - 등록 정보 66
 - 브로커 구성 요소 54
 - 정보 63
 - 플러그 인 지속성 297
 - JDBC 데이터 저장소 299
- 지속성 메시지 46
- 지점간 전달 44

카

- 컨테이너
 - EJB 42

MDB 42

쿼리
 브로커 159
 연결 서비스 163, 165, 167

클라이언트
 런타임, 클라이언트 런타임 참조
 식별자(ClientID) 45
 응용 프로그램, 클라이언트 응용 프로그램 참조
 프로그래밍 모델 38

클라이언트 런타임
 구성 245
 메시지 흐름 조정 289
 정보 87

클라이언트 응용 프로그램
 공급자 독립성 43
 성능에 영향을 미치는 요소 232
 시스템 등록 정보 91
 예 29, 294, 295, 296

클러스터 구성 파일 86

클러스터 연결 서비스
 설정, 보안 143, 219
 포트 번호 141

클러스터, 브로커 클러스터 참조

키 도구 69

키 쌍
 다시 생성 221
 생성 220

키 저장소
 등록 정보 220
 파일 220, 320

타

터널 서블릿, HTTP/HTTPS 터널 서블릿 참조

트랜잭션
 관리 180
 롤백 180
 분산, 분산 트랜잭션 참조
 성능 영향 235
 완결 180

정보 46, 180
 확인 및 60

파

파일 설명자 제한 338

판, 제품
 엔터프라이즈 34
 정보 33
 플랫폼 33

페이로버 82

포트 매핑
 정보 55
 포트 지정 57, 138

포트, 동적 할당 56

프로그래밍 도메인 44

프로토콜 유형
 HTTP 55, 164
 TCP 54, 164
 TLS 54, 164

프로토콜, 전송 프로토콜 참조

플랫폼판 33

플러그인 지속성
 설정 297
 성능 조정 287
 정보 65

하

하드웨어, 성능 영향 240

확인
 대기 시간 187
 브로커 59
 전달 60
 정보 46, 59
 클라이언트 60, 88
 트랜잭션 및 60

환경 변수, 디렉토리 변수 참조

A

흐름 제어, [메시지 흐름 제어 참조](#)

A

admin 연결 서비스 [55, 164](#)

API 설명서 [29, 294, 295, 296](#)

Application Server 및 Message Queue [42](#)

지원 [307](#)

터널 서블릿, [HTTPS 터널 서블릿 참조](#)

HTTPS 터널 서블릿

배포 [320](#)

정보 [308](#)

httpsjms 연결 서비스

구성 [322](#)

설정 [319](#)

정보 [55, 164](#)

C

Cloudscape [297](#)

H

HTTP

연결 서비스, [httpjms 연결 서비스 참조](#)

전송 드라이버 [307](#)

지원 구조 [307](#)

프록시 [307](#)

HTTP 연결

다중 브로커, [313](#)

요청 간격 [311](#)

지원 [307](#)

터널 서블릿, [HTTPS 터널 서블릿 참조](#)

HTTP 터널 서블릿

배포 [309](#)

정보 [308](#)

httpjms 연결 서비스

구성 [310](#)

설정 [309](#)

정보 [55, 164](#)

HTTPS

연결 서비스, [httpsjms 연결 서비스 참조](#)

지원 구조 [307](#)

HTTPS 연결

다중 브로커, [325](#)

요청 간격 [323](#)

I

imq.accesscontrol.enabled 등록 정보 [69, 130](#)

imq.accesscontrol.file.filename 등록 정보 [70, 130](#)

imq.authentication.basic.user_repository 등록 정보 [69, 130](#)

imq.authentication.client.response.timeout 등록 정보 [69, 130](#)

imq.authentication.type 등록 정보 [69, 130](#)

imq.autocreate.destination.isLocalOnly 등록 정보 [80, 130](#)

imq.autocreate.destination.limitBehavior 등록 정보 [80, 130](#)

imq.autocreate.destination.maxBytesPerMsg 등록 정보 [80, 130](#)

imq.autocreate.destination.maxCount 등록 정보 [79, 130](#)

imq.autocreate.destination.maxNumMsgs 등록 정보 [79](#)

imq.autocreate.destination.maxNumProducers 등록 정보 [80, 130](#)

imq.autocreate.destination.maxTotalMsgBytes 등록 정보 [80, 130](#)

imq.autocreate.queue 등록 정보 [79, 130, 160](#)

imq.autocreate.queue.consumerFlowLimit 등록 정보 [81, 131](#)

imq.autocreate.queue.localDeliveryPreferred 등록 정보 [81, 131](#)

imq.autocreate.queue.maxNumActiveConsumers 등록 정보 80, 131, 160
 imq.autocreate.queue.maxNumBackupConsumers 등록 정보 80, 131, 160
 imq.autocreate.topic 등록 정보 79, 131, 160
 imq.cluster.brokerlist 등록 정보 140
 imq.cluster.masterbroker 등록 정보 140
 imq.cluster.port 등록 정보 141
 imq.cluster.transport 등록 정보 141
 imq.cluster.url 등록 정보 141, 160
 imq.hostname 등록 정보 57, 131
 imq.httpjms.http.connectionTimeout 등록 정보 312
 imq.httpjms.http.pullPeriod 등록 정보 311
 imq.httpjms.http.servletHost 등록 정보 311
 imq.httpjms.http.servletPort 등록 정보 311
 imq.httpsjms.https.connectionTimeout 등록 정보 323
 imq.httpsjms.https.pullPeriod 등록 정보 323
 imq.httpsjms.https.servletHost 등록 정보 323
 imq.httpsjms.https.servletPort 등록 정보 323
 imq.keystore.dirpath 등록 정보 220
 imq.keystore.file.name 등록 정보 220
 imq.keystore.password 등록 정보 221, 225
 imq.log.console.output 등록 정보 75, 131
 imq.log.console.stream 등록 정보 75, 131
 imq.log.file.dirpath 등록 정보 74, 131
 imq.log.file.filename 등록 정보 74
 imq.log.file.name 등록 정보 131
 imq.log.file.output 등록 정보 74, 131
 imq.log.file.rolloverbytes 등록 정보 75, 131, 160
 imq.log.file.rolloversecs 등록 정보 75, 131, 160
 imq.log.level 등록 정보 74, 131, 160
 imq.log.syslog.facility 등록 정보 75, 131
 imq.log.syslog.identity 등록 정보 75, 131
 imq.log.syslog.logconsole 등록 정보 75, 132
 imq.log.syslog.logpid 등록 정보 75, 132
 imq.log.syslog.output 등록 정보 75, 132
 imq.log.timezone 등록 정보 76, 132
 imq.message.expiration.interval 등록 정보 62, 132
 imq.message.max_size 등록 정보 62, 132, 160
 imq.metrics.enabled 등록 정보 74, 132
 imq.metrics.interval 등록 정보 74, 132
 imq.metrics.topic.enabled 등록 정보 76, 132
 imq.metrics.topic.interval 등록 정보 76, 132
 imq.metrics.topic.persist 등록 정보 76, 132
 imq.metrics.topic.timetolive 등록 정보 76, 132
 imq.passfile.dirpath 등록 정보 70, 132
 imq.passfile.enabled 등록 정보 70, 132
 imq.passfile.name 등록 정보 70, 132
 imq.persist.file.destination.message.filepool.limit 등록 정보 66, 132
 imq.persist.file.message.cleanup 등록 정보 66, 132
 imq.persist.file.message.filepool.cleanratio 등록 정보 66, 132
 imq.persist.file.message.max_record_size 등록 정보 66, 132
 imq.persist.file.message.vrfile.max_record_size 등록 정보 64
 imq.persist.file.sync.enabled 등록 정보 66, 132
 imq.persist.jdbc.brokerid 등록 정보 300
 imq.persist.jdbc.closedburl 등록 정보 300
 imq.persist.jdbc.createdburl 등록 정보 300
 imq.persist.jdbc.driver 등록 정보 300
 imq.persist.jdbc.needpassword 등록 정보 301
 imq.persist.jdbc.opendburl 등록 정보 300
 imq.persist.jdbc.password 등록 정보 225, 301
 imq.persist.jdbc.table.IMQCCREC35 등록 정보 301
 imq.persist.jdbc.table.IMQDEST35 등록 정보 301
 imq.persist.jdbc.table.IMQILIST35 등록 정보 302
 imq.persist.jdbc.table.IMQINT3 등록 정보 301
 imq.persist.jdbc.table.IMQMSG35 등록 정보 302
 imq.persist.jdbc.table.IMQPROPS35 등록 정보 302
 imq.persist.jdbc.table.IMQSV35 등록 정보 301
 imq.persist.jdbc.table.IMQTACK35 등록 정보 302
 imq.persist.jdbc.table.IMQTXN35 등록 정보 302
 imq.persist.jdbc.user 등록 정보 300

imq.persist.store 등록 정보 66, 133, 300
 imq.ping.interval 등록 정보 57, 133
 imq.portmapper.backlog 등록 정보 57, 133
 imq.portmapper.hostname 등록 정보 57, 133
 imq.portmapper.port 등록 정보 57, 133, 160
 imq.protocol protocol_type inbufsz 284
 imq.protocol protocol_type nodelay 284
 imq.protocol protocol_type outbufsz 284
 imq.resource_state.count 등록 정보 63, 133
 imq.resource_state.threshold 등록 정보 63, 133
 imq.service.activelist 등록 정보 57, 133
 imq.service_name.accesscontrol.enabled 등록 정보 70, 133
 imq.service_name.accesscontrol.file.filename 등록 정보 70, 133
 imq.service_name.authentication.type 등록 정보 69, 133
 imq.service_name.max_threads 등록 정보 58, 133
 imq.service_name.min_threads 등록 정보 58, 133
 imq.service_name.protocol_type.hostname 등록 정보 58, 133, 141
 imq.service_name.protocol_type.port 등록 정보 57, 134
 imq.service_name.threadpool_model 등록 정보 58, 134
 imq.shared.connectionMonitor_limit 등록 정보 58, 134
 imq.system.max_count 등록 정보 62, 134, 160
 imq.system.max_size 등록 정보 62, 134, 160
 imq.transaction.autorollback 등록 정보 63, 134, 182
 imq.user_repository.ldap.base 등록 정보 210
 imq.user_repository.ldap.gidattr 등록 정보 211
 imq.user_repository.ldap.grpbase 등록 정보 211
 imq.user_repository.ldap.grpfilter 등록 정보 211
 imq.user_repository.ldap.grpsearch 등록 정보 211
 imq.user_repository.ldap.memattr 등록 정보 211
 imq.user_repository.ldap.password 등록 정보 210, 225
 imq.user_repository.ldap.principal 등록 정보 210
 imq.user_repository.ldap.server 등록 정보 210

imq.user_repository.ldap.ssl.enabled 등록 정보 211
 imq.user_repository.ldap.timeout 등록 정보 211
 imq.user_repository.ldap.uidattr 등록 정보 210
 imq.user_repository.ldap.usrfilter 등록 정보 211
 IMQ_HOME 디렉토리 변수 26
 IMQ_JAVAHOME 디렉토리 변수 27
 IMQ_VARHOME 디렉토리 변수 27
 imqAckOnAcknowledge 속성 187
 imqAckOnProduce 속성 187
 imqAckTimeout 속성 187
 imqAddressList 속성 187
 imqAddressListBehavior 속성 187
 imqAddressListIterations 속성 187
 imqbrokerd 명령
 명령 구문 135
 사용 135
 옵션 136
 정보 98
 imqBrokerHostName 속성(Message Queue 3.0) 187
 imqBrokerHostPort 속성(Message Queue 3.0) 187
 imqBrokerServicePort 속성(Message Queue 3.0) 187
 imqcmd 명령
 대상 관리 168
 메트릭 모니터링 246
 명령 구문 152
 브로커에 보안 연결 155, 223
 브로커에 연결 156
 옵션 154
 용도 152
 정보 98
 트랜잭션 관리 180
 하위 명령 152
 imqConfiguredClientID 속성 187
 imqConnectionFlowCount 속성 187
 imqConnectionFlowLimit 속성 187
 imqConnectionFlowLimitEnabled 속성 187
 imqConnectionType 속성(Message Queue 3.0) 187
 imqConnectionURL 속성(Message Queue 3.0) 188
 imqConsumerFlowLimit 속성 188

imqConsumerFlowThreshold 속성 188

imqdbmgr 명령
 명령 구문 303
 옵션 305
 정보 99
 하위 명령 304

imqDefaultPassword 속성 188

imqDefaultUsername 속성 188

imqDestinationDescription 속성 91, 189

imqDestinationName 속성 91, 189

imqDisableSetClientID 속성 188

imqFlowControlLimit 속성 188

imqJMSDeliveryMode 속성 188

imqJMSExpiration 속성 188

imqJMSPriority 속성 188

imqkeytool 명령
 명령 구문 219, 320
 사용 219, 320
 정보 99

imqLoadMaxToServerSession 속성 188

imqobjmgr 명령
 명령 구문 189
 옵션 190
 용도 189
 정보 98
 하위 명령 190

imqOverrideJMSDeliveryMode 속성 188

imqOverrideJMSExpiration 속성 188

imqOverrideJMSHeadersToTemporaryDestinations
 속성 188

imqOverrideJMSPriority 속성 188

imqQueueBrowserMax MessagesPerRetrieve 속
 성 188

imqQueueBrowserRetrieveTimeout 속성 188

imqReconnectAttempts 속성 188

imqReconnectEnabled 속성 188

imqReconnectInterval 속성 188

imqSetJMSXAppID 속성 188

imqSetJMSXConsumerTXID 속성 188

imqSetJMSXProducerTXID 속성 188

imqSetJMSXRcvTimestamp 속성 188

imqSetJMSXUserID 속성 188

imqSSLIsHostTrusted 속성(Message Queue 3.0) 188

imqsvcadm 명령
 명령 구문 334
 옵션 335
 용도 334
 정보 99
 하위 명령 334

imqusermgr 명령
 명령 구문 204
 비밀번호 206
 사용자 아이디 206
 옵션 204
 용도 203
 정보 99
 하위 명령 204

J

J2EE 응용 프로그램

EJB 사양 40

JMS 40

Message-Driven Bean, Message-Driven Bean 참조

Java 가상 머신, JVM 참조

JDBC 지원

드라이버 297, 300

설정 297

정보 65

JDK

경로 지정 137

경로 지정 옵션 154, 191, 335

JMS

메시지 구조 38

사양 29, 31, 38

프로그래밍 모델 38

jms 연결 서비스 54, 164

JNDI

객체 저장소 98, 184

L

- 객체 저장소 속성 184, 192
- 관리 객체 및 40, 43
- 위치(공급자 URL) 184, 186
- 조회 89, 91, 115, 192
- 조회 이름 192, 196
- 초기 컨텍스트 184, 186
- Message Queue 지원 32
- Message-Driven Beans 42

JVM

- 메트릭, [JVM 메트릭 참조](#)
- 성능 영향 241
- 성능 조정 283

JVM 메트릭

- 메시지 기반 모니터링 사용 253
- 메트릭 개수 257
- 브로커 로그 파일 사용 252
- imqcmd metrics 사용 248

L

LDAP 서버

- 객체 저장소 속성 184
- 사용자 저장소 액세스 210
- 인증 페일오버 210

M

MDB, [Message-Driven Bean 참조](#)

Message-Driven Bean

- 배치 설명자 42
- 응용 프로그램 서버 지원 42
- 정보 41
- MDB 컨테이너 42

O

Oracle 297

P

passfile

- 명령줄 옵션 138
- 브로커 구성 등록 정보 70
- 사용 225
- 위치 225, 294, 295, 296

PointBase 297

S

Secure Socket Layer 표준, [SSL 참조](#)

Simple Object Access Protocol, SOAP [참조](#)

SOAP 32

SSL

- 암호화 218
- 연결 서비스, [SSL 기반 연결 서비스 참조](#)
- 정보 68
- HTTP에서 224
- TCP/IP에서 219

SSL 기반 연결 서비스

- 설정 201, 219
- 시작 222

ssladmin 연결 서비스

- 설정 219
- 정보 55, 164

ssljms 연결 서비스

- 설정 219
- 정보 54, 164

syslog 72, 149

T

TCP 54, 164

TLS 54, 164

W

Windows 서비스, 브로커 실행 [333](#)

X

XA 연결 팩토리

[연결 팩토리 관리 객체 참조](#)

XA 연결 팩토리, 정보 [48](#)

XA 자원 관리자, [분산 트랜잭션 참조](#)

