관리자 설명서

Sun ™ ONE Message Queue

버전 3.0.1

817-5019-10
2002년 10월

Copyright ⓒ 2002 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A. 모든 권리는 저작권자의 소유입니다.

Sun Microsystems, Inc.는 이 제품에 포함된 기술 관련 지적 재산권을 소유합니다. 특히 이 지적 재산권에는 http://www.sun.com/patents에 나열된 하나 이상의 미국 특허권이 포함될 수 있으며, 하나 이상의 추가 특허권 또는 미국 및 다른 국가에서 특허 출원 중인 응용 프로그램이 제한 없이 포함될 수 있습니다.

본 제품은 사용권에 따라 사용, 복사, 배포 및 디컴파일이 제한됩니다. Sun과 사용 허여자(있을 경우)의 사전 서면 승인 없이는 본 제품을 일체 복제할 수 없습니다.

글꼴 기술을 비롯한 타사 소프트웨어의 저작권 및 사용권은 Sun 공급업체에 있습니다.

Sun, Sun Microsystems, Sun 로고, Java, Solaris, iPlanet, JDK, Java Naming and Directory Interface 및 Java Coffee Cup 로고는 미국 및 다른 국가에서 Sun Microsystems, Inc.의 상표 또는 등록 상표입니다.

SPARC 상표는 사용 허가를 받았으며 미국 및 다른 국가에서 SPARC International, Inc.의 상표 또는 등록 상표입니다. SPARC 상표를 사용하는 제품은 Sun Microsystems, Inc.에서 개발한 구조에 기반을 두고 있습니다.

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

연방 정부 취득: 상용 소프트웨어 - 정부 사용자는 표준 사용권 조항 및 규정을 준수해야 합니다.

목차

그림 목차 11

표 목차 13

절차 목록 15

머리말 17

대상 17
구성 18
표기 규칙 19
텍스트 표기 규칙 19
디렉토리 변수 규칙 20
기타 설명서 자원 22
MQ 설명서 세트 22
온라인 도움말 22
JavaDoc 23
클라이언트 응용 프로그램의 예 23
JMS (Java Message Service) 사양 23

1장 개요 25
Sun ONE Message Queue 소개 25
제품 판 26
플랫폼판 26
엔터프라이즈판 27
엔터프라이즈 메시징 시스템 27
엔터프라이즈 메시징 시스템의 요구 사항 28
중앙 집중식 메시징과 피어 투 피어(Peer to Peer) 메시징 비교 29
메시징 시스템 개념 30
메시지 30
메시지 서비스 구조 30
메시지 전달 모델 31

JMS 사양 **31**
JMS 메시지 구조 31
JMS 프로그래밍 모델 32
관리 객체 33
JMS/J2EE 프로그래밍: Message-Driven Bean **34**
Message-Driven Bean 35
응용 프로그램 서버 지원 36
JMS 메시징 문제 **36**
JMS 공급자 독립성 37
프로그래밍 도메인 37
클라이언트 식별자 38
안정적인 메시징 39
확인/트랜잭션 39
영구 저장소 41
성능 균형 41
메시지 선택 42
메시지 순서 및 우선 순위 42

2장 MQ 메시징 시스템 43
MQ 메시지 서버 **44**
브로커 44
연결 서비스 46
메시지 라우터 50
지속성 관리자 54
보안 관리자 57
로거 61
물리적 대상 65
대기열 대상 65
주제 대상 66
자동 작성(대 관리 작성) 대상 67
임시 대상 68
다중 브로커 클러스터(엔터프라이즈판) 68
다중 브로커 구조 69
개발 환경에서의 클러스터 사용 71
클러스터 구성 등록 정보 71
MQ 클라이언트 런타임 **73**
메시지 생성 74
메시지 사용 74
MQ 관리 객체 **75**
연결 팩토리 관리 객체 77
대상 관리 객체 78
클라이언트 시작 시 속성 값 무시 78

3장 MQ 관리 81
MQ 관리 작업 81
개발 환경 81
작업 환경 82
설정 작업 82
유지 보수 작업 83
MQ 관리 도구 84
관리 콘솔 84
명령줄 유틸리티 요약 84
명령줄 구문 86
공통 명령줄 옵션 87

4장 관리 콘솔 자습서 89
준비 90
관리 콘솔 시작 90
도움말 확인 92
브로커 작업 93
브로커 시작 94
브로커 추가 95
관리 비밀번호 변경 96
브로커에 연결 97
연결 서비스 보기 98
브로커에 물리적 대상 추가 99
물리적 대상 작업 101
주제 대상에 대한 정보 얻기 102
객체 저장소 작업 103
객체 저장소 추가 103
객체 저장소 등록 정보 확인 106
객체 저장소에 연결 106
연결 팩토리 관리 객체 추가 106
대상 관리 객체 추가 108
관리 객체 등록 정보 110
콘솔 정보 업데이트 111
샘플 응용 프로그램 실행 111

5장 브로커 시작 및 구성 113
구성 파일 114
등록 정보 값 병합 114
등록 정보 이름 지정 구문 115
인스턴스 구성 파일 편집 116
브로커 시작 120

클러스터를 사용한 작업(엔터프라이즈판) 126
클러스터 구성 등록 정보 126
브로커 연결 128
방법 1: 클러스터 구성 파일 없음 128
방법 2: 클러스터 구성 파일 사용 128
클러스터에 브로커 추가 129
클러스터에서 브로커 다시 시작 129
클러스터에서 브로커 제거 130
마스터 브로커의 구성 변경 기록 백업 130
마스터 브로커의 구성 변경 기록 복원 131
로깅 131
기본 로깅 구성 131
로그 메시지 형식 132
로거 구성 변경 132
출력 채널 변경 133
롤오버 기준 변경 134
브로커 성능 메트릭 로깅 135

6장 브로커 및 응용 프로그램 관리 137
명령 유틸리티 138
명령 구문 138
imqcmd 하위 명령 138
imqcmd 옵션 요약 140
imqcmd 사용의 전제 조건 142
예 142
브로커 상태 제어 143
브로커 등록 정보 쿼리 및 업데이트 145
브로커 쿼리 146
브로커 업데이트 147
연결 서비스 관리 148
연결 서비스 나열 150
서비스 등록 정보 쿼리 및 업데이트 150
서비스 일시 중지 및 다시 시작 152
대상 관리 152
대상 만들기 153
대상 관련 정보 얻기 154
대상 업데이트 154
대상 제거 155
대상 완전 삭제 155
영구 가입 관리 155
트랜잭션 관리 157

7장 관리 객체 관리 161
객체 저장소 정보 162
관리 객체 162
객체 관리자 유틸리티(imqobjmgr) 163
명령 구문 163
imqobjmgr 하위 명령 163
imqobjmgr 명령 옵션 요약 164
필요한 정보 165
관리 객체 속성 167
연결 팩토리 관리 객체 167
대상 관리 객체 169
객체 저장소 속성 169
초기 컨텍스트 및 위치 정보 169
보안 정보(LDAP 전용) 170
입력 파일 사용 171
관리 객체 추가 및 삭제 174
연결 팩토리 추가 174
주제 또는 대기열 추가 175
관리 객체 삭제 177
정보 얻기 177
관리 객체 나열 177
단일 객체 관련 정보 178
관리 객체 업데이트 180

8장 보안 관리 181
사용자 인증 182
플랫 파일 사용자 저장소 사용 182
사용자 관리자 유틸리티(imqusermgr) 184
그룹 185
상태 186
사용자 아이디 및 비밀번호 형식 186
사용자 저장소 채우기 및 관리 187
기본 관리자 비밀번호 변경 188
사용자 저장소에 LDAP 서버 사용 189
사용자 권한 부여: 액세스 제어 등록 정보 파일 192
액세스 규칙 구문 193
권한 계산 194
연결 액세스 제어 195
대상 액세스 제어 196
대상 자동 작성 액세스 제어 197

암호화: SSL 서비스를 사용한 작업(엔터프라이즈판) 198
TCP/IP에서 SSL 서비스 설정 198
1단계. 자체 서명된 인증서 생성 198
2단계. 브로커에서 SSL 기반 서비스 활성화 200
3단계. 브로커 시작 201
4단계. SSL기반 클라이언트 구성 및 실행 202
HTTP에서 SSL 서비스 설정 203
Passfile 사용 204

부록 A 플러그 인 지속성 설정 205
소개 205
JDBC로 액세스할 수 있는 데이터 저장소 플러그 인 206
JDBC 관련 브로커 구성 등록 정보 207
데이터베이스 관리자 유틸리티(imqdbmgr) 210
명령 구문 210
imqdbmgr 하위 명령 211
imqdbmgr 명령 옵션 요약 211

부록 B HTTP/HTTPS 지원(엔터프라이즈판) 213
HTTP/HTTPS 지원 구조 213
HTTP 지원 구현 215
1단계. 웹 서버에 HTTP 터널 서브릿 배포 215
Jar 파일로 배포 215
웹 아카이브 파일로 배포 216
2단계. httpjms 연결 서비스 구성 216
3단계. HTTP 연결 구성 218
연결 팩토리 속성 설정 218
단일 서브릿을 사용하여 다중 브로커에 액세스 218
HTTP 프록시 사용 219
예: HTTP 터널 서브릿 배포 219
Jar 파일로 배포 219
WAR 파일로 배포 221
HTTPS 지원 구현 222
1단계. HTTPS 터널 서브릿에 대해 자체 서명된 인증서 생성 223
2단계. 웹 서버에 HTTPS 터널 서브릿 배포 224
Jar 파일로 배포 224
웹 아카이브 파일로 배포 225
3단계. httpsjms 연결 서비스 구성 225

4단계. HTTPS 연결 구성 227
JSSE 구성 227
루트 인증서 가져오기 227
연결 팩토리 속성 설정 228
단일 서브릿을 사용하여 다중 브로커에 액세스 228
HTTP 프록시 사용 229
예: HTTPS 터널 서브릿 배포 229
Jar 파일로 배포 229
WAR 파일로 배포 232

부록 C 브로커를 Windows 서비스로 사용 235
브로커를 Windows 서비스로 실행 **235**
서비스 관리자 유틸리티(imqsvcadmin) **236**
명령 구문 236
imqsvcadmin 하위 명령 236
imqsvcadmin 옵션 요약 237
브로커 서비스 제거 238
브로커 서비스 재구성 238
대체 Java Runtime 사용 238
브로커 서비스 쿼리 238
문제 해결 239

부록 D MQ 데이터의 위치 241

부록 E MQ 인터페이스의 안정성 243

용어집 247

색인 251

그림 목차

그림 1-1 중앙 집중식 메시징과 피어 투 피어(Peer to Peer) 메시징 비교 29
그림 1-2 메시지 서비스 구조 30
그림 1-3 JMS 프로그래밍 객체 32
그림 1-4 MDB와의 메시지 35
그림 2-1 MQ 시스템 구조 43
그림 2-2 브로커 구성 요소 45
그림 2-3 연결 서비스 지원 47
그림 2-4 지속성 관리자 지원 54
그림 2-5 보안 관리자 지원 58
그림 2-6 로깅 체계 62
그림 2-7 다중 브로커(클러스터) 구조 69
그림 2-8 메시징 작업 74
그림 2-9 MQ 클라이언트 런타임으로의 메시지 전달 75
그림 3-1 로컬 및 원격 관리 유틸리티 85
그림 5-1 브로커 구성 파일 115
그림 B-1 HTTP/HTTPS 지원 구조 214

표 목차

표 1 설명서 내용 18
표 2 문서 표기 규칙 19
표 3 MQ 디렉토리 변수 20
표 4 MQ 설명서 세트 22
표 1-1 JMS 프로그래밍 객체 38
표 2-1 주요 브로커 구성 요소 및 기능 45
표 2-2 브로커가 지원하는 연결 서비스 46
표 2-3 연결 서비스 등록 정보 49
표 2-4 메시지 라우터 등록 정보 53
표 2-5 지속성 등록 정보 56
표 2-6 보안 등록 정보 60
표 2-7 로깅 범주 62
표 2-8 로거 등록 정보 63
표 2-9 자동 작성 구성 등록 정보 67
표 2-10 클러스터 구성 등록 정보 72
표 2-11 대상 속성 78
표 3-1 공통 MQ 명령줄 옵션 87
표 5-1 브로커 인스턴스 구성 등록 정보 116
표 5-2 imqbrokerd 옵션 121
표 5-3 클러스터 구성 등록 정보 126
표 5-4 imqbrokerd 로거 옵션 및 해당 등록 정보 133
표 5-5 연결 서비스에 대해 수집한 메트릭 136
표 5-6 각 브로커에 대해 수집한 메트릭 136
표 6-1 imqcmd 하위 명령 138
표 6-2 imqcmd 옵션 140
표 6-3 브로커 제어에 사용되는 imqcmd 하위 명령 144
표 6-4 정보를 얻고 브로커를 업데이트할 때 사용되는 imqcmd 하위 명령 145

표 6-5 브로커 등록 정보 147
표 6-6 연결 서비스 관리에 사용되는 imqcmd 하위 명령 148
표 6-7 브로커에 지원되는 연결 서비스 149
표 6-8 연결 서비스 속성 151
표 6-9 대상 관리에 사용되는 imqcmd 하위 명령 152
표 6-10 대상 속성 153
표 6-11 영구 가입 관리에 사용되는 imqcmd 하위 명령 156
표 6-12 트랜잭션 관리에 사용되는 imqcmd 하위 명령 157
표 7-1 imqobjmgr 하위 명령 163
표 7-2 imqobjmgr 옵션 164
표 7-3 연결 팩토리 속성 167
표 7-4 대상 속성 169
표 7-5 객체 저장소의 보안 속성 170
표 8-1 사용자 저장소 초기 항목 183
표 8-2 imqusermgr 하위 명령 184
표 8-3 imqusermgr 옵션 185
표 8-4 사용자 아이디 및 비밀번호에 유효하지 않은 문자 186
표 8-5 LDAP 관련 등록 정보 189
표 8-6 액세스 규칙의 구문 요소 193
표 8-7 대상 액세스 제어 규칙의 요소 196
표 8-8 키 저장소 등록 정보 200
표 8-9 Passfile의 비밀번호 204
표 A-1 JDBC 관련 등록 정보 207
표 A-2 imqdbmgr 하위 명령 211
표 A-3 imqdbmgr 옵션 211
표 B-1 httpjms 연결 서비스 등록 정보 217
표 B-2 HTTP 터널 서브릿 Jar 파일 배포에 사용되는 서브릿 인수 220
표 B-3 httpsjms 연결 서비스 등록 정보 226
표 B-4 HTTPS 터널 서브릿 Jar 파일 배포에 사용되는 서브릿 인수 230
표 C-1 imqsvcadmin 하위 명령 236
표 C-2 imqsvcadmin 옵션 237
표 D-1 MQ 3.0 데이터의 위치 241
표 E-1 MQ 3.0.1 인터페이스의 안정성 243
표 E-2 인터페이스 안정성 분류 체계 245

절차 목록

관리 콘솔을 시작하는 방법 91
관리 콘솔 도움말 정보를 표시하는 방법 92
브로커를 시작하는 방법 94
관리 콘솔에 브로커를 추가하는 방법 95
관리자 비밀번호를 변경하는 방법 96
브로커에 연결하는 방법 97
사용 가능한 연결 서비스를 보는 방법 98
브로커에 대기열 대상을 추가하는 방법 100
물리적 대상의 등록 정보를 보는 방법 101
대상에서 메시지를 제거하는 방법 102
대상을 삭제하는 방법 102
파일 시스템 객체 저장소를 추가하는 방법 103
객체 저장소 등록 정보를 표시하는 방법 106
객체 저장소에 연결하는 방법 106
객체 저장소에 연결 팩토리를 추가하는 방법 107
객체 저장소에 대상을 추가하는 방법 109
대상 객체의 등록 정보를 확인하거나 업데이트하는 방법 110
`SimpleJNDIClient` 응용 프로그램을 실행하는 방법 111
브로커를 클러스터에 연결하는 방법 128
클러스터 구성 파일을 사용하는 경우 브로커를 클러스터에 추가하는 방법 129
문제가 발생한 경우 마스터 브로커를 복원하는 방법 131
브로커의 로거 구성을 변경하는 방법 132
LDAP 서버를 사용하도록 구성 파일을 편집하는 방법 189
ssljms 연결 서비스를 설정하는 방법 198
키 쌍을 다시 생성하는 방법 200
JDBC로 액세스할 수 있는 데이터 저장소를 플러그 인하는 방법 206
HTTP 지원을 구현하는 방법 215

httpjms 연결 서비스를 사용하는 방법 217
터널 서브릿을 추가하는 방법 220
터널 서브릿에 대한 가상 경로(서브릿 URL)를 구성하는 방법 221
웹 서버 시작 시 터널 서브릿을 로드하는 방법 221
서버 액세스 로그를 비활성화하는 방법 221
http 터널 서브릿을 WAR 파일로 배포하는 방법 222
HTTPS 지원을 구현하는 방법 223
httpsjms 연결 서비스를 사용하는 방법 226
JSSE를 구성하는 방법 227
터널 서브릿을 추가하는 방법 230
터널 서브릿에 대한 가상 경로(서브릿 URL)를 구성하는 방법 231
웹 서버 시작 시 터널 서브릿을 로드하는 방법 231
서버 액세스 로그를 비활성화하는 방법 232
HTTPS 터널 서브릿 WAR 파일을 수정하는 방법 232
https 터널 서브릿을 WAR 파일로 배포하는 방법 233
기록된 서비스 오류 이벤트를 보는 방법 239

머리말

이 문서 Sun™ ONE Message Queue (MQ) 3.0.1 *관리자 설명서*는 MQ 메시징 시스템에서 관리 작업을 수행하는 데 필요한 배경 지식 및 정보를 제공합니다.

머리말에는 다음과 같은 절이 있습니다.

- 대상
- 구성
- 표기 규칙
- 기타 설명서 자원

대상

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

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

또한 응용 프로그램 개발자가 MQ 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"	MQ 메시징 시스템 및 용어를 상위 개념 수준에서 요약합니다.
2장, "MQ 메시징 시스템"	MQ 메시징 시스템에 대해, 특히 메시징 서비스를 함께 제공하는 MQ 브로커 및 MQ 클라이언트 런타임에 대해 중점적으로 설명합니다.
3장, "MQ 관리"	MQ 관리 작업과 도구에 대해 설명하고 관리 용도로 사용하는 명령줄 유틸리티 및 그 공통적인 기능에 대해 소개합니다.
4장, "관리 콘솔 자습서"	MQ 메시지 서버의 그래픽 인터페이스인 관리 콘솔을 익힐 수 있는 실습 자습서를 제공합니다.
5장, "브로커 시작 및 구성"	MQ 브로커 및 브로커 클러스터를 시작하고 구성하는 방법에 대해 설명합니다.
6장, "브로커 및 응용 프로그램 관리"	MQ 브로커 관리 관련 작업(응용 프로그램에 영향을 받지 않음) 및 메시징 응용 프로그램 관리 작업의 수행 방법에 대해 설명합니다.
7장, "관리 객체 관리"	MQ 관리 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.
부록 A, "플러그 인 지속성 설정"	지속성 기능을 수행하고자 JDBC 호환 데이터베이스를 사용하도록 MQ를 설정하는 방법에 대해 설명합니다.
부록 B, "HTTP/HTTPS 지원(엔터프라이즈판)"	메시징 클라이언트와 MQ 메시지 서버 사이에 HTTP 연결 서비스를 설정하는 방법에 대해 설명합니다.
부록 C, "브로커를 Windows 서비스로 사용"	(Windows 서비스로 실행되는) 브로커 설치, 쿼리 및 제거를 위해 MQ 서비스 관리 유틸리티(`imqsvcadmin`)를 사용하는 방법에 대해 설명합니다.
부록 D, "MQ 데이터의 위치"	다양한 MQ 데이터 범주의 위치에 대해 설명합니다.

표 1 설명서 내용(*계속*)

장	설명
부록 E, "MQ 인터페이스의 안정성"	다양한 MQ 인터페이스의 안정성에 대해 설명합니다.
"용어집"	MQ 설명서에서 사용하는 용어를 정의합니다.

표기 규칙

이 절에서는 본 설명서에 사용되는 표기 규칙에 관한 정보를 제공합니다.

텍스트 표기 규칙

표 2 문서 표기 규칙

형식	설명
기울임꼴	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 또는 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(MQ, JSP)를 나타냅니다.
키+키	동시 키 입력은 더하기 기호와 함께 표시됩니다. Ctrl+A는 두 키를 동시에 누르라는 의미입니다.
키-키	연속적인 키 입력은 하이픈과 함께 표시됩니다. Esc-S는 Esc를 누르고 이를 놓은 다음 S 키를 누르라는 의미입니다.

디렉토리 변수 규칙

MQ에서는 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 MQ 디렉토리 변수

변수	설명
IMQ_HOME	보통은 다음과 같이 MQ 설명서에서 루트 MQ 설치 디렉토리를 참조할 때 사용됩니다. • Solaris에는 루트 MQ 설치 디렉토리가 없습니다. 따라서 MQ 설명서에서 Solaris의 파일 위치를 참조하는 경우에는 IMQ_HOME이 사용되지 않습니다. • Solaris에서 Sun ONE Application Server 평가판의 루트 MQ 설치 디렉토리는 다음과 같습니다. *root Application Server installation directory*/imq • Windows의 경우, 루트 MQ 설치 디렉토리는 MQ 설치 프로그램에서 설정합니다(기본값은 C:\Program Files\Sun Microsystems\Message Queue 3.0.1). • Windows에서 Sun ONE Application Server의 루트 MQ 설치 디렉토리는 다음과 같습니다. *root Application Server installation directory*/imq • Linux에서 루트 MQ 설치 디렉토리의 기본값은 다음과 같습니다. /opt/imq
IMQ_VARHOME	MQ 임시 또는 동적으로 작성된 구성 및 데이터 파일이 저장되는 /var 디렉토리입니다. 모든 디렉토리를 가리키는 환경 변수로 설정할 수 있습니다. • Solaris에서 IMQ_VARHOME의 기본값은 /var/imq 디렉토리입니다. • Solaris에서 Sun ONE Application Server 평가판의 IMQ_VARHOME 기본값은 IMQ_HOME/var입니다. • Windows에서 IMQ_VARHOME의 기본값은 IMQ_HOME/var입니다. • Windows에서 Sun ONE Application Server의 IMQ_VARHOME 기본값은 IMQ_HOME/var입니다. • Linux에서 IMQ_VARHOME의 기본값은 IMQ_HOME/var입니다.

표 3 MQ 디렉토리 변수(*계속*)

변수	설명
IMQ_JAVAHOME	MQ 실행 파일에 필요한 Java runtime (JRE 1.4)의 위치를 가리키는 환경 변수입니다. • Solaris에서 IMQ_JAVAHOME의 기본값은 /usr/j2se/jre 디렉토리이지만, 사용자는 JRE 1.4가 있는 위치로 값을 설정할 수 있습니다. • Windows에서 IMQ_JAVAHOME의 기본값은 IMQ_HOME/jre 이지만, 사용자는 JRE 1.4가 있는 위치로 값을 설정할 수 있습니다. • Linux에서 IMQ_JAVAHOME의 기본값은 /usr/java/j2sdk1.0/jre 디렉토리이지만, 사용자는 JRE 1.4가 있는 위치로 값을 설정할 수 있습니다.

이 설명서에서 IMQ_HOME, IMQ_VARHOME과 IMQ_JAVAHOME은 플랫폼별 환경 변수 표시나 구문(예: UNIX의 $IMQ_HOME) *없이* 표시됩니다. 모든 경로 이름에는 UNIX 파일 구분자 표시(/)를 사용합니다.

기타 설명서 자원

MQ에는 이 설명서 외에도 추가 설명서 자원이 제공됩니다.

MQ 설명서 세트

MQ 설명서 세트를 구성하는 문서는 표 4에서 일반적으로 사용되는 순서에 따라 나열되어 있습니다.

표 4 MQ 설명서 세트

문서	대상	설명
MQ *설치 설명서*	개발자와 관리자	Solaris, Linux, Windows 플랫폼에서 MQ 소프트웨어를 설치하는 방법을 설명합니다.
릴리스 노트	개발자와 관리자	새로운 기능, 제한, 알려진 버그 및 기술 노트에 관한 설명이 포함되어 있습니다.
MQ *Developer's Guide*	개발자	JMS의 MQ 구현 관련 빠른 시작 자습서와 프로그래밍 정보를 제공합니다.
MQ *관리자 설명서*	관리자, 개발자에게도 권장	MQ 관리 도구를 사용한 관리 작업 수행 시 필요한 배경 및 정보를 제공합니다.

온라인 도움말

MQ 3.0.1은 MQ 메시지 서비스 관리 작업을 수행하는 명령줄 유틸리티를 포함합니다. 이 유틸리티에 대한 온라인 도움말을 보려면 87페이지의 "공통 명령줄 옵션"을 참조하십시오.

또한 MQ 3.0.1에는 그래픽 사용자 인터페이스(GUI) 관리 도구인 관리 콘솔(`imqadmin`)이 포함되어 있습니다. 컨텍스트 관련 온라인 도움말은 관리 콘솔에 포함되어 있습니다.

JavaDoc

JavaDoc 형식의 JMS 및 MQ API 설명서는 다음 위치에 있습니다.

`IMQ_HOME/javadoc/index.html`
(Solaris의 경우 `/usr/share/javadoc/imq/index.html`)

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

클라이언트 응용 프로그램의 예

다음 위치에는 샘플 클라이언트 응용 프로그램 코드를 제공하는 여러 응용 프로그램의 예가 포함되어 있습니다.

`IMQ_HOME/demo` (Solaris의 경우 `/usr/demo/imq`)

해당 디렉토리와 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.

1장

개요

이 장에서는 Sun™ ONE Message Queue (MQ)를 소개하며 관리자와 프로그래머가 관심을 가질 내용으로 구성됩니다.

Sun ONE Message Queue 소개

MQ 제품은 응용 프로그램간 통신 및 안정적인 메시지 전달 문제에 대한 표준 기반 솔루션입니다. MQ는 JMS (Java Message Service) 개방형 표준을 구현하는 엔터프라이즈 메시징 시스템이며 JMS 공급자입니다.

JMS 사양은 분산 환경에서 Java 응용 프로그램으로 메시지를 작성하고, 전송하고, 수신하며 읽을 수 있는 공통의 방식을 제공하는 프로그래밍 인터페이스 집합을 설명합니다(32페이지의 "JMS 프로그래밍 모델" 참조).

Sun ONE Message Queue 소프트웨어를 사용하면 서로 다른 플랫폼과 운영 체제에서 실행되는 프로세스들이 공통의 MQ 메시지 서비스(30페이지의 "메시지 서비스 구조" 참조)에 연결하여 정보를 보내고 받을 수 있습니다. 응용 프로그램 개발자들은 네트워크를 통한 응용 프로그램의 통신 방식과 관련된 하위 수준의 세부 사항보다는 응용 프로그램의 비즈니스 논리에 좀 더 집중할 수 있습니다.

MQ는 JMS 사양의 최소 요구 사항을 능가하는 기능을 제공합니다. 다음은 그러한 기능의 예입니다.

중앙 집중식 관리 MQ 메시지 서비스를 관리하고 대상과 보안 등 응용 프로그램별 메시징 요소를 관리하는 명령줄 및 GUI 도구를 제공합니다.

확장 가능한 메시지 서비스 탠덤으로 작동하는 많은 MQ 메시지 서비스 구성 요소(*브로커*)(다중 브로커 클러스터) 간에 로드 균형을 조정하여 더 많은 수의 JMS 클라이언트(구성 요소 또는 응용 프로그램) 서비스를 제공할 수 있습니다.

조정 가능한 성능 전달 안정성을 더 낮출 수 있는 경우 MQ 메시지 서비스의 성능을 높일 수 있습니다.

다중 전송 TCP와 HTTP 등의 다양한 전송 방식을 통해 그리고 보안(SSL) 연결을 사용하여 상호 통신이 가능한 JMS 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 MQ *Developer's Guide*를 참조하십시오.

JMS 호환성에 대한 정보는 MQ 3.0.1 *릴리스 노트*를 참조하십시오.

제품 판

Sun ONE Message Queue 제품은 플랫폼판과 엔터프라이즈판의 두 버전으로 제공되며, 이 버전들은 다음과 같이 사용권 용량이 서로 다릅니다(MQ를 다른 판으로 업그레이드하는 방법은 MQ *설치 설명서* 지침 참조).

플랫폼판

이 판은 Sun 웹 사이트에서 무료로 다운로드할 수 있으며 최신 Sun ONE Application Server 플랫폼과 함께 제공됩니다. 플랫폼판의 경우 각 MQ 메시지 서비스가 지원하는 JMS 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(다중 브로커 메시지 서비스), HTTP/HTTPS 연결, 보안 연결 서비스, 확장 가능한 연결 기능, 다중 대기열 전달 정책 등의 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(다중 브로커 메시지 서비스, HTTP/HTTPS 연결, 보안 연결 서비스, 확장 가능한 연결 기능, 다중 대기열 전달 정책 등)이 포함됩니다. 하지만 소프트웨어는 사용권에 90일의 제한을 두기 때문에 해당 제품의 엔터프라이즈판에서 사용 가능한 엔터프라이즈 기능 평가에 적합합니다(27페이지의 "엔터프라이즈판" 참조).

주 90일 시험 사용권은 `-license` 명령줄 옵션으로 MQ 메시지 서비스(MQ 브로커 인스턴스)를 시작하고(121페이지의 표 5-2 참조) 사용할 사용권으로 "`try`"를 전달하면 활성화됩니다.

```
imqbrokerd -license try
```

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

엔터프라이즈판

이 판은 생산 환경에서 메시징 응용 프로그램을 배포 및 실행할 때 사용됩니다. 여기에는 다중 브로커 메시지 서비스, HTTP/HTTPS 연결, 보안 연결 서비스, 확장 가능한 연결 기능, 다중 대기열 전달 정책에 대한 지원이 포함됩니다. 엔터프라이즈판은 메시징 응용 프로그램과 구성 요소의 개발, 디버깅, 로드 테스트에도 사용할 수 있습니다. 엔터프라이즈판에는 다중 브로커 메시지 서비스의 브로커 수는 제한하지 않지만 지원되는 CPU의 수를 지정하는 무기한 사용권이 있습니다.

엔터프라이즈 메시징 시스템

엔터프라이즈 메시징 시스템에서 독립 분산 구성 요소나 응용 프로그램은 메시지를 통해 상호 작용할 수 있습니다. 동일한 시스템이나 네트워크에 있거나 또는 인터넷을 통해 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 처리 능력을 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.

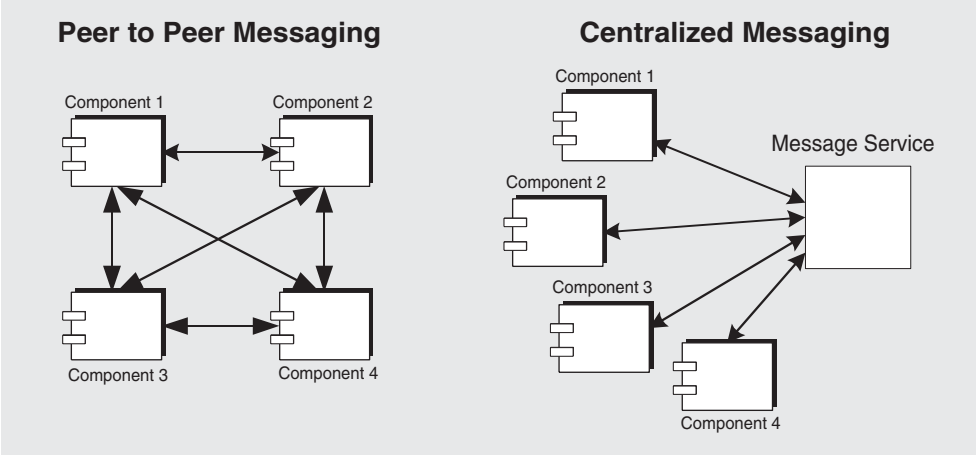
확장성 메시징 시스템은 성능이나 메시지 처리 능력이 실제로 저하되지 않으면서 로드 증가(사용자 수 및 메시지 수 증가)를 수용할 수 있어야 합니다. 업무와 응용 프로그램이 늘어나면 이는 매우 중요한 요구사항이 됩니다.

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

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 관리합니다. 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 경로 지정 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.

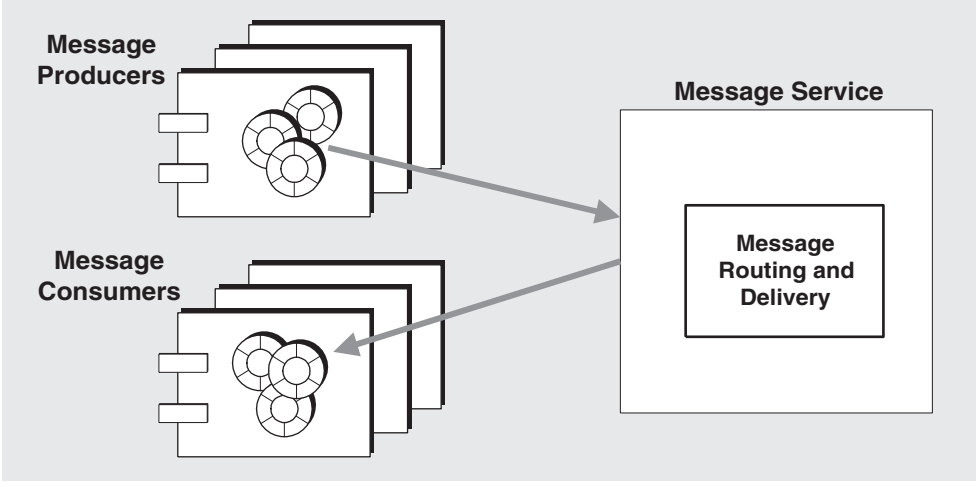
메시지

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 30페이지의 그림 1-2와 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 경로를 지정하고 전달 구성 요소를 사용하는 방식으로 해당 메시지에 대해 인터레스트를 등록한 하나 이상의 메시지 사용자에게 메시지를 전달합니다. 메시지 경로 지정 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) API는 Java 메시징을 위한 공통 프로그래밍 방식을 마련하고자 개발되었습니다. 이 API는 지점간 및 게시/가입 메시지 전달 모델을 모두 지원합니다(37페이지의 "프로그래밍 도메인" 참조).

JMS 사양

JMS는 메시징 작업에 적용되는 메시지 구조, 프로그래밍 모델 그리고 규칙 및 의미 집합을 지정합니다. MQ는 JMS 구현을 제공하기 때문에 JMS 개념은 MQ 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 사양에서 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.

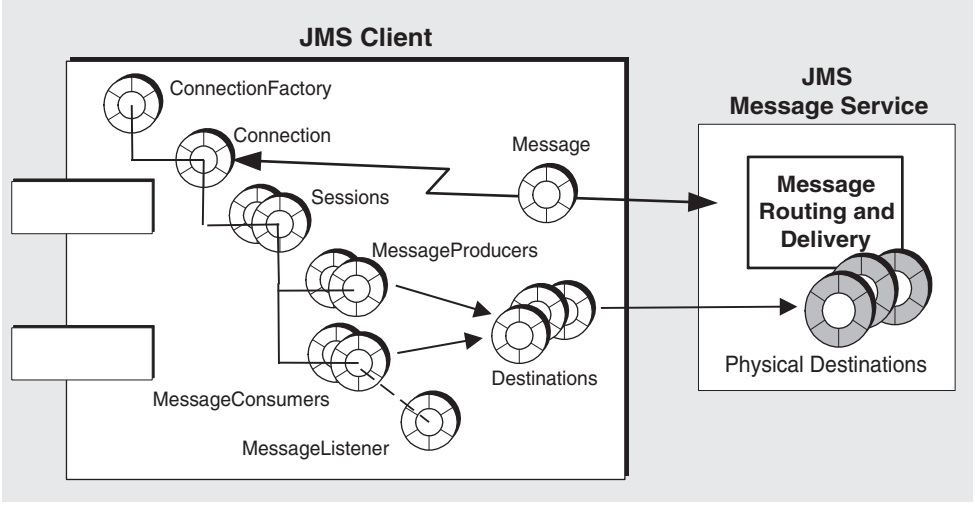
메시지 본문 메시지 본문에는 교환할 실제 데이터가 포함됩니다. JMS는 6가지 본문 유형을 지원합니다.

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 MQ *Developer's Guide* 참조). 32페이지의 그림 1-3은 메시지 전달 프로그램에 사용되는 JMS 객체들입니다.

그림 1-3 JMS 프로그래밍 객체



JMS 프로그래밍 모델에서 JMS 클라이언트는 `ConnectionFactory` 객체를 사용하여 JMS 메시지 서비스와 메시지를 송수신할 연결을 생성합니다. `Connection`은 JMS 클라이언트와 메시지 서비스간의 활성 연결을 의미합니다. 연결되면 통신 자원 할당 및 클라이언트 인증이 이루어집니다. 이는 비교적 중량급 객체이며, 대부분의 클라이언트는 단일 연결을 사용하여 모든 메시징을 수행합니다.

연결은 세션을 생성할 때 사용합니다. `Session`은 메시지 생성 및 사용을 위한 단일 스레드 컨텍스트입니다. 메시지를 보내고 받는 메시지 생성자 및 사용자를 생성할 때 사용하며, 전달할 메시지의 일련 순서를 정의합니다. 세션은 다양한 확인 옵션이나 (분산 트랜잭션 관리자가 관리할 수 있는) 트랜잭션을 통해 안정적인 전달을 지원합니다.

JMS 클라이언트는 `MessageProducer`를 사용하여 API에서 대상 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 JMS 클라이언트는 `MessageConsumer`를 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(MQ *Developer's Guide* 참조). 사용자에게 `MessageListener`를 등록하면 비동기식 사용이 이루어집니다. 세션 스레드가 `MessageListener` 객체의 `onMessage()` 메소드를 호출하면 클라이언트가 해당 메시지를 사용합니다.

관리 객체

32페이지의 "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 클라이언트 코드가 공급자 독립성을 갖게 하려면 JMS 사양에서 공급자별 구현 및 구성 정보가 소위 *관리 객체*로 캡슐화되어야 합니다. 그러면 표준화되고 특정 공급자에 국한되지 않는 방식으로 객체에 액세스할 수 있습니다.

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

JMS는 연결 팩토리와 대상 객체의 2가지 일반적인 관리 객체 유형을 제공합니다. 두 객체 모두 공급자별 정보를 캡슐화하지만 JMS 클라이언트 내부에서 그 용도는 매우 다릅니다. 연결 팩토리는 메시지 서버와의 연결을 생성할 때 사용하며, 대상 객체는 JMS 메시지 서비스가 사용하는 물리적 대상을 식별할 때 사용합니다.

관리 객체에 대한 자세한 내용은 75페이지의 "MQ 관리 객체"를 참조하십시오.

JMS/J2EE 프로그래밍: Message-Driven Bean

32페이지의 "JMS 프로그래밍 모델"에서 소개한 일반적인 JMS 클라이언트 프로그래밍 모델 외에도 J2EE (Java 2 Enterprise Edition) 응용 프로그램 컨텍스트에서 사용하는, 보다 특수화된 JMS 버전이 있습니다. 이 특수화한 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.

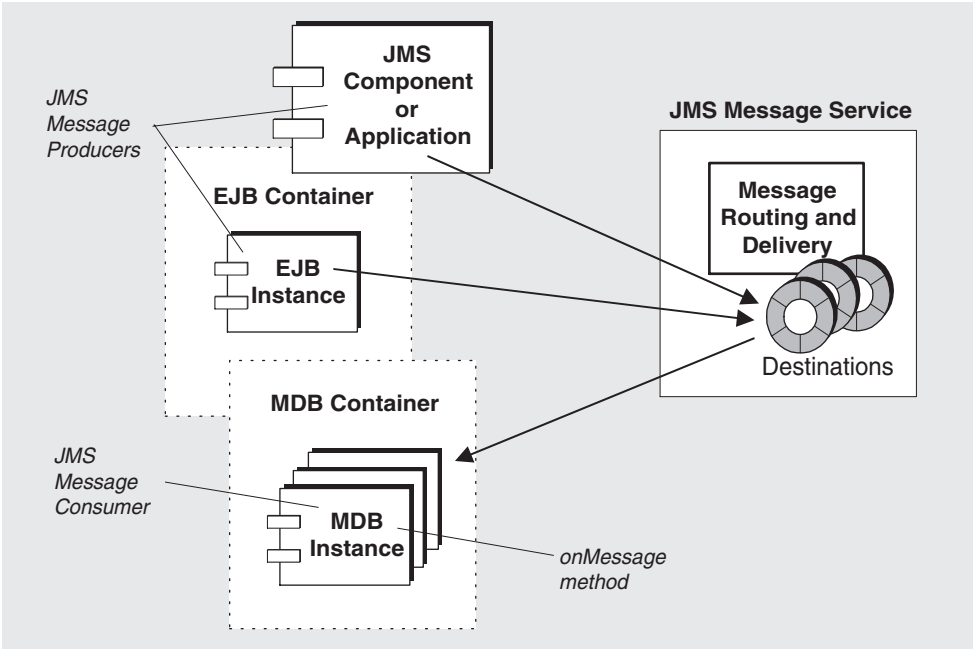
그러나 비동기식 메시징은 많은 엔터프라이즈 응용 프로그램에서 필요합니다. 그러한 응용 프로그램 중 대부분에서는 서버측 구성 요소가 서버 자원을 독점하지 않으면서 상호 통신하고 응답할 수 있어야 합니다. 즉 메시지 생성자와 밀접하게 연결되지 않으면서 메시지 수신 및 사용이 가능한 EJB 구성 요소가 필요합니다. 이 기능은 서버측 구성 요소가 응용 프로그램 이벤트에 응답해야 하는 모든 응용 프로그램에서 필요합니다. 또한 엔터프라이즈 응용 프로그램에서도 로드 증가에 따라 이 기능이 확장되어야 합니다.

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. 35페이지의 그림 1-4에서 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 객체, 즉 연결 팩토리와 대상의 JNDI 조회 이름을 지정합니다. 또한 배치 설명자는 배치 도구가 컨테이너 구성 시 사용 가능한 다른 정보를 포함할 수 있습니다. 이 컨테이너 각각은 단일 MDB로만 이루어진 인스턴스를 지원합니다.

응용 프로그램 서버 지원

J2EE 구조(`http://java.sun.com/j2ee/download.html#platformspec`의 J2EE 플랫폼 사양 참조)에서 EJB 컨테이너는 응용 프로그램 서버가 호스팅합니다. 응용 프로그램 서버는 다양한 컨테이너, 즉 트랜잭션 관리자, 지속성 관리자, 이름 서비스 그리고 메시징 및 MDB의 경우 JMS 공급자가 필요로 하는 자원을 제공합니다.

Sun ONE Application Server에서 메시징 자원은 Sun ONE Message Queue가 제공합니다. 즉 MQ 메시징 시스템(2장, "MQ 메시징 시스템" 참조)이 Sun ONE Application Server에 통합되어 있으며, 이 메시징 시스템은 해당 응용 프로그램 서버 환경에서 실행되는 MDB 및 기타 JMS 메시징 구성 요소로 JMS 메시지를 보내는 데 필요한 지원을 제공합니다.

JMS 메시징 문제

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

JMS 공급자 독립성

JMS는 다른 JMS 공급자에게 이식 가능한 클라이언트 응용 프로그램 개발을 지원하도록 관리 객체(33페이지의 "관리 객체" 참조)의 사용을 지정합니다. 관리 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 객체는 MQ 관리자가 생성 및 구성하는 MQ 시스템 객체입니다. 이 객체는 JNDI 디렉토리 서비스에 위치하며, JMS 클라이언트는 JNDI 조회를 사용하여 이 객체에 액세스합니다.

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

관리 객체에 대한 자세한 내용은 75페이지의 "MQ 관리 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(65페이지의 "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 *가입*한 *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 사용자에 대해 영구 가입이 없는 한 받은 메시지를 보관하지 않습니다.

이 2가지 메시지 전달 모델은 표 1-1에서 확인할 수 있듯이 각기 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 서비스간의 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(37페이지의 "게시/가입(주제 대상)" 참조)과 함께 사용됩니다.

JMS 사양에서는 클라이언트가 API 메소드 호출을 통해 클라이언트 식별자를 설정할 수 있지만, 관리자가 연결 팩토리 관리 객체(33페이지의 "관리 객체" 참조)를 사용하여 설정하는 것이 좋습니다. 그러나 연결 팩토리에 고정된 경우, 각 사용자는 개별 연결 팩토리마다 고유 아이디를 가져야 합니다.

MQ에서는 `ConnectionFactory` 객체에서 구성 가능한 특수 변수 대체 구문을 사용하여 클라이언트 식별자가 ConnectionFactory 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 MQ 기능 사용 방법에 대한 자세한 내용은 MQ *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 모드*를 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. 그러한 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 MQ 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

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 *트랜잭션된* 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 종단간 트랜잭션은 만들 수 없습니다. (즉, 대상으로 메시지를 전달하는 것과 나중에 메시지를 클라이언트로 전달하는 것을 하나의 트랜잭션으로 분류할 수 없습니다.)

분산 트랜잭션

또한 MQ는 *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 MQ 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.

MQ는 XA 연결 팩토리를 통해 분산 트랜잭션 지원을 구현합니다. 이를 통해 XA 연결을 생성하고, XA 연결을 통해 XA 세션을 생성할 수 있습니다(32페이지의 "JMS 프로그래밍 모델" 참조). 또한 분산 트랜잭션을 지원하려면 타사의 JTS나 (JTS를 제공하는) J2EE 호환 응용 프로그램 서버가 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(54페이지의 "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 *성능*을 극대화할 수 있습니다. 한편 지속성 메시지를 생성 및 사용하고 트랜잭션된 세션을 사용할 경우 *안정성*을 극대화할 수 있습니다. 이 둘 간에는 MQ별 연결 및 확인 등록 정보의 사용을 비롯하여 응용 프로그램의 필요사항에 따라 다양한 옵션이 존재합니다(MQ *Developer's Guide* 참조).

메시지 선택

JMS는 메시지 선택기에 설정된 기준을 토대로 메시지 서비스가 메시지 필터링 및 경로 지정을 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(65페이지의 "대기열 대상" 참조) 및 메시지 서비스 가용성 등 시간에 영향을 미치는 다양한 요소에 따라 결정되기 때문입니다.

MQ 메시지 서버가 상호 연결된 여러 브로커를 사용하는 경우(68페이지의 "다중 브로커 클러스터(엔터프라이즈판)" 참조), 클라이언트의 메시지 사용 순서는 더 복잡해지는데, 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 사용자에 대해 우선 순위가 높은 메시지는 우선 순위가 낮은 메시지보다 먼저 처리됩니다.

MQ 메시징 시스템

이 장에서는 그림 2-1에서 보여주는 시스템의 주요 부분에 중점을 두고 Sun™ ONE Message Queue (MQ) 메시징 시스템에 대해 소개합니다. 또한 각 부분이 어떻게 상호 작용하면서 안정적인 메시지 전달을 제공하는지 설명합니다.

그림 2-1 MQ 시스템 구조

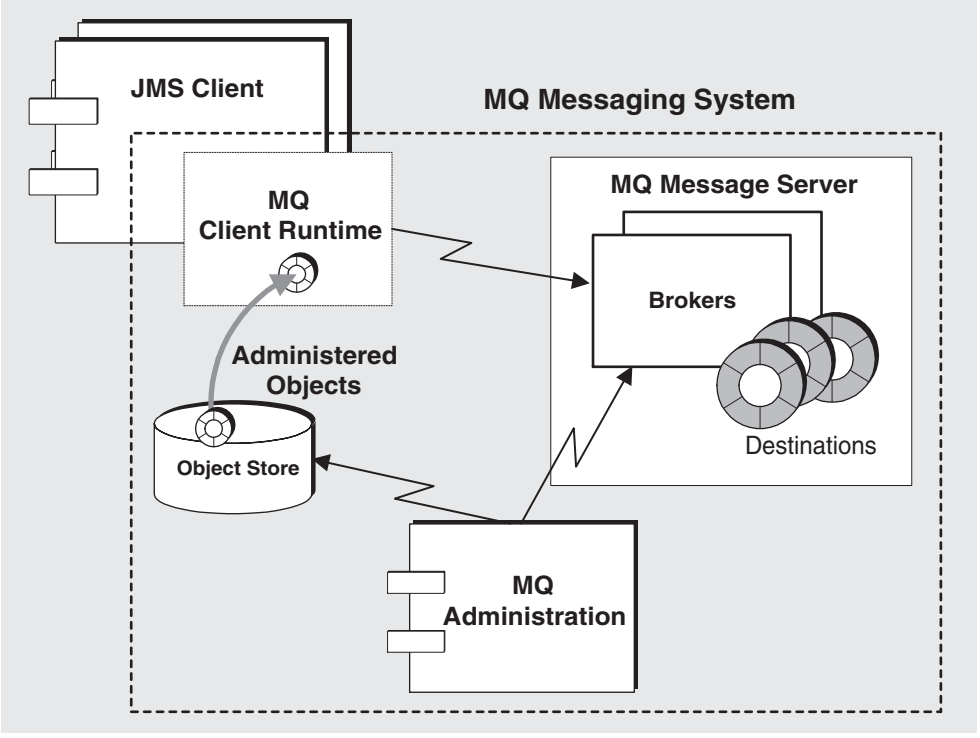


그림 2-1에서 보여주는 MQ 메시징 시스템의 주요 부분은 다음과 같습니다.

- MQ 메시지 서버
- MQ 클라이언트 런타임
- MQ 관리 객체
- MQ 관리

첫 세 항목에 대해서는 다음 절에서 다룹니다. 마지막 항목은 3장, "MQ 관리"에서 소개합니다.

MQ 메시지 서버

이 절에서는 43페이지의 그림 2-1에서 보여주는 MQ 메시지 서버의 다양한 부분을 설명합니다. 여기에는 다음 항목이 포함됩니다.

브로커 MQ 브로커는 MQ 메시징 시스템의 전달 서비스를 제공합니다. 메시지 전달은 연결 서비스, 메시지 경로 지정 및 전달, 지속성, 보안 및 로깅을 처리하는 수많은 지원 구성 요소에 따라 달라집니다(자세한 정보는 44페이지의 "브로커" 참조). 메시지 서버는 하나 이상의 브로커 인스턴스를 사용할 수 있습니다(68페이지의 "다중 브로커 클러스터(엔터프라이즈판)" 참조).

물리적 대상 메시지 전달은 생성자 클라이언트에서 브로커가 관리하는 물리적 대상으로의 전달 그리고 대상에서 하나 이상의 사용자 클라이언트로의 전달의 2단계로 구성되는 과정입니다. 물리적 대상은 브로커의 물리적 메모리 및/또는 영구 저장소에서의 위치를 나타냅니다(자세한 정보는 65페이지의 "물리적 대상" 참조).

브로커

MQ 메시징 시스템의 메시지 전달, 즉 생성자 클라이언트에서 대상까지의 전달 그리고 대상에서 하나 이상의 사용자 클라이언트까지의 전달은 브로커(또는 탠덤으로 작동하는 브로커 인스턴스로 구성된 클러스터)가 수행합니다. 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 구성 요소를 사용합니다. 로드 상태, 응용 프로그램 복잡성 등에 따라 브로커 성능을 최적화하도록 이 내부 구성 요소들을 구성할 수 있습니다. 주요 브로커 구성 요소는 그림 2-2에서 확인할 수 있으며, 표 2-1에서도 간략히 설명합니다.

그림 2-2 브로커 구성 요소

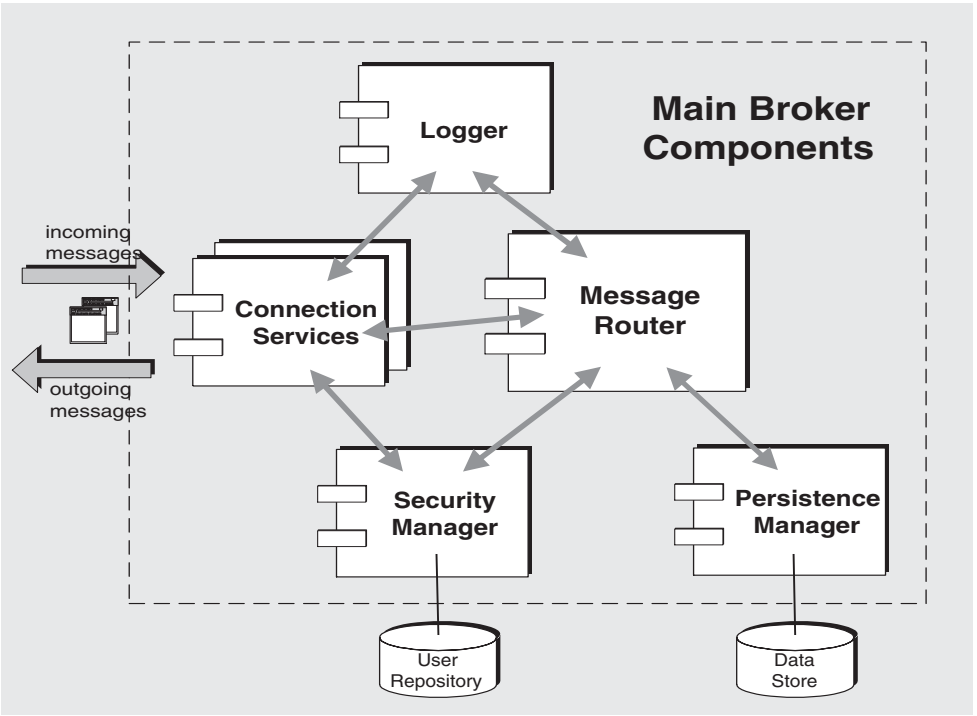


표 2-1 주요 브로커 구성 요소 및 기능

구성 요소	설명/기능
연결 서비스	브로커와 클라이언트 사이의 물리적 연결을 관리하면서 받고 보내는 메시지 전송을 담당합니다.
메시지 라우터	메시지의 경로 지정 및 전달을 관리합니다. 여기에는 JMS 메시지를 비롯하여 MQ 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
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.
로거	로그 파일이나 콘솔에 모니터링 및 진단 정보를 기록하여 관리자가 브로커를 관리할 수 있게 합니다.

다음 절에서는 다양한 브로커 구성 요소가 수행하는 기능 및 그 동작에 영향을 주도록 구성 가능한 등록 정보를 더 자세히 살펴 봅니다.

연결 서비스

MQ 브로커는 JMS 클라이언트와 MQ 관리 클라이언트와의 통신을 모두 지원합니다(84 페이지의 "MQ 관리 도구" 참조). 각 서비스는 서비스 유형 및 프로토콜 유형을 통해 지정됩니다.

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

프로토콜 유형 서비스를 지원하는 기본 전송 프로토콜 계층을 지정합니다.

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

표 2-2 브로커가 지원하는 연결 서비스

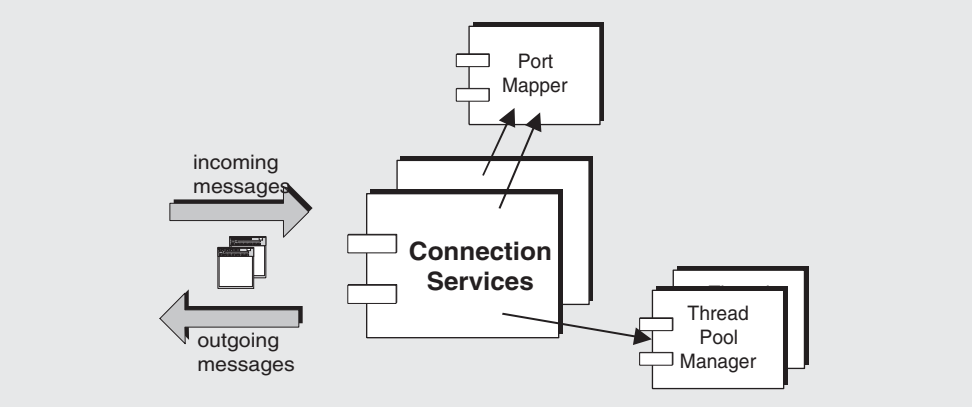
서비스 이름	서비스 유형	프로토콜 유형
jms	NORMAL (JMS 메시지 전달)	tcp
ssljms (엔터프라이즈판)	NORMAL (JMS 메시지 전달)	tls (SSL 기반 보안)

표 2-2 브로커가 지원하는 연결 서비스*(계속)*

서비스 이름	서비스 유형	프로토콜 유형
httpjms (엔터프라이즈판)	NORMAL (JMS 메시지 전달)	http
httpsjms (엔터프라이즈판)	NORMAL (JMS 메시지 전달)	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 서비스는 그림 2-3과 같이 스레드 풀 관리자가 있으며 공통 포트 매퍼 서비스에 해당 서비스를 등록합니다.

그림 2-3 연결 서비스 지원



각 연결 서비스는 특정 포트에서 사용 가능하며, 브로커의 호스트 이름과 포트 번호로 지정됩니다. 포트는 정적으로 또는 동적으로 할당할 수 있습니다. MQ는 동적으로 할당된 포트를 여러 연결 서비스와 매핑하는 *포트 매퍼*를 제공합니다. 포트 매퍼 자체는 표준 포트 번호 7676에 위치합니다. 클라이언트가 브로커와 연결을 설정할 때 이 클라이언트는 먼저 포트 매퍼에 접속하여 원하는 연결 서비스의 포트 번호를 요청합니다.

또한 jms, ssljms, admin 및 ssladmin 연결 서비스에 대해 이 연결 서비스를 구성할 때 *정적* 포트 번호를 지정할 수 있지만, 권장되는 방법은 아닙니다. httpjms 및 httpsjms 서비스는 각각 부록 B, "HTTP/HTTPS 지원(엔터프라이즈판)"에서 217페이지의 표 B-1과 226페이지의 표 B-3에 설명되어 있는 등록 정보를 사용하여 구성합니다.

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

공유 모델(엔터프라이즈판) 공유 스레드 모델에서는 메시지를 보내거나 받을 때에만 스레드에 연결이 지정됩니다. 연결 시 스레드를 공유하는 이 모델에서는 연결 서비스(브로커)가 지원하는 연결의 수는 늘어나지만, 어느 정도의 성능 오버헤드가 발생합니다. 스레드 풀 관리자는 연결 활동을 모니터링하고 필요에 따라 스레드에 연결을 지정하는 분산자 스레드 집합을 사용합니다. 각 분산자 스레드가 모니터링하는 연결 수를 제한하여 성능을 높일 수 있습니다.

각 연결 서비스는 특정 인증 및 권한 부여(액세스 제어) 기능을 지원합니다(57페이지의 "보안 관리자" 참조).

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
`imq.service.activelist`	브로커 시작 시 활성화되는 쉼표로 구분된 이름별 연결 서비스 목록입니다. 지원되는 서비스: `jms`, `ssljms`, `httpjms`, `httpsjms`, `admin`, `ssladmin`. 기본값: `jms`, `admin`
`imq.`*service_name*`.min_threads`	이 스레드 수에 도달하면 명명된 연결 서비스가 사용할 수 있도록 스레드 풀에서 스레드가 유지 관리됩니다. 기본값: 연결 서비스에 따라 다릅니다(116페이지의 표 5-1 참조).
`imq.`*service_name*`.max_threads`	이 스레드 수를 초과하면 명명된 연결 서비스가 사용할 수 있는 신규 스레드가 더 이상 스레드 풀에 추가되지 않습니다. 이 수는 0보다 크고 `min_threads` 값보다 커야 합니다. 기본값: 연결 서비스에 따라 다릅니다(116페이지의 표 5-1 참조).
`imq.`*service_name*`.threadpool_model`	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, `jms` 및 `admin` 연결 서비스에 대해서만 구현됩니다. 기본값: 연결 서비스에 따라 다릅니다(116페이지의 표 5-1 참조).
`imq.shared.connectionMonitor_limit`	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 0은 제한이 없음을 의미합니다. 기본값: 운영 체제에 따라 다릅니다(116페이지의 표 5-1 참조).
`imq.portmapper.port`	브로커의 기본 포트, 즉 포트 매퍼가 위치한 포트입니다. 한 호스트에서 둘 이상의 브로커 인스턴스를 실행하는 경우 각 인스턴스는 고유한 포트 매퍼 포트를 지정 받아야 합니다. 기본값: 7676

표 2-3 연결 서비스 등록 정보(*계속*)

등록 정보 이름	설명
`imq.`*service_name*`.` *protocol_type*[1] `.port`	jms, ssljms, admin 및 ssladmin 서비스에 한해 명명된 연결 서비스의 포트 번호를 지정합니다. 기본값: `0` (포트 매퍼가 동적으로 포트를 할당) httpjms 및 httpsjms 연결 서비스를 구성하려면 부록 B, "HTTP/HTTPS 지원(엔터프라이즈판)"을 참조하십시오.
`imq.`*service_name*`.` *protocol_type*[1] `.hostname`	jms, ssljms, admin 및 ssladmin 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: `null` (모든 호스트)

1 protocol_type은 표 2-2에 나와 있습니다.

메시지 라우터

지원되는 연결 서비스를 사용하여 클라이언트와 브로커 사이에 연결이 설정되면 경로 지정 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(39페이지의 "안정적인 메시징" 참조) 브로커의 영구 데이터 저장소에서 해당 메시지를 제거합니다.

안정적인 전달: 확인 및 트랜잭션

지금까지 설명한 전달 메커니즘은 *안정적인* 전달(39페이지의 "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 MQ는 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(실제로는 생성자가 JMS 메시지를 지속성 메시지로 지정한 경우에만 MQ가 이 기능을 수행함). 생성자 클라이언트는 브로커 확인을 통해 대상에게 확실히 전달되도록 합니다(74페이지의 "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(74페이지의 "메시지 사용" 참조). 이는 어떤 이유로 브로커가 클라이언트 확인을 받지 못한 경우에 발생할 수 있습니다.

브로커가 클라이언트 확인을 받지 않고 JMS 메시지를 두번째 전달하는 경우, 메시지에 재전송 플래그가 표시됩니다. 일반적으로 브로커가 클라이언트 확인을 받기 전에 클라이언트 연결이 끊기고 이어서 새 연결이 설정되면 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 지원합니다(40페이지의 "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 브로커가 실패할 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(54페이지의 "지속성 관리자" 참조) 브로커 실패 시 메시지가 복구될 수 있게 합니다.

시스템 자원 관리

브로커의 성능은 사용 가능한 시스템 자원 그리고 메모리와 같은 자원이 얼마나 효율적으로 활용되는가에 따라 달라집니다. 예를 들어 메시지 라우터는 시스템 상의 메모리를 살펴보는 메모리 관리 메커니즘을 가지고 있습니다. 메모리 자원이 부족해지면 메모리를 확보하고 받는 메시지 속도를 낮추는 메커니즘이 활성화됩니다.

메모리 관리 메커니즘은 메모리 자원 상태, 즉 초록(사용 가능 메모리 충분), 노랑(브로커 메모리 감소 중), 주황(브로커 메모리 부족) 및 빨강(브로커 메모리 부족)에 따라 달라집니다. 메모리 자원 상태가 초록에서 노랑 및 주황을 거쳐 빨강으로 바뀌면 브로커는 메모리를 확보하고 메시지 생성자를 계속해서 억제한 뒤 결국 브로커로 향하는 메시지 흐름을 중지시킵니다.

메모리의 메시지 총 수 및 총 크기를 제한하고 메모리 자원이 새로운 상태로 바뀌는 사용 임계값을 조정하는 등록 정보를 사용하여 브로커의 메모리 관리 기능을 구성할 수 있습니다.

이 등록 정보는 표 2-4에서 자세히 설명합니다(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`	(스왑 때문에) 메모리와 디스크 모두에 위치한 메시지의 최대 수를 지정합니다. 추가 메시지가 거부됩니다. 값 0은 제한이 없음을 의미합니다. 기본값: `0`
`imq.system.max_size`	(스왑 때문에) 메모리와 디스크 모두에 위치한 메시지의 총 크기(바이트, KB, MB 단위)를 지정합니다. 추가 메시지가 거부됩니다. 값 0은 제한이 없음을 의미합니다. 기본값: `0`
`imq.message.max_size`	메시지 본문의 최대 허용 크기(바이트, KB, MB 단위)를 지정합니다. 이 크기를 초과하는 메시지는 거부됩니다. 값 0은 제한이 없음을 의미합니다. 기본값: `70m` (MB)
`imq.`*resource_state*`.threshold`	각 메모리 자원 상태가 트리거되는 메모리 사용률을 지정합니다. 자원 상태는 초록, 노랑, 주황 및 빨강으로 표시됩니다. 기본값: 각각 0, 60, 75, 90
`imq.redelivered.optimization`	메시지가 재전송될 때마다(`true`) 또는 논리적으로 필요한 경우에 한해(`false`) 메시지 라우터가 재전송 플래그를 설정하여 성능을 최적화하는지 지정합니다(`true`/`false`). 기본값: `true`
`imq.transaction.autorollback`	`PREPARED` 상태에 있는 분산 트랜잭션이 브로커 시작 시 자동으로 롤백되는지 여부를 지정합니다(`true`/`false`). `false`라면 `imqcmd`를 사용하여 수동으로 트랜잭션을 완결하거나 롤백해야 합니다(157페이지의 "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 경로 지정 및 전달 정보를 데이터 저장소에 저장해야 합니다. *지속성 관리자* 구성 요소는 이 정보의 작성 및 검색을 관리합니다.

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

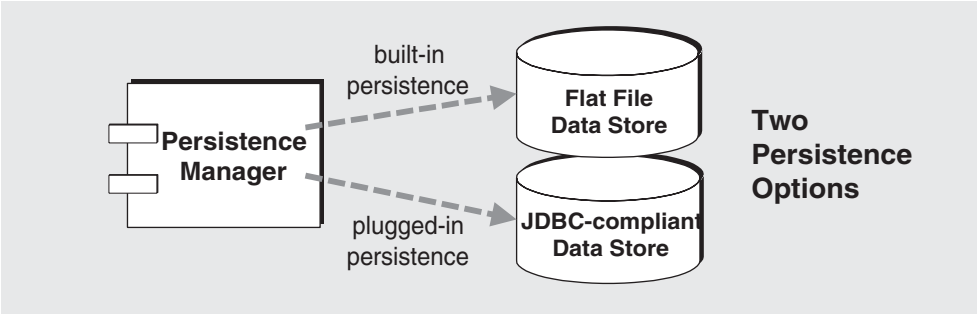
- 대상 다시 작성
- 각 주제의 영구 가입 목록 복원
- 각 메시지의 확인 목록 복원
- 완결된 모든 트랜잭션 상태 복제

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 경로 지정 테이블을 다시 작성합니다. 그런 다음 메시지 전달을 다시 시작할 수 있습니다.

MQ는 기본 제공 및 플러그 인 지속성 모듈을 모두 지원합니다(그림 2-4 참조). 기본 제공 지속성은 플랫 파일 데이터 저장소를 기반으로 합니다. 플러그 인 지속성은 JDBC (Java Database Connectivity) 인터페이스를 사용하며 JDBC 호환 데이터 저장소가 필요합니다. 일반적으로 기본 제공 지속성은 플러그 인 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 MQ 영구 저장소 솔루션은 플랫 파일 저장소입니다. 이 방법에서는 개별 파일을 사용하여 메시지, 대상, 영구 가입, 트랜잭션과 같은 지속성 데이터를 저장합니다.

플랫 파일 데이터 저장소의 위치는 다음과 같습니다.

`IMQ_VARHOME/instances/`*brokerName*`/filestore/`
(Solaris의 경우 `/var/imq/instances/`*brokerName*`/filestore/`)

여기서 *brokerName*은 브로커 인스턴스를 식별하는 이름입니다.

파일 기반 데이터 저장소는 지속성 메시지가 각자 해당 파일에 저장되도록, 즉 파일당 메시지 하나씩 포함하도록 구성됩니다. 그러나 대상, 영구 가입 및 트랜잭션은 모두 하나의 개별 파일에, 즉 모든 대상은 하나의 파일에, 모든 영구 가입은 또 다른 파일에 저장됩니다.

데이터 저장소에 메시지가 추가되고 삭제되면서 파일을 작성하고 삭제하려면 고비용의 파일 시스템 작업이 필요합니다. 따라서 MQ 구현에서는 이 메시지 파일을 재사용합니다. 어떤 파일이 더 이상 필요 없어지면 삭제하는 대신 재사용 가능 파일 풀에 추가됩니다. 이 파일 풀의 크기를 구성할 수 있습니다. 또한 재사용 태그 표시(잘라내지 않음)와 반대되는 개념으로 파일 풀에서 지울(0으로 잘라냄) 사용 가능 파일의 비율을 지정할 수 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. 파일을 지우면 디스크 공간은 덜 차지하지만, 브로커 종료 속도가 느려집니다.

플랫 파일 저장소에서의 메시지 저장 속도는 데이터 저장소가 사용 가능한 파일 설명자의 수에 따라 달라집니다. 설명자가 많으면 시스템은 많은 양의 지속성 메시지를 더 빨리 처리할 수 있습니다. 파일 설명자 수를 늘리는 것에 대한 정보는 MQ *릴리스 노트*의 "기술 노트" 절을 참조하십시오.

또한 대상 파일 저장소의 경우 대상이 추가되면 파일 크기를 늘리는 방법보다는 특정 크기의 파일에 대상을 추가하는 것이 더 효율적입니다. 따라서 최종적으로 저장할 대상의 수를 예상하여 대상 파일의 원래 크기를 설정하면 성능을 높일 수 있습니다(각 대상은 약 500바이트 사용).

데이터 저장소에는 소유 정보가 포함된 메시지가 있을 수 있으므로 *brokerName*`/filestore/` 디렉토리는 인증되지 않은 액세스로부터 보호하는 것이 바람직합니다. 자세한 지침은 *MQ 릴리스 노트*의 "기술 노트" 절을 참조하십시오.

플러그 인 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 A, "플러그 인 지속성 설정"을 참조하십시오.

지속성 관련 구성 등록 정보는 56페이지의 표 2-5를 참조하십시오(등록 정보 설정 지침은 5장, "브로커 시작 및 구성" 참조).

표 2-5 지속성 등록 정보

등록 정보 이름	설명
`imq.persist.store`	브로커가 기본 제공 파일 기반(`file`) 지속성 또는 플러그 인 JDBC 호환(`jdbc`) 지속성을 사용하고 있는지 지정합니다. 기본값: `file`
`imq.persist.file.destination.file.size`	기본 제공 파일 기반 지속성에 대해 대상 저장에 사용하는 파일의 초기 크기를 지정합니다. 기본값: `1m` (MB)
`imq.persist.file.message.filepool.limit`	기본 제공 파일 기반 지속성에 대해 파일 풀에서 재사용 가능한 최대 파일 수를 지정합니다. 이 수가 클수록 브로커는 지속성 데이터를 더 빠르게 처리할 수 있습니다. 이 값을 초과하는 사용 가능 파일은 삭제됩니다. 이 한도를 초과할 경우 브로커는 필요에 따라 추가 파일을 작성하고 삭제합니다. 기본값: `10000`
`imq.persist.file.message.filepool.cleanratio`	기본 제공 파일 기반 지속성에 대해 파일 풀에서 *clean* 상태(0으로 잘라낸 상태)에 있는 사용 가능 파일의 비율을 지정합니다. 이 값이 클수록 작업 중 파일을 지우는 데 필요한 오버헤드가 늘어나지만, 파일 풀에서 필요한 디스크 공간은 줄어듭니다. 기본값: `0`
`imq.persist.file.message.cleanup`	기본 제공 파일 기반 지속성에 대해 브로커가 종료 시 파일 저장소에서 사용 가능 파일을 지울 것인지 여부를 지정합니다. false 값은 브로커 종료 속도를 향상시키지만, 파일 저장소에 더 많은 디스크 공간이 필요합니다. 기본값: `false`

표 2-5 지속성 등록 정보(*계속*)

등록 정보 이름	설명
`imq.persist.file.message.fdpool.limit`	기본 제공 파일 기반 지속성에 대해 열린 상태를 유지할 데이터 최대 파일 수(즉, 파일 설명자 풀의 크기)를 지정합니다. 이 수가 클수록 지속성 작업 성능이 향상되지만, 클라이언트 연결 작성과 같이 파일 설명자를 필요로 하는 다른 브로커 작업은 지장을 받습니다. 기본값: `25` (Solaris 및 Linux), `1024` (Windows)
`imq.persist.file.sync.enabled`	지속성 작업이 메모리 상태에서 물리적 저장 장치와 동기화될 것인지 여부를 지정합니다. 값이 `true`이면 시스템 충돌로 인한 데이터 손실은 줄어들지만 지속성 작업의 성능이 저하됩니다. 기본값: `false`

보안 관리자

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

인증 및 권한 부여 기능은 사용자 저장소(58페이지의 그림 2-5 참조), 즉 메시징 시스템 사용자에 대한 정보(예: 아이디, 비밀번호, 그룹 멤버쉽)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 멤버쉽이 액세스 제어 파일과 함께 사용됩니다.

MQ 관리자는 MQ 제공 사용자 저장소(182페이지의 "플랫 파일 사용자 저장소 사용" 참조)를 채우거나 기존 LDAP 사용자 저장소를 보안 관리자 구성 요소에 플러그 인합니다. 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

MQ 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 59페이지의 "암호화(엔터프라이즈판)"를 참조하십시오. 별도로 각 연결 서비스가 사용하는 인코딩 유형을 구성하거나 브로커 전체에 대한 인코딩을 설정할 수 있습니다.

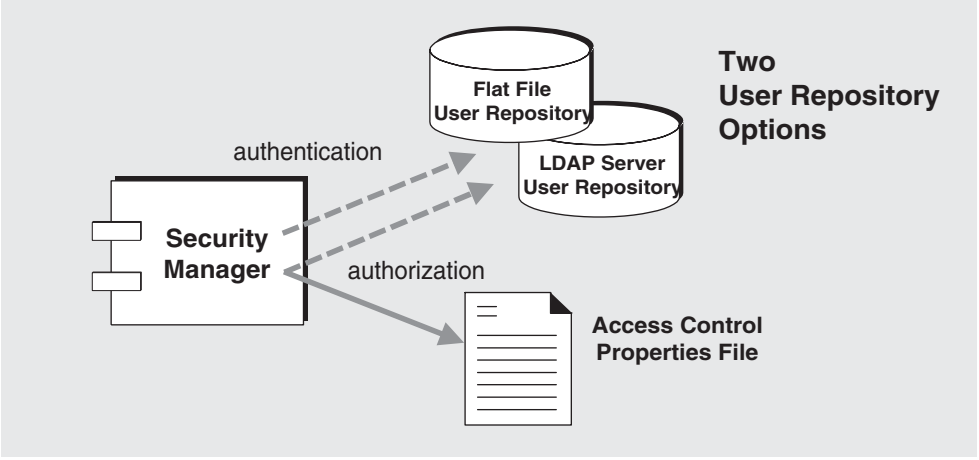
권한 부여

클라이언트 응용 프로그램 사용자가 인증되면 이 사용자는 여러 MQ 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 MQ 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 멤버쉽을 (사용자 저장소에 있는) 아이디/그룹 멤버쉽과 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



MQ 3.0.1에서는 기본 액세스 제어 등록 정보 파일이 *admin*이라는 단일 그룹만 명시적으로 참조합니다(185페이지의 "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.

MQ 관리자는 사용자 저장소에서 그룹 및 이 그룹의 사용자를 정의할 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(67페이지의 "자동 작성(대관리 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.

MQ SSL 기반 연결 서비스를 사용하기 위해 키 도구 유틸리티(`imqkeytool`)를 사용하여 개인 키/공용 키 쌍을 생성합니다. 이 유틸리티는 자체 서명된 인증서에 공용 키를 내장하고 이를 MQ 키 저장소에 저장합니다. MQ 키 저장소 자체도 비밀번호로 보호됩니다. 이 잠금을 해제하려면 시작할 때 키 저장소 비밀번호를 제공해야 합니다. 198페이지의 "암호화: SSL 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 2-6에서 확인할 수 있습니다(등록 정보 구성 지침은 5장, "브로커 시작 및 구성" 참조).

표 2-6 보안 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` 설정값을 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 189페이지의 표 8-5를 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 MQ 작업을 수행할 권한이 있는가를 시스템이 확인할 것인지 여부를 표시합니다. 기본값: `true`
`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 MQ 작업을 수행할 권한이 있는가를 시스템이 확인할 것인지 여부를 표시합니다. 기본값: `imq.accesscontrol.enabled` 등록 정보의 설정을 상속합니다.

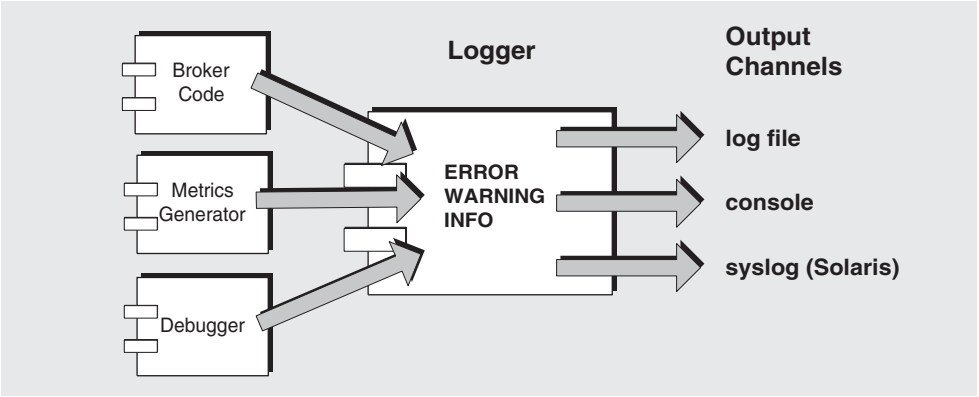
표 2-6 보안 등록 정보*(계속)*

등록 정보 이름	설명
`imq.accesscontrol.file.filename`	브로커가 지원하는 모든 연결 서비스에 대한 액세스 제어 등록 정보 파일의 이름을 지정합니다. 파일 이름은 `IMQ_HOME/etc` (Solaris의 경우 `/etc/imq`) 디렉토리에 대한 상대 파일 경로를 지정합니다. 기본값: `accesscontrol.properties`
`imq.`*service_name*`.accesscontrol.file.filename`	명명된 연결 서비스의 액세스 제어 등록 정보 파일 이름을 지정합니다. 파일 이름은 `IMQ_HOME/etc` (Solaris의 경우 `/etc/imq`) 디렉토리에 대한 상대 파일 경로를 지정합니다. 기본값: `imq.accesscontrol.file.filename`에서 지정한 설정을 상속합니다.
`imq.passfile.enabled`	안전한 통신을 위한 사용자 비밀번호(SSL, LDAP, JDBC)가 passfile에 설정되는지 여부를 지정합니다(true/false). 기본값: `false`
`imq.passfile.dirpath`	passfile을 포함하는 디렉토리 경로를 지정합니다. 기본값: Solaris의 경우 `IMQ_HOME/etc` (`/etc/imq`)
`imq.passfile.name`	passfile 이름을 지정합니다. 기본값: `passfile`
`imq.keystore.`*property_name*	SSL 기반 서비스에 대해 SSL 키 저장소와 관련된 보안 등록 정보를 지정합니다. 200페이지의 표 8-8을 참조하십시오.

로거

브로커는 로그 작업을 모니터링하고 진단할 다양한 구성 요소를 포함합니다. 그 중에는 데이터를 생성하는 구성 요소(브로커 코드, 메트릭 생성기, 디버거)가 있으며 또한 다양한 출력 채널(로그 파일, 콘솔, Solaris syslog)을 통해 정보를 기록하는 로거 구성 요소가 있습니다. 그림 2-6은 그러한 체계를 보여줍니다.

그림 2-6 로깅 체계



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

또한 가장 심각하고 중요한 정보(오류)부터 상대적으로 중요도가 낮은 정보(메트릭 데이터)까지 로거 수준을 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.

로거는 표준 출력(콘솔), 로그 파일 그리고 Solaris 플랫폼에서는 syslog 데몬 프로세스 등 다양한 출력 채널에 데이터를 기록할 수 있습니다.

각 출력 채널마다 로거에 대해 설정된 범주 중 어느 것이 해당 채널에 기록될 것인지 지정할 수 있습니다. 예를 들어, 로거 수준이 ERROR로 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 페이지를 참조하십시오.

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 파일이 저장되는 위치는 다음과 같습니다.

```
IMQ_VARHOME/instances/brokerName/log/
(Solaris의 경우 /var/imq/instances/brokerName/log/)
```

새 롤오버 로그 파일을 작성하면서 가장 최신의 로그 파일 9개로 구성된 아카이브를 보존합니다. 로그 파일은 다음과 같이 순차적 이름을 갖는 텍스트 파일입니다.

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

log.txt가 가장 최신 파일이며, 번호가 가장 높은 파일이 가장 오래된 것입니다.

브로커의 정보 생성 및 로깅을 설정하는 구성 가능한 등록 정보는 표 2-8에서 확인할 수 있습니다(등록 정보 구성 지침은 5장, "브로커 시작 및 구성" 참조).

표 2-8 로거 등록 정보

등록 정보 이름	설명
imq.metrics.enabled	메트릭 정보가 수집 중인지 여부를 지정합니다 (true/false). 기본값:true
imq.metrics.interval	메트릭 정보를 보고하는 시간 간격(초)을 지정합니다. 값이 0이면 보고하지 않습니다. 기본값: 0
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 세트, ALL 또는 NONE입니다. 기본값: ALL

표 2-8 로거 등록 정보*(계속)*

등록 정보 이름	설명
imq.log.file.dirpath	로그 파일을 포함하는 디렉토리 경로를 지정합니다. 기본값: IMQ_VARHOME/instances/*brokerName*/log/ (Solaris의 경우 /var/imq/instances/*brokerName*/log/)
imq.log.file.filename	로그 파일의 이름을 지정합니다. 기본값: log.txt
imq.log.file.rolloverbytes	새 로그 파일로 출력을 롤오버할 로그 파일 크기(바이트)를 지정합니다. 값 0은 파일 크기를 기준으로 한 롤오버가 없음을 의미합니다. 기본값: 0
imq.log.file.rolloversecs	새 로그 파일로 출력을 롤오버할 로그 파일 표시 시간(초)을 지정합니다. 값 0은 파일 표시 시간을 기준으로 한 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 세트, 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) MQ 브로커가 어떤 syslog 기능으로 로깅할 것인지 지정합니다. 값은 syslog(3C) 설명서 페이지에 있는 값을 미러링합니다. MQ에서 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.console	(Solaris에만 적용) 메시지를 syslog에 보낼 수 없을 경우 시스템 콘솔에 메시지를 기록할 것인지 여부를 지정합니다(true/false). 기본값: false
imq.log.syslog.identity	(Solaris에만 적용) syslog에 기록될 모든 메시지의 앞에 추가될 아이디 문자열을 지정합니다. 기본값: imqbrokerd_브로커 인스턴스 이름

표 2-8 로거 등록 정보*(계속)*

등록 정보 이름	설명
`imq.log.syslog.output`	(Solaris에만 적용) syslogd(1M)에 기록하는 로깅 정보의 범주를 지정합니다. 사용 가능한 값은 세로 막대(|)로 구분된 임의의 로깅 범주 세트, `ALL` 또는 `NONE`입니다. 기본값: `ERROR`

물리적 대상

MQ 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(37페이지의 "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 전달되기 전에 마샬링됩니다.

MQ 관리 도구(152페이지의 "대상 관리" 참조)를 사용하여 물리적 대상을 작성합니다. 또한 67페이지의 "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. 생성자 클라이언트로부터 도착한 메시지는 대기열에 있다가 사용자 클라이언트에게 전달됩니다.

대기 메시지의 경로 지정은 대기열의 전달 정책에 따라 달라집니다. MQ는 3가지 대기열 전달 정책을 구현합니다.

- **단일** 이 대기열에서는 단일 메시지 사용자에게 메시지를 전달하는 것만 가능합니다. 제2의 메시지 사용자가 해당 대기열에 등록을 시도하면 거부됩니다. 등록한 메시지 사용자가 연결을 끊으면 메시지는 더 이상 전달되지 않으며, 새 사용자가 등록할 때까지 메시지는 저장됩니다.

- **페일오버(엔터프라이즈판)** 이 대기열은 둘 이상의 메시지 사용자에게 메시지를 전달할 수 있지만, 기본 메시지 사용자(해당 브로커에 처음으로 등록한 사용자)가 연결을 끊을 때에만 그렇게 할 수 있습니다. 그 경우 메시지는 그 다음으로 등록한 메시지 사용자에게 전달되며, 이 사용자가 실패하는 경우처럼 특정 시점까지는 계속 그 사용자에게 전달됩니다. 등록한 메시지 사용자가 없으면 사용자가 등록할 때까지 메시지를 저장합니다.
- **라운드 로빈(엔터프라이즈판)** 이 대기열은 둘 이상의 메시지 사용자에게 메시지를 전달할 수 있습니다. 한 대기열에 대해 여러 사용자가 등록했다고 가정하면 대기열에 들어간 첫 번째 메시지는 첫 번째로 등록한 메시지 사용자에게 전달되고, 두 번째 메시지는 두 번째로 등록한 사용자에게 전달되는 방식입니다. 동일한 순서대로 동일한 사용자 세트에 추가 메시지는 전달됩니다. 사용자가 대기열에 등록하기 전에 여러 메시지가 대기열에 있을 경우, 한 사용자에게 집중되지 않도록 배치 방식으로 메시지가 전달됩니다. 한 메시지 사용자가 연결이 끊기면 그 사용자에게 경로가 지정되었던 메시지는 나머지 활성 사용자들에게 다시 배포됩니다. 이러한 재배포 때문에 사용자들에게 메시지가 전달되는 순서가 대기열에서 메시지가 수신되는 순서와 동일하다고 할 수는 없습니다.

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(153페이지의 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 전달됩니다. 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(153페이지의 표 6-10 참조).

자동 작성(대 관리 작성) 대상

JMS 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 영구성에 따라) 많은 자원을 사용할 수 있으므로 메시지 서버 성능 및 안정성을 보장하도록 면밀하게 관리해야 합니다. 따라서 MQ 관리자가 응용 프로그램을 대신하여 대상을 작성하고 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 나을 수 있습니다.

MQ는 이 *자동 작성* 기능을 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. 197페이지의 "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 MQ 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 2-9에 있는 등록 정보를 사용하여 자동 작성 기능이 사용 가능/불가능하도록 MQ 메시지 서버를 구성할 수 있습니다(등록 정보 구성 지침은 5장, "브로커 시작 및 구성" 참조).

표 2-9 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.queue.deliverypolicy`	자동 작성된 대기열의 기본 전달 정책을 지정합니다. 값은 `single`, `round-robin` 또는 `failover` 중에서 선택합니다. 기본값: `single`

임시 대상

임시 대상은 다른 클라이언트에게 보낸 메시지에 대해 응답을 받을 대상이 필요한 클라이언트 응용 프로그램이 (JMS API를 사용하여) 명시적으로 작성 및 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) 관리에 의해 작성되거나 자동으로 작성된 대상과 달리 영구 저장되지 않으며 브로커가 다시 시작할 때 다시 작성되지 않습니다. 또한 MQ 관리 도구에 표시되지 않습니다.

다중 브로커 클러스터(엔터프라이즈판)

MQ 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 파일 설명자, 메모리 제한과 같은 자원 제한 사항을 초과하게 됩니다. 늘어나는 로드를 수용하는 한 가지 방법은 MQ 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 클라이언트와 브로커를 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*. 클러스터 중에서 한 브로커를 실패하는 다른 브로커의 자동 백업으로 사용할 수 없습니다. 브로커의 자동 페일오버 보호는 MQ 버전 3.0.1에서 지원하지 않습니다(그러나 여러 브로커를 사용하여 사용자 정의 페일오버 체계를 구현하는 응용 프로그램 설계는 가능).

브로커 클러스터 구성 및 관리에 대해서는 126페이지의 "클러스터를 사용한 작업(엔터프라이즈판)"을 참조하십시오.

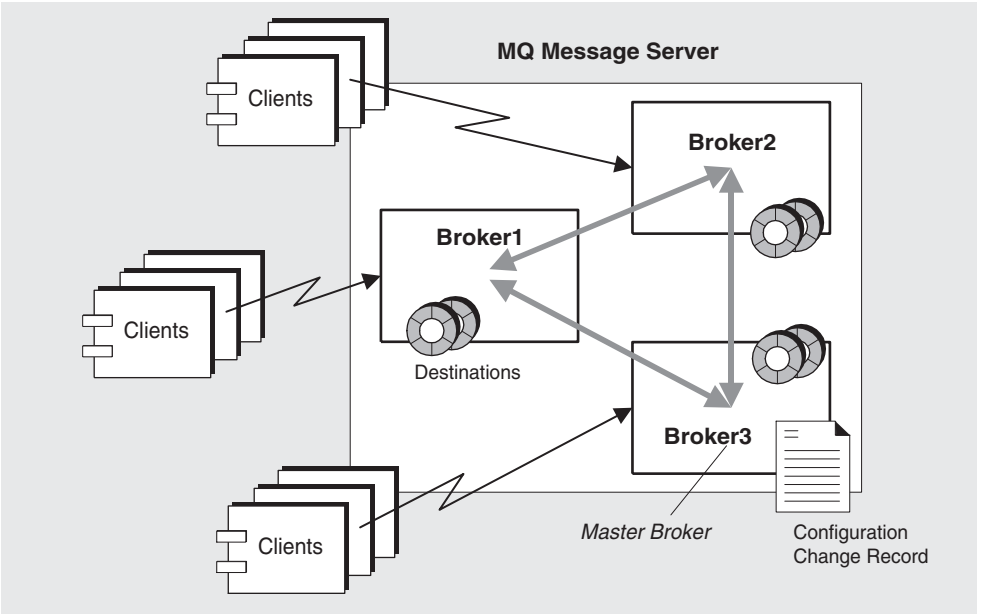
다음 절에서는 MQ 브로커 클러스터의 구조 및 내부 기능에 대해 설명합니다.

다중 브로커 구조

다중 브로커 메시지 서버에서는 그림 2-7과 같이 여러 브로커 인스턴스 사이에 클라이언트 연결을 분산시킬 수 있습니다. 클라이언트 관점에서 각 클라이언트는 개별 브로커(자신의 *홈* 브로커)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.

일반적으로 클러스터 내부의 브로커는 임의의 토폴로지로 연결할 수 있습니다. 그러나 MQ 버전 3.0.1은 완전 연결된 클러스터, 즉 69페이지의 그림 2-7에서처럼 각 브로커가 클러스터의 다른 모든 브로커와 직접 연결되는 토폴로지만 지원합니다.

그림 2-7 다중 브로커(클러스터) 구조



다중 브로커 구성에서는 클러스터의 모든 브로커에 각 대상의 인스턴스가 존재합니다. 또한 각 브로커는 다른 모든 브로커에 등록된 메시지 사용자에 대해서도 알고 있습니다. 따라서 각 브로커는 자신과 직접 연결된 메시지 생성자로부터 원격 메시지 사용자에게 메시지를 전달하고, 원격 생성자로부터 자신과 직접 연결된 사용자에게 메시지를 전달합니다.

클러스터 구성에서 각 메시지 생성자와 직접 연결된 브로커는 해당 사용자가 자신에게 보내는 메시지에 대해 경로 지정을 수행합니다. 따라서 지속성 메시지는 그 메시지의 *홈* 브로커에 의해 저장되고 경로가 지정됩니다.

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 됩니까?

오프라인된 브로커(또는 새로 추가된 브로커)를 수용하기 위해 MQ는 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, MQ는 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 `imqbrokerd` 명령의 `-backup` 옵션을 사용하면 됩니다(121페이지의 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(72페이지의 표 2-10 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 *않아도* 되는 개발 환경에서는 마스터 브로커가 별로 필요하지 않습니다. 마스터 브로커 *없이* 구성된 환경에서 MQ는 다른 브로커를 시작하기 위해 마스터 브로커가 반드시 실행 중이어야할 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(67페이지의 "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, MQ는 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 브로커 간 연결을 설정할 때 사용합니다. 또한 마스터 브로커를 사용할 경우 각 브로커는 마스터 브로커의 호스트 이름 및 포트 번호를 알고 있어야 합니다.

클러스터의 모든 브로커는 동일한 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 이 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.)

MQ 클러스터 구성 등록 정보는 표 2-10에서 확인할 수 있습니다(등록 정보 설정 지침은 126페이지의 "클러스터를 사용한 작업(엔터프라이즈판)" 참조).

표 2-10 클러스터 구성 등록 정보

등록 정보 이름	설명
`imq.cluster.brokerlist`	클러스터의 모든 브로커를 지정합니다. *host:port* 항목이 쉼표로 구분된 목록으로 구성되며, 여기서 *host*는 각 브로커의 호스트 이름, *port*는 포트 매퍼의 포트 번호입니다.
`imq.cluster.masterbroker`	클러스터 중에서 상태 변경을 추적하는 마스터 브로커(있는 경우)를 지정합니다. 등록 정보는 *host:port*로 구성되며, 여기서 *host*는 마스터 브로커의 호스트 이름, *port*는 포트 매퍼의 포트 번호입니다.
`imq.cluster.url`	클러스터 구성 파일의 위치를 지정합니다. 브로커들이 클러스터 등록 정보 값을 개별적으로 갖기보다는 단일 중앙 구성 파일을 참조할 경우 사용합니다. URL 문자열로 구성됩니다. 웹 서버에서 관리할 경우 일반 `http:`*URL*을 사용하여 액세스할 수 있습니다. 공유 드라이브에 있는 경우 `file:`*URL*을 사용하여 액세스할 수 있습니다.
`imq.cluster.port`	클러스터에 있는 *각* 브로커는 클러스터 연결 서비스를 위한 포트 번호를 지정하는 데 사용할 수 있습니다. 클러스터 연결 서비스는 클러스터 내의 브로커 간 내부 통신에 사용됩니다. 기본값: 0 (포트가 동적으로 할당됨)

표 2-10 클러스터 구성 등록 정보*(계속)*

등록 정보 이름	설명
`imq.cluster.hostname`	클러스터에 있는 *각* 브로커는 사용 가능한 호스트가 두 개 이상 있는 경우(예: 컴퓨터에 네트워크 인터페이스 카드가 두 개 이상인 경우) 클러스터 연결 서비스를 바인드할 호스트(호스트 이름 또는 IP 주소)를 지정하는 데 사용할 수 있습니다. 클러스터 연결 서비스는 클러스터 내의 브로커 간 내부 통신에 사용됩니다. 기본값: `null` (사용 가능한 모든 호스트)

클러스터 구성 파일은 브로커 세트가 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.

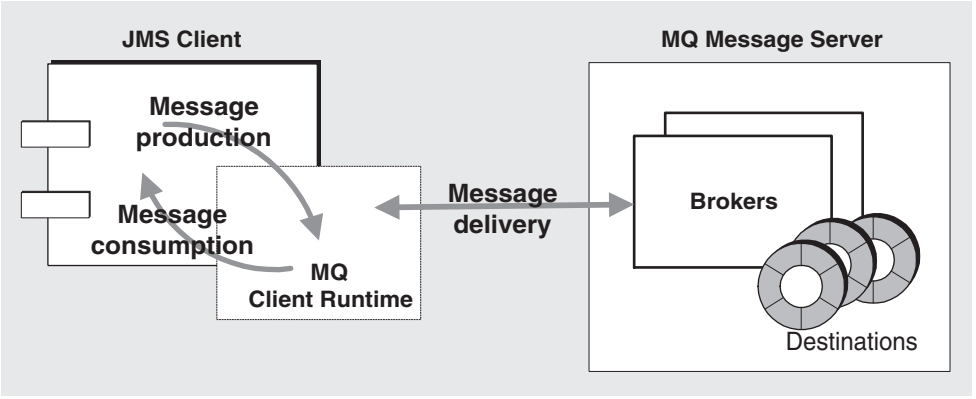
MQ 클라이언트 런타임

MQ 클라이언트 런타임은 클라이언트 응용 프로그램에 MQ 메시지 서버와의 인터페이스를 제공합니다. 즉, 클라이언트 응용 프로그램에 32페이지의 "JMS 프로그래밍 모델"에서 소개한 모든 JMS 프로그래밍 객체를 제공합니다. 클라이언트가 대상에게 메시지를 보내고 대상으로부터 메시지를 받는 데 필요한 모든 작업을 지원합니다.

이 절에서는 MQ 클라이언트 런타임의 작동 방식을 간략하게 설명합니다. 성능 관련 요소들은 클라이언트 응용 프로그램 설계 및 성능에 영향을 미치므로 MQ *Developer's Guide*에서 다룹니다.

74페이지의 그림 2-8은 메시지 생성 및 사용 시 클라이언트 응용 프로그램과 MQ 클라이언트 런타임 사이의 상호 작용 그리고 메시지 전달 시 MQ 클라이언트 런타임과 MQ 메시지 서버 사이의 상호 작용이 어떻게 이루어지는지 보여줍니다.

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 전달되었으며 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.

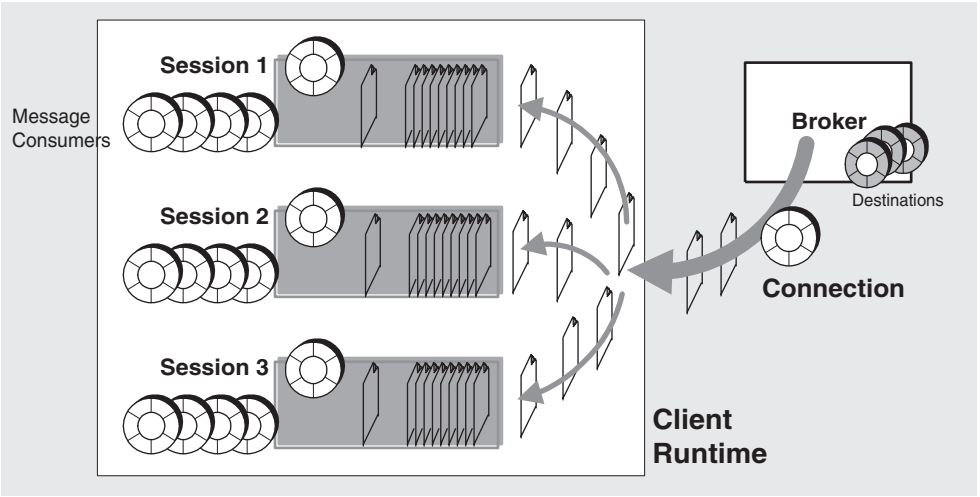
메시지 사용

메시지 사용은 생성보다 더 복잡합니다. 브로커의 대상에 도착한 메시지는 다음 조건에서 MQ 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 MQ 세션으로 배포되고, 그림 2-9에서처럼 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 MQ 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.

MQ 관리 객체

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

MQ는 ConnectionFactory와 Destination의 2가지 관리 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 사용합니다.

관리 객체를 사용하면 MQ 메시지 서버를 매우 쉽게 제어하고 관리할 수 있습니다.

- 클라이언트 응용 프로그램이 사전 구성된 ConnectionFactory 객체에 액세스하도록 요구함으로써 연결의 동작을 제어할 수 있습니다(167페이지의 "관리 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. 67페이지의 "자동 작성(대 관리 작성) 대상" 참조).
- 클라이언트 응용 프로그램이 설정한 메시지 헤더 값을 무시함으로써 MQ 메시지 서버 자원을 제어할 수 있습니다(167페이지의 "관리 객체 속성" 참조).

따라서 이러한 방법을 통해 MQ 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(37페이지의 "JMS 공급자 독립성" 참조) 또는 공급자별 구성 등록 정보를 알 필요가 없습니다.

7장, "관리 객체 관리"에서 설명하는 것처럼 MQ 관리 도구를 사용하여 관리 객체를 만듭니다. 관리 객체를 만들 때 읽기 전용으로, 즉 객체 생성 시 설정한 MQ 특정 구성 값을 클라이언트 응용 프로그램에서 변경할 수 없도록 지정할 수 있습니다. 다시 말해 클라이언트 코드는 읽기 전용 관리 객체에 대해 속성 값을 설정할 수 없으며, 관리자는 78페이지의 "클라이언트 시작 시 속성 값 무시"에서 설명하는 것처럼 응용 프로그램 시작 옵션을 사용하여 이 값을 무시할 수 없습니다.

클라이언트 응용 프로그램에서 ConnectionFactory와 Destination 관리 객체 모두를 자체적으로 인스턴스화할 수 있지만, 이는 MQ 관리자가 응용 프로그램에서 필요로 하는 브로커 자원을 제어하고 그 성능을 조정하게 하려는 관리 객체의 기본 목표에 어긋납니다. 또한 관리 객체를 직접 인스턴스화하면 클라이언트 응용 프로그램은 공급자의 영향을 받지 않고 공급자별 특성을 갖게 됩니다.

연결 팩토리 관리 객체

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

분산 트랜잭션을 지원하려면(40페이지의 "로컬 트랜잭션" 참조) 분산 트랜잭션을 지원하는 특정 XAConnectionFactory 객체를 사용해야 합니다.

ConnectionFactory 관리 객체를 만들려면 174페이지의 "연결 팩토리 추가"를 참조하십시오.

ConnectionFactory 관리 객체를 구성하여 이 객체가 생성하는 모든 연결에 공통적인 속성 값(등록 정보)을 지정할 수 있습니다. ConnectionFactory 및 XAConnectionFactory 객체는 동일한 속성 세트를 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
- 응용 프로그램 서버 지원
- JMS 정의 등록 정보 지원

각 범주와 관련 속성은 MQ *Developer's Guide*에서 자세히 다룹니다. MQ 관리자로서 이 속성 값을 조정해야 하는 경우가 있지만, 일반적으로 클라이언트 응용 프로그램의 성능 조정을 위해 어떤 속성을 조정할 것인가는 응용 프로그램 개발자가 결정합니다. 167페이지의 표 7-3은 이 속성을 알파벳 순서로 요약합니다.

대상 관리 객체

Destination 관리 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 객체를 만들려면 175페이지의 "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. 기본값: Untitled_Destination_Object
imqDestinationDescription	객체 관리에 유용한 정보를 지정합니다. 기본값: 대상 객체의 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 객체의 속성 값을 무시할 때 사용할 수 있습니다. 예를 들어 클라이언트 응용 프로그램 코드 중 JNDI 조회를 통해 액세스하는 관리 객체 구성을 무시할 수 있습니다.

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

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

여기서 *attribute*는 167페이지의 "연결 팩토리 관리 객체"에서 소개하는 ConnectionFactory 관리 객체 속성 중 하나를 의미합니다.

예를 들어 클라이언트 코드에서 액세스하는 ConnectionFactory 관리 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 객체가 읽기 전용으로 설정된 경우 그 속성 값은 명령줄 무시로 변경할 수 없습니다. 그러한 무시는 적용되지 않습니다.

MQ 관리

Sun™ ONE Message Queue (MQ) 관리는 다양한 작업 및 이 작업을 수행하기 위한 다양한 도구로 구성됩니다.

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

MQ 관리 작업

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

개발 환경

개발 환경에서의 작업은 MQ 클라이언트 응용 프로그램을 프로그래밍하는 데 중점을 둡니다. MQ 메시지 서버는 주로 테스트 용도로 필요합니다. 개발 환경에서는 유연성이 강조되고 관리는 주로 개발자가 테스트에 사용할 브로커를 시작하는 정도로 최소화됩니다. 일반적으로 개발 테스트에는 데이터 저장소, 사용자 저장소, 액세스 제어 등록 정보 및 객체 저장소의 기본 구현이 적합합니다. 다중 브로커 테스트를 수행 중이라면 마스터 브로커는 필요하지 않을 것입니다. 또한 테스트 대상 응용 프로그램은 일반적으로 자동 작성된 대상을 사용하며, 중앙에서 관리하는 관리 객체는 별로 사용하지 않을 것입니다.

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

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

- 보안(8장, "보안 관리" 참조)
  - 파일 기반 사용자 저장소에 항목을 만들거나 브로커가 기존 LDAP 사용자 저장소를 사용하도록 구성합니다.

    (최소한 관리 기능은 비밀번호로 보호합니다.)
  - 액세스 제어 등록 정보 파일에서 액세스 설정을 수정합니다.
  - SSL 기반 연결 서비스를 설정합니다.
- 관리 객체(7장, "관리 객체 관리" 참조)
  - LDAP 객체 저장소를 구성하거나 설정합니다.
  - 연결 팩토리 및 대상 관리 객체를 만듭니다.
- 브로커 클러스터(126페이지의 "클러스터를 사용한 작업(엔터프라이즈판)" 참조)
  - 중앙 구성 파일을 작성합니다.
  - 마스터 브로커를 사용합니다.
- 지속성: 브로커가 기본 제공 지속성보다는 플러그 인 지속성을 사용하도록 구성합니다(부록 A, "플러그 인 지속성 설정" 참조).

유지 보수 작업

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

- 응용 프로그램 관리
  - 브로커의 자동 작성 기능을 사용 불가능하게 합니다(67페이지의 표 2-9 참조).
  - 응용 프로그램을 대신하여 물리적 대상을 작성합니다(153페이지의 "대상 만들기" 참조).
  - 대상에 대한 사용자 액세스를 설정합니다(192페이지의 "사용자 권한 부여: 액세스 제어 등록 정보 파일" 참조).
  - 대상을 모니터하고 관리합니다(152페이지의 "대상 관리" 참조).
  - 영구 가입을 모니터하고 관리합니다(155페이지의 "영구 가입 관리" 참조).
  - 트랜잭션을 모니터하고 관리합니다(157페이지의 "트랜잭션 관리" 참조).
- 브로커 관리 및 조정
  - 브로커 메트릭을 사용하여 브로커를 조정 및 재구성합니다.
  - 브로커 메모리 자원을 관리합니다.
  - 클러스터에 브로커를 추가하여 로드 균형을 조정합니다.
  - 실패한 브로커를 복구합니다.
- 관리 객체 관리
  - 필요에 따라 연결 팩토리 및 대상 관리 객체를 추가로 작성합니다.
  - 연결 팩토리 속성을 조정하여 성능 및 처리 능력을 향상시킵니다(77페이지의 "연결 팩토리 관리 객체" 참조).

MQ 관리 도구

MQ 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 MQ 관리 객체를 관리할 수 있습니다. 나머지 명령줄 유틸리티(imqbrokerd, imqusermgr, imqdbmgr 및 imqkeytool)는 그림 3-1에서처럼 해당 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
- 객체 저장소에 연결
- 객체 저장소에 관리 객체 추가

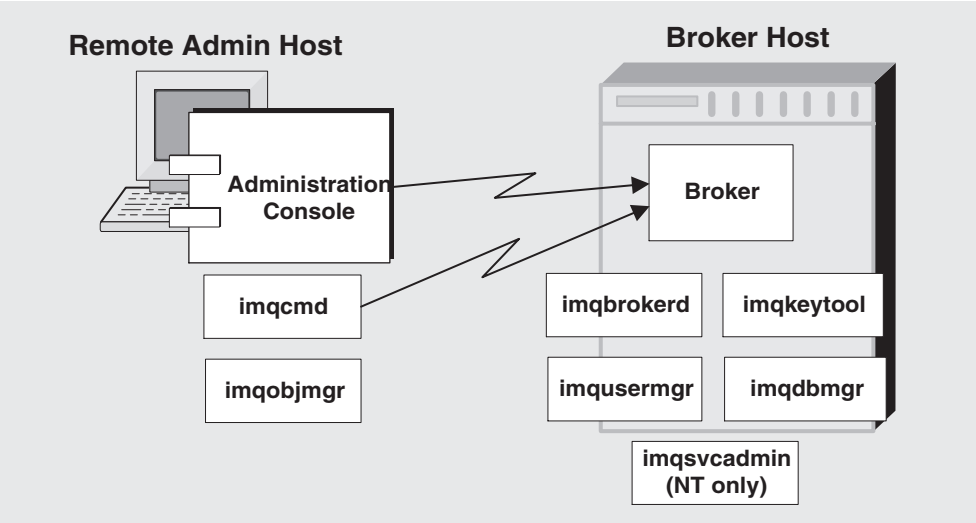
관리 콘솔로 수행할 수 없는 작업이 있으며, 그 대표적인 예로 브로커 시작, 브로커 클러스터 작성, 보다 특별한 브로커 등록 정보 구성, 사용자 데이터베이스 관리를 들 수 있습니다.

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

그림 3-1 로컬 및 원격 관리 유틸리티



모든 MQ 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. 이 유틸리티는 5장, "브로커 시작 및 구성"에 설명되어 있습니다.

명령(imqcmd) 브로커를 시작한 후 명령 유틸리티를 사용하여 물리적 대상을 작성, 업데이트 및 삭제하고 브로커 및 연결 서비스를 제어하며 브로커 자원을 관리합니다. `imqcmd` 명령을 사용하여 이 유틸리티를 실행합니다. 이 유틸리티는 6장, "브로커 및 응용 프로그램 관리"에 설명되어 있습니다.

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` 명령을 사용하여 이 유틸리티를 실행합니다. 이 유틸리티는 7장, "관리 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 명령을 사용하여 이 유틸리티를 실행합니다. 이 유틸리티는 8장, "보안 관리"에 설명되어 있습니다.

키 도구(imqkeytool) 키 도구 유틸리티는 SSL 인증에 사용하는 자체 서명된 인증서를 생성할 때 사용합니다. imqkeytool 명령을 사용하여 이 유틸리티를 실행하며, 자세한 내용은 8장, "보안 관리" 및 부록 B, "HTTP/HTTPS 지원(엔터프라이즈판)"을 참조하십시오.

데이터베이스 관리자(imqdbmgr) 데이터베이스 관리자 유틸리티는 영구 저장소에 사용하는 JDBC 호환 데이터베이스를 작성하고 관리할 때 사용합니다. imqdbmgr 명령을 사용하여 이 유틸리티를 실행합니다. 자세한 내용은 부록 A, "플러그 인 지속성 설정"을 참조하십시오.

서비스 관리자(imqsvcadmin) 서비스 관리자 유틸리티는 브로커를 Windows 서비스로서 설치, 쿼리 및 제거할 때 사용합니다. 자세한 내용은 부록 C, "브로커를 Windows 서비스로 사용"을 참조하십시오.

명령줄 구문

MQ 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 MQ 유틸리티, 예를 들어 imqcmd, imqobjmgr, imqusermgr 등의 이름을 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 87페이지의 표 3-1을 참조하십시오.
- 하위 명령, 인수, 옵션 및 옵션 인수는 공백으로 구분합니다.

다음은 하위 명령절이 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 myQueue라는 이름의 대기열 유형 대상을 제거합니다.

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

공통 명령줄 옵션

표 3-1은 모든 MQ 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 MQ 명령줄 옵션

옵션	설명
-h	지정한 유틸리티의 사용 도움말을 표시합니다.
-H	속성 목록 및 예를 포함하여 확장 사용 도움말을 표시합니다(imqcmd및 imqobjmgr만 지원).
-s	자동 모드를 시작합니다. 표시되는 출력이 없습니다. imqbrokerd의 경우 -silent로 지정합니다.
-v	버전 정보를 표시합니다.
-f	사용자 확인 프롬프트 없이 해당 작업을 수행합니다.
-pre	(imqobjmgr에서만 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.
-javahome *path*	사용할 대체 Java 2 runtime의 위치를 지정합니다.

관리 콘솔 자습서

이 자습서는 Sun™ ONE Message Queue (MQ) 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 객체 추가

이 자습서에서는 간단한 JMS 호환 응용 프로그램, `SimpleJNDIClient`를 실행하는 데 필요한 대상 및 관리 객체를 설정합니다. 자습서 마지막 부분에서 이 응용 프로그램을 실행합니다.

이 자습서는 관리 콘솔을 사용하여 기본 관리 작업을 수행하는 방법을 안내하기 위해 작성된 것입니다. 이 자습서를 참조하더라도 MQ *Developer's Guide*나 이 *관리자 설명서*의 나머지 장을 읽어 보아야 합니다.

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

- 일부 브로커 등록 정보 구성

  일부 브로커 등록 정보는 관리 콘솔을 사용하여 구성할 수 없습니다. 이 등록 정보는 113페이지의 5장, "브로커 시작 및 구성" 또는 147페이지의 "브로커 업데이트"에서 설명한 대로 구성할 수 있습니다.

- 브로커 클러스터 만들기

  자세한 내용은 126페이지의 "클러스터를 사용한 작업(엔터프라이즈판)"을 참조하십시오.

- 사용자 데이터베이스 관리

  자세한 내용은 182페이지의 "사용자 인증"을 참조하십시오.

준비

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

이 자습서에서 항목1 > 항목2 > 항목3을 선택하는 것은 항목1이라는 풀다운 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
- MQ 관리 객체를 저장하는 객체 저장소에 연결

  관리 객체를 사용하여 JMS 호환 응용 프로그램의 메시징 요구 사항을 관리할 수 있습니다. 자세한 내용은 75페이지의 "MQ 관리 객체"를 참조하십시오.

➤ **관리 콘솔을 시작하는 방법**

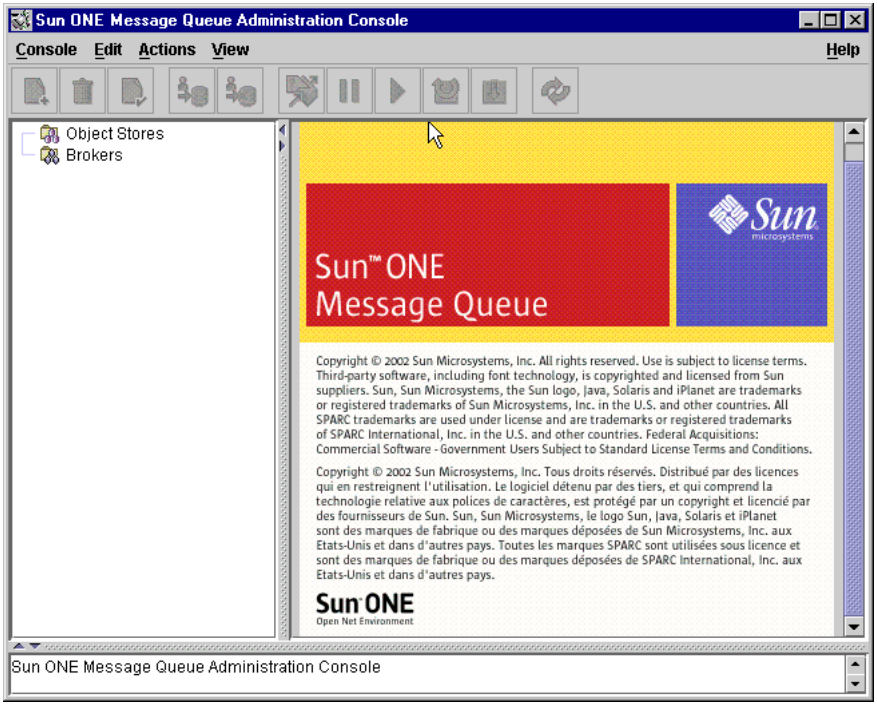
1. 시작 > 프로그램 > Sun ONE Message Queue 3.0 > 관리를 선택합니다.

   콘솔 창이 표시되려면 몇 초 정도 기다려야 합니다.

   비 Windows 사용자: 명령 프롬프트에 다음 명령을 입력합니다.

   `$IMQ_HOME/bin/imqadmin (Solaris의 경우 /usr/bin/imqadmin)`

2. 몇 초간 콘솔 창을 검사합니다.



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

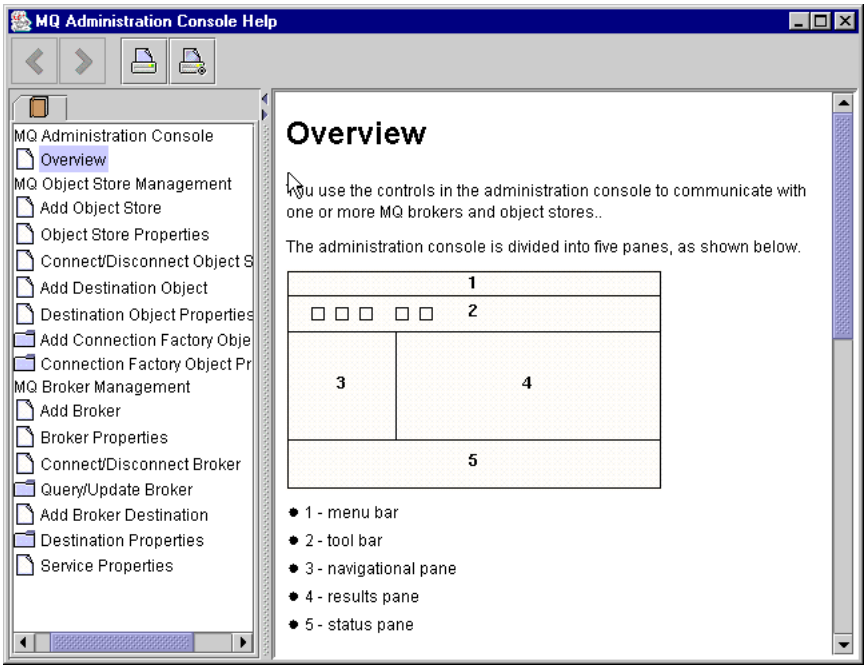
자습서만으로는 완전한 정보를 제공할 수는 없으므로, 우선 관리 콘솔에 대한 도움말 정보를 얻는 방법을 알아 봅니다.

도움말 확인

메뉴 표시줄 맨 오른쪽에서 Help 메뉴를 찾습니다.

➤ **관리 콘솔 도움말 정보를 표시하는 방법**

1. Help 풀다운 메뉴에서 Overview를 선택합니다. 도움말 창이 표시됩니다.



도움말 정보의 구성 방식에 유의하십시오. 왼쪽 창은 목차를, 오른쪽 창은 왼쪽에서 선택한 항목의 내용을 표시합니다.

Help 창의 오른쪽 창을 확인합니다. 이 창에는 관리 콘솔의 구조 보기를 표시하고 각 콘솔 창의 사용을 보여줍니다.

2. Help 창의 목차 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, 각 파일은 단순한 대화 상자나 탭에 대한 도움말을 제공합니다.

   첫 번째 콘솔 관리 작업인 95페이지의 "브로커 추가"에서는 콘솔을 통해 관리하는 브로커에 대한 참조를 작성합니다. 그러나 시작하기 전에 온라인 도움말에서 정보를 확인하십시오.

3. Help 창의 목차 창에서 Add Broker 항목을 누릅니다.

   목차 창이 변경됩니다. 이제 브로커 추가의 의미 및 브로커 추가 대화 상자에서의 각 필드 사용법을 설명하는 텍스트가 표시됩니다. 필드 이름은 굵은 체로 표시됩니다.

4. 도움말 텍스트를 읽어 봅니다.

5. Help 창을 닫습니다.

브로커 작업

브로커는 MQ 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 처리 능력 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 46페이지의 "연결 서비스"를 참조하십시오.

브로커 시작

관리 콘솔을 사용하여 브로커를 시작할 수 없습니다. 아래 설명대로 브로커를 시작합니다 (또한 5장, "브로커 시작 및 구성" 참조).

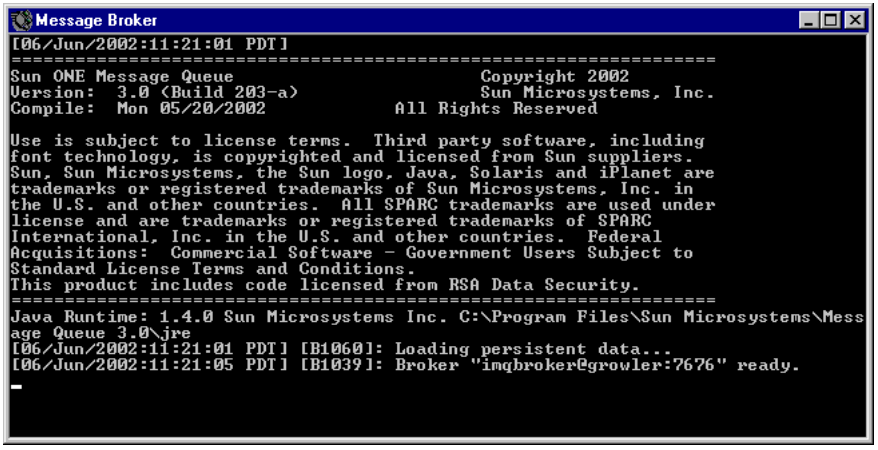
➤ 브로커를 시작하는 방법

1. 시작 > 프로그램 > Sun ONE Message Queue 3.0 > 브로커를 선택합니다.

   비 Windows: 다음 명령을 입력하여 브로커를 시작합니다.

   `%$IMQ_HOME/bin/imqbrokerd` (Solaris의 경우 `/usr/bin/imqbrokerd`)

   브로커 프로세스 창이 표시됩니다. 브로커 이름이 지정되고 준비 중인 상태로 표시됩니다.



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 Brokers를 마우스 오른쪽 버튼으로 누른 다음 Add Broker를 선택합니다.
2. Broker Label 필드에 `MyBroker`라고 입력합니다.

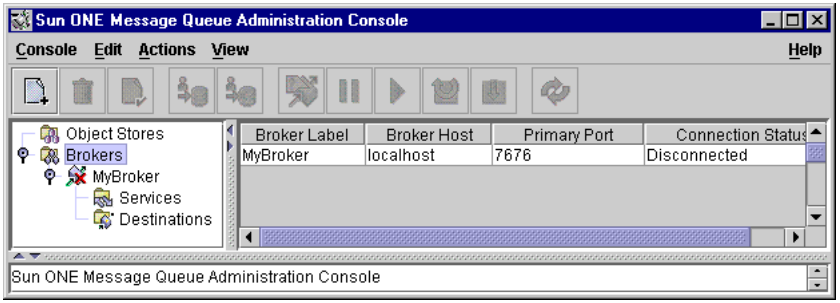
   관리 콘솔에서 이 브로커를 식별하는 레이블을 제공합니다.

   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.

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

3. OK를 눌러 브로커를 추가합니다.

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



4. MyBroker를 마우스 오른쪽 버튼으로 누르고 팝업 메뉴에서 Properties를 선택합니다.

   브로커 등록 정보 대화 상자가 표시됩니다. 이 대화 상자에서는 브로커를 추가할 때 지정했던 등록 정보를 업데이트할 수 있습니다.

관리 비밀번호 변경

브로커를 추가할 때 비밀번호를 지정하지 않았다면 브로커에 연결할 때 비밀번호를 입력하라는 프롬프트가 표시됩니다. 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
   ```

   (Solaris 및 Linux에서 이 작업을 수행하려면 루트로 로그인해야 합니다)

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

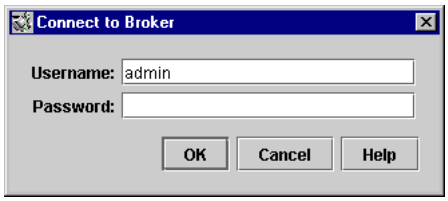
클라이언트는 관리자와 다른 연결 서비스를 사용하지만, 클라이언트에도 기본 아이디와 비밀번호가 지정되어 광범위한 관리 설정 작업 없이 MQ를 테스트할 수 있습니다. 기본적으로 클라이언트는 `guest` 아이디와 `guest` 비밀번호를 사용하여 브로커에 연결할 수 있습니다. 그러나 가급적 일찍 클라이언트를 위한 안전한 아이디와 비밀번호를 설정해야 합니다. 자세한 정보는 182페이지의 "사용자 인증"을 참조하십시오.

브로커에 연결

➤ **브로커에 연결하는 방법**

1. `MyBroker`를 마우스 오른쪽 버튼으로 누르고 Connect to Broker를 선택합니다.

   아이디와 비밀번호를 지정할 수 있는 대화 상자가 표시됩니다.



2. Password 필드에 `admin`을 입력하거나, 96페이지의 "관리 비밀번호 변경"에서 비밀번호로 지정한 값을 입력합니다.

   `admin` 아이디와 정확한 비밀번호를 제시하면 관리자 권한으로 브로커에 연결됩니다.

3. OK를 눌러 브로커에 연결합니다.

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

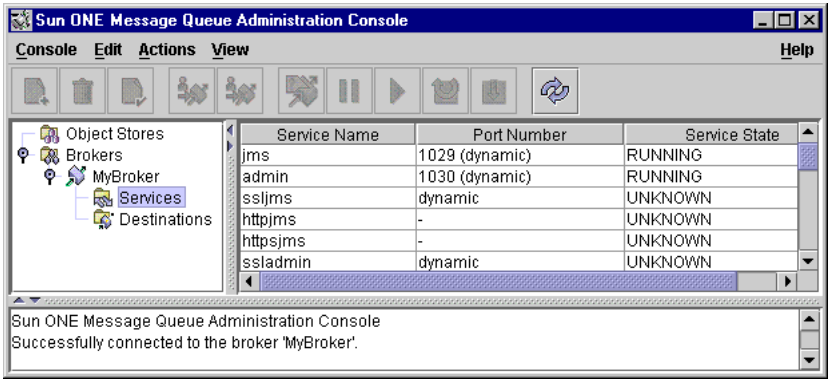
연결 서비스 보기

브로커는 해당 브로커에서 제공하는 연결 서비스와 지원하는 물리적 대상으로 구별됩니다.

➤ 사용 가능한 연결 서비스를 보는 방법

1. 탐색 창에서 Services를 선택합니다.

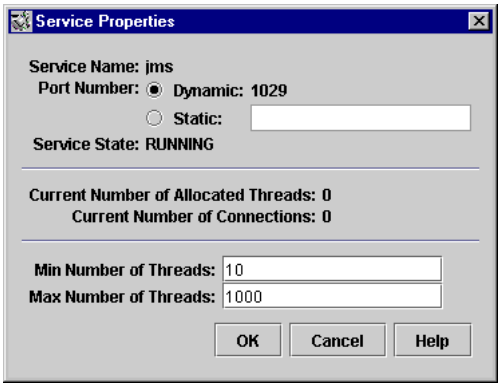
   사용 가능한 서비스가 결과 창에 표시됩니다. 각 서비스마다 이름, 포트 번호 및 상태가 제시됩니다.



2. 결과 창에서 jms 서비스를 눌러 선택합니다.
3. Actions 풀다운 메뉴를 누르고 강조 표시된 항목을 확인합니다.

   jms 서비스를 중지하거나 그 등록 정보를 확인하고 업데이트할 수 있습니다.
4. Actions 메뉴에서 Properties를 선택합니다.

   Service Properties 대화 상자를 사용하여 서비스에 정적 포트 번호를 지정하고 이 서비스에 할당된 최소 및 최대 스레드 수를 변경할 수 있습니다.



5. OK나 Cancel을 눌러 등록 정보 대화 상자를 닫습니다.
6. 결과 창에서 관리 서비스를 선택합니다.
7. Actions 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. Actions > Properties를 선택하여 관리 서비스의 등록 정보를 확인합니다.
9. 작업을 마쳤으면 OK나 Cancel을 누릅니다.

브로커에 물리적 대상 추가

브로커에 물리적 대상을 명시적으로 만들어야 JMS 호환 응용 프로그램이 제대로 실행될 수 있습니다. 브로커에 대상 자동 작성을 사용 가능하게 하여 물리적 대상이 동적으로 작성될 수 있다면 이 작업을 수행할 필요가 없습니다.

대상 자동 작성은 개발 환경에 적합합니다. 그러나 작업 설정에서는 이 기능을 해제하고 관리자가 명시적으로 작성한 물리적 대상을 브로커가 사용하는 것이 바람직합니다. 그렇게 하면 관리자는 브로커에서 사용 중인 대상을 완전히 파악할 수 있습니다.

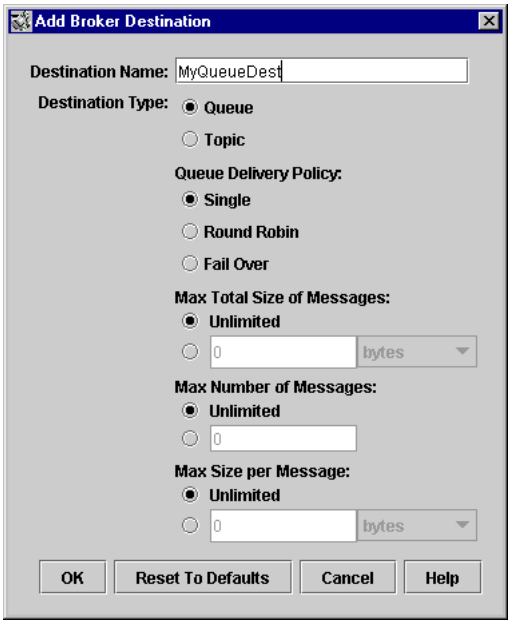
imq.autocreate.topic 또는 imq.autocreate.queue 등록 정보를 설정하여 브로커가 자동 작성된 대상을 추가할 수 있는지 여부를 제어합니다. 자세한 내용은 67페이지의 "자동 작성(대 관리 작성) 대상"을 참조하십시오.

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

➤ 브로커에 대기열 대상을 추가하는 방법

1. MyBroker의 Destinations 노드를 마우스 오른쪽 버튼으로 누르고 Add Broker Destination을 선택합니다.

   다음 대화 상자가 표시됩니다.



2. Destination Name 필드에 MyQueueDest를 입력합니다.
3. Queue 라디오 버튼이 선택되어 있지 않으면 선택합니다.
4. Queue Delivery Policy는 Single로 선택해야 합니다.
5. OK를 눌러 물리적 대상을 추가합니다.

   이제 결과 창에 대상이 표시됩니다.

물리적 대상 작업

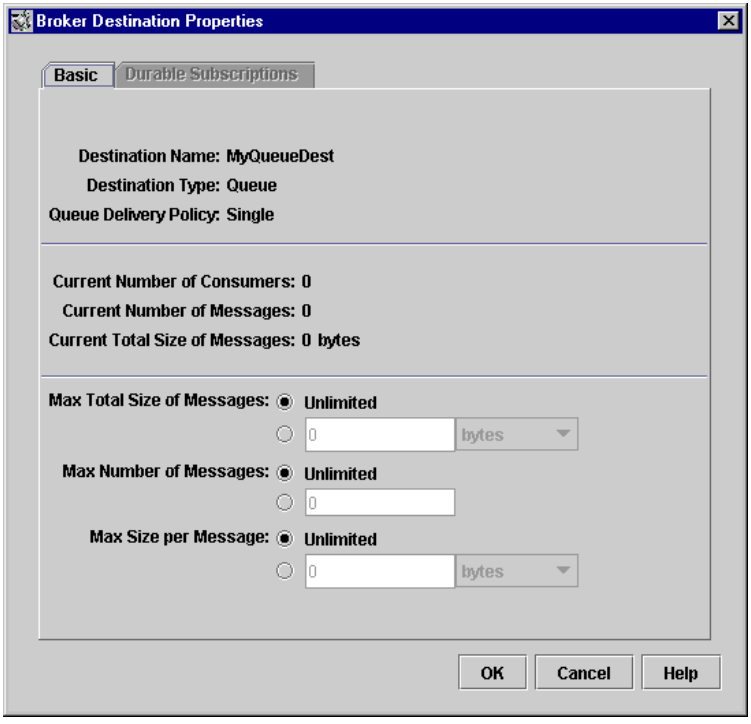
브로커에 물리적 대상을 추가했다면 아래 절차에 설명된 대로 다음 작업을 수행할 수 있습니다.

- 물리적 대상의 등록 정보 확인 및 업데이트
- 대상에서 메시지 제거
- 대상 삭제

➤ 물리적 대상의 등록 정보를 보는 방법

1. MyBroker의 Destinations 노드를 선택합니다.
2. 결과 창에서 MyQueueDest를 선택합니다.
3. Actions > Properties를 선택합니다.

   다음 대화 상자가 표시됩니다.



대기열에 대해 변경 가능한 등록 정보는 해당 대기열로 전달되는 메시지의 크기 및 수와 관련되어야 합니다.

4. Cancel을 눌러 대화 상자를 닫습니다.

➤ **대상에서 메시지를 제거하는 방법**

1. 결과 창에서 물리적 대상을 선택합니다.
2. Actions > Purge Messages를 선택합니다.

   확인 대화 상자가 표시됩니다.

메시지 제거 기능은 메시지를 삭제하고 빈 대상을 남겨 둡니다.

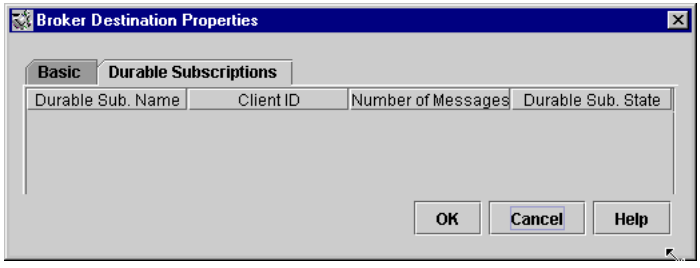
➤ **대상을 삭제하는 방법**

1. 결과 창에서 물리적 대상을 선택합니다.
2. Edit > Delete를 선택합니다.

대상 삭제 기능은 대상에서 메시지를 제거하고 대상도 제거합니다.

주제 대상에 대한 정보 얻기

주제 대상 정보 대화 상자에는 영구 가입에 대한 정보를 나열하는 추가 탭이 있습니다.



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 MQ 특정 구현 및 구성 정보를 캡슐화하는 MQ 관리 객체 저장에 사용합니다.

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

관리 객체에 대한 자세한 내용은 75페이지의 "MQ 관리 객체"를 참조하십시오.

관리 콘솔을 사용하여 객체 저장소를 *작성*할 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

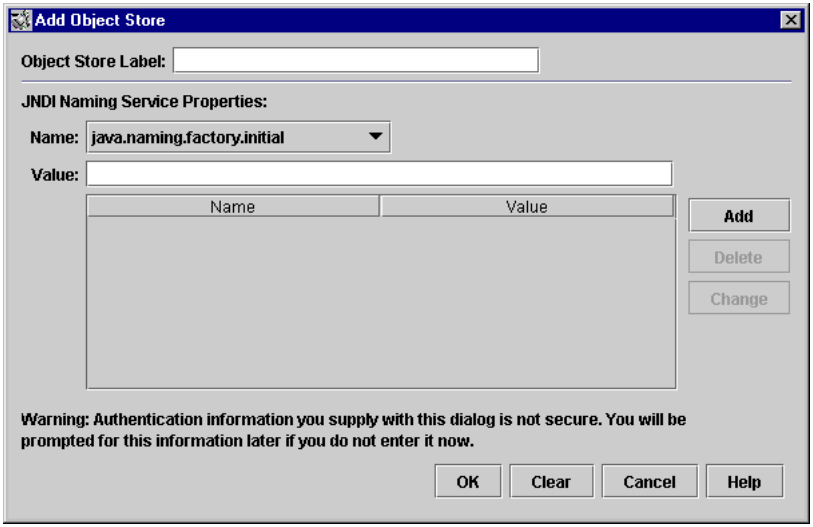
1. 아직 C 드라이브에 `Temp`라는 이름의 폴더가 없다면 지금 만듭니다.

   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. Object Stores를 마우스 오른쪽 버튼으로 눌러 Add Object Store를 선택합니다.

   다음 대화 상자가 표시됩니다.



3. Object Store Label이라는 이름의 필드에 `MyObjectStore`라고 입력합니다.

   이는 관리 콘솔에서 객체 저장소를 표시할 레이블을 제공합니다.

   다음 단계에서는 JNDI 이름/값 쌍을 입력해야 합니다. 이 쌍은 JMS 호환 응용 프로그램에서 관리 객체를 조회할 때 사용합니다.

4. Name 풀다운 메뉴에서 `java.naming.factory.initial`을 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. Value 필드에 다음을 입력합니다.

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

   이는 파일 시스템 저장소를 사용한다는 의미입니다(LDAP 저장소의 경우 `com.sun.jndi.ldap.LdapCtxFactory`를 지정합니다).

   작업 환경에서는 LDAP 디렉토리 서버를 객체 저장소로 사용할 것입니다. 서버 설정 및 JNDI 조회 수행에 대한 정보는 169페이지의 "객체 저장소 속성"을 참조하십시오.

6. Add 버튼을 누릅니다.

   등록 정보 및 그 값은 이제 등록 정보 요약 창에 표시됩니다.

7. Name 풀다운 메뉴에서 `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. Value 필드에 다음을 입력합니다.

   `file:///C:/Temp`

   (Solaris 및 Linux의 경우 `file:///tmp`)

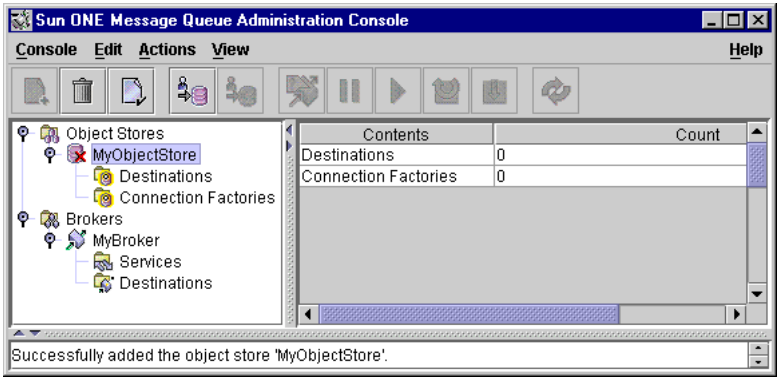
9. Add 버튼을 누릅니다.

   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. OK를 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 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 Properties를 선택합니다.

   객체 저장소를 추가할 때 지정한 모든 등록 정보를 보여주는 대화 상자가 표시됩니다. 이 등록 정보를 변경하고 OK를 누르면 기존 정보를 업데이트합니다.
3. 대화 상자를 닫으려면 OK나 Cancel을 누릅니다.

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 Connect to Object Store를 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 사용합니다. 연결 팩토리 구성을 통해 이 연결 팩토리를 사용하여 작성되는 연결의 동작을 제어할 수 있습니다.

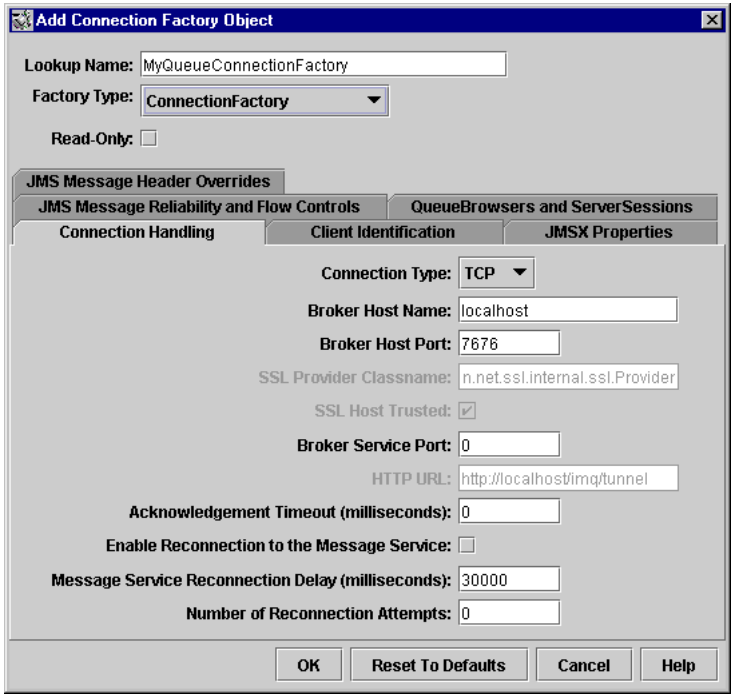
연결 팩토리 구성에 대한 정보는 온라인 도움말 및 MQ *Developer's Guide*를 참조하십시오.

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

➤ 객체 저장소에 연결 팩토리를 추가하는 방법

1. 아직 MyObjectStore에 연결되지 않았다면 지금 연결합니다(106페이지의 "객체 저장소에 연결" 참조).
2. Connection Factories 노드를 마우스 오른쪽 버튼으로 누르고 Add Connection Factory Object를 선택합니다.

   Add Connection Factory Object 대화 상자가 표시됩니다.



3. Lookup Name 필드에 "MyQueueConnectionFactory"라고 입력합니다.

   이 이름은 SimpleAdmin.java의 다음 행에서 알 수 있듯이 클라이언트 코드가 연결 팩토리를 조회할 때 사용합니다.

   ```
   qcf=(javax.jms.QueueConnectionFactory)
                        ctx.lookup("MyQueueConnectionFactory")
   ```

4. 풀다운 메뉴에서 QueueConnectionFactory를 선택하여 연결 팩토리 유형을 지정합니다.

5. Broker Host Name 및 Broker Host Port 필드에 클라이언트가 연결할 브로커의 호스트 이름 및 포트를 입력합니다.

   이 자습서에서는 클라이언트가 기본 브로커, 즉 7676 포트의 localhost에 있는 브로커에 연결합니다. 따라서 이 필드를 변경할 필요가 없습니다.

6. 이 대화 상자에 있는 탭을 차례로 눌러 연결 팩토리에 대해 구성 가능한 정보 종류를 확인합니다. Add Connection Factory Object 대화 상자의 오른쪽 아래에 있는 Help 버튼을 사용하여 각 탭에 대한 정보를 얻을 수 있습니다. 지금은 기본값을 바꾸지 마십시오.

7. OK를 눌러 대기열 연결 팩토리를 만듭니다.

8. 결과 창을 확인합니다. 새로 만든 연결 팩토리의 조회 이름 및 유형이 표시됩니다.

대상 관리 객체 추가

대상 관리 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.

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

- 관리 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.

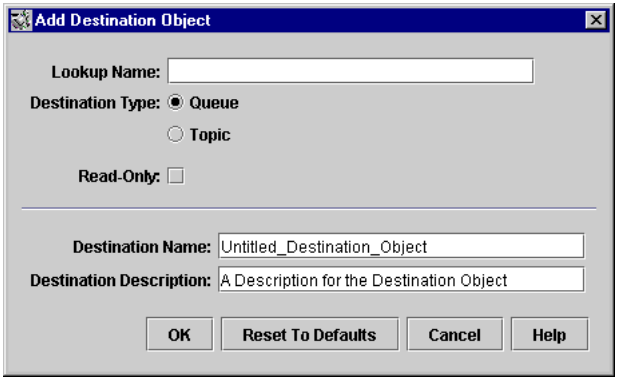
- 물리적 대상을 만들지 않았고 물리적 대상의 자동 작성 기능도 *사용 불가* 상태라면 브로커는 물리적 대상을 생성할 수도, 메시지를 전달할 수도 없습니다.

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

➤ 객체 저장소에 대상을 추가하는 방법

1. 탐색 창의 Destinations 노드를 마우스 오른쪽 버튼으로 누릅니다.
2. Add Destination Object를 선택합니다.

   관리 콘솔은 객체에 대한 정보를 지정할 때 사용하는 Add Destination Object 대화 상자를 표시합니다.



3. Lookup Name 필드에 "`MyQueue`"라고 입력합니다.

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

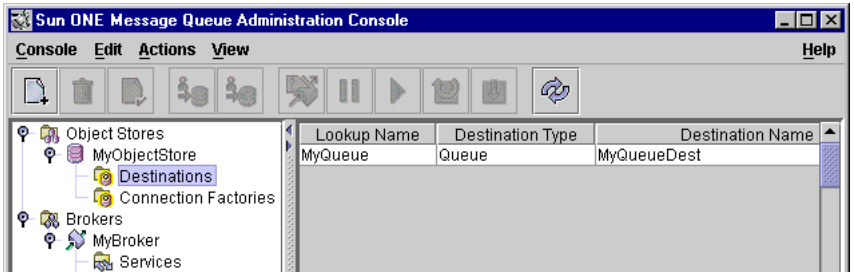
   ```
   queue=(javax.jms.Queue)ctx.lookup("MyQueue");
   ```

4. Destination Type의 Queue 라디오 버튼을 선택합니다.
5. Destination Name 필드에 `MyQueueDest`를 입력합니다.

   이것은 브로커에 물리적 대상을 추가했을 때 지정한 이름입니다.

6. OK를 누릅니다.

7. 탐색 창에서 Destinations를 선택하고 방금 추가한 대기열 대상 관리 객체에 대한 정보가 결과 창에 어떻게 표시되는지 확인합니다.



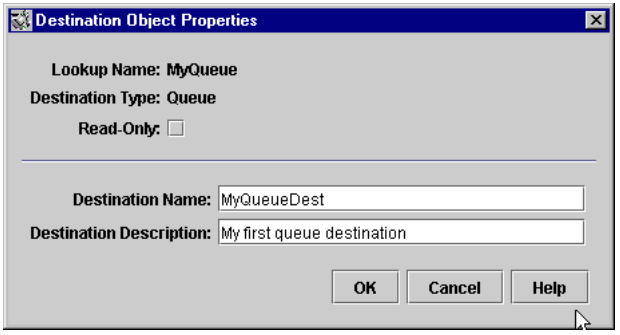
관리 객체 등록 정보

관리 객체의 등록 정보를 확인하거나 업데이트하려면 탐색 창에서 Destinations나 Connection Factories를 선택하고 결과 창에서 특정 객체를 선택한 다음 Actions > Properties를 선택해야 합니다.

➤ **대상 객체의 등록 정보를 확인하거나 업데이트하는 방법**

1. 탐색 창에서 Destinations를 선택합니다.
2. 결과 창에서 MyQueue를 선택합니다.
3. Actions > Properties를 선택하여 Destination Object Properties 대화 상자를 확인합니다.

   변경 가능한 값은 대상 이름과 설명뿐입니다. 조회 이름을 변경하려면 객체를 삭제한 다음 원하는 조회 이름으로 새 대기열 관리 객체를 추가해야 합니다.



콘솔 정보 업데이트

객체 저장소나 브로커로 작업하는 모든 경우에 View > Refresh를 선택하면 어떤 요소나 요소 그룹의 시각적 표시를 업데이트할 수 있습니다.

샘플 응용 프로그램 실행

이 자습서에서 사용할 수 있도록 샘플 응용 프로그램 `SimpleJNDIClient`가 제공됩니다. 이 응용 프로그램은 이 자습서에서 지금까지 생성한 관리 객체 및 대상, 즉 `MyQueueDest`라는 이름의 대기열 대상, `MyQueueConnectionFactory`라는 이름의 대기열 연결 팩토리 관리 객체 그리고 `MyQueue`라는 이름의 대기열 관리 객체를 사용합니다.

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

➤ **`SimpleJNDIClient` 응용 프로그램을 실행하는 방법**

1. `JAVA_HOME` 환경 변수는 JDK가 설치된 디렉토리를 가리켜야 합니다.
2. `SimpleJNDIClient` 응용 프로그램을 포함하는 디렉토리가 현재 디렉토리가 되게 합니다. 예를 들면 다음과 같습니다.

   `cd IMQ_HOME/demo/jms` (Solaris의 경우 `/usr/demo/imq/jms`)

   `SimpleJNDIClient.class` 파일이 존재해야 합니다(응용 프로그램을 변경할 경우, MQ *Developer's Guide*의 Quick Start Tutorial에서 소개하는 클라이언트 응용 프로그램 컴파일 지침을 사용하여 다시 컴파일해야 합니다).
3. `CLASSPATH` 변수가 `SimpleJNDIClient.java`를 비롯하여 `jms.jar`, `imq.jar` 및 `fscontext.jar` 파일이 있는 현재 디렉토리를 포함하도록 설정합니다.
4. 응용 프로그램을 실행하기 전에 `SimpleJNDIClient.java` 소스 파일을 열고 소스를 확인합니다.

   소스는 간단하지만 확실하게 기록되어 있으며, 이 자습서에서 만든 관리 객체 및 대상을 어떻게 사용하는지에 대해 명확하게 설명합니다.

5. SimpleJNDIClient 응용 프로그램을 실행합니다.

```
C:> java SimpleJNDIClient
```
(Windows)

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

응용 프로그램이 성공적으로 실행되면 다음과 같이 출력됩니다.

```
==================================================
$JAVA SimpleJDNIClient file:///tmp
Usage: java SimpleJDNIClinet [Context.PROVIDER_URL]

On Unix:
        java SimpleJDNIClient file:///tmp
On Windows:
        java SimpleJDNIClient file:///C:Temp

Using Windows default file:///C/Temp for Context.PROVIDER_URL

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

5장

브로커 시작 및 구성

Sun™ ONE Message Queue (MQ)를 설치하고 나면 `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 235페이지의 "브로커를 Windows 서비스로 실행"을 참조하십시오.

구성 파일

브로커를 구성할 때 사용되는 설치된 구성 파일은 다음 디렉토리에 있습니다.

`IMQ_HOME/lib/props/broker`
(Solaris의 경우 `/usr/share/lib/imq/props/broker`)

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

또한 브로커를 처음 실행하면 브로커 인스턴스의 구성 등록 정보를 지정할 때 사용할 수 있는 인스턴스 구성 파일이 작성됩니다. 이 파일은 관리 명령어를 사용하여 브로커 인스턴스에 의해 유지 관리되며 주의해서 작업하면 직접 편집할 수도 있습니다. 인스턴스 구성 파일은 다음 위치에 저장됩니다.

`IMQ_VARHOME/instances/`*`brokerName`*`/props/config.properties`
(Solaris의 경우 `/var/imq/instances/`*`brokerName`*`/props/config.properties`)

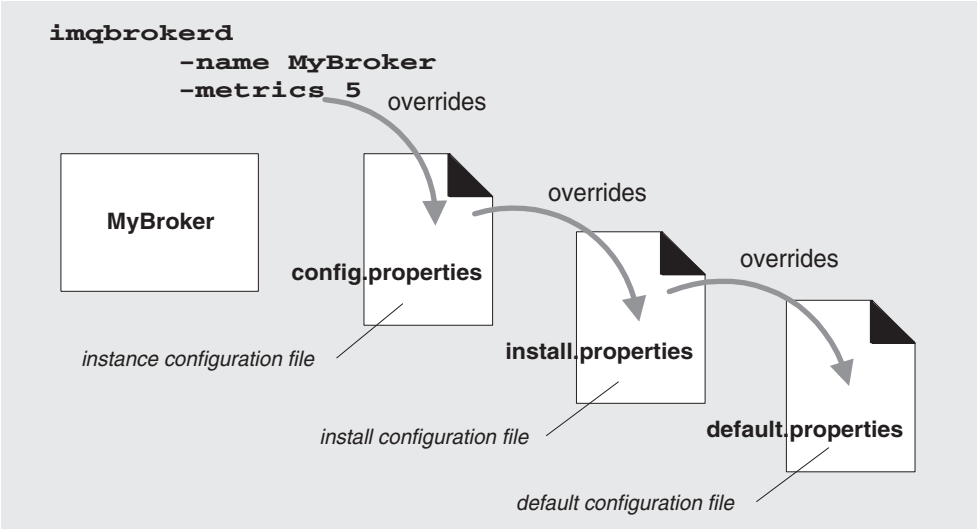
여기서 *brokerName*은 브로커 인스턴스의 이름(기본적으로 `imqbroker`)입니다. 인스턴스 구성 파일을 편집하여 구성을 변경할 수 있습니다(116페이지의 "인스턴스 구성 파일 편집" 참조).

클러스터에서 브로커 인스턴스를 연결하는 경우에는(68페이지의 "다중 브로커 클러스터(엔터프라이즈판)" 참조) *클러스터 구성 파일*을 사용하여 클러스터 구성 정보를 지정해야 할 수도 있습니다. 자세한 내용은 126페이지의 "클러스터 구성 등록 정보"를 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. `imqbrokerd` 명령 옵션을 사용하면 결과 값을 대체할 수 있습니다. 이 체계가 115페이지의 그림 5-1에 표시되어 있습니다.

그림 5-1 브로커 구성 파일



등록 정보 이름 지정 구문

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

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

예를 들어, 다음 항목은 자동 작성 대기열의 대기열 유형을 정의합니다.

```
imq.queue.default=single
```

다음 항목은 메시지 만료 시간 초과 값을 정의합니다.

```
imq.message.expiration.timeout=90
```

116페이지의 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 147페이지의 표 6-5를 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 갖습니다.)

표 5-1에는 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	60페이지의 표 2-6
imq.accesscontrol.file.filename	문자열	accesscontrol.properties	60페이지의 표 2-6
imq.authentication.user_repository	문자열	file	60페이지의 표 2-6
imq.authentication.client.response.timeout	정수 (초)	180	60페이지의 표 2-6
imq.authentication.type	문자열	digest	60페이지의 표 2-6
imq.autocreate.queue	부울	true	67페이지의 표 2-9
imq.autocreate.topic	부울	true	67페이지의 표 2-9
imq.cluster.url	문자열	null	72페이지의 표 2-10
imq.keystore.*property_name*			200페이지의 표 8-8
imq.log.console.output	문자열	ERROR|WARNING	63페이지의 표 2-8
imq.log.console.stream	문자열	ERR	63페이지의 표 2-8

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
imq.log.file.dirpath	문자열	IMQ_VARHOME/instances/*brokerName*/log (Solaris의 경우 /var/imq/...)	63페이지의 표 2-8
imq.log.file.name	문자열	log.txt	63페이지의 표 2-8
imq.log.file.output	문자열	ALL	63페이지의 표 2-8
imq.log.file.rolloverbytes	정수 (바이트)	0	63페이지의 표 2-8
imq.log.file.rolloversecs	정수 (초)	604800	63페이지의 표 2-8
imq.log.level	문자열	INFO	63페이지의 표 2-8
imq.log.syslog.facility	문자열	LOG_DAEMON	63페이지의 표 2-8
imq.log.syslog.logpid	부울	true	63페이지의 표 2-8
imq.log.syslog.logconsole	부울	false	63페이지의 표 2-8
imq.log.syslog.identity	문자열	imqbrokerd_${imq.*brokerName*}	63페이지의 표 2-8
imq.log.syslog.output	문자열	ERROR	63페이지의 표 2-8
imq.message.expiration.interval	정수 (초)	60	53페이지의 표 2-4
imq.message.max_size	바이트 문자열[1] 0 (제한 없음)	70m	53페이지의 표 2-4
imq.metrics.enabled	부울	true	63페이지의 표 2-8
imq.metrics.interval	정수 (초)	0	63페이지의 표 2-8
imq.passfile.enabled	부울	false	60페이지의 표 2-6
imq.passfile.dirpath	문자열	IMQ_HOME/etc (Solaris의 경우/etc/imq)	60페이지의 표 2-6
imq.passfile.name	문자열	passfile	60페이지의 표 2-6
imq.persist.file.destination.file.size	바이트 문자열[1]	1m	56페이지의 표 2-5
imq.persist.file.message.cleanup	부울	false	56페이지의 표 2-5

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
`imq.persist.file.message.fdpool.limit`	정수	`25` (Solaris & Linux) `1024` (Windows)	56페이지의 표 2-5
`imq.persist.file.message.filepool.cleanratio`	정수	`0`	56페이지의 표 2-5
`imq.persist.file.message.filepool.limit`	정수	`10000`	56페이지의 표 2-5
`imq.persist.file.sync.enabled`	부울	`false`	56페이지의 표 2-5
`imq.persist.jdbc.`*`property_name`*			207페이지의 표 A-1
`imq.persist.store`	문자열	`file`	56페이지의 표 2-5
`imq.portmapper.port`	정수	`7676`	49페이지의 표 2-3
`imq.queue.deliverypolicy`	문자열	`single`	67페이지의 표 2-9
`imq.redelivered.optimization`	부울	`true`	53페이지의 표 2-4
`imq.`*`resource_state`*`.threshold`	정수 (퍼센트)	`0` (녹색) `60` (노랑) `75` (주황) `90` (빨강)	53페이지의 표 2-4
`imq.service.activelist`	목록	`jms,admin`	49페이지의 표 2-3
`imq.`*`service_name`*`.accesscontrol.enabled`	부울	시스템 차원 등록 정보에서 값 상속	60페이지의 표 2-6
`imq.`*`service_name`*`.accesscontrol.file.filename`	문자열	시스템 차원 등록 정보에서 값 상속	60페이지의 표 2-6
`imq.`*`service_name`*`.authentication.type`	문자열	시스템 차원 등록 정보에서 값 상속	60페이지의 표 2-6
`imq.`*`service_name`*`.max_threads`	정수	`1000` (jms) `500` (ssljms) `500` (httpjms) `500` (httpsjms) `50` (admin)	49페이지의 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
`imq.`*service_name*`.min_threads`	정수	`10` (jms) `10` (ssljms) `10` (httpjms) `10` (httpsjms) `4` (admin)	49페이지의 표 2-3
`imq.`*service_name*`.`*protocol type*`.hostname`	문자열	`null`	49페이지의 표 2-3
`imq.`*service_name*`.`*protocol type*`.port`	정수	`0`	49페이지의 표 2-3
`imq.`*service_name*`.threadpool_model`	문자열	`dedicated` (jms) `dedicated` (ssljms) `dedicated` (httpjms) `dedicated` (httpsjms) `dedicated` (admin)	49페이지의 표 2-3
`imq.shared.connectionMonitor_limit`	정수	`512` (Solaris & Linux) 64 (Windows)	49페이지의 표 2-3
`imq.system.max_count`	정수, 0 (제한 없음)	`0`	53페이지의 표 2-4
`imq.system.max_size`	바이트 문자열[1], 0 (제한 없음)	`0`	53페이지의 표 2-4
`imq.transaction.autorollback`	부울	`false`	53페이지의 표 2-4
`imq.user_repository.ldap.`*property_name*			189페이지의 표 8-5

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.

브로커 시작

브로커 인스턴스를 시작하려면 `imqbrokerd` 명령을 사용합니다.

주 관리 콘솔(imqadmin)이나 명령 유틸리티(imqcmd)를 사용해서는 브로커 인스턴스를 시작할 수 없습니다. 이 MQ 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` 명령의 구문은 다음과 같습니다(옵션 및 인수는 공백으로 구분).

```
imqbrokerd [[ -Dproperty=value]...]
    [ -backup fileName]
    [ -cluster "[broker] [[,broker]...]"
    [ -dbuser userName] [ -dbpassword password]
    [ -force]
    [ -h]
    [ -javahome path | -jrehome path]
    [ -ldappassword password]
    [ -license name]
    [ -loglevel level]
    [ -metrics number]
    [ -name brokerName ] [ -port number]
    [ -shared]
    [ -password keypassword] [ -passfile fileName]
    [ -remove instance]
    [ -reset data]
    [ -restore fileName]
    [ -shared]
    [ -silent] [ -tty]
    [ -version] [ -vmargs arg [[arg]...]
```

예를 들어, 기본 브로커 이름과 구성을 사용하는 브로커를 시작하려면 다음 명령을 사용합니다.

```
imqbrokerd
```

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

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.

표 5-2에는 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
-backup *fileName*	없음	브로커 클러스터에만 적용됩니다. 마스터 브로커의 구성 변경 기록을 지정된 파일에 백업합니다. 130 페이지의 "마스터 브로커의 구성 변경 기록 백업"을 참조하십시오.
-cluster "[*broker*] [[,*broker*]...]" *broker*는 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. *host*에 값을 지정하지 않으면 localhost가 사용됩니다. *port*에 값을 지정하지 않으면 7676이 사용됩니다. 이 옵션을 사용해서 여러 브로커에 연결하는 방법에 대한 자세한 내용은 126페이지의 "클러스터를 사용한 작업(엔터프라이즈판)"을 참조하십시오.
-dbpassword *password*	imq.persist.jdbc.password를 지정한 비밀번호로 설정합니다.	플러그 인 JDBC 호환 데이터 저장소의 비밀번호를 지정합니다. 부록 A, "플러그 인 지속성 설정"을 참조하십시오.
-dbuser *userName*	imq.persist.jdbc.user를 지정한 사용자 아이디로 설정합니다.	플러그 인 JDBC 호환 데이터베이스의 사용자 아이디를 지정합니다. 부록 A, "플러그 인 지속성 설정"을 참조하십시오.

표 5-2 imqbrokerd 옵션*(계속)*

옵션	영향을 받는 등록 정보	설명
-D*property*=*value*	시스템 등록 정보를 설정합니다. 인스턴스 구성 파일의 해당 등록 정보 값을 대체합니다.	지정한 등록 정보를 지정한 값으로 설정합니다. 브로커 구성 등록 정보는 116페이지의 표 5-1을 참조하십시오. **경고:** D 옵션으로 설정한 등록 정보의 맞춤법과 형식에 주의하십시오. 잘못된 값을 전달한 경우 시스템에서는 경고를 하지 않으며 MQ에서는 해당 값을 설정할 수 없습니다.
-force	없음	사용자의 확인 없이 작업을 수행합니다. 이 옵션은 보통 확인이 필요한 -remove instance 옵션에만 적용됩니다.
-h	없음	도움말을 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
-javahome *path*	없음	대체 Java 2 호환 JDK의 경로를 지정합니다. 기본값은 번들형 런타임을 사용하는 것입니다.
-jrehome *path*	없음	Java 2 JRE의 경로를 지정합니다.
-ldappassword *password*	imq.user_repository.ldap.password를 지정한 비밀번호로 설정합니다.	LDAP 사용자 저장소에 액세스할 때 필요한 비밀번호를 지정합니다. 189페이지의 "사용자 저장소에 LDAP 서버 사용"을 참조하십시오.
-license [*name*]	없음	로드할 사용권이 MQ 제품의 기본값과 다른 경우 해당 사용권을 지정합니다. 사용권 이름을 지정하지 않으면 시스템에 설치된 모든 사용권이 표시됩니다. *name* 값에는 설치된 MQ에 따라 pe (플랫폼판-기본 기능), try (플랫폼판-90일 시험 기업용 기능), unl (엔터프라이즈판) 등이 있습니다. 26페이지의 "제품 판"을 참조하십시오.
-loglevel *level*	imq.broker.log.level을 지정한 수준으로 설정합니다.	로깅 수준을 NONE, ERROR, WARNING 또는 INFO 중 하나로 지정합니다. 기본값은 INFO입니다. 자세한 내용은 61페이지의 "로거"를 참조하십시오.

표 5-2 imqbrokerd 옵션*(계속)*

옵션	영향을 받는 등록 정보	설명
-metrics *int*	imq.metrics.report.interval을 지정한 초 수로 설정합니다.	지정한 초 간격으로 메트릭이 보고되도록 지정합니다.
-name *brokerName*	imq.instancename을 지정한 이름으로 설정합니다.	이 브로커의 인스턴스 이름을 지정하고 해당 인스턴스 구성 파일을 사용합니다. 브로커 이름을 지정하지 않으면 파일 이름은 imqbroker로 설정됩니다. **주**: 같은 호스트에서 두 개 이상의 브로커 인스턴스를 실행하는 경우에는 각각에 고유한 이름이 있어야 합니다.
-passfile *fileName*	imq.passfile.enabled를 true로 설정합니다. jmq.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 204페이지의 "Passfile 사용"을 참조하십시오.
-password *keypassword*	imq.keystore.password를 지정한 비밀번호로 설정합니다.	SSL 인증 키 저장소의 비밀번호를 지정합니다. 자세한 내용은 57페이지의 "보안 관리자"를 참조하십시오.
-port *number*	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 브로커의 포트 매퍼에 서로 다른 포트 번호가 있어야 합니다. JMS 클라이언트는 이 포트 번호를 사용해서 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, 영구 가입, 트랜잭션 정보 등의 모든 지속성 데이터가 지워집니다. 그러면 브로커 인스턴스를 초기 상태로 시작할 수 있습니다. 지속성 메시지만 지우거나 영구 가입만 지울 수도 있습니다. (앞으로 시작할 때 영구 저장소가 재설정되지 않게 하려면 -reset 옵션을 사용하지 않고 브로커 인스턴스를 다시 시작합니다.) 자세한 내용은 54페이지의 "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 기록을 지정한 백업 파일로 교체합니다. 이 파일은 -backup 옵션을 사용해서 미리 작성해 두어야 합니다. 131페이지의 "마스터 브로커의 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 46페이지의 "연결 서비스"를 참조하십시오.
-silent	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.

표 5-2 imqbrokerd 옵션*(계속)*

옵션	영향을 받는 등록 정보	설명
-version	없음	설치된 제품의 버전 번호를 표시합니다.
-vmargs *arg* [[*arg*]...]	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 사용한 작업(엔터프라이즈판)

이 절에서는 여러 브로커의 클러스터를 구성할 때 사용되는 등록 정보, 브로커를 연결하는 두 가지 메소드, 클러스터 관리 방법 등에 대해 설명합니다. 클러스터에 대한 소개는 68 페이지의 "다중 브로커 클러스터(엔터프라이즈판)"를 참조하십시오.

클러스터를 사용해서 작업을 할 때에는 클러스터에 있는 모든 브로커의 호스트 사이에서 시계가 동기화되어 있는지 확인합니다.

클러스터 구성 등록 정보

브로커를 클러스터에 연결할 때에는 연결된 모든 브로커가 클러스터 구성 등록 정보에 대해 같은 값을 지정해야 합니다. 다음 등록 정보는 클러스터의 브로커 참가에 대해 설명합니다. 표 5-3에는 클러스터와 관련된 구성 등록 정보가 요약되어 있습니다.

표 5-3 클러스터 구성 등록 정보

등록 정보	설명
`imq.cluster.brokerlist`	클러스터에 있는 모든 브로커를 쉼표로 구분된 목록에 지정합니다. 각 항목은 브로커의 호스트와 포트를 지정합니다. 예: `host1:3000,host2:8000,ctrhost`
`imq.cluster.masterbroker`	마스터 브로커의 호스트와 포트를 지정합니다. 작업 환경에 대해 이 값을 설정합니다. 예: `ctrhost:7676`
`imq.cluster.url`	클러스터 구성 파일의 위치를 지정합니다. 예: `http://webserver/imq/cluster.properties` `file:/net/mfsserver/imq/cluster.properies`
`imq.cluster.port`	클러스터에 있는 *각* 브로커는 클러스터 연결 서비스를 위한 포트 번호를 지정하는 데 사용할 수 있습니다. 클러스터 연결 서비스는 클러스터 내의 브로커 간 내부 통신에 사용됩니다. 기본값: 0 (포트가 동적으로 할당됨)

표 5-3 클러스터 구성 등록 정보*(계속)*

등록 정보	설명
imq.cluster.hostname	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 브로커 간 내부 통신에 사용됩니다. 기본값: null (사용 가능한 모든 호스트)

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. 이 등록 정보에는 연결할 브로커 목록(imq.cluster.brokerlist), 그리고 선택적으로 마스터 브로커의 주소(imq.cluster.masterbroker)를 포함할 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. host1과 ctrlhost는 모두 기본 포트에서 실행됩니다. 다음 등록 정보는 host1과 ctrlhost가 클러스터에서 연결되고 ctrlhost가 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 url이 포함되어야 합니다.

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

브로커 연결

이 절에서는 브로커를 클러스터에 연결하는 두 가지 방법에 대해 설명합니다. 사용하는 방법에 관계 없이 시작하는 각 브로커는 5초마다 다른 브로커와의 연결을 시도합니다. 이 시도가 성공하면 클러스터에 있는 다른 브로커가 시작됩니다.

브로커를 클러스터에 연결하는 경우 마스터 브로커를 먼저 시작할 필요는 없습니다. 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 없음

➤ **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 host1의 기본 포트에서 실행 중인 브로커와 host2의 포트 7677에서 실행 중인 브로커, localhost의 포트 7678에서 실행 중인 브로커에 연결합니다.

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

방법 2: 클러스터 구성 파일 사용

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

클러스터에 브로커 추가

브로커 클러스터를 설정하고 나면 새 브로커를 추가하거나 이미 클러스터에 속해 있는 브로커를 다시 시작해야 할 수 있습니다.

기존 클러스터에 새 브로커를 추가하려면 다음 중 하나를 수행합니다.

클러스터 구성 파일을 사용하지 않는 경우에는 새 브로커를 시작할 때 명령줄에서 -D 옵션을 사용하여 imq.cluster.brokerlist와 (필요한 경우) imq.cluster.masterbroker 등록 정보를 지정합니다.

➤ **클러스터 구성 파일을 사용하는 경우 브로커를 클러스터에 추가하는 방법**

1. 클러스터 구성 파일의 imq.cluster.brokerlist 등록 정보에 새 브로커를 추가합니다.
2. 클러스터의 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.

클러스터에서 브로커 다시 시작

이미 클러스터에 속해 있는 브로커를 다시 시작하려면 다음 중 하나를 수행합니다.

- 클러스터 구성 파일을 사용해서 클러스터를 정의하는 경우에는 브로커를 시작할 때 사용되는 명령줄의 -D 옵션을 사용하여 imq.cluster.url 등록 정보를 지정합니다.
- 클러스터 구성 파일을 사용해서 클러스터를 정의하지 않은 경우에는 새 브로커를 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 새 브로커를 시작할 때 클러스터에 있는 브로커 목록을 지정할 수 있습니다.

클러스터에서 브로커 제거

클러스터에서 브로커를 제거할 때에는 다음 사항에 주의해야 합니다.

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

  ```
  imqbrokerd -cluster A,B,C
  ```

  대신 다음 명령줄을 사용하여 다른 모든 브로커를 다시 시작해야 합니다.

  ```
  imqbrokerd -cluster B,C
  ```

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

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

마스터 브로커의 구성 변경 기록 백업

각 클러스터는 클러스터에서 영구 상태의 모든 변경 사항(영구 가입, 관리자가 작성한 물리적 대상 포함)을 추적하는 마스터 브로커 하나를 가질 수 있습니다. 모든 브로커는 시작할 때 마스터 브로커를 참조하여 이런 지속성 객체에 대한 정보를 동기화합니다. 따라서 마스터 브로커에 문제가 생기면 클러스터 전체가 손상될 수 있습니다. 따라서 정기적으로 `imqbrokerd` 명령의 `-backup` 옵션을 사용해서 마스터 브로커의 변경 기록을 백업하는 것이 중요합니다. 예를 들면 다음과 같습니다.

```
imqbrokerd -backup mybackuplog
```

이 작업은 적절한 시기에 수행해야 합니다. 너무 오래된 백업 파일을 복원하면 백업을 마지막으로 수행한 후에 작성된 모든 지속성 객체가 손실됩니다.

마스터 브로커의 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 기록에 오래된 데이터가 다시 로드되는 것을 피할 수 없습니다. 하지만 앞 절의 설명과 같이 정기적으로 백업을 해 두면 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 구성에 대해 설명하며, 구성을 변경하여 로그 정보를 대체 출력 채널로 리디렉션하고, 롤오버 기준을 변경하고, 브로커 메트릭을 보고하는 방법에 대해서도 설명합니다. 로깅에 대한 소개는 61페이지의 "로거"를 참조하십시오.

기본 로깅 구성

브로커는 시작 시 로그 출력을 다음 위치의 로그 파일에 저장하도록 자동으로 구성됩니다.

`IMQ_VARHOME/instances/`*brokerName*`/log/`
(Solaris의 경우 `/var/imq/instances/`*brokerName*`/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 `imq.log.file.dirpath` 등록 정보를 원하는 경로로 설정합니다.
- 로그 파일의 기본 이름을 `log`가 아닌 다른 이름으로 변경하려면 `imq.log.file.filename` 등록 정보를 설정합니다.

브로커는 `ERROR`, `WARNING`, `INFO` 등 세 가지 로그 범주를 지원합니다(62페이지의 표 2-7 참조). 로깅 수준을 설정하면 해당 수준 이하의 모든 메시지를 수집합니다. 기본 로그 수준은 `INFO`입니다. 즉, `ERROR`, `WARNING`, `INFO` 메시지가 기록됩니다.

로그 메시지 형식

기록된 메시지는 타임스탬프, 메시지 코드, 메시지 자체로 이루어집니다. 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 63페이지의 표 2-8에 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 5-4에는 로깅에 영향을 주는 `imqbrokerd` 옵션이 나와 있습니다.

표 5-4 `imqbrokerd` 로거 옵션 및 해당 등록 정보

imqbrokerd 옵션	설명
`-metrics` *number*	메트릭 정보를 수집하는 간격(초)을 지정합니다.
`-loglevel` *level*	로그 수준을 `ERROR`, `WARNING`, `INFO` 중 하나로 설정합니다.
`-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
- 브로커 메트릭 정보 기록

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 로그 메시지의 대상을 변경하기 전에, 로깅 수준이 출력 채널에 매핑되는 로그 범주에 맞는 값으로 설정되었는지 확인해야 합니다. 예를 들어, 로그 수준을 `ERROR`로 설정하고 `imq.log.console.output` 등록 정보를 `WARNING`으로 설정한 경우에는 해당 수준 메시지의 로깅을 활성화하지 않았기 때문에 메시지가 기록되지 않습니다.

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 성능 메트릭 로깅

브로커의 기본 구성에는 다음과 같은 설정이 포함됩니다.

- imq.metrics.enabled=true
- imq.metrics.interval=0
- imq.log.level=INFO

이런 설정의 결과 브로커는 활성 연결 서비스뿐 아니라 성능 메트릭에 관한 정보도 수집하게 되지만 메트릭 보고서가 생성되지는 않습니다.

브로커가 메트릭 보고서를 생성하도록 하려면 두 가지 방법 중 하나를 사용합니다.

- imqbrokerd 명령에 -metrics 옵션을 사용하여 브로커가 보고서를 생성하는 간격(초)을 지정합니다.
- imq.metrics.interval 등록 정보를 브로커가 보고서를 생성할 간격(초)으로 설정합니다.

메트릭 보고서가 INFO 범주에 포함되기 때문에, 기본적으로 메트릭 보고서는 로그 파일 출력 채널에 기록됩니다.

다음에는 샘플 메트릭 정보가 나와 있습니다.

```
[31/Jan/2001:15:00:50 PST]
Connections: 0 JVM Heap: 6291456 bytes (5186320 free)
       In: 0 mesgs (0bytes) 0 pkts (0 bytes)
      Out: 0 mesgs (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)
```

표 5-5에서는 각 연결 서비스에 대해 생성된 메트릭의 의미에 대해 설명합니다.

표 5-5 연결 서비스에 대해 수집한 메트릭

메트릭	설명
Pkts in (전체)	마지막으로 재설정한 후 브로커에서 읽은 전체 패킷 수입니다. 여기에는 JMS 메시지뿐 아니라 MQ 프로토콜 패킷도 포함됩니다.
Pkts out (전체)	마지막으로 재설정한 후 브로커에서 쓴 전체 패킷 수입니다.
JMS Messages in (전체)	마지막으로 재설정한 후 브로커에서 읽은 전체 JMS 메시지 수입니다.
JMS Messages out (전체)	마지막으로 재설정한 후 브로커에서 쓴 전체 JMS 메시지 수입니다.
Message Bytes in (전체)	마지막으로 재설정한 후 브로커에서 읽은 전체 메시지 바이트 수입니다.
Message Bytes out (전체)	쓴 전체 메시지 바이트 수입니다.
Current # connections	현재 열린 연결의 수입니다.

표 5-6에서는 각 브로커에 대해 수집 및 보고되는 메트릭을 설명합니다.

표 5-6 각 브로커에 대해 수집한 메트릭

메트릭	설명
VM heap size (바이트)	Java VM 힙의 최대 크기입니다.
VM heap free spave (바이트)	Java VM 힙에서 사용할 수 있는 남은 공간의 크기입니다.

주 이 정보는 `imqcmd metrics` 명령을 통해서도 얻을 수 있습니다.

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 쿼리 및 업데이트
- 자원 할당 및 관리
- 연결 서비스 관리

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

- MQ 메시지는 브로커 대상을 통해 수신기 또는 가입자에게 라우팅됩니다. 브로커에서 이러한 대상을 만드는 것은 관리자의 책임입니다.
- MQ는 영구 가입이 있는 클라이언트가 비활성화되어 있는 동안에도 영구 가입자의 자원을 할당 및 관리합니다. 영구 가입에 대한 정보를 얻고 영구 가입을 완전 삭제하거나 메시지를 제거하여 MQ 자원을 확보하려면 MQ 명령 도구를 사용합니다.
- MQ 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. MQ 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.

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)에는 표 6-1에 나열된 하위 명령이 포함되어 있습니다.

표 6-1 imqcmd 하위 명령

하위 명령 및 인수	설명
create dst	대상을 만듭니다.
commit txn	트랜잭션을 완결합니다.
destroy dst	대상을 완전 삭제합니다.
destroy dur	영구 가입을 완전 삭제합니다.
list dst	브로커의 대상을 나열합니다.
list dur	주제에 해당하는 영구 가입을 나열합니다.
list svc	브로커의 서비스를 나열합니다.
list txn	브로커의 트랜잭션을 나열합니다.
metrics bkr	브로커 메트릭을 표시합니다.
metrics svc	서비스 메트릭을 표시합니다.

표 6-1 imqcmd 하위 명령(*계속*)

하위 명령 및 인수	설명
pause bkr	브로커의 모든 서비스를 일시 중지합니다.
pause svc	한 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 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 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 *hostName:port*	브로커의 호스트 이름과 해당 포트 번호를 지정합니다. 기본값은 localhost:7676입니다. 포트만 지정하려면: -b :7878 이름만 지정하려면: -b somehost
-c *clientID*	주제에 영구 가입의 아이디를 지정합니다. 자세한 내용은 155페이지의 "영구 가입 관리"를 참조하십시오.
-d *topicName*	주제의 이름을 지정합니다. list dur 및 destroy dur 하위 명령에 사용됩니다. 155페이지의 "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 *interval*	imqcmd에서 브로커 메트릭을 표시하는 간격을 초 단위로 지정합니다. (metrics 하위 명령에 사용됩니다.)
-javahome	Java 2와 호환할 수 있는 대체 런타임을 지정하여 사용합니다(기본값은 제품과 함께 제공되는 런타임 사용).
-m *metricType*	표시할 메트릭 정보의 유형을 지정합니다. 유형은 다음 중 하나가 될 수 있습니다. ttl 브로커 내부 및 외부의 모든 메시지(기본값) rts ttl과 같은 정보를 제공하지만 초당 메시지 수를 지정합니다. cxn 연결, 가상 메모리 힙, 스레드 이 옵션은 metrics bkr 또는 metrics svc 하위 명령에 사용됩니다. 다음 명령은 기본 브로커에 대해 5초마다 cxn 유형의 메트릭을 표시합니다. imqcmd metrics bkr -m cxn -int 5

표 6-2 imqcmd 옵션*(계속)*

옵션	설명
-n *argumentName*	하위 명령 인수의 이름을 지정합니다. 하위 명령에 따라 서비스나 물리적 대상, 영구 가입의 이름일 수도 있고 트랜잭션 아이디일 수도 있습니다.
-o *attribute=value*	속성의 값을 지정합니다. 하위 명령 인수에 따라 브로커(145페이지의 "브로커 등록 정보 쿼리 및 업데이트" 참조), 서비스(148페이지의 "연결 서비스 관리" 참조) 또는 대상(152페이지의 "대상 관리" 참조)의 속성이 될 수 있습니다.
-secure	ssladmin 연결 서비스를 사용해서 브로커에 대한 보안 관리 연결을 지정합니다(202페이지의 "명령 유틸리티(imqcmd)"에서 "4단계. SSL기반 클라이언트 구성 및 실행" 참조).
-p *password*	자신(관리자)의 비밀번호를 지정합니다. 이 값을 생략하면 값을 묻는 메시지가 표시됩니다.
-s	비대화형 모드입니다. 출력이 표시되지 않습니다.
-t *destinationType*	대상의 유형을 지정합니다. t (주제) 또는 q (대기열).
-tmp	임시 대상을 표시합니다.
-u *name*	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), 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.

imqcmd 사용의 전제 조건

imqcmd 명령을 사용해서 브로커를 관리하려면 다음을 수행해야 합니다.

- imqbrokerd 명령을 사용해서 브로커를 시작합니다.

  120페이지의 "브로커 시작"를 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.

  MQ를 설치하면 기본 플랫 파일 사용자 저장소가 설치됩니다. 파일의 이름은 IMQ_HOME/etc/passwd (Solaris의 경우 /etc/imq/passwd)입니다. 저장소에는 admin 사용자와 guest 사용자 각각에 대해 하나씩 두 개의 항목이 있습니다. 이러한 항목을 통해 추가 작업을 하지 않고도 브로커에 연결할 수 있습니다. 예를 들어, MQ를 테스트만 하는 경우에는 기본 사용자 아이디와 비밀번호(admin/admin)를 사용해서 유틸리티를 실행할 수 있습니다.

  작업 시스템을 설정하는 경우에는 추가 작업을 통해 사용자를 인증하고 권한을 부여해야 합니다. 사용자 저장소에 기존 LDAP 디렉토리 서버를 사용할 수도 있습니다. 자세한 내용은 182페이지의 "사용자 인증"을 참조하십시오.

예

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.

브로커 상태 제어

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

- 브로커 일시 중지

  브로커를 일시 중지하면 브로커의 서비스 스레드가 지연되고 브로커에서는 포트 수신을 중지합니다. 그리고 나면 브로커에 대한 메시지의 흐름을 규제하는 관리 작업을 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.

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

  ```
  imqcmd pause bkr -b myhost:1588
  ```

- 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
  ```

- 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
  ```

- 브로커를 처음부터 다시 시작

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

  ```
  imqcmd restart bkr
  ```

표 6-3에서는 브로커의 제어에 사용되는 imqcmd 하위 명령을 요약합니다. localhost의 포트 7676에서 실행 중인 브로커가 대상이 아닌 경우에는 브로커 호스트 이름과 포트 번호를 지정해야 합니다.

표 6-3 브로커 제어에 사용되는 imqcmd 하위 명령

하위 명령	설명
`pause bkr [-b` *hostName:port*`]`	기본 브로커 또는 지정한 호스트 및 포트의 브로커를 일시 중지합니다.
`resume bkr [-b` *hostName:port*`]`	기본 브로커 또는 지정한 호스트 및 포트의 브로커를 다시 시작합니다.
`shutdown bkr [-b` *hostName:port*`]`	기본 브로커 또는 지정한 호스트 및 포트의 브로커를 종료합니다.
`restart bkr [-b` *hostName:port*`]`	기본 브로커 또는 지정한 호스트 및 포트의 브로커를 종료한 후 처음부터 다시 시작합니다. 이 명령은 브로커를 처음 시작했을 때 지정한 옵션을 사용해서 브로커를 다시 시작합니다. 다른 옵션을 적용하려면 브로커를 종료한 후 원하는 옵션을 지정하여 다시 시작해야 합니다.

브로커 등록 정보 쿼리 및 업데이트

명령 유틸리티에는 브로커에 대한 정보를 얻고 브로커 등록 정보를 업데이트할 때 사용할 수 있는 하위 명령이 포함되어 있습니다. 표 6-4에 이러한 하위 명령이 나열되어 있습니다.

표 6-4 정보를 얻고 브로커를 업데이트할 때 사용되는 imqcmd 하위 명령

하위 명령 구문	설명
`query bkr -b` *hostName:port*	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.
`reload cls`	브로커 클러스터에만 적용됩니다. 클러스터에 있는 모든 브로커에서 `imq.cluster.brokerlist` 등록 정보를 다시 로드하고 클러스터 정보를 업데이트하도록 합니다. 자세한 내용은 129페이지의 "클러스터에 브로커 추가"를 참조하십시오.

`update bkr [-b` *hostName:port*`]`
`-o` *attribute=value*
`[-o` *attribute=value1*`]...`
	기본 브로커 또는 지정한 호스트 및 포트의 브로커에 해당하는 지정된 속성을 변경합니다.

`metrics bkr [-b` *hostName:port*`]`
`[-m` *metricType*`]`
`[-int` *interval*`]`
	기본 브로커 또는 지정한 호스트 및 포트의 브로커 메트릭을 표시합니다.
	-m 옵션을 사용하여 표시할 메트릭 유형을 지정합니다.
	`ttl` 브로커 내부 및 외부의 모든 메시지(기본값)
	`rts` ttl과 같은 정보를 제공하지만 초당 메시지의 수를 지정합니다.
	`cxn` 연결, 가상 메모리 힙, 스레드
	-int 옵션을 사용하여 메트릭 표시 간격(초)을 지정합니다. 기본값은 5초입니다.

localhost의 포트 7676에서 실행 중인 브로커가 대상이 아닌 경우에는 표 6-4에 나열된 하위 명령을 사용할 때 브로커 호스트 이름과 포트 번호를 지정해야 합니다.

브로커 쿼리

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

```
imqcmd query bkr
```

다음과 같은 출력이 표시됩니다.

```
%imqcmd query bkr

Querying the broker specified by:

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

Auto Create Queues                      true
Auto Create Topics                      true
Auto Created Queue Delivery Policy      Round Robin
Cluster Broker List (active)            myhost/192.18.116.221:7676
Cluster Broker List (configured)
Cluster Master Broker
Cluster URL
Current Number of Messages in System    0
Current Size of Messages in System      0
Instance Name                           imqbroker
Log Level                               INFO
Log Rollover Interval (seconds)         604800
Log Rollover Size (bytes)               0 (unlimited)
Max Message Size                        70m
Max Number of Messages in System        0 (unlimited)
Max Size of Messages in System          0 (unlimited)
Primary Port                            7676
Version                                 3.0

Successfully queried the broker.
```

브로커 업데이트

update 하위 명령을 사용하면 표 6-5에 나열된 브로커 등록 정보를 업데이트할 수 있습니다. 브로커에서 업데이트된 사항은 브로커의 인스턴스 구성 파일에 자동으로 기록됩니다.

표 6-5 브로커 등록 정보

등록 정보	설명
imq.autocreate.queue	브로커에 대기열 대상 자동 작성이 허용되는지 여부를 지정합니다. 기본값은 True입니다.
imq.autocreate.topic	브로커에 주제 대상의 자동 작성이 허용되는지 여부를 지정합니다. 기본값은 True입니다.
imq.queue.deliverypolicy	자동 작성된 대기열의 기본 전달 정책을 지정합니다. 값은 s(단일), r(라운드 로빈) 또는 f(페일오버)입니다. 기본값은 s입니다.
imq.cluster.url	클러스터 구성 파일의 위치를 지정합니다. 자세한 내용은 126페이지의 "클러스터 구성 등록 정보"를 참조하십시오.
imq.log.level	로그 수준을 NONE, ERROR, WARNING, INFO 중 하나로 지정합니다. 기본값은 INFO입니다.
imq.log.file.rolloversecs	로그 파일을 롤오버할 때까지의 시간(초)입니다. 값 0은 파일 시간을 기준으로 한 롤오버가 없음을 의미합니다. 기본값은 604800 (7일)입니다.
imq.log.file.rolloverbytes	로그 파일을 롤오버하기 전에 최대 로그 파일 크기를 지정합니다. 값 0은 파일 크기를 기준으로 한 롤오버가 없음을 의미합니다. 기본값은 0입니다.
imq.message.max_size	메시지의 최대 크기를 바이트 단위로 지정합니다. 기본값은 70m입니다.
imq.system.max_count	메모리와 디스크의 최대 메시지 수를 지정합니다. 값 0은 제한이 없음을 의미합니다. 기본값은 0입니다.
imq.system.max_size	메모리와 디스크의 최대 전체 메시지 크기를 지정합니다. 값 0은 제한이 없음을 의미합니다. 기본값은 0입니다.
imq.portmapper.port	포트 매퍼 포트의 번호를 지정합니다. 기본값은 7676입니다.

예를 들어, 다음 명령은 대기열의 기본 전달 정책을 단일에서 라운드 로빈으로 변경합니다.

```
imqcmd update bkr -o "imq.queue.deliverypolicy=r"
```

연결 서비스 관리

명령 유틸리티에는 다음을 수행할 수 있는 여러 하위 명령이 포함되어 있습니다.

- 사용 가능한 연결 서비스를 나열합니다.
- 특정 서비스에 대한 정보를 표시합니다.
- 서비스의 속성을 업데이트합니다.
- 서비스를 일시 중지하고 다시 시작합니다.

MQ 연결 서비스에 대한 개요는 46페이지의 "연결 서비스"를 참조하십시오.

표 6-6에 연결 서비스를 제어하는 `imqcmd` 하위 명령이 나열되어 있습니다. 호스트 이름 또는 포트가 지정되어 있지 않으면 `localhost`, `7676`인 것으로 가정합니다.

표 6-6 연결 서비스 관리에 사용되는 `imqcmd` 하위 명령

하위 명령 구문	설명
`list svc [-b` *hostName:port*`]`	기본 브로커 또는 지정한 호스트 및 포트의 브로커에 있는 모든 연결 서비스를 나열합니다.

`metrics svc -n` *serviceName*
`[-b` *hostName:port*`]`
`[-m` *metricType*`] [-int` *interval*`]`
	기본 브로커 또는 지정한 호스트 및 포트의 브로커에서 지정된 서비스의 메트릭을 나열합니다.
	-m 옵션을 사용하여 표시할 메트릭 유형을 지정합니다.
	`ttl` 브로커 내부 및 외부의 모든 메시지(기본값)
	`rts` ttl과 같은 정보를 제공하지만 초당 메시지 수를 지정합니다.
	`cxn` 연결, 가상 메모리 힙, 스레드
	-int 옵션을 사용하여 메트릭 표시 간격(초)을 지정합니다. 기본값은 5초입니다.

표 6-6 연결 서비스 관리에 사용되는 imqcmd 하위 명령*(계속)*

하위 명령 구문	설명
`query svc -n` *serviceName* `[-b` *hostName:port*`]`	기본 브로커 또는 지정한 호스트 및 포트의 서비스에서 실행 중인, 지정된 서비스에 대한 정보를 표시합니다.
`pause svc -n` *serviceName* `[-b` *hostName:port*`]`	기본 브로커 또는 지정한 호스트 및 포트의 브로커에서 실행 중인, 지정된 서비스를 일시 중지합니다. 관리 서비스는 일시 중지할 수 없습니다.
`resume svc -n` *serviceName* `[-b` *hostName:port*`]`	기본 브로커 또는 지정한 호스트 및 포트에서 실행 중인, 지정된 서비스를 다시 시작합니다.
`update svc -n` *serviceName* `[-b` *hostName:port*`]` `-o` *attribute=value* `[-o` *attribute=value1*`] . . .`	기본 브로커 또는 지정한 호스트 및 포트의 브로커에서 실행 중인, 지정된 서비스의 지정된 속성을 업데이트합니다. 서비스 속성에 대한 설명은 151페이지의 표 6-8을 참조하십시오.

브로커는 응용 프로그램 클라이언트 및 관리 클라이언트 모두와의 통신을 지원합니다. 현재 MQ 브로커에서 사용할 수 있는 연결 서비스가 149페이지의 표 6-7에 표시되어 있습니다. 서비스 이름 열에 있는 값은 -n 옵션의 서비스 이름을 지정할 때 사용되는 값입니다. (표에 있는 것과 같이 각 서비스는 NORMAL (JMS) 또는 ADMIN 중에서 사용하는 서비스 유형과 전송 계층에 따라 지정됩니다.)

표 6-7 브로커에 지원되는 연결 서비스

서비스 이름	서비스 유형	프로토콜 유형
jms	NORMAL (JMS 메시지 전달)	tcp
ssljms (엔터프라이즈판)	NORMAL (JMS 메시지 전달)	tls (SSL 기반 보안)
httpjms (엔터프라이즈판)	NORMAL (JMS 메시지 전달)	http
httpsjms (엔터프라이즈판)	NORMAL (JMS 메시지 전달)	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]
```

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

```
imqcmd list svc -b MyServer:6565
```

다음 명령은 `localhost`의 포트 7676에서 실행 중인 브로커의 모든 서비스를 나열합니다.

```
imqcmd list svc
```

이 명령은 다음과 같은 정보를 출력합니다.

```
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          dynamic            UNKNOWN
ssljms            dynamic            UNKNOWN

Successfully listed services.
```

서비스 등록 정보 쿼리 및 업데이트

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

```
imqcmd query svc -n jms
```

다음과 같은 출력이 표시됩니다.

```
Querying the service where:

Service Name
jms

On the broker specified by:

Host                          Primary Port
localhost                     7676


Current Number of Allocated Threads    120
Current Number of Connections          20
Max Number of Threads                  1000
Min Number of Threads                  50
Port Number                            42019 (dynamic)
Service Name                           jms
Service State                          RUNNING

Successfully queried the service.
```

update 하위 명령을 사용해서 표 6-8에 나열된 서비스 속성 중 하나 이상을 변경할 수 있습니다.

표 6-8 연결 서비스 속성

속성	설명
port	업데이트할 서비스에 할당된 포트입니다(httpjms 또는 httpsjms에는 적용되지 않음).
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"
```

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
```

서비스를 다시 시작하려면 다음과 같은 명령을 사용합니다.

```
imqcmd resume svc -n serviceName
```

대상 관리

모든 MQ 메시지는 특정 브로커에 작성되는 대상, 대기열 및 주제를 통해 사용자 클라이언트로 라우팅됩니다. 브로커에서 이러한 대상을 관리하는 것은 관리자의 책임입니다. 여기에는 명령 유틸리티를 사용해서 대상을 만들고 완전 삭제하고, 대상을 나열하고, 대상 관련 정보를 표시하고, 메시지를 제거하는 작업이 해당됩니다. 대상에 대한 소개는 65페이지의 "물리적 대상"을 참조하십시오.

표 6-9에서는 `imqcmd` 대상 하위 명령을 요약합니다. 기본(`localhost:7676`) 브로커가 아닌 경우 브로커의 호스트 이름과 포트를 지정해야 합니다.

표 6-9 대상 관리에 사용되는 `imqcmd` 하위 명령

하위 명령	설명
`list dst [-tmp]`	대상을 모두 나열합니다. 임시 대상을 함께 나열할 수도 있습니다(68페이지의 "임시 대상" 참조).

`create dst -t` *type*
`-n` *destName*
`[-o` *attribute=value*`]`
`[-o` *attribute=value*`1]...`
	지정한 유형의 대상을 지정한 이름과 지정한 속성을 사용하여 만듭니다. 대상 이름은 영문자와 숫자만 포함할 수 있으며(공백 없음) 영문자 또는 "_" 문자로 시작할 수 있습니다.

`destroy dst -t` *type* `-n` *destName*	지정한 유형과 이름의 대상을 완전 삭제합니다.
`purge dst -t` *type* `-n` *destName*	지정한 유형과 이름의 대상에서 메시지를 제거합니다.
`query dst -t` *type* `-n` *destName*	지정한 유형과 이름의 대상에 대한 정보를 나열합니다.

표 6-9 대상 관리에 사용되는 imqcmd 하위 명령(*계속*)

하위 명령	설명
`update dst -t` *type* `-n` *destName* `-o` *attribute=value* `[-o` *attribute=value1*`] . . .`	지정한 대상에서 지정한 속성의 값을 업데이트합니다. 속성 이름에는 표 6-10에 설명된 속성을 사용할 수 있습니다.

대상 만들기

대상을 만들 때에는 유형(주제 또는 대기열)을 지정하고, 필요한 경우 대상의 속성 값을 지정해야 합니다. 이 속성의 기본값은 브로커의 구성 파일(114페이지의 "구성 파일" 참조)에서 설정합니다.

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

표 6-10에서는 대상을 만들 때 각 대상 유형에 설정할 수 있는 속성에 대해 설명합니다.

표 6-10 대상 속성

대상 유형	속성	기본값	설명
대기열	`queueDeliveryPolicy`	`Single`	메시지 라우팅에 사용되는 알고리즘을 설명합니다. 값은 다음과 같습니다. f = 페일오버 r = 라운드 로빈 s = 단일
대기열	`maxTotalMsgBytes`	0 (제한 없음)	대기열에 허용되는 메시지의 최대 전체 크기를 바이트 수로 나타낸 값입니다.
대기열	`maxNumMsgs`	0 (제한 없음)	대기열에 허용되는 최대 메시지 수입니다.
대기열	`maxBytesPerMsg`	0 (제한 없음)	대기열에 허용되는 단일 메시지의 최대 크기입니다.
주제	`maxBytesPerMsg`	0 (제한 없음)	주제에 게시된 단일 메시지의 최대 크기입니다.

- 대기열 대상을 만들려면 다음과 같이 명령을 입력합니다.

  ```
  imqcmd create dst -n myQueue -t q -o "queueDeliveryPolicy=f"
  ```

  대상 이름은 유효한 Java 식별자여야 합니다.
- 주제 대상을 만들려면 다음과 같이 명령을 입력합니다.

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

대상 관련 정보 얻기

대상 속성의 현재 값에 대한 정보를 얻으려면 다음과 같은 명령을 사용합니다.

```
imqcmd query dst -t q -n XQueue
```

그런 다음 `update imqcmd` 하위 명령을 사용해서 하나 이상의 속성 값을 변경할 수 있습니다.

`myHost`의 포트 4545에서 실행 중인 브로커와 같은 특정 브로커의 모든 대상을 나열하려면 다음과 같은 명령을 사용합니다.

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

선택에 따라 `list` 명령에 임시 대상을 포함할 수도 있습니다(`-tmp` 옵션 사용). 이러한 임시 대상은 다른 클라이언트로 보낸 메시지의 답장을 받는 대상이 필요한 클라이언트 응용 프로그램에서 만듭니다(68페이지의 "임시 대상" 참조). 이 대상은 사용자가 완전 삭제할 수 없습니다. 이 대상은 활성 메시지 사용자가 더 이상 없을 때 클라이언트 응용 프로그램에서 수행한 API 호출을 통해서만 완전 삭제할 수 있습니다.

대상 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 `1000`으로 변경하고 `MaxNumMsgs` 속성을 `2000`으로 변경합니다.

```
imqcmd update dst -t q -n myQueue -o "maxBytesPerMsg=1000"
                  -o maxNumMsgs=2000
```

업데이트할 수 있는 속성의 목록을 보려면 153페이지의 표 6-10을 참조하십시오.

`update dst` 하위 명령을 사용해서 대상의 *유형*을 업데이트하거나 대기열의 대기열 전달 정책을 업데이트할 수 없습니다.

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 다음과 같이 명령을 입력합니다.

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

```
imqcmd purge dst -n myTopic -t t
```

브로커를 종료했다가 다시 시작할 때 오래된 메시지가 전달되지 않게 하려면 `imqbrokerd` 명령의 `reset` 하위 명령을 사용해서 오래된 메시지를 제거합니다. 예를 들면 다음과 같습니다.

```
imqbrokerd -reset messages
```

그러면 브로커를 다시 시작한 후에 대상을 제거할 필요가 없습니다.

대상 완전 삭제

대상을 완전 삭제하려면 다음과 같이 명령을 입력합니다.

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

영구 가입 관리

`imqcmd` 하위 명령을 사용해서 브로커의 영구 가입을 관리할 수 있습니다. *영구 가입*은 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 6-11 영구 가입 관리에 사용되는 imqcmd 하위 명령

하위 명령	설명
`list dur -d` *destination*	지정한 대상의 모든 영구 가입을 나열합니다.
`destroy dur -n` *subscrName* `-c` *client_id*	지정한 클라이언트 식별자에 해당하는 지정된 영구 가입을 완전 삭제합니다(38페이지의 "클라이언트 식별자" 참조).
`purge dur -n` *subscrName* `-c` *client_id*	지정한 클라이언트 식별자에 해당하는 모든 메시지를 제거합니다(38페이지의 "클라이언트 식별자" 참조).

예를 들어, 다음 명령은 SPQuotes 주제의 모든 영구 가입을 나열합니다.

```
imqcmd list dur -d SPQuotes
```

list 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.

```
Listing all the durable subscriptions on the topic myTopic
on the broker specified by:
-------------------------
Host          Primary Port
-------------------------
localhost     7676

Name         Client ID       Number of    Durable Sub
                             Messages       State
------------------------------------------------------------------
myDurable    myClientID       1            INACTIVE

Successfully listed durable subscriptions.
```

list 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 MQ 트랜잭션일 수도 있고 XA 자원 관리자가 관리하는 분산 트랜잭션일 수도 있습니다(40페이지의 "로컬 트랜잭션" 참조). 모든 트랜잭션에는 브로커의 트랜잭션을 고유하게 나타내는 64비트 숫자, 즉 MQ 트랜잭션 아이디가 있습니다. 분산 트랜잭션에도 분산 트랜잭션 관리자가 할당하는 최대 128 바이트의 분산 트랜잭션 아이디(XID)가 있습니다. MQ에서는 MQ 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 모니터하고 롤백 또는 완결해야 합니다.

표 6-12에서는 imqcmd 트랜잭션 하위 명령을 요약합니다. 기본(localhost:7676) 브로커가 아닌 경우 브로커의 호스트 이름과 포트를 지정해야 합니다.

표 6-12 트랜잭션 관리에 사용되는 imqcmd 하위 명령

하위 명령	설명
`list txn`	브로커에서 추적하는 모든 트랜잭션을 나열합니다.
`query txn -n` *transaction_id*	지정한 트랜잭션에 대한 정보를 나열합니다.
`commit txn -n` *transaction_id*	지정한 트랜잭션을 완결합니다.
`rollback txn -n` *transaction_id*	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.

```
Listing all the transactions on the broker specified by:
-------------------------
Host          Primary Port
-------------------------
localhost     7676

------------------------------------------------------------
Transaction ID  State     User name    # Msgs/    Creation time
                                       # Acks
------------------------------------------------------------

64248349708800  PREPARED  guest        4/0        1/30/02 10:08:31 AM
64248371287808  PREPARED  guest        0/4        1/30/02 10:09:55 AM

Successfully listed transactions.
```

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 false로 설정되어 있으면(53페이지의 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 같이 출력됩니다.

```
Querying the transaction where:
-------------------------
Transaction ID
-------------------------
64248349708800

On the broker specified by:

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


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

Successfully queried the transaction.
```

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 53페이지의 표 2-4의 imq.transaction.autorollback 등록 정보를 참조하십시오.

7장

관리 객체 관리

관리 객체를 사용하면 다른 JMS 공급자에 이식할 수 있는 클라이언트 응용 프로그램을 개발할 수 있습니다. *관리 객체*는 공급자별 구성 및 이름 지정 정보를 캡슐화하는 객체입니다. 이 객체는 보통 Sun™ ONE Message Queue (MQ) 관리자가 만들며, 클라이언트 응용 프로그램에서 물리적 대상에 대해 메시지를 보내거나 받을 때 사용되는 브로커에 연결을 설정할 때 사용됩니다.

관리 객체에 대한 개요는 75페이지의 "MQ 관리 객체"를 참조하십시오.

MQ에는 관리 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 객체를 추가 또는 삭제합니다.
- 기존 관리 객체를 나열합니다.
- 관리 객체 관련 정보를 쿼리 및 표시합니다.
- 객체 저장소에 있는 기존 관리 객체를 수정합니다.

이 장에서는 객체 관리자 유틸리티(`imqobjmgr`)를 사용해서 이런 작업을 수행하는 방법을 설명합니다. 관리 콘솔에 대한 자세한 내용은 4장, "관리 콘솔 자습서"를 참조하십시오.

객체 저장소 정보

관리 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 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. MQ 관리 도구는 LDAP 서버에 사용할 수 있도록 디자인되었습니다.

파일 시스템 저장소 MQ에서는 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. 해당 디렉토리에 액세스할 수 있는 모든 사용자는 MQ 관리 도구를 사용하여 관리 객체를 만들고 관리할 수 있습니다.

관리 객체

관리 객체에 대한 개요는 75페이지의 "MQ 관리 객체"를 참조하십시오.

MQ 관리 객체에는 기본적으로 연결 팩토리와 대상의 두 가지 종류가 있습니다. *연결 팩토리* 관리 객체는 클라이언트 응용 프로그램에서 브로커에 대한 연결을 만들 때 사용됩니다. *대상* 관리 객체는 클라이언트 응용 프로그램에서 생성자가 메시지를 보내거나 사용자가 메시지를 받는 대상을 나타낼 때 사용됩니다. (SOAP 메시징에서는 특수한 *SOAP 종점* 관리 객체를 사용합니다. 자세한 내용은 MQ *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 객체 유형도 사용할 수 있으며, 분산 트랜잭션을 지원하는 연결 팩토리 유형도 마찬가지입니다(지원되는 모든 유형은 38페이지의 표 1-1 참조).

객체 관리자 유틸리티(imqobjmgr)

객체 관리자 유틸리티를 사용하면 MQ 관리 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.

명령 구문

imqcmd 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 7-1에 나열된 하위 명령이 포함되어 있습니다.

표 7-1 imqobjmgr 하위 명령

하위 명령	설명
add	객체 저장소에 관리 객체를 추가합니다.
delete	객체 저장소에서 관리 객체를 삭제합니다.
list	객체 저장소에 있는 관리 객체를 나열합니다.
query	지정한 관리 객체에 대한 정보를 표시합니다.
update	객체 저장소에 있는 기존 관리 객체를 수정합니다.

imqobjmgr 명령 옵션 요약

표 7-2에는 `imqobjmgr` 명령의 옵션이 나열되어 있습니다. 사용 설명은 다음에서 해당 작업 기반 절을 참조하십시오.

표 7-2 `imqobjmgr` 옵션

옵션	설명
`-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.
`-i` *fileName*	하위 명령 절의 일부 또는 전부를 포함하는 입력 파일 이름을 지정하며, 객체 유형, 조회 이름, 객체 속성, 객체 저장소 속성 또는 기타 옵션 등을 지정합니다. 보통은 객체 저장소 속성과 같은 반복 정보에 사용됩니다.
`-j` *attribute=value*	JNDI 객체 저장소를 확인하고 액세스할 때 필요한 속성을 지정합니다.
`-javahome`	Java 2와 호환할 수 있는 대체 런타임을 지정하여 사용합니다(기본값은 제품과 함께 제공되는 런타임 사용).
`-l` *lookupName*	관리 객체의 JNDI 조회 이름을 지정합니다. 이 이름은 관리 객체 저장소의 컨텍스트에서 고유해야 합니다.
`-o` *attribute=value*	관리 객체의 속성을 지정합니다.
`-pre`	미리 보기 모드입니다. 명령을 수행하지 않고 수행 결과를 확인합니다.
`-r` *read-only_state*	관리 객체가 읽기 전용인지 여부를 지정합니다. `true` 값은 관리 객체가 읽기 전용임을 나타냅니다. JMS 클라이언트는 읽기 전용 관리 객체의 속성을 수정할 수 없습니다. 읽기 전용 상태는 기본적으로 `false`로 설정됩니다.
`-s`	비대화형 모드입니다. 출력이 표시되지 않습니다.

표 7-2 imqobjmgr 옵션*(계속)*

옵션	설명
-t *type*	MQ 관리 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 객체 유형은 SOAP 메시지 지원에 사용됩니다(MQ *Developer's Guide* 참조).

다음 절에서는 imqobjmgr 하위 명령을 사용해서 작업할 때 제공해야 하는 정보에 대해 설명합니다.

필요한 정보

관리 객체와 관련된 대부분의 작업을 수행할 때 관리자는 imqobjmgr 하위 명령의 옵션으로 다음 정보를 지정해야 합니다.

- 관리 객체의 유형:

  허용되는 유형은 표 7-2에 나와 있습니다.

- 관리 객체의 JNDI 조회 이름:

  클라이언트 코드에서 객체 저장소에 있는 관리 객체를 참조할 때(JNDI 사용) 사용되는 논리적 이름입니다.

- 관리 객체의 속성:

- 대기열 및 주제의 경우: 브로커에 있는 물리적 대상의 이름. `imqcmd create` 하위 명령의 `-n` 옵션으로 지정한 이름입니다. 이름을 지정하지 않으면 기본 이름인 `Untitled_Destination_Object`가 사용됩니다.
- 연결 팩토리의 경우: 클라이언트가 연결하는 브로커의 호스트 이름과 포트 번호. 이 정보를 지정하지 않으면 로컬 호스트와 기본 포트 번호(7676)가 사용됩니다. 167페이지의 "관리 객체 속성" 절에서는 객체 속성을 지정하는 방법을 설명합니다.

추가 속성에 대한 내용은 167페이지의 "관리 객체 속성"을 참조하십시오.

- JNDI 객체 저장소의 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 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 169페이지의 "객체 저장소 속성"을 참조하십시오.

관리 객체 속성

관리 객체의 속성은 속성-값 쌍을 사용해서 지정합니다. 다음 절에서는 이런 속성에 대해 설명합니다.

연결 팩토리 관리 객체

연결 팩토리(및 XA 연결 팩토리) 관리 객체에는 표 7-3에 나열된 것과 같은 속성이 있습니다. 주로 고려해야 하는 두 속성은 클라이언트 응용 프로그램에서 연결할 브로커를 지정할 때 사용하는 `imqBrokerHostPort`와 `imqBrokerHostName`입니다. 174페이지의 "연결 팩토리 추가" 절에서는 객체 저장소에 연결 팩토리 관리 객체를 추가할 때 이런 속성을 지정하는 방법을 설명합니다.

연결 팩토리 속성과 그 사용 방법에 대한 자세한 설명은 MQ *Developer's Guide*와 JavaDoc API 설명서에서 MQ 클래스 `com.sun.messaging.ConnectionConfiguration` 부분을 참조하십시오.

표 7-3 연결 팩토리 속성

속성/등록 정보 이름	유형	기본값
imqAckOnAcknowledge	문자열	지정하지 않음
imqAckOnProduce	문자열	지정하지 않음
imqAckTimeout	문자열	0밀리초
imqBrokerHostName	문자열	localhost
imqBrokerHostPort	문자열	7676
imqBrokerServicePort	문자열	0
imqConfiguredClientID	문자열	지정하지 않음
imqConnectionType	문자열	TCP
imqConnectionURL	문자열	http://localhost/imq/tunnel
imqDefaultPassword	문자열	guest
imqDefaultUsername	문자열	guest
imqDisableSetClientID	문자열	false
imqFlowControlCount	문자열	100
imqFlowControlIsLimited	문자열	false
imqFlowControlLimit	문자열	1000

표 7-3 연결 팩토리 속성*(계속)*

속성/등록 정보 이름	유형	기본값
imqLoadMaxToServerSession	문자열	false
imqQueueBrowserMax MessagesPerRetrieve	문자열	1000
imqQueueBrowserRetrieve Timeout	문자열	60,000밀리초
imqReconnect	부울	false
imqReconnectDelay	문자열	30,000밀리초
imqReconnectRetries	문자열	0
imqSetJMSXAppID	문자열	false
imqSetJMSXConsumerTXID	문자열	false
imqSetJMSXProducerTXID	문자열	false
imqSetJMSXRcvTimestamp	문자열	false
imqSetJMSXUserID	문자열	false
imqSSLIsHostTrusted	문자열	true
imqJMSDeliveryMode	정수	2(영구)
imqJMSExpiration	정수	0(만료되지 않음)
imqJMSPriority	정수	4(일반 우선 순위)
imqOverrideJMSDeliveryMode	부울	false
imqOverrideJMSExpiration	부울	false
imqOverrideJMSPriority	부울	false
imqOverrideJMSHeadersTo TemporaryDestinations	부울	false

대상 관리 객체

물리적 주제 또는 대기열 대상을 나타내는 대상 관리 객체는 표 7-4에 나열된 것과 같은 속성을 갖습니다. 175페이지의 "주제 또는 대기열 추가" 절에서는 객체 저장소에 대상 관리 객체를 추가할 때 이런 속성을 지정하는 방법을 설명합니다.

여기에서 주로 고려해야 할 속성은 `imqDestinationName`입니다. 이 속성은 주제 또는 대기열 관리 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 MQ 클래스 `com.sun.messaging.DestinationConfiguration` 부분을 참조하십시오.

표 7-4 대상 속성

속성/등록 정보 이름	유형	기본값
`imqDestinationDescription`	문자열	대상 객체의 설명
`imqDestinationName`	문자열[1]	`Untitled_Destination_Object`

1 대상 이름은 영문자와 숫자만 포함할 수 있으며(공백 없음) 영문자 또는 "_" 문자나 "$" 문자로 시작해야 합니다.

객체 저장소 속성

객체 저장소의 속성은 `-j` 옵션을 사용해서 지정하며 속성-값 쌍으로 구성됩니다. 일반적으로 다음 속성을 지정해야 합니다.

초기 컨텍스트 및 위치 정보

이 항목의 형식은 파일 시스템 저장소와 LDAP 서버 중 어느 것을 사용하는지에 따라 달라집니다.

파일 시스템 저장소 파일 시스템 저장소를 사용하는 예로, C 드라이브에 *MyObjstore*라는 폴더를 만들고 초기 컨텍스트 및 위치 속성에 다음과 같은 값을 각각 지정합니다.

```
-j "java.naming.factory.initial=
          com.sun.jndi.fscontext.RefFSContextFactory"

-j "java.naming.provider.url=file:///C:/MyObjStore"
```

LDAP 서버 LDAP 서버 사용의 예로, 초기 컨텍스트와 위치 속성에 다음과 같은 값을 각각 지정합니다.

```
-j "java.naming.factory.initial=
          com.sun.jndi.ldap.LdapCtxFactory"

-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
```

보안 정보(LDAP 전용)

이 항목의 형식은 LDAP 공급자에 따라 달라집니다. 또한 LDAP 구현에 제공되는 설명서를 참고하면 보안 정보가 모든 작업에 대해 필요한지 아니면 저장된 데이터를 변경하는 작업에만 필요한지 확인할 수 있습니다.

보안 속성은 다음과 같습니다.

```
-j "java.naming.security.principal=
      uid=fooUser, ou=People, o=imq"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

표 7-5에서 이 항목에 대해 설명합니다.

표 7-5 객체 저장소의 보안 속성

속성	설명
…principal	서비스에 대해 호출자를 인증할 때 사용되는 principal의 아이디입니다. 이 항목의 형식은 인증 방법에 따라 달라집니다. 이 등록 정보를 지정하지 않으면 서비스 공급자에 의해 동작이 결정됩니다.
…credentials	서비스에 대해 호출자를 인증할 때 사용되는 principal의 자격 증명입니다. 등록 정보의 값은 인증 방법에 따라 달라집니다. 예를 들어, 해시 비밀번호, 단순 텍스트 비밀번호, 키, 인증서 등이 될 수 있습니다. 이 등록 정보를 지정하지 않으면 서비스 공급자에 의해 동작이 결정됩니다.
…authentication	사용할 보안 수준입니다. 값은 none, simple, strong의 세 키워드 중 하나입니다. 이 등록 정보를 지정하지 않으면 서비스 공급자에 의해 동작이 결정됩니다. simple을 지정하면 imqobjmgr이 누락된 principal 또는 자격 증명 값을 묻는 프롬프트를 표시합니다. 그러면 아이디 정보를 좀더 안전하게 제공할 수 있습니다.

입력 파일 사용

imqobjmgr 명령을 사용하면 imqobjmgr 하위 명령 절의 일부 또는 전부를 나타내는 Java 등록 정보 파일 구문을 사용하는 입력 파일의 이름을 지정할 수 있습니다.

객체 관리자 유틸리티(imqobjmgr)에 입력 파일을 사용하는 것은 여러 imqobjmgr 실행에 동일하게 사용되면서 입력할 내용이 많은 객체 저장소 속성을 지정하는 경우에 특히 유용합니다. 입력 파일을 사용하면 명령줄에 허용된 최대 문자 수를 초과하는 상황을 방지할 수도 있습니다.

imqobjmgr 입력 파일의 일반 구문은 다음과 같습니다(명령줄 옵션이 아닌 version 등록 정보는 입력 파일에만 적용되며 값은 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
...
```

입력 파일을 사용하는 방법의 예로, 다음과 같은 `imqobjmgr` 명령이 있습니다.

```
imqobjmgr add
       -t qf
       -l "cn=myQCF"
       -o "imqBrokerHostName=foo"
       -o "imqBrokerHostPort=777"
       -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 `MyCmdFile`과 같은 파일에 캡슐화할 수 있습니다.

```
    version=2.0
    cmdtype=add
    obj.type=qf
    obj.lookupName=cn=myQCF
    obj.attrs.imqBrokerHostName=foo
    obj.attrs.imqBrokerHostPort=777
    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
```

입력 파일에 몇 가지 옵션을 지정하고 명령줄을 사용해서 또다른 옵션을 지정할 수도 있습니다. 이런 방법으로 입력 파일을 사용하면 유틸리티가 실행될 때마다 항상 같은 내용을 가지는 하위 명령 절을 지정할 수 있습니다. 예를 들어, 다음 명령은 관리 객체의 저장 위치를 제외하고 연결 팩토리 관리 객체를 추가할 때 필요한 모든 옵션을 지정합니다.

```
imqobjmgr add
        -t qf
        -l "cn=myQCF"
        -o "imqBrokerHostName=foo"
        -o "imqBrokerHostPort=777"
         -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=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
```

다음 위치에 입력 파일에 대한 추가 예가 나와 있습니다.

```
IMQ_HOME/demo/imqobjmgr
```

관리 객체 추가 및 삭제

이 절에서는 연결 팩토리 및 주제 또는 대기열 대상의 관리 객체를 객체 저장소에 추가하는 방법을 설명합니다.

주 객체 관리자 유틸리티(imqobjmgr)는 MQ 관리 객체만을 나열하고 표시합니다. 객체 저장소에 추가할 관리 객체와 동일한 조회 이름을 가진 비 MQ 객체를 포함해야 하는 경우 추가 작업을 시도하면 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 객체를 추가합니다.

대기열 연결 팩토리를 추가하려면 다음과 같은 명령을 사용합니다.

```
imqobjmgr add
  -t qf
  -l "cn=myQCF"
  -o "imqBrokerHostName=myHost"
  -o "imqBrokerHostPort=7272"
  -j "java.naming.factoryinitial=
              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"
```

앞의 명령은 조회 이름이 cn=myQCF이고, myHost에서 실행되며 포트 7272를 수신하는 브로커에 연결되는 관리 객체를 만듭니다. 관리 객체는 LDAP 서버에 저장됩니다.

주 LDAP 서버를 사용하여 관리 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.

imqobjmgr 명령의 인수로 입력 파일을 지정해도 같은 결과를 얻을 수 있습니다. 자세한 내용은 171페이지의 "입력 파일 사용"을 참조하십시오.

주제 또는 대기열 추가

클라이언트 응용 프로그램에서 브로커의 물리적 대상에 액세스할 수 있게 하려면 이 대상에 해당하는 관리 객체를 객체 저장소에 추가합니다.

객체 저장소에 해당 관리 객체를 추가하기 전에 물리적 대상을 먼저 만드는 것이 좋습니다. 명령 유틸리티(imqcmd)를 사용하여 브로커에 객체 저장소의 대상 관리 객체에 해당하는 물리적 위치를 만듭니다. 물리적 대상을 만드는 자세한 내용은 152페이지의 "대상 관리"를 참조하십시오.

다음 명령은 조회 이름이 myTopic이고 물리적 대상 이름이 TestTopic인 주제 대상에 해당하는 관리 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 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 입력 파일 `MyCmdFile`을 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 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 객체 삭제

관리 객체를 삭제하려면 delete 하위 명령을 사용합니다. 객체의 조회 이름과 유형, 위치를 지정해야 합니다.

다음 명령은 조회 이름이 cn=myTopic이고 LDAP 서버에 저장되는 주제의 관리 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 객체를 나열하고 개별 객체에 대한 정보를 표시하려면 list와 query 하위 명령을 사용합니다.

관리 객체 나열

모든 관리 객체의 목록을 보거나 특정 유형에 해당하는 모든 관리 객체의 목록을 보려면 list 하위 명령을 사용합니다. 다음 샘플 코드에서는 관리 객체가 LDAP 서버에 저장되어 있다고 가정합니다.

다음 명령은 모든 객체를 나열합니다.

```
imqobjmgr list
    -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"
```

다음 명령은 queue 유형의 모든 객체를 나열합니다.

```
imqobjmgr list
  -t q
  -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"
```

단일 객체 관련 정보

특정 관리 객체에 대한 정보를 얻으려면 query 하위 명령을 사용합니다. 객체의 조회 이름과 관리 객체를 포함하는 객체 저장소의 속성(초기 컨텍스트, 위치 등)을 지정해야 합니다.

다음 예에서는 query 하위 명령을 사용하여 조회 이름이 myTopic인 객체에 관련된 정보를 표시합니다.

```
imqobjmgr query
  -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"
```

관리 객체 업데이트

관리 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 객체의 속성을 변경합니다.

```
imqobjmgr update
 -t tf
 -l "cn=MyTCF"
 -o imqReconnectRetries=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"
```

8장

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. 이 장의 첫 부분에서는 저장소를 만들고, 채우고, 관리하는 방법에 대해 설명합니다. Sun™ ONE Message Queue (MQ) 보안에 대한 소개는 57페이지의 "보안 관리자"를 참조하십시오.

사용자 권한 부여 브로커 작업에 대한 사용자 액세스를 사용자 아이디 또는 그룹에 매핑하는 등록 정보 파일을 편집해야 합니다. 이 장의 두 번째 부분에서는 이 등록 정보 파일을 사용자 정의하는 방법을 설명합니다.

암호화: SSL 서비스 설정 Secure Socket Layer (SSL) 표준 기반의 연결 서비스를 사용하면 클라이언트와 브로커 사이에 전달되는 메시지를 암호화할 수 있습니다. MQ에서 암호화를 처리하는 방법에 대한 소개는 59페이지의 "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 액세스를 확보할 때 비밀번호가 필요한 경우에는 다음과 같은 세 가지 방법으로 비밀번호를 얻을 수 있습니다.

- 브로커가 시작할 때 시스템에서 프롬프트를 표시하게 합니다.
- 브로커를 시작할 때 명령줄 옵션으로 비밀번호를 전달합니다(120페이지의 "브로커 시작" 및 121페이지의 표 5-2 참조).
- 시스템에서 브로커를 시작할 때 액세스하는 passfile에 비밀번호를 저장합니다(204페이지의 "Passfile 사용" 참조).

사용자 인증

사용자가 브로커에 연결하려고 하면 브로커는 제공되는 아이디와 비밀번호를 검사하여 사용자를 인증한 후 브로커가 참조하는 사용자 저장소의 정보와 일치하면 연결을 허용합니다. 이 저장소의 유형에는 두 가지가 있습니다.

- MQ와 함께 제공되는 플랫 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 182페이지의 "플랫 파일 사용자 저장소 사용"을 참조하십시오.

- LDAP 서버

  사용자 저장소에 LDAP v2 또는 v3 프로토콜을 사용하는 기존 또는 새 LDAP 디렉토리 서버가 될 수 있습니다. 플랫 파일 저장소만큼 사용이 쉽지는 않지만 안전하기 때문에 작업 환경에 적합합니다.

  LDAP 사용자 저장소를 사용하는 경우에는 LDAP 공급업체에서 제공하는 도구를 사용하여 사용자 저장소를 채우고 관리해야 합니다. 자세한 내용은 189페이지의 "사용자 저장소에 LDAP 서버 사용"을 참조하십시오.

플랫 파일 사용자 저장소 사용

MQ에는 플랫 파일 사용자 저장소와 플랫 파일 사용자 저장소를 채우고 관리할 때 사용할 수 있는 명령줄 도구인 MQ 사용자 관리자(`imqusermgr`)가 제공됩니다. 다음 절에서는 플랫 파일 사용자 저장소와 해당 초기 항목, 그리고 저장소를 채우고 관리하는 방법에 대해 설명합니다.

기본 플랫 파일 저장소는 다음 위치에 있습니다.

`IMQ_HOME/etc/passwd` (Solaris의 경우 `/etc/imq/passwd`)

저장소는 다음 표와 같이 두 항목(행)이 이미 정의된 상태로 제공됩니다.

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 MQ 브로커를 즉시 사용할 수 있습니다. 즉, MQ 브로커를 사용하기 위해 초기 사용자/비밀번호 설정을 할 필요가 없습니다.

초기 guest 사용자 항목을 사용하면 JMS 클라이언트에서 기본 guest 사용자 아이디와 비밀번호를 사용하여 브로커에 연결할 수 있습니다(예를 들어 테스트 목적으로).

초기 admin 사용자 항목을 사용하면 imqcmd 명령을 사용하여 기본 admin 사용자 아이디와 비밀번호로 브로커를 관리할 수 있습니다. 이 초기 항목을 업데이트하여 비밀번호를 변경하는 것이 좋습니다.

사용자 관리자 유틸리티를 사용하면 먼저 브로커를 구성 또는 시작하지 않고도 플랫 파일 사용자 저장소를 편집하거나 채울 수 있습니다. 사용자 관리자 유틸리티를 사용하기 위한 유일한 요구 사항은 브로커를 설치한 호스트에서 실행해야 하고 저장소에 쓰려면 적절한 권한이 있어야 한다는 것입니다.

- Solaris의 경우, 사용자 관리자 유틸리티는 루트 사용자나 Solaris 역할 기반 액세스 제어를 통해 액세스가 허용된 루트가 아닌 사용자로 실행할 수 있습니다. 이러한 액세스를 허용하려면 루트 사용자가 다음과 같이 /etc/user_attr에 항목을 입력해야 합니다.

  username::::type=normal;profiles=Message Queue Management

  그러면 지정한 사용자가 MQ 권한 프로필에 추가됩니다. 이 기능을 사용하려면 먼저 프로필 셸(pfsh, pfksh, pfcsh)을 실행해야 합니다. 예를 들면 다음과 같습니다.

  ```
  % /usr/bin/pfsh
  ```

  그런 다음 원하는 사용자 관리자(imqusermgr) 명령을 실행해야 합니다.

  Solaris 역할 기반 액세스 제어에 대한 자세한 내용은 다음을 참조하십시오. http://docs.sun.com/?q=Rights+Profile&p=/doc/806-4078/6jd6cjrvl&a=view)

- Windows의 경우에는 설치 후 운영 체제에서 사용자 이름 기반의 권한 속성을 제어하지 않기 때문에 모든 사용자가 사용자 저장소 파일에 쓰기를 수행할 수 있습니다.

다음 절에서는 플랫 파일 사용자 저장소를 채우고 관리하는 방법을 설명합니다.

사용자 관리자 유틸리티(imqusermgr)

사용자 관리자 유틸리티를 사용하면 파일 기반 사용자 저장소를 관리할 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.

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 -u` *name* `-p` *passwd* `[-g` *group*`] [-s]`	저장소에 사용자와 관련 비밀번호를 추가하고 선택적으로 사용자 그룹을 지정합니다.
`delete -u` *name* `[-s] [-f]`	지정한 사용자를 저장소에서 삭제합니다.
`list [-u` *name*`]`	지정한 사용자 또는 모든 사용자에 대한 정보를 표시합니다.
`update -u` *name* `-p` *passwd* `[-a` *state*`] [-s] [-f]` `update -u` *name* `-a` *state* `[-p` *passwd*`] [-s] [-f]`	지정한 사용자의 비밀번호 및/또는 상태를 업데이트합니다.

imqusermgr 명령 옵션 요약

표 8-3에는 `imqusermgr` 명령의 옵션이 나열되어 있습니다.

표 8-3 `imqusermgr` 옵션

옵션	설명
`-a` *active_state*	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.
`-p` *passwd*	사용자의 비밀번호를 지정합니다.
`-g` *group*	사용자 그룹을 지정합니다. 유효한 값에는 `admin`, `user`, `anonymous`가 있습니다.
`-s`	자동 모드를 설정합니다.
`-u` *name*	사용자 아이디를 지정합니다.
`-v`	버전 정보를 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.

그룹

저장소에 사용자 항목을 추가할 때, 관리자는 사용자에 대해 사전 정의된 `admin`, `user`, `anonymous`의 세 가지 그룹 중 하나를 지정할 수 있습니다. 그룹을 지정하지 않으면 기본 그룹인 `user`가 할당됩니다.

- *admin* 그룹은 브로커 관리자에 사용됩니다. 이 그룹에 할당된 사용자는 기본적으로 브로커를 구성하고 관리할 수 있습니다. 관리자는 admin 그룹에 사용자를 두 명 이상 할당할 수도 있습니다.
- *user* 그룹은 일반(관리 외의) JMS 클라이언트 응용 프로그램에 사용됩니다. 대부분의 MQ 클라이언트 응용 프로그램은 user 그룹에서 인증된 브로커에 액세스합니다. 따라서 클라이언트 응용 프로그램은 기본적으로 모든 주제와 대기열에 해당하는 메시지를 생성 및 사용하거나 모든 대기열의 메시지를 찾아볼 수 있습니다.
- *anonymous* 그룹은 브로커에 알려진 사용자 아이디를 사용하지 않는(응용 프로그램에서 사용하는 실제 사용자의 아이디를 알지 못하기 때문일 수 있음) JMS 클라이언트 응용 프로그램에 사용됩니다. 이 그룹은 대부분의 FTP 서버에 있는 anonymous 계정과 유사합니다. 관리자는 anonymous 그룹에 한 번에 한 사용자만 할당할 수 있습니다. 액세스 제어를 통해 이 그룹의 액세스 권한을 사용자 그룹에 비해 작게 설정하거나 배포할 때 이 그룹에서 사용자를 제거해야 합니다.

사용자의 그룹을 변경하려면, 관리자가 사용자 항목을 삭제한 후 그 사용자에 다른 항목을 추가하여 새 그룹을 지정해야 합니다.

그룹의 구성원이 수행할 수 있는 작업을 정의하는 액세스 규칙을 지정할 수 있습니다. 자세한 내용은 192페이지의 "사용자 권한 부여: 액세스 제어 등록 정보 파일"을 참조하십시오.

상태

관리자가 사용자를 저장소에 추가하면 그 사용자는 기본적으로 활성 상태가 됩니다. 사용자를 비활성 상태로 만들려면 관리자가 update 명령을 사용해야 합니다. 예를 들어, 다음 명령을 통해 사용자 JoeD가 비활성 상태가 됩니다.

```
imqusermgr update -u JoeD -a false
```

비활성 상태인 사용자의 항목은 저장소에 보존되지만, 비활성 상태인 사용자가 새 연결을 열 수는 없습니다. 사용자가 비활성 상태일 때 관리자가 같은 이름을 가진 다른 사용자를 추가하면 작업이 실패합니다. 관리자는 비활성 상태인 사용자 항목을 삭제하거나, 새 사용자의 이름을 변경하거나, 또는 새 사용자에게 다른 이름을 사용해야 합니다. 그러면 관리자가 중복되는 이름 또는 비밀번호를 추가하지 않게 됩니다.

사용자 아이디 및 비밀번호 형식

사용자 아이디와 비밀번호는 다음과 같은 지침을 따라야 합니다.

- 사용자 아이디와 비밀번호에는 표 8-4에 있는 문자를 포함할 수 없습니다.

표 8-4 사용자 아이디 및 비밀번호에 유효하지 않은 문자

문자	설명
*	별표
,	쉼표
:	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 비밀번호가 `sesame`인 `Katharine`이라는 사용자를 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 Katharine의 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
```

```
-----------------------------------
User Name    Group     Active State
-----------------------------------
isa          admin    true
```

다음 명령은 모든 사용자에 대한 정보를 나열합니다.

```
imqusermgr list
```

```
---------------------------------------
User Name    Group        Active State
---------------------------------------
testuser3    user         true
testuser2    user         true
testuser1    user         true
isa          admin        true
admin        admin        true
guest        anonymous    true
testuser5    user         false
testuser4    user         false
```

기본 관리자 비밀번호 변경

보안을 위해 admin의 기본 비밀번호를 혼자만 아는 값으로 변경해야 합니다. 그러려면 imqusermgr 도구를 사용해야 합니다.

다음 명령은 기본 비밀번호를 grandpoobah로 변경합니다.

```
imqusermgr update -u admin -p grandpoobah
```

브로커가 실행 중일 때 명령줄 도구 중 하나를 실행하여 변경 사항의 적용 여부를 빠르게 확인할 수 있습니다. 예를 들어, 다음 명령은

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

이전 비밀번호 사용이 실패하더라도 제대로 작동해야 합니다.

비밀번호를 변경한 후에 관리 콘솔을 포함한 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 브로커에 연결하거나 특정 작업을 수행하려 할 때 브로커가 LDAP 서버로부터 사용자 및 그룹에 대한 정보를 쿼리할 수 있습니다. 인스턴스 구성 파일은 다음 위치에 있습니다.

`IMQ_VARHOME/instances/`*`brokerName`*`/props/config.properties`
(Solaris의 경우 `/var/imq/instances/`*`brokerName`*`/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 digest (`digest`) 중 어느 쪽으로 할 것인지 결정합니다. 사용자 저장소에 LDAP 디렉토리를 사용하는 경우에는 인증 유형을 `basic`으로 설정해야 합니다. 예를 들면 다음과 같습니다.

   ```
   imq.authentication.type=basic
   ```

3. LDAP 액세스를 제어하는 브로커 등록 정보도 설정해야 합니다. 브로커의 인스턴스 구성 파일에 저장되는 이러한 등록 정보에 대한 설명은 표 8-5에 나와 있습니다. MQ에서는 JNDI API를 사용하여 LDAP 디렉토리 서버와 통신합니다. 이러한 등록 정보에 사용되는 구문과 용어에 대한 자세한 내용은 JNDI 설명서를 참조하십시오. MQ 3.0.1에서는 Sun JNDI LDAP 공급자를 사용하며 단순 인증을 사용합니다.

표 8-5 LDAP 관련 등록 정보

등록 정보	설명
`imq.user_repository.ldap.server`	LDAP 서버의 host:port입니다. host는 디렉토리 서버를 실행 중인 호스트의 정규화된 DNS 이름을 지정합니다. port는 디렉토리 서버에서 통신에 사용하는 포트 번호를 지정합니다.
`imq.user_repository.ldap.principal`	브로커에서 검색할 디렉토리 서버에 바인드할 때 사용하는 고유 이름입니다. 디렉토리 서버에서 익명 검색을 허용하는 경우에는 이 등록 정보에 값을 할당할 필요가 없습니다.

표 8-5 LDAP 관련 등록 정보(*계속*)

등록 정보	설명
`imq.user_repository.ldap.password`	브로커에서 사용하는 고유 이름에 연결된 비밀번호입니다. passfile에만 지정될 수 있습니다(204페이지의 "Passfile 사용" 참조). 보안을 강화하려면 브로커가 비밀번호를 묻는 프롬프트를 표시하게 하거나 `imqbrokerd -ldappassword` 명령줄 옵션을 사용하여 비밀번호를 지정합니다. 디렉토리 서버가 익명 검색을 허용하는 경우에는 비밀번호가 필요 없습니다.
`imq.user_repository.ldap.base`	사용자 항목에 사용되는 디렉토리 기반입니다.
`imq.user_repository.ldap.uidattr`	사용자를 고유하게 식별하는 값을 가진 공급자별 속성 식별자입니다. `uid`, `cn` 등이 있습니다.
`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/tutorial에 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. MQ 3.0.1에서는 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.

표 8-5 LDAP 관련 등록 정보(*계속*)

등록 정보	설명
`imq.user_repository.ldap.grpfiltler`	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 `false`로 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 192페이지의 "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 사용자 저장소 등록 정보에 보안 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 브로커에 연결한 후 메시지를 생성하거나, 대상에서 메시지를 사용하거나, 대기열 대상에서 메시지를 찾아보는 경우가 있습니다. 사용자가 이러한 작업을 시도하면 브로커는 *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. 액세스 제어 등록 정보 파일을 편집하여 이런 작업을 특정 사용자 및 그룹으로 제한할 수 있습니다.

ACL 파일은 사용자 정보가 플랫 파일 저장소에 있는지, LDAP 저장소에 있는지와 관계없이 모두 사용할 수 있습니다. 기본 ACL 등록 정보 파일은 브로커와 함께 설치됩니다. 이름은 `accesscontrol.properties`이며 설치 프로그램에서 다음 디렉토리에 설치합니다.

```
IMQ_HOME/etc (Solaris의 경우 /etc/imq)
```

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
resourceType	`connection`, `queue` 또는 `topic` 중 하나입니다.
resourceVariant	*resourceType*에서 지정한 유형의 인스턴스입니다. 예를 들면 myQueue가 있습니다. 와일드카드 문자(*)를 사용하여 모든 연결 서비스 유형 또는 모든 대상을 나타낼 수 있습니다.
operation	값은 표현할 액세스 규칙의 종류에 따라 달라집니다.
access	`allow` 또는 `deny` 중 하나입니다.
principalType	`user` 또는 `group` 중 하나입니다. 자세한 내용은 185페이지의 "그룹"을 참조하십시오.
principals	규칙의 왼쪽에 지정한 액세스를 갖는 사람입니다. `principalType`이 `user`이면 개별 사용자 또는 사용자 목록(쉼표로 구분)이 될 수 있고, `principalType`이 `group`이면 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/1.3/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 사람에게는 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 `NORMAL`과 `ADMIN` 두 가지입니다. 기본적으로 `NORMAL` 유형에는 모든 사용자가 액세스할 수 있지만 `ADMIN` 유형의 연결 서비스에는 그룹이 `admin`인 사용자만 액세스할 수 있습니다.

연결 액세스 제어 규칙을 편집하여 사용자의 연결 액세스 권한을 제한할 수 있습니다. 예를 들어, 다음 규칙에서는 `NORMAL`에 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
resourceType	queue 또는 topic 중 하나여야 합니다.
resourceVariant	대상 이름 또는 모든 대기열이나 모든 주제를 나타내는 모든 대상(*)입니다.
operation	produce, consume 또는 browse 중 하나여야 합니다.
access	allow 또는 deny 중 하나여야 합니다.
principalType	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 보여줍니다.

- 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 `topic`입니다.

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 서비스를 사용한 작업(엔터프라이즈판)

MQ 엔터프라이즈판은 Secure Socket Layer (SSL) 표준에 따라 TCP/IP (ssljms 및 ssladmin)와 HTTP (httpsjms)에서 연결 서비스를 지원합니다. 이러한 SSL 기반 연결 서비스를 사용하면 클라이언트와 브로커 사이에서 보내는 메시지를 암호화할 수 있습니다. 현재 MQ 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.

MQ의 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 서비스 설정

TCP/IP에서 직접 보안 연결을 제공하는 SSL 기본 연결 서비스에는 두 가지가 있습니다.

ssljms 이 연결 서비스는 클라이언트와 브로커 사이에서 암호화된 보안 연결을 통해 JMS 메시지를 전달할 때 사용됩니다.

ssladmin 이 연결 서비스는 명령줄 관리 도구인 명령 유틸리티(`imqcmd`)와 브로커 사이에서 암호화된 보안 연결을 만들 때 사용됩니다. 관리 콘솔(`imqadmin`)에서는 보안 연결이 지원되지 않습니다.

➤ **ssljms 연결 서비스를 설정하는 방법**

1. 자체 서명된 인증서를 생성합니다.
2. 브로커에서 ssljms 연결 서비스를 활성화합니다.
3. 브로커를 시작합니다.
4. 클라이언트를 구성하고 실행합니다.

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

MQ 3.0.1의 SSL 지원은 클라이언트가 알려지고 신뢰할 수 있는 서버와 통신한다는 가정하에 전송 데이터를 보호하기 위한 것입니다. 따라서 MQ 3.0.1에서는 자체 서명된 인증서만을 사용해서 SSL이 구현됩니다.

imqkeytool 명령을 실행하여 브로커에 자체 서명된 인증서를 생성합니다. ssljms와 ssladmin 연결 서비스 모두에 같은 인증서를 사용할 수 있습니다. 명령 프롬프트에서 다음을 입력합니다.

```
imqkeytool -broker
```

유틸리티는 필요한 정보를 묻는 메시지를 표시합니다. (Unix 시스템에서 키 저장소를 만들 권한을 가지려면 imqkeytool을 수퍼유저(루트)로 실행해야 할 수도 있습니다.)

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 잠글 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 passfile에 저장할 수도 있습니다(204페이지의 "Passfile 사용" 참조).

imqkeytool을 실행하면 JDK keytool 유틸리티를 실행하여 자체 서명된 인증서를 생성하고 생성된 키를 다음 위치의 MQ 키 저장소에 넣을 수 있습니다.

```
IMQ_HOME/etc/keystore (Solaris의 경우 /etc/imq/keystore)
```

키 저장소의 형식은 JDK1.2 keytool 유틸리티에서 지원하는 것과 같습니다.

MQ 키 저장소의 구성 가능한 등록 정보는 표 8-8에 나와 있습니다(등록 정보 구성 지침은 5장, "브로커 시작 및 구성" 참조).

표 8-8 키 저장소 등록 정보

등록 정보 이름	설명
`imq.keystore.file.dirpath`	SSL 기반 서비스의 경우: 키 저장소 파일이 있는 디렉토리의 경로를 지정합니다. 기본값: `IMQ_HOME/etc` (Solaris의 경우 `/etc/imq/`)
`imq.keystore.file.name`	SSL 기반 서비스의 경우: 키 저장소 파일의 이름을 지정합니다. 기본값: `keystore`
`imq.keystore.password`	SSL 기반 서비스의 경우: 키 저장소 비밀번호를 지정합니다. passfile에만 저장할 수 있습니다(204페이지의 "Passfile 사용" 참조). 보안을 강화하려면 브로커가 비밀번호를 묻는 프롬프트를 표시하게 하거나 `imqbrokerd -password` 명령줄 옵션을 사용하여 비밀번호를 지정합니다.

문제를 해결하기 위해 키 쌍을 다시 생성해야 할 수도 있습니다. 예를 들면 다음과 같습니다.

- 키 저장소 비밀번호를 잊은 경우
- 브로커를 시작할 때 SSL 서비스를 초기화할 수 없고 다음과 같은 예외가 발생하는 경우 `java.security.UnrecoverableKeyException: Cannot recover key.`

  이 예외는 198페이지의 "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. 다음 위치에서 브로커의 키 저장소를 제거합니다.

   `IMQ_HOME/etc/keystore` (Solaris의 경우 `/etc/imq/keystore`)

2. `imqkeytool`을 다시 실행하여 198페이지의 "1단계. 자체 서명된 인증서 생성"의 설명과 같이 키 쌍을 생성합니다.

2단계. 브로커에서 SSL 기반 서비스 활성화

브로커에서 SSL 서비스를 활성화하려면 `imq.service.activelist` 등록 정보에 `ssljms` (`ssladmin`)를 추가해야 합니다.

1. 브로커의 인스턴스 구성 파일을 엽니다. 이 파일은 다음 위치에 있습니다.

   `IMQ_VARHOME/instances/`*`brokerName`*`/props/config.properties`
   (Solaris의 경우 `/var/imq/instances/`*`brokerName`*`/props/config.properties`)

   여기서 *brokerName*은 브로커 인스턴스의 이름입니다.

2. `ssljms` 또는 `ssladmin` 값 또는 두 값 모두를 원하는 서비스에 따라 `imq.service.activelist` 등록 정보에 추가합니다.

   ```
   imq.service.activelist=jms,admin,httpjms,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(204페이지의 "Passfile 사용" 참조)에 비밀번호를 입력합니다. 먼저 다음과 같은 브로커 구성 등록 정보를 설정해야 합니다(116페이지의 "인스턴스 구성 파일 편집" 참조).

  ```
  imq.passfile.enabled=true
  ```

  이 등록 정보가 설정되면 두 가지 중 한 가지 방법으로 passfile에 액세스할 수 있습니다.

  - passfile의 위치를 `imqbrokerd` 명령에 전달합니다.

    ```
    imqbrokerd -passfile /tmp/mypassfile
    ```

  - -passfile 옵션 없이 브로커를 시작하되, 다음의 두 브로커 구성 등록 정보를 사용해서 passfile의 위치를 지정합니다.

    ```
    imq.passfile.dirpath=/tmp
    ```

    ```
    imq.passfile.name=mypassfile
    ```

passfile 관련 브로커 등록 정보의 목록을 보려면 60페이지의 표 2-6을 참조하십시오.

SSL을 사용하는 브로커나 클라이언트를 시작할 때 몇 초간 CPU 사이클을 과도하게 사용하는 경우가 있을 수 있습니다. 이것은 MQ에서 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 돌아갑니다.

4단계. SSL기반 클라이언트 구성 및 실행

마지막으로 보안 연결 서비스를 사용하도록 클라이언트를 구성해야 합니다. 클라이언트의 종류에는 사용하는 연결 서비스에 따라 ssljms를 사용하는 JMS 클라이언트와 ssladmin을 사용하는 MQ 관리 명령 유틸리티(`imqcmd`)의 두 가지가 있습니다. 다음 절에서 이 두 가지에 대해 별도로 설명합니다.

JMS 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 MQ 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 응용 프로그램에서의 ssljms 연결 서비스 사용에 대한 자세한 내용은 MQ *Developer's Guide*의 관리 객체 사용에 대한 장을 참조하십시오.

명령 유틸리티(imqcmd)

`imqcmd`를 사용할 때 `-secure` 옵션을 포함하면 보안 관리 연결을 설정할 수 있습니다(140페이지의 표 6-2 참조). 예를 들면 다음과 같습니다.

```
imqcmd list svc -b hostName:port -u adminName -p adminPassword -secure
```

여기서 *adminName*과 *adminPassword*는 MQ 사용자 저장소의 유효한 항목입니다(플랫 파일 저장소를 사용하는 경우 188페이지의 "기본 관리자 비밀번호 변경" 참조).

이 예에서 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 서브릿을 통해 보안 연결을 설정합니다. HTTPS 지원의 구조와 구현에 대한 설명은 213페이지의 부록 B, "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.

passfile은 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. passfile의 권한은 브로커를 시작하는 사용자가 이 파일을 읽을 수 있도록 설정해야 합니다.

passfile에는 표 8-9에 있는 것과 같은 비밀번호를 포함할 수 있습니다.

표 8-9 Passfile의 비밀번호

비밀번호	설명
`imq.keystore.password`	SSL 기반 서비스의 키 저장소 비밀번호를 지정합니다.
`imq.user_repository.ldap.password`	구성된 LDAP 사용자 저장소에 바인드할 수 있도록 브로커에 할당한 고유 이름에 연결되는 비밀번호를 지정합니다.
`imq.persist.jdbc.password`	필요한 경우 데이터베이스 연결을 여는 비밀번호를 지정합니다.

다음 위치에 샘플 passfile이 있습니다.

`IMQ_HOME/etc/`passfile.sample (Solaris의 경우 `/etc/imq/passfile.sample`)

플러그 인 지속성 설정

이 부록에서는 플러그 인 지속성을 사용해서 JDBC로 액세스할 수 있는 데이터 저장소에 액세스하도록 브로커를 설정하는 방법을 설명합니다.

소개

Sun™ ONE Message Queue (MQ) 브로커에는 지속성 정보의 기록과 복원을 관리하는 지속성 관리자 구성 요소가 있습니다(54페이지의 "지속성 관리자" 참조). 지속성 관리자는 기본적으로 기본 제공되는 파일 기반 데이터 저장소에 액세스하도록 구성되어 있지만 JDBC 호환 드라이버를 통해 액세스할 수 있는 데이터 저장소를 플러그 인으로 사용하도록 재구성할 수도 있습니다.

플러그 인 지속성을 사용하도록 브로커를 구성하려면 브로커 인스턴스 구성 파일에 JDBC 관련 등록 정보를 설정해야 합니다. MQ 지속성 작업을 수행할 수 있는 적절한 데이터베이스 스키마도 만들어야 합니다. MQ에는 JDBC 드라이버와 브로커 구성 등록 정보를 사용해서 플러그 인 데이터베이스를 만들고 관리하는 데이터베이스 관리자 유틸리티(`imqdbmgr`)가 제공됩니다.

이 부록에 설명된 절차에서는 Java 2 SDK Enterprise Edition과 함께 제공되는 Cloudscape DBMS를 사용하는 경우를 예로 들었습니다(J2EE SDK는 java.sun.com에서 다운로드 가능). 예에서는 (클라이언트/서버 버전 대신) Cloudscape에 내장된 버전을 사용합니다. 절차의 지침에는 Cloudscape 예에 있는 경로 이름과 등록 정보를 사용했습니다. 이러한 지침은 "예:"라는 말로 구분됩니다.

다른 예는 다음 위치에 있습니다.

`IMQ_HOME/demo/jdbc` (Solaris의 경우 `/usr/demo/imq/jdbc`)

JDBC로 액세스할 수 있는 데이터 저장소 플러그 인

JDBC로 액세스할 수 있는 데이터 저장소를 플러그 인하는 작업은 단 몇 단계로 이루어집니다.

➤ JDBC로 액세스할 수 있는 데이터 저장소를 플러그 인하는 방법

1. 브로커의 구성 파일에서 JDBC 관련 등록 정보를 설정합니다.

   207페이지의 표 A-1에 설명된 등록 정보를 참조하십시오.

2. 다음 경로에 JDBC 드라이버 jar 파일의 사본 또는 심볼릭 링크를 넣습니다.

   `IMQ_VARHOME/lib/ext` (Solaris의 경우 `/usr/share/lib/imq/ext/`)

   복사 예(Solaris):

   ```
   % cp j2sdk_install_directory/lib/cloudscape/cloudscape.jar
   IMQ_VARHOME/lib/ext
   ```

   심볼릭 링크 예(Solaris):

   ```
   % ln -s j2sdk_install_directory/lib/cloudscape/cloudscape.jar
   IMQ_VARHOME/lib/ext
   ```

3. MQ 지속성에 필요한 데이터베이스 스키마를 만듭니다.

   `imqdbmgr create all` 명령(내장 데이터베이스의 경우) 또는 `imqdbmgr create tbl` 명령(외부 데이터베이스의 경우)을 사용합니다. 210페이지의 "데이터베이스 관리자 유틸리티(imqdbmgr)"를 참조하십시오.

   예:

   ```
   % cd IMQ_HOME/bin (Solaris의 경우 /usr/bin)
   % imqdbmgr create all
   ```

주 내장 데이터베이스를 사용하는 경우에는 데이터베이스를 다음 디렉토리에 만드는 것이 좋습니다.

`IMQ_VARHOME/instances/`*brokerName*`/dbstore/`*dabatabseName*
(Solaris의 경우 /var/imq/instances/*brokerName*`/dbstore/`)

내장 데이터베이스가 사용자 아이디와 비밀번호로 보호되지 않는 경우에는 파일 시스템 권한으로 보호합니다. 브로커에서 데이터베이스를 읽고 쓸 수 있도록 하려면 브로커를 실행하는 사용자가 imqdbmgr 명령을 사용해서 내장 데이터베이스를 만든 사용자와 같아야 합니다(210페이지의 "데이터베이스 관리자 유틸리티(imqdbmgr)" 참조).

JDBC 관련 브로커 구성 등록 정보

브로커의 인스턴스 구성 파일은 다음 위치에 있습니다.

`IMQ_VARHOME/instances/`*brokerName*`/props/config.properties`
(Solaris의 경우 `/var/imq/instances/`*brokerName*`/props/config.properties`)

파일이 아직 없는 경우에는 MQ에서 파일을 만들 수 있도록 `-name brokerName` 옵션을 사용해서 브로커를 시작해야 합니다.

표 A-1에는 JDBC로 액세스할 수 있는 데이터 저장소를 플러그 인할 때 설정해야 하는 구성 등록 정보가 있습니다. 이 등록 정보는 각 브로커 인스턴스의 구성 파일(`config.properties`)에서 플러그 인 지속성을 사용하여 설정합니다. 테이블에는 Cloudscape DBMS 예에 지정할 값이 포함되어 있습니다.

표 A-1 JDBC 관련 등록 정보

등록 정보 이름	설명
`imq.persist.store`	파일 기반 또는 JDBC 기반 데이터 저장소를 지정합니다. *예*: `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. *예*: Cloudscape에는 필요 없음
`imq.persist.jdbc.driver`	데이터베이스에 연결할 JDBC 드라이버의 Java 클래스 이름을 지정합니다. *예*: `COM.cloudscape.core.JDBCDriver`

표 A-1 JDBC 관련 등록 정보*(계속)*

등록 정보 이름	설명
`imq.persist.jdbc.opendburl`	기존 데이터베이스에 대한 연결을 여는 데이터베이스 URL을 지정합니다. *예*: `jdbc:cloudscape:IMQ_VARHOME/instances/`*brokerName*`/dbstore/imqdb` (Solaris의 경우 `jdbc:cloudscape:var/imq..`)
`imq.persist.jdbc.createdburl` (선택 사항)	데이터베이스를 만들기 위해 연결을 여는 데이터베이스 URL을 지정합니다. (`imqdbmgr`을 사용해서 데이터베이스를 만드는 경우에만 지정합니다.) *예*: `jdbc:cloudscape:IMQ_VARHOME/instances/`*brokerName*`/dbstore/imqdb;create=true` (Solaris의 경우 `jdbc:cloudscape:var/imq..`)
`imq.persist.jdbc.closedburl` (선택 사항)	브로커를 종료할 때 현재 데이터베이스 연결을 종료하는 데이터베이스 URL을 지정합니다. *예*(Cloudscape에 필요): `jdbc:cloudscape:;shutdown=true`
`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`

`imq.persist.jdbc.needpassword`
(선택 사항)
	브로커에 액세스할 때 데이터베이스에 비밀번호가 필요한지 여부를 지정합니다. `true` 값은 비밀번호가 필요하다는 것을 의미합니다. 비밀번호는 다음과 같은 명령줄 옵션을 사용해서 지정할 수 있습니다.
	`imqbrokerd -dbpassword`
	`imqdbmgr -p`
	명령줄 옵션과 passfile (204페이지의 "Passfile 사용" 참조) 중 어느 것으로도 비밀번호를 제공하지 않은 경우 브로커는 비밀번호를 묻는 프롬프트를 표시합니다.

표 A-1 JDBC 관련 등록 정보*(계속)*

등록 정보 이름	설명

`imq.persist.jdbc.password`
(선택 사항)
	필요한 경우 데이터베이스 연결을 열 때 사용되는 비밀번호를 지정합니다. passfile에만 지정될 수 있습니다(204페이지의 "Passfile 사용" 참조). 보안을 강화하려면 브로커가 비밀번호를 묻는 프롬프트를 표시하게 하거나 다음 명령줄 옵션을 사용하여 비밀번호를 지정합니다.
	`imqbrokerd -dbpassword`
	`imqdbmgr -p`

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:

Cloudscape 내장 데이터베이스의 예에서는 데이베이스 연결 URL에 데이터베이스 절대 경로를 지정하는 대신(표 A-1 예 참조), -D 명령줄 옵션을 사용해서 Cloudscape 시스템 디렉토리를 정의할 수 있습니다.

```
-Dcloudscape.system.home=IMQ_VARHOME/instances/brokerName/dbstore
```

이 경우 데이터베이스를 만들고 여는 URL을 다음과 같이 간단하게 지정할 수 있습니다.

```
imq.persist.jdbc.createdburl=jdbc:cloudscape:imqdb;create=true
```

그리고

```
imq.persist.jdbc.opendburl=jdbc:cloudscape:imqdb
```

가 각각 적용됩니다.

데이터베이스 관리자 유틸리티(imqdbmgr)

MQ에는 지속성에 필요한 스키마를 설정하는 데이터베이스 관리자 유틸리티(`imqdbmgr`)가 포함되어 있습니다. 이 유틸리티는 테이블이 손상되거나 데이터 저장소로 다른 데이터베이스를 사용하는 경우에 MQ 데이터베이스 테이블을 삭제하는 용도로도 사용할 수 있습니다.

주 브로커 인스턴스가 비정상적으로 중단하면 참조되지 않은 지속성 정보가 데이터 저장소에 남을 수 있습니다. 브로커를 시작한 후 종료하면 보통은 데이터 저장소가 지워집니다.

이 절에서는 기본 `imqdbmgr` 명령 구문을 설명하고, 하위 명령 목록을 제공하고, `imqdbmgr` 명령 옵션을 요약합니다.

명령 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 A-2에 나열된 하위 명령이 포함되어 있습니다.

표 A-2 imqdbmgr 하위 명령

하위 명령 및 인수	설명
`create all`	새 데이터베이스와 MQ 영구 저장소 스키마를 만듭니다. 이 명령은 내장 데이터베이스 시스템에 사용되며, 사용하는 경우에는 `imq.persist.jdbc.createdburl` 등록 정보를 지정해야 합니다.
`create tbl`	기존 데이터베이스 시스템에 MQ 영구 저장소 스키마를 만듭니다. 이 명령은 외부 데이터베이스 시스템에 사용됩니다.
`delete tbl`	현재 영구 저장소 데이터베이스에서 기존 MQ 데이터베이스 테이블을 삭제합니다.
`recreate tbl`	현재 영구 저장소 데이터베이스에서 기존 MQ 데이터베이스 테이블을 삭제한 후 MQ 영구 저장소 스키마를 다시 만듭니다.

imqdbmgr 명령 옵션 요약

표 A-3에는 imqdbmgr 명령의 옵션이 나열되어 있습니다.

표 A-3 imqdbmgr 옵션

옵션	설명
`-D`*property=value*	지정한 등록 정보를 지정한 값으로 설정합니다.
`-b` *brokerName*	브로커 인스턴스 이름을 지정하고 해당 인스턴스 구성 파일을 사용합니다.
`-h`	사용 도움말을 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
`-p` *password*	데이터베이스 비밀번호를 지정합니다.
`-u` *name*	데이터베이스 사용자 아이디를 지정합니다.
`-v`	버전 정보를 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.

HTTP/HTTPS 지원(엔터프라이즈판)

Sun™ ONE Message Queue (MQ) 엔터프라이즈판(26페이지의 "제품 판" 참조)은 HTTP 및 HTTPS 연결을 모두 지원합니다. (HTTPS는 Secure Socket Layer 표준을 사용하는 HTTP상의 보안 연결입니다.) 이 지원을 사용하면 클라이언트 응용 프로그램에서 직접적인 TCP 연결 대신 HTTP 프로토콜을 사용하여 브로커와 통신할 수 있습니다. 이 부록에서는 이 지원의 구현에 사용되는 구조를 설명하고, 클라이언트에서 MQ 메시징에 HTTP 기반 연결을 사용할 때 필요한 설정 작업에 대해서도 설명합니다.

HTTP/HTTPS 지원 구조

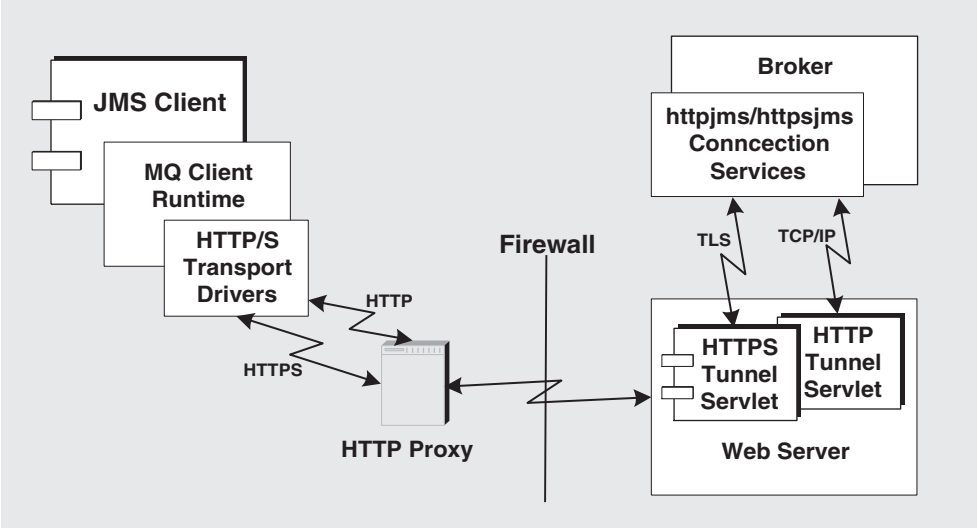
MQ 메시징은 HTTP/HTTPS 연결 위에서 실행할 수 있습니다. HTTP/HTTPS 연결은 보통 방화벽을 통해서도 허용되기 때문에 이렇게 하면 방화벽으로 클라이언트 응용 프로그램과 브로커를 분리할 수 있습니다.

214페이지의 그림 B-1에서는 HTTP/HTTPS 지원과 관련된 주요 구성 요소를 보여줍니다.

- 클라이언트쪽에서는 HTTP 전송 드라이버가 MQ 메시지를 HTTP 요청으로 캡슐화하고 요청이 정확한 순서로 웹 서버에 전송되는지 확인합니다.
- JMS 클라이언트는 필요한 경우 HTTP 프록시 서버를 사용해서 브로커와 통신할 수 있습니다. 프록시의 주소는 클라이언트를 시작할 때 명령줄 옵션을 사용해서 지정합니다. 자세한 내용은 219페이지의 "HTTP 프록시 사용"을 참조하십시오.

- HTTP 또는 HTTPS 터널 서브릿(모두 MQ와 함께 제공)이 웹 서버에 로드되고 JMS 메시지를 브로커로 전달하기 전에 클라이언트 HTTP 요청으로부터 추출하는 데 사용됩니다. HTTP/HTTPS 터널 서브릿은 클라이언트에서 이루어진 HTTP 요청에 대한 응답으로 브로커 메시지를 클라이언트에 다시 보내는 작업도 수행합니다. 한 HTTP/HTTPS 터널 서브릿을 사용해서 여러 브로커에 액세스할 수 있습니다.

그림 B-1 HTTP/HTTPS 지원 구조



- 브로커쪽에서는 httpjms 또는 httpsjms 연결 서비스가 해당 터널 서브릿에서 들어오는 메시지를 분해하고 다중화를 해제합니다.
- 웹 서버가 오류로 다시 시작되면 모든 연결이 복원되며 클라이언트에는 아무 영향도 주지 않습니다. 브로커가 오류로 다시 시작되면 예외가 발생하며 클라이언트는 연결을 다시 설정해야 합니다. 드물기는 하지만, 웹 서버와 브로커에 모두 오류가 발생하고 브로커가 다시 시작되지 않은 경우에는 웹 서버가 클라이언트 연결을 복원한 후 클라이언트에 알리지 않고 브로커 연결을 계속 기다릴 수 있습니다. 이런 상황을 방지하려면 항상 브로커를 다시 시작해야 합니다.

그림 B-1에서 알 수 있듯이, HTTP와 HTTPS가 지원하는 구조는 서로 매우 비슷합니다. 가장 큰 차이는 HTTPS (httpsjms 연결 서비스)의 경우 터널 서브릿이 클라이언트 응용 프로그램과 브로커 모두에 대해 보안 연결을 갖는다는 점입니다.

브로커에 대한 보안 연결은 MQ의 HTTPS 터널 서브릿에 해당하는 SSL 사용 터널 서브릿을 통해 제공되며 연결을 요청하는 모든 브로커에 자체 서명된 인증서를 전달합니다. 인증서는 브로커가 HTTPS 터널 서브릿에 대해 암호화된 연결을 설정할 때 사용합니다. 이 연결이 설정되고 나면 클라이언트 응용 프로그램과 웹 서버에서 클라이언트 응용 프로그램과 터널 서브릿 사이의 보안 연결을 처리할 수 있습니다.

HTTP 지원 구현

다음 절에서는 HTTP 지원을 구현하기 위해 수행해야 하는 단계에 대해 설명합니다.

➤ **HTTP 지원을 구현하는 방법**

1. HTTP 터널 서브릿을 웹 서버에 배포합니다.
2. 브로커의 httpjms 연결 서비스를 구성하고 브로커를 시작합니다.
3. HTTP 연결을 구성합니다.

1단계. 웹 서버에 HTTP 터널 서브릿 배포

웹 서버에 HTTP 터널 서브릿을 배포하는 일반적인 방법에는 두 가지가 있습니다.

- jar 파일로 배포 - Servlet 2.1 이전 버전을 지원하는 웹 서버용
- 웹 아카이브(WAR) 파일로 배포 - Servlet 2.2 이상 버전을 지원하는 웹 서버용

Jar 파일로 배포

MQ 터널 서브릿 배포는 호스트 웹 서버에 액세스할 수 있는 적절한 jar 파일을 작성하고 시작 시 서브릿을 로드하도록 웹 서버를 구성하여 이루어집니다.

터널 서브릿 jar 파일(imqservlet.jar)은 HTTP 터널 서브릿에 필요한 모든 클래스를 포함하며 다음 디렉토리에 있습니다.

IMQ_HOME/lib (Solaris의 경우 /usr/share/lib/imq)

서브릿 2.x를 지원하는 모든 웹 서버를 사용하여 이 서브릿을 로드할 수 있습니다. 서브릿 클래스 이름은 다음과 같습니다.

```
com.sun.messaging.jmq.transport.
httptunnel.servlet.HttpTunnelServlet
```

웹 서버에서 imqservlet.jar 파일을 볼 수 있어야 합니다. 웹 서버와 브로커를 서로 다른 호스트에서 실행하려면 imqservlet.jar 파일의 복사본을 웹 서버에서 액세스할 수 있는 위치에 저장해야 합니다.

시작 시 이 서브릿을 로드하도록 웹 서버를 구성해야 합니다(219페이지의 "예: HTTP 터널 서브릿 배포" 참조).

또한 성능 향상을 위해 웹 서버의 액세스 로깅 기능은 사용하지 않는 것이 좋습니다.

웹 아카이브 파일로 배포

HTTP 터널 서브릿을 WAR 파일로 배포하는 작업은 웹 서버에서 제공하는 메커니즘을 사용하여 수행합니다. HTTP 터널 서브릿 WAR 파일(imqhttp.war)은 다음 디렉토리에 있습니다.

IMQ_HOME/lib (Solaris의 경우 /usr/share/lib/imq)

WAR 파일에는 웹 서버에서 서브릿을 로드하여 실행하는 데 필요한 기본 구성 정보가 들어 있는 배포 설명자가 포함되어 있습니다.

2단계. httpjms 연결 서비스 구성

MQ 3.0.1 브로커에는 HTTP 지원이 기본적으로 사용되지 않기 때문에 httpjms 연결 서비스를 사용하도록 브로커를 재구성해야 합니다. 재구성한 경우 120페이지의 "브로커 시작"에 설명된 대로 브로커를 시작할 수 있습니다.

➤ httpjms 연결 서비스를 사용하는 방법

1. 다음 위치에서 브로커의 인스턴스 구성 파일을 엽니다.

   `IMQ_VARHOME/instances/`*brokerName*`/props/config.properties`
   (Solaris의 경우 `/var/imq/instances/`*brokerName*`/props/config.properties`)

   여기서 *brokerName*은 브로커 인스턴스의 이름입니다.

2. `imq.service.activelist` 등록 정보에 `httpjms` 값을 추가합니다.

   ```
   imq.service.activelist=jms,admin,httpjms
   ```

브로커는 시작할 때 호스트 시스템에서 실행 중인 웹 서버와 HTTP 터널 서브릿을 찾습니다. 그러나 원격 터널 서브릿에 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 217페이지의 표 B-1을 참조하십시오.

표 B-1 httpjms 연결 서비스 등록 정보

등록 정보 이름	설명
`imq.httpjms.http.servletHost`	필요한 경우 이 값을 변경하여 HTTP 터널 서브릿을 실행하는 호스트의 이름(호스트 이름 또는 IP 주소)을 지정합니다(원격 호스트이거나 로컬 호스트의 특정 호스트 이름일 수 있음). 기본값: `localhost`
`imq.httpjms.http.servletPort`	이 값을 변경하여 브로커가 HTTP 터널 서브릿에 액세스하는 데 사용하는 포트 번호를 지정합니다(웹 서버에서 기본 포트를 변경한 경우 이 등록 정보를 적절히 변경해야 함). 기본값: `7675`
`imq.httpjms.http.pullPeriod`	브로커에서 메시지를 가져오기 위해 각 클라이언트가 만든 HTTP 요청 사이의 간격(초)을 지정합니다. 값이 0 또는 음수인 경우 클라이언트는 하나의 HTTP 요청을 항상 보류 상태로 두고 가능한 빨리 메시지를 가져오도록 준비합니다. 클라이언트 수가 많은 경우 웹 서버 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 웹 서버 자원이 절약됩니다. 기본값: `-1`

3단계. HTTP 연결 구성

클라이언트 응용 프로그램에서는 제대로 구성된 연결 팩토리 관리 객체를 사용해서 브로커에 대한 HTTP 연결을 설정해야 합니다. 이 절에서는 HTTP 연결 구성 문제에 대해 논의합니다.

연결 팩토리 속성 설정

HTTP 지원을 구현하려면 다음 연결 팩토리 속성을 설정합니다(167페이지의 "연결 팩토리 관리 객체" 참조).

- `imqConnectionType` 속성을 `HTTP`로 설정합니다.
- `imqConnectionURL`을 HTTP 터널 서브릿 URL로 설정합니다.

  `http://`*`hostName`*`:`*`port`*`/imq/tunnel`

다음 방법 중 하나를 사용하여 연결 팩토리 속성을 설정할 수 있습니다.

- 연결 팩토리 관리 객체를 만드는 `imqobjmgr` 명령에 `-o` 옵션을 사용하거나(174페이지의 "연결 팩토리 추가" 참조) 관리 콘솔(`imqadmin`)을 사용하여 연결 팩토리 관리 객체를 만들 때 속성을 설정합니다.
- 클라이언트 응용 프로그램을 시작하는 명령에 `-D` 옵션을 사용합니다(MQ *Developer's Guide* 참조).
- JMS API 호출을 사용하여 연결 팩토리를 클라이언트 응용 프로그램 코드에서 프로그래밍 방식으로 만든 후 해당 연결 팩토리의 속성을 설정합니다(MQ *Developer's Guide* 참조).

단일 서브릿을 사용하여 다중 브로커에 액세스

다중 브로커를 실행할 경우 다중 웹 서버 및 서브릿 인스턴스를 구성하지 않아도 됩니다. 동시에 실행 중인 브로커 간에 단일 웹 서버와 HTTP 터널 서브릿을 공유할 수 있습니다. 이를 위해 `imqConnectionURL` 연결 팩토리 속성을 다음과 같이 구성해야 합니다.

`http://`*`hostName:port/imq/tunnel`*`?ServerName=`*`hostName`*`:`*`brokerName`*

여기서 *hostName*은 브로커 호스트 이름이고 *brokerName*은 클라이언트가 액세스할 특정 브로커 인스턴스의 이름입니다.

*hostName*과 *brokerName*에 정확한 문자열을 입력했는지 확인하려면 브라우저에서 서브릿 URL에 액세스하여 HTTP 터널 서브릿의 상태 보고서를 생성합니다. 보고서에는 서브릿이 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 서브릿에 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: HTTP 터널 서브릿 배포

이 절에서는 Sun ONE WEB Server에서 HTTP 터널 서브릿을 jar 파일과 WAR 파일로 배포하는 방법을 모두 설명합니다. 사용 방법은 Sun ONE Web Server의 버전에 따라 다르지만, Servlet 2.2 이상 버전을 지원하지 않는 경우 WAR 파일 배포를 처리할 수 없습니다.

Jar 파일로 배포

다음은 브라우저 기반 관리 GUI를 사용하는 Sun ONE Web Server, FastTrack Edition 4.1에 배포 시의 참조 지침입니다. 이 절차는 다음과 같은 일반적인 단계로 구성됩니다.

1. 서브릿 추가
2. 서브릿 가상 경로 구성
3. 서브릿 로드
4. 서브릿 액세스 로그 비활성화

이러한 단계는 다음 하위 절에 설명되어 있습니다. 웹 브라우저를 사용하여 서브릿 URL에 액세스함으로써 HTTP 터널 서브릿 배포 성공 여부를 확인할 수 있습니다. 상태 정보를 표시해야 합니다.

서브릿 추가

➤ **터널 서브릿을 추가하는 방법**

1. 서브릿 탭을 선택합니다.
2. 서브릿 속성 구성을 선택합니다.
3. 서브릿 이름 필드에 터널 서브릿의 이름을 지정합니다.
4. 서브릿 코드(클래스 이름) 필드를 다음 값으로 설정합니다.

   ```
   com.sun.messaging.jmq.transport.
   httptunnel.servlet.HttpTunnelServlet
   ```

5. 서브릿 클래스 경로 필드에 `imqservlet.jar`에 대한 전체 경로를 입력합니다. 예를 들면 다음과 같습니다.

   ```
   IMQ_HOME/lib/imqservlet.jar
   ```
   (Solaris의 경우 `/usr/share/lib/imq/imqservlet.jar`)

6. 서브릿 인수 필드에 표 B-2와 같은 선택 인수를 입력합니다.

표 B-2 HTTP 터널 서브릿 Jar 파일 배포에 사용되는 서브릿 인수

인수	기본값	참조
servletHost	모든 호스트	217페이지의 표 B-1 참조
servletPort	7675	217페이지의 표 B-1 참조

두 인수를 모두 사용하는 경우에는 인수 사이를 쉼표로 구분합니다.

`servletPort=`*portnumber*`, servletHost=...`

`serverHost` 및 `serverPort` 인수는 웹 서버와 브로커 간 통신에만 적용되며 기본값에 문제가 있는 경우에만 설정됩니다. 그런 경우에도 브로커 구성 등록 정보를 적절하게 설정해야 합니다(217페이지의 표 B-1 참조). 예를 들면 다음과 같습니다.

```
imq.httpjms.http.servletPort
```

서브릿 가상 경로(서브릿 URL) 구성

➤ **터널 서브릿에 대한 가상 경로(서브릿 URL)를 구성하는 방법**

1. 서브릿 탭을 선택합니다.
2. 서브릿 가상 경로 변환 구성을 선택합니다.
3. 가상 경로 필드를 설정합니다.

   예를 들어, URL을 `http://`*hostName*`:`*port*`/imq/tunnel`로 지정하려는 경우 가상 경로 필드에 다음 문자열을 입력합니다.

   ```
   /imq/tunnel
   ```

4. 서브릿 이름 필드를 220페이지의 "서브릿 추가"의 3단계와 동일한 값으로 설정합니다.

서브릿 로드

➤ **웹 서버 시작 시 터널 서브릿을 로드하는 방법**

1. 서브릿 탭을 선택합니다.
2. 전역 속성 구성을 선택합니다.
3. 시작 서브릿 필드에 220페이지의 "서브릿 추가"의 3단계와 동일한 서브릿 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 ONE Web Server 6.0 서비스 팩 2에 배포 시의 참조 지침입니다.

웹 브라우저를 사용하여 서브릿 URL에 액세스하면 HTTP 터널 서브릿의 배포 성공 여부를 확인할 수 있습니다. 상태 정보를 표시해야 합니다.

➤ **http 터널 서브릿을 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 위치와 WAR 파일 경로 필드에서 `imqhttp.war` 파일을 가리키는 적절한 값을 선택합니다. 이 파일은 다음 디렉토리에 있습니다.

   `IMQ_HOME/lib` (Solaris의 경우 `/usr/share/lib/imq`)
8. 응용 프로그램 URI 필드에 "`/imq`"(따옴표 없이)를 입력합니다.
9. 서브릿을 배포할 설치 디렉토리 경로(일반적으로 Sun ONE Web Server 설치 루트 아래)를 입력합니다.
10. 확인을 누릅니다.
11. 웹 서버 인스턴스를 다시 시작합니다.

이제 서브릿을 다음 주소에서 사용할 수 있습니다.

`http://`*`hostName`*`:`*`port`*`/imq/tunnel`

이제 클라이언트에서 이 URL을 통해 HTTP 연결을 사용하는 메시지 서비스에 연결할 수 있습니다.

HTTPS 지원 구현

다음 절에서는 HTTPS 지원을 구현하기 위해 수행해야 하는 단계에 대해 설명합니다. 215 페이지의 "HTTP 지원 구현"과 비슷한 내용에 SSL 인증서 생성 및 액세스에 필요한 단계가 추가되어 있습니다.

➤ **HTTPS 지원을 구현하는 방법**

1. HTTPS 터널 서브릿에 대해 자체 서명된 인증서를 생성합니다.
2. HTTPS 터널 서브릿을 웹 서버에 배포합니다.
3. 브로커의 httpsjms 연결 서비스를 구성하고 브로커를 시작합니다.
4. HTTPS 연결을 구성합니다.

각 단계에 대해서는 뒤에서 자세히 설명합니다.

1단계. HTTPS 터널 서브릿에 대해 자체 서명된 인증서 생성

MQ 3.0.1의 SSL 지원은 클라이언트가 알려지고 신뢰할 수 있는 서버와 통신한다는 가정 하에 전송 데이터를 보호하기 위한 것입니다. 따라서 MQ 3.0.1에서는 자체 서명된 서버 인증서만 사용하여 SSL을 구현합니다. httpsjms 연결 서비스 구조에서 HTTPS 터널 서브릿은 브로커와 응용 프로그램 클라이언트에 대해 서버 역할을 담당합니다.

`imqkeytool` 유틸리티를 실행하여 터널 서브릿에 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 `imqkeytool`을 수퍼유저(루트)로 실행해야 할 수도 있습니다.)

`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 잠글 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 서브릿이 키 저장소를 열 수 있습니다.

imqkeytool을 실행하면 JDK keytool 유틸리티를 실행하여 자체 서명된 인증서를 생성하고 *keystore_location* 인수에서 지정한 위치의 MQ 키 저장소 파일에 넣을 수 있습니다. (키 저장소의 형식은 JDK1.2 keytool에서 지원하는 것과 같습니다.)

주 HTTPS 터널 서브릿에서 키 저장소를 볼 수 있어야 합니다. *keystore_location*에 생성된 키 저장소를 HTTPS 터널 서브릿에서 액세스할 수 있는 위치로 이동/복사합니다(224페이지의 "2단계. 웹 서버에 HTTPS 터널 서브릿 배포" 참조).

2단계. 웹 서버에 HTTPS 터널 서브릿 배포

웹 서버에 HTTPS 터널 서브릿을 배포하는 일반적인 방법에는 두 가지가 있습니다.

- jar 파일로 배포 - Servlet 2.1 이전 버전을 지원하는 웹 서버용
- 웹 아카이브(WAR) 파일로 배포 - Servlet 2.2 이상 버전을 지원하는 웹 서버용

두 경우 모두 클라이언트와 브로커 사이에서 종단간 보안 통신이 이루어질 수 있도록 웹 서버에 암호화가 활성화되었는지 확인해야 합니다.

Jar 파일로 배포

MQ 터널 서브릿 배포는 호스트 웹 서버에 액세스할 수 있는 적절한 jar 파일을 작성하고 시작 시 서브릿을 로드하도록 웹 서버를 구성하여 이루어집니다.

터널 서브릿 jar 파일(imqservlet.jar)은 HTTPS 터널 서브릿에 필요한 모든 클래스를 포함하며 다음 디렉토리에 있습니다.

```
IMQ_HOME/lib (Solaris의 경우 /usr/share/lib/imq)
```

서브릿 2.x를 지원하는 모든 웹 서버를 사용하여 이 서브릿을 로드할 수 있습니다. 서브릿 클래스 이름은 다음과 같습니다.

```
com.sun.messaging.jmq.transport.
httptunnel.servlet.HttpsTunnelServlet
```

웹 서버에서 imqservlet.jar 파일을 볼 수 있어야 합니다. 웹 서버와 브로커를 서로 다른 호스트에서 실행하려면 imqservlet.jar 파일의 복사본을 웹 서버에서 액세스할 수 있는 위치에 저장해야 합니다.

시작 시 이 서브릿을 로드하도록 웹 서버를 구성해야 합니다(229페이지의 "예: HTTPS 터널 서브릿 배포" 참조).

JSSE jar 파일이 웹 서버에서 서브릿을 실행하는 클래스 경로에 있는지 확인합니다. 수행 방법은 웹 서버 설명서를 확인하십시오.

웹 서버 구성에서 중요한 부분 중 하나는 HTTPS 터널 서브릿에서 브로커와 보안 연결을 설정할 때 사용할 자체 서명된 인증서의 위치와 비밀번호를 지정하는 것입니다. 223페이지의 "1단계. HTTPS 터널 서브릿에 대해 자체 서명된 인증서 생성"에서 만든 키 저장소를 HTTPS 터널 서브릿에서 액세스할 수 있는 위치에 두어야 합니다.

또한 성능 향상을 위해 웹 서버의 액세스 로깅 기능은 사용하지 않는 것이 좋습니다.

웹 아카이브 파일로 배포

HTTPS 터널 서브릿을 WAR 파일로 배포하는 작업은 웹 서버에서 제공하는 배포 메커니즘을 사용하여 수행합니다. HTTPS 터널 서브릿 WAR 파일(imqhttps.war)은 다음 디렉토리에 있습니다.

IMQ_HOME/lib (Solaris의 경우 /usr/share/lib/imq)

WAR 파일에는 웹 서버에서 서브릿을 로드하여 실행하는 데 필요한 기본 구성 정보가 들어 있는 배포 설명자가 포함되어 있습니다.

하지만 imqhttps.war 파일의 배포 설명자에서는 터널 서브릿에 필요한 키 저장소 파일이 있는 위치를 알 수 없습니다(223페이지의 "1단계. HTTPS 터널 서브릿에 대해 자체 서명된 인증서 생성" 참조). 따라서 imqhttps.war 파일을 배포하기 전에 터널 서브릿의 배포 설명자(XML 파일)를 편집하여 키 저장소 위치를 지정해야 합니다.

3단계. httpsjms 연결 서비스 구성

MQ 3.0.1 브로커에는 HTTPS 지원이 기본적으로 사용되지 않기 때문에 httpsjms 연결 서비스를 사용하도록 브로커를 재구성해야 합니다. 재구성한 경우 120페이지의 "브로커 시작"에 설명된 대로 브로커를 시작할 수 있습니다.

➤ httpsjms 연결 서비스를 사용하는 방법

1. 다음 위치에서 브로커의 인스턴스 구성 파일을 엽니다.

   `IMQ_VARHOME/instances/`*brokerName*`/props/config.properties`
   (Solaris의 경우 `/var/imq/instances/`*brokerName*`/props/config.properties`)

   여기서 *brokerName*은 브로커 인스턴스의 이름입니다.

2. `imq.service.activelist` 등록 정보에 `httpsjms` 값을 추가합니다.

   ```
   imq.service.activelist=jms,admin,httpsjms
   ```

브로커는 시작할 때 호스트 시스템에서 실행 중인 웹 서버와 HTTPS 터널 서브릿을 찾습니다. 그러나 원격 터널 서브릿에 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 B-3을 참조하십시오.

표 B-3 httpsjms 연결 서비스 등록 정보

등록 정보 이름	설명
`imq.httpsjms.https servletHost`	필요한 경우 이 값을 변경하여 HTTPS 터널 서브릿을 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 서브릿에 액세스하는 데 사용하는 포트 번호를 지정합니다(웹 서버에서 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. 값이 0 또는 음수인 경우 클라이언트는 하나의 HTTP 요청을 항상 보류 상태로 두고 가능한 빨리 메시지를가져오도록 준비합니다. 클라이언트 수가 많은 경우 웹 서버 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 웹 서버 자원이 절약됩니다. 기본값: `-1`

4단계. HTTPS 연결 구성

클라이언트 응용 프로그램에서는 제대로 구성된 연결 팩토리 관리 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

웹 서버 인증서에 서명한 CA의 루트 인증서가 기본적으로 트러스트 데이터베이스에 있지 않거나 독자적인 웹 서버 인증서를 사용하는 경우에는 인증서를 트러스트 데이터베이스에 추가해야 합니다. 여기에 해당되면 다음 지침에 따르고, 그렇지 않으면 "연결 팩토리 속성 설정"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 `YES`로 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 속성 설정

HTTPS 지원을 구현하려면 다음 연결 팩토리 속성을 설정합니다(167페이지의 "연결 팩토리 관리 객체" 참조).

- imqConnectionType 속성을 HTTP로 설정합니다.

  브로커에 대한 보안 연결을 설정하려면 HTTP 터널 서브릿 대신 HTTPS 터널 서브릿을 통해 배포 및 연결을 수행합니다. 두 서브릿에는 같은 연결 유형이 사용됩니다.

- imqConnectionURL을 HTTPS 터널 서브릿 URL로 설정합니다.

  ```
  https://hostName:port/imq/tunnel
  ```

다음 방법 중 하나를 사용하여 연결 팩토리 속성을 설정할 수 있습니다.

- 연결 팩토리 관리 객체를 만드는 imqobjmgr 명령에 -o 옵션을 사용하거나(174페이지의 "연결 팩토리 추가" 참조) 관리 콘솔(imqadmin)을 사용하여 연결 팩토리 관리 객체를 만들 때 속성을 설정합니다.
- 클라이언트 응용 프로그램을 시작하는 명령에 -D 옵션을 사용합니다(MQ *Developer's Guide* 참조).
- JMS API 호출을 사용하여 연결 팩토리를 클라이언트 응용 프로그램 코드에서 프로그래밍 방식으로 만든 후 해당 연결 팩토리의 속성을 설정합니다(MQ *Developer's Guide* 참조).

단일 서브릿을 사용하여 다중 브로커에 액세스

다중 브로커를 실행할 경우 다중 웹 서버 및 서브릿 인스턴스를 구성하지 않아도 됩니다. 동시에 실행 중인 브로커 간에 단일 웹 서버와 HTTPS 터널 서브릿을 공유할 수 있습니다. 이를 위해 imqConnectionURL 연결 팩토리 속성을 다음과 같이 구성해야 합니다.

```
https://hostName:port/imq/tunnel?ServerName=hostName:brokerName
```

여기서 *hostName*은 브로커 호스트 이름이고 *brokerName*은 클라이언트가 액세스할 특정 브로커 인스턴스의 이름입니다.

*hostName*과 *brokerName*에 정확한 문자열을 입력했는지 확인하려면 브라우저에서 서브릿 URL에 액세스하여 HTTPS 터널 서브릿의 상태 보고서를 생성합니다. 보고서에는 서브릿이 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 서브릿에 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: HTTPS 터널 서브릿 배포

이 절에서는 Sun ONE Web Server에서 HTTPS 터널 서브릿을 jar 파일과 WAR 파일로 배포하는 방법을 모두 설명합니다. 사용 방법은 Sun ONE Web Server의 버전에 따라 다르지만, Servlet 2.2 이상 버전을 지원하지 않는 경우 WAR 파일 배포를 처리할 수 없습니다.

Jar 파일로 배포

다음은 브라우저 기반 관리 GUI를 사용하는 Sun ONE Web Server, FastTrack Edition 4.1에 배포 시의 참조 지침입니다. 이 절차는 다음과 같은 일반적인 단계로 구성됩니다.

1. 서브릿 추가
2. 서브릿 가상 경로 구성
3. 서브릿 로드
4. 서브릿 액세스 로그 비활성화

이러한 단계는 다음 하위 절에 설명되어 있습니다. 웹 브라우저를 사용하여 서브릿 URL에 액세스함으로써 HTTP 터널 서브릿 배포 성공 여부를 확인할 수 있습니다. 상태 정보를 표시해야 합니다.

서브릿 추가

➤ 터널 서브릿을 추가하는 방법

1. 서브릿 탭을 선택합니다.
2. 서브릿 속성 구성을 선택합니다.
3. 서브릿 이름 필드에 터널 서브릿의 이름을 지정합니다.
4. 서브릿 코드(클래스 이름) 필드를 다음 값으로 설정합니다.

   ```
   com.sun.messaging.jmq.transport.
   httptunnel.servlet.HttpsTunnelServlet
   ```

5. 서브릿 클래스 경로 필드에 `imqservlet.jar`에 대한 전체 경로를 입력합니다. 예를 들면 다음과 같습니다.

   ```
   IMQ_HOME/lib/imqservlet.jar
   ```
   (Solaris의 경우 `/usr/share/lib/imq/imqservlet.jar`)

6. 서브릿 인수 필드에 표 B-4와 같은 필수 및 선택 인수를 입력합니다.

표 B-4 HTTPS 터널 서브릿 Jar 파일 배포에 사용되는 서브릿 인수

인수	기본값	필수?	참조
`keystoreLocation`	없음	예	200페이지의 표 8-8
`keystorePassword`	없음	예	200페이지의 표 8-8
`serverHost`	모든 호스트	아니요	226페이지의 표 B-3
`serverPort`	7674	아니요	226페이지의 표 B-3

인수를 다음 예와 같이 쉼표로 분리합니다.

`keystoreLocation=`*keystore_location*`,keystorePassword=`*keystore_password*`, servletPort=`*portnumber*

`serverHost` 및 `serverPort` 인수는 웹 서버와 브로커 간 통신에만 적용되며 기본값에 문제가 있는 경우에만 설정됩니다. 그런 경우에도 브로커 구성 등록 정보를 적절하게 설정해야 합니다(226페이지의 표 B-3 참조). 예를 들면 다음과 같습니다.

```
imq.httpsjms.https.servletPort
```

서브릿 가상 경로(서브릿 URL) 구성

➤ **터널 서브릿에 대한 가상 경로(서브릿 URL)를 구성하는 방법**

1. 서브릿 탭을 선택합니다.
2. 서브릿 가상 경로 변환 구성을 선택합니다.
3. 가상 경로 필드를 설정합니다.

   예를 들어, URL을 `http://`*hostName*`:`*port*`/imq/tunnel`로 지정하려는 경우 가상 경로 필드에 다음 문자열을 입력합니다.

   ```
   /imq/tunnel
   ```

4. 서브릿 이름 필드를 230페이지의 "서브릿 추가"의 3단계와 동일한 값으로 설정합니다.

서브릿 로드

➤ **웹 서버 시작 시 터널 서브릿을 로드하는 방법**

1. 서브릿 탭을 선택합니다.
2. 전역 속성 구성을 선택합니다.
3. 시작 서브릿 필드에 230페이지의 "서브릿 추가"의 3단계와 동일한 서브릿 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 ONE Web Server 6.0 서비스 팩 2에 배포 시의 참조 지침입니다.
웹 브라우저를 사용하여 서브릿 URL에 액세스하면 HTTPS 터널 서브릿의 배포 성공 여부를 확인할 수 있습니다. 상태 정보를 표시해야 합니다.

HTTPS 터널 서브릿을 배포하기 전에 웹 서버의 클래스 경로에 JSSE jar 파일이 포함되어 있는지 확인합니다. 이 작업을 수행하는 가장 간단한 방법은 `jsse.jar`, `jnet.jar`, `jcert.jar` 파일을 `IWS60_TOPDIR/bin/https/jre/lib/ext`로 복사하는 것입니다.

또한, HTTPS 터널 서브릿을 배포하기 전에 키 저장소 파일의 위치를 가리키고 키 저장소 비밀번호를 지정하도록 배포 설명자를 수정해야 합니다.

➤ **HTTPS 터널 서브릿 WAR 파일을 수정하는 방법**

1. WAR 파일을 임시 디렉토리로 복사합니다.

   ```
   $ cp IMQ_HOME/lib/imqhttps.war /tmp
   ```
   (Solaris 의 경우 `$ cp /usr/share/lib/imq/imqhttps.war /tmp`)

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 `serverPort` 및 `serverHost` 인수도)에 정확한 값을 제공하도록 `web.xml` 파일을 편집합니다.
6. WAR 파일의 내용을 다시 어셈블합니다.

```
$ jar uvf imqhttps.war WEB-INF/web.xml
```

이제 수정된 `imqhttps.war` 파일을 사용하여 HTTPS 터널 서브릿을 배포할 수 있습니다. (키 저장소 비밀번호 노출이 우려되는 경우에는 파일 시스템 권한을 사용하여 `imqhttps.war` 파일에 대한 액세스를 제한할 수 있습니다.)

➤ https 터널 서브릿을 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 `imqhttps.war` 파일(232페이지의 "HTTPS 터널 서브릿 WAR 파일을 수정하는 방법" 참조)을 가리키는 적절한 값을 선택합니다.
8. 응용 프로그램 URI 필드에 "`/imq`"(따옴표 없이)를 입력합니다.
9. 서브릿을 배포할 설치 디렉토리 경로(일반적으로 Sun ONE Web Server 설치 루트 아래)를 입력합니다.
10. 확인을 누릅니다.
11. 웹 서버 인스턴스를 다시 시작합니다.

이제 서브릿을 다음 주소에서 사용할 수 있습니다.

`https://`*`hostName`*`:`*`port`*`/imq/tunnel`

이제 클라이언트에서 이 URL을 통해 보안 HTTPS 연결을 사용하는 메시지 서비스에 연결할 수 있습니다.

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

Sun™ ONE Message Queue (MQ) 3.0.1을 설치할 때 브로커를 Windows 서비스로 설치할 수 있습니다. MQ 3.0.1을 설치한 후에 imqsvcadmin을 사용해서 브로커를 Windows 서비스로 설치할 수도 있습니다.

브로커를 Windows 서비스로 설치한다는 것은 시스템을 시작할 때 시작하여 종료할 때까지 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(237페이지의 표 C-2 참조) imqbrokerd 명령에 사용하는 것과 같은 옵션을 지정합니다(120페이지의 "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.

명령 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

MQ 서비스 관리자 유틸리티(imqsvcadmin)에는 표 C-1에 나열된 하위 명령이 포함됩니다.

표 C-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 runtime 위치, 시작할 때 브로커에 전달되는 인수 값이 포함됩니다.
remove	서비스를 제거합니다.

imqsvcadmin 옵션 요약

표 C-2에는 imqsvcadmin 명령의 옵션이 나열되어 있습니다. 사용 설명은 다음에서 해당 작업 기반 절을 참조하십시오.

표 C-2 imqsvcadmin 옵션

옵션	설명
-h	사용 도움말을 표시합니다. 명령줄에 있는 명령은 실행되지 않습니다.
-javahome *path*	대체 Java2 호환 JDK의 경로를 지정합니다. 기본값은 번들형 런타임을 사용하는 것입니다. 예: imqsvcadmin -install -javahome d:\jdk1.4
-jrehome *path*	Java 2 호환 JRE의 경로를 지정합니다. 예: imqsvcadmin -install -jrehome d:\jre\1.4
-vmargs *arg* [[*arg*]...]	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 *arg* [[*arg*]...]	브로커 서비스에 전달할 추가 명령줄 인수를 지정합니다. imqbrokerd 옵션에 대한 설명은 120페이지의 "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 Runtime 사용

`-javahome` 또는 `-jrehome` 옵션을 사용해서 대체 Java runtime의 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 3.0\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 > 갱신을 선택하여 오류 이벤트를 봅니다.

MQ 데이터의 위치

Sun™ ONE Message Queue (MQ)에서는 표 D-1과 같이 각각 다른 위치에 저장되는 여러 범주의 데이터를 사용합니다.

표 D-1 MQ 3.0 데이터의 위치

데이터 범주	데이터의 위치
브로커 인스턴스 구성 등록 정보	`IMQ_VARHOME/instances/`*brokerName*`/props/config.properties` (Solaris의 경우 `var/imq/instances/`*brokerName*`/props/config.properties`)
영구 저장소(메시지, 대상, 영구 가입, 트랜잭션)	`IMQ_VARHOME/instances/`*brokerName*`/filestore/` (Solaris의 경우 `/var/imq/instances/`*brokerName*`/filestore/`) 또는 JDBC 액세스 가능 데이터 저장소
브로커 인스턴스 로그 파일	`IMQ_VARHOME/instances/`*brokerName*`/log/` (Solaris의 경우 `/var/imq/instances/`*brokerName*`/log/`)
관리 객체(객체 저장소)	선택한 로컬 디렉토리 또는 LDAP 서버
보안: 사용자 저장소	`IMQ_HOME/etc/passwd` (Solaris의 경우 `/etc/imq/passwd`) 또는 `LDAP` 서버
보안: 액세스 제어 파일	`IMQ_HOME/etc/accesscontrol.properties` (Solaris의 경우 `/etc/imq/accesscontrol.properties`)
보안: passfile	`IMQ_HOME/etc/`*passfileName* (Solaris의 경우 `/etc/imq/`*passfileName*)

표 D-1 MQ 3.0 데이터의 위치(*계속*)

데이터 범주	데이터의 위치
보안: 브로커의 키 저장소	`IMQ_HOME/etc/keystore` (Solaris의 경우 `/etc/imq/keystore`)

부록 E

MQ 인터페이스의 안정성

Sun™ ONE Message Queue (MQ)는 관리자가 관리 작업을 자동화할 때 유용할 수 있는 다양한 인터페이스를 사용합니다. 표 E-1은 안정성, 즉 향후 제품 버전에서 변경되지 않을 확률을 기준으로 이러한 인터페이스를 분류합니다. 분류 체계는 245페이지의 표 E-2에서 확인할 수 있습니다.

표 E-1 MQ 3.0.1 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 E-1 MQ 3.0.1 인터페이스의 안정성*(계속)*

인터페이스	분류
JMS API (`javax.jms`)	표준
JAXM API (`javax.xml`)	표준
관리 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
`imqxm.jar` 위치 및 이름	변화
`jaxm-api.jar` 위치 및 이름	변화
`saaj-api.jar` 위치 및 이름	변화
`saaj-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`, `imqusermgr`의 출력	불안정
브로커 로그 파일 위치 및 내용 형식	불안정
passfile	불안정
`accesscontrol.properties`	불안정

표 E-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 대해 명시적으로 종속하지 않는 것이 바람직합니다.

용어집

이 용어집에서는 Sun™ ONE Message Queue (MQ) 사용 중 접할 수 있는 용어 및 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 객체 사전 구성된 MQ 객체로서, 관리자가 작성하고 하나 이상의 JMS 클라이언트가 사용하는 연결 팩토리나 대상

관리 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 MQ 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 경로 지정 및 이후 사용자로의 전달을 위해 이동하는 MQ 메시지 서버상의 물리적 대상. 물리적 대상은 JMS 클라이언트가 자신이 누구를 위해 메시지를 생성하며 누구로부터 받은 메시지를 사용하는지 그 대상을 지정할 때 사용하는 관리 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 세트. 지점간 전달 모델을 위한 도메인과 게시/가입 전달 모델을 위한 도메인 등 두 가지 유형의 프로그래밍 도메인이 있습니다.

메시징 엔터프라이즈 응용 프로그램이 사용하는 비동기 요청, 보고서 또는 이벤트 시스템으로서, 느슨하게 연결된 응용 프로그램들이 신뢰성 있고 안전하게 정보를 전송할 수 있게 합니다.

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.

메시지 서비스 MQ 메시지 서버 참조

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 경로 지정을 수행합니다.

브로커 메시지 경로 지정, 전달, 지속성, 보안 및 로깅을 관리하고 관리자가 성능 및 자원 사용을 모니터하고 조정할 수 있는 인터페이스를 제공하는 MQ 실체

비동기 통신 메시지 발신자가 발신 메소드가 반환될 때까지 기다릴 필요 없이 다른 작업을 진행할 수 있는 통신 모드

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 JMS 클라이언트 사용자가 연결, 대상과 같은 MQ 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 사용합니다. 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) MQ 메시지 서버와의 활성 연결. 대기열 연결이거나 주제 연결이 될 수 있습니다. 2) MQ 메시지 서버의 기반이 되는 연결을 사용하여 메시지를 생성하고 사용하는 세션의 팩토리

연결 팩토리 클라이언트가 MQ 메시지 서버와 연결을 생성할 때 사용하는 관리 객체. 대기열 연결 팩토리 객체나 주제 연결 팩토리 객체가 될 수 있습니다.

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 모드 메시징의 신뢰성 지표. 메시지가 단 한 차례 전달되어 성공적으로 사용되는지(지속성 전달 모드) 또는 최대 1회 전달되는지(비지속성 전달 모드) 여부

전달 정책 둘 이상의 메시지 사용자가 등록된 경우 대기열에서 메시지 경로를 지정하는 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 런타임 MQ 클라이언트 런타임 참조

클라이언트 식별자 연결 및 그 객체를 JMS 클라이언트를 대신하여 MQ 메시지 서버가 관리하는 상태와 연관시키는 식별자

클러스터 함께 작동하면서 메시징 서비스를 제공하는, 상호 연결된 둘 이상의 브로커

트랜잭션 완료하거나 완전히 롤백해야 하는 작업 기본 단위

JMS (Java Message Service) JMS 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

JMS 클라이언트 다른 JMS 클라이언트와 상호 작용하면서 메시지 서비스를 사용하여 메시지를 교환하는 응용 프로그램(또는 소프트웨어 구성 요소)

MQ 메시지 서버 JMS 클라이언트와의 연결, 메시지 경로 지정 및 전달, 지속성, 보안 및 로깅을 비롯하여 MQ 메시징 시스템을 위해 전달 서비스를 제공하는 소프트웨어. 메시지 서버는 JMS 클라이언트가 보내는 메시지를 받고 또한 사용자 클라이언트에게 그 메시지를 전달하는 물리적 대상을 관리합니다.

MQ 클라이언트 런타임 JMS 클라이언트에게 MQ 메시지 서버와의 인터페이스를 제공하는 소프트웨어. 클라이언트 런타임은 클라이언트가 대상에게 메시지를 보내고 대상으로부터 메시지를 받는 데 필요한 모든 작업을 지원합니다.

색인

가

가입
  영구 관리 155
  영구 아이디 140
  영구 완전 삭제 157
객체 저장소
  위치 241
  정보 162
  파일 시스템 저장소 162
  파일 시스템 저장소 속성 169
  LDAP 서버 162
  LDAP 서버 속성 170
경로 지정, 메시지 라우터 *참조*
공급자 독립성
  관리 객체 76
  정보 37
관리 객체
  객체 저장소, 객체 저장소 *참조*
  공급자 독립성 76
  나열 178
  대기열, 대기열 *참조*
  대상, 대상 관리 객체 *참조*
  삭제 177
  속성 167
  업데이트 180
  연결 팩토리, 연결 팩토리 관리 객체 *참조*
  유형 34, 76, 162
  정보 33, 75
  조회 이름 164
  쿼리 179
  필요한 정보 165
  항목, 항목 *참조*
  XA 연결 팩토리, 연결 팩토리 관리 객체 *참조*
관리 도구
  관리 콘솔 84
  명령줄 유틸리티 84
  정보 84
관리 작업
  개발 환경 81
  작업 환경 82
구성 변경 기록 70
구성 요소
  EJB 34
  MDB 35
구성 파일
  기본값 114
  설정된 값 대체 114
  설치 114
  위치 114, 241
  인스턴스 114, 127, 147, 241
  편집 116
  config.properties 114
권한
  계산 194
  내장 데이터베이스 206
  데이터 저장소 55
  사용자 저장소 183
  액세스 제어 등록 정보 파일 58, 192
  키 저장소 223
  admin 서비스 59
  MQ 작업 58
  passfile 204
권한 부여

관리 192
사용자 그룹 59
정보 58
액세스 제어 파일 *참조*
기본 제공 지속성 55

다

대기열 65
관리 객체 추가 176
속성 153
자동 작성 67, 116
전달 정책, 대기열 전달 정책 *참조*
대기열 대상, 대기열 *참조*
대기열 전달 정책
단일 65
라운드 로빈 66
속성 153
정보 65
페일오버 66
대상
관리 152
나열 152
대기열, 대기열 *참조*
만들기 153
메시지 제거 152, 155
물리적 65
설명 44
속성 153
속성 업데이트 153, 154
액세스 제어 196
완전 삭제 152, 153
임시 68, 154
자동 작성 67, 197
정보 152, 154
항목, 항목 *참조*
대상 관리 객체 78
설명 33
속성 169
대상 자동 작성
등록 정보 67
정보 67
데이터 저장소
위치 241
재설정 124
정보 54
플랫 파일 55
JDBC 액세스 가능 56
데이터, MQ, 위치 241
도구, 관리, 관리 도구 *참조*
도메인 37
등록 정보
로거 63
메시지 라우터 53
보안 60
브로커 인스턴스 구성 116
브로커, 업데이트 147
연결 서비스 49
자동 작성 67
지속성 56
클러스터 구성 72
키 저장소 200
httpjms 연결 서비스 217
httpsjms 연결 서비스 226
JDBC 관련 207
LDAP 관련 189
디렉토리 변수
IMQ_HOME 20
IMQ_JAVAHOME 21
IMQ_VARHOME 20

라

로거
구성 변경 133
기본 구성 131
등록 정보 63
로그 메시지 리디렉션 134
롤오버 기준 134
메시지 형식 132
메트릭 정보 63, 135
범주 62
브로커 구성 요소 46

수준 62, 63, 122
아카이브 파일 63
정보 61
출력 채널 62, 133
콘솔에 쓰기 64, 124
로그 파일
기본 위치 63, 241
롤오버 기준 64
로깅, 로거 *참조*

마

마스터 브로커 70, 71
메모리 관리 52
메시지 30
경로 지정 및 전달 50
구조 31
대상에서 제거 152
만료 재생 이용 53
브로커 제한 53
사용 74
생성 74
설명 31
수신기 33, 75
순서 42
안정적인 전달 39
우선 순위 42
재전송 52
전달 게시/가입 37
전달 모델 31, 37
전달 모드, 전달 모드 *참조*
제어 51
제한 153
지속성 39, 52, 54
지점간 전달 37
필터링 42
확인 51, 167
SOAP 26
메시지 라우터
등록 정보 53
브로커 구성 요소 45
정보 50
메시지 사용자, 사용자 *참조*
메시지 생성자, 생성자 *참조*
메시지 서버
다중 브로커, 브로커 클러스터 *참조* 68
정보 44
메시지 서비스 30
메시지 서비스에 메시지를 보내는 메시지 생성자를 표시하는 다이어그램으로 메시지를 메시지 사용자에게 전달합니다. 30
메시지 수신기, 수신기 *참조*
메시지 전달 모델 31, 37
메시징 시스템
구조 30
메시지 서비스 30
MQ 구조 44
메트릭
보고 간격 123
요약 136
정보 62
명령 옵션 87
명령줄 구문 86
명령줄 유틸리티
공통 옵션 87
기본 구문 86
정보 84
imqcmd 85, 138
imqdbmgr 86, 211
imqkeytool 86, 199, 223
imqobjmgr 85, 163
imqsvcadmin 86, 236
imqusermgr 86, 184

바

방화벽 213
보안
객체 저장소 170
관리자, 보안 관리자 *참조*
권한 부여, 권한 부여 *참조*
암호화, 암호화 *참조*

인증, 인증 *참조*
보안 관리자
등록 정보 60
브로커 구성 요소 46
정보 57
분산 트랜잭션
정보 40
XA 자원 관리자 40, 157
XA 연결 팩토리 *참조*
브로커
구성 파일, 구성 파일 *참조*
다시 시작 54, 144
대상 등록 정보 자동 작성 67
등록 정보 147
로깅, 로거 *참조*
마스터 브로커 70
메시지 경로 지정, 메시지 라우터 *참조*
메시지 용량 53
메트릭, 메트릭 *참조*
보안 관리자, 보안 관리자 *참조*
상호 연결, 브로커 클러스터 *참조*
서비스 나열 150
시스템 자원 52
시작 120
액세스 제어, 권한 부여 *참조*
업데이트 145
여러 브로커의 클러스터, 브로커 클러스터 *참조*
연결 142
연결 서비스, 연결 서비스 *참조*
오류 복구 54
인스턴스 구성 등록 정보 116
인스턴스 이름 123
일시 중지 144
작업 45
정보 44
제어 144
종료 144
지속성 관리자, 지속성 관리자 *참조*
쿼리 145, 146
클러스터, 브로커 클러스터 *참조*
함께 연결 128
확인(Ack) 51
HTTP 지원 215
httpjms 연결 서비스 등록 정보 217
HTTPS, 지원 223
httpsjms 연결 서비스 등록 정보 226
JDBC 지원, JDBC 지원 *참조*
SSL 기반 서비스 시작 201
Windows 서비스, 실행 235
브로커 인스턴스, 브로커 *참조*
브로커 클러스터
개발 전용 환경 71
구성 등록 정보 71, 72, 126
구성 변경 기록 70, 71
구조 69
등록 정보 설정 127
마스터 브로커 70, 71
브로커 다시 시작 129
브로커 연결 128
브로커 추가 129
사용 이유 68
정보 전파 70
지정할 옵션 121
클러스터 구성 파일 72, 126
비밀번호
기본값 167
이름 지정 규약 186
인코딩 60
JDBC 204
LDAP 204
passfile, passfile *참조*
SSL 키 저장소 123, 200, 204
비밀번호 파일, passfile *참조*

사

사용권
로드 122
MQ 판 26
사용자 33
사용자 그룹
기본값 59
사전 정의됨 185
정보 58
할당 삭제 186

사용자 아이디
  기본값 167, 183
  형식 186
사용자 저장소
  관리 187
  사용자 그룹 186
  사용자 상태 186
  위치 182, 241
  유형 60
  정보 57
  채우기 187
  플랫 파일 182
  플랫폼 의존성 183
  LDAP 서버 189
생성자 33
서비스 유형
  ADMIN 46
  NORMAL 46
성능, 안정성 41
세션
  설명 33
  트랜잭션 39
  확인 옵션 39
수신기 33, 35
스레드 풀 관리자
  공유 스레드 48
  전용 스레드 48
  정보 48
시스템 등록 정보, 설정 78

아

안정적인 전달 39
암호화
  정보 59
  키 도구 및 59
  SSL 기반 서비스 198
액세스 규칙 194
액세스 제어 등록 정보 파일, 액세스 제어 파일 *참조*
액세스 제어 파일
  버전 192
  액세스 규칙 194
  용도 192
  위치 192, 241
  형식 193
엔터프라이즈판 27
연결
  설명 33
  재연결 168
  재연결 시도 168
  재연결 지연 168
연결 서비스
  다시 시작 149, 152
  등록 정보 49
  서비스 유형 46
  스레드 풀 관리자 48
  스레드 할당 151
  시작 시 활성화 49
  액세스 제어 60
  업데이트 149, 150
  연결 유형 46
  영향을 주는 명령 148
  일시 중지 149, 152
  정보 45
  정적 포트 49
  쿼리 149, 150
  포트 매퍼, 포트 매퍼 *참조*
  admin 47, 149
  HTTP, HTTP 연결 *참조*
  httpjms 47, 149
  HTTPS, HTTPS 연결 *참조*
  httpsjms 47, 149
  jms 46, 149
  SSL 기반 200
  ssladmin 47, 149, 198
  ssljms 46, 149, 198
연결 팩토리 관리 객체
  무시 78
  설명 33
  속성 77, 167
  정보 77
  추가 174
  ClientID 및 38
  JNDI 조회 34
영구 가입

나열 156
메시지 제거 156
완전 삭제 156
정보 37
ClientID 및 38
영구 가입자, 영구 가입자 *참조*
응용 프로그램 서버 36
이식성, 공급자 독립성 *참조*
인스턴스 구성 파일, 구성 파일 *참조*
인증
관리 182
정보 57
인증서 198, 223
임시 대상 68, 154
입력 파일 171

자

자체 서명된 인증서 198, 223
재전송 플래그 52
전달 게시/가입 37
전달 모드
비지속성 39
지속성 39
전달, 안정성 39
전송 프로토콜
HTTP 47, 149
TCP 46, 149
TLS 46, 149
제어 메시지 51
주제
관리 객체 추가 175
물리적 대상 66
속성 153
자동 작성 67, 116
정보 37
주제 대상, 주제 *참조*
지속성
기본 제공 55
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 56
브로커 구성 요소 46
정보 54
플러그 인 지속성 205
JDBC 데이터 저장소 207
지속성 메시지 39
지점간 전달 37

카

컨테이너
EJB 36
MDB 36
클라이언트
런타임 73
식별자(ClientID) 38
응용 프로그램, 클라이언트 응용 프로그램 *참조*
프로그래밍 모델 32
클라이언트 응용 프로그램
공급자 독립성 37
시스템 등록 정보 78
클러스터 구성 파일 72
클러스터, 브로커 클러스터 *참조*
키 도구 59
키 쌍
다시 생성 200
생성 199
키 저장소
등록 정보 200
파일 199, 200, 224

타

트랜잭션
  관리 157
  롤백 157
  분산, 분산 트랜잭션 *참조*
  완결 157
  정보 39, 157
  확인 및 52

파

판, 제품
  엔터프라이즈 27
  정보 26
  플랫폼 26
포트 매퍼
  정보 48
  포트 지정 49, 123
포트, 동적 할당 48
프로그래밍 도메인 37
프로토콜 유형
  HTTP 47, 149
  TCP 46, 149
  TLS 46, 149
프로토콜, 전송 프로토콜 *참조*
플랫폼판 26
플러그 인 지속성
  설정 205
  정보 56

하

확인
  대기 시간 167
  브로커 51, 167
  전달 51
  정보 39, 51
  클라이언트 51
  트랜잭션 및 52
환경 변수, 디렉토리 변수 *참조*

A

admin 연결 서비스 47, 149

C

config.properties 파일 114

H

HTTP 47, 149
HTTP 연결
  다중 브로커, 218
  요청 간격 217
  지원 213
  터널 서브릿, HTTP 터널 서브릿 *참조*
HTTP 전송 드라이버 213
HTTP 지원 구조 213
HTTP 터널 서브릿 214, 219
HTTP 프록시 213
httpjms 연결 서비스 47, 149
HTTPS 연결
  다중 브로커, 228
  요청 간격 226
  지원 213
  터널 서브릿, HTTPS 터널 서브릿 *참조*
HTTPS 지원 구조 213
HTTPS 터널 서브릿 203, 214
httpsjms 연결 서비스 47, 149

I

imq.accesscontrol.enabled 등록 정보 60, 116
imq.accesscontrol.file.filename 등록 정보 61, 116
imq.authentication.basic.user_repository 등록 정보 60, 116
imq.authentication.client.response.timeout 등록 정보 60, 116
imq.authentication.type 등록 정보 60, 116
imq.autocreate.queue 등록 정보 67, 116, 147
imq.autocreate.topic 등록 정보 67, 116, 147
imq.cluster.brokerlist 등록 정보 72, 126
imq.cluster.hostname 등록 정보 73, 127
imq.cluster.masterbroker 등록 정보 72, 126
imq.cluster.port 등록 정보 72, 126
imq.cluster.url 등록 정보 72, 116, 126, 147
imq.httpjms.http.pullPeriod 등록 정보 217
imq.httpjms.http.servletHost 등록 정보 217
imq.httpjms.http.servletPort 등록 정보 217
imq.httpsjms.https.pullPeriod 등록 정보 226
imq.httpsjms.https.servletHost 등록 정보 226
imq.httpsjms.https.servletPort 등록 정보 226
imq.keystore.dirpath 등록 정보 200
imq.keystore.file.name 등록 정보 200
imq.keystore.password 등록 정보 200, 204
imq.log.console.output 등록 정보 64, 116
imq.log.console.output.stream 등록 정보 64, 116
imq.log.console.syslog.facility 등록 정보 64, 117
imq.log.console.syslog.identity 등록 정보 64, 117
imq.log.console.syslog.logconsole 등록 정보 64, 117
imq.log.console.syslog.logpid 등록 정보 64, 117
imq.log.console.syslog.output 등록 정보 65, 117
imq.log.file.dirpath 등록 정보 64, 117
imq.log.file.filename 등록 정보 64, 117
imq.log.file.output 등록 정보 63, 117
imq.log.file.rolloverbytes 등록 정보 64, 117, 147
imq.log.file.rolloversecs 등록 정보 64, 117, 147
imq.log.level 등록 정보 63, 117, 147
imq.message.expiration.interval 등록 정보 53, 117
imq.message.max_size 등록 정보 53, 117, 147
imq.metrics.enabled 등록 정보 63, 117
imq.metrics.interval 등록 정보 63, 117
imq.passfile.dirpath 등록 정보 61, 117
imq.passfile.enabled 등록 정보 61, 117
imq.passfile.name 등록 정보 61, 117
imq.persist.file.destination.file.size 등록 정보 56, 117
imq.persist.file.message.cleanup 등록 정보 56, 117
imq.persist.file.message.fdpool.limit 등록 정보 57, 118
imq.persist.file.message.filepool.cleanratio 등록 정보 56, 118
imq.persist.file.message.filepool.limit 등록 정보 56, 118
imq.persist.file.sync.enabled 등록 정보 57, 118
imq.persist.jdbc.brokerid 등록 정보 207
imq.persist.jdbc.closedburl 등록 정보 208
imq.persist.jdbc.createdburl 등록 정보 208
imq.persist.jdbc.driver 등록 정보 207
imq.persist.jdbc.needpassword 등록 정보 208
imq.persist.jdbc.opendburl 등록 정보 208
imq.persist.jdbc.password 등록 정보 204, 209
imq.persist.jdbc.user 등록 정보 208
imq.persist.store 등록 정보 56, 118, 207
imq.portmapper.port 등록 정보 49, 118, 147
imq.queue.deliverypolicy 등록 정보 67, 118, 147
imq.redelivered.optimization 등록 정보 53, 118
imq.*resource_state*.threshold 등록 정보 53, 118
imq.service.activelist 등록 정보 49, 118
imq.*service_name*.accesscontrol.enabled 등록 정보 60, 118
imq.*service_name*.accesscontrol.file.filename 등록 정보 61, 118
imq.*service_name*.authentication.type 등록 정보 60, 118
imq.*service_name*.max_threads 등록 정보 49, 118
imq.*service_name*.min_threads 등록 정보 49, 119
imq.*service_name.protocol_type*.hostname 등록 정보 50, 119

imq.*service_name.protocol_type*.port 등록 정보 50, 119
imq.*service_name*.threadpool_model 등록 정보 49, 119
imq.shared.connectionMonitor_limit 등록 정보 49, 119
imq.system.max_count 등록 정보 53, 119, 147
imq.system.max_size 등록 정보 53, 119, 147
imq.transaction.autorollback 등록 정보 53, 119, 159
imq.user_repository.ldap.base 등록 정보 190
imq.user_repository.ldap.gidattr 등록 정보 190
imq.user_repository.ldap.grpbase 등록 정보 190
imq.user_repository.ldap.grpfilter 등록 정보 191
imq.user_repository.ldap.grpsearch 등록 정보 190
imq.user_repository.ldap.memattr 등록 정보 190
imq.user_repository.ldap.password 등록 정보 190, 204
imq.user_repository.ldap.principal 등록 정보 189
imq.user_repository.ldap.server 등록 정보 189
imq.user_repository.ldap.ssl.enabled 등록 정보 191
imq.user_repository.ldap.timeout 등록 정보 191
imq.user_repository.ldap.uidattr 등록 정보 190
imq.user_repository.ldap.usrfilter 등록 정보 190
IMQ_HOME 디렉토리 변수 20
IMQ_JAVAHOME 디렉토리 변수 21
IMQ_VARHOME 디렉토리 변수 20
imqAckOnAcknowledge 속성 167
imqAckOnProduce 속성 167
imqAckTimeout 속성 167
imqbrokerd 명령
  명령 구문 120
  사용 120
  옵션 121
  정보 85
imqBrokerHostName 속성 167
imqBrokerHostPort 속성 167
imqBrokerServicePort 속성 167
imqcmd 명령
  대상 관리 152
  명령 구문 138
  브로커에 보안 연결 141, 202
  브로커에 연결 142
  옵션 140
  용도 138, 163, 184
  정보 85
  트랜잭션 관리 157
  하위 명령 138
imqConfiguredClientID 속성 167
imqConnectionType 속성 167
imqConnectionURL 속성 167
imqdbmgr 명령
  명령 구문 210
  옵션 211
  정보 86
  하위 명령 211
imqDefaultPassword 속성 167
imqDefaultUsername 속성 167
imqDestinationDescription 속성 78, 169
imqDestinationName 속성 78, 169
imqDisableSetClientID 속성 167
imqFlowControlCount 속성 167
imqFlowControlIsLimited 속성 167
imqFlowControlLimit 속성 167
imqJMSDeliveryMode 속성 168
imqJMSExpiration 속성 168
imqJMSPriority 속성 168
imqkeytool 명령
  명령 구문 199, 223
  사용 199, 223
  정보 86
imqLoadMaxToServerSession 속성 168
imqobjmgr 명령
  명령 구문 163
  옵션 164
  정보 85
  하위 명령 163
imqOverrideJMSDeliveryMode 속성 168
imqOverrideJMSExpiration 속성 168
imqOverrideJMSHeadersToTemporaryDestinations 속성 168
imqOverrideJMSPriority 속성 168

imqQueueBrowserMaxMessagesPerRetrieve 속성 168
imqQueueBrowserRetrieveTimeout 속성 168
imqReconnect 속성 168
imqReconnectDelay 속성 168
imqReconnectRetries 속성 168
imqSetJMSXAppID 속성 168
imqSetJMSXConsumerTXID 속성 168
imqSetJMSXProducerTXID 속성 168
imqSetJMSXRcvTimestamp 속성 168
imqSetJMSXUserID 속성 168
imqSSLIsHostTrusted 속성 168
imqsvcadmin 명령
  명령 구문 236
  옵션 237
  용도 236
  정보 86
  하위 명령 236
imqusermgr 명령
  명령 구문 184
  비밀번호 186
  사용자 아이디 186
  옵션 184
  정보 86
  하위 명령 184

J

J2EE 응용 프로그램
  EJB 사양 34
  JMS 34
  Message-Driven Bean, Message Driven-Bean *참조*
JDBC 지원
  드라이버 205, 207
  설정 205
  정보 56
JDK
  경로 지정 122
  경로 지정 옵션 140, 164, 237
JMS
  메시지 구조 31
  사양 31
  프로그래밍 모델 32
JMS 사양 23, 25
jms 연결 서비스 46, 149
JNDI
  객체 저장소 85, 162
  객체 저장소 속성 166, 169
  관리 객체 및 34, 37
  위치(공급자 URL) 169
  조회 75, 78, 103, 165
  초기 컨텍스트 169
  Message-Driven Beans 36
  MQ 지원 26
JRE, 경로 지정 122

L

LDAP 서버
  객체 저장소 속성 170
  사용자 저장소 액세스 189

M

MDB, Message-Driven Bean *참조*
Message-Driven Bean
  응용 프로그램 서버 지원 36
Message-Driven Beans
  배치 설명자 36
  정보 35
  MDB 컨테이너 36

P

passfile
  명령줄 옵션 123
  브로커 구성 등록 정보 61
  사용 204

위치 204, 241

S

Secure Socket Layer 표준, SSL *참조*
Simple Object Access Protocol, SOAP *참조*
SOAP 26
SSL
  서비스, 설정 181
  암호화 198
  연결 서비스 및 46, 149
  정보 59
  HTTP에서 203
  TCP/IP에서 198
SSL 기반 서비스
  시작 201
SSL 기반 연결 서비스
  설정 198
ssladmin 연결 서비스 47, 149
  구성 201
  설정 198
ssljms 연결 서비스 46, 149
  구성 201
  설정 198
syslog 63, 134

T

TCP 46, 149
TLS 46, 149

W

Windows 서비스, 브로커 실행 235

X

XA 연결 팩토리
  정보 41
  연결 팩토리 관리 객체 *참조*
XA 자원 관리자, 분산 트랜잭션 *참조*

x