



Sun Java System Application Server Enterprise Edition 8.2 고가용성 관리 설명서



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

부품 번호: 820-0853
2007년 4월

Copyright 2007 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. 모든 권리는 저작권자의 소유입니다.

Sun Microsystems, Inc.는 이 문서에 설명된 제품의 기술 관련 지적 재산을 소유합니다. 특히 이 지적 재산권에는 하나 이상의 미국 특허권 또는 미국 및 다른 국가에서 특허 출원 중인 응용 프로그램이 포함될 수 있습니다.

미국 정부의 권리 - 상용 소프트웨어. 정부 사용자는 Sun Microsystems, Inc. 표준 사용권 계약과 해당 FAR 규정 및 보충 규정을 준수해야 합니다.

이 배포에는 타사에서 개발한 자료가 포함되어 있을 수 있습니다.

제품 중에는 캘리포니아 대학에서 허가한 Berkeley BSD 시스템에서 파생된 부분이 포함되어 있을 수 있습니다. UNIX는 미국 및 다른 국가에서 X/Open Company, Ltd.를 통해 독점적으로 사용권이 부여되는 등록 상표입니다.

Sun, Sun Microsystems, Sun 로고, Solaris 로고, Java Coffee Cup 로고, docs.sun.com, Java 및 Solaris는 미국 및 다른 국가에서 Sun Microsystems, Inc.의 상표 또는 등록 상표입니다. 모든 SPARC 상표는 사용 허가를 받았으며 미국 및 다른 국가에서 SPARC International, Inc.의 상표 또는 등록 상표입니다. SPARC 상표를 사용하는 제품은 Sun Microsystems, Inc.에서 개발한 구조에 기반을 두고 있습니다.

OPEN LOOK 및 SunTM 그래픽 사용자 인터페이스(GUI)는 Sun Microsystems, Inc.가 자사의 사용자 및 정식 사용자로 개발했습니다. Sun은 컴퓨터 업계를 위한 시각적 또는 그래픽 사용자 인터페이스(GUI)의 개념을 연구 개발한 Xerox사의 선구적인 노력을 높이 평가하고 있습니다. Sun은 Xerox와 Xerox 그래픽 사용자 인터페이스(GUI)에 대한 비독점적 사용권을 보유하고 있습니다. 이 사용권은 OPEN LOOK GUI를 구현하는 Sun의 정식 사용자에게도 적용되며 그렇지 않은 경우에는 Sun의 서면 사용권 계약을 준수해야 합니다.

이 설명서에서 다루는 제품과 수록된 정보는 미국 수출 관리법에 의해 규제되며 다른 국가의 수출 또는 수입 관리법의 적용을 받을 수도 있습니다. 이 제품과 정보를 직간접적으로 핵무기, 미사일 또는 생화학 무기에 사용하거나 핵과 관련하여 해상에서 사용하는 것은 엄격하게 금지합니다. 거부된 사람과 특별히 지정된 국민 목록을 포함하여 미국의 수출 금지 국가 또는 미국의 수출 제의 목록에 나와 있는 대상으로의 수출이나 재수출은 엄격하게 금지됩니다.

설명서는 "있는 그대로" 제공되며, 법률을 위반하지 않는 범위 내에서 상품성, 특정 목적에 대한 적합성 또는 비침해에 대한 묵시적인 보증을 포함하여 모든 명시적 또는 묵시적 조건, 표현 및 보증을 배제합니다.

목차

머리말	15
1 Application Server의 고가용성	21
고가용성 개요	21
고가용성 세션 지속성	21
고가용성 Java Message Service	22
RMI-IIOP 로드 균형 조정 및 페일오버	22
추가 정보	23
Application Server에서 고가용성을 제공하는 방법	24
로드 밸런서 플러그인	24
고가용성 데이터베이스	24
고가용성 클러스터	25
오류 복구	26
Sun Cluster 사용	26
수동 복구	27
Netbackup 사용	28
Domain Administration Server 다시 만들기	29
▼도메인 관리 서버 마이그레이션	29
2 고가용성 데이터베이스 설치 및 설정	33
HADB 설정 준비	33
필수 조건 및 제한 사항	33
네트워크 중복 구성	34
공유 메모리 및 세마포 구성	37
▼ Solaris에서 공유 메모리 및 세마포 구성	37
▼ Linux에서 공유 메모리 구성	39
시스템 클럭 동기화	39

설치	40
HADB 설치	40
노드 수퍼바이저 프로세스 권한	41
▼ 노드 수퍼바이저 프로세스 루트 권한 부여	42
고가용성 설정	42
▼ 고가용성을 위한 시스템 준비	43
HADB 관리 에이전트 시작	43
고가용성을 위한 클러스터 구성	43
고가용성을 위한 응용 프로그램 구성	44
클러스터 다시 시작	44
Web Server 다시 시작	44
▼ 로드 밸런서로 작동하는 Web Server 인스턴스 정리	45
HADB 업그레이드	45
▼ HADB를 새 버전으로 업그레이드	45
HADB 패키지 등록	46
HADB 패키지 등록 취소	47
관리 에이전트 시작 스크립트 교체	48
▼ HADB 업그레이드 확인	48
 3 고가용성 데이터베이스 관리	51
HADB 관리 에이전트 사용	51
관리 에이전트 시작	51
관리 에이전트 명령 구문	56
관리 에이전트 구성 사용자 정의	58
▼ HADB 호스트의 관리 에이전트 구성을 사용자 정의하는 방법	58
hadbm 관리 명령 사용	59
명령 구문	60
보안 옵션	60
일반 옵션	62
환경 변수	62
HADB 구성	64
관리 도메인 만들기	64
데이터베이스 만들기	65
▼ 데이터베이스를 만드는 방법	65
구성 속성 보기 및 수정	70

JDBC 연결 풀 구성	75
HADB 관리	78
도메인 관리	78
노드 관리	79
데이터베이스 관리	82
세션 데이터 손상에서 복원	86
▼ 세션 저장소를 일관된 상태로 되돌리는 방법	86
HADB 확장	87
기존 노드에 저장 공간 추가	87
시스템 추가	88
▼ 기존 HADB 인스턴스에 새 시스템을 추가하는 방법	88
노드 추가	88
데이터베이스 재조각화	90
데이터베이스를 다시 만들어 노드 추가	91
▼ 데이터베이스를 다시 만들어 노드를 추가하는 방법	91
HADB 모니터링	92
HADB 상태 가져오기	92
장치 정보 가져오기	95
런타임 자원 정보 가져오기	96
HADB 시스템 유지 관리	99
▼ 단일 시스템에서 유지 관리를 수행하는 방법	99
▼ 모든 HADB 시스템에 계획된 유지 관리를 수행하는 방법	100
▼ 모든 HADB 시스템에 계획된 유지 관리를 수행하는 방법	100
▼ 실패가 발생한 경우 계획되지 않은 유지 관리를 수행하는 방법	101
내역 파일 지우기 및 아카이브	101
4 로드 균형 조정을 사용하도록 웹 서버 구성	103
Sun Java System Web Server 사용	103
Apache Web Server 사용	104
Apache Web Server 사용 요구 사항	105
로드 밸런서 플러그인을 설치하기 전 Apache 구성	107
▼ SSL 인식 Apache 설치	107
로드 밸런서 플러그인 설치 프로그램에서 수정한 사항	110
로드 밸런서 플러그인을 설치한 후 Apache 구성	111
▼ 로드 밸런서에서 작동하도록 Apache 보안 파일 구성	111

▼ Apache 2용 보안 인증서 만들기	111
Solaris 및 Linux에서 Apache 시작	113
Microsoft IIS 사용	113
▼ 로드 밸런서 플러그인을 사용하도록 Microsoft IIS 구성	113
자동으로 구성된 sun-passthrough 등록 정보	115
5 HTTP 로드 균형 조정 구성	117
HTTP 로드 밸런서 작동 방식	117
할당된 요청 및 할당되지 않은 요청	118
HTTP 로드 균형 조정 알고리즘	118
샘플 응용 프로그램	119
HTTP 로드 균형 조정 설정	119
로드 균형 조정 설정을 위한 필수 조건	119
HTTP 로드 밸런서 배포	120
로드 균형 조정 설정 절차	120
▼ 로드 균형 조정 설정	121
로드 밸런서 구성	121
HTTP 로드 밸런서 구성 만들기	122
HTTP 로드 밸런서 참조 만들기	123
로드 균형 조정을 위해 서버 인스턴스 활성화	123
로드 균형 조정을 위해 응용 프로그램 활성화	123
HTTP 상태 검사기 만들기	124
로드 밸런서 구성 파일 내보내기	125
▼ 로드 밸런서 구성 내보내기	125
로드 밸런서 구성 변경	126
동적 재구성 활성화	126
서버 인스턴스 또는 클러스터 비활성화(정지)	127
▼ 서버 인스턴스나 클러스터 비활성화	127
응용 프로그램 비활성화(정지)	127
▼ 응용 프로그램 비활성화	128
HTTP 및 HTTPS 페일오버 구성	128
로드 밸런서와 함께 리디렉션 사용	129
멥등원(Idempotent) URL 구성	132
다중 웹 서버 인스턴스 구성	133
▼ 다중 웹 서버 인스턴스 구성	133

가용성 손실 없이 응용 프로그램 업그레이드	134
응용 프로그램 호환성	134
단일 클러스터에서 업그레이드	135
▼ 단일 클러스터에서 응용 프로그램 업그레이드	135
여러 클러스터에서 업그레이드	136
▼ 두 개 이상의 클러스터에서 호환되는 응용 프로그램 업그레이드	136
호환되지 않는 응용 프로그램 업그레이드	138
▼ 또 다른 클러스터를 만들어 호환되지 않는 응용 프로그램 업그레이드	138
HTTP 로드 밸런서 플러그인 모니터링	140
로그 메시지 구성	140
로그 메시지 유형	140
로드 밸런서 로깅 활성화	142
▼ 로드 밸런서 로깅 설정	142
모니터링 메시지 이해	143
6 Application Server 클러스터 사용	145
클러스터 개요	145
클러스터 작업	145
▼ 클러스터 만들기	146
▼ 클러스터에 대한 서버 인스턴스 만들기	147
▼ 클러스터 구성	148
▼ 클러스터링된 인스턴스 시작, 중지 및 삭제	149
▼ 클러스터의 서버 인스턴스 구성	149
▼ 클러스터에 대한 응용 프로그램 구성	150
▼ 클러스터에 대한 자원 구성	150
▼ 클러스터 삭제	151
▼ EJB 타이머 마이그레이션	151
▼ 서비스 손실 없이 구성 요소 업그레이드	152
7 구성 관리	155
구성 사용	155
구성	155
default-config 구성	156
인스턴스 또는 클러스터에 대한 구성	156
고유한 포트 번호 및 구성	157

명명된 구성 작업	158
▼ 명명된 구성 만들기	158
명명된 구성의 등록 정보 편집	158
▼ 명명된 구성의 등록 정보 편집	159
▼ 구성을 참조하는 인스턴스의 포트 번호 편집 방법	160
▼ 명명된 구성의 대상 보기	160
▼ 명명된 구성 삭제	161
8 노드 에이전트 구성	163
노드 에이전트란?	163
노드 에이전트 배포	165
▼ 노드 에이전트를 온라인으로 배포	165
▼ 노드 에이전트를 오프라인으로 배포	166
노드 에이전트 및 도메인 관리 서버 동기화	167
노드 에이전트 동기화	167
서버 인스턴스 동기화	168
라이브러리 파일 동기화	169
고유한 설정 및 구성 관리	169
대용량 응용 프로그램 동기화	170
노드 에이전트 로그 보기	171
노드 에이전트 작업	171
노드 에이전트 작업을 수행하는 방법	171
노드 에이전트 자리 표시자	172
▼ 노드 에이전트 자리 표시자 만들기	172
노드 에이전트 만들기	173
노드 에이전트 시작	174
노드 에이전트 중지	174
노드 에이전트 삭제	175
▼ 일반 노드 에이전트 정보 보기	175
▼ 노드 에이전트 구성 삭제	176
▼ 노드 에이전트 구성 편집	176
▼ 노드 에이전트 영역 편집	177
▼ JMX에 대해 노드 에이전트 Listener 편집	177

9	고가용성 세션 지속성 및 페일오버 구성	179
	세션 지속성 및 페일오버 개요	179
	요구 사항	179
	제한 사항	180
	샘플 응용 프로그램	180
	고가용성 세션 지속성 설정	181
	▼ 고가용성 세션 지속성 설정	181
	세션 가용성 활성화	182
	HTTP 세션 페일오버	183
	웹 컨테이너에 대한 가용성 구성	183
	▼ 관리 콘솔을 사용하여 웹 컨테이너에 대한 가용성 활성화	184
	개별 웹 응용 프로그램에 대한 가용성 구성	185
	세션 페일오버와 함께 단일 사인 온 사용	186
	Stateful Session Bean 페일오버	187
	EJB 컨테이너에 대한 가용성 구성	188
	▼ EJB 컨테이너에 대한 가용성 활성화	188
	개별 응용 프로그램 또는 EJB 모듈에 대한 가용성 구성	190
	개별 Bean에 대한 가용성 구성	190
	검사점을 지정할 메소드 지정	191
10	Java Message Service 로드 균형 조정 및 페일오버	193
	Java Message Service 개요	193
	샘플 응용 프로그램	193
	추가 정보	194
	Java Message Service 구성	194
	Java Message Service 통합	195
	JMS 호스트 목록	196
	연결 풀링 및 페일오버	197
	로드 균형 조정된 메시지 흐름	198
	Application Server에서 MQ 클러스터 사용	198
	▼ Application Server 클러스터가 있는 MQ 클러스터 활성화	198
11	RMI-IIOP 로드 균형 조정 및 페일오버	203
	개요	203
	요구 사항	204

알고리즘	204
샘플 응용 프로그램	205
RMI-IIOP 로드 균형 조정 및 페일오버 설정	205
▼ Application Client Container에 대해 RMI-IIOP 로드 균형 조정 설정	205
▼ 독립 실행형 클라이언트에 대해 RMI-IIOP 로드 균형 조정 및 페일오버 설정	207
 색인	 209

표

표 2-1	hadbm registerpackage 옵션	47
표 3-1	관리 에이전트 공통 옵션	57
표 3-2	관리 에이전트 서비스 옵션(Windows에만 해당)	57
표 3-3	구성 파일 설정	58
표 3-4	hadbm 보안 옵션	61
표 3-5	hadbm 일반 옵션	62
표 3-6	HADB 옵션 및 환경 변수	62
표 3-7	hadbm create 옵션	66
표 3-8	구성 속성	71
표 3-9	HADB 연결 풀 설정	76
표 3-10	HADB 연결 풀 등록 정보	76
표 3-11	HADB JDBC 자원 설정	78
표 3-12	hadbm clear 옵션	85
표 3-13	hadbm addnodes 옵션	89
표 3-14	HADB 상태	93
표 3-15	hadbm resourceinfo 명령 옵션	97
표 5-1	로드 밸런서 구성 매개 변수	122
표 5-2	상태 검사기 매개 변수	124
표 5-3	상태 검사기 수동 등록 정보	125
표 8-1	원격 서버 인스턴스 사이에서 동기화되는 파일과 디렉토리	168
표 8-2	노드 에이전트 작업을 수행하는 방법	172

코드 예

예 2-1	다중 경로 설정	35
예 2-2	HADB 등록 취소의 예	48
예 3-1	hadbm 명령의 예	60
예 3-2	HADB 관리 도메인 만들기	65
예 3-3	데이터베이스 만들기의 예	68
예 3-4	hadbm get 사용의 예	70
예 3-5	연결 풀 만들기	77
예 3-6	노드 시작의 예	81
예 3-7	노드 중지의 예	81
예 3-8	노드 재시작의 예	82
예 3-9	데이터베이스 시작의 예	82
예 3-10	데이터베이스 중지의 예	83
예 3-11	데이터베이스 제거의 예	86
예 3-12	데이터 장치 크기 설정의 예	88
예 3-13	노드 추가의 예	89
예 3-14	데이터베이스 재조각화의 예	91
예 3-15	HADB 상태 가져오기의 예	93
예 3-16	장치 정보 가져오기의 예	96
예 3-17	데이터 버퍼 풀 정보의 예	97
예 3-18	잠금 정보의 예	98
예 3-19	로그 버퍼 정보의 예	98
예 3-20	내부 로그 버퍼 정보의 예	99
예 8-1	노드 에이전트 작성 예	173
예 9-1	가용성이 활성화된 EJB 배포 설명자의 예	190
예 9-2	메소드 검사점을 지정하는 EJB 배포 설명자의 예	191

머리말

본 설명서는 HTTP 로드 균형 조정, 세션 지속성 및 페일오버,고가용성 데이터베이스(HADB)를 포함한 Application Server Enterprise Edition의 고가용성 기능에 대해 설명합니다.

이 장에서는 전체 Sun Java™ System Application Server 설명서 세트에 대한 정보와 규칙이 제공됩니다.

Application Server 설명서 세트

Application Server 설명서 세트에서는 배포 계획 및 시스템 설치에 대해 설명합니다. 독립 실행형 Application Server 설명서에 대한 URL(Uniform Resource Locator)은 <http://docs.sun.com/app/docs/coll/1310.4>입니다. Sun Java Enterprise System(Java ES) Application Server 설명서에 대한 URL은 <http://docs.sun.com/app/docs/coll/1310.3>입니다. Application Server에 대한 소개 내용을 보려면 다음 표에 나열된 설명서를 순서대로 참조하십시오.

표 P-1 Application Server 설명서 세트에 포함된 설명서

설명서 제목	설명
릴리스 노트	소프트웨어 및 설명서 관련 최신 정보로 지원되는 하드웨어, 운영 체제, JDK™(Java Development Kit) 및 데이터베이스 드라이버를 표를 기반으로 종합적으로 요약합니다.
Quick Start Guide(빠른 시작 설명서)	Application Server 제품 시작 방법에 대해 설명합니다.
설치 설명서	소프트웨어와 해당 구성 요소 설치에 대해 설명합니다.
Deployment Planning Guide	사용자 시스템 요구 사항과 기업 평가를 통해 Application Server를 사용자 사이트에 가장 적합한 방식으로 배포하는 방법에 대해 설명합니다. 서버 배포 시 알아야 할 일반적인 문제와 관심을 기울여야 할 사항에 대해서도 설명합니다.
Developer's Guide	Application Server에서 실행할 J2EE 구성 요소 및 API의 개방형 Java 표준 모델을 따르는 Java 2 Platform, Enterprise Edition(J2EE™ 플랫폼) 응용 프로그램의 생성 및 구현에 대해 설명합니다. 개발자 도구, 보안, 디버깅, 배포 및 라이프 사이클 모듈 만들기에 대한 정보를 제공합니다.

표 P-1 Application Server 설명서 세트에 포함된 설명서 (계속)

설명서 제목	설명
J2EE 1.4 Tutorial	J2EE 응용 프로그램을 개발하기 위한 J2EE 1.4 플랫폼 기술 및 API를 사용하는 방법에 대해 설명합니다.
관리 설명서	관리 콘솔에서 Application Server 하위 시스템 및 구성 요소를 구성, 관리 및 배포하는 방법에 대해 설명합니다.
고가용성 관리 설명서	고가용성 데이터베이스를 위한 설치 후 구성 및 관리 방법에 대해 설명합니다.
Administration Reference	Application Server 구성 파일인 domain.xml을 편집하는 방법에 대해 설명합니다.
Upgrade and Migration Guide	응용 프로그램 특히 Application Server 6.x 및 7에서 새로운 Application Server 프로그래밍 모델로 마이그레이션하는 방법에 대해 설명합니다. 제품 사양과 호환되지 않을 수 있는 제품 릴리스와 구성 옵션 간의 차이점에 대해서도 설명합니다.
Performance Tuning Guide	성능 향상을 위해 Application Server를 조정하는 방법에 대해 설명합니다.
Troubleshooting Guide	Application Server의 문제를 해결하는 방법에 대해 설명합니다.
Error Message Reference	Application Server의 오류 메시지를 해결하는 방법에 대해 설명합니다.
Reference Manual	Application Server와 함께 사용할 수 있는 유틸리티 명령에 대해 설명합니다(설명서 페이지 스타일로 작성). asadmin 명령줄 인터페이스를 포함합니다.

관련 설명서

Application Server는 단독으로 구입하거나 네트워크 또는 인터넷 환경에서 배포된 엔터프라이즈 응용 프로그램을 지원하는 소프트웨어 인프라인 Java ES의 구성 요소로 구입할 수 있습니다. Application Server를 Java ES의 구성 요소로 구입한 경우에는 <http://docs.sun.com/coll/1286.2>의 시스템 설명서를 잘 이해해야 합니다. Java ES 및 해당 구성 요소에 대한 모든 설명서의 URL은 <http://docs.sun.com/prod/entsys.5> 및 <http://docs.sun.com/prod/entsys.5?l=ko>입니다.

기타 Sun Java System 서버 설명서를 보려면 다음을 참조하십시오.

- Message Queue 설명서
- Directory Server 설명서
- Web Server 설명서

또한 다음과 같은 자원을 사용할 수 있습니다.

- J2EE 1.4 사양(<http://java.sun.com/j2ee/1.4/docs/index.html>)
- J2EE 1.4 Tutorial(<http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html>)
- J2EE Blueprints(<http://java.sun.com/reference/blueprints/index.html>)

기본 경로 및 파일 이름

다음 표에서는 본 설명서에서 사용한 기본 경로와 파일 이름에 대해 설명합니다.

표 P-2 기본 경로 및 파일 이름

자리 표시자	설명	기본값
<i>install-dir</i>	Application Server의 기본 설치 디렉토리를 나타냅니다.	<p>Solaris™ 플랫폼에 Sun Java Enterprise System(Java ES)을 설치한 경우:</p> <p><code>/opt/SUNWappserver/appserver</code></p> <p>Linux 플랫폼에 Java ES를 설치한 경우:</p> <p><code>/opt/sun/appserver/</code></p> <p>기타 Solaris 및 Linux 설치, 루트가 아닌 사용자의 경우:</p> <p><code>user's home directory/SUNWappserver</code></p> <p>기타 Solaris 및 Linux 설치, 루트 사용자의 경우:</p> <p><code>/opt/SUNWappserver</code></p> <p>Windows, 모든 설치:</p> <p><code>SystemDrive:\Sun\AppServer</code></p>
<i>domain-root-dir</i>	모든 도메인을 포함한 디렉토리를 나타냅니다.	<p>Solaris 플랫폼에 Java ES를 설치한 경우:</p> <p><code>/var/opt/SUNWappserver/domains/</code></p> <p>Linux 플랫폼에 Java ES를 설치한 경우:</p> <p><code>/var/opt/sun/appserver/domains/</code></p> <p>다른 모든 설치:</p> <p><code>install-dir/domains/</code></p>
<i>domain-dir</i>	<p>도메인용 디렉토리를 나타냅니다.</p> <p>구성 파일에서 <i>domain-dir</i>이 다음과 같이 표시되어 있을 수 있습니다.</p> <p><code>\${com.sun.aas.instanceRoot}</code></p>	<i>domain-root-dir/domain-dir</i>
<i>instance-dir</i>	서버 인스턴스용 디렉토리를 나타냅니다.	<i>domain-dir/instance-dir</i>

활자체 규약

다음 표에서는 본 설명서에 사용된 활자체 규약에 대해 설명합니다.

표 P-3 활자체 규약

서체	의미	예
AaBbCc123	명령, 파일 및 디렉토리의 이름, 그리고 컴퓨터 화면에 출력되는 내용입니다.	.login 파일을 편집하십시오. ls -a를 사용하여 모든 파일을 나열하십시오. machine_name% you have mail.
AaBbCc123	화면 상의 컴퓨터 출력과는 반대로 사용자가 직접 입력하는 사항입니다.	machine_name% su Password:
AaBbCc123	명령줄 자리 표시자: 실제 이름이나 값으로 대체됩니다.	파일을 삭제하려면 rmfilename을 입력하십시오.
AaBbCc123	책 제목, 새로 나오는 용어, 강조 표시할 단어입니다. (강조 표시된 일부 항목은 온라인에서 볼드로 표시됩니다.)	사용자 설명서 의 6장을 읽으십시오. cache는 로컬로 저장된 복사본입니다. 파일을 저장하지 마십시오 .

기호 규칙

다음 표에서는 본 설명서에 사용된 기호에 대해 설명합니다.

표 P-4 기호 규칙

기호	설명	예	의미
[]	선택 인수 및 명령 옵션을 포함합니다.	ls [-l]	-l 옵션은 사용하지 않아도 됩니다.
{ }	필수 명령 옵션에 대한 일련의 선택 항목을 포함합니다.	-d {y n}	-d 옵션에서는 y 인수나 n 인수를 사용해야 합니다.
\${ }	변수 참조를 나타냅니다.	\${com.sun.javaRoot}	com.sun.javaRoot 변수 값을 참조합니다.
-	동시에 입력하는 여러 키를 결합합니다.	Ctrl-A	Ctrl 키를 누른 채로 A 키를 누릅니다.
+	연속해서 입력하는 여러 키를 결합합니다.	Ctrl+A+N	Ctrl 키를 눌렀다가 놓은 다음 후속 키를 누릅니다.

표 P-4 기호 규칙 (계속)

기호	설명	예	의미
→	그래픽 사용자 인터페이스의 메뉴 항목 선택을 나타냅니다.	파일 → 새로 만들기 → 템플릿	파일 메뉴에서 새로 만들기를 선택합니다. 새로 만들기 하위 메뉴에서 템플릿을 선택합니다.

설명서, 지원 및 교육

Sun 웹 사이트에서는 다음 추가 자원에 대한 정보가 제공됩니다.

- 설명서(<http://www.sun.com/documentation/>)
- 지원(<http://www.sun.com/support/>)
- 교육(<http://www.sun.com/training/>)

타사 웹 사이트 참조 사항

이 설명서에서는 추가 관련 정보를 제공하기 위해 타사 URL을 참조하기도 합니다.

주 - Sun은 이 설명서에 언급된 타사 웹 사이트의 가용성에 대해 책임지지 않습니다. Sun은 이러한 사이트나 자원을 통해 사용할 수 있는 내용, 광고, 제품 또는 기타 자료에 대해서는 보증하지 않으며 책임지지 않습니다. Sun은 해당 사이트 또는 자원을 통해 사용 가능한 내용, 제품 또는 서비스의 사용과 관련해 발생하거나 발생했다고 간주되는 손해나 손실에 대해 책임이나 의무를 지지 않습니다.

사용자 의견 환영

Sun은 설명서의 내용을 지속적으로 개선하고자 하며 사용자 여러분의 의견과 제안을 환영합니다. 사용자 의견을 보내시려면 <http://docs.sun.com>에서 의견 보내기를 누르십시오. 온라인 양식에서 문서 전체 제목과 부품 번호를 기입해 주십시오. 부품 번호는 해당 설명서의 제목 페이지나 문서의 URL에 있으며 7자리 또는 9자리 숫자로 되어 있습니다. 예를 들어, 이 설명서의 부품 번호는 820-0853입니다.

Application Server의 고가용성

이 장은 Sun Java System Application Server Enterprise Edition의 고가용성 기능에 대해 설명하며 다음 항목으로 구성되어 있습니다.

- 21 페이지 “고가용성 개요”
- 24 페이지 “Application Server에서 고가용성을 제공하는 방법”
- 26 페이지 “오류 복구”

고가용성 개요

고가용성 응용 프로그램 및 서비스는 하드웨어 및 소프트웨어 장애 발생 여부에 관계없이 해당 기능을 지속적으로 제공합니다. 시간의 99.999% 동안 사용할 수 있도록 되어 있기 때문에 이러한 응용 프로그램이 종종 **5개 9(five nines)**에 해당하는 안정성을 제공한다고 말합니다.

Application Server는 다음과 같은 고가용성 기능을 제공합니다.

- 고가용성 세션 지속성
- 고가용성 Java Message Service
- RMI-IIOP 로드 균형 조정 및 페일오버

고가용성 세션 지속성

Application Server는 HTTP 요청 및 세션 데이터(HTTP 세션 데이터 및 Stateful Session Bean 데이터)에 대해 고가용성을 제공합니다.

J2EE 응용 프로그램은 일반적으로 많은 양의 세션 상태 데이터를 포함하게 됩니다. 웹 장비구니가 세션 상태의 일반적인 예에 해당합니다. 또한 응용 프로그램은 자주 필요한 데이터를 세션 객체에 캐시할 수 있습니다. 실제로 사용자 상호 작용이 자주 발생하는 거의 모든 응용 프로그램에서는 세션 상태가 유지되어야 합니다. HTTP 세션과 Stateful Session Bean(SFSB)에는 세션 상태 데이터가 있습니다.

서버 오류가 발생해도 세션 상태를 지속하는 것은 최종 사용자에게 아주 중요합니다. 고가용성을 위해 Application Server는 HADB에서 세션 상태를 지속하기 위한 기능을 제공합니다. 사용자 세션을 호스팅하는 Application Server 인스턴스에 오류가 발생하면 세션 상태가 복구될 수 있으며 세션은 정보 손실 없이 계속될 수 있습니다.

고가용성 세션 지속성을 설정하는 방법에 대한 자세한 내용은 [9 장](#)을 참조하십시오.

고가용성 Java Message Service

Java Message Service(JMS) API는 J2EE 응용 프로그램 및 구성 요소가 메시지를 작성하고, 보내고, 받고, 읽을 수 있도록 하는 메시징 표준입니다. 또한 느슨하게 결합되고 안정적인 비동기식 분산 통신을 가능하게 합니다. JMS를 구현하는 Sun Java System Message Queue(MQ)는 Application Server와 긴밀하게 통합되어 MDB(Message-Driven Bean)와 같은 JMS에 의존하는 구성 요소를 만들 수 있도록 합니다.

JMS는 연결 풀링 및 페일오버와 MQ 클러스터링을 통해 고가용성을 제공합니다. 자세한 내용은 [10 장](#)을 참조하십시오.

연결 풀링 및 페일오버

Application Server는 JMS 연결 풀링 및 페일오버를 지원합니다. Application Server는 JMS 연결을 자동으로 풀링합니다. 기본적으로 Application Server는 지정된 호스트 목록에서 기본 MQ 브로커를 무작위로 선택합니다. 페일오버가 발생하면 MQ는 로드를 투명하게 다른 브로커로 전송하고 JMS 의미를 유지 관리합니다.

JMS 연결 풀링 및 페일오버에 대한 자세한 내용은 [197 페이지](#) “연결 풀링 및 페일오버”를 참조하십시오.

MQ 클러스터링

MQ Enterprise Edition은 브로커 클러스터로 알려져 있는 상호 연결된 여러 브로커 인스턴스를 지원합니다. 브로커 클러스터를 사용하면 클라이언트 연결이 클러스터에 있는 모든 브로커 간에 분산됩니다. 클러스터링은 수평적 확장을 제공하며 가용성을 향상시킵니다.

MQ 클러스터링에 대한 자세한 내용은 [198 페이지](#) “Application Server에서 MQ 클러스터 사용”을 참조하십시오.

RMI-IIOP 로드 균형 조정 및 페일오버

RMI-IIOP 로드 균형 조정에서 IIOP 클라이언트 요청은 다른 서버 인스턴스나 이름 서버에 배포되며 이를 통해 클러스터에서 로드가 균등하게 분산되어 확장성을 제공합니다. IIOP 로드 균형 조정 기능과 EJB 클러스터링 및 가용성을 함께 사용하면 EJB 페일오버가 구현됩니다.

클라이언트가 객체에 대한 JNDI 조회를 수행할 경우 이름 지정 서비스는 기본적으로 요청을 특정 서버 인스턴스에 바인딩합니다. 이후, 해당 클라이언트의 모든 조회 요청은 동일한 서버 인스턴스로 보내지므로 모든 EJBHome 객체가 동일한 대상 서버에서 호스트됩니다. 이후에 가져온 모든 Bean 참조 또한 동일한 대상 호스트에서 만들어집니다. 이 경우 JNDI 조회를 수행할 때 모든 클라이언트가 활성 대상 서버 목록을 임의화하므로 로드 균형 조정이 효과적으로 제공될 수 있습니다. 대상 서버 인스턴스가 작동 중단되면 조회 또는 EJB 메소드 호출은 다른 서버 인스턴스로 페일오버됩니다.

IIOP 로드 균형 조정 및 페일오버는 투명하게 발생합니다. 응용 프로그램 배포 중에 특별한 단계가 필요하지는 않습니다. 그러나 클러스터에서 새 인스턴스를 추가 또는 삭제해도 클러스터의 기존 클라이언트 뷰가 업데이트되지 않습니다. 이렇게 하려면 클라이언트의 종점 목록을 수동으로 업데이트해야 합니다.

RMI-IIOP 로드 균형 조정 및 페일오버에 대한 자세한 내용은 11 장을 참조하십시오.

추가 정보

하드웨어 요구 사항 평가, 네트워크 구성 계획 및 토폴로지 선택을 비롯한 고가용성 배포 계획에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Deployment Planning Guide**를 참조하십시오. 이 설명서에는 다음과 같은 개념이 자세히 설명되어 있습니다.

- 노드 에이전트, 도메인 및 클러스터와 같은 응용 프로그램 서버 구성 요소
- 클러스터의 IIOP 로드 균형 조정
- HADB 구조
- 메시지 대기열 페일오버

고가용성 기능의 이점을 활용하는 응용 프로그램의 개발에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide**를 참조하십시오.

고가용성을 구현하여 최상의 성능을 얻을 수 있도록 응용 프로그램 및 Application Server를 구성하고 조정하는 방법에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Performance Tuning Guide**를 참조하십시오. 이 설명서에는 다음과 같은 항목이 설명되어 있습니다.

- 지속성 빈도 및 지속성 범위 조정
- Stateful Session Bean 검사점 지정
- JDBC 연결 풀 구성
- 세션 크기
- HADB 디스크 사용, 메모리 할당, 성능 및 운영 체제 구성 조정
- 최상의 성능을 얻을 수 있도록 로드 밸런서 구성

Application Server에서 고가용성을 제공하는 방법

Application Server는 다음의 하위 구성 요소 및 기능을 통해 고가용성을 제공합니다.

- 24 페이지 “로드 밸런서 플러그인”
- 24 페이지 “고가용성 데이터베이스”
- 25 페이지 “고가용성 클러스터”

로드 밸런서 플러그인

로드 밸런서 플러그인은 HTTP/HTTPS 요청을 받아들이고 후 클러스터의 응용 프로그램 서버 인스턴스로 전달합니다. 한 인스턴스에 오류가 발생하거나 네트워크 결함 때문에 사용할 수 없게 되거나 응답하지 않으면, 로드 밸런서는 기존에 있던 사용 가능한 시스템으로 요청을 리디렉션합니다. 또한 로드 밸런서는 실패한 인스턴스가 복구되었을 때를 인식하고 그에 따라 로드를 재분산할 수 있습니다. Application ServerEnterprise Edition 및 Standard Edition에는 Sun Java System Web Server, Apache Web Server 및 Microsoft Internet Information Server용 로드 밸런서 플러그인이 포함되어 있습니다.

로드 밸런서는 여러 물리적 시스템으로 작업 로드를 분산시켜 전반적인 시스템 처리량을 늘립니다. 또한 HTTP 요청의 페일오버를 통해 보다 높은 가용성을 제공합니다. HTTP 세션 정보가 지속되려면 HTTP 세션 지속성을 구성해야 합니다.

상태 비보존형의 경량 응용 프로그램의 경우 로드 균형 조정된 클러스터만으로 충분할 수 있습니다. 그러나 세션 상태를 포함하는 업무에 중요한 응용 프로그램의 경우 HADB와 함께 로드 균형 조정된 클러스터를 사용하십시오.

로드 균형 조정에 참여하는 서버 인스턴스와 클러스터의 환경은 같습니다. 일반적으로 이것은 서버 인스턴스가 동일한 서버 구성을 참조하고, 동일한 물리적 자원에 액세스할 수 있으며, 서버 인스턴스에 동일한 응용 프로그램이 배포되어 있다는 것을 의미합니다. 동일한 환경은 작업 실패 전과 후에 로드 밸런서가 클러스터의 활성 인스턴스에 항상 로드를 균등하게 분산합니다.

로드 균형 조정 및 페일오버 구성에 대한 자세한 내용은 [5 장](#)을 참조하십시오.

고가용성 데이터베이스

Application ServerEnterprise Edition에서는 HTTP 세션 및 Stateful Session Bean 데이터의 고가용성 저장을 위해 고가용성 데이터베이스(HADB)를 제공합니다. HADB는 로드 균형 조정, 페일오버 및 상태 보존형 복구 등을 통해 최고 99.999%의 서비스 및 데이터 가용성을 지원하도록 설계되었습니다. 일반적으로 HADB는 Application Server와는 별도로 구성하고 관리해야 합니다.

Application Server와 별도로 상태 관리를 수행하면 큰 이점을 얻을 수 있습니다. Application Server 인스턴스는 외부 고가용성 상태 서비스에 상태 복제를 위임하는 확장 가능하고 성능이 뛰어난 응용 프로그램 컨테이너의 역할을 수행합니다. 이와 같이

구조가 느슨하게 결합되어 있으므로 Application Server 인스턴스를 클러스터에서 아주 쉽게 추가 또는 삭제할 수 있습니다. HADB 상태 보존형 복제 서비스는 최적의 가용성 및 성능을 위해 독립적으로 확장될 수 있습니다. 또한 Application Server 인스턴스가 복제를 수행하면 J2EE 응용 프로그램의 성능이 저하될 수 있으며 가비지 모음이 더 오랫동안 중단될 수 있습니다.

하드웨어 구성 결정, 크기 조정 및 토폴로지를 비롯하여 HADB를 통한 고가용성 구현을 위해 응용 프로그램 서버 설치를 계획 및 설정하는 방법에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Deployment Planning Guide**의 “Planning for Availability”와 **Sun Java System Application Server Enterprise Edition 8.2 Deployment Planning Guide**의 3 장, “Selecting a Topology”를 참조하십시오.

고가용성 클러스터

클러스터는 하나의 논리 엔티티로 함께 작동하는 Application Server 인스턴스 모음입니다. 클러스터는 하나 이상의 J2EE 응용 프로그램에 런타임 환경을 제공합니다. **고가용성 클러스터**는 상태 보존형 응용 프로그램 서비스를 클러스터 및 로드 밸런서에 통합합니다.

클러스터를 사용하면 다음과 같은 이점을 얻을 수 있습니다.

- **고가용성:** 클러스터 내 서버 인스턴스에 대한 페일오버 보호를 허용합니다. 한 서버 인스턴스가 중지되면 다른 서버 인스턴스에서 사용 불가능한 서버 인스턴스가 처리하던 요청을 넘겨 받습니다.
- **확장성:** 서버 인스턴스를 클러스터에 추가할 수 있어 시스템 용량을 늘릴 수 있습니다. 로드 밸런서 플러그인은 클러스터 내의 사용 가능한 서버 인스턴스로 요청을 분산합니다. 관리자가 서버 인스턴스를 클러스터에 추가하므로 서비스를 중단할 필요가 없습니다.

클러스터의 모든 인스턴스는 다음과 같습니다.

- 동일한 구성을 참조합니다.
- 동일한 집합의 배포된 응용 프로그램을 갖습니다(예: J2EE 응용 프로그램 EAR 파일, 웹 모듈 WAR 파일 또는 EJB JAR 파일).
- 동일한 자원 집합을 가지므로 JNDI 이름 공간도 같습니다.

도메인의 모든 클러스터에는 고유한 이름이 있습니다. 여기서 이 이름은 모든 노드 에이전트 이름, 서버 인스턴스 이름 및 구성 이름에 대해 고유해야 합니다. 이름은 domain이 아니어야 합니다. 클러스터링되지 않은 서버 인스턴스에서 수행하는 것과 동일한 작업(예: 응용 프로그램 배포 및 자원 작성)을 클러스터에서 수행합니다.

클러스터 및 구성

클러스터의 설정은 다른 클러스터와 공유될 수 있는 명명된 구성에서 파생됩니다. 다른 서버 인스턴스나 클러스터와 구성을 공유하지 않는 클러스터는 **독립 실행형 구성**을

갖는다고 할 수 있습니다. 기본적으로 이 구성의 이름은 `cluster_name-config`입니다. 여기서 `cluster_name`은 클러스터의 이름입니다.

다른 클러스터나 인스턴스와 구성을 공유하는 클러스터는 **공유 구성**을 갖는다고 할 수 있습니다.

클러스터, 인스턴스, 세션 및 로드 균형 조정

클러스터, 서버 인스턴스, 로드 밸런서 및 세션은 다음과 같이 관련되어 있습니다.

- 서버 인스턴스가 클러스터의 일부일 필요는 없습니다. 그러나 클러스터의 일부가 아닌 인스턴스는 세션 간의 세션 상태 전송을 통한고가용성을 이용할 수 없습니다.
- 클러스터 내의 서버 인스턴스를 하나 또는 여러 대의 시스템에 호스트할 수 있습니다. 다른 시스템의 서버 인스턴스를 한 클러스터로 묶을 수 있습니다.
- 특정한 로드 밸런서는 요청을 여러 클러스터의 서버 인스턴스로 전달할 수 있습니다. 로드 밸런서의 이 기능을 사용하면 서비스 손실 없이 온라인 업그레이드를 수행할 수 있습니다. 자세한 내용은 [152 페이지 “서비스 손실 없이 구성 요소 업그레이드”](#)를 참조하십시오.
- 한 클러스터에서 여러 로드 밸런서로부터 요청을 수신할 수 있습니다. 클러스터를 여러 로드 밸런서에서 사용할 경우 각 로드 밸런서에서 동일한 방법으로 클러스터를 구성해야 합니다.
- 각 세션은 특정 클러스터에 연결됩니다. 따라서 여러 클러스터에 응용 프로그램을 배포할 수 있지만 세션 페일오버는 단일 클러스터 내에서만 발생합니다.

클러스터는 클러스터 내 서버 인스턴스의 세션 페일오버에 대한 안전한 경계 역할을 합니다. 로드 밸런서를 사용하여 Application Server 내 구성 요소를 서비스 손실 없이 업그레이드할 수 있습니다.

오류 복구

- [26 페이지 “Sun Cluster 사용”](#)
- [27 페이지 “수동 복구”](#)
- [28 페이지 “Netbackup 사용”](#)
- [29 페이지 “Domain Administration Server 다시 만들기”](#)

Sun Cluster 사용

Sun Cluster는 도메인 관리 서버, 노드 에이전트, Application Server 인스턴스, Message Queue 및 HADB의 자동 페일오버를 제공합니다. 자세한 내용은 **Sun Cluster Data Service for Sun Java System Application Server Guide for Solaris OS**를 참조하십시오.

표준 이더넷 상호 연결 및 Sun Cluster 제품의 하위 집합을 사용합니다. 이 기능은 Java ES에 포함되어 있습니다.

수동 복구

다양한 기술을 사용하여 개별 하위 구성 요소를 수동으로 복구할 수 있습니다.

- 27 페이지 “도메인 관리 서버 복구”
- 27 페이지 “노드 에이전트 및 서버 인스턴스 복구”
- 28 페이지 “로드 밸런서 및 웹 서버 복구”
- 28 페이지 “Message Queue 복구”
- 28 페이지 “HADB 복구”

도메인 관리 서버 복구

도메인 관리 서버(DAS)가 손실되면 관리에만 영향을 줍니다. DAS에 연결할 수 없는 경우에도 Application Server 클러스터 및 응용 프로그램은 계속해서 이전과 같이 작동합니다.

다음 방법 중 하나를 사용하여 DAS를 복구합니다.

- 주기적인 스냅샷을 가질 수 있도록 `asadmin` 백업 명령을 주기적으로 실행합니다. 하드웨어 오류가 발생한 후 동일한 네트워크 아이디로 새 시스템에서 Application Server를 설치하고 이전에 만든 백업에서 `asadmin` 복원을 실행합니다. 자세한 내용은 29 페이지 “Domain Administration Server 다시 만들기”를 참조하십시오.
- 견고한 공유 파일 시스템(예: NFS)에 도메인 설치와 구성을 놓습니다. 기본 DAS 시스템이 실패한 경우 수동 개입 또는 사용자 제공 자동화를 통해 동일한 IP 주소를 사용하여 두 번째 시스템이 작동한 후 인계 받습니다. Sun Cluster는 DAS에 내결함성을 제공하기 위해 비슷한 방법을 사용합니다.
- Application Server 설치 및 도메인 루트 디렉토리를 압축합니다. 새 시스템에서 복원하고 동일한 네트워크 아이디를 할당합니다. 파일 기반 설치를 사용하는 중이면 이 방법이 가장 간단한 방법일 수 있습니다.
- DAS 백업에서 복원합니다. AS8.1 UR2 패치 4 지침을 참조하십시오.

노드 에이전트 및 서버 인스턴스 복구

노드 에이전트와 서버 인스턴스를 복구하는 두 가지 방법이 있습니다.

백업 zip 파일 유지. 노드 에이전트와 서버 인스턴스를 백업하기 위한 명시적인 명령이 없습니다. 노드 에이전트 디렉토리의 내용을 포함하는 `zip` 파일을 만들면 됩니다. 오류가 발생한 후 동일한 호스트 이름과 IP 주소를 사용하여 새 시스템에서 저장된 백업의 압축을 풉니다. 동일한 설치 디렉토리 위치, OS 등을 사용합니다. 파일 기반 설치, 패키지 기반 설치 또는 복원된 백업 이미지가 시스템에 있어야 합니다.

수동 복구. 동일한 IP 주소가 있는 새 호스트를 사용해야 합니다.

1. Application Server 노드 에이전트 비트를 시스템에 설치합니다.
2. AS8.1 UR2 패치 4 설치에 대한 지침을 확인합니다.
3. 노드 에이전트를 다시 만듭니다. 서버 인스턴스를 만들 필요는 없습니다.

4. 동기화는 DAS에서 구성과 데이터를 복사 및 업데이트합니다.

로드 밸런서 및 웹 서버 복구

웹 서버 구성만 백업하는 명시적인 명령은 없습니다. 웹 서버 설치 디렉토리를 압축하면 됩니다. 오류가 발생한 후 동일한 네트워크 아이디를 사용하여 새 시스템에서 저장된 백업의 압축을 풉니다. 새 시스템에 다른 IP 주소가 있는 경우 DNS 서버나 라우터를 업데이트합니다.

주 - 이 경우 먼저 웹 서버가 이미지에서 다시 설치되거나 복원된다고 가정합니다.

로드 밸런서 플러그인(플러그인 디렉토리) 및 구성은 웹 서버 설치 디렉토리(일반적으로 /opt/SUNWwbsvr)에 있습니다. web-install/web-instance/config 디렉토리에는 loadbalancer.xml 파일이 포함되어 있습니다.

Message Queue 복구

Message Queue(MQ) 구성 및 자원은 DAS에 저장되며 인스턴스에 동기화될 수 있습니다. 다른 모든 데이터 및 구성 정보는 일반적으로 /var/imq 아래에 있는 MQ 디렉토리에 있으므로 필요에 따라 이러한 디렉토리를 백업 및 복원합니다. 새 시스템에는 MQ 설치가 이미 포함되어 있어야 합니다. 시스템을 복원할 때 이전처럼 MQ 브로커를 시작해야 합니다.

HADB 복구

두 개의 활성 HADB 노드가 있는 경우 오류 시 인계 받을 수 있는 두 개의 예비 노드를 별개의 시스템에 구성할 수 있습니다. HADB 백업 및 복원으로 인해 복원 중인 유효하지 않은 세션이 발생할 수 있으므로 이 방법은 더 완벽한 방법입니다.

예비 노드가 있는 데이터베이스를 만드는 방법에 대한 자세한 내용은 [65 페이지](#) “데이터베이스 만들기”를 참조하십시오. 예비 노드를 데이터베이스에 추가하는 방법에 대한 자세한 내용은 [88 페이지](#) “노드 추가”를 참조하십시오. 복구 및 자체 복원이 실패할 경우 예비 노드에서 자동으로 인계합니다.

Netbackup 사용

주 - 이 절차는 Sun QA에서 테스트되지 않았습니다.

Veritas Netbackup을 사용하여 각 시스템의 이미지를 저장합니다. BPIP의 경우 웹 서버 및 응용 프로그램 서버가 있는 네 개의 시스템을 백업합니다.

복원된 각 시스템에 대해 원래 구성과 동일한 구성(예: 동일한 호스트 이름, IP 주소 등)을 사용합니다.

Application Server와 같은 파일 기반 제품의 경우 관련 디렉토리만 백업 및 복원합니다. 그러나 웹 서버 이미지와 같은 패키지 기반 설치의 경우 전체 시스템을 백업 및 복원해야 합니다. 패키지는 Solaris 패키지 데이터베이스에 설치됩니다. 따라서 디렉토리만 백업한 다음 새 시스템에 복원할 경우 결과적으로 패키지 데이터베이스에 정보가 없는 "배포된" 웹 서버가 됩니다. 이로 인해 이후의 패치 작업이나 업데이트에 문제가 발생할 수 있습니다.

Solaris 패키지 데이터베이스를 수동으로 복사 및 복원하지 마십시오. 또 다른 방법은 웹 서버와 같은 구성 요소가 설치된 후에 시스템의 이미지를 백업하는 것입니다. 이 백업을 기본 tar 파일이라고 합니다. 웹 서버를 변경할 경우 예를 들어 /opt/SUNWwbsvr 아래에 이러한 디렉토리를 백업합니다. 복원하려면 기본 tar 파일에서 시작한 다음 수정된 웹 서버 디렉토리에 복사합니다. 마찬가지로 MQ(BPIP용 패키지 기반 설치)에 이 절차를 사용할 수 있습니다. 원래 시스템을 업그레이드하거나 패치할 경우 새 기본 tar 파일을 만들어야 합니다.

DAS가 있는 시스템이 다운될 경우 복원할 때까지 일정 시간 동안 사용할 수 없습니다.

DAS는 중앙 저장소입니다. 서버 인스턴스를 복원하고 다시 시작할 경우 해당 서버 인스턴스는 DAS의 정보와만 동기화됩니다. 이후 모든 변경 사항은 asadmin 또는 관리 콘솔을 통해 수행되어야 합니다.

이전 응용 프로그램 세션 상태가 이미지에 포함되었을 수도 있으므로 HADB의 일별 백업 이미지가 작동하지 않을 수 있습니다.

Domain Administration Server 다시 만들기

도메인 관리 서버(DAS)를 백업한 경우 호스트 시스템에 오류가 있을 때 DAS를 다시 만들 수 있습니다. DAS의 작업 복사본을 다시 만들려면 다음 사항이 필요합니다.

- 원래 DAS를 포함하는 첫 번째 시스템(machine1)
- 응용 프로그램을 실행하고 클라이언트에 제공하는 서버 인스턴스가 있는 클러스터가 포함된 두 번째 시스템(machine2). DAS를 사용하여 첫 번째 시스템에 클러스터가 구성됩니다.
- 첫 번째 시스템에 문제가 있을 경우 DAS를 다시 만들어야 하는 세 번째 백업 시스템(machine3)

주 - 첫 번째 시스템에서 DAS의 백업을 보존해야 합니다. asadmin backup-domain을 사용하여 현재 도메인을 백업합니다.

▼ 도메인 관리 서버 마이그레이션

DAS를 첫 번째 시스템(machine1)에서 세 번째 시스템(machine3)으로 마이그레이션하려면 다음 단계를 수행합니다.

- 1 첫 번째 시스템에 설치한 대로 세 번째 시스템에 응용 프로그램 서버를 설치합니다.
DAS를 세 번째 시스템에 제대로 복원하고 경로 충돌을 방지하려면 이 단계가 필요합니다.

a. 명령줄(대화식) 모드를 사용하여 Application Server 관리 패키지를 설치합니다.

대화형 명령줄 모드를 활성화하려면 console 옵션을 사용하여 설치 프로그램을 호출하십시오.

```
./bundle-filename -console
```

명령줄 인터페이스를 사용하여 설치하려면 루트 권한이 있어야 합니다.

b. 기본 도메인을 설치하려면 옵션을 선택 해제합니다.

백업한 도메인을 복원하는 것은 동일한 구조뿐만 아니라 정확하게 동일한 설치 경로를 가진 두 시스템(즉, 두 시스템에서 동일한 *install-dir* 및 *domain-root-dir* 사용)에서만 지원됩니다.

- 2 첫 번째 시스템에서 백업 ZIP 파일을 세 번째 시스템의 *domain-root-dir*에 복사합니다.
파일을 FTP에 올릴 수도 있습니다.

- 3 `asadmin restore-domain` 명령을 실행하여 zip 파일을 세 번째 시스템에 복원합니다.

```
asadmin restore-domain --filename domain-root-dir/sjsas_backup_v00001.zip domain1
```

모든 도메인을 백업할 수 있습니다. 그러나 도메인을 다시 만들 때 도메인 이름이 원본과 동일해야 합니다.

- 4 첫 번째 시스템의 동일한 디렉토리의 권한과 일치하도록 세 번째 시스템의 *domain-root-dir/domain1/generated/tmp* 디렉토리의 권한을 변경합니다.

이 디렉토리의 기본 권한은 `drwx-----`(또는 700)입니다.

예를 들면 다음과 같습니다.

```
chmod 700 domain-root-dir/domain1/generated/tmp
```

위 예는 `domain1`을 백업하는 것을 가정합니다. 다른 이름으로 도메인을 백업할 경우 위 `domain1`을 백업할 도메인 이름으로 대체해야 합니다.

- 5 세 번째 시스템의 경우 `domain.xml` 파일의 등록 정보에 대한 호스트 값을 변경합니다.

- 6 세 번째 시스템의 *domain-root-dir/domain1/config/domain.xml*을 업데이트합니다.

예를 들어, `machine1`을 검색하여 이를 `machine3`으로 대체합니다. 그러면 다음을 변경할 수 있습니다.

```
<jmx-connector><property name=client-hostname value=machine1/>...
```

변경 후:

```
<jmx-connector><property name=client-hostname value=machine3/>...
```

7 다음을 변경합니다.

```
<jms-service... host=machine1.../>
```

변경 후:

```
<jms-service... host=machine3.../>
```

8 machine3에서 복구된 도메인을 시작합니다.

```
asadmin start-domain --user admin-user --password admin-password domain1
```

9 machine2의 노드 에이전트에서 등록 정보에 대한 DAS 호스트 값을 변경합니다.

10 machine2의 *install-dir/nodeagents/nodeagent/agent/config/das.properties*에서 *agent.das.host* 등록 정보 값을 변경합니다.

11 machine2에서 노드 에이전트를 다시 시작합니다.

주 - asadmin start-instance 명령을 사용하여 클러스터 인스턴스를 시작하면 클러스터 인스턴스를 복원된 도메인과 동기화할 수 있습니다.

고가용성 데이터베이스 설치 및 설정

이 장은 다음 내용으로 구성되어 있습니다.

- 33 페이지 “HADB 설정 준비”
- 40 페이지 “설치”
- 42 페이지 “고가용성 설정”
- 45 페이지 “HADB 업그레이드”

HADB 설정 준비

이 절에서는 다음 항목에 대해 설명합니다.

- 33 페이지 “필수 조건 및 제한 사항”
- 34 페이지 “네트워크 중복 구성”
- 37 페이지 “공유 메모리 및 세마포 구성”
- 39 페이지 “시스템 클럭 동기화”

이러한 작업을 수행한 후에 3 장을 참조하십시오.

HADB에 대한 최신 정보를 보려면 **Sun Java System Application Server Enterprise Edition 8.2 릴리스 노트**를 참조하십시오.

필수 조건 및 제한 사항

HADB를 설치 및 구성하기 전에 **Sun Java System Application Server Enterprise Edition 8.2 릴리스 노트**에 설명된 요구 사항을 네트워크 및 하드웨어 환경에서 충족하는지 확인합니다. 또한 Veritas의 경우와 같이 특정 파일 시스템에 대한 제한 사항이 있습니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 릴리스 노트**를 참조하십시오.

HADB는 공유 메모리 세그먼트를 만들어 연결할 때 Intimate Shared Memory(SHM_SHARE_MMU 플래그)를 사용합니다. 이 플래그를 사용하면 공유 메모리

세그먼트가 물리적 메모리로 잠기기 때문에 페이지 아웃되지 않습니다. 따라서 HADB의 공유 메모리가 실제 메모리에 잠기며 이로 인해 성능이 낮은 시스템에서 설치에 영향을 주기 쉽습니다. Application Server와 HADB를 공동 배치할 경우 권장량의 메모리를 설치해야 합니다.

네트워크 중복 구성

중복 네트워크를 구성하면 단일 네트워크 장애가 발생해도 HADB가 사용 가능한 상태를 유지합니다. 다음의 두 가지 방식으로 중복 네트워크를 구성할 수 있습니다.

- Solaris 9에서 네트워크 다중 경로를 설정할 수 있습니다.
- Windows Server 2003을 제외한 모든 플랫폼에서 이중 네트워크를 구성할 수 있습니다.

네트워크 다중 경로 설정

네트워크 다중 경로를 설정하기 전에 **IP Network Multipathing Administration Guide**의 2장, “Administering Network Multipathing (Task)”를 참조하십시오.

▼ 이미 IP 다중 경로를 사용하는 HADB 호스트 시스템 구성

1 네트워크 인터페이스 실패 감지 시간을 설정합니다.

HADB가 다중 경로 페일오버를 제대로 지원하기 위해서는 네트워크 인터페이스 실패 감지 시간이 /etc/default/mpathd의 FAILURE_DETECTION_TIME 매개 변수로 지정된 대로 1초(1000밀리초)를 초과해서는 안 됩니다. 원래 값이 큰 경우 파일을 편집하여 이 매개 변수의 값을 1000으로 변경합니다.

```
FAILURE_DETECTION_TIME=1000
```

변경 내용을 적용하려면 다음 명령을 사용하십시오.

```
pkill -HUP in.mpathd
```

2 HADB에서 사용하도록 IP 주소를 설정합니다.

IP Network Multipathing Administration Guide에 설명된 대로 다중 경로는 물리적 네트워크 인터페이스를 다중 경로 인터페이스 그룹으로 그룹화하는 것입니다. 해당 그룹 내의 각 물리적 인터페이스는 다음과 같이 그와 연관된 두 가지의 IP 주소를 갖습니다.

- 데이터 전송에 사용되는 물리적 인터페이스 주소
- Solaris 내부용으로 사용되는 테스트 주소

hadbm create --hosts를 사용할 때는 다중 경로 그룹에서 물리적 인터페이스 주소를 하나만 지정합니다.

예 2-1 다중 경로 설정

이름이 host1 및 host2인 두 대의 시스템이 있고 각각 두 개의 물리적 네트워크 인터페이스가 있는 경우 두 개의 인터페이스를 다중 경로 그룹으로 설정합니다. 각 호스트에서 `ifconfig -a`를 실행합니다.

host1에서 다음과 같은 결과가 출력됩니다.

```
bge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 5 inet 129.159.115.10 netmask ffffffff00 broadcast 129.159.115.255
groupname mp0

bge0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 5 inet 129.159.115.11 netmask ffffffff00 broadcast 129.159.115.255

bge1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 6 inet 129.159.115.12 netmask ffffffff00 broadcast 129.159.115.255
groupname mp0

bge1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 6 inet 129.159.115.13 netmask ff000000 broadcast 129.159.115.255

host2에서 다음과 같은 결과가 출력됩니다.
```

```
bge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 3 inet 129.159.115.20 netmask ffffffff00 broadcast 129.159.115.255
groupname mp0

bge0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 3 inet 129.159.115.21 netmask ff000000 broadcast 129.159.115.255

bge1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 4 inet 129.159.115.22 netmask ffffffff00 broadcast 129.159.115.255
groupname mp0

bge1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 4 inet 129.159.115.23 netmask ff000000 broadcast 129.159.115.255
```

이 예에서는 두 호스트의 물리적 네트워크 인터페이스가 bge0 및 bge1 뒤에 나열됩니다. bge0:1과 bge1:1로 나열된 인터페이스는 **IP Network Multipathing Administration Guide**에 설명된 대로 다중 경로 테스트 인터페이스(ifconfig 출력에 DEPRECATED로 표시됨)입니다.

이 환경에서 HADB를 설정하려면 각 호스트에서 물리적 인터페이스 주소 하나를 선택합니다. 이 예에서 HADB는 host1의 IP 주소 129.159.115.10과 host2의 129.159.115.20을 사용합니다. 호스트당 데이터베이스 노드가 한 개인 데이터베이스를 만들려면 `hadbm create --hosts` 명령을 사용합니다. 예를 들면 다음과 같습니다.

```
hadbm create --hosts 129.159.115.10,129.159.115.20
```

각 호스트에 데이터베이스 노드가 두 개인 데이터베이스를 만들려면 다음 명령을 사용합니다.

```
hadbm create --hosts 129.159.115.10,129.159.115.20,
129.159.115.10,129.159.115.20
```

두 가지 경우에 별도의 매개 변수를 사용하여 host1 및 host2의 에이전트가 사용할 시스템의 인터페이스를 지정하도록 에이전트를 구성해야 합니다. 따라서 host1에서는 다음을 사용합니다.

```
ma.server.mainternal.interfaces=129.159.115.10
```

또한 host2에서는 다음을 사용합니다.

```
ma.server.mainternal.interfaces=129.159.115.20
```

ma.server.mainternal.interfaces 변수에 대한 자세한 내용은 58 페이지 “구성 파일”을 참조하십시오.

이중 네트워크 구성

단일 네트워크 장애를 허용하도록 HADB를 활성화하려면 운영 체제(예: Solaris)에서 지원할 경우 IP 다중 경로를 사용합니다. Windows Server 2003의 경우 이중 네트워크에서 제대로 작동하지 않으므로 이 운영 체제에서는 이중 네트워크를 사용하여 HADB를 구성하지 않도록 합니다.

운영 체제가 IP 다중 경로에 대해 구성되지 않았으며 HADB 호스트에 두 개의 NIC가 장착되어 있으면 이중 네트워크를 사용하도록 HADB를 구성할 수 있습니다. 모든 호스트에 대해 각 네트워크 인터페이스 카드(NIC)의 IP 주소가 별도의 IP 서브넷에 있어야 합니다.

데이터베이스 내에서 모든 노드가 단일 네트워크에 연결되어 있거나 모든 노드가 두 개의 네트워크에 연결되어 있어야 합니다.

주- 서브넷 간 라우터는 서브넷 간에 UDP 멀티캐스트 메시지를 전달하도록 구성되어야 합니다.

HADB 데이터베이스를 만들 때 -hosts 옵션을 사용하여 각 노드, 즉 각 NIC IP 주소에 대해 하나씩 두 개의 IP 주소나 호스트 이름을 지정합니다. 노드마다 첫 번째 IP 주소는 net-0에 있고 두 번째 주소는 net-1에 있습니다. 구문은 다음과 같습니다. 이 경우 동일한 노드에 대한 호스트 이름을 플러스 기호(+)로 구분해서 입력합니다.

```
--hosts=node0net0name+node0net1name
,node1net0name+node1net1name
,node2net0name+node2net1name
, ...
```

예를 들어, 다음 인수는 각각 두 개의 네트워크 인터페이스를 갖는 두 개의 노드를 만듭니다. 이러한 노드를 만드는데 다음 호스트 옵션이 사용됩니다.

```
--hosts 10.10.116.61+10.10.124.61,10.10.116.62+10.10.124.62
```

따라서 네트워크 주소는 다음과 같이 지정됩니다.

- node0의 경우 10.10.116.61 및 10.10.124.61
- node1의 경우 10.10.116.62 및 10.10.124.62

10.10.116.61 및 10.10.116.62가 동일한 서브넷에 있고, 10.10.124.61 및 10.10.124.62가 동일한 서브넷에 있습니다.

이 예에서 관리 에이전트는 동일한 서브넷을 사용해야 합니다. 따라서 구성 변수 `ma.server.maininternal.interfaces`를 10.10.116.0/24 등으로 설정해야 합니다. 이러한 설정은 이 예의 두 에이전트에 사용될 수 있습니다.

공유 메모리 및 세마포 구성

HADB를 설치하기 전에 공유 메모리 및 세마포를 구성해야 합니다. 절차는 운영 체제에 따라 다릅니다.

호스트에서 HADB가 아닌 응용 프로그램을 실행할 경우 응용 프로그램의 공유 메모리 및 세마포 사용을 계산하고 HADB에 필요한 값에 추가합니다. 이 절에서 권장하는 값은 각 호스트에서 최대 6개의 HADB 노드를 실행하는데 충분합니다. 7개 이상의 HADB 노드를 실행하거나 추가 공유 메모리와 세마포가 필요한 응용 프로그램을 호스트에서 실행할 경우에만 값을 늘리면 됩니다.

세마포 수가 너무 적은 경우 HADB가 실패하고 `No space left on device` 오류 메시지가 표시될 수 있습니다. 이 오류는 데이터베이스를 시작하는 중 또는 런타임 중에 발생할 수 있습니다.

▼ Solaris에서 공유 메모리 및 세마포 구성

세마포가 전역 운영 체제 자원이기 때문에 구성은 HADB뿐만 아니라 호스트에서 실행 중인 모든 프로세스에 의존합니다. Solaris에서 `/etc/system` 파일을 편집하여 세마포 설정을 구성합니다.

1 루트로 로그인합니다.

2 공유 메모리를 구성합니다.

- 호스트에서 단일 공유 메모리 세그먼트의 최대 크기를 지정하는 `shminfo_shmmax`를 설정합니다. 이 값을 HADB 호스트 시스템에 설치된 총 RAM 크기로 설정하되 2GB가 초과되지 않는 16진수 값으로 설정합니다.

예를 들어, 2GB RAM의 경우 `/etc/system` 파일에서 다음과 같이 값을 설정합니다.

```
set shmsys:shminfo_shmmax=0x80000000
```

주 - 호스트 시스템의 메모리를 확인하려면 다음 명령을 사용합니다.

```
prtconf | grep Memory
```

- Solaris 8 이전 버전에서는 하나의 프로세스가 연결할 수 있는 최대 공유 메모리 세그먼트 수인 `shminfo_shmseg`를 설정합니다. 이 값을 호스트당 노드 수의 6배로 설정합니다. 호스트당 최대 6개 노드인 경우 `/etc/system` 파일에 다음을 추가합니다.

```
set shmsys:shminfo_shmseg=36
```

Solaris 9 이상에서는 `shmsys:shminfo_shmseg`가 더 이상 사용되지 않습니다.

- 전체 시스템의 최대 공유 메모리 세그먼트 수인 `shminfo_shmmni`를 설정합니다. 각 HADB 노드가 6개의 공유 메모리 세그먼트를 할당하므로 HADB에 필요한 값은 호스트당 노드 수의 6배 이상이어야 합니다. Solaris 9에서는 호스트당 최대 6개 노드인 경우 기본값을 변경할 필요가 없습니다.

3 세마포를 구성합니다.

예를 들어, `/etc/system` 파일에서 다음 세마포 구성 항목을 확인합니다.

```
set semsys:seminfo_semmni=10
set semsys:seminfo_semmns=60
set semsys:seminfo_semmnu=30
```

해당 항목이 있을 경우 아래와 같이 값을 늘립니다.

`/etc/system` 파일에 이러한 항목이 없으면 파일 끝에 추가합니다.

- 최대 세마포 식별자 수인 `seminfo_semmni`를 설정합니다. 각 HADB 노드에는 하나의 세마포 식별자가 필요합니다. Solaris 9에서는 호스트당 최대 6개 노드인 경우 기본값을 변경할 필요가 없습니다. 예를 들면 다음과 같습니다.

```
set semsys:seminfo_semmni=10
```

- 전체 시스템의 최대 세마포 수인 `seminfo_semmns`를 설정합니다. 각 HADB 노드에는 8개의 세마포가 필요합니다. Solaris 9에서는 호스트당 최대 6개 노드인 경우 기본값을 변경할 필요가 없습니다. 예를 들면 다음과 같습니다.

```
set semsys:seminfo_semmns=60
```

- 시스템의 최대 실행 취소 구조 수인 `seminfo_semmnu`를 설정합니다. 각 연결에는 하나의 실행 취소 구조가 필요합니다(구성 변수 `NumberOfSessions`, 기본값 100). 호스트당 최대 6개 노드인 경우 다음과 같이 600으로 설정합니다.

```
set semsys:seminfo_semmnu=600
```

4 시스템을 재부트합니다.

▼ Linux에서 공유 메모리 구성

Linux에서는 공유 메모리 설정을 구성해야 합니다. 기본 세마포 설정을 조정할 필요가 없습니다.

1 루트로 로그인합니다.

2 /etc/sysctl.conf 파일을 편집합니다.

Redhat Linux에서는 sysctl.conf를 수정하여 커널 매개 변수를 설정할 수도 있습니다.

3 다음과 같이 kernel.shmax 및 kernel.shmall 값을 설정합니다.

```
echo MemSize > /proc/sys/shmmax
echo MemSize > /proc/sys/shmall
```

여기서 *MemSize*는 바이트 수입니다.

kernel.shmax 매개 변수는 공유 메모리 세그먼트의 최대 크기(바이트)를 정의합니다. kernel.shmall 매개 변수는 시스템에서 한 번에 사용될 수 있는 페이지의 총 공유 메모리 양을 설정합니다. 이러한 매개 변수 값을 시스템의 실제 메모리 양으로 설정합니다. 값을 십진 바이트 값으로 지정합니다.

예를 들어, 두 값을 모두 2GB로 설정하려면 다음을 사용합니다.

```
echo 2147483648 > /proc/sys/kernel/shmmax
echo 2147483648 > /proc/sys/kernel/shmall
```

4 다음 명령을 사용하여 시스템을 재부트합니다.

```
sync; sync; reboot
```

Windows 절차

Windows에는 특수한 시스템 설정이 필요하지 않습니다. 그러나 기존 J2SE 설치를 사용하려면 JAVA_HOME 환경 변수를 J2SE가 설치된 위치로 설정합니다.

시스템 클럭 동기화

HADB는 시스템 클럭을 기반으로 하는 타임스탬프를 사용하므로 HADB 호스트의 클럭을 동기화해야 합니다. HADB는 시스템 클럭을 사용하여 시간 초과를 관리하고 내역 파일에 기록된 이벤트에 시간을 표시합니다. HADB는 분산 시스템이므로 문제 해결을 위해서는 모든 내역 파일을 함께 분석해야 합니다. 따라서 모든 호스트의 클럭을 동기화하는 것이 중요합니다.

HADB 시스템이 실행 중인 상태에서는 시스템 클럭을 조정하지 마십시오. 클럭을 조정하면 운영 체제 또는 기타 소프트웨어 구성 요소에서 문제가 발생할 수 있으며 이로 인해 HADB 노드의 중단 또는 다시 시작과 같은 문제가 발생할 수 있습니다. 클럭을 거꾸로 조정하면 클럭이 조정될 때 일부 HADB 서버 프로세스가 중단될 수 있습니다.

클럭 동기화

- Solaris에서는 xntpd(네트워크 시간 프로토콜 데몬)를 사용합니다.
- Linux에서는 ntpd를 사용합니다.
- Windows에서는 Windows의 NTPTIME을 사용합니다.

HADB에서 클럭이 2초 이상 조정되었음을 감지하면 다음과 같이 노드 내역 파일에 기록합니다.

```
NSUP INF 2003-08-26 17:46:47.975 Clock adjusted.
Leap is +195.075046 seconds.
```

설치

일반적으로 Application Server가 있는 동일한 시스템(같은 시스템 위치 토폴로지)이나 별도의 호스트(별도의 계층 토폴로지)에 HADB를 설치할 수 있습니다. 이러한 두 옵션에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Deployment Planning Guide**의 3 장, “Selecting a Topology”를 참조하십시오.

`asadmin configure-ha-cluster` 명령으로고가용성을 설정할 수 있도록 HADB 관리 클라이언트를 설치해야 합니다. Java Enterprise System 설치 프로그램을 사용할 때는 별도의 계층에 노드를 설치하려는 경우에도 전체 HADB 인스턴스를 설치하여 관리 클라이언트를 설치해야 합니다.

HADB 설치

최소 2GB의 메모리가 있는 단일 또는 이중 CPU 시스템에서는 HADB와 Application Server를 모두 설치할 수 있습니다. 그렇지 않은 경우 별도의 시스템에 HADB를 설치하거나 추가 하드웨어를 사용하십시오. `asadmin configure-ha-cluster` 명령을 사용하려면 HADB와 Application Server를 모두 설치해야 합니다.

각 HADB 노드에는 512MB의 메모리가 필요하므로 시스템에서 두 개의 HADB 노드를 실행하려면 1GB의 메모리가 있어야 합니다. 시스템의 메모리가 부족하면 각 노드를 다른 시스템에 설치합니다. 예를 들어, 다음 시스템에 두 노드를 설치할 수 있습니다.

- 각각 512MB~1GB의 메모리가 있는 두 대의 단일 CPU 시스템
- 1GB~2GB의 메모리가 있는 단일 또는 이중 CPU 시스템

Java Enterprise System 설치 프로그램 또는 Application Server 독립 실행형 설치 프로그램을 사용하여 HADB를 설치할 수 있습니다. 이러한 설치 프로그램을 사용할 경우 구성 요소 선택 페이지에서 HADB(Java ES에서는 고가용성 세션 저장소임) 설치 옵션을 선택하십시오. 호스트에서 설치를 완료합니다. Application Server 독립 실행형 설치 프로그램을 사용하고 HADB를 실행하기 위해 별도의 두 시스템을 선택할 경우 두 시스템에서 동일한 설치 디렉토리를 선택해야 합니다.

기본 설치 디렉토리

이 설명서에서 *HADB_install_dir*은 HADB가 설치되는 디렉토리를 나타냅니다. 기본 설치 디렉토리는 HADB를 Java Enterprise System의 일부로 설치할지 여부에 따라 다릅니다. Java Enterprise System의 경우 기본 설치 디렉토리는 */opt/SUNWhadb/4*이고 독립 실행형 Application Server 설치 프로그램의 경우에는 */opt/SUNWappserver/hadb/4*입니다.

노드 수퍼바이저 프로세스 권한

노드 수퍼바이저 프로세스(NSUP)는 "I'm alive" 메시지를 서로 교환하여 HADB의 가용성을 보장합니다. NSUP 실행 파일은 가능한 빠르게 응답할 수 있도록 루트 권한을 가져야 합니다. *clu_nsup_srv* 프로세스는 많은 양의 CPU 자원을 소비하지 않고 사용 공간도 작으므로 실시간 우선 순위에 따라 실행해도 성능이 저하되지 않습니다.

주 - Java Enterprise System 설치 프로그램은 NSUP 권한을 자동으로 적절하게 설정하므로 다른 작업이 필요하지 않습니다. 그러나 독립 실행형 Application Server(비루트) 설치 프로그램을 사용할 경우 데이터베이스를 만들기 전에 수동으로 권한을 설정해야 합니다.

권한이 부족할 경우 나타나는 현상

NSUP에 적절한 권한이 없으면 다음과 같은 자원 부족 현상이 나타날 수 있습니다.

- HADB 내역 파일에 "Process blocked for *n* seconds" 경고가 발생한 후에 잘못된 네트워크 분할 및 노드 다시 시작
- 트랜잭션 중단 및 기타 예외 발생

제한 사항

NSUP가 실시간 우선 순위를 설정할 수 없는 경우 Solaris 및 Linux에서 *errno*가 *EPERM*으로 설정됩니다. Windows의 경우 경고 "Could not set real-time priority"가 발생합니다. *ma.log* 파일에 오류가 기록되고 실시간 우선 순위 없이 프로세스가 계속됩니다.

다음의 경우에는 실시간 우선 순위를 설정할 수 없습니다.

- HADB가 Solaris 10 비전역 영역에 설치된 경우

- PRIV_PROC_LOCK_MEMORY(프로세스가 물리적 메모리에서 페이지를 잠그도록 허용) 및/또는 PRIV_PROC_PRIOCNTRL 권한이 Solaris 10에서 호출된 경우
- 사용자가 setuid 권한을 해제한 경우
- 사용자가 소프트웨어를 tar 파일로 설치한 경우(Application Server에 대한 비루트 설치 옵션)

▼ 노드 슈퍼바이저 프로세스 루트 권한 부여

1 루트로 로그인합니다.

2 작업 디렉토리를 *HADB_install_dir/lib/server*로 변경합니다.
NSUP 실행 파일은 *clu_nsup_srv*입니다.

3 다음 명령을 사용하여 파일의 suid 비트를 설정합니다.
`chmod u+s clu_nsup_srv`

4 다음 명령을 사용하여 파일의 소유권을 루트로 설정합니다.
`chown root clu_nsup_srv`

따라서 *clu_nsup_srv* 프로세스는 루트로 실행되고 실시간 우선 순위를 자체적으로 갖게 됩니다.

보안에 영향을 주지 않기 위해 프로세스가 시작된 직후에 실시간 우선 순위가 설정되고 우선 순위가 변경되면 프로세스에는 유효 UID가 다시 지정됩니다. 다른 HADB 프로세스는 일반 우선 순위로 실행됩니다.

고가용성 설정

이 절에서는 고가용성 클러스터를 만들고 HTTP 세션 지속성을 테스트하기 위한 단계를 설명합니다.

이 절은 다음 내용으로 구성되어 있습니다.

- 43 페이지 “고가용성을 위한 시스템 준비”
- 43 페이지 “HADB 관리 에이전트 시작”
- 43 페이지 “고가용성을 위한 클러스터 구성”
- 44 페이지 “고가용성을 위한 응용 프로그램 구성”
- 44 페이지 “클러스터 다시 시작”
- 44 페이지 “Web Server 다시 시작”
- 45 페이지 “로드 밸런서로 작동하는 Web Server 인스턴스 정리”

▼ 고가용성을 위한 시스템 준비

- 1 **Application Server 인스턴스 및 로드 밸런서 플러그인을 설치합니다.**
자세한 내용은 *Java Enterprise System 설치 설명서* (Java ES를 사용할 경우) 또는 **Sun Java System Application Server Enterprise Edition 8.2 Installation Guide** (독립 실행형 Application Server 설치 프로그램을 사용할 경우)를 참조하십시오.
- 2 **Application Server 도메인 및 클러스터를 만듭니다.**
도메인을 만드는 방법에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 “도메인 만들기”를 참조하십시오. 클러스터를 만드는 방법에 대한 자세한 내용은 **146 페이지 “클러스터 만들기”**를 참조하십시오.
- 3 **웹 서버 소프트웨어를 설치 및 구성합니다.**
자세한 내용은 **4 장**을 참조하십시오.
- 4 **로드 균형 조정 기능을 설정 및 구성합니다.**
자세한 내용은 **119 페이지 “HTTP 로드 균형 조정 설정”**을 참조하십시오.

HADB 관리 에이전트 시작

관리 에이전트 **ma**는 HADB 호스트에서 관리 명령을 실행하고 HADB 노드에 장애가 발생할 경우 다시 시작하여 HADB 노드 수퍼바이저 프로세스의 가용성을 유지합니다.

다음 두 가지 방법으로 관리 에이전트를 시작할 수 있습니다.

- 프로덕션 사용을 위해 서비스로 시작합니다. **52 페이지 “관리 에이전트를 서비스로 시작”**을 참조하십시오. 관리 에이전트의 가용성을 위해 시스템이 재부트될 때 자동으로 다시 시작되는지 확인합니다. **53 페이지 “관리 에이전트의 자동 재시작 확인”**을 참조하십시오.
- 평가, 테스트 또는 개발을 위해 일반 프로세스로 (콘솔 모드에서) 시작합니다. **55 페이지 “콘솔 모드에서 관리 에이전트 시작”**을 참조하십시오.
각 경우 절차는 Java Enterprise System과 독립 실행형 Application Server 중 무엇을 사용하는지에 따라 다릅니다.

고가용성을 위한 클러스터 구성

이 절을 시작하기 전에 하나 이상의 Application Server 클러스터를 만들어야 합니다. 클러스터를 만드는 방법에 대한 자세한 내용은 **146 페이지 “클러스터 만들기”**를 참조하십시오.

DAS(Domain Administration Server)가 실행 중인 시스템에서 다음 명령을 사용하여 HADB를 사용하도록 클러스터를 구성합니다.

```
asadmin configure-ha-cluster --user admin --hosts hadb_hostname1,hadb_hostname2
[,...] --devicesize 256 clusterName
```

여기서 *hadb_hostname1*, *hadb_hostname2* 등은 HADB가 실행 중인 시스템의 호스트 이름이고 *clusterName*은 클러스터의 이름입니다. 예를 들면 다음과 같습니다.

```
asadmin configure-ha-cluster --user admin --hosts host1,host2,host1,host2
--devicesize 256 cluster1
```

이 예에서는 HADB 페일오버의 경우에도 고가용성을 제공하는 두 개의 노드를 각 시스템에 만듭니다. `-hosts` 옵션 다음에 오는 호스트 이름의 순서가 중요합니다. 따라서 앞의 예는 `--hosts host1,host1,host2,host2`와 다릅니다.

시스템을 한 대만 사용할 경우에는 호스트 이름을 두 번 제공해야 합니다.

고가용성을 위한 응용 프로그램 구성

관리 콘솔의 응용 프로그램 > 엔터프라이즈 응용 프로그램에서 응용 프로그램을 선택합니다. 가용성 사용을 설정하고 저장을 누릅니다.

클러스터 다시 시작

관리 콘솔에서 클러스터를 다시 시작하려면 클러스터 > *cluster-name*을 선택합니다. 인스턴스 중지를 누릅니다. 인스턴스가 중지되면 "인스턴스 시작" 또는

다음 `asadmin` 명령을 사용합니다.

```
asadmin stop-cluster --user admin cluster-name
asadmin start-cluster --user admin cluster-name
```

이러한 명령에 대한 자세한 내용은 `stop-cluster(1)` 및 `start-cluster(1)`을 참조하십시오.

Web Server 다시 시작

웹 서버를 다시 시작하려면 사용 중인 웹 서버에 적합한 명령을 사용합니다. 예를 들어, Sun Java System Web Server를 다시 시작하려면 다음 명령을 사용합니다.

```
web_server_root/https-hostname/restart
```

*web_server_root*를 Web Server 루트 디렉토리로 대체하고 *hostname*을 호스트 시스템의 이름으로 대체합니다.

▼ 로드 밸런서로 작동하는 Web Server 인스턴스 정리

- 1 다음과 같이 로드 밸런서 구성을 삭제합니다.
`asadmin delete-http-lb-ref --user admin --config MyLbConfig FirstCluster`
`asadmin delete-http-lb-config --user admin MyLbConfig`
- 2 새 Web Server 인스턴스를 만든 경우 다음을 수행하여 삭제할 수 있습니다.
 - a. Web Server의 관리 콘솔로 로그인합니다.
 - b. 인스턴스를 중지합니다.
인스턴스를 삭제합니다.

HADB 업그레이드

HADB는 소프트웨어 업그레이드로 인해 인터럽트되지 않는 "Always On" 서비스를 제공하도록 설계되었습니다. 이 절에서는 데이터베이스를 오프라인으로 만들거나 가용성을 손상시키지 않으면서 새 버전의 HADB로 업그레이드하는 방법에 대해 설명합니다. 이 방법을 **온라인 업그레이드**라고 합니다. HADB 버전 4.4.x는 온라인 업그레이드를 지원하지만 이전 버전은 지원하지 않습니다.

다음 절에서는 HADB 설치를 업그레이드하는 방법에 대해 설명합니다.

- 45 페이지 “HADB를 새 버전으로 업그레이드”
- 46 페이지 “HADB 패키지 등록”
- 47 페이지 “HADB 패키지 등록 취소”
- 48 페이지 “관리 에이전트 시작 스크립트 교체”
- 48 페이지 “HADB 업그레이드 확인”

▼ HADB를 새 버전으로 업그레이드

- 1 새 버전의 HADB를 설치합니다.
- 2 **46 페이지 “HADB 패키지 등록”**에 설명된 대로 새 HADB 버전을 등록합니다.
 HADB 관리 도메인에 HADB 패키지를 등록하면 HADB 패키지를 쉽게 업그레이드하거나 변경할 수 있습니다. 관리 에이전트는 소프트웨어 패키지의 위치뿐만 아니라 도메인에 있는 호스트에 대한 버전 정보를 추적합니다. 기본 패키지 이름은 V로 시작하며 hadbm 프로그램의 버전 번호를 포함하는 문자열입니다.

- 3 데이터베이스에서 사용하는 패키지를 변경합니다.
다음 명령을 입력합니다.
`hadbm set PackageName=package`
여기서 *package*는 새 HADB 패키지의 버전 번호입니다.
- 4 47 페이지 “HADB 패키지 등록 취소”에 설명된 대로 기존 HADB 설치의 등록을 취소합니다.
- 5 필요한 경우 관리 에이전트 시작 스크립트를 바꿉니다.
자세한 내용은 48 페이지 “관리 에이전트 시작 스크립트 교체”를 참조하십시오.
- 6 48 페이지 “HADB 업그레이드 확인”에 설명된 대로 결과를 확인합니다.
- 7 (옵션) 이전 HADB 버전에 대한 이전 파일을 제거합니다.
HADB가 올바르게 업그레이드되었는지 확인한 후 이전 HADB 패키지를 삭제할 수 있습니다.

HADB 패키지 등록

`hadbm registerpackage` 명령을 사용하여 관리 도메인의 호스트에 설치된 HADB 패키지를 등록합니다. `hadbm create`를 사용하여 데이터베이스를 만들 때 HADB 패키지를 등록할 수도 있습니다.

`hadm registerpackage` 명령을 사용하기 전에 모든 관리 에이전트가 호스트 목록의 모든 호스트에서 구성되고 실행되고 있는지, 관리 에이전트의 저장소를 업데이트할 수 있는지, 동일한 패키지 이름으로 이미 등록된 소프트웨어 패키지가 없는지 확인합니다.

명령 구문은 다음과 같습니다.

```
hadbm registerpackage --packagepath=path [--hosts=hostlist] [--adminpassword=password | --adminpasswordfile=file] [--agent=maurl] [[package-name]]
```

package-name 피연산자는 패키지의 이름입니다.

다음 표에서는 특수한 `hadbm registerpackage` 명령 옵션에 대해 설명합니다. 다른 명령 옵션에 대한 설명을 보려면 60 페이지 “보안 옵션” 및 62 페이지 “일반 옵션”을 참조하십시오.

표 2-1 hadbm registerpackage 옵션

옵션	설명
--hosts= <i>hostlist</i>	침표로 구분되거나 큰따옴표로 묶이고 공백으로 구분된 호스트 목록
-H	
--packagepath= <i>path</i>	HADB 소프트웨어 패키지에 대한 경로.
-L	

예를 들어, 다음 명령을 사용하면 소프트웨어 패키지 v4가 호스트 host1, host2, host3에 등록됩니다.

```
hadbm registerpackage
--packagepath=hadb_install_dir/SUNWHadb/4.4
--hosts=host1,host2,host3 v4
```

다음과 같은 응답이 제공됩니다.

```
Package successfully registered.
```

--hosts 옵션을 생략하면 도메인에서 활성화된 모든 호스트에 패키지가 등록됩니다.

HADB 패키지 등록 취소

hadbm unregisterpackage 명령을 사용하여 관리 도메인에 등록된 HADB 패키지를 제거합니다.

hadbm unregisterpackage 명령을 사용하기 전에 다음을 확인합니다.

- 모든 관리 에이전트가 구성되었으며 *hostlist*의 모든 호스트에서 실행 중입니다.
- 관리 에이전트의 저장소를 업데이트에 사용할 수 있습니다.
- 새 HADB 패키지가 관리 도메인에 등록되어 있습니다.
- 등록 취소하려는 패키지에서 실행하도록 구성된 기존 데이터베이스가 없습니다.

명령 구문은 다음과 같습니다.

```
hadbm unregisterpackage
--hosts=hostlist
[--adminpassword=password | --adminpasswordfile= file]
[--agent= maurl]
[package-name ]
```

package-name 피연산자는 패키지의 이름입니다.

--hosts 옵션에 대한 설명은 46 페이지 “HADB 패키지 등록”을 참조하십시오. --hosts 옵션을 생략하면 **hostlist**는 기본적으로 패키지가 등록된 활성 호스트가 됩니다. 다른 명령 옵션에 대한 설명을 보려면 60 페이지 “보안 옵션” 및 62 페이지 “일반 옵션”을 참조하십시오.

예 2-2 HADB 등록 취소의 예

도메인의 특정 호스트에서 소프트웨어 패키지 v4를 등록 취소하려면 다음을 입력합니다.

```
hadbm unregisterpackage --hosts=host1,host2,host3 v4
```

다음과 같은 응답이 제공됩니다.

```
Package successfully unregistered.
```

관리 에이전트 시작 스크립트 교체

새 버전의 HADB를 설치할 때 /etc/init.d/ma-initd에서 관리 에이전트 시작 스크립트를 교체해야 하는 경우가 있습니다. *HADB_install_dir/lib/ma-initd* 파일의 내용을 확인합니다. 내용이 이전 *ma-initd* 파일과 다르면 이전 파일을 새 파일로 교체합니다.

▼ HADB 업그레이드 확인

다음 절차에 따라 HADB가 올바르게 업그레이드되었는지 확인합니다.

1 실행 중인 HADB 프로세스의 버전을 확인합니다.

모든 HADB 노드에서 다음 명령을 입력하여 HADB 버전을 표시합니다.

```
new-path/bin/ma -v
```

```
new-path/bin/hadbm -v
```

여기서 *new-path*는 새 HADB 설치의 경로입니다.

결과에 새 HADB 버전 번호가 표시되어야 합니다.

2 데이터베이스가 실행 중인지 확인합니다.

다음 명령을 입력합니다.

```
new-path/bin/hadbm status -n
```

업그레이드에 성공한 경우 결과에서 모든 HADB 노드가 **running** 상태로 표시됩니다.

3 HADB를 사용하는 제품의 구성 설정이 새 HADB 경로로 변경되었는지 확인합니다.

- 4 HADB를 사용하는 제품에 대해 모든 업그레이드 테스트를 실행합니다.

고가용성 데이터베이스 관리

이 장에서는 Sun Java System Application Server Enterprise Edition 환경의 고가용성 데이터베이스(HADB)에 대해 설명하고 이를 구성하고 관리하는 방법에 대해 설명합니다. HADB를 만들고 관리하려면 먼저 시스템의 토폴로지를 확인하고 여러 시스템에 HADB 소프트웨어를 설치해야 합니다.

이 장은 다음 내용으로 구성되어 있습니다.

- 51 페이지 “HADB 관리 에이전트 사용”
- 59 페이지 “hadbm 관리 명령 사용”
- 64 페이지 “HADB 구성”
- 78 페이지 “HADB 관리”
- 87 페이지 “HADB 확장”
- 92 페이지 “HADB 모니터링”
- 99 페이지 “HADB 시스템 유지 관리”

HADB 관리 에이전트 사용

관리 에이전트 `ma`는 HADB 호스트에서 관리 명령을 실행합니다. 또한 관리 에이전트는 HADB 노드 슈퍼바이저 프로세스가 실패한 경우 다시 시작하여 HADB의 가용성을 보장합니다.

- 51 페이지 “관리 에이전트 시작”
- 56 페이지 “관리 에이전트 명령 구문”
- 58 페이지 “관리 에이전트 구성 사용자 정의”

관리 에이전트 시작

다음과 같은 방법으로 관리 에이전트를 시작할 수 있습니다.

- 프로덕션 사용을 위해 서비스로 시작합니다. [52 페이지 “관리 에이전트를 서비스로 시작”](#)을 참조하십시오. 관리 에이전트의 가용성을 위해, 시스템을 재부트할 때 자동으로 다시 시작되는지 확인하십시오. [53 페이지 “관리 에이전트의 자동 재시작 확인”](#)을 참조하십시오.
- 평가, 테스트 또는 개발을 위해 일반 프로세스로(콘솔 모드에서) 시작합니다. [55 페이지 “콘솔 모드에서 관리 에이전트 시작”](#)을 참조하십시오.
- Solaris 10에서 SMF(Service Management Facility)를 사용하여 시작합니다.

각 경우 절차는 Java Enterprise System과 독립 실행형 Application Server 중 무엇을 사용하는지에 따라 다릅니다.

관리 에이전트를 서비스로 시작

관리 에이전트를 서비스로 시작하면 시스템이 종료되거나 관리 에이전트를 명시적으로 중지하기 전까지는 계속 실행됩니다. 명령은 설치 및 플랫폼에 따라 다릅니다.

- [52 페이지 “Solaris 또는 Linux에서의 Java Enterprise System”](#)
- [52 페이지 “Windows에서의 Java Enterprise System”](#)
- [53 페이지 “Solaris 또는 Linux에서의 독립 실행형 Application Server”](#)
- [53 페이지 “Windows에서의 독립 실행형 Application Server”](#)

Solaris 또는 Linux에서의 Java Enterprise System

관리 에이전트를 서비스로 시작하려면 다음 명령을 사용합니다.

```
/etc/init.d/ma-initd start
```

서비스를 중지하려면 다음 명령을 사용합니다.

```
/etc/init.d/ma-initd stop
```

Windows에서의 Java Enterprise System

관리 에이전트를 Windows 서비스로 시작하려면 다음 명령을 사용합니다.

```
HADB_install_dir\bin\ma -i [config-file]
```

선택 인수 *config-file*은 관리 에이전트 구성 파일을 지정합니다. 구성 파일은 기본 관리 에이전트 구성을 변경하려는 경우에만 사용합니다. 자세한 내용은 [58 페이지 “관리 에이전트 구성 사용자 정의”](#)를 참조하십시오.

관리 에이전트를 서비스로 중지하고 제거(등록 해제)하려면 다음 명령을 사용합니다.

```
HADB_install_dir\bin\ma -r [config-file]
```

관리를 수행하려면 관리 도구 | 서비스를 선택합니다. 이렇게 하면 서비스를 시작하고 중지하거나, 자동 시작을 비활성화하는 등의 작업을 할 수 있습니다.

Solaris 또는 Linux에서의 독립 실행형 Application Server

관리 에이전트를 서비스로 시작하려면 다음 명령을 사용합니다.

```
HADB_install_dir/bin/ma-initd start
```

서비스를 중지하려면 다음 명령을 사용합니다.

```
HADB_install_dir/bin/ma-initd stop
```

기본값을 변경하려면 쉘 스크립트 `HADB_install_dir/bin/ma-initd`를 편집합니다. `ma-initd`를 `/etc/init.d` 디렉토리로 복사합니다. 스크립트에서 설치에 맞게 `HADB_ROOT` 및 `HADB_MA_CFG`의 기본값을 대체합니다.

- `HADB_ROOT`는 HADB 설치 디렉토리 `HADB_install_dir`입니다.
- `HADB_MA_CFG`는 관리 에이전트 구성 파일의 위치입니다. 자세한 내용은 [58 페이지](#) “관리 에이전트 구성 사용자 정의”를 참조하십시오.

Windows에서의 독립 실행형 Application Server

관리 에이전트를 Windows 서비스로 시작하려면 다음 명령을 사용합니다.

```
HADB_install_dir\bin\ma -i [config-file ]
```

선택 인수 `config-file`은 관리 에이전트 구성 파일을 지정합니다. 구성 파일은 기본 관리 에이전트 구성을 변경하려는 경우에만 사용합니다.

관리 에이전트를 서비스로 중지하고 제거(등록 해제)하려면 다음 명령을 사용합니다.

```
HADB_install_dir\bin\ma -r [config-file ]
```

관리를 수행하려면 관리 도구 | 서비스를 선택합니다. 이렇게 하면 서비스를 시작하고 중지하거나, 자동 시작을 비활성화하는 등의 작업을 할 수 있습니다.

관리 에이전트의 자동 재시작 확인

프로덕션 배포에서 관리 에이전트를 자동으로 재시작하도록 구성합니다. 이렇게 하면 `ma` 프로세스가 실패하거나 운영 체제가 재부트될 경우 관리 에이전트의 가용성이 보장됩니다.

Windows 플랫폼의 경우, 관리 에이전트를 서비스로 시작한 후 Windows 관리 도구를 사용하여 서비스 시작 유형을 “자동”으로 설정하고 원하는 복원 옵션을 설정합니다.

Solaris 및 Linux 플랫폼의 경우, 이 절의 절차를 사용하여 관리 에이전트의 자동 재시작을 구성합니다. 이러한 절차는 시스템이 다음 수준에 들어갈 때만 관리 에이전트가 시작되도록 합니다.

- Solaris 실행 수준 3(기본값)
- RedHat Linux 실행 수준 5(그래픽 모드 기본값)

다른 실행 수준에 들어가면 관리 에이전트가 중지됩니다.

▼ Solaris 또는 Linux에서 Java Enterprise System을 사용하여 자동 재시작을 구성하는 방법

시작하기 전에 이 절에서는 사용자가 운영 체제 초기화 및 실행 수준에 대한 기본 사항을 이해하고 있는 것으로 가정합니다. 이 항목의 내용에 대해서는 운영 체제 설명서를 참조하십시오.

1 시스템의 기본 실행 수준이 3 또는 5인지 확인합니다.

시스템의 기본 실행 수준을 확인하려면 `/etc/inittab` 파일을 검사하여 맨 위 근처의 행에서 다음과 비슷한 내용을 찾습니다.

```
id:5:initdefault:
```

이 예에서는 기본 실행 수준 5를 보여줍니다.

2 55 페이지 "소프트 링크 만들기"에 설명된 대로 `/etc/init.d/ma-initd` 파일에 대한 소프트 링크를 만듭니다.

3 시스템을 재부트합니다.

다음 순서 에이전트의 자동 시작 및 중지를 비활성화하려면 링크를 제거하거나 링크 이름의 문자 K와 S를 소문자로 변경합니다.

▼ Solaris 또는 Linux에서 독립 실행형 Application Server를 사용하여 자동 재시작을 구성하는 방법

1 셸에서 현재 디렉토리를 `HADB_install_dir/bin`으로 변경합니다.

2 셸 스크립트 `ma-initd`를 편집합니다.

스크립트의 `HADB_ROOT` 및 `HADB_MA_CFG` 기본값에 설치가 반영되어 있는지 확인합니다.

- `HADB_ROOT`는 HADB 설치 디렉토리 `HADB_install_dir`입니다.
- `HADB_MA_CFG`는 관리 에이전트 구성 파일의 위치입니다. 자세한 내용은 58 페이지 "관리 에이전트 구성 사용자 정의"를 참조하십시오.

3 `ma-initd`를 `/etc/init.d` 디렉토리로 복사합니다.

4 55 페이지 "소프트 링크 만들기"에 설명된 대로 `/etc/init.d/ma-initd` 파일에 대한 소프트 링크를 만듭니다.

다음 순서 에이전트의 자동 시작 및 중지를 비활성화하려면 링크를 제거하거나 링크 이름의 문자 K와 S를 소문자로 변경합니다.

소프트 링크 만들기

Solaris의 경우 다음 소프트 링크를 만듭니다.

```
/etc/rc0.d/K20ma-initd
/etc/rc1.d/K20ma-initd
/etc/rc2.d/K20ma-initd
/etc/rc3.d/S99ma-initd
/etc/rc5.d/K20ma-initd(Sun 4m 및 4u 아키텍처 전용)
/etc/rc6.d/K20ma-initd
/etc/rcS.d/K20ma-initd
```

Linux의 경우 다음 소프트 링크를 만듭니다.

```
/etc/rc0.d/K20ma-initd
/etc/rc1.d/K20ma-initd
/etc/rc3.d/S99ma-initd
/etc/rc5.d/S99ma-initd
/etc/rc6.d/K20ma-initd
```

콘솔 모드에서 관리 에이전트 시작

평가나 테스트를 위해 관리 에이전트를 콘솔 모드에서 시작해야 하는 경우가 있습니다. **ma** 프로세스는 시스템 또는 프로세스 실패 후 다시 시작되지 않고 명령 창을 닫으면 종료되므로 프로덕션 환경에서는 관리 에이전트를 이 방법으로 시작하지 마십시오. 명령은 플랫폼 및 설치에 따라 다릅니다.

- 55 페이지 “Solaris 또는 Linux에서의 Java Enterprise System”
- 55 페이지 “Windows에서의 Java Enterprise System”
- 56 페이지 “Windows에서의 독립 실행형 Application Server”
- 56 페이지 “Solaris 또는 Linux에서의 독립 실행형 Application Server”

Solaris 또는 Linux에서의 Java Enterprise System

HADB 관리 에이전트를 콘솔 모드에서 시작하려면 다음 명령을 사용합니다.

```
opt/SUNWhadb/bin/ma [config-file]
```

기본 관리 에이전트 구성 파일은 `/etc/opt/SUNWhadb/mgt.cfg`입니다.

관리 에이전트를 중지하려면 프로세스를 종료하거나 쉘 창을 닫습니다.

Windows에서의 Java Enterprise System

관리 에이전트를 콘솔 모드에서 시작하려면 다음 명령을 사용합니다.

```
HADB_install_dir\bin\ma [config-file]
```

선택 인수 *config-file*은 관리 에이전트 구성 파일의 이름입니다. 구성 파일에 대한 자세한 내용은 58 페이지 “관리 에이전트 구성 사용자 정의”를 참조하십시오.

에이전트를 중지하려면 프로세스를 종료합니다.

Windows에서의 독립 실행형 Application Server

관리 에이전트를 콘솔 모드에서 시작하려면 다음 명령을 사용합니다.

```
HADB_install_dir\bin\ma [config-file]
```

선택 인수인 *config-file*은 관리 에이전트 구성 파일의 이름입니다. 자세한 내용은 58 페이지 “관리 에이전트 구성 사용자 정의”를 참조하십시오.

관리 에이전트를 중지하려면 프로세스를 중지합니다.

Solaris 또는 Linux에서의 독립 실행형 Application Server

HADB 관리 에이전트를 콘솔 모드에서 시작하려면 다음 명령을 사용합니다.

```
HADB_install_dir/bin/ma [config-file]
```

기본 관리 에이전트 구성 파일은 *HADB_install_dir/bin/ma.cfg*입니다.

관리 에이전트를 중지하려면 프로세스를 종료하거나 쉘 창을 닫습니다.

Solaris 10 Service Management Facility를 사용하여 관리 에이전트 실행

SMF(Service Management Facility)는 Solaris 10에서 서비스를 재시작, 보기 및 관리할 수 있는 메커니즘을 제공합니다. SMF를 사용하여 HADB 관리 에이전트를 시작, 재시작 및 관리할 수 있습니다.

관리 에이전트에 대한 FMRI(Fault Management Resource Identifier)는 `svc:/application/hadb-ma`입니다.

관리 에이전트 명령 구문

관리 에이전트 `ma` 명령의 구문은 다음과 같습니다.

```
ma [common-options]
[ service-options]
config-file
```

여기서,

- *common-options*는 56 페이지 “관리 에이전트 명령 구문”에서 설명하는 공통 옵션 중 하나 이상입니다.
- *service-options*는 56 페이지 “관리 에이전트 명령 구문”에서 설명하는 Windows 서비스 옵션 중 하나입니다.
- *config-file*은 관리 에이전트 구성 파일에 대한 전체 경로입니다. 자세한 내용은 58 페이지 “관리 에이전트 구성 사용자 정의”를 참조하십시오.

표 3-1 관리 에이전트 공통 옵션

옵션	설명	기본값
<code>--define name=value-D</code>	<i>value</i> 를 등록 정보 <i>name</i> 에 할당합니다. 여기서 등록 정보는 58 페이지 “구성 파일”에 정의된 등록 정보 중 하나입니다. 이 옵션은 여러 번 반복하여 사용할 수 있습니다.	없음
<code>--help-?</code>	도움말 정보를 표시합니다.	False
<code>--javahome path-j</code>	<i>path</i> 에 있는 Java Runtime Environment(1.4 이상)를 사용합니다.	없음
<code>--systemroot path-y</code>	일반적으로 %SystemRoot%에 설정된 운영 체제 루트에 대한 경로입니다.	없음
<code>--version-V</code>	버전 정보를 표시합니다.	False

표 3-2에서는 관리 에이전트를 Windows 서비스로 시작하기 위한 옵션에 대해 설명합니다. *-i*, *-r* 및 *-s* 옵션은 상호 배타적입니다. 즉 함께 사용할 수 없습니다.

Windows의 경우 구성 파일 또는 명령줄에 등록 정보 값에 대한 경로를 지정할 때, 공백이 포함된 파일 경로에 큰따옴표(")를 이스케이프 문자로 사용합니다. 드라이브 및 디렉토리 구분자인 : 및 \은 \" : 및 \"과 같이 큰따옴표와 백슬러시를 이스케이프 문자로 사용합니다.

표 3-2 관리 에이전트 서비스 옵션(Windows에만 해당)

옵션	설명	기본값
<code>--install-i</code>	에이전트를 Windows 서비스로 설치하고 서비스를 시작합니다. <i>-i</i> , <i>-r</i> 및 <i>-s</i> 옵션 중 하나만 사용합니다.	False
<code>--name servicename-n</code>	호스트에 여러 에이전트를 실행하는 경우 서비스에 대해 지정된 이름을 사용합니다.	HADBMgrAgent
<code>--remove-r</code>	Windows 서비스 관리자에서 서비스를 중지하고 에이전트를 삭제합니다. <i>-i</i> , <i>-r</i> 및 <i>-s</i> 옵션 중 하나만 사용합니다.	False
<code>--service-s</code>	에이전트를 Windows 서비스로 실행합니다. <i>-i</i> , <i>-r</i> 및 <i>-s</i> 옵션 중 하나만 사용합니다.	False

관리 에이전트 구성 사용자 정의

HADB에는 관리 에이전트 설정을 사용자 정의할 때 사용할 수 있는 구성 파일이 포함되어 있습니다. 구성 파일을 지정하지 않고 관리 에이전트를 시작하면 기본값이 사용됩니다. 구성 파일을 지정하는 경우 관리 에이전트는 해당 파일의 설정을 사용합니다. 도메인의 모든 호스트에 있는 구성 파일을 다시 사용할 수 있습니다.

▼ HADB 호스트의 관리 에이전트 구성을 사용자 정의하는 방법

- 1 관리 에이전트 구성 파일을 편집하고 값을 원하는 대로 설정합니다.
- 2 사용자 정의된 구성 파일을 인수로 지정하여 관리 에이전트를 시작합니다.

구성 파일

Java Enterprise System을 사용할 경우 구성 파일의 모든 항목이 주석으로 처리되어 있습니다. 기본 구성을 사용할 때는 변경할 필요가 없습니다. 관리 에이전트 구성을 사용자 정의하려면 파일에서 주석을 제거하고, 값을 원하는 대로 변경한 다음 구성 파일을 인수로 지정하여 관리 에이전트를 시작합니다.

관리 에이전트 구성 파일은 다음 위치에 설치됩니다.

- Solaris 및 Linux: `/etc/opt/SUNWhadb/mgt.cfg`
- Windows: `install_dir\lib\mgt.cfg`

독립 실행형 설치 프로그램을 사용하면 관리 에이전트 구성 파일은 다음 위치에 설치됩니다.

- Solaris 및 Linux: `HADB_install_dir/bin/ma.cfg`
- Windows: `HADB_install_dir\bin\ma.cfg`

다음 표에서는 구성 파일의 설정에 대해 설명합니다.

표 3-3 구성 파일 설정

설정 이름	설명	기본값
console.loglevel	콘솔 로그 수준. 유효한 값은 SEVERE, ERROR, WARNING, INFO, FINE, FINER, FINEST입니다.	WARNING
logfile.loglevel	로그 파일 로그 수준. 유효한 값은 SEVERE, ERROR, WARNING, INFO, FINE, FINER, FINEST입니다.	INFO
logfile.name	로그 파일의 이름 및 위치. 읽기/쓰기 액세스 권한이 있는 유효한 경로여야 합니다.	Solaris 및 Linux: <code>/var/opt/SUNWhadb/ma/ma.log</code> Windows: <code>HADB_install_dir\ma.log</code>

표 3-3 구성 파일 설정 (계속)

설정 이름	설명	기본값
ma.server.type	클라이언트 프로토콜. JMXMP만 지원됩니다.	jmxp
ma.server.jmxmp.port	내부(UDP) 및 외부(TCP) 통신을 위한 포트 번호. 양의 정수여야 합니다. 권장 범위는 1024-49151입니다.	1862
ma.server.maininternal.interfaces	인터페이스가 여러 개인 시스템의 내부 통신을 위한 인터페이스. 유효한 IPv4 주소 마스크여야 합니다. 도메인의 모든 관리 에이전트는 동일한 서브넷을 사용해야 합니다. 예를 들어, 호스트에 10.10.116.61과 10.10.124.61의 두 인터페이스가 있는 경우 첫 번째 인스턴스를 사용하려면 10.10.116.0/24를 사용합니다. 슬래시 뒤의 숫자는 서브넷 마스크의 비트 수를 나타냅니다.	없음
ma.server.dbdevicepath	HADB 장치 정보를 저장하기 위한 경로	Solaris 및 Linux: /var/opt/SUNWhadb/4 Windows: HADB_install_dir\device
ma.server.dbhistorypath	HADB 내역 파일을 저장하기 위한 경로	Solaris 및 Linux: /var/opt/SUNWhadb Windows: REPLACEDIR(런타임에 실제 URL로 바뀜)
ma.server.dbconfigpath	노드 구성 데이터를 저장하기 위한 경로	Solaris 및 Linux: /var/opt/SUNWhadb/dbdef Windows: C:\Sun\SUNWhadb\dbdef
repository.dr.path	도메인 저장소 파일에 대한 경로	Solaris 및 Linux: /var/opt/SUNWhadb/repository Windows: C:\Sun\SUNWhadb\repository

hadbm 관리 명령 사용

HADB 도메인, 데이터베이스 인스턴스 및 노드를 관리하려면 **hadbm** 명령줄 유틸리티를 사용합니다. **hadbm** 유틸리티(관리 클라이언트라고도 함)는 지정된 관리 에이전트에 관리 요청을 보내고, 저장소의 데이터베이스 구성에 대한 액세스 권한이 있는 관리 서버의 역할을 합니다.

이 절에서는 다음 항목이 있는 **hadbm** 명령줄 유틸리티에 대해 설명합니다.

- 60 페이지 “명령 구문”
- 60 페이지 “보안 옵션”
- 62 페이지 “일반 옵션”
- 62 페이지 “환경 변수”

명령 구문

hadbm 유틸리티는 *HADB_install_dir/bin* 디렉토리에 있습니다. *hadbm* 명령의 일반 구문은 다음과 같습니다.

```
hadbm subcommand
[-short-option [option-value]]
[--long-option [option-value]]
[operands]
```

하위 명령은 수행할 연산 또는 작업을 식별하며 대소문자를 구분합니다. 대부분의 하위 명령에는 한 개의 피연산자(일반적으로 *dbname*)가 있습니다.

옵션은 *hadbm*이 하위 명령을 수행하는 방식을 수정하며 대소문자를 구분합니다. 각 옵션에는 긴 형식과 짧은 형식이 있습니다. 짧은 형식 앞에는 한 개의 대시(-)를 붙이고 긴 형식 앞에는 두 개의 대시(--)를 붙입니다. 기능을 사용하기 위해 있어야 하는 부울 옵션을 제외하고 대부분의 옵션에는 인수 값이 필요합니다. 옵션은 명령을 성공적으로 실행하기 위해 필요합니다.

하위 명령에는 데이터베이스 이름이 필요한데 이 데이터베이스 이름을 지정하지 않으면 *hadbm*은 기본 데이터베이스인 *hadb*를 사용합니다.

예 3-1 *hadbm* 명령의 예

다음은 *status* 하위 명령의 예입니다.

```
hadbm status --nodes
```

보안 옵션

보안상의 이유로, 모든 *hadbm* 명령에는 관리자 암호가 필요합니다. 데이터베이스나 도메인을 만들 때 암호를 설정하려면 *--adminpassword* 옵션을 사용합니다. 그런 다음 데이터베이스나 도메인에 대해 작업을 수행할 때 해당 암호를 지정해야 합니다.

보안 기능을 향상시키려면 *--adminpasswordfile* 옵션을 사용하여 명령줄에 암호를 입력하는 대신 암호가 포함된 파일을 지정합니다. 다음 행을 사용하여 암호 파일에 암호를 정의합니다.

```
HADBM_ADMINPASSWORD=password
```

*password*를 암호로 바꿉니다. 파일의 다른 내용은 모두 무시됩니다.

--adminpassword 및 *--adminpasswordfile* 옵션을 모두 지정하면 *--adminpassword*가 우선합니다. 암호가 필요하지만 명령에 암호를 지정하지 않으면 *hadbm*은 암호를 지정하라는 메시지를 표시합니다.

주 - 관리자 암호는 데이터베이스나 도메인을 만들 때만 설정할 수 있으며 나중에 변경할 수 없습니다.

HADB에는 관리자 암호 이외에 데이터베이스 스키마를 수정하는 작업을 수행하기 위한 데이터베이스 암호도 필요합니다. 명령 `hadbm create`, `hadbm addnodes` 및 `hadbm refragment`를 사용할 때는 두 암호를 모두 사용해야 합니다.

--dbpassword 옵션을 사용하여 명령줄에 데이터베이스 암호를 지정합니다. 관리자 암호와 마찬가지로, 파일에 암호를 넣고 파일 위치를 지정하여 --dbpasswordfile 옵션을 사용할 수 있습니다. 다음 행을 사용하여 암호 파일에 암호를 설정합니다.

HADBM_DBPASSWORD=password

테스트나 평가에서는 데이터베이스나 도메인을 만들 때 --no-adminauthentication 옵션으로 암호 인증을 해제할 수 있습니다. 자세한 내용은 65 페이지 “데이터베이스 만들기” 및 64 페이지 “관리 도메인 만들기”를 참조하십시오.

다음 표에서는 hadbm 보안 명령줄 옵션을 요약하여 설명합니다.

표 3-4 hadbm 보안 옵션

옵션(짧은 형식)	설명
--adminpassword=password -w	데이터베이스나 도메인에 대한 관리자 암호를 지정합니다. 데이터베이스나 도메인을 만들 때 이 옵션을 사용하면 hadbm을 사용하여 데이터베이스나 도메인 관련 작업을 수행할 때마다 암호를 지정해야 합니다. 이 옵션 또는 --adminpasswordfile 중 하나만 사용할 수 있으며, 둘 다 사용할 수는 없습니다.
--adminpasswordfile=filepath -W	데이터베이스나 도메인에 대한 관리자 암호가 포함된 파일을 지정합니다. 데이터베이스나 도메인을 만들 때 이 옵션을 사용하면 hadbm을 사용하여 데이터베이스나 도메인 관련 작업을 수행할 때마다 암호를 지정해야 합니다. 이 옵션 또는 --adminpassword 중 하나만 사용할 수 있으며, 둘 다 사용할 수는 없습니다.
--no-adminauthentication -U	데이터베이스나 도메인을 만들 때 관리자 암호가 필요하지 않음을 지정하는 옵션입니다. 보안상의 이유로, 프로덕션 배포에서는 이 옵션을 사용하지 마십시오.
--dbpassword=password -P	데이터베이스 암호를 지정합니다. 데이터베이스를 만들 때 이 옵션을 사용하면 hadbm 명령을 사용하여 데이터베이스 관련 작업을 수행할 때마다 암호를 지정해야 합니다. HADB 시스템 사용자에게 대한 암호를 만듭니다. 암호는 8자 이상이어야 합니다. 이 옵션 또는 --dbpasswordfile 중 하나만 사용할 수 있으며, 둘 다 사용할 수는 없습니다.
--dbpasswordfile=filepath -P	HADB 시스템 사용자에게 대한 암호가 포함된 파일을 지정합니다. 이 옵션 또는 --dbpassword 중 하나만 사용할 수 있으며, 둘 다 사용할 수는 없습니다.

일반 옵션

일반 명령 옵션은 모든 **hadbm** 하위 명령에 사용할 수 있습니다. 모두 기본적으로 **false**인 부울 옵션입니다. 다음 표에서는 **hadbm** 일반 명령 옵션에 대해 설명합니다.

표 3-5 **hadbm** 일반 옵션

옵션(짧은 형식)	설명
--quiet -q	하위 명령을 설명 메시지 없이 자동으로 실행합니다.
--help -?	이 명령 및 지원되는 모든 하위 명령에 대한 간략한 설명을 표시합니다. 하위 명령이 필요하지 않습니다.
--version -V	hadbm 명령의 버전 세부 정보를 표시합니다. 하위 명령이 필요하지 않습니다.
--yes -y	하위 명령을 비대화식 모드에서 실행합니다.
--force -f	명령을 비대화식으로 실행하고 명령의 사후 조건이 이미 달성되면 오류가 발생하지 않습니다.
--echo -e	모든 옵션 및 사용자 정의된 값이나 기본값과 함께 하위 명령을 표시한 다음 하위 명령을 실행합니다.
--agent= <i>URL</i> -m	관리 에이전트에 대한 URL입니다. <i>URL</i> : <i>hostlist:port</i> . 여기서 <i>hostlist</i> 는 웹포로 구분된 호스트 이름 또는 IP 주소의 목록이며 <i>port</i> 는 관리 에이전트가 작동 중인 포트 번호입니다. 기본값은 localhost:1862입니다. 참고: 이 옵션은 hadbm addnodes 에 유효하지 않습니다.

환경 변수

편의상 명령 옵션을 지정하는 대신 환경 변수를 설정할 수 있습니다. 다음 표에서는 **hadbm** 명령 옵션에 해당하는 환경 변수에 대해 설명합니다.

표 3-6 **HADB** 옵션 및 환경 변수

긴 형식	짧은 형식	기본값	환경 변수
--adminpassword	-w	없음	\$HADB_ADMINPASSWORD

표 3-6 HADB 옵션 및 환경 변수 (계속)

긴 형식	짧은 형식	기본값	환경 변수
--agent	--m	localhost:1862	\$HADBM_AGENT
--datadevices	-a	1	\$HADBM_DATADEVICES
dbname	없음	hadb	\$HADBM_DB
--dbpassword	-p	없음	\$HADBM_DBPASSWORD
--dbpasswordfile	-P	없음	\$HADBM_DBPASSWORDFILE
--devicepath	-d	Solaris 및 Linux: /var/opt/SUNWhadb Windows: C:\Sun\AppServer\SUNWhadb\vers. 여기서 vers는 HADB 버전 번호입니다.	\$HADBM_DEVICEPATH
--devicesize	-z	없음	\$HADBM_DEVICESIZE
--echo	-e	False	\$HADBM_ECHO
--fast	-F	False	\$HADBM_FAST
--force	-f	False	\$HADBM_FORCE
--help	-?	False	\$HADBM_HELP
--historypath	-t	Solaris 및 Linux: /var/opt/SUNWhadb Windows: REPLACEDIR(런타임에 실제 URL로 바뀜)	\$HADBM_HISTORYPATH
--hosts	-H	없음	\$HADBM_HOSTS
--interactive	-i	True	\$HADBM_INTERACTIVE
--no-refragment	-r	False	\$HADBM_NOREFRAGMENT
--portbase	-b	15200	\$HADBM_PORTBASE
--quiet	-q	False	\$HADBM_QUIET
--repair	-R	True	\$HADBM_REPAIR
--rolling	-g	True	\$HADBM_ROLLING
--saveto	-o	없음	\$HADBM_SAVETO
--set	-S	없음	\$HADBM_SET
--spares	-s	0	\$HADBM_SPARES

표 3-6 HADB 옵션 및 환경 변수 (계속)

긴 형식	짧은 형식	기본값	환경 변수
--startlevel	-l	normal	\$HADBM_STARTLEVEL
--version	-V	False	\$HADBM_VERSION
--yes	-y	False	\$HADBM_YES

HADB 구성

이 절에서는 다음과 같은 기본 HADB 구성 작업에 대해 설명합니다.

- 64 페이지 “관리 도메인 만들기”
- 65 페이지 “데이터베이스 만들기”
- 70 페이지 “구성 속성 보기 및 수정”
- 75 페이지 “JDBC 연결 풀 구성”

관리 도메인 만들기

`hadbm createdomain` 명령은 지정된 HADB 호스트가 포함된 관리 도메인을 만들고 호스트 및 지속성 구성 저장소 간의 내부 통신 채널을 초기화합니다.

명령 구문은 다음과 같습니다.

```
hadbm createdomain
  [--adminpassword=password | --adminpasswordfile=
file | --no-adminauthentication] [--agent=maurl]
  hostlist
```

hostlist 피연산자는 쉼표로 구분된 HADB 호스트의 목록이며 각 호스트는 유효한 IPv4 네트워크 주소입니다. 새 도메인에 원하는 모든 호스트를 *hostlist*에 포함시킵니다.

명령 옵션에 대한 설명은 62 페이지 “일반 옵션”을 참조하십시오.

이 명령을 사용하기 전에 *hostlist*의 모든 호스트에 HADB 관리 에이전트가 실행되고 있는지 확인하십시오. 또한 관리 에이전트는 다음 조건에 맞아야 합니다.

- 기존 도메인의 구성원이 아니어야 합니다.
- 동일한 포트를 사용하도록 구성되어야 합니다.
- UDP, TCP를 통해, 그리고 IP 멀티캐스트를 사용하여 서로 통신할 수 있어야 합니다.

`hadbm`은 관리 도메인을 만든 후에 도메인의 모든 호스트를 활성화합니다. 그런 다음 관리 에이전트는 데이터베이스를 관리할 수 있게 됩니다. HADB 도메인을 만들고 나면, 다음 단계에서 HADB 데이터베이스를 만듭니다. HADB 데이터베이스 만들기에 대한 자세한 내용은 65 페이지 “데이터베이스 만들기”를 참조하십시오.

예 3-2 HADB 관리 도메인 만들기

다음 예에서는 네 개의 지정된 호스트에 관리 도메인을 만듭니다.

```
hadbm createdomain --adminpassword=password host1,host2,host3,host4
```

hadbm이 명령을 성공적으로 실행하면 다음 메시지가 표시됩니다.

```
Domain host1,host2,host3, host4 created.
```

HADB 도메인을 만든 후에는 HADB 패키지의 경로와 버전을 관리 에이전트에 등록합니다.

데이터베이스 만들기

hadbm create 명령을 사용하여 데이터베이스를 수동으로 만듭니다.

이 명령을 사용하여 데이터베이스를 만들기 전에 관리 도메인을 만들고 HADB 패키지를 등록합니다. **hadbm create**를 실행할 때 이 두 단계를 수행하지 않은 경우 이 명령은 해당 단계를 암시적으로 수행합니다. 거의 동작하지 않는 것처럼 보일 수 있지만 이 명령 중 하나라도 실패하면 디버깅이 어려워질 수 있습니다. **hadbm create**는 기본 단위가 아닙니다. 즉, 암시적 명령 중 하나가 실패하면 성공적으로 실행된 명령이 롤백되지 않습니다. 따라서 도메인을 만들고 HADB 패키지를 등록한 후에만 데이터베이스를 만드는 것이 가장 좋습니다.

예를 들어, **hadbm createdomain** 및 **hadbm registerpackage**는 성공적으로 실행되지만 **hadbm create database**가 실패하는 경우 **hadbm createdomain** 및 **hadbm registerpackage**에 의한 변경 사항이 지속됩니다.

▼ 데이터베이스를 만드는 방법

- 1 관리 도메인을 만듭니다.
자세한 내용은 [64 페이지 “관리 도메인 만들기”](#)를 참조하십시오.
- 2 HADB 패키지를 등록합니다.
자세한 내용은 [46 페이지 “HADB 패키지 등록”](#)을 참조하십시오.
- 3 **hadbm create** 명령을 사용하여 데이터베이스를 만듭니다.
명령 구문에 대한 자세한 내용은 다음 절을 참조하십시오.

hadbm create 명령 구문

```
hadbm create [--package=name] [--packagepath=path] [--historypath=path]  
[--devicepath=path] [--datadevices=number] [--portbase=number]
```

```
[--spares=number] [--set=attr-val-list] [--agent=maurl] [--no-cleanup]
[ --no-clear ] [ --devicesize =size] [--dbpassword=password | --dbpasswordfile=file
] --hosts=host list [--adminpassword=password | --adminpasswordfile=file |
--no-adminauthentication ] [dbname ]
```

dbname 피연산자는 고유해야 하는 데이터베이스 이름을 지정합니다. 데이터베이스 이름이 고유한지 확인하려면 `hadbm list` 명령을 사용하여 기존 데이터베이스 이름을 나열합니다. 여러 데이터베이스를 만들 필요가 없으면 기본 데이터베이스 이름을 사용합니다. 예를 들어 동일한 HADB 시스템 집합에 독립적인 데이터베이스가 있는 여러 클러스터를 만들려면 각 클러스터에 별도의 데이터베이스 이름을 사용합니다.

`hadbm create` 명령은 로그 파일이 아닌 콘솔에 오류 메시지를 기록합니다.

표 3-7에서는 특수한 `hadbm create` 명령 옵션에 대해 설명합니다. 추가적인 명령 옵션에 대한 설명은 62 페이지 “일반 옵션”을 참조하십시오.

표 3-7 hadbm create 옵션

옵션(짧은 형식)	설명	기본값
--datadevices= <i>number</i> -a	각 노드의 데이터 장치 수(1~8)데이터 장치에는 0부터 시작하는 번호가 지정됩니다.	1
--devicepath= <i>path</i> -d	장치에 대한 경로. 다음 네 개의 장치가 있습니다. <ul style="list-style-type: none"> ■ DataDevice ■ NiLogDevice(노드 내부 로그 장치) ■ RelalgDevice(관계형 algebra 쿼리 장치) ■ NoManDevice(노드 관리자 장치). 이 경로는 반드시 존재해야 하며 쓰기 가능해야 합니다. 각 노드 또는 각 장치마다 이 경로를 서로 다르게 설정하려면 69 페이지 “이기종 장치 경로 설정”을 참조하십시오. 	Solaris 및 Linux: /var/opt/SUNWhadb Windows: C:\Sun\AppServer\SUNWhadb\vers. 여기서 vers는 HADB 버전 번호입니다. 기본값은 관리 에이전트 구성 파일의 ma.server.dbdevicepath로 지정됩니다. 자세한 내용은 58 페이지 “구성 파일”을 참조하십시오.
--devicesize= <i>size</i> -z	각 노드의 장치 크기. 자세한 내용은 68 페이지 “장치 크기 지정”을 참조하십시오. 87 페이지 “기존 노드에 저장 공간 추가”에 설명된 대로 장치 크기를 늘립니다.	1024MB 최대 크기는 최대 운영 체제 파일 크기 또는 256GB 중 작은 크기입니다. 최대 크기는 다음과 같습니다. $(4 \times \text{LogbufferSize} + 16\text{MB}) / n$ 여기서 <i>n</i> 은 옵션 --datadevices에서 제공된 데이터 장치 수입니다.

표 3-7 hadbm create 옵션 (계속)

옵션(짧은 형식)	설명	기본값
--historypath= <i>path</i> -t	내역 파일에 대한 경로. 이 경로는 반드시 존재해야 하며 쓰기 가능해야 합니다. 내역 파일에 대한 자세한 내용은 101 페이지 “내역 파일 지우기 및 아카이브”를 참조하십시오.	기본값은 관리 에이전트 구성 파일의 <code>ma.server.dbhistorypath</code> 로 지정됩니다. 자세한 내용은 58 페이지 “구성 파일”을 참조하십시오. Solaris 및 Linux: <code>/var/opt/SUNWhadb</code> Windows: <code>REPLACEDIR</code> (런타임에 실제 URL로 바뀜)
--hosts= <i>hostlist</i> -H	데이터베이스의 노드에 대한 호스트 이름 또는 IP 주소(IPv4만)의 집합으로 구분된 목록입니다. DNS 조회에 대한 종속성을 방지하려면 IP 주소를 사용합니다. 호스트 이름은 절대적이어야 합니다. <code>localhost</code> 또는 <code>127.0.0.1</code> 을 호스트 이름으로 사용할 수 없습니다. 호스트 이름에 대한 자세한 내용은 68 페이지 “호스트 지정”을 참조하십시오.	없음
--package= <i>name</i> -k	HADB 패키지의 이름(버전). 패키지가 없으면 기본 패키지가 등록됩니다. 이 옵션은 더 이상 사용되지 않습니다. 도메인의 패키지를 등록하려면 <code>hadbm registerpackage</code> 명령을 사용합니다.	없음
--packagepath= <i>path</i> -L	HADB 소프트웨어 패키지에 대한 경로. 패키지가 도메인에 등록되지 않은 경우에만 사용합니다. 이 옵션은 더 이상 사용되지 않습니다. 도메인의 패키지를 등록하려면 <code>hadbm registerpackage</code> 명령을 사용합니다.	없음
--portbase= <i>number</i> -b	노드 0에 사용되는 포트 기본 번호입니다. 연속되는 노드는 이 숫자부터 20단계로 자동 할당되는 포트 기본 번호입니다. 각 노드는 포트 기본 번호 및 다음 다섯 개의 연속되는 번호가 지정된 포트를 사용합니다. 동일한 시스템에서 여러 데이터베이스를 실행하려면 포트 번호를 명시적으로 할당하도록 계획해야 합니다.	15200
--spares= <i>number</i> -s	예비 노드의 수. 이 숫자는 짝수여야 하며 --hosts 옵션에 지정된 노드 수보다 작아야 합니다.	0

표 3-7 hadbm create 옵션 (계속)

옵션(짧은 형식)	설명	기본값
--set=attr-val-list	데이터베이스 구성 속성(name=value 형식)의 쉼표로 구분된 목록. 데이터베이스 구성 속성에 대한 설명은 101 페이지 “내역 파일 지우기 및 아카이브” 를 참조하십시오.	없음
-S		

예 3-3 데이터베이스 만들기의 예

다음 명령은 데이터베이스를 만드는 예입니다.

```
hadbm create --spares 2 --devicesize 1024 --hosts n0,n1,n2,n3,n4,n5
```

호스트 지정

--hosts 옵션을 사용하여 데이터베이스의 노드에 대한 호스트 이름 또는 IP 주소의 쉼표로 구분된 목록을 지정합니다. hadbm create 명령은 목록의 각 호스트 이름(또는 IP 주소)에 대해 한 개의 노드를 만듭니다. 노드 수는 짝수여야 합니다. 중복되는 호스트 이름을 사용하여 동일한 시스템에 포트 번호가 다른 여러 노드를 만듭니다. 동일한 시스템의 노드가 미리 노드가 아니며 다른 DRU에서 제공되지 않았는지 확인합니다.

노드에는 이 옵션에 나열된 순서로 0부터 시작하여 번호가 지정됩니다. 첫 번째 미리되는 쌍은 노드 0과 1이고, 두 번째는 2와 3 등의 순서입니다. 홀수 번호가 지정된 노드는 한 DRU에 있고, 짝수 번호가 지정된 노드는 다른 DRU에 있습니다. --spares 옵션을 사용하는 예비 노드는 숫자가 가장 큼니다.

이중 네트워크 인터페이스 구성에 대한 자세한 내용은 [34 페이지 “네트워크 중복 구성”](#)을 참조하십시오.

장치 크기 지정

--devicesize 옵션을 사용하여 장치 크기를 지정합니다. 권장하는 장치 크기는 다음과 같습니다.

$$(4x / nd + 4l/d) / 0.99$$

여기서,

- x 는 사용자 데이터의 총 크기입니다.
- n 은 노드 수(--hosts 옵션으로 제공됨)입니다.
- d 는 노드당 장치 수(--datadevices 옵션으로 제공됨)입니다.
- l 은 로그 버퍼 크기(속성 LogBufferSize로 제공됨)입니다.

재단편화가 발생할 수 있는 경우(예: hadbm addnodes 사용) 권장하는 장치 크기는 다음과 같습니다.

$$(8x / nd + 4l/d) / 0.99$$

이기종 장치 경로 설정

각 노드 또는 서비스에 대해 서로 다른 장치 경로를 설정하려면 `hadbm create`의 `-- set` 옵션을 사용합니다. 장치 유형에는 `DataDevice`, `NiLogDevice`(노드 내부 로그 장치), `RelalgDevice`(관계형 algebra 쿼리 장치) 및 `NoManDevice`(노드 관리자 장치)의 네 가지 유형이 있습니다. 각 `name=value` 쌍의 구문은 다음과 같습니다. 여기서 `-devno`는 `device`가 `DataDevice`인 경우에만 필수입니다.

```
node-nodeno.device-devno.Devicepath
```

예를 들면 다음과 같습니다.

```
--set Node-0.DataDevice-0.DevicePath=/disk0,
Node-1.DataDevice-0.DevicePath=/disk 1
```

다음과 같이 내역 파일에 이기종 경로를 설정할 수도 있습니다.

```
node-nodeno.historypath=path
```

내역 파일에 대한 자세한 내용은 [101 페이지 “내역 파일 지우기 및 아카이브”](#)를 참조하십시오.

특정 노드 또는 장치로 설정되지 않은 장치 경로의 기본값은 `--devicepath`입니다.

주 - `hadbm set` 및 `hadbm addnodes` 명령을 사용하여 내역 파일의 위치와 장치 경로를 변경합니다.

문제 해결

데이터베이스를 만들 때 문제가 발생하는 경우 다음을 확인하십시오.

- 모든 호스트에서 관리 에이전트를 시작했으며 HADB 도메인을 정의했는지 확인합니다. 자세한 내용은 [51 페이지 “관리 에이전트 시작”](#)을 참조하십시오.
- 다음 사용자에 대해 설치, 내역, 장치 및 구성 경로에 대한 읽기, 쓰기 및 실행 액세스 권한이 허용되도록 파일 및 디렉토리 권한을 설정해야 합니다.
 - Sun Java System Application Server 관리 사용자(설치 도중 설정됨)
 - HADB 시스템 사용자

사용자 권한 설정에 대한 자세한 내용은 [33 페이지 “HADB 설정 준비”](#)를 참조하십시오.

Application Server 및 HADB 포트 할당은 동일한 시스템의 다른 포트 할당과 충돌하지 않아야 합니다. 권장하는 기본 포트 할당은 다음과 같습니다.

- Sun Java SystemMessage Queue: 7676
- IIOP: 3700
- HTTP 서버: 80

- 관리 서버: 4848
- HADB 노드: 각 노드는 6개의 연속되는 포트를 사용합니다. 예를 들어 기본 포트 15200의 경우, 노드 0은 15200부터 15205까지 사용하고 노드 1은 15220부터 15225까지 사용합니다.

디스크 공간이 충분해야 합니다. **Sun Java System Application Server Enterprise Edition 8.2 릴리스 노트**를 참조하십시오.

구성 속성 보기 및 수정

속성 `hadbm get` 및 `hadbm set` 명령을 각각 사용하여 데이터베이스 구성을 보고 수정할 수 있습니다.

구성 속성의 값 가져오기

구성 속성의 값을 가져오려면 `hadbm get` 명령을 사용합니다. 유효한 속성 목록은 [71 페이지 “구성 속성”](#)을 참조하십시오. 명령 구문은 다음과 같습니다.

```
hadbm get attribute-list | --all
[dbname]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

`dbname` 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 `hadb`입니다.

`attribute-list` 피연산자는 쉼표로 구분되거나 따옴표가 붙은 공백으로 구분된 속성 목록입니다. `--all` 옵션은 모든 속성의 값을 표시합니다. `hadbm get`에 대한 모든 속성의 목록은 [71 페이지 “구성 속성”](#)을 참조하십시오.

명령 옵션에 대한 설명은 [62 페이지 “일반 옵션”](#)을 참조하십시오.

예 3-4 `hadbm get` 사용의 예

```
hadbm get JdbcUrl,NumberOfSessions
```

구성 속성의 값 설정

구성 속성의 값을 설정하려면 `hadbm set` 명령을 사용합니다. 유효한 속성 목록은 [71 페이지 “구성 속성”](#)을 참조하십시오.

```
hadbm set [dbname] attribute
=value[,attribute=
value...]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 *hadb*입니다.

attribute=value 목록은 쉼표로 구분되거나 따옴표가 붙은 공백으로 구분된 속성 목록입니다.

명령 옵션에 대한 설명은 [62 페이지 “일반 옵션”](#)을 참조하십시오.

이 명령이 성공적으로 실행되면 데이터베이스가 이전 상태 또는 이전보다 향상된 상태로 다시 시작됩니다. 데이터베이스 상태에 대한 자세한 내용은 [92 페이지 “HADB 상태 가져오기”](#)를 참조하십시오. [83 페이지 “데이터베이스 재시작”](#)에 설명된 대로 HADB를 다시 시작합니다.

*hadbm set*으로는 다음 속성을 설정할 수 없습니다. 대신, 데이터베이스를 만들 때 해당 속성을 설정합니다([65 페이지 “데이터베이스 만들기”](#) 참조).

- DatabaseName
- DevicePath
- HistoryPath
- NumberOfDataDevices
- Portbase
- JdbcUrl(값은 *--hosts* 및 *--portbase* 옵션을 기반으로 데이터베이스 작성 도중 설정됨)

주 - *hadbm set*을 사용하여 *ConnectionTrace* 또는 *SQLTraceMode*를 제외한 구성 속성을 설정하면 HADB의 롤링이 재시작됩니다. 롤링 재시작에서 각 노드가 중지되고 한 번에 하나씩 새 구성으로 시작됩니다. HADB 서비스는 인터럽트되지 않습니다.

ConnectionTrace 또는 *SQLTraceMode*를 설정하면 롤링 재시작이 발생하지 않지만 변경 내용은 Application Server 인스턴스에서 만든 새 HADB에만 적용됩니다.

구성 속성

다음 표에서는 *hadbm set*으로 수정하고 *hadbm get*으로 검색할 수 있는 구성 속성을 나열합니다.

표 3-8 구성 속성

속성	설명	기본값	범위
ConnectionTrace	값이 true이면, 클라이언트 연결(JDBC, ODBC)이 초기화 또는 종료될 때 HADB 내역 파일에 메시지를 기록합니다.	False	True 또는 False

표 3-8 구성 속성 (계속)

속성	설명	기본값	범위
CoreFile	기본값을 변경하지 마십시오.	False	True 또는 False
DatabaseName	데이터베이스의 이름	hadb	
DataBufferPoolSize	공유 메모리에 할당된 데이터 버퍼 풀의 크기	200MB	16 - 2047MB
DataDeviceSize	<p>노드의 장치 크기를 지정합니다. 권장하는 <code>DataDeviceSize</code>에 대한 자세한 내용은 68 페이지 “장치 크기 지정”을 참조하십시오.</p> <p>최대 값은 256GB 또는 최대 운영 체제 파일 크기 중 작은 값입니다. 최소 값은 다음과 같습니다.</p> $(4 \times \text{LogbufferSize} + 16\text{MB}) / n$ <p>여기서 n은 데이터 장치 수입니다.</p>	1024MB	32 - 262144MB
PackageName	데이터베이스에서 사용하는 HADB 소프트웨어 패키지의 이름	V4.x.x.x	없음
DevicePath	<p>장치의 위치. 장치는 다음과 같습니다.</p> <ul style="list-style-type: none"> ■ 데이터 장치(DataDevice) ■ 노드 내부 로그 장치(NiLogDevice) ■ 관계형 algebra 쿼리 장치(RelalgDevice) 	<p>Solaris 및 Linux: /var/opt/SUNWhadb</p> <p>Windows: C:\Sun\AppServer\SUNWhadb\vers. 여기서 vers는 HADB 버전 번호입니다.</p>	
EagerSessionThreshold	<p>일반 또는 eager 유틸 세션 만료 중 사용되는 항목을 확인합니다.</p> <p>일반 유틸 세션 만료의 경우 SessionTimeout초를 초과하여 유틸 상태인 세션이 만료됩니다.</p> <p>동시 세션의 수가 최대 세션 수의 EagerSessionThreshold 퍼센트를 초과하면 EagerSessionTimeout초를 초과하여 유틸 상태인 세션이 만료됩니다.</p>	NumberOfSessions	0 - 100 속성의 절반
EagerSessionTimeout	eager 세션 만료가 사용될 때 데이터베이스 연결이 만료되기 전까지 유틸 상태일 수 있는 시간(초)입니다.	120초	0-2147483647초

표 3-8 구성 속성 (계속)

속성	설명	기본값	범위
EventBufferSize	데이터베이스 이벤트가 기록되는 이벤트 버퍼의 크기. 0으로 설정하면 이벤트 버퍼 로깅이 수행되지 않습니다. 장애 발생 시에는 이벤트 버퍼가 덤프되는데, 이는 실패의 원인에 대한 중요한 정보를 제공하며 테스트 배포 도중 유용합니다. 이벤트를 메모리에 기록하면 성능 저하가 발생할 수 있습니다.	0MB	0-2097152MB
HistoryPath	정보, 경고 및 오류 메시지가 포함된 HADB 내역 파일의 위치 읽기 전용 속성입니다.	Solaris 및 Linux: /var/opt/SUNWhadb Windows: REPLACEDIR(런타임에 실제 URL로 바뀜)	
InternalLogbufferSize	데이터 저장 관련 작업을 추적하는 노드 내부 로그 장치의 크기	12MB	4 - 128MB
JdbcUrl	데이터베이스에 대한 JDBC 연결 URL 읽기 전용 속성입니다.	없음	
LogbufferSize	데이터 관련 작업을 추적하는 로그 버퍼의 크기	48MB	4 - 2048MB
MaxTables	HADB 데이터베이스에서 허용되는 최대 테이블 수	1100	100 - 1100
NumberOfDatadevices	HADB 노드에서 사용하는 데이터 장치 수 읽기 전용 속성입니다.	1	1 - 8
NumberOfLocks	HADB 노드에서 할당된 잠금 수	50000	20000-1073741824
NumberOfSessions	HADB 노드에 대해 열 수 있는 최대 세션(데이터베이스 연결) 수	100	1 - 10000
PortBase	HADB 프로세스마다 서로 다른 포트 번호를 만드는 데 사용되는 기본 포트 번호 읽기 전용 속성입니다.	15200	10000 - 63000
RelalgDeviceSize	관계형 algebra 쿼리에 사용되는 장치의 크기	128MB	32 - 262144MB
SessionTimeout	일반 세션 만료가 사용될 때 데이터베이스 연결이 만료되기 전까지 유휴 상태일 수 있는 시간(초)입니다.	1800초	0-2147483647초

표 3-8 구성 속성 (계속)

속성	설명	기본값	범위
SQLTraceMode	내역 파일에 기록되어 실행된 SQL 쿼리에 대한 정보의 양 SHORT인 경우 SQL 세션의 로그인과 로그아웃이 기록됩니다. FULL인 경우 매개 변수 값을 포함하여 준비 및 실행 중인 모든 SQL 쿼리가 기록됩니다.	NONE	NONE/SHORT/ FULL
StartRepairDelay	실패한 활성 노드가 노드 복원을 수행하도록 예비 노드가 허용하는 최대 시간. 실패한 노드가 이 시간 간격 내에 복원될 수 없으면 예비 노드는 실패한 노드의 미러에서 데이터를 복사하기 시작하고 활성 상태가 됩니다. 기본값은 변경하지 않는 것이 좋습니다.	20초	0 - 100000초
StatInterval	HADB 노드가 처리량과 응답 시간 통계를 내역 파일에 기록하는 간격. 이 값을 비활성화하려면 0으로 설정합니다. 통계 행의 예는 다음과 같습니다. Req-reply time: # 123, min= 69 avg= 1160 max= 9311 %=100.0 # 기호 뒤의 숫자는 StatInterval을 통해 서비스된 요청의 수입입니다. 다음의 세 숫자는 StatInterval을 통해 완료된 트랜잭션에 걸린 최소, 평균 및 최대 시간(밀리초)입니다. 백분을 기호% 뒤의 숫자는 StatInterval을 통해 15밀리초 안에 성공적으로 완료된 트랜잭션의 수입입니다.	600초	0 - 600초
SyslogFacility	syslog에 보고할 때 사용되는 기능syslog 데몬이 구성되어야 합니다(자세한 내용은 man syslogd.conf 참조). 동일한 시스템에서 실행 중인 다른 응용 프로그램에서 사용하지 않는 기능을 사용합니다. syslog 로깅을 비활성화하려면 none으로 설정합니다.	local0	local0, local1, local2, local3, local4, local5, local6, local7, kern, user, mail, daemon, auth, syslog, lpr, news, uucp, cron, none
SysLogging	값이 true이면, HADB 노드는 운영 체제의 syslog 파일에 정보를 기록합니다.	True	True 또는 False

표 3-8 구성 속성 (계속)

속성	설명	기본값	범위
SysLogLevel	운영 체제의 <code>syslog</code> 파일에 저장된 HADB 메시지의 최소 수준. 이 수준과 같거나 높은 모든 메시지가 기록됩니다. 예를 들어 “info”는 모든 메시지를 기록합니다.	warning	none, alert, error, warning, info
SyslogPrefix	HADB에서 기록된 모든 <code>syslog</code> 메시지 앞에 삽입되는 텍스트 문자열	hadb-dbname	
TakeoverTime	노드가 실패한 시간과 미러를 시작하는 시간 사이의 시간. 기본값을 변경하지 마십시오.	10000(밀리초)	500 - 16000밀리초

JDBC 연결 풀 구성

Application Server는 JDBC(Java Database Connectivity) API를 사용하여 HADB와 통신합니다. `asadmin configure-ha-cluster` 명령은 `cluster-name` 클러스터에 대해 HADB와 함께 사용할 JDBC 연결 풀을 자동으로 만듭니다. 연결 풀의 이름은 “`cluster-name-hadb-pool`”입니다. JDBC 자원의 JNDI URL은 “`jdbc/cluster-name-hastore`”입니다.

일반적으로 연결 풀의 초기 구성이면 충분합니다. 노드를 추가할 때 각 HADB 활성 노드에 8개의 연결이 있도록 고정 풀 크기를 변경합니다. [88 페이지 “노드 추가”](#)를 참조하십시오.

이 절에서는 다음 항목에 대해 설명합니다.

- [75 페이지 “JDBC URL 가져오기”](#)
- [76 페이지 “연결 풀 만들기”](#)
- [77 페이지 “JDBC 자원 만들기”](#)

연결 풀 및 JDBC 자원에 대한 일반 정보는 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 3 장, “JDBC 자원”을 참조하십시오.

JDBC URL 가져오기

JDBC 연결 풀을 설정하려면 먼저 다음과 같이 `hadbm get` 명령을 사용하여 HADB의 JDBC URL을 확인해야 합니다.

```
hadbm get JdbcUrl [dbname]
```

예를 들면 다음과 같습니다.

```
hadbm get JdbcUrl
```

이 명령은 다음과 같은 형식의 JDBC URL을 표시합니다.

```
jdbc:sun:hadb:host:port,  
host:port,...
```

jdbc:sun:hadb: 접두어를 제거하고 *host:port, host:port...* 부분을 표 3-10에 설명된 대로 `serverList` 연결 풀 등록 정보의 값으로 사용합니다.

연결 풀 만들기

다음 표에서는 HADB에 필요한 연결 풀 설정을 요약하여 설명합니다. 노드를 추가할 때 고정 풀 크기를 변경합니다. 그러나 다른 설정은 변경하지 마십시오.

표 3-9 HADB 연결 풀 설정

설정	HADB에 필요한 값
Name	HADB JDBC 자원의 풀 이름 설정은 이 이름을 참조해야 합니다.
Database Vendor	HADB 4.4
Global Transaction Support	선택하지 않음/false
DataSource Classname	com.sun.hadb.jdbc.ds.HadbDataSource
Steady Pool Size	각 활성 HADB노드에 대해 8개의 연결을 사용합니다. 자세한 내용은 System Deployment Guide 를 참조하십시오.
Connection Validation Required	선택함/true
Validation Method	meta-data
Table Name	지정하지 않습니다.
Fail All Connections	선택하지 않음/false
Transaction Isolation	repeatable-read
Guarantee Isolation Level	선택함/true

다음은 HADB에 필요한 연결 풀 등록 정보를 요약한 표입니다. 노드를 추가할 때 `serverList`를 변경합니다. 그러나 다른 등록 정보는 변경하지 마십시오.

표 3-10 HADB 연결 풀 등록 정보

등록 정보	설명
username	asadmin create-session-store 명령에 사용할 storeuser의 이름입니다.
password	asadmin create-session-store 명령에 사용할 암호(storepassword)입니다.

표 3-10 HADB 연결 풀 등록 정보 (계속)

등록 정보	설명
serverList	HADB의 JDBC URL입니다. 이 값을 확인하려면 75 페이지 “JDBC URL 가져오기”를 참조하십시오. 데이터베이스에 노드를 추가하는 경우 이 값을 변경해야 합니다. 88 페이지 “노드 추가”를 참조하십시오.
cacheDatabaseMetaData	값이 false이면, Connection.getMetaData()를 호출할 때 데이터베이스에 대한 호출이 이루어져 연결이 유효한지 확인합니다.
eliminateRedundantEndTransaction	값이 true이면, 중복되는 완결/롤백 요청을 제거하고 열려 있는 트랜잭션이 없을 때 이들 요청을 무시하여 성능을 향상시킵니다.
maxStatement	드라이버 문 풀에 캐시된 열려 있는 연결당 문의 최대 수입니다. 이 등록 정보를 20으로 설정하십시오.

예 3-5 연결 풀 만들기

HADB JDBC 연결 풀을 만드는 `asadmin create-jdbc-connection-pool` 명령의 예는 다음과 같습니다.

```
asadmin create-jdbc-connection-pool
--user adminname --password secret
--datasourceclassname com.sun.hadb.jdbc.ds.HadbDataSource
--steadypoolsize=32
--isolationlevel=repeatable-read
--isconnectvalidatereq=true
--validationmethod=meta-data
--property username=storename:password=secret456:serverList=
host\:port,host\:port,
host\:port,host\:port,
host\:port,host\:port
:cacheDatabaseMetaData=false:eliminateRedundantEndTransaction=true hadbpool
```

Solaris에서는 등록 정보 값 내의 콜론 문자(:)에 이중 백슬래시(\\)를 이스케이프 문자로 사용합니다. Windows에서는 콜론 문자(:)에 단일 백슬래시(\)를 이스케이프 문자로 사용합니다.

JDBC 자원 만들기

다음 표에서는 HADB에 필요한 JDBC 자원 설정을 요약하여 설명합니다.

표 3-11 HADB JDBC 자원 설정

설정	설명
JNDI Name	다음 JNDI 이름은 세션 지속성 구성의 기본값입니다. jdbc/hastore. 기본 이름을 사용하거나 다른 이름을 사용할 수 있습니다. 또한 가용성 서비스를 활성화할 때 이 JNDI 이름을 store-pool-jndi-name 지속성 저장소 등록 정보의 값으로 지정해야 합니다.
Pool Name	이 JDBC 자원에서 사용하는 HADB 연결 풀의 이름 또는 아이디를 목록에서 선택합니다. 자세한 내용은 34 페이지 “네트워크 중복 구성” 을 참조하십시오.
Data Source Enabled	선택함/true

HADB 관리

일반적으로 네트워크, 하드웨어, 운영 체제 또는 HADB 소프트웨어를 바꾸거나 업그레이드할 때 관리 작업을 수행해야 합니다. 다음 절에서는 다양한 관리 작업에 대해 설명합니다.

- [78 페이지 “도메인 관리”](#)
- [79 페이지 “노드 관리”](#)
- [82 페이지 “데이터베이스 관리”](#)
- [86 페이지 “세션 데이터 손상에서 복원”](#)

도메인 관리

HADB 도메인에서 다음 작업을 수행할 수 있습니다.

- 도메인 만들기에 대한 자세한 내용은 [64 페이지 “관리 도메인 만들기”](#)를 참조하십시오.
- [78 페이지 “도메인 확장”](#)
- [79 페이지 “도메인 삭제”](#)
- [79 페이지 “도메인의 호스트 나열”](#)
- [79 페이지 “도메인에서 호스트 제거”](#)

명령 옵션에 대한 설명은 [60 페이지 “보안 옵션”](#) 및 [62 페이지 “일반 옵션”](#)을 참조하십시오.

도메인 확장

기존 관리 도메인에 호스트를 추가하려면 `extenddomain`을 사용합니다. 명령 구문은 다음과 같습니다.

```
hadbm extenddomain
[--adminpassword=password | --adminpasswordfile=file]
```

```
[--agent=maurl]
hostlist
```

HADB 호스트의 IP 주소는 IPv4 주소여야 합니다.

자세한 내용은 `hadbm-extenddomain(1)`을 참조하십시오.

도메인 삭제

관리 도메인을 제거하려면 `deletedomain to`를 사용합니다. 명령 구문은 다음과 같습니다.

```
hadbm deletedomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

자세한 내용은 `hadbm-deletedomain(1)`을 참조하십시오.

도메인에서 호스트 제거

관리 도메인에서 호스트를 제거하려면 `reducedomain`을 사용합니다. 명령 구문은 다음과 같습니다.

```
hadbm reducedomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
host_list
```

자세한 내용은 `hadbm-reducedomain(1)`을 참조하십시오.

도메인의 호스트 나열

관리 도메인에 정의된 모든 호스트를 나열하려면 `listdomain`을 사용합니다. 명령 구문은 다음과 같습니다.

```
hadbm listdomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

자세한 내용은 `hadbm-listdomain(1)`을 참조하십시오.

노드 관리

개별 노드에서 다음 작업을 수행할 수 있습니다.

- [80 페이지 “노드 시작”](#)
- [81 페이지 “노드 중지”](#)

- 81 페이지 “노드 재시작”

노드 시작

하드웨어나 소프트웨어의 업그레이드 또는 교체로 인해 호스트가 오프라인이 되어 중지된 HADB 노드를 수동으로 시작해야 하는 경우가 있습니다. 또한 이중 장애 이외의 다른 이유로 인해 재시작에 실패한 경우에도 수동으로 시작해야 하는 경우가 있습니다. 이중 장애로부터 복원하는 방법에 대한 자세한 내용은 [84 페이지 “데이터베이스 지우기”](#)를 참조하십시오.

대부분의 경우 먼저 **normal** 시작 수준을 사용하여 노드 시작을 시도해야 합니다. **normal** 시작 수준이 실패하거나 시간 초과된 경우 **repair** 시작 수준을 사용해야 합니다.

데이터베이스의 노드를 시작하려면 **hadbm startnode** 명령을 사용합니다. 구문은 다음과 같습니다.

```
hadbm startnode
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[--startlevel=level]
nodeno
[dbname]
```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 **hadb**입니다.

nodeno 피연산자는 시작할 노드 수를 지정합니다. 데이터베이스의 모든 노드 수를 표시하려면 **hadbm status**를 사용합니다.

자세한 내용은 **hadbm-startnode(1)**을 참조하십시오.

시작 수준 옵션

hadbm startnode 명령에는 노드를 시작할 수준을 지정하는 한 개의 특수 옵션 **--startlevel**(짧은 형식: **-l**)이 있습니다.

노드 시작 수준은 다음과 같습니다.

- **normal**(기본값): (메모리 및 디스크의 데이터 장치 파일의) 노드에서 로컬에 있는 데이터로 노드를 시작하고 누락된 최근 업데이트를 위해 미러와 동기화합니다.
- **repair**: 노드가 로컬 데이터를 무시하고 미러에서 복사합니다.
- **clear**: 노드에 대한 장치를 다시 초기화하고 미러 노드에서 데이터를 복구합니다. 장치 파일이 손상되었거나 장치 파일이 포함된 디스크를 교체한 경우, 장치 파일을 초기화해야 할 때 사용합니다.

다른 명령 옵션의 설명에 대해서는 [62 페이지 “일반 옵션”](#)을 참조하십시오.

예 3-6 노드 시작의 예

```
hadbm startnode 1
```

노드 중지

호스트 시스템의 하드웨어나 소프트웨어를 복구 또는 업그레이드하기 위해 노드를 중지해야 하는 경우가 있습니다. 노드를 중지하려면 `hadbm stopnode` 명령을 사용합니다. 명령 구문은 다음과 같습니다.

```
hadbm stopnode
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[--no-repair]
nodeno
[dbname]
```

`nodeno` 피연산자는 중지할 노드 수를 지정합니다. 이 노드 번호의 미리 노드는 실행 중이어야 합니다. 데이터베이스의 모든 노드 수를 표시하려면 `hadbm status`를 사용합니다.

`dbname` 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 `hadb`입니다.

`hadbm stopnode` 명령에는 중지된 노드를 대체할 예비 노드가 없음을 나타내는 한 개의 특수 옵션 `--no-repair`(짧은 형식: `-R`)가 있습니다. 이 옵션을 사용하지 않으면 예비 노드가 시작되어 중지된 노드의 기능을 대신 수행합니다.

다른 명령 옵션의 설명에 대해서는 [62 페이지 “일반 옵션”](#)을 참조하십시오. 자세한 내용은 `hadbm-stopnode(1)`을 참조하십시오.

예 3-7 노드 중지의 예

```
hadbm stopnode 1
```

노드 재시작

과도한 CPU 소비 등과 같은 비정상적인 동작이 발견된 경우 노드를 재시작해야 하는 경우가 있습니다.

데이터베이스의 노드를 재시작하려면 `hadbm restartnode` 명령을 사용합니다. 명령 구문은 다음과 같습니다.

```
hadbm restartnode
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[--startlevel=level]
nodeno
[dbname]
```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 *hadb*입니다.

nodeno 피연산자는 재시작할 노드 수를 지정합니다. 데이터베이스의 모든 노드 수를 표시하려면 *hadbm status*를 사용합니다.

hadbm restartnode 명령에는 노드를 시작할 수준을 지정하는 한 개의 특수 옵션 *--startlevel*(짧은 형식: *-l*)이 있습니다. 자세한 내용은 [80 페이지 “시작 수준 옵션”](#)을 참조하십시오.

다른 명령 옵션의 설명에 대해서는 [62 페이지 “일반 옵션”](#)을 참조하십시오. 자세한 내용은 *hadbm-restartnode(1)*을 참조하십시오.

예 3-8 노드 재시작의 예

```
hadbm restartnode 1
```

데이터베이스 관리

HADB 데이터베이스에 대해 다음 작업을 수행할 수 있습니다.

- [82 페이지 “데이터베이스 시작”](#)
- [83 페이지 “데이터베이스 중지”](#)
- [83 페이지 “데이터베이스 재시작”](#)
- [84 페이지 “데이터베이스 나열”](#)
- [84 페이지 “데이터베이스 지우기”](#)
- [85 페이지 “데이터베이스 제거”](#)

데이터베이스 시작

데이터베이스를 시작하려면 *hadbm start* 명령을 사용합니다. 이 명령은 데이터베이스가 중지되기 전에 실행 중이던 모든 노드를 시작합니다. 개별적으로 중지된(오프라인) 노드는 데이터베이스를 중지한 후 시작할 때 시작되지 않습니다.

명령 구문은 다음과 같습니다.

```
hadbm start
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maur!]
[dbname]
```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 *hadb*입니다.

명령 옵션에 대한 설명은 [62 페이지 “일반 옵션”](#)을 참조하십시오. 자세한 내용은 *hadbm-start(1)*을 참조하십시오.

예 3-9 데이터베이스 시작의 예

```
hadbm start
```

데이터베이스 중지

별도 작업으로 데이터베이스를 중지하고 시작하는 경우 데이터베이스가 중지된 동안에는 데이터를 사용할 수 없습니다. 데이터를 사용할 수 있도록 하려면 [83 페이지 “데이터베이스 재시작”](#)에 설명된 대로 데이터베이스를 재시작할 수 있습니다.

데이터베이스를 중지하여 다음을 수행합니다.

- 데이터베이스를 제거합니다.
- 모든 HADB 노드에 영향을 주는 시스템 유지 관리를 수행합니다.

데이터베이스를 중지하기 전에 데이터베이스를 사용하고 있는 개별 Application Server 인스턴스를 중지하거나 ha 이외의 지속성 유형을 사용하도록 구성합니다.

데이터베이스를 중지하면 데이터베이스에서 실행되고 있는 모든 노드가 중지되고 데이터베이스가 중지된 상태로 됩니다. 데이터베이스 상태에 대한 자세한 내용은 [93 페이지 “데이터베이스 상태”](#)를 참조하십시오.

데이터베이스를 중지하려면 `hadbm stop` 명령을 사용합니다. 명령 구문은 다음과 같습니다.

```
hadbm stop
[--adminpassword=password | --adminpasswordfile= file]
[--agent=maurl]
[dbname]
```

`dbname` 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 `hadb`입니다.

명령 옵션에 대한 설명은 [62 페이지 “일반 옵션”](#)을 참조하십시오. 자세한 내용은 `hadbm-stop(1)`을 참조하십시오.

예 3-10 데이터베이스 중지의 예

```
hadbm stop
```

데이터베이스 재시작

이상 동작을 발견한 경우(예: 일관된 시간 초과 문제) 데이터베이스를 재시작해야 하는 경우가 있습니다. 경우에 따라 재시작으로 문제가 해결될 수도 있습니다.

데이터베이스(y)를 재시작할 때 데이터베이스와 데이터는 계속 사용할 수 있습니다. 별도 작업으로 HADB를 중지하고 시작하는 경우 HADB가 중지된 동안에는 데이터와 데이터베이스 서비스를 사용할 수 없습니다. 기본적으로 `hadbm restart`는 노드의 롤링 재시작을 수행하기 때문입니다. 노드는 하나씩 중지하고 시작합니다. 이와 반대로 `hadbm stop`은 모든 노드를 동시에 중지합니다.

데이터베이스를 재시작하려면 `hadbm restart` 명령을 사용합니다. 명령 구문은 다음과 같습니다.

```

hadbm restart
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[--no-rolling]
[dbname]

```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 **hadb**입니다.

특수 옵션 **--no-rolling**(짧은 형식: **-g**)은 모든 노드를 한 번에 재시작하도록 지정하기 때문에 서비스 손실이 발생합니다. 이 옵션을 사용하지 않으면 이 명령은 데이터베이스의 각 노드를 현재 상태 또는 향상된 상태로 재시작합니다.

다른 명령 옵션의 설명에 대해서는 [62 페이지 “일반 옵션”](#)을 참조하십시오. 자세한 내용은 **hadbm-restart(1)**을 참조하십시오.

예를 들면 다음과 같습니다.

```
hadbm restart
```

데이터베이스 나열

HADB 인스턴스의 모든 데이터베이스를 나열하려면 **hadbm list** 명령을 사용합니다. 명령 구문은 다음과 같습니다.

```

hadbm list
[--agent=maurl]
[--adminpassword=password | --adminpasswordfile=file]

```

명령 옵션에 대한 설명은 [62 페이지 “일반 옵션”](#)을 참조하십시오. 자세한 내용은 **hadbm-list(1)**을 참조하십시오.

데이터베이스 지우기

다음과 같은 경우 데이터베이스를 지웁니다.

- **hadbm status** 명령이 데이터베이스가 작동 중이 아님을 나타냅니다. [92 페이지 “HADB 상태 가져오기”](#)를 참조하십시오.
- 여러 노드가 응답하지 않으며 오랜 시간 동안 대기 상태에 있습니다.
- 세션 데이터 손상에서 복원하고 있습니다. [86 페이지 “세션 데이터 손상에서 복원”](#)을 참조하십시오.

hadbm clear 명령은 데이터베이스 노드를 중지하고 데이터베이스 장치를 지운 후 노드를 시작합니다. 이 명령은 테이블, 사용자 이름 및 암호를 포함하여 HADB의 Application Server 스키마 데이터 저장소를 지웁니다. **hadbm clear**를 실행한 후 **asadmin configure-ha-cluster**를 사용하여 데이터 스키마를 다시 만들고 JDBC 연결 풀을 재구성한 다음, 세션 지속성 저장소를 다시 로드합니다.

명령 구문은 다음과 같습니다.

```

hadbm clear [--fast] [--spares=number]
[--dbpassword=password | --dbpasswordfile=file]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]

```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 *hadb*입니다.

다음 표에서는 특수 *hadbm clear* 명령 옵션에 대해 설명합니다. 다른 옵션에 대한 설명은 [62 페이지 “일반 옵션”](#)을 참조하십시오.

자세한 내용은 *hadbm-clear(1)*을 참조하십시오.

표 3-12 *hadbm clear* 옵션

옵션	설명	기본값
--fast	데이터베이스를 초기화하는 동안 장치 초기화를 건너뛰니다. 디스크 저장 장치가 손상된 경우에는 사용하지 마십시오.	적용할 수 없음
-F		
--spares= number	재초기화된 데이터베이스가 사용할 예비 노드 수. 이 숫자는 짝수여야 하며 데이터베이스에 있는 노드 수보다 작아야 합니다.	이전 예비 노드 수
-s		

예를 들면 다음과 같습니다.

```
hadbm clear --fast --spares=2
```

데이터베이스 제거

기존 데이터베이스를 제거하려면 *hadbm delete* 명령을 사용합니다. 이 명령은 데이터베이스의 구성 파일, 장치 파일 및 내역 파일을 삭제하고 공유 메모리 자원을 해제합니다. 제거할 데이터베이스는 반드시 존재해야 하고 중지되어야 합니다. [83 페이지 “데이터베이스 중지”](#)를 참조하십시오.

명령 구문은 다음과 같습니다.

```

hadbm delete
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]

```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 *hadb*입니다.

명령 옵션에 대한 설명은 [62 페이지 “일반 옵션”](#)을 참조하십시오. 자세한 내용은 *hadbm-delete(1)*을 참조하십시오.

예 3-11 데이터베이스 제거의 예

다음 명령은

```
hadbm delete
```

기본 데이터베이스 `hadb`를 삭제합니다.

세션 데이터 손상에서 복원

다음은 세션 데이터가 손상되었을 수 있음을 나타냅니다.

- 응용 프로그램이 세션 상태를 저장하려고 할 때마다 Application Server 시스템 로그(`server.log`)에 오류 메시지가 나타납니다.
- 서버 로그의 오류 메시지는 세션을 찾을 수 없거나 세션 활성화 도중 로드하지 못했음을 나타냅니다.
- 이전에 비활성화된 후에 활성화된 세션에 비어 있거나 잘못된 세션 데이터가 포함되어 있습니다.
- 인스턴스가 실패한 경우 페일오버된 세션에 비어 있거나 잘못된 세션 데이터가 포함되어 있습니다.
- 인스턴스가 실패한 경우 페일오버 세션을 로드하려는 인스턴스는 서버 로그에 세션을 찾을 수 없거나 로드하지 못했음을 나타내는 오류를 기록합니다.

▼ 세션 저장소를 일관된 상태로 되돌리는 방법

세션 저장소가 손상된 것으로 판단되면 다음 절차에 따라 세션 저장소를 일관된 상태로 되돌립니다.

1 세션 저장소를 지웁니다.

이 작업으로 문제가 해결되는지 확인합니다. 문제가 해결되면 중지합니다. 문제가 해결되지 않으면(예: 서버 로그에 오류가 계속 나타나는 경우) 계속합니다.

2 모든 노드에서 데이터 공간을 다시 초기화하고 데이터베이스에서 데이터를 지웁니다.

84 페이지 “데이터베이스 지우기”를 참조하십시오.

이 작업으로 문제가 해결되는지 확인합니다. 문제가 해결되면 중지합니다. 문제가 해결되지 않으면(예: 서버 로그에 오류가 계속 나타나는 경우) 계속합니다.

3 데이터베이스를 삭제한 다음 다시 만듭니다.

85 페이지 “데이터베이스 제거” 및 65 페이지 “데이터베이스 만들기”를 참조하십시오.

HADB 확장

원래 HADB 구성을 확장하는 이유에는 다음의 두 가지가 있습니다.

- 저장되어 있는 세션 데이터의 양이 데이터 장치의 기존 저장 공간을 초과합니다. 데이터 장치가 꽉 차서 트랜잭션이 중단되기 시작할 수 있습니다.
- 사용자 로드가 증가하여 시스템 자원이 많이 소모됩니다. 호스트를 더 추가해야 합니다.

이 절에서는 Application Server 클러스터 또는 데이터베이스를 종료하지 않고 HADB를 확장하는 방법에 대해 설명합니다.

- 87 페이지 “기존 노드에 저장 공간 추가”
- 88 페이지 “시스템 추가”
- 88 페이지 “노드 추가”
- 90 페이지 “데이터베이스 재조각화”
- 91 페이지 “데이터베이스를 다시 만들어 노드 추가”

또한 99 페이지 “HADB 시스템 유지 관리”의 관련 정보도 참조하십시오.

기존 노드에 저장 공간 추가

HADB 저장 공간 추가:

- 사용자 트랜잭션이 다음 오류 메시지 중 하나로 인해 반복적으로 중단되는 경우:
 - 4592: No free blocks on data devices
 - 4593: No unreserved blocks on data devices
- `hadbm deviceinfo` 명령이 일관되게 여유 공간이 부족함을 보고하는 경우. 95 페이지 “장치 정보 가져오기”를 참조하십시오.

노드에 사용되지 않은 디스크 공간이 있거나 디스크 용량을 추가하는 경우에도 기존 노드에 저장 공간을 추가하려 할 수 있습니다. 권장하는 데이터 장치 크기에 대한 자세한 내용은 68 페이지 “장치 크기 지정”을 참조하십시오.

저장 공간을 노드에 추가하려면 `hadbm set` 명령을 사용하여 데이터 장치 크기를 늘립니다.

명령 구문은 다음과 같습니다.

```
hadbm set DataDeviceSize=size
```

여기서 *size*는 데이터 장치 크기(MB)입니다.

명령 옵션에 대한 설명은 62 페이지 “일반 옵션”을 참조하십시오.

FaultTolerant 이상의 상태에 있는 데이터베이스에 대한 데이터 장치 크기를 변경하면 데이터 또는 가용성의 손실 없이 시스템이 업그레이드됩니다. 재구성하는 동안에도 데이터베이스는 작동 상태를 유지합니다. FaultTolerant 이상의 상태가 아닌 시스템에서 장치 크기를 변경하면 데이터가 손실됩니다. 데이터베이스 상태에 대한 자세한 내용은 93 페이지 “데이터베이스 상태”를 참조하십시오.

예 3-12 데이터 장치 크기 설정의 예

다음 명령은 데이터 장치 크기 설정의 예입니다.

```
hadbm set DataDeviceSize=1024
```

시스템 추가

HADB에 처리 또는 저장 용량이 더 필요한 경우 시스템을 추가할 수 있습니다. HADB를 실행할 새 시스템을 추가하려면 2 장에 설명된 대로 Application Server의 여부에 관계 없이 HADB 패키지를 설치합니다. 노드 토폴로지 대안에 대한 설명을 보려면 **Sun Java System Application Server Enterprise Edition 8.2 Deployment Planning Guide**의 3 장, “Selecting a Topology”를 참조하십시오.

▼ 기존 HADB 인스턴스에 새 시스템을 추가하는 방법

- 1 새 노드에서 관리 에이전트를 시작합니다.
- 2 관리 도메인을 새 호스트로 확장합니다.
자세한 내용은 `hadbm extenddomain` 명령을 참조하십시오.
- 3 이 호스트에서 새 노드를 시작합니다.
자세한 내용은 88 페이지 “노드 추가”를 참조하십시오.

노드 추가

HADB 시스템의 처리 및 저장 용량을 늘리려면 새 노드를 만들고 이 노드를 데이터베이스에 추가합니다.

노드를 추가하면 HADB JDBC 연결 풀의 다음 등록 정보를 업데이트합니다.

- `serverlist` 등록 정보
- 고정 풀 크기. 일반적으로 새 노드마다 8개의 연결을 더 추가합니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Deployment Planning Guide**의 “System Sizing”을 참조하십시오.

노드를 추가하려면 `hadbm addnodes` 명령을 사용합니다. 명령 구문은 다음과 같습니다.


```

hadbm addnodes [--no-refragment] [--spares=sparecount]
[--historypath=path]
[--devicepath=path]
[--set=attr-name-value-list]
[--dbpassword=password | --dbpasswordfile=file ]
[--adminpassword=password | --adminpasswordfile=file]
--hosts=hostlist [dbname]

```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 *hadb*입니다. 데이터베이스는 *HAFaultTolerant* 또는 *FaultTolerant* 상태여야 합니다. 데이터베이스 상태에 대한 자세한 내용은 [93 페이지 “데이터베이스 상태”](#)를 참조하십시오.

--devicepath 및 *--historypath* 옵션을 지정하지 않으면 새 노드는 기존 데이터베이스와 같은 장치 경로 및 같은 내역 파일을 사용합니다.

노드를 추가하면 기존 데이터의 재조각화 및 재배포가 수행되어 시스템에 새 노드가 포함됩니다. 온라인 재조각화를 수행하려면 재조각화가 끝날 때까지 *HADB* 노드에 대한 디스크에 기존 데이터와 새 데이터를 동시에 포함할 수 있는 충분한 공간이 있어야 합니다. 즉, 사용자 데이터 크기는 사용자 데이터에 사용 가능한 공간의 50%를 초과하지 않아야 합니다. 자세한 내용은 [95 페이지 “장치 정보 가져오기”](#)를 참조하십시오.

주 - 시스템 로드가 적을 때가 노드를 추가하기 가장 좋습니다.

예 3-13 노드 추가의 예

예를 들면 다음과 같습니다.

```
hadbm addnodes -adminpassword=password --hosts n6,n7,n8,n9
```

다음 표에서는 특수 *hadbm addnodes* 명령 옵션에 대해 설명합니다. 다른 옵션에 대한 설명은 [62 페이지 “일반 옵션”](#)을 참조하십시오.

표 3-13 *hadbm addnodes* 옵션

옵션	설명	기본값
<i>--no-refragment</i> <i>-r</i>	<p>노드를 만드는 동안에는 데이터베이스를 재조각화하지 마십시오. 이 경우 나중에 <i>hadbm refragment</i> 명령으로 새 노드를 사용할 수 있도록 데이터베이스를 재조각화합니다. 재조각화에 대한 자세한 내용은 90 페이지 “데이터베이스 재조각화”를 참조하십시오.</p> <p>재조각화를 위한 디스크 공간이 충분하지 않은 경우 데이터베이스를 더 많은 노드로 다시 만듭니다. 91 페이지 “데이터베이스를 다시 만들어 노드 추가”를 참조하십시오.</p>	적용할 수 없음

표 3-13 hadbm addnodes 옵션 (계속)

옵션	설명	기본값
--spares= <i>number</i> -s	이미 존재하는 예비 노드를 포함하여 새 예비 노드의 수인 숫자는 짝수여야 하며 추가된 노드 수보다 크지 않아야 합니다.	0
--devicepath= <i>path</i> -d	장치에 대한 경로. 장치는 다음과 같습니다. <ul style="list-style-type: none"> ■ DataDevice ■ NiLogDevice(노드 내부 로그 장치) ■ RelalgDevice(관계형 algebra 쿼리 장치) 이 경로는 반드시 존재해야 하며 쓰기 가능해야 합니다. 각 노드 또는 각 장치마다 이 경로를 서로 다르게 설정하려면 69 페이지 “이기종 장치 경로 설정”을 참조하십시오.	Solaris 및 Linux: <i>HADB_install_dir/device</i> Windows: C:\Sun\AppServer\SUNWhadb\vers. 여기서 <i>vers</i> 는 HADB 버전 번호입니다.
--hosts= <i>hostlist</i> -H	데이터베이스의 새 노드에 대한 새 호스트 이름의 쉼표로 구분된 목록. 목록에서 쉼표로 구분된 각 항목에 대해 한 개의 노드가 만들어집니다. 노드 수는 짝수여야 합니다. HADB 호스트의 IP 주소는 IPv4 주소여야 합니다. 중복되는 호스트 이름을 사용하여 동일한 시스템에 포트 번호가 다른 여러 노드를 만듭니다. 동일한 시스템의 노드가 미리 노드가 아닌지 확인합니다. 홀수 번호가 지정된 노드는 한 DRU에 있고, 짝수 번호가 지정된 노드는 다른 DRU에 있습니다. --spares를 사용하는 경우 새 예비 노드는 숫자가 가장 큼니다. 이중 네트워크 인터페이스로 데이터베이스를 만든 경우 새 노드를 동일한 방법으로 구성해야 합니다. 34 페이지 “네트워크 중복 구성”을 참조하십시오.	없음

데이터베이스 재조각화

데이터베이스를 재조각화하여 새로 만든 노드에 데이터를 저장합니다. 재조각화는 데이터베이스를 모든 활성 노드에 균등하게 분산합니다.

데이터베이스를 재조각화하려면 `hadbm refragment` 명령을 사용합니다. 명령 구문은 다음과 같습니다.

```
hadbm refragment [--dbpassword=password | --dbpasswordfile=file]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 `hadb`입니다. 데이터베이스는 `HAFaultTolerant` 또는 `FaultTolerant` 상태여야 합니다. 데이터베이스 상태에 대한 자세한 내용은 92 페이지 “HADB 상태 가져오기”를 참조하십시오.

명령 옵션에 대한 설명은 62 페이지 “일반 옵션”을 참조하십시오. 자세한 내용은 `hadbm-refragment(1)`을 참조하십시오.

온라인 재조각화를 수행하려면 재조각화가 끝날 때까지 HADB 노드에 대한 디스크에 기존 데이터와 새 데이터를 동시에 포함할 수 있는 충분한 공간이 있어야 합니다. 즉, 사용자 데이터 크기는 사용자 데이터에 사용 가능한 공간의 50%를 초과하지 않아야 합니다. 자세한 내용은 95 페이지 “장치 정보 가져오기”를 참조하십시오.

주 - 시스템 로드가 적을 때가 데이터베이스를 재조각화하기 가장 좋습니다.

여러 번 시도한 후에도 이 명령이 실패하는 경우 91 페이지 “데이터베이스를 다시 만들어 노드 추가”를 참조하십시오.

예 3-14 데이터베이스 재조각화의 예

예를 들면 다음과 같습니다.

```
hadbm refragment
```

데이터베이스를 다시 만들어 노드 추가

새 노드를 추가할 때 데이터 장치 공간이 부족하거나 기타 이유로 인해 온라인 재조각화가 영구적으로 실패하는 경우 데이터베이스를 새 노드로 다시 만듭니다. 이렇게 하면 기존 사용자 데이터 및 스키마 데이터가 손실됩니다.

▼ 데이터베이스를 다시 만들어 노드를 추가하는 방법

이 절차를 사용하면 프로세스 전체에 HADB 가용성을 유지할 수 있습니다.

1 각 Application Server 인스턴스에 대해 다음을 수행합니다.

a. 로드 밸런서에서 Application Server 인스턴스를 비활성화합니다.

b. 세션 지속성을 비활성화합니다.

c. Application Server 인스턴스를 다시 시작합니다.

d. 로드 밸런서에서 Application Server 인스턴스를 다시 활성화합니다.

가용성을 유지할 필요가 없는 경우 로드 밸런서에서 모든 서버 인스턴스를 한 번에 비활성화한 후에 다시 활성화할 수 있습니다. 이렇게 하면 시간이 절약되고 오래된 세션 데이터의 페일오버를 방지할 수 있습니다.

2 83 페이지 “데이터베이스 중지”에 설명된 대로 데이터베이스를 중지합니다.

- 3 85 페이지 “데이터베이스 제거”에 설명된 대로 데이터베이스를 삭제합니다.
 - 4 65 페이지 “데이터베이스 만들기”에 설명된 대로 데이터베이스에 추가 노드를 다시 만듭니다.
 - 5 75 페이지 “JDBC 연결 풀 구성”에 설명된 대로 JDBC 연결 풀을 재구성합니다.
 - 6 세션 지속성 저장소를 다시 로드합니다.
 - 7 각 Application Server 인스턴스에 대해 다음을 수행합니다.
 - a. 로드 밸런서에서 Application Server 인스턴스를 비활성화합니다.
 - b. 세션 지속성을 활성화합니다.
 - c. Application Server 인스턴스를 다시 시작합니다.
 - d. 로드 밸런서에서 Application Server 인스턴스를 다시 활성화합니다.
- 가용성을 유지할 필요가 없는 경우 로드 밸런서에서 모든 서버 인스턴스를 한 번에 비활성화한 후에 다시 활성화할 수 있습니다. 이렇게 하면 시간이 절약되고 오래된 세션 데이터의 페일오버를 방지할 수 있습니다.

HADB 모니터링

다음 방법으로 HADB의 활동을 모니터링할 수 있습니다.

- 92 페이지 “HADB 상태 가져오기”
- 95 페이지 “장치 정보 가져오기”
- 96 페이지 “런타임 자원 정보 가져오기”

이 절에서는 `hadbm status`, `hadbm deviceinfo` 및 `hadbm resourceinfo` 명령에 대해 간략하게 설명합니다. HADB 정보 해석에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Performance Tuning Guide**의 “Performance”를 참조하십시오.

HADB 상태 가져오기

데이터베이스나 노드의 상태를 표시하려면 `hadbm status` 명령을 사용합니다. 명령 구문은 다음과 같습니다.

```
hadbm status
[--nodes]
```

```
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

dbname 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 `hadb`입니다.

`--nodes` 옵션(짧은 형식 `-n`)은 데이터베이스의 각 노드에 대한 정보를 표시합니다. 자세한 내용은 [93 페이지 “노드 상태”](#)를 참조하십시오. 다른 명령 옵션의 설명에 대해서는 [62 페이지 “일반 옵션”](#)을 참조하십시오.

자세한 내용은 `hadbm-status(1)`을 참조하십시오.

예 3-15 HADB 상태 가져오기의 예

예를 들면 다음과 같습니다.

```
hadbm status --nodes
```

데이터베이스 상태

데이터베이스의 *state*는 현재 상태를 요약합니다. 다음 표에서는 가능한 데이터베이스 상태에 대해 설명합니다.

표 3-14 HADB 상태

데이터베이스 상태	설명
High-Availability Fault Tolerant(HAFaultTolerant)	데이터베이스에 내결함성이 있으며 각 DRU에 하나 이상의 예비 노드가 있습니다.
Fault Tolerant	미러된 모든 노드 쌍이 실행 중입니다.
Operational	미러된 각 노드 쌍에서 하나 이상의 노드가 실행 중입니다.
Non Operational	미러된 하나 이상의 노드 쌍에 두 노드가 모두 없습니다. 데이터베이스가 작동 중이 아닌 경우 84 페이지 “데이터베이스 지우기” 에 설명된 대로 데이터베이스를 지웁니다.
Stopped	데이터베이스에서 실행 중인 노드가 없습니다.
Unknown	데이터베이스의 상태를 확인할 수 없습니다.

노드 상태

`--nodes` 옵션을 사용하면 `hadbm status` 명령은 데이터베이스의 각 노드에 대해 다음 정보를 표시합니다.

- 노드 번호
- 노드가 실행 중인 시스템의 이름
- 노드의 포트 번호

- 노드의 역할. 역할과 의미에 대한 목록은 94 페이지 “노드의 역할”을 참조하십시오.
- 노드의 상태. 상태와 의미에 대한 목록은 94 페이지 “노드의 상태”를 참조하십시오.
- 해당 미러 노드 수

노드의 역할 및 상태는 이 절에 설명된 대로 변경할 수 있습니다.

- 94 페이지 “노드의 역할”
- 94 페이지 “노드의 상태”

노드의 역할

노드는 만드는 동안 역할이 할당되며 다음 역할 중 하나를 가질 수 있습니다.

- **Active:** 데이터를 저장하고 클라이언트 액세스를 허용합니다. 활성 노드는 미러된 쌍에 있습니다.
- **Spare:** 클라이언트 액세스는 허용하지만 데이터는 저장하지 않습니다. 데이터 장치를 초기화한 후 다른 노드를 모니터링하여 다른 노드를 사용할 수 없으면 복구를 시작합니다.
- **Offline:** 역할이 변경될 때까지 서비스를 제공하지 않습니다. 다시 온라인 상태가 되면 역할이 이전 역할로 변경될 수 있습니다.
- **Shutdown:** 활성과 오프라인의 중간 단계로, 예비 노드가 기능을 수행할 때까지 기다립니다. 예비 노드가 기능을 수행하게 되면 노드가 오프라인됩니다.

노드의 상태

노드는 다음 상태 중 하나일 수 있습니다.

- **Starting:** 노드가 시작 중입니다.
- **Waiting:** 노드의 시작 상태를 확인할 수 없으며 오프라인입니다. 단일 노드가 2분 이상 이 상태에 있으면 노드를 중지한 후에 **repair** 수준으로 시작합니다. 81 페이지 “노드 중지”, 80 페이지 “노드 시작” 및 84 페이지 “데이터베이스 지우기”를 참조하십시오.
- **Running:** 노드가 이 역할에 적합한 모든 서비스를 제공하고 있습니다.
- **Stopping:** 노드가 중지 프로세스에 있습니다.
- **Stopped:** 노드가 비활성 상태입니다. 중지된 노드는 복구할 수 없습니다.
- **Recovering:** 노드를 복원 중입니다. 노드가 실패하면, 실패한 노드의 기능을 미러 노드가 수행합니다. 실패한 노드는 주 기억 장치 또는 디스크의 데이터와 로그 레코드를 사용하여 복원을 시도합니다. 실패한 노드는 미러 노드의 로그 레코드를 사용하여 중지되었을 때 수행된 트랜잭션을 적용합니다. 복원에 성공하면 노드가 활성 상태가 됩니다. 복원에 실패하면 노드 상태가 복구 중으로 변경됩니다.
- **Repairing:** 노드를 복구 중입니다. 이 작업은 노드를 다시 초기화하고 미러 노드에서 데이터와 로그 레코드를 복사합니다. 복구는 복원보다 시간이 더 오래 걸립니다.

장치 정보 가져오기

HADB 데이터(디스크 저장소) 장치의 사용 가능한 공간 모니터:

- 일반적인 디스크 공간 사용 경향을 확인합니다.
- 예방 차원의 유지 관리 작업의 일부로: 사용자 로드가 증가하여 데이터베이스 구성의 크기를 조정하거나 확장하려 합니다.
- 데이터베이스 확장의 일부로: `hadbm addnodes`를 실행하여 시스템에 새 노드를 추가하기 전에 충분한 장치 공간이 있는지 확인합니다. 노드를 추가하려면 기존 노드에 약 40-50%의 사용 가능 공간이 있어야 합니다.
- 내역 파일 및 `server.log` 파일에 다음과 같은 오류 메시지가 있는 경우
 - No free blocks on data devices
 - No unreserved blocks on data devices

데이터 장치의 사용 가능 공간에 대한 정보를 가져오려면 `hadbm deviceinfo` 명령을 사용합니다. 이 명령은 데이터베이스의 각 노드에 대해 다음 정보를 표시합니다.

- 할당된 총 장치 크기(MB)(Totalsize)
- 사용 가능 공간(MB)(Freesize)
- 현재 사용 중인 장치의 백분율(Usage)

명령 구문은 다음과 같습니다.

```
hadbm deviceinfo [--details]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl] [dbname]
```

`dbname` 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 `hadb`입니다.

`--details` 옵션은 다음과 같은 추가 정보를 표시합니다.

- 장치에서 읽기 작업의 수
- 장치에서 쓰기 작업의 수
- 장치의 이름

다른 명령 옵션의 설명에 대해서는 [62 페이지 “일반 옵션”](#)을 참조하십시오.

자세한 내용은 `hadbm-deviceinfo(1)`을 참조하십시오.

사용자 데이터에 사용 가능한 공간을 확인하려면 총 장치 크기를 확인한 다음 `LogBufferSize`의 네 배에 장치 크기의 1%를 더한 HADB에 대해 예약된 공간을 뺍니다. 로그 버퍼의 크기를 모르는 경우 `hadbm get logbufferSize`를 사용합니다. 예를 들어, 총 장치 크기가 128MB이고 `LogBufferSize`가 24MB인 경우 사용자 데이터에 사용 가능한 공간은 $128 - (4 \times 24) = 32\text{MB}$ 입니다. 32MB에서 절반은 복제된 데이터에 사용되고 약 1퍼센트는 색인에 사용되며 25퍼센트만 실제 사용자 데이터에 사용할 수 있습니다.

사용자 데이터에 사용 가능한 공간은 총 크기와 예약된 크기가 다릅니다. 나중에 데이터를 재조각화하는 경우 여유 공간은 사용자 데이터에 사용 가능한 공간의 약 50%와 같아야 합니다. 재조각화가 관련되지 않으면 데이터 장치를 최대한 활용할 수 있습니다. 장치 공간이 부족한 상태에서 시스템이 실행 중인 경우 내역 파일에 자원 소모 경고가 기록됩니다.

HADB 조정에 대한 자세한 내용은 *Sun Java System Application Server Performance Tuning Guide*를 참조하십시오.

예 3-16 장치 정보 가져오기의 예

다음 명령은

```
hadbm deviceinfo --details
```

다음과 같은 예의 결과를 표시합니다.

NodeNO	Totalsize	Freesize	Usage	NReads	NWrites	DeviceName
0	128	120	6%	10000	5000	C:\Sun\SUNWhadb\hadb.data.0
1	128	124	3%	10000	5000	C:\Sun\SUNWhadb\hadb.data.1
2	128	126	2%	9500	4500	C:\Sun\SUNWhadb\hadb.data.2
3	128	126	2%	9500	4500	C:\Sun\SUNWhadb\hadb.data.3

런타임 자원 정보 가져오기

`hadbm resourceinfo` 명령은 HADB 런타임 자원 정보를 표시합니다. 이 정보를 사용하여 자원 경쟁을 식별하고 성능 병목 현상을 줄일 수 있습니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Performance Tuning Guide**의 “Tuning HADB”를 참조하십시오.

명령 구문은 다음과 같습니다.

```
hadbm resourceinfo [--databuf] [--locks] [--logbuf] [--nilogbuf]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

`dbname` 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 `hadb`입니다.

다음 표에서는 `hadbm resourceinfo` 특수 명령 옵션에 대해 설명합니다. 다른 명령 옵션의 설명에 대해서는 [62 페이지 “일반 옵션”](#)을 참조하십시오.

자세한 내용은 `hadbm-resourceinfo(1)`을 참조하십시오.

표 3-15 hadbm resourceinfo 명령 옵션

옵션	설명
--databuf	데이터 버퍼 풀 정보를 표시합니다.
-d	자세한 내용은 아래의 97 페이지 “데이터 버퍼 풀 정보”를 참조하십시오.
--locks	잠금 정보를 표시합니다.
-l	자세한 내용은 아래의 98 페이지 “잠금 정보”를 참조하십시오.
--logbuf	로그 버퍼 정보를 표시합니다.
-b	자세한 내용은 아래의 98 페이지 “로그 버퍼 정보”를 참조하십시오.
--nilogbuf	노드 내부 로그 버퍼 정보를 표시합니다.
-n	자세한 내용은 아래의 99 페이지 “노드 내부 로그 버퍼 정보”를 참조하십시오.

데이터 버퍼 풀 정보

데이터 버퍼 풀 정보에는 다음이 포함됩니다.

- **NodeNo:** 노드 번호
- **Avail:** 풀에서 사용 가능한 총 공간(MB)
- **Free:** 사용 가능 공간(MB)
- **Access:** 시작부터 현재까지 데이터베이스에서 데이터 버퍼로의 누적 액세스 수
- **Misses:** 데이터베이스 시작부터 현재까지 발생한 누적 페이지 오류 수
- **Copy-on-Write:** 검사점으로 인해 데이터 버퍼에서 내부적으로 복사된 누적 페이지 수

사용자 트랜잭션이 레코드에 대해 작업을 수행할 때 레코드가 포함된 페이지가 데이터 버퍼 풀에 있어야 합니다. 그렇지 않으면 miss 또는 페이지 오류가 발생합니다. 그런 다음 트랜잭션은 디스크의 데이터 장치 파일에서 페이지가 검색될 때까지 기다립니다.

실패율이 높으면 데이터 버퍼 풀을 늘리십시오. 실패는 누적되므로 **hadbm resourceinfo**를 주기적으로 실행하고 두 번 실행 간의 차이를 사용하여 실패율을 확인하십시오. 사용 가능 공간이 매우 작은 경우에는 검사점 메커니즘이 사용 가능한 새 블록을 만들기 때문에 걱정하지 않아도 됩니다.

예 3-17 데이터 버퍼 풀 정보의 예

예를 들면 다음과 같습니다.

```
NodeNO Avail Free Access Misses Copy-on-Write
0 256 128 100000 50000 10001 256 128 110000 45000 950
```

잠금 정보

잠금 정보는 다음과 같습니다.

- NodeNo: 노드 번호
- Avail: 노드에서 사용 가능한 총 잠금 수
- Free: 사용 가능 잠금의 수
- Waits: 잠금을 얻기 위해 대기 중인 트랜잭션의 수. 이 수는 누적됩니다.

단일 트랜잭션은 노드에서 사용 가능한 잠금의 25%까지 사용할 수 있습니다. 따라서 대규모로 작업을 수행하는 트랜잭션은 이 제한 내용을 알고 있어야 합니다. 이러한 트랜잭션은 일괄 작업으로 수행하는 것이 좋습니다. 여기서 각 일괄 작업은 별도의 트랜잭션, 즉 각 일괄 작업 커밋으로 취급되어야 합니다. `repeatable read` 격리 수준으로 실행 중인 읽기 작업과 `delete`, `insert` 및 `update` 작업은 트랜잭션이 종료된 후에만 해제되는 잠금을 사용하기 때문입니다.

NumberOfLocks를 변경하려면 101 페이지 “내역 파일 지우기 및 아카이브”를 참조하십시오.

예 3-18 잠금 정보의 예

예를 들면 다음과 같습니다.

```
NodeNO Avail Free Waits
0 50000 20000 101 50000 20000 0
```

로그 버퍼 정보

로그 버퍼 정보는 다음과 같습니다.

- NodeNo: 노드 번호
- Available: 로그 버퍼에 할당된 메모리 양(MB)
- Free: 사용 가능 메모리의 양(MB)

사용 가능 공간이 매우 작은 경우에는 HADB가 로그 버퍼의 압축을 시작하기 때문에 걱정하지 않아도 됩니다. HADB는 로그 버퍼의 처음부터 압축을 시작하고 연속되는 로그 레코드에 대해 수행합니다. HADB가 노드에서 실행된 적이 없고 미리 노드에서 수신되지 않은 로그 레코드를 발견할 때까지 압축이 계속됩니다.

예 3-19 로그 버퍼 정보의 예

예를 들면 다음과 같습니다.

```
NodeNO Avail Free
0 16 21 16 3
```

노드 내부 로그 버퍼 정보

노드 내부 로그 버퍼 정보는 다음과 같습니다.

- 노드 번호
- Available: 로그 장치에 할당된 메모리 양(MB)
- Free: 사용 가능 메모리의 양(MB)

예 3-20 내부 로그 버퍼 정보의 예

예를 들면 다음과 같습니다.

NodeNO Avail Free

0 16 21 16 3

HADB 시스템 유지 관리

HADB는 미리 노드에 데이터를 복제하여 내결함성을 아카이브합니다. 프로덕션 환경의 경우, **Sun Java System Application Server Enterprise Edition 8.2 Deployment Planning Guide**에 설명된 대로 미리 노드는 미리하는 노드와 별개의 DRU에 있습니다.

실패는 하드웨어 실패, 전원 실패 또는 운영 체제 재부트 등과 같은 예기치 않은 이벤트입니다. HADB는 한 노드, 한 시스템(미러 노드 쌍 없음), 동일한 DRU에 속한 하나 이상의 시스템 또는 하나의 전체 DRU의 단일 실패에 대해 내결함성을 가집니다. 하지만 HADB는 하나 이상의 미리 노드 쌍이 동시에 실패하는 이중 실패에서 자동으로 복원되지 **않습니다**. 이중 실패가 발생하면 HADB를 지우고 세션 저장소를 다시 만들어야 합니다. 이 경우 모든 데이터가 지워집니다.

단일 시스템에서 작업해야 하는지 또는 여러 시스템에서 작업해야 하는지 여부에 따라 유지 관리 절차가 다릅니다.

▼ 단일 시스템에서 유지 관리를 수행하는 방법

이 절차는 계획 및 계획되지 않은 유지 관리 모두에 적용되며 HADB 가용성이 인터럽트되지 않습니다.

1 유지 관리 절차를 수행하고 시스템을 실행합니다.

2 ma가 실행되고 있는지 확인합니다.

ma는 Windows 서비스로 실행되거나 init.d 스크립트(배포용으로 권장됨)에서 실행되는 경우 운영 체제에서 시작되어야 합니다. 시작되지 않으면 수동으로 시작합니다.

51 페이지 “관리 에이전트 시작”을 참조하십시오.

3 시스템의 모든 노드를 시작합니다.

자세한 내용은 80 페이지 “노드 시작”을 참조하십시오.

4 노드가 활성화되어 실행 중인지 확인합니다.

자세한 내용은 92 페이지 “HADB 상태 가져오기”를 참조하십시오.

▼ 모든 HADB 시스템에 계획된 유지 관리를 수행하는 방법

계획된 유지 관리에는 하드웨어 및 소프트웨어 업그레이드 등의 작업이 포함됩니다. 이 절차는 HADB 가용성을 인터럽트하지 않습니다.

- 1 첫 번째 DRU의 각 예비 시스템에 대해 99 페이지 “단일 시스템에서 유지 관리를 수행하는 방법”에 설명된 대로 단일 시스템 절차를 하나씩 반복합니다.
- 2 첫 번째 DRU의 각 활성 시스템에 대해 99 페이지 “단일 시스템에서 유지 관리를 수행하는 방법”에 설명된 대로 단일 시스템 절차를 하나씩 반복합니다.
- 3 두 번째 DRU에 대해 단계 1과 2를 반복합니다.

▼ 모든 HADB 시스템에 계획된 유지 관리를 수행하는 방법

이 절차는 HADB가 단일 또는 여러 시스템에 있을 때 사용할 수 있습니다. 유지 관리 절차 도중 HADB 서비스가 인터럽트됩니다.

- 1 HADB를 중지합니다. 83 페이지 “데이터베이스 중지”를 참조하십시오.
- 2 유지 관리 절차를 수행하고 모든 시스템을 실행합니다.
- 3 ma가 실행되고 있는지 확인합니다.
- 4 HADB를 시작합니다.
자세한 내용은 82 페이지 “데이터베이스 시작”을 참조하십시오.
마지막 단계를 완료하면 HADB 데이터를 다시 사용할 수 있게 됩니다.

▼ 실패가 발생한 경우 계획되지 않은 유지 관리를 수행하는 방법

- 데이터베이스 상태를 확인합니다.

92 페이지 “HADB 상태 가져오기”를 참조하십시오.

- 데이터베이스 상태가 Operational 이상인 경우:

계획되지 않은 유지 관리가 필요한 시스템에 미리 노드가 포함되어 있지 않습니다. 실패한 각 시스템에 대해 한 번에 한 DRU씩 단일 시스템 절차를 따릅니다. HADB 서비스는 인터럽트되지 않습니다.

- 데이터베이스 상태가 Non-Operational인 경우:

계획되지 않은 유지 관리가 필요한 시스템에 미리 노드가 포함됩니다. 전체 HADB가 실패한 단일 시스템에 있는 경우입니다. 먼저 모든 시스템을 실행합니다. 그런 다음 HADB를 지우고 세션 저장소를 다시 만듭니다. 84 페이지 “데이터베이스 지우기”를 참조하십시오. 이 경우 HADB 서비스가 인터럽트됩니다.

내역 파일 지우기 및 아카이브

HADB 내역 파일은 모든 데이터베이스 작업과 오류 메시지를 기록합니다. HADB는 기존 내역 파일의 끝에 추가하므로 시간이 지날수록 파일 크기가 커집니다. 디스크 공간을 절약하고 파일이 너무 커지지 않도록 하려면 내역 파일을 주기적으로 지우고 아카이브합니다.

데이터베이스의 내역 파일을 지우려면 `hadbm clearhistory` 명령을 사용합니다.

명령 구문은 다음과 같습니다.

```
hadbm clearhistory
[--saveto=path]
[dbname]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

`dbname` 피연산자는 데이터베이스 이름을 지정합니다. 기본값은 `hadb`입니다.

기존 내역 파일을 저장할 디렉토리를 지정하려면 `--saveto` 옵션(짧은 형식: `-o`)을 사용합니다. 이 디렉토리에는 적절한 쓰기 권한이 있어야 합니다. 다른 명령 옵션의 설명에 대해서는 62 페이지 “일반 옵션”을 참조하십시오.

자세한 내용은 `hadbm-clearhistory(1)`을 참조하십시오.

`hadbm create` 명령의 `--historypath` 옵션은 내역 파일의 위치를 결정합니다. 내역 파일의 이름 형식은 `dbname.out.nodeno`입니다. `hadbm create`에 대한 자세한 내용은 [65 페이지 “데이터베이스 만들기”](#)를 참조하십시오.

내역 파일 형식

내역 파일의 각 메시지는 다음 정보가 포함되어 있습니다.

- 메시지를 생성한 HADB 프로세스의 약어 이름
- 메시지의 유형:
 - INF - 일반 정보
 - WRN - 경고
 - ERR - 오류
 - DBG - 디버그 정보
- 타임스탬프. 시간은 호스트 시스템 클럭에서 가져옵니다.
- 노드가 중지되거나 시작될 때 시스템에서 발생하는 서비스 세트 변경 내용

자원 부족에 대한 메시지에는 “HIGH LOAD” 문자열이 포함되므로

내역 파일의 모든 항목에 대해 자세히 알 필요는 없습니다. 특정한 이유로 인해 내역 파일을 좀더 자세히 알아야 할 경우에는 Sun 고객 지원에 문의하십시오.

로드 균형 조정을 사용하도록 웹 서버 구성

로드 밸런서 플러그인 설치 프로그램은 웹 서버의 구성 파일을 일부 수정합니다. 변경 사항은 웹 서버에 따라 다릅니다. 또한 일부 웹 서버의 경우 로드 밸런서가 제대로 작동하려면 수동으로 구성해야 합니다.

주 - 로드 밸런서 플러그인은 Sun Java System Application Server Enterprise Edition과 함께 설치할 수도 있고 지원되는 웹 서버를 실행하는 시스템에 따로 설치할 수도 있습니다. 설치 절차에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Installation Guide**의 1 장, “Installing Application Server Software” 또는 **Sun Java Enterprise System 5 Installation Guide for UNIX**(Java Enterprise System을 사용할 경우)를 참조하십시오.

- 103 페이지 “Sun Java System Web Server 사용”
- 104 페이지 “Apache Web Server 사용”
- 113 페이지 “Microsoft IIS 사용”

Sun Java System Web Server 사용

Sun Java System Web Server의 경우 로드 밸런서 플러그인을 설치하면 설치 프로그램은 필요한 모든 구성을 자동으로 수행합니다. 수동 구성이 필요하지 않습니다.

설치 프로그램은 Sun Java System Web Server의 구성 파일에 다음 항목을 추가합니다.

웹 서버 인스턴스의 `magnus.conf` 파일에 다음이 추가됩니다.

```
##EE lb-pluginInit
fn="load-modules"
shlib="web-server-install-dir/plugins/lbplugin/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrough" Thread="no"
Init fn="init-passthrough"
##end addition for EE lb-plugin
```

웹 서버 인스턴스의 `obj.conf` 파일에 다음이 추가됩니다.

```
<Object name=default>
NameTrans fn="name-trans-passthrough" name="lbplugin"
config-file="web-server-install-dir/web-server-instance/config/loadbalancer.xml"
<Object name="lbplugin">
  ObjectType fn="force-type" type="magnus-internal/lbplugin"
  PathCheck fn="deny-existence" path="*/WEB-INF/*"
  Service type="magnus-internal/lbplugin"
  fn="service-passthrough"
  Error reason="Bad Gateway"
  fn="send-error"
  uri="$docroot/badgateway.html"
</object>
```

위의 코드에서 `lbplugin`은 `Object`를 고유하게 식별하는 이름이고 `web-server-install-dir/web-server-instance/config/loadbalancer.xml`은 로드 밸런서가 실행되도록 구성된 가상 서버에 대한 XML 구성 파일의 위치입니다.

`NameTrans` 항목이 `obj.conf`에 나타나는 순서는 매우 중요합니다. 설치 프로그램은 `NameTrans` 항목을 올바른 위치에 놓지만 다른 목적으로 `obj.conf`를 편집하는 경우 순서가 올바른 상태로 유지되는지 확인해야 합니다. 특히 로드 밸런서 정보가 `document-root` 함수 앞에 와야 합니다. `obj.conf` 파일에 대한 자세한 내용은 **Sun Java System Web Server 7.0 Administrator's Configuration File Reference**를 참조하십시오.

설치 후에 119 페이지 “[HTTP 로드 균형 조정 설정](#)”에 설명된 대로 로드 밸런서를 구성합니다.

Apache Web Server 사용

Apache Web Server를 사용하려면 로드 밸런서 플러그인을 설치하기 전에 특정 구성 단계를 수행해야 합니다. 로드 밸런서 플러그인을 설치하면 Apache Web Server가 다음과 같이 추가로 수정됩니다. 플러그인 설치 후에 추가 구성 단계를 수행해야 합니다.

주 - Apache 1.3의 경우 여러 Apache 하위 프로세스를 실행할 경우 프로세스마다 고유한 로드 균형 조정 라운드 로빈 시퀀스가 있습니다. 예를 들어, 두 개의 Apache 하위 프로세스가 실행되는 중이고 로드 밸런서 플러그인이 두 개의 Application Server 인스턴스에 대해 로드 균형 조정을 할 경우 첫 번째 요청을 인스턴스 #1로 보내고 두 번째 요청도 인스턴스 #1로 보냅니다. 세 번째 요청을 인스턴스 #2로 보내고 네 번째 요청도 다시 인스턴스 #2로 보냅니다. 이 패턴이 반복됩니다(instance1, instance1, instance2, instance2 등). 이 동작은 예상되는 instance1, instance2, instance1, instance2 등의 패턴과는 다릅니다. Sun Java System Application Server에서 Apache용 로드 밸런서 플러그인은 각 Apache 프로세스에 대해 로드 밸런서 인스턴스를 인스턴스화하여 독립적인 로드 균형 조정 시퀀스를 만듭니다.

--with-mpm=worker 옵션을 사용하여 컴파일할 경우 Apache 2에는 멀티스레드된 동작이 발생합니다.

- 105 페이지 “Apache Web Server 사용 요구 사항”
- 107 페이지 “로드 밸런서 플러그인을 설치하기 전 Apache 구성”
- 110 페이지 “로드 밸런서 플러그인 설치 프로그램에서 수정한 사항”
- 111 페이지 “로드 밸런서 플러그인을 설치한 후 Apache 구성”
- 113 페이지 “Solaris 및 Linux에서 Apache 시작”

Apache Web Server 사용 요구 사항

Apache Web Server의 경우 Apache 버전에 따라 설치된 프로그램은 다음의 최소 요구 사항을 만족해야 합니다.

Apache 1.3의 최소 요구 사항

Apache 1.3을 사용할 경우 로드 밸런서 플러그인에는 다음이 필요합니다.

- openssl-0.9.8b(소스)
- mod_ssl-2.8.n-1.3.x(소스). 여기서 *n*은 사용자의 Apache 버전에 대한 올바른 버전의 mod_ssl을 나타내고 *x*는 Apache 버전을 나타냅니다.
- gcc-3.3-sol9-sparc-local 패키지(Solaris 9 SPARC용)
- gcc-3.3-sol9-intel-local 패키지(Solaris 9 x86용)
- 사전 설치된 gcc(Solaris 10용)
- flex-2.5.4a-sol9-sparc-local 패키지(Solaris 9 SPARC용)
- flex-2.5.4a-sol9-intel-local 패키지(Solaris 9 x86용)
- 사전 설치된 flex(Solaris 10용)

소프트웨어 소스는 <http://www.sunfreeware.com>에서 구할 수 있습니다.

사용자의 Apache 버전과 함께 사용할 올바른 버전의 mod_ssl을 비롯한 mod_ssl에 대한 자세한 내용은 <http://www.modssl.org>를 참조하십시오.

Apache를 컴파일하기 전에 다음을 수행합니다.

- Linux 2.1 플랫폼에서 동일한 시스템에 Sun Java System Application Server를 설치합니다.
- Solaris 9 운영 체제에서 pkgadd를 사용하여 gcc 및 flex를 설치합니다. pkgadd에는 루트 액세스 권한이 필요합니다.
- Solaris 9 운영 체제에서 gcc 버전 3.3 및 make가 PATH에 있고 flex가 설치되어 있는지 확인합니다.
- Solaris 10 운영 체제의 Java Enterprise System 설치에서 OpenSSL용 make를 실행하기 전에 Solaris SPARC의 경우 `/usr/local/lib/gcc-lib/sparc-sun-solaris2.9/3.3/install-tools`, Solaris x86의 경우 `/usr/local/lib/gcc-lib/i386-pc-solaris2.9/3.3/install-tools`에 있는 `mkheaders`를 실행합니다.
- Red Hat Enterprise Linux Advanced Server 2.1에서 gcc를 사용할 경우 gcc 3.0 이후 버전이어야 합니다.

주 - gcc 이외의 다른 C 컴파일러를 사용하려면 C 컴파일러의 경로를 설정하고 PATH 환경 변수에서 유틸리티를 작성합니다. 예를 들어, sh 셸을 사용할 경우 `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:application-server-install-dir /lib`를 사용합니다.

Apache 2의 최소 요구 사항

Apache 2를 사용할 경우 로드 밸런서 플러그인에는 다음이 필요합니다.

- openssl-0.9.8b(소스)
- httpd-2.0.49(소스)
- gcc-3.3-sol9-sparc-local 패키지(Solaris 9 SPARC용)
- gcc-3.3-sol9-intel-local 패키지(Solaris 9 x86용)
- 사전 설치된 gcc(Solaris 10용)
- flex-2.5.4a-sol9-sparc-local 패키지(Solaris 9 SPARC용)
- flex-2.5.4a-sol9-intel-local 패키지(Solaris 9 x86용)
- 사전 설치된 flex(Solaris 10용)

소프트웨어 소스는 <http://www.sunfreeware.com>에서 구할 수 있습니다.

Apache를 컴파일하기 전에 다음을 수행합니다.

- Linux 플랫폼에서 동일한 시스템에 Sun Java System Application Server를 설치합니다.
- Solaris 9 운영 체제에서 pkgadd를 사용하여 gcc 및 flex를 설치합니다. pkgadd에는 루트 액세스 권한이 필요합니다.
- Solaris 9 운영 체제에서 gcc 버전 3.3 및 make가 PATH에 있고 flex가 설치되어 있는지 확인합니다.

- Solaris 10 운영 체제에서 OpenSSL용 make를 실행하기 전에 Solaris SPARC의 경우 `/usr/local/lib/gcc-lib/sparc-sun-solaris2.9/3.3/install-tools`, Solaris x86의 경우 `/usr/local/lib/gcc-lib/i386-pc-solaris2.9/3.3/install-tools`에 있는 `mkheaders`를 실행합니다.
- Red Hat Enterprise Linux Advanced Server 2.1에서 gcc를 사용할 경우 gcc 3.0 이후 버전이어야 합니다.

주 - gcc 이외의 다른 C 컴파일러를 사용하려면 C 컴파일러의 경로를 설정하고 PATH 환경 변수에서 유틸리티를 작성합니다.

로드 밸런서 플러그인을 설치하기 전 Apache 구성

Apache용 로드 밸런서 플러그인을 설치하기 전에 Apache Web Server를 설치합니다. Apache 소스를 컴파일하고 SSL과 함께 실행되도록 빌드해야 합니다. 이 절에서는 로드 밸런서 플러그인을 실행하도록 Apache Web Server를 성공적으로 컴파일하는 데 필요한 최소 요구 사항 및 고급 단계를 설명합니다. 이러한 요구 사항 및 단계는 Solaris 및 Linux 버전의 소프트웨어에만 적용됩니다. Windows 버전의 Apache에 대한 자세한 내용은 Apache 웹 사이트를 참조하십시오.

주 - 여기에 포함된 지침은 <http://httpd.apache.org/docs>의 지침을 토대로 작성되었습니다. SSL 인식 Apache를 설치하는 방법에 대한 자세한 지침은 이 웹 사이트를 참조하십시오.

▼ SSL 인식 Apache 설치

시작하기 전에 Apache 소프트웨어를 미리 다운로드하여 압축을 풀어야 합니다.

- 1 <http://openssl.org>에서 제공되는 OpenSSL 소스를 다운로드하여 압축을 풉니다.

- 2 OpenSSL을 컴파일하고 빌드합니다.

OpenSSL 0.9.7.e가 설치된 경우 Linux 3.0 플랫폼에서는 이 단계가 필요하지 않습니다. Linux 4.0에서는 이 단계가 필요합니다.

전체 설치 지침은 OpenSSL의 압축을 푼 디렉토리에서 `INSTALL`이라는 파일을 참조하십시오. 이 파일에는 사용자가 지정한 위치에 OpenSSL을 설치하는 방법에 대한 정보가 있습니다.

OpenSSL에 대한 자세한 내용은 <http://www.openssl.org/>를 참조하십시오.

- 3 Apache를 다운로드하여 압축을 풉니다.

Apache는 <http://httpd.apache.org>에서 제공됩니다.

4 Apache를 컴파일하고 빌드합니다.

Apache 버전에 따라 다음 절차 중 하나를 수행하십시오.

- Apache 1.3에서 다음 단계를 수행하여 `mod_ssl`을 사용하여 Apache를 구성합니다.

a. `mod_ssl` 소스의 압축을 풉니다.

b. 다음을 입력합니다.

```
cd mod_ssl-2.8.n-1.3.x
```

c. 다음을 입력합니다.

```
./configure --with-apache=../apache_1.3.x --with-ssl=../openssl-0.9.8b
--prefix=Apache-install-path --enable-module=ssl --enable-shared=ssl
--enable-rule=SHARED_CORE --enable-module=so
```

위의 명령에서 *n*은 사용자의 Apache 버전과 함께 사용할 올바른 버전의 `mod_ssl`이고 *x*는 Apache 버전 번호이고 *Apache-install-path*는 Apache를 설치할 디렉토리입니다.

사용자의 Apache 버전과 함께 사용할 올바른 버전의 `mod_ssl`을 비롯한 `mod_ssl`에 대한 자세한 내용은 <http://www.modssl.org>를 참조하십시오.

- Apache 2의 경우 다음과 같이 소스 트리를 구성합니다.

a. `cd http-2.0.x`.

b. 다음 명령을 실행합니다.

```
./configure --with-ssl=OpenSSL-install-path --prefix=Apache-install-path
--enable-ssl --enable-so
```

위의 명령에서 *x*는 Apache 버전 번호이고 *open-ssl-install-path*는 OpenSSL이 설치된 디렉토리의 절대 경로이고 *Apache-install-path*는 Apache를 설치할 디렉토리입니다.

Apache 2 서버에서 HTTPS 요청을 수락하는 경우에는 `--enable-ssl --enable-so` 옵션만 사용하면 됩니다.

5 Apache 2의 경우 Apache의 `ssl.conf` 및 `httpd.conf` 파일에 사용자의 환경에 맞는 올바른 값이 포함되어 있는지 확인합니다.

- `ssl.conf`의 `VirtualHost default:port`에서 기본 호스트 이름과 포트를 Apache 2가 설치되는 로컬 시스템의 호스트 이름 및 서버의 포트 번호로 바꿉니다.
이 변경을 수행하지 않으면 로드 밸런서가 작동하지 않습니다. Solaris의 경우 Apache가 시작되지 않거나 Linux의 경우 HTTPS 요청이 작동하지 않을 수 있습니다.
- `ssl.conf`의 `ServerName www.example.com:443`에서 `www.example.com`을 Apache 2가 설치되는 로컬 시스템의 호스트 이름으로 바꿉니다.

이 변경을 수행하지 않으면 보안 인증서가 설치된 경우 Apache를 시작할 때 다음 경고가 나타납니다.

```
[warn] RSA server certificate CommonName (CN) 'hostname' does NOT match server name!
```

Apache 2용 인증서를 설치하는 방법에 대한 자세한 내용은 [111 페이지 “Apache 2용 보안 인증서 만들기”](#)를 참조하십시오.

- httpd.conf의 ServerName `www.example.com:80`에서 `www.example.com`을 Apache 2가 설치되는 로컬 시스템의 호스트 이름으로 바꿉니다.
이 변경을 수행하지 않으면 시스템에서 서버의 정규화된 도메인을 이름을 확인하지 못하고 중복된 VirtualHost 항목이 있을 경우 Apache를 시작하면 경고가 표시됩니다.

Apache를 루트 사용자로 설치한 경우 `apache-install-location/conf/httpd.conf`에서 사용자 및 그룹을 구성하는 방법에 대한 정보를 숙지해야 합니다. Apache는 httpd.conf에 설명된 사용자로 실행됩니다. Apache가 시작될 때 로드 밸런서 플러그인이 초기화되도록 하려면 `loadbalancer.xml` 파일 및 `sun-loadbalancer_1.1.dtd` 파일(`apache-install-location/conf`에 있음)에 이 사용자와 일치하는 파일 사용 권한이 있어야 합니다.

6 Linux 2.1의 Apache의 경우 컴파일 전에 다음을 수행합니다.

- src/MakeFile을 열고 자동으로 생성된 섹션 맨 끝을 찾습니다.
- 자동으로 생성된 섹션 다음에 나오는 처음 네 줄 뒤에 다음 줄을 추가합니다.

```
LIBS+= -licuuc -licui18n -lnspr4 -lpthread -lxerces-c
-lsupport -lnsprwrap -lns-httpd40
LDFLAGS+= -L/application-server-install-dir/lib -L/opt/sun/private/lib

-L/opt/sun/private/lib는 Application Server를 Java Enterprise System 설치의 일부로
설치한 경우에만 필요합니다.
```

예를 들면 다음과 같습니다.

```
##(자동으로 생성된 섹션의 끝)
##
CFLAGS=$(OPTIM) $(CFLAGS1) $(EXTRA_CFLAGS)
LIBS=$(EXTRA_LIBS) $(LIBS1)
INCLUDES=$(INCLUDES1) $(INCLUDES0) $(EXTRA_INCLUDES)
LDFLAGS=$(LDFLAGS1) $(EXTRA_LDFLAGS)
"LIBS+= -licuuc -licui18n -lnspr4 -lpthread
-lxerces-c -lsupport -lnsprwrap -lns-httpd40
LDFLAGS+= -L/application-server-install-dir /lib -L/opt/sun/private/lib
```

- 환경 변수 `LD_LIBRARY_PATH`를 설정합니다.

독립 실행형 설치에서는 이 환경 변수를 Application Server: `install-dir/lib`로 설정합니다.

Java Enterprise System 설치에서는 이 환경 변수를 Application Server:
`install-dir/lib:opt/sun/private/lib`로 설정합니다.

Solaris 9를 사용하는 경우에는 LD_LIBRARY_PATH에 /usr/local/lib를 추가합니다.

7 사용 중인 버전 에 대한 설치 지침에 설명된 대로 Apache를 컴파일합니다.

자세한 내용은 <http://httpd.apache.org/>를 참조하십시오.

일반적으로 단계는 다음과 같습니다.

a. make

b. make certificate(Apache 1.3에만 해당)

c. make install

make certificate 명령은 비밀번호를 요청합니다. Apache의 안전한 시작을 위해 이 비밀번호가 필요하므로 기억하는 것이 좋습니다.

로드 밸런서 플러그인 설치 프로그램에서 수정한 사항

로드 밸런서 플러그인 설치 프로그램은 필요한 파일을 웹 서버의 루트 디렉토리에 있는 디렉토리에 추출합니다.

- Apache 1.3에서 이 디렉토리는 libexec입니다.
- Apache 2에서 이 디렉토리는 modules입니다.

웹 서버 인스턴스의 httpd.conf 파일에 다음 항목을 추가합니다.

```
<VirtualHost machine-name:443>
##Addition for EE lb-plugin
LoadFile /usr/lib/libCstd.so.1
LoadModule apachelbplugin_module libexec/mod_loadbalancer.so
#AddModule mod_apachelbplugin.cpp
<IfModule mod_apachelbplugin.cpp>
    config-file webserver-instance/conf/loadbalancer.xml
    locale en
</IfModule><VirtualHost machine-ip-address>
    DocumentRoot "webserver-instance/htdocs"
    ServerName server-name
</VirtualHost>
##END EE LB Plugin ParametersVersion 7
```

Apache 2의 경우 모듈은 mod_apache2lbplugin.cpp입니다.

로드 밸런서 플러그인을 설치한 후 Apache 구성

Apache Web Server에서는 로드 밸런서 플러그인이 작동하려면 올바른 보안 파일이 있어야 합니다. 로드 밸런서는 이러한 보안 데이터베이스 파일이 필요한 NSS(Network Security Service) 라이브러리에 의존합니다. 웹 서버에서 액세스할 수 있는 위치에서 Application Server 설치를 사용할 수 있도록 Application Server에서 이러한 보안 데이터베이스 파일을 가져와야 합니다.

▼ 로드 밸런서에서 작동하도록 Apache 보안 파일 구성

- 1 *Apache-install-dir* 아래에 *sec_db_files*라는 디렉토리를 만듭니다.
- 2 Application Server의 보안 데이터베이스 파일을 만든 디렉토리에 복사합니다.
domain-dir/config/.db*를 *Apache-install-dir/sec_db_files*에 복사합니다.
- 3 플랫폼에 따라 추가 구성을 수행합니다.
 - Solaris 플랫폼의 Java Enterprise System 설치인 경우:
/usr/lib/mps/secv1 경로를 *Apache-install-dir/bin/apachectl* 스크립트의 LD_LIBRARY_PATH에 추가합니다. 이 경로는 /usr/lib/mps 앞에 추가해야 합니다.
 - Linux 플랫폼의 Java Enterprise System 설치인 경우:
/opt/sun/private/lib 경로를 *Apache-install-dir/bin/apachectl* 스크립트의 LD_LIBRARY_PATH에 추가합니다. 이 경로는 /usr/lib 앞에 추가해야 합니다.
 - Microsoft Windows:
 - a. Path 환경 변수에 새 경로를 추가합니다.
시작 ⇒ 설정 ⇒ 제어판 ⇒ 시스템 ⇒ 고급 ⇒ 환경 변수 ⇒ 시스템 변수를 누릅니다.
Application Server\install-dir\bin을 Path 환경 변수에 추가합니다.
 - b. 환경 변수 NSPR_NATIVE_THREADS_ONLY를 1로 설정합니다.
환경 변수 창의 시스템 변수에서 새로 만들기를 누릅니다. 변수 이름 NSPR_NATIVE_THREADS_ONLY 및 변수 값 1을 입력합니다.
 - c. 시스템을 다시 시작합니다.

▼ Apache 2용 보안 인증서 만들기

Apache에서 HTTPS 요청을 지원하려면 다음 단계가 필요합니다.

Apache에서 보안 인증서를 설정하는 방법에 대한 자세한 내용은 http://http.apache.org/docs/2.2/ssl/ssl_faq.html 및 http://www.modssl.org/docs/2.8/ssl_faq.html의 지침을 참조하십시오. 다음 절차는 이러한 웹 사이트에서 제공되었습니다.

1 해당 OpenSSL 디렉토리로 이동합니다.

- Solaris 10을 사용하는 경우 인증서 만들기를 위해 사전 설치된 OpenSSL을 사용합니다.
`cd /usr/sfw/bin`
- Solaris 9 또는 Linux를 사용하는 경우 OpenSSL이 설치된 디렉토리로 이동합니다.
107 페이지 “SSL 인식 Apache 설치”에 설명된 대로 OpenSSL에 대해 configure 및 make가 이미 실행되었어야 합니다.
환경 변수 `OPENSSL_CONF=OpenSSL-installation-directory/apps/openssl.cnf`를 설정합니다.

2 다음 명령을 실행하여 서버 인증서 및 키를 만듭니다.

```
openssl req -new -x509 -keyout newreq.pem -out newreq.pem -days 365
```

일반 이름을 묻는 메시지가 나타나면 Apache를 실행할 호스트 이름을 제공합니다. 다른 모든 프롬프트에 대해서는 사용자의 특정 요구 사항에 맞는 값을 입력합니다.

이 명령은 newreq.pem을 만듭니다.

3 openssl 명령이 실행된 위치에서 새로 만든 newreq.pem을 엽니다.

4 BEGIN CERTIFICATE로 시작하고 END CERTIFICATE로 끝나는 줄을 복사하여 *Apache-install-dir/conf/ssl.crt/server.crt*에 붙여넣습니다. 예를 들면 다음과 같습니다.

```
-----BEGIN CERTIFICATE-----
....
...
-----END CERTIFICATE-----
```

5 BEGIN RSA PRIVATE KEY로 시작하고 END RSA PRIVATE KEY로 끝나는 줄을 복사하여 *Apache-install-dir/conf/ssl.key/server.key*에 붙여넣습니다. 예를 들면 다음과 같습니다.

```
-----BEGIN RSA PRIVATE KEY-----
...
...
...
-----END RSA PRIVATE KEY-----
```

6 *Apache-install-dir/conf/ssl.conf*의 SSLCertificateKeyFile 및 SSLCertificateFile 변수에 올바른 값이 있는지 확인합니다.

- 7 **ServerName**이 **www.example.com**이 아닌지 확인합니다. **ServerName**은 서버 인증서와 키를 만들 때 입력했던 일반 이름과 일치하는 **Apache**가 실행될 실제 호스트 이름이어야 합니다.

Solaris 및 Linux에서 Apache 시작

일반적으로 **Application Server**를 설치한 동일한 사용자로 **Apache**를 시작해야 합니다. 다음과 같은 상황에서는 루트로 **Apache**를 시작해야 합니다.

- Java Enterprise System 사용자인 경우
- 1024보다 작은 포트 번호를 사용한 경우
- **Apache**를 시작한 사용자와 다른 사용자로 **Apache**가 실행되는 경우

SSL 모드에서 **Apache**를 시작하려면 다음 명령 중 하나를 사용합니다.

apachectl startssl 또는 **apachectl -k start -DSSL**

필요한 경우 **Apache** 웹 사이트에서 **Apache** 서버를 시작하는 방법에 대한 최신 정보를 확인합니다.

Microsoft IIS 사용

로드 밸런서 플러그인을 사용하도록 **Microsoft** 인터넷 정보 서비스(IIS)를 구성하려면 **Windows** 인터넷 서비스 관리자에서 특정 등록 정보를 수정합니다. 인터넷 서비스 관리자는 제어판 폴더의 관리 도구 폴더에 있습니다.

Sun Java System Application Server를 설치한 후에 다음과 같이 수정합니다.

▼ 로드 밸런서 플러그인을 사용하도록 Microsoft IIS 구성

- 1 인터넷 서비스 관리자를 엽니다.
- 2 플러그인을 사용할 웹 사이트를 선택합니다.
이 웹 사이트는 일반적으로 기본 웹 사이트로 명명됩니다.
- 3 웹 사이트를 마우스 오른쪽 버튼으로 누른 다음 등록 정보를 선택하여 등록 정보 노트북을 엽니다.
- 4 다음 단계에 따라 새 ISAPI 필터를 추가합니다.
 - a. ISAPI 필터 탭을 엽니다.

- b. 추가를 누릅니다.
 - c. 필터 이름 필드에 **Application Server**를 입력합니다.
 - d. 실행 파일 필드에서 **C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll**을 입력합니다.
 - e. 확인을 누르고 등록 정보 노트북을 닫습니다.
- 5 새로운 가상 디렉토리를 작성하고 구성합니다.
 - a. 기본 웹 사이트를 마우스 오른쪽 버튼으로 누른 다음 가상 디렉토리를 선택합니다.
가상 디렉토리 작성 마법사가 열립니다.
 - b. 별칭 필드에 **sun-passthrough**를 입력합니다.
 - c. 디렉터리 필드에 **C:\Inetpub\wwwroot\sun-passthrough**를 입력합니다.
 - d. 실행 권한 확인란을 선택합니다.
다른 모든 권한 관련 확인란은 선택 해제한 채로 둡니다.
 - e. 마침을 누릅니다.
- 6 **sun-passthrough.dll** 파일, **Application Server** *install-dir/bin* 및 **Application Server** *install-dir/lib* 경로를 시스템의 PATH 환경 변수에 추가합니다.
- 7 IIS 6.0 사용자의 경우 다음 단계를 수행하여 IIS 6에서 실행되도록 로드 밸런서 웹 서비스 확장을 구성합니다.
 - a. IIS 관리자에서 로컬 컴퓨터를 확장하고 웹 서비스 확장을 누릅니다.
 - b. 작업 창에서 새 웹 서비스 확장 추가를 선택합니다.
 - c. 확장 이름으로 **Sun-Passthrough**를 입력하고 추가를 누릅니다.
 - d. **sun-passthrough.dll**의 경로인 **C:\Inetpub\wwwroot\sun-passthrough**를 입력합니다.
 - e. 확인을 누릅니다.
 - f. 확장 상태를 허용됨으로 설정을 선택합니다.

- 8 IIS 6.0 사용자의 경우 C:\inetpub\wwwroot\sun-passthrough\lb.log 파일을 만들고 NTFS 쓰기 및 수정 권한을 이 파일의 IIS_WPG 그룹에 제공합니다.

IIS 6.0은 작업자 프로세스 격리 모드에서 실행되기 때문에 IIS_WPG 그룹의 보안 권한과 함께 IIS 서버를 실행합니다.

- 9 모든 IIS 사용자는 시스템을 다시 시작합니다.

- 10 웹 서버, 로드 밸런서 플러그인 및 Application Server가 제대로 작동하는지 확인합니다. 웹 브라우저에 다음 사항을 입력하여 웹 응용 프로그램 컨텍스트 루트에 액세스합니다. **http://web-server-name/web-application**. 여기서 *web-server-name*은 웹 서버의 호스트 이름 또는 IP 주소이고 *web-application*은 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties 파일에 나열한 컨텍스트 루트입니다.

정보 - ISAPI 필터 상태가 녹색이어야 합니다. 필터 상태를 확인하려면 웹 사이트의 등록 정보 노트북에 액세스하여 ISAPI 필터 탭을 누릅니다. 상태가 녹색이 아닌 경우 임의의 HTTP 요청을 IIS HTTP 포트에 보내 봅니다. 요청이 실패하면 정상입니다. ISAPI 필터 상태를 다시 확인합니다.

자동으로 구성된 sun-passthrough 등록 정보

설치 프로그램은 sun-passthrough.properties에서 다음 등록 정보를 자동으로 구성합니다. 기본값을 변경할 수 있습니다.

등록 정보	정의	기본값
lb-config-file	로드 밸런서 구성 파일 경로	IIS-www-root\sun-passthrough\loadbalancer.xml
log-file	로드 밸런서 로그 파일 경로	IIS-www-root\sun-passthrough\lb.log
log-level	웹 서버의 로그 수준	INFO

HTTP 로드 균형 조정 구성

이 절에서는 HTTP 로드 밸런서 플러그인에 대해 설명합니다. 다음 항목으로 구성됩니다.

- 117 페이지 “HTTP 로드 밸런서 작동 방식”
- 119 페이지 “HTTP 로드 균형 조정 설정”
- 121 페이지 “로드 밸런서 구성”
- 133 페이지 “다중 웹 서버 인스턴스 구성”
- 134 페이지 “가용성 손실 없이 응용 프로그램 업그레이드”
- 140 페이지 “HTTP 로드 밸런서 플러그인 모니터링”

다른 유형의 로드 균형 조정에 대한 자세한 내용은 10 장 및 11 장을 참조하십시오.

이 절에서는 Application Server에 포함된 HTTP 로드 균형 조정 플러그인을 사용하는 방법에 대해 설명합니다. 또 다른 HTTP 로드 균형 조정 옵션은 하드웨어 기반 로드 균형 조정 솔루션을 위해 Application Server와 함께 Sun Secure Application Switch를 사용하는 것입니다. 이 솔루션을 구성하는 방법에 대한 자습서는 [Clustering and Securing Web Applications: A Tutorial \(http://developers.sun.com/prodtech/appserver/reference/techart/load-balancing.html\)](http://developers.sun.com/prodtech/appserver/reference/techart/load-balancing.html) 기사를 참조하십시오.

HTTP 로드 밸런서 작동 방식

로드 밸런서는 여러 Application Server 인스턴스(독립 실행형 또는 클러스터링) 간에 작업 로드를 균일하게 분산시켜 시스템의 전반적인 처리량을 늘립니다.

로드 밸런서를 사용하면 서버 인스턴스 간에 요청을 페일오버할 수 있습니다. HTTP 세션 정보를 유지하려면 Enterprise Edition을 사용하고 HADB가 설치 및 설정되고 HTTP 세션 지속성이 구성되어야 합니다. 자세한 내용은 9 장을 참조하십시오.

주 - 로드 밸런서는 8k보다 큰 URI/URL을 처리하지 않습니다.

관리 콘솔이 아니라 **asadmin** 유틸리티를 사용하여 HTTP 로드 균형 조정을 구성합니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 118 페이지 “할당된 요청 및 할당되지 않은 요청”
- 118 페이지 “HTTP 로드 균형 조정 알고리즘”
- 119 페이지 “샘플 응용 프로그램”

할당된 요청 및 할당되지 않은 요청

처음에 HTTP 클라이언트에서 로드 밸런서로 수행된 요청은 새로운 세션에 대한 요청에 해당합니다. 새로운 세션에 대한 요청을 **할당되지 않은** 요청이라고 합니다. 로드 밸런서는 라운드 로빈 알고리즘에 따라 클러스터의 Application Server 인스턴스로 이 요청을 보냅니다.

Application Server 인스턴스에 세션이 만들어지면 로드 밸런서는 이 세션에 대한 모든 후속 요청의 경로를 특정 인스턴스로만 지정합니다. 기존 세션에 대한 요청을 **할당된** 요청 또는 **고정** 요청이라고 합니다.

HTTP 로드 균형 조정 알고리즘

Sun Java System Application Server 로드 밸런서는 **고정 라운드 로빈 알고리즘**을 사용하여 들어오는 HTTP 및 HTTPS 요청을 로드 균형 조정합니다. 지정된 세션의 모든 요청이 동일한 Application Server 인스턴스로 전송됩니다. 고정 로드 밸런서를 사용하면 세션 데이터가 클러스터의 모든 인스턴스에 분산되기 보다는 한 Application Server에 캐시됩니다.

따라서 고정 라운드 로빈 체계는 순수 라운드 로빈 체계가 주는 로드의 균형 분산이라는 이점을 뛰어난 성능 이점으로 대체합니다.

새 HTTP 요청이 로드 밸런서 플러그인으로 보내지면 경량 라운드 로빈 체계에 따라 Application Server 인스턴스로 전달됩니다. 계속해서 이 요청은 쿠키를 사용하거나 명시적 URL을 다시 작성하여 특정한 Application Server 인스턴스에 "고정"됩니다. 로드 밸런서는 고정 방법을 자동으로 결정합니다.

로드 밸런서 플러그인은 다음 방법을 사용하여 세션 고정을 판별합니다.

- **쿠키 방법:** 로드 밸런서 플러그인은 별도의 쿠키를 사용하여 라우트 정보를 기록합니다. 쿠키 기반 방법을 사용하려면 HTTP 클라이언트에서 쿠키를 지원해야 합니다.
- **명시적 URL 다시 쓰기:** 고정 정보가 URL에 추가됩니다. HTTP 클라이언트에서 쿠키를 지원하지 않더라도 이 방법을 사용할 수 있습니다.

고정 정보를 사용해서 로드 밸런서 플러그인은 먼저 이전에 요청을 전달했던 인스턴스를 판별합니다. 해당 인스턴스가 정상일 경우 로드 밸런서 플러그인은 특정 Application Server 인스턴스에 그 요청을 전달합니다. 따라서 지정한 세션의 모든 요청이 동일한 Application Server 인스턴스로 전달됩니다.

샘플 응용 프로그램

다음 디렉토리에는 로드 균형 조정 및 페일오버를 보여주는 샘플 응용 프로그램이 포함되어 있습니다.

`install-dir/samples/ee-samples/highavailabilityinstall-dir/samples/ee-samples/failover`

`ee-samples` 디렉토리에는 샘플을 실행할 수 있도록 환경을 설정하는 데 필요한 정보도 포함되어 있습니다.

HTTP 로드 균형 조정 설정

이 절에서는 로드 밸런서 플러그인을 설정하는 방법을 설명하며 다음 내용으로 구성됩니다.

- 119 페이지 “로드 균형 조정 설정을 위한 필수 조건”
- 120 페이지 “HTTP 로드 밸런서 배포”
- 120 페이지 “로드 균형 조정 설정 절차”

로드 균형 조정 설정을 위한 필수 조건

로드 밸런서를 구성하기 전에 다음을 수행해야 합니다.

- 웹 서버를 설치합니다.
- Apache Web Server를 사용할 경우 설치하기 전에 적절하게 설정해야 합니다. 자세한 내용은 104 페이지 “Apache Web Server 사용”을 참조하십시오.
- 로드 밸런서 플러그인을 설치합니다.
설치 절차에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Installation Guide**(독립 실행형 Application Server를 사용할 경우) 또는 **Sun Java Enterprise System 5 Installation Guide for UNIX**(Java Enterprise System을 사용할 경우)를 참조하십시오.
- Apache Web Server 또는 Microsoft IIS를 사용할 경우 웹 서버를 구성합니다. 4 장을 참조하십시오.
- 로드 균형 조정에 참여할 Application Server 클러스터 또는 서버 인스턴스를 만듭니다.
- 이러한 클러스터나 인스턴스에 응용 프로그램을 배포합니다.

HTTP 로드 밸런서 배포

다음 절에 설명된 것처럼 작업 목표 및 환경에 따라 여러 다른 방법으로 로드 밸런서를 구성할 수 있습니다.

- 120 페이지 “클러스터링된 서버 인스턴스 사용”
- 120 페이지 “역 프록시 플러그인으로 사용되는 로드 밸런서와 함께 단일 독립 실행형 인스턴스 사용”
- 120 페이지 “여러 독립 실행형 인스턴스 사용”

클러스터링된 서버 인스턴스 사용

로드 밸런서를 배포하는 가장 일반적인 방법은 서버 인스턴스 클러스터를 사용하는 것입니다. 기본적으로 클러스터의 모든 인스턴스는 동일한 구성을 가지며 동일한 응용 프로그램이 배포됩니다. 로드 밸런서는 서버 인스턴스 간에 작업 로드를 분산시키며 상태가 나쁜 인스턴스에서 상태가 좋은 인스턴스로 요청이 패일오버됩니다. HTTP 세션 지속성을 구성한 경우 요청이 패일오버될 때 세션 정보가 유지됩니다.

여러 개의 클러스터가 있는 경우 요청이 로드 균형 조정된 후 단일 클러스터에 있는 인스턴스 간에 패일오버됩니다. 로드 밸런서의 여러 클러스터를 사용하여 응용 프로그램의 롤링 업그레이드를 쉽게 수행할 수 있습니다. 자세한 내용은 [134 페이지 “가용성 손실 없이 응용 프로그램 업그레이드”](#)를 참조하십시오.

역 프록시 플러그인으로 사용되는 로드 밸런서와 함께 단일 독립 실행형 인스턴스 사용

클러스터 대신 독립 실행형 서버 인스턴스를 사용하도록 로드 밸런서를 구성할 수도 있습니다. 이렇게 구성하면 로드 밸런서 플러그인이 역 프록시 플러그인(통과 플러그인이라고도 함)으로 작동합니다. 웹 서버가 로드 밸런서에서 활성화된 응용 프로그램에 대한 요청을 받으면 Application Server로 직접 요청을 전달합니다.

서버 인스턴스의 클러스터로 구성할 때와 동일한 절차에 따라 통과 플러그인에 대한 로드 밸런서를 구성합니다.

여러 독립 실행형 인스턴스 사용

여러 독립 실행형 인스턴스를 사용하고 인스턴스 간에 로드 균형 조정 및 요청을 패일오버하도록 로드 밸런서를 구성할 수도 있습니다. 그러나 이 구성에서 독립 실행형 인스턴스가 동기종 환경을 가지며 동일한 응용 프로그램이 배포되도록 직접 확인해야 합니다. 클러스터는 자동으로 동기종 환경을 유지하므로 대부분의 환경에서는 클러스터를 사용하는 것이 더 쉽고 적절합니다.

로드 균형 조정 설정 절차

asadmin 도구를 사용하여 환경의 로드 균형 조정을 구성합니다. 이 단계에서 사용되는 asadmin 명령에 대한 자세한 내용은 [121 페이지 “로드 밸런서 구성”](#)을 참조하십시오.

▼ 로드 균형 조정 설정

- 1 `asadmin create-http-lb-config`를 사용하여 로드 밸런서 구성을 만듭니다.
- 2 `asadmin create-http-lb-ref`를 사용하여 로드 밸런서로 관리할 클러스터 또는 독립 실행형 서버 인스턴스에 대한 참조를 추가합니다.
대상과 함께 로드 밸런서 구성을 만들었고 그 대상이 로드 밸런서가 참조하는 클러스터나 독립 실행형 서버 인스턴스일 경우 이 단계를 생략합니다.
- 3 `asadmin enable-http-lb-server`를 사용하여 로드 밸런서가 참조하는 클러스터나 독립 실행형 서버 인스턴스를 활성화합니다.
- 4 `asadmin enable-http-lb-application`을 사용하여 로드 균형 조정을 할 수 있도록 응용 프로그램을 활성화합니다.
로드 밸런서에서 참조하는 클러스터나 독립 실행형 인스턴스에서 사용할 수 있도록 응용 프로그램을 이미 배포하여 활성화되어 있어야 합니다. 로드 균형 조정을 위한 응용 프로그램 활성화는 사용을 위한 활성화와 다른 단계입니다.
- 5 `asadmin create-health-checker`를 사용하여 상태 검사기를 만듭니다.
상태 검사기는 비정상적인 서버 인스턴스가 다시 정상적이 되면 로드 밸런서가 새로운 요청을 전송할 수 있도록 이들을 모니터링합니다.
- 6 `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 생성합니다.
이 명령을 사용하면 Sun Java System Application Server에 포함되어 있는 로드 밸런서 플러그인과 함께 사용할 수 있는 구성 파일이 생성됩니다.
- 7 로드 밸런서 플러그인 구성 파일이 저장되는 웹 서버 `config` 디렉토리로 로드 밸런서 구성 파일을 복사합니다.

로드 밸런서 구성

로드 밸런서 구성은 `domain.xml` 파일에 있는 구성입니다. 로드 밸런서 구성은 매우 융통성이 있습니다.

- 로드 밸런서마다 하나의 로드 밸런서 구성만 갖지만, 각 로드 밸런서 구성에 여러 로드 밸런서를 연결할 수 있습니다.
- 로드 밸런서는 하나의 도메인만 지원하지만 한 도메인은 여러 로드 밸런서를 연결할 수 있습니다.

이 절에서는 다음 내용을 비롯하여 로드 밸런서 구성을 만들고 수정하고 사용하는 방법을 설명합니다.

- 122 페이지 “HTTP 로드 밸런서 구성 만들기”
- 123 페이지 “HTTP 로드 밸런서 참조 만들기”
- 123 페이지 “로드 균형 조정을 위해 서버 인스턴스 활성화”
- 123 페이지 “로드 균형 조정을 위해 응용 프로그램 활성화”
- 124 페이지 “HTTP 상태 검사기 만들기”
- 125 페이지 “로드 밸런서 구성 파일 내보내기”
- 126 페이지 “로드 밸런서 구성 변경”
- 126 페이지 “동적 재구성 활성화”
- 127 페이지 “서버 인스턴스 또는 클러스터 비활성화(정지)”
- 127 페이지 “응용 프로그램 비활성화(정지)”
- 128 페이지 “HTTP 및 HTTPS 페일오버 구성”
- 129 페이지 “로드 밸런서와 함께 리디렉션 사용”
- 132 페이지 “멱등원(Idempotent) URL 구성”

HTTP 로드 밸런서 구성 만들기

asadmin 명령 create-http-lb-config를 사용하여 로드 밸런서 구성을 만듭니다.

표 5-1에서는 매개 변수에 대해 설명합니다. 자세한 내용은 create-http-lb-config, delete-http-lb-config 및 list-http-lb-configs에 대한 설명서를 참조하십시오.

표 5-1 로드 밸런서 구성 매개 변수

매개 변수	설명
response timeout	서버 인스턴스에서 응답을 반환해야 하는 시간(초)입니다. 기간 내에 응답을 받지 못하면 서버가 비정상인 것으로 간주됩니다. 기본값은 60입니다.
HTTPS 라우팅	로드 밸런서에 대한 HTTPS 요청으로 서버 인스턴스에 대한 HTTPS 또는 HTTP 요청이 발생하는지 여부입니다. 자세한 내용은 129 페이지 “HTTPS 라우팅 구성”을 참조하십시오.
reload interval	로드 밸런서 구성 파일 loadbalancer.xml의 변경 사항을 확인하는 간격입니다. 변경 사항이 확인되면 구성 파일을 다시 로드합니다. 값 0은 다시 로드를 비활성화합니다. 자세한 내용은 126 페이지 “동적 재구성 활성화”를 참조하십시오.
monitor	로드 밸런서에 대한 모니터링을 활성화할지 여부입니다.
routecookie	경로 정보를 기록할 때 로드 밸런서 플러그인에서 사용하는 쿠키 이름입니다. HTTP 클라이언트가 쿠키를 지원해야 합니다. 쿠키를 저장하기 전에 확인하도록 브라우저를 설정한 경우 쿠키 이름은 JROUTE입니다.
target	로드 밸런서 구성의 대상입니다. 대상을 지정한 경우 대상에 대한 참조를 추가한 것과 같습니다. 클러스터나 독립 실행형 인스턴스가 대상이 될 수 있습니다.

HTTP 로드 밸런서 참조 만들기

로드 밸런서에 독립 실행형 서버나 클러스터에 대한 참조를 만들면 로드 밸런서를 통해 제어되는 대상 서버나 클러스터 목록에 해당 서버나 클러스터가 추가됩니다. 참조된 서버나 클러스터를 `enable-http-lb-server`를 사용하여 활성화해야 관련 요청의 로드 균형 조절을 수행할 수 있습니다. 대상과 함께 로드 밸런서 구성을 만들었다면 해당 대상이 이미 참조로 추가되어 있습니다.

`create-http-lb-ref`를 사용하여 참조를 만듭니다. 로드 밸런서 구성 이름과 대상 서버 인스턴스 또는 클러스터를 제공해야 합니다.

참조를 삭제하려면 `delete-http-lb-ref`를 사용합니다. 참조를 삭제하려면 `disable-http-lb-server`를 사용하여 참조된 서버나 클러스터를 비활성화해야 합니다.

자세한 내용은 `create-http-lb-ref` 및 `delete-http-lb-ref`에 대한 설명서를 참조하십시오.

로드 균형 조절을 위해 서버 인스턴스 활성화

서버 인스턴스나 클러스터에 대한 참조를 만든 후에 `enable-http-lb-server`를 사용하여 서버 인스턴스나 클러스터를 활성화합니다. 로드 밸런서 구성을 만들 때 서버 인스턴스나 클러스터를 대상으로 사용한 경우 이를 활성화해야 합니다.

자세한 내용은 `enable-http-lb-server`에 대한 설명서를 참조하십시오.

로드 균형 조절을 위해 응용 프로그램 활성화

로드 밸런서를 통해 관리되는 모든 서버는 동기종 구성을 가져야 하며 동일한 응용 프로그램들이 배포되어야 합니다. 응용 프로그램을 배포하고 액세스할 수 있도록 활성화한 경우(배포 단계 중 또는 이후에 발생함) 로드 균형 조절을 위해 응용 프로그램을 활성화해야 합니다. 로드 균형 조절을 위해 응용 프로그램을 활성화하지 않은 경우 응용 프로그램이 배포된 서버에 대한 요청은 로드 균형 조절되고 페일오버되더라도 이 응용 프로그램에 대한 요청은 로드 균형 조절되지 않고 페일오버됩니다.

응용 프로그램을 활성화할 경우 응용 프로그램 이름과 대상을 지정합니다. 로드 밸런서에서 여러 대상(예: 두 개의 클러스터)을 관리할 경우 모든 대상에서 응용 프로그램을 활성화합니다.

자세한 내용은 `enable-http-lb-application`에 대한 온라인 도움말을 참조하십시오.

새로운 응용 프로그램을 배포할 경우 로드 균형 조절을 위해 응용 프로그램을 활성화하고 로드 밸런서 구성을 다시 내보내야 합니다.

HTTP 상태 검사기 만들기

로드 밸런서의 상태 검사는 비정상적으로 표시된 구성되어 있는 모든 Application Server 인스턴스를 주기적으로 검사합니다. 상태 검사는 필수가 아닙니다. 그러나 상태 검사기가 없거나 상태 검사기가 비활성화된 경우 비정상 인스턴스의 정기적인 상태 검사가 수행되지 않습니다. 로드 밸런서는 비정상 인스턴스가 정상이 되는 시점을 확인할 수 없습니다.

로드 밸런서의 상태 검사 기법은 HTTP를 사용하여 Application Server 인스턴스와 통신합니다. 상태 검사는 지정한 URL에 HTTP 요청을 보내고 응답을 기다립니다. HTTP 응답 헤더의 상태 코드가 100과 500 사이에 있으면 인스턴스가 정상임을 의미합니다.

상태 검사기 만들기

상태 검사기를 만들려면 `asadmin create-http-health-checker` 명령을 사용합니다. 다음 매개 변수를 지정합니다.

표 5-2 상태 검사기 매개 변수

매개 변수	설명	기본값
url	로드 밸런서가 검사하여 상태를 확인할 Listener의 URL을 지정합니다.	"/"
interval	인스턴스의 상태 검사가 발생하는 간격(초)을 지정합니다. 0을 지정하면 상태 검사기가 비활성화됩니다.	30초
timeout	Listener를 정상으로 간주하기 위해 응답을 받아야 하는 시간 초과 간격(초)을 지정합니다.	10초

Application Server 인스턴스가 비정상적으로 표시되면 상태 검사는 비정상 인스턴스를 폴링하여 인스턴스가 정상적으로 되었는지 확인합니다. 상태 검사는 지정된 URL을 사용하여 모든 비정상 Application Server 인스턴스를 검사하고 정상 상태로 되었는지 확인합니다.

상태 검사기에 비정상 인스턴스가 정상이 되었음을 확인하면 해당 인스턴스는 정상 인스턴스 목록에 추가됩니다.

자세한 내용은 `create-http-health-checker` 및 `delete-http-health-checker`에 대한 설명서를 참조하십시오.

정상 인스턴스에 대한 추가 상태 검사 등록 정보

`create-http-health-checker`로 만든 상태 검사기만 비정상 인스턴스를 검사합니다. 정상 인스턴스를 정기적으로 검사하려면 내보낸 `loadbalancer.xml` 파일에 추가 등록 정보를 설정합니다.

주 - 이러한 등록 정보는 `loadbalancer.xml`을 내보낸 후에 수동으로 편집해야만 설정할 수 있습니다. 사용할 수 있는 동등한 `asadmin` 명령이 없습니다.

정상 인스턴스를 검사하려면 다음 등록 정보를 설정합니다.

표 5-3 상태 검사기 수동 등록 정보

등록 정보	정의
<code>active-healthcheck-enabled</code>	정상 서버 인스턴스를 ping하여 정상인지 확인할지 여부를 나타내는 <code>True/false</code> 플래그입니다. 서버 인스턴스를 ping하려면 플래그를 <code>true</code> 로 설정합니다.
<code>number-healthcheck-retries</code>	서버 인스턴스를 비정상적으로 표시하기 전에 로드 밸런서의 상태 검사기가 응답이 없는 서버 인스턴스를 ping하는 횟수를 지정합니다. 유효한 값은 1과 1000 사이입니다. 설정된 기본값은 3입니다.

`loadbalancer.xml` 파일을 편집하여 등록 정보를 설정합니다. 예를 들면 다음과 같습니다.

```
<property name="active-healthcheck-enabled" value="true"/>
<property name="number-healthcheck-retries" value="3"/>
```

이러한 등록 정보를 추가한 다음 `loadbalancer.xml` 파일을 편집하여 다시 내보낼 경우 새로 내보낸 구성에는 이러한 등록 정보가 없습니다. 새로 내보낸 구성에 이러한 등록 정보를 다시 추가해야 합니다.

로드 밸런서 구성 파일 내보내기

Sun Java System Application Server에 포함되어 있는 로드 밸런서 플러그인은 `loadbalancer.xml`이라는 구성 파일을 사용합니다. `asadmin` 유틸리티를 사용하여 `domain.xml` 파일에 로드 밸런서 구성을 만듭니다. 로드 균형 조정 환경을 구성한 후 `domain.xml`에서 `loadbalancer.xml` 파일로 내보냅니다.

▼ 로드 밸런서 구성 내보내기

- 1 `asadmin` 명령 `export-http-lb-config`를 사용하여 `loadbalancer.xml` 파일을 내보냅니다. 특정 로드 밸런서 구성에 대한 `loadbalancer.xml` 파일을 내보냅니다. 경로 및 다른 파일 이름을 지정할 수 있습니다. 파일 이름을 지정하지 않으면 `loadbalancer.xml`. `load-balancer-config-name`으로 이름이 지정됩니다. 경로를 지정하지 않을 경우 파일은 `domain-dir/generated` 디렉토리에 만들어집니다.

Windows에서 경로를 지정하려면 경로를 따옴표로 묶습니다. 예를 들면 `"C:\Sun\AppServer\loadbalancer.xml"`과 같습니다.

2 내보낸 로드 밸런서 구성 파일을 웹 서버의 구성 디렉토리에 복사합니다.

예를 들어, Sun Java System Web Server의 경우 일반적인 위치는 `web-server-root/config`입니다.

웹 서버의 구성 디렉토리에 있는 로드 밸런서 구성 파일의 이름은 `loadbalancer.xml`입니다. 파일 이름이 다를 경우(예: `loadbalancer.xml`, `load-balancer-config-name`) 이름을 변경해야 합니다.

로드 밸런서 구성 변경

서버에 대한 참조를 만들거나 삭제하고, 새 응용 프로그램을 배포하고, 서버나 응용 프로그램을 활성화 또는 비활성화하는 등의 작업을 수행하여 로드 밸런서 구성을 변경할 경우 로드 밸런서 파일을 다시 내보낸 후 웹 서버의 `config` 디렉토리에 복사합니다. 자세한 내용은 [125 페이지](#) “로드 밸런서 구성 파일 내보내기”를 참조하십시오.

로드 밸런서 플러그인은 로드 밸런서 구성에 지정한 다시 로드 간격을 기반으로 정기적으로 업데이트된 구성이 있는지 확인합니다. 지정한 시간 후 로드 밸런서가 새로운 구성 파일을 발견하면 해당 구성을 사용하기 시작합니다.

동적 재구성 활성화

동적 재구성을 수행할 경우 로드 밸런서 플러그인은 업데이트된 구성을 주기적으로 확인합니다.

동적 재구성을 활성화하려면 다음 작업을 수행합니다.

- 로드 밸런서 구성을 만들 때는 `asadmin create-http-lb-config`와 함께 `--reloadinterval` 옵션을 사용합니다.
이 옵션은 로드 밸런서 구성 파일 `loadbalancer.xml`의 변경 사항을 검사하는 간격을 설정합니다. 값이 0이면 동적 재구성이 비활성화됩니다. 기본적으로 동적 재구성은 60초의 다시 로드 간격으로 활성화됩니다.
- 동적 재구성을 이전에 비활성화했거나 다시 로드 간격을 변경하려면 `asadmin set` 명령을 사용합니다.
다시 로드 간격을 변경한 후에 로드 밸런서 구성 파일을 다시 내보내고 웹 서버의 `config` 디렉토리에 복사한 경우 웹 서버를 다시 시작합니다.

주- 재구성하는 중 로드 밸런서에 하드 디스크 읽기 오류가 발생하는 경우 현재 메모리에 있는 구성을 사용합니다. 로드 밸런서는 기존 구성을 덮어쓰기 전에 수정된 구성 데이터가 DTD와 호환되는지 확인합니다.

디스크 읽기 오류가 발생한 경우 경고 메시지가 웹 서버의 오류 로그 파일에 로깅됩니다.

Sun Java System Web Server의 오류 로그는 다음 위치에 있습니다.

`web-server-install-dir/web-server-instance /logs/`

서버 인스턴스 또는 클러스터 비활성화(정지)

어떤 이유로 Application Server를 중지하기 전에 인스턴스가 요청 처리를 완료해야 합니다. 서버 인스턴스나 클러스터를 적절하게 비활성화하는 프로세스를 정지라고 합니다.

로드 밸런서는 Application Server 인스턴스를 정지하는 데 다음과 같은 정책을 사용합니다.

- 인스턴스(독립 실행형 또는 클러스터의 일부)를 비활성화했지만 시간 초과가 만기되지 않은 경우 고정된 요청이 계속 해당 인스턴스로 전달됩니다. 그러나 새로운 요청은 비활성화된 인스턴스로 전송되지 않습니다.
- 시간 초과가 만료되면 인스턴스가 비활성화됩니다. 로드 밸런서에서 인스턴스로 열려있는 모든 연결이 닫힙니다. 이 인스턴스에 고정된 모든 세션이 잘못되지 않았더라도 로드 밸런서는 이 인스턴스에 요청을 전송하지 않습니다. 대신 로드 밸런서는 고정된 요청을 다른 정상적인 인스턴스에 페일오버합니다.

▼ 서버 인스턴스나 클러스터 비활성화

- 1 `asadmin disable-http-lb-server`를 실행하여 시간 초과(분)를 설정합니다.
- 2 `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.
- 3 내보낸 구성을 웹 서버 config 디렉토리로 복사합니다.
- 4 서버 인스턴스나 인스턴스를 중지합니다.

응용 프로그램 비활성화(정지)

웹 응용 프로그램의 배포를 취소하기 전에 응용 프로그램이 요청 처리 완료해야 합니다. 응용 프로그램을 적절하게 비활성화하는 프로세스를 정지라고 합니다. 응용 프로그램을 정지할 경우 시간 초과 기간을 지정해야 합니다. 시간 초과 기간에 따라 로드 밸런서는 응용 프로그램 정지를 위해 다음 정책을 사용합니다.

- 시간 초과가 만료되지 않으면 로드 밸런서는 응용 프로그램에 새 요청을 전달하지 않고 웹 서버로 반환합니다. 그러나 로드 밸런서는 시간 초과가 만료될 때까지 고정 요청을 전달합니다.
- 시간 초과가 만료되면 로드 밸런서는 고정 요청을 비롯하여 응용 프로그램에 대한 어떠한 요청도 받아들이지 않습니다.

로드 밸런서가 참조하는 모든 서버 인스턴스나 클러스터에서 응용 프로그램을 비활성화할 경우 비활성화된 응용 프로그램의 사용자는 응용 프로그램이 다시 활성화될 때까지 서비스가 손실됩니다. 다른 서버 인스턴스나 클러스터에서 응용 프로그램을 활성화한 채로 두고 하나의 서버 인스턴스나 클러스터에서만 응용 프로그램을 비활성화할 경우 사용자는 계속 그 응용 프로그램에 액세스할 수 있습니다. 자세한 내용은 [134 페이지](#) “가용성 손실 없이 응용 프로그램 업그레이드”를 참조하십시오.

▼ 응용 프로그램 비활성화

- 1 다음을 지정하여 `asadmin disable-http-lb-application`을 사용합니다.
 - 시간 초과(분)
 - 비활성화할 응용 프로그램의 이름
 - 비활성화할 대상 클러스터 또는 인스턴스
- 2 `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.
- 3 내보낸 구성을 웹 서버 `config` 디렉토리로 복사합니다.

HTTP 및 HTTPS 페일오버 구성

세션이 연결된 원래 Application Server 인스턴스를 사용할 수 없게 된 경우 로드 밸런서 플러그인은 HTTP/HTTPS 세션을 다른 Application Server 인스턴스로 페일오버합니다. 이 절에서는 HTTP/HTTPS 라우팅 및 세션 페일오버를 활성화하도록 로드 밸런서 플러그인을 구성하는 방법에 대해 설명합니다.

HTTPS 라우팅

로드 밸런서 플러그인은 들어오는 모든 HTTP 또는 HTTPS 요청의 경로를 응용 프로그램 서버 인스턴스로 지정합니다. 그러나 HTTPS 라우팅이 활성화된 경우 로드 밸런서 플러그인은 HTTPS 포트만 사용하는 Application Server로 HTTPS 요청을 전달합니다. 새로운 요청과 고정된 요청 모두에 대해 HTTPS 라우팅을 수행합니다.

HTTPS 요청을 수신하고 진행 중인 세션이 없을 경우 로드 밸런서 플러그인은 HTTPS 포트가 구성된 사용 가능한 Application Server 인스턴스를 선택하고 요청을 인스턴스로 전달합니다.

진행 중인 HTTP 세션에서 동일한 세션에 대한 새로운 HTTPS 요청을 받으면 세션과 HTTP 세션 중에 저장된 고정된 정보를 사용하여 HTTPS 요청을 라우팅합니다. 마지막 HTTP 요청이 처리된 동일한 서버로 새로운 HTTPS 요청 경로가 지정되지만 HTTPS 포트에 지정됩니다.

HTTPS 라우팅 구성

`create-http-lb-config` 명령의 `httpsrouting` 옵션은 로드 균형 조정에 참여하는 모든 Application Server에 대해 HTTPS 라우팅의 설정 또는 해제 여부를 제어합니다. 이 옵션을 `false`로 설정하면 모든 HTTP 및 HTTPS 요청이 HTTP로 전달됩니다. `true`로 설정하면 HTTPS가 HTTPS 요청으로 전달됩니다. 로드 밸런서 구성을 새로 만들 경우 HTTPS 라우팅을 설정하거나 나중에 `asadmin set` 명령을 사용하여 변경합니다.

주-

- HTTPS 라우팅을 사용하려면 HTTPS listener를 하나 이상 구성해야 합니다.
- `https-routing`을 `true`로 설정한 경우 새로운 요청이나 고정된 요청이 들어오는데 클러스터에 정상적인 HTTPS listener가 없으면 그 요청에 대해 오류가 발생합니다.

알려진 문제점

로드 밸런서를 사용하여 HTTP/HTTPS 요청을 처리할 때 다음과 같은 제한이 적용됩니다.

- 세션에 HTTP 및 HTTPS 요청이 모두 있을 경우 첫 번째 요청은 HTTP 요청이어야 합니다. 첫 번째 요청이 HTTPS 요청일 경우 뒤에 HTTP 요청이 올 수 없습니다. 이는 HTTPS 세션과 연관된 쿠키를 브라우저에서 반환하지 않기 때문입니다. 브라우저는 두 개의 다른 프로토콜을 두 개의 다른 서버로 해석하고 새로운 세션을 시작합니다. `httpsrouting`을 `true`로 설정한 경우에만 이 제한 사항이 유효합니다.
- 세션에 HTTP와 HTTPS 요청이 모두 있을 경우 Application Server 인스턴스가 HTTP와 HTTPS listener 모두에 대해 구성되어야 합니다. `httpsrouting`을 `true`로 설정한 경우에만 이 제한 사항이 유효합니다.
- 세션에 HTTP 및 HTTPS 요청이 모두 있을 경우 Application Server 인스턴스를 표준 포트 번호(HTTP의 경우 80, HTTPS의 경우 443)를 사용하는 HTTP 및 HTTPS listener와 함께 구성해야 합니다. `httpsrouting`에 설정된 값과 상관없이 이 제한 사항이 유효합니다.

로드 밸런서와 함께 리디렉션 사용

리디렉션을 사용하여 한 URL에서 다른 URL로 요청을 리디렉션합니다. 예를 들어, 리디렉션을 사용하여 사용자를 다른 웹 사이트로 보내거나(예: 이전 버전의 응용 프로그램에서 새 버전으로 리디렉션) HTTP에서 HTTPS로 또는 HTTPS에서 HTTP로 보냅니다. 응용 프로그램에서 다양한 방법으로 리디렉션을 활성화할 수 있습니다(예:

서블릿 기반 리디렉션, web.xml 리디렉션). 그러나 로드 밸런서를 통해 리디렉션 URL을 보내려면 Application Server 또는 로드 밸런서의 일부 추가 구성이 필요할 수 있습니다. 리디렉션은 HTTPS 라우팅을 사용하여 전달되는 요청과 다릅니다. 리디렉션을 사용할 경우 httpsrouting을 false로 설정합니다. HTTP에 전달할 HTTPS 요청을 구성할 경우 128 페이지 “HTTPS 라우팅”을 사용합니다.

리디렉션에 영향을 주는 등록 정보는 HTTP 서비스 또는 HTTP Listener의 authPassthroughEnabled 및 proxyHandler 등록 정보와 loadbalancer.xml 파일의 rewrite-location 등록 정보입니다.

authPassthroughEnabled 등록 정보

Application Server authPassthroughEnabled 등록 정보가 true로 설정된 경우 원래 클라이언트 요청에 대한 정보(예: 클라이언트 IP 주소, SSL 키 크기 및 인증된 클라이언트 인증서 체인)가 사용자 요청 헤더를 사용하여 HTTP Listener에 보내집니다.

authPassthroughEnabled 등록 정보를 사용하면 더 빠른 SSL 인증을 위해 하드웨어 가속기(설치되어 있는 경우)를 활용할 수 있습니다. 클러스터링된 각 Application Server 인스턴스 대신에 로드 밸런서에서 하드웨어 가속기를 구성하는 것이 더 쉽습니다.



주의 - Application Server가 방화벽 뒤에 있는 경우에만 authPassthroughEnabled를 true로 설정합니다.

asadmin set 명령을 사용하여 HTTP 서비스 또는 개별 HTTP Listener에서 authPassthroughEnabled 등록 정보를 설정합니다. 개별 HTTP Listener에 대한 설정은 HTTP 서비스에 대한 설정보다 우선합니다.

모든 HTTP/HTTPS Listener에서 authPassthroughEnabled 등록 정보를 설정하려면 다음 명령을 사용합니다.

```
asadmin set  
cluster-name-config.http-service.property.authPassthroughEnabled=true
```

개별 수신기에서 설정하려면 다음 명령을 사용합니다.

```
asadmin set cluster-name-config.http-service.http-listener.listener-name.property.  
authPassthroughEnabled=true
```

proxyHandler 등록 정보

Application Server에 대한 프록시 처리기는 프록시 서버(이 경우에는 로드 밸런서)에서 인터셉트하여 Application Server에 전달한 원래 클라이언트 요청에 대한 정보를 검색하고 클라이언트 요청의 대상인 웹 응용 프로그램(Application Server에 배포된)에서 이 정보를 사용할 수 있게 만드는 역할을 수행합니다. 인터셉트 프록시 서버가 SSL 종단인 경우 프록시 처리기는 원래 요청에 대한 추가 정보(예: 원래 요청이 HTTPS

요청이었는지 여부 및 SSL 클라이언트 인증이 활성화되었는지 여부)를 검색하여 사용할 수 있게 만듭니다. `authPassThroughEnabled`가 `true`로 설정된 경우에만 `proxyHandler` 등록 정보를 사용합니다.

프록시 처리기는 들어오는 요청에서 프록시 서버가 원래 클라이언트 요청에 대한 정보를 전달하는 데 사용하는 사용자 정의 요청 헤더를 검사하고 표준 `ServletRequest` API를 사용하여 `Application Server`의 웹 응용 프로그램에서 이 정보를 사용할 수 있게 만듭니다.

프록시 처리기 구현은 `proxyHandler` 등록 정보를 사용하여 HTTP 서비스 수준에서 전역적으로 구성하거나 개별 HTTP Listener에 대해 구성할 수 있습니다. 이 등록 정보의 값은 `com.sun.appserv.ProxyHandler` 추상 클래스의 구현에 대한 정규화된 클래스 이름을 지정합니다. 구성 가능한 프록시 처리기 구현을 사용하면 `Application Server`에서 모든 프록시 서버를 사용할 수 있습니다. 단, 이 경우 프록시 처리기 구현은 HTTP 요청 헤더 이름과 해당 값의 형식을 알고 있어야 합니다. 이를 통해 프록시 서버는 원래 클라이언트 요청에 대한 정보를 전달합니다.

`Application Server`에 대한 프록시 처리기는 요청 헤더에서 SSL 인증서 체인을 읽고 구문 분석합니다. 따라서 백엔드 `Application Server` 인스턴스는 SSL 종단 프록시 서버(이 경우, 로드 밸런서)가 인터셉트한 원래 클라이언트 요청에 대한 정보를 검색할 수 있습니다. 기본 프록시 처리기 설정을 사용하거나 HTTP 서비스 또는 HTTP/HTTPS Listener의 `proxyHandler` 등록 정보를 사용하여 고유한 설정을 구성할 수 있습니다. `proxyHandler` 등록 정보는 이 수신기나 모든 수신기에 사용되는 `com.sun.appserv.ProxyHandler` 추상 클래스의 사용자 구현에 대한 정규화된 클래스 이름을 지정합니다.

이 추상 클래스 구현은 지정된 요청에서 사용자 요청 헤더를 검사합니다. 사용자 요청 헤더를 통해 프록시 서버는 원래 클라이언트 요청에 대한 정보를 `Application Server` 인스턴스에 전달하고 해당 정보를 호출자에게 반환합니다. 기본 구현은 `Proxy-ip`라는 HTTP 요청 헤더에서 클라이언트 IP 주소를, `Proxy-keysize`라는 HTTP 요청 헤더에서 SSL 키 크기를, `Proxy-auth-cert`라는 HTTP 요청 헤더에서 SSL 클라이언트 인증서 체인을 읽습니다. `Proxy-auth-cert` 값은 BEGIN CERTIFICATE 및 END CERTIFICATE 경계가 없고 `%d%a`로 바뀌는 `\n`이 있는 BASE-64로 인코딩된 클라이언트 인증서 체인을 포함해야 합니다.

`authPassThroughEnabled`가 `true`로 설정된 경우에만 이 등록 정보를 사용할 수 있습니다. 개별 HTTP 또는 HTTPS Listener에서 `proxyHandler` 등록 정보를 설정한 경우 모든 수신기에 대한 기본 설정을 대체합니다.

`asadmin set` 명령을 사용하여 HTTP 서비스 또는 개별 HTTP Listener에서 `proxyHandler` 등록 정보를 설정합니다.

`proxyHandler` 등록 정보를 모든 HTTP/HTTPS Listener에서 설정하려면 다음 명령을 사용합니다.

```
asadmin set cluster-name-config.http-service.property.proxyHandler=classname
```

개별 수신기에서 설정하려면 다음 명령을 사용합니다.

asadmin set

cluster-name-config.http-service.http-listener.listener-name.property.proxyHandler=classname

rewrite-location 등록 정보

true로 설정된 경우 rewrite-location 등록 정보는 원래 요청 정보를 다시 쓰고 프로토콜(HTTP 또는 HTTPS), 호스트 및 포트 정보를 포함합니다. 기본적으로 rewrite-location 등록 정보는 이전 Application Server 릴리스와의 호환성을 유지하기 위해 true로 설정됩니다.

rewrite-location 등록 정보는 `asadmin create-http-lb-config` 또는 `asadmin set` 명령을 통해 사용할 수 없습니다. 이 등록 정보를 사용하려면 로드 밸런서 구성을 내보낸 후에 `loadbalancer.xml` 파일에 수동으로 추가합니다. 예를 들어, 내보낸 `loadbalancer.xml` 파일에 다음을 추가합니다.

```
<property name="rewrite-location" value="false"/>
```

rewrite-location 등록 정보를 설정할 때 다음 사항에 주의합니다.

- `httpsrouting`이 false이고 `authPassthroughEnabled`가 Application Server에서 활성화되지 않은 경우 rewrite-location 등록 정보를 true로 설정합니다. `authPassthroughEnabled`가 활성화되지 않은 경우 Application Server는 원래 요청의 프로토콜(HTTP 또는 HTTPS)을 인식하지 않습니다. rewrite-location을 true로 설정하면 로드 밸런서는 rewrite location의 프로토콜 부분을 적절하게 수정합니다. 즉, 클라이언트에서 HTTPS 요청을 보내는 중이면 로드 밸런서는 클라이언트를 로드 밸런서의 HTTPS 사용 가능 수신기 포트에 리디렉션합니다. HTTP 요청의 경우 프로세스는 동일합니다.
- `httpsrouting`이 false이고 `authPassthroughEnabled`가 Application Server에서 활성화된 경우 Application Server에서 클라이언트 요청이 HTTP인지 아니면 HTTPS인지 인식하므로 rewrite-location을 true 또는 false로 설정할 수 있습니다. `authPassthroughEnabled`가 활성화된 경우 Application Server는 rewrite location의 프로토콜 부분을 적절하게 수정합니다. rewrite-location이 false로 설정된 경우 로드 밸런서는 리디렉션된 URL의 위치를 다시 쓰지 않습니다. true로 설정된 경우에는 리디렉션된 URL의 위치를 다시 씁니다. 그러나 Application Server에서 클라이언트의 HTTPS 연결을 인식했으므로 이 다시 쓰기는 필요하지 않습니다. 또한 응용 프로그램이 HTTP에서 HTTPS로 또는 HTTPS에서 HTTP로 리디렉션해야 하는 경우 rewrite-location 매개 변수를 false로 설정해야 합니다.

멱등원(Idempotent) URL 구성

멱등원(idempotent) 요청은 다시 시도할 때 응용 프로그램에 변경이나 불일치를 일으키지 않는 요청입니다. HTTP의 경우 일부 메소드(예: GET)는 멱등원(Idempotent)이지만 다른

메소드(예: POST)는 멱등원(Idempotent)이 아닙니다. 멱등원(Idempotent) URL을 재시도할 경우 값이 서버나 데이터베이스에서 변경되지 않습니다. 사용자가 수신한 응답의 변화만이 유일한 차이점입니다.

멱등원(Idempotent) 요청의 예에는 검색 엔진 쿼리와 데이터베이스 쿼리가 포함됩니다. 다시 시도할 때 데이터가 업데이트되거나 수정되지 않는 것이 기본 원칙입니다.

배포된 응용 프로그램의 가용성을 향상시키려면 로드 밸런서가 사용되는 모든 응용 프로그램 서버 인스턴스에서 실패한 멱등원(Idempotent) HTTP 요청을 다시 시도하도록 환경을 구성합니다. 예를 들어, 검색 요청을 다시 시도하려면 읽기 전용 요청에 대해 이 옵션을 사용합니다.

sun-web.xml 파일에 멱등원(Idempotent) URL을 구성합니다. 로드 밸런서 구성을 내보낼 때 멱등원(Idempotent) URL 정보가 loadbalancer.xml 파일에 자동으로 추가됩니다.

멱등원(Idempotent) URL 구성에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide**의 “Configuring Idempotent URL Requests”를 참조하십시오.

다중 웹 서버 인스턴스 구성

Sun Java System Application Server 설치 프로그램을 사용할 경우 단일 시스템에 여러 로드 밸런서 플러그인을 설치할 수 없습니다. 단일 시스템에 로드 밸런서 플러그인과 함께 복수 웹 서버를 두려면 단일 클러스터 또는 복수 클러스터든 상관없이 로드 밸런서 플러그인을 구성하기 위한 몇 가지 수동 단계가 필요합니다.

▼ 다중 웹 서버 인스턴스 구성

- 1 로드 밸런서 플러그인을 사용하도록 새 웹 서버 인스턴스를 구성합니다.

4장의 단계를 따릅니다.

- 2 DTD 파일을 복사합니다.

기존 웹 서버 인스턴스의 config 디렉토리에서 새 인스턴스의 config 디렉토리로 sun-loadbalancer_1_1.dtd를 복사합니다.

- 3 로드 밸런서 구성 파일을 설정합니다. 다음의 두 가지 방법 중 하나를 사용합니다.

- 기존 로드 밸런서 구성을 복사합니다.

기존 로드 밸런서 구성을 사용하고 기존 웹 서버 인스턴스의 config 디렉토리에서 새 인스턴스의 config 디렉토리로 loadbalancer.xml 파일을 복사합니다.

- 새 로드 밸런서 구성을 만듭니다.
 - a. `asadmin create-http-lb-config`를 사용하여 새 로드 밸런서 구성을 만듭니다.
 - b. `asadmin export http-lb-config`를 사용하여 새 구성을 `loadbalancer.xml` 파일로 내보냅니다.
 - c. 해당 `loadbalancer.xml` 파일을 새 웹 서버의 `config` 디렉토리로 복사합니다.
로드 밸런서 구성을 만든 후 `loadbalancer.xml` 파일로 내보내는 방법에 대한 자세한 내용은 [122 페이지 “HTTP 로드 밸런서 구성 만들기”](#)를 참조하십시오.

가용성 손실 없이 응용 프로그램 업그레이드

사용자에 대한 가용성 손실 없이 응용 프로그램을 새 버전으로 업그레이드하는 것을 **롤링 업그레이드**라고 합니다. 업그레이드 중에 응용 프로그램의 두 버전을 주의해서 관리하면 응용 프로그램의 현재 사용자가 중단 없이 작업을 완료하면서 새로운 사용자가 새 버전의 응용 프로그램을 투명하게 가져올 수 있습니다. 롤링 업그레이드 수행 시 사용자는 업그레이드가 발생하는지 모릅니다.

응용 프로그램 호환성

롤링 업그레이드는 두 응용 프로그램 버전 간 변경 사항의 정도에 따라 간단할 수도 있고 복잡할 수도 있습니다.

정적 텍스트나 이미지가 변경된 경우처럼 표면적인 부분만 변경되면 응용 프로그램의 두 버전은 **호환되며** 동일한 클러스터에서 실행될 수 있습니다.

호환되는 응용 프로그램은 다음과 같아야 합니다.

- 동일한 세션 정보를 사용합니다.
- 호환되는 데이터베이스 스키마를 사용합니다.
- 일반적으로 호환되는 응용 프로그램 수준 비즈니스 논리를 갖습니다.
- 동일한 물리적 데이터 소스를 사용합니다.

단일 클러스터나 다중 클러스터에서 호환되는 응용 프로그램의 롤링 업그레이드를 수행할 수 있습니다. 자세한 내용은 [135 페이지 “단일 클러스터에서 업그레이드”](#)를 참조하십시오.

응용 프로그램의 두 버전이 위의 기준을 만족하지 않으면 응용 프로그램은 **호환되지 않는** 것으로 간주됩니다. 한 클러스터에서 응용 프로그램의 호환되지 않는 버전을 실행하면 응용 프로그램 데이터가 손상되고 세션 페일오버가 제대로 작동하지 않을 수 있습니다. 이러한 문제는 비호환 상태의 유형 및 정도에 따라 다릅니다. 새 버전을 배포할 "새도우 클러스터"를 만들어 호환되지 않는 응용 프로그램을 업그레이드하고 이전 클러스터 및 응용 프로그램을 천천히 정지하는 것이 좋습니다. 자세한 내용은 [138 페이지 “호환되지 않는 응용 프로그램 업그레이드”](#)를 참조하십시오.

응용 프로그램 버전의 호환 여부는 응용 프로그램 개발자 및 관리자는 판단하는 것이 가장 좋습니다. 호환 여부가 불확실하면 안전하게 호환되지 않는 것으로 간주합니다.

단일 클러스터에서 업그레이드

클러스터 구성이 다른 클러스터와 공유되지 않을 경우, 단일 클러스터에 배포된 응용 프로그램의 롤링 업그레이드를 수행할 수 있습니다.

▼ 단일 클러스터에서 응용 프로그램 업그레이드

- 1 이전 버전의 응용 프로그램을 저장하거나 도메인을 백업합니다.
도메인을 백업하려면 `asadmin backup-domain` 명령을 사용합니다.
- 2 클러스터에 대해 동적 재구성(활성화된 경우)을 해제합니다.
관리 콘솔에서 다음 작업을 수행합니다.
 - a. 구성 노드를 확장합니다.
 - b. 클러스터의 구성 이름을 누릅니다.
 - c. 구성 시스템 등록 정보 페이지에서 동적 재구성 사용 가능 확인란을 선택 해제합니다.
 - d. 저장을 누릅니다.
 또는 다음 명령을 사용합니다.


```
asadmin set --user user --passwordfile password-file
cluster-name-config.dynamic-reconfiguration-enabled=false
```
- 3 업그레이드된 응용 프로그램을 대상 domain에 다시 배포합니다.
관리 콘솔을 사용하여 재배포할 경우 도메인이 자동으로 대상이 됩니다. `asadmin`을 사용할 경우 대상 domain을 지정합니다. 동적 재구성을 사용할 수 없기 때문에 이전 응용 프로그램은 계속해서 클러스터에서 실행됩니다.
- 4 `asadmin enable-http-lb-application`을 사용하여 인스턴스에서 재배포된 응용 프로그램을 활성화합니다.
- 5 로드 밸런서에서 클러스터에 있는 한 서버 인스턴스를 정지합니다.
다음 단계를 수행합니다.
 - a. `asadmin disable-http-lb-server`를 사용하여 서버 인스턴스를 비활성화합니다.
 - b. `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.

c. 내보낸 구성 파일을 웹 서버 인스턴스의 구성 디렉토리에 복사합니다.

예를 들어, Sun Java System Web Server의 경우 위치는 `web-server-install-dir/https-host-name /config/loadbalancer.xml`입니다. 로드 밸런서가 새 구성 파일을 로드하게 하려면 로드 밸런서 구성에서 `reloadinterval`을 설정하여 동적 재구성이 활성화되도록 합니다.

d. 시간 초과가 만료될 때까지 대기합니다.

로드 밸런서의 로그 파일을 모니터링하여 인스턴스가 오프라인인지 확인합니다. 재시도 URL을 만나면 정지 기간을 건너뛰고 서버를 즉시 다시 시작합니다.

6 클러스터의 다른 인스턴스가 계속 실행되는 동안 비활성화된 서버 인스턴스를 다시 시작합니다.

다시 시작하면 서버가 도메인과 동기화되고 응용 프로그램이 업데이트됩니다.

7 다시 시작한 서버의 응용 프로그램을 테스트하여 제대로 실행되는지 확인합니다.

8 로드 밸런서에서 서버 인스턴스를 다시 활성화합니다.

다음 단계를 수행합니다.

a. `asadmin enable-http-lb-server`를 사용하여 서버 인스턴스를 활성화합니다.

b. `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.

c. 구성 파일을 웹 서버의 구성 디렉토리에 복사합니다.

9 클러스터의 인스턴스마다 5-8단계를 반복합니다.

10 모든 서버 인스턴스에 새로운 응용 프로그램이 있고 실행 중인 경우 클러스터에 대한 동적 재구성을 활성화할 수 있습니다.

여러 클러스터에서 업그레이드

▼ 두 개 이상의 클러스터에서 호환되는 응용 프로그램 업그레이드

1 이전 버전의 응용 프로그램을 저장하거나 도메인을 백업합니다.

도메인을 백업하려면 `asadmin backup-domain` 명령을 사용합니다.

2 모든 클러스터에 대해 동적 재구성(활성화된 경우)을 해제합니다.

관리 콘솔에서 다음 작업을 수행합니다.

a. 구성 노드를 확장합니다.

- b. 한 클러스터의 구성 이름을 누릅니다.
 - c. 구성 시스템 등록 정보 페이지에서 동적 재구성 사용 가능 확인란을 선택 해제합니다.
 - d. 저장을 누릅니다.
 - e. 다른 클러스터에 대해서도 반복합니다.
- 또는 다음 명령을 사용합니다.

```
asadmin set --user user --passwordfile password-file
cluster-name-config.dynamic-reconfiguration-enabled=false
```

3 업그레이드된 응용 프로그램을 대상 domain에 다시 배포합니다.

관리 콘솔을 사용하여 재배포할 경우 도메인이 자동으로 대상이 됩니다. asadmin을 사용할 경우 대상 domain을 지정합니다. 동적 재구성이 비활성화되어 있기 때문에 이전 응용 프로그램은 계속해서 클러스터에서 실행됩니다.

4 asadmin enable-http-lb-application을 사용하여 클러스터에 재배포된 응용 프로그램을 활성화합니다.

5 로드 밸런서에서 한 클러스터를 정지합니다.

- a. asadmin disable-http-lb-server를 사용하여 클러스터를 비활성화합니다.
- b. asadmin export-http-lb-config를 사용하여 로드 밸런서 구성 파일을 내보냅니다.

c. 내보낸 구성 파일을 웹 서버 인스턴스의 구성 디렉토리에 복사합니다.

예를 들어, Sun Java System Web Server의 경우 위치는

`web-server-install-dir/https-host-name /config/loadbalancer.xml`입니다. 로드 밸런서 구성에서 reloadinterval을 설정하여 로드 밸런서에 대해 동적 재구성을 활성화해야 합니다. 이렇게 해야 새 로드 밸런서 구성 파일이 자동으로 로드됩니다.

d. 시간 초과가 만료될 때까지 대기합니다.

로드 밸런서의 로그 파일을 모니터링하여 인스턴스가 오프라인인지 확인합니다. 재시도 URL을 만나면 정지 기간을 건너뛰고 서버를 즉시 다시 시작합니다.

6 다른 클러스터를 계속 실행하면서 비활성화된 클러스터를 다시 시작합니다.

다시 시작하면 클러스터가 도메인과 동기화되고 응용 프로그램이 업데이트됩니다.

7 다시 시작한 클러스터의 응용 프로그램을 테스트하여 제대로 실행되는지 확인합니다.

- 8 로드 밸런서에서 클러스터를 활성화합니다.
 - a. `asadmin enable-http-lb-server`를 사용하여 클러스터를 활성화합니다.
 - b. `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.
 - c. 구성 파일을 웹 서버의 구성 디렉토리에 복사합니다.
- 9 다른 클러스터에 대해서도 5-8 단계를 반복합니다.
- 10 모든 서버 인스턴스에 새로운 응용 프로그램이 있고 실행 중인 경우 모든 클러스터에 대한 동적 재구성을 활성화할 수 있습니다.

호환되지 않는 응용 프로그램 업그레이드

새 버전의 응용 프로그램이 이전 버전과 호환되지 않는 경우 다음 절차를 수행합니다. 응용 프로그램을 호환되게 만드는 방법에 대한 자세한 내용은 [134 페이지 “응용 프로그램 호환성”](#)을 참조하십시오. 또한 두 개 이상의 클러스터에서 호환되지 않는 응용 프로그램을 업그레이드해야 합니다. 클러스터가 하나만 있으면 아래에 설명된 것처럼 업그레이드를 위해 “새도우 클러스터”를 만듭니다.

호환되지 않는 응용 프로그램을 업그레이드할 경우:

- 새 버전의 응용 프로그램에 이전 버전의 응용 프로그램과는 다른 이름을 지정합니다. 아래 단계에서는 응용 프로그램의 이름이 변경된다고 가정합니다.
- 데이터 스키마가 호환되지 않으면 데이터 마이그레이션을 계획한 후에 다른 물리적 데이터 소스를 사용합니다.
- 이전 버전이 배포된 클러스터와는 다른 클러스터로 새 버전을 배포합니다.
- 이전 응용 프로그램에 대한 요청이 새 클러스터로 페일오버되지 않으므로 해당 응용 프로그램이 실행되는 클러스터를 오프라인으로 만들기 전에 해당 클러스터에 대해 적절히 긴 시간 초과를 설정합니다. 이러한 사용자 세션은 실패하게 됩니다.

▼ 또 다른 클러스터를 만들어 호환되지 않는 응용 프로그램 업그레이드

- 1 이전 버전의 응용 프로그램을 저장하거나 도메인을 백업합니다.
도메인을 백업하려면 `asadmin backup-domain` 명령을 사용합니다.

- 2 기존 클러스터와 동일하거나 다른 시스템 집합에 "새도우 클러스터"를 만듭니다.또 다른 클러스터가 이미 있는 경우 이 단계를 건너뛵니다.
 - a. 관리 콘솔을 사용하여 새 클러스터를 만들고 기존 클러스터의 명명된 구성을 참조합니다.
각 시스템에서 새 인스턴스에 대한 포트를 사용자 정의하여 기존 활성 포트와 충돌하지 않도록 합니다.
 - b. 클러스터와 관련된 모든 자원에 대해 `asadmin create-resource-ref`를 사용하여 새로 만들어진 클러스터에 대한 자원 참조를 추가합니다.
 - c. `asadmin create-application-ref`를 사용하여 새로 만들어진 클러스터에서 해당 클러스터로 배포된 다른 모든 응용 프로그램(현재 재배포된 응용 프로그램 제외)에 대한 참조를 만듭니다.
 - d. `asadmin configure-ha-cluster`를 사용하여 클러스터를고가용성 클러스터로 구성합니다.
 - e. `asadmin create-http-lb-ref`를 사용하여 로드 밸런서 구성 파일에 새로 만들어진 클러스터에 대한 참조를 만듭니다.
- 3 새 버전의 응용 프로그램에 이전 버전과는 다른 이름을 지정합니다.
- 4 새 클러스터에 새 응용 프로그램을 대상으로 배포합니다. 다른 컨텍스트 루트를 사용합니다.
- 5 `asadmin enable-http-lb-application`을 사용하여 클러스터에 배포된 응용 프로그램을 활성화합니다.
- 6 다른 클러스터를 계속 실행하면서 새 클러스터를 시작합니다.
시작하면 클러스터가 도메인과 동기화되고 새 응용 프로그램으로 업데이트됩니다.
- 7 새 클러스터의 응용 프로그램을 테스트하여 제대로 실행되는지 확인합니다.
- 8 `asadmin disable-http-lb-server`를 사용하여 로드 밸런서에서 이전 클러스터를 비활성화합니다.
- 9 느린 세션이 지속되는 기간에 대한 시간 초과를 설정합니다.
- 10 `asadmin enable-http-lb-server`를 사용하여 로드 밸런서에서 새 클러스터를 활성화합니다.
- 11 `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.

12 내보낸 구성 파일을 웹 서버 인스턴스의 구성 디렉토리에 복사합니다.

예를 들어, Sun Java System Web Server의 경우 위치는 `web-server-install-dir/https-host-name /config/loadbalancer.xml`입니다. 로드 밸런서 구성에서 `reloadinterval`을 설정하여 로드 밸런서에 대해 동적 재구성을 활성화해야 합니다. 이렇게 해야 새 로드 밸런서 구성 파일이 자동으로 로드됩니다.

13 시간 초과 기간이 만료되거나 이전 응용 프로그램의 모든 사용자가 종료하면 이전 클러스터를 중지하고 이전 응용 프로그램을 삭제하십시오.

HTTP 로드 밸런서 플러그인 모니터링

- 140 페이지 “로그 메시지 구성”
- 140 페이지 “로그 메시지 유형”
- 142 페이지 “로드 밸런서 로깅 활성화”
- 143 페이지 “모니터링 메시지 이해”

로그 메시지 구성

로드 밸런서 플러그인은 웹 서버의 로깅 메커니즘을 사용하여 로그 메시지를 씁니다. Application Server의 기본 로깅 수준은 Sun Java System Web Server의 기본 로깅 수준(INFO), Apache Web Server의 기본 로깅 수준(WARN) 및 Microsoft IIS의 기본 로깅 수준(INFO)으로 설정됩니다. Application Server 로깅 수준 FINE, FINER 및 FINEST는 웹 서버의 DEBUG 수준에 매핑됩니다.

이 로그 메시지는 웹 서버 로그 파일에 기록되고, 스크립트를 사용하여 구문 분석하거나 스프레드시트로 가져와서 필수 메트릭을 계산할 수 있는 원시 데이터 형식입니다.

로그 메시지 유형

로드 밸런서 플러그인은 다음과 같은 로그 메시지 유형을 생성합니다.

- 141 페이지 “로드 밸런서 구성 프로그램 로그 메시지”
- 141 페이지 “요청 디스패치 및 런타임 로그 메시지”
- 142 페이지 “구성 프로그램 오류 메시지”

로드 밸런서 구성 프로그램 로그 메시지

멱등원(Idempotent) URL과 오류 페이지 설정을 사용할 경우 이 메시지가 기록됩니다.

멱등원(Idempotent) URL 패턴 구성 출력에는 다음 정보가 포함됩니다.

- 로그 수준을 FINE으로 설정한 경우:

```
CONFxxxx: IdempotentUrlPattern configured <url-pattern> <no-of-retries> for
web-module : <web-module>
```

- 로그 수준을 SEVERE로 설정한 경우:

```
CONFxxxx: Duplicate entry of Idempotent URL element <url-pattern> for
webModule <web-module> in loadbalancer.xml."
```

- 로그 수준을 WARN으로 설정한 경우:

```
CONFxxxx: Invalid IdempotentUrlPatternData <url-pattern> for web-module
<web-module>
```

오류 페이지 URL 구성의 출력에는 다음 정보가 포함됩니다(WARN으로 설정된 로그 수준).

```
CONFxxxx: Invalid error-url for web-module <web-module>
```

요청 디스패치 및 런타임 로그 메시지

요청을 로드 균형 조정하고 디스패치하는 동안 이 로그 메시지가 생성됩니다.

- 메소드 시작을 위한 표준 로깅 출력에는 다음 정보가 포함됩니다(FINE으로 설정된 로그 수준).

```
ROUTxxxx: Executing Router method <method_name>
```

- 메소드 시작을 위한 라우터 로그 출력에는 다음 정보가 포함됩니다(INFO로 설정된 로그 수준).

```
ROUTxxxx: Successfully Selected another ServerInstance for idempotent request
<Request-URL>
```

- 런타임 로그 출력에는 다음 정보가 포함됩니다(INFO로 설정된 로그 수준).

```
RNTMxxxx: Retrying Idempotent <GET/POST/HEAD> Request <Request-URL>
```

구성 프로그램 오류 메시지

구성 문제가 있을 경우, 예를 들어 참조하는 사용자 정의 오류 페이지가 누락된 경우 이 오류가 표시됩니다.

- INFO로 설정된 로그 수준:

ROUTxxxx: Non Idempotent Request <Request-URL> cannot be retried

예를 들면 다음과 같습니다. ROUTxxxx: Non Idempotent Request
http://sun.com/addToDB?x=11&abc=2 cannot be retried

- FINE으로 설정된 로그 수준:

RNTMxxxx: Invalid / Missing Custom error-url / page: <error-url> for
web-module: <web-module>

예를 들면 다음과 같습니다. RNTMxxxx: Invalid / Missing Custom error-url / page:
myerror1xyz for web-module: test

로드 밸런서 로깅 활성화

로드 밸런서 플러그인은 다음 정보를 기록합니다.

- 모든 요청의 요청 시작/중지 정보
- 비정상 인스턴스에서 정상 인스턴스로 요청이 페일오버된 경우 페일오버된 요청 정보
- 모든 상태 검사 주기가 끝나는 시점의 비정상 인스턴스 목록

주- 로드 밸런서에서 로깅이 활성화된 경우, 그리고 웹 서버 로그 수준을 DEBUG로 설정하거나 자세한 메시지를 인쇄할 경우 로드 밸런서는 HTTP 세션 아이디를 웹 서버 로그 파일에 기록합니다. 따라서 로드 밸런서 플러그인을 호스트하는 웹 서버가 DMZ에 있을 경우 프로덕션 환경에서 DEBUG 또는 유사한 로그 수준을 사용하지 마십시오.

DEBUG 로그 수준을 사용해야 할 경우 loadbalancer.xml에서 require-monitor-data 등록 정보를 false로 설정하여 로드 밸런서 로깅을 해제하십시오.

▼ 로드 밸런서 로깅 설정

- 1 웹 서버에 로깅 옵션을 설정합니다. 절차는 웹 서버에 따라 다릅니다.

- Sun Java System Web Server를 사용할 경우

서버의 관리 콘솔에서 Magnus Editor 탭으로 이동하여 Log Verbose 옵션을 On으로 설정합니다.

- Apache Web Server의 경우 로그 수준을 DEBUG로 설정합니다.

- Microsoft IIS의 경우 `sun-passthrough.properties` 파일에서 로그 수준을 FINE으로 설정합니다.
- 2 로드 밸런서 구성의 `monitor` 옵션을 `true`로 설정합니다.
- 로드 밸런서 구성을 처음 만들 경우 `asadmin create-http-lb-config` 명령을 사용하여 모니터링을 `true`로 설정하거나, `asadmin set` 명령을 사용하여 나중에 `true`로 설정합니다. 모니터링은 기본적으로 비활성화되어 있습니다.

모니터링 메시지 이해

로드 밸런서 플러그인 로그 메시지의 형식은 다음과 같습니다.

- HTTP 요청의 시작에는 다음 정보가 포함되어 있습니다.
`RequestStart Sticky(New) <req-id> <time-stamp> <URL>`
 타임스탬프 값은 1970년 1월 1일부터의 밀리초입니다. 예를 들면 다음과 같습니다.
`RequestStart New 123456 602983`
`http://austen.sun.com/Webapps-simple/servlet/Example1`
- HTTP 요청의 끝에는 다음과 같이 `RequestExit` 메시지가 포함됩니다.
`RequestExit Sticky(New) <req-id> <time-stamp> <URL> <listener-id>`
`<response-time> Failure-<reason for error>(incase of a failure)`
 예를 들면 다음과 같습니다.
`RequestExit New 123456 603001`
`http://austen.sun.com/Webapps-simple/servlet/Example1 http://austen:2222 18`

주 - `RequestExit` 메시지에서 *response-time*은 로드 밸런서 플러그인의 관점에서 전체 요청 반환 시간(밀리초)을 나타냅니다.

- 비정상 인스턴스 목록은 다음과 같습니다.
`UnhealthyInstances <cluster-id> <time-stamp> <listener-id>, <listener-id>...`
 예를 들면 다음과 같습니다.
`UnhealthyInstances cluster1 701923 http://austen:2210, http://austen:3010`
- 페일오버된 요청 목록은 다음과 같습니다.
`FailedoverRequest <req-id> <time-stamp> <URL> <session-id>`
`<failed-over-listener-id> <unhealthy-listener-id>`
 예를 들면 다음과 같습니다.
`FailedoverRequest 239496 705623`
`http://austen.sun.com/Apps/servlet/SessionTest 16dfdac3c7e80a40`
`http://austen:4044 http://austen:4045`

Application Server 클러스터 사용

이 장에서는 Application Server 클러스터의 사용 방법을 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- 145 페이지 “클러스터 개요”
- 145 페이지 “클러스터 작업”

클러스터 개요

클러스터는 동일한 응용 프로그램, 자원 및 구성 정보를 공유하는 명명된 서버 인스턴스 모음입니다. 여러 시스템의 서버 인스턴스를 하나의 논리적 클러스터에 그룹화한 후 하나의 단위로 관리할 수 있습니다. DAS를 사용하여 다중 시스템 클러스터의 수명 주기를 쉽게 제어할 수 있습니다.

클러스터를 구축하면 수평적인 확장, 로드 균형 조정 및 페일오버 보호가 구현됩니다. 정의에 따르면 한 클러스터의 모든 인스턴스에 동일한 자원 및 응용 프로그램 구성이 포함됩니다. 클러스터의 서버 인스턴스나 시스템에 장애가 발생하면 로드 밸런서는 장애를 감지하고, 실패한 인스턴스에서 클러스터의 다른 인스턴스로 트래픽을 리디렉션하고, 사용자 세션 상태를 복구합니다. 동일한 응용 프로그램 및 자원이 클러스터의 모든 인스턴스에 있으므로 한 인스턴스에서 클러스터의 다른 인스턴스로 페일오버가 수행될 수 있습니다.

클러스터 작업

- 146 페이지 “클러스터 만들기”
- 147 페이지 “클러스터에 대한 서버 인스턴스 만들기”
- 148 페이지 “클러스터 구성”
- 149 페이지 “클러스터링된 인스턴스 시작, 중지 및 삭제”
- 149 페이지 “클러스터의 서버 인스턴스 구성”
- 150 페이지 “클러스터에 대한 응용 프로그램 구성”

- 150 페이지 “클러스터에 대한 자원 구성”
- 151 페이지 “클러스터 삭제”
- 151 페이지 “EJB 타이머 마이그레이션”
- 152 페이지 “서비스 손실 없이 구성 요소 업그레이드”

▼ 클러스터 만들기

- 1 트리 구성 요소에서 클러스터 노드를 선택합니다.
- 2 클러스터 페이지에서 새로 만들기를 누릅니다.
클러스터 만들기 페이지가 표시됩니다.
- 3 이름 필드에서 클러스터 이름을 입력합니다.
이름은 다음과 같아야 합니다.
 - 대소문자, 숫자, 밑줄, 하이픈 및 마침표(.)로만 구성됩니다.
 - 모든 노드 에이전트 이름, 서버 인스턴스 이름, 클러스터 이름 및 구성 이름에서 고유합니다.
 - domain이 아닙니다.
- 4 구성 필드의 드롭다운 목록에서 원하는 구성을 선택합니다.
 - 공유 구성을 사용하지 않는 클러스터를 만들려면 default-config를 선택합니다.
"선택한 구성의 복사본 만들기" 라디오 버튼을 선택합니다. 기본 구성의 복사본 이름은 `cluster_name-config`입니다.
 - 공유 구성을 사용하는 클러스터를 만들려면 드롭다운 목록에서 해당 구성을 선택합니다.
"선택한 구성 참조" 라디오 버튼을 선택하여 지정된 기존 공유 구성을 사용하는 클러스터를 만듭니다.
- 5 필요에 따라 서버 인스턴스를 추가합니다.
클러스터를 만든 후에 서버 인스턴스를 추가할 수도 있습니다.

클러스터에 대한 서버 인스턴스를 만들기 전에 먼저 하나 이상의 노드 에이전트나 노드 에이전트 자리 표시자를 만듭니다. 172 페이지 “노드 에이전트 자리 표시자 만들기”를 참조하십시오.

서버 인스턴스 만들기
 - a. 만들 서버 인스턴스 영역에서 추가를 누릅니다.

- b. 인스턴스 이름 필드에서 인스턴스 이름을 입력합니다.
- c. 노드 에이전트 드롭다운 목록에서 원하는 노드 에이전트를 선택합니다.
- 6 확인을 누릅니다.
- 7 표시되는 "클러스터를 성공적으로 만들었습니다" 페이지에서 확인을 누릅니다.

자세한 정보 해당 asadmin 명령

create-cluster

- 참조
- 148 페이지 "클러스터 구성"
 - 147 페이지 "클러스터에 대한 서버 인스턴스 만들기"
 - 150 페이지 "클러스터에 대한 응용 프로그램 구성"
 - 150 페이지 "클러스터에 대한 자원 구성"
 - 151 페이지 "클러스터 삭제"
 - 152 페이지 "서비스 손실 없이 구성 요소 업그레이드"

클러스터, 서버 인스턴스 및 노드 에이전트 관리 방법에 대한 자세한 내용은 [165 페이지 "노드 에이전트 배포"](#)를 참조하십시오.

▼ 클러스터에 대한 서버 인스턴스 만들기

시작하기 전에 클러스터에 대한 서버 인스턴스를 만들려면 먼저 노드 에이전트나 노드 에이전트 자리 표시자를 만들어야 합니다. [172 페이지 "노드 에이전트 자리 표시자 만들기"](#)를 참조하십시오.

- 1 트리 구성 요소에서 클러스터 노드를 확장합니다.
- 2 원하는 클러스터 노드를 선택합니다.
- 3 인스턴스 탭을 눌러 클러스터링된 서버 인스턴스 페이지를 표시합니다.
- 4 새로 만들기를 눌러 클러스터링된 서버 인스턴스 만들기 페이지를 표시합니다.
- 5 이름 필드에서 서버 인스턴스 이름을 입력합니다.
- 6 노드 에이전트 드롭다운 목록에서 노드 에이전트를 선택합니다.
- 7 확인을 누릅니다.

자세한 정보 **해당 asadmin 명령**

create-instance

- 참조
- 163 페이지 “노드 에이전트란?”
 - 146 페이지 “클러스터 만들기”
 - 148 페이지 “클러스터 구성”
 - 150 페이지 “클러스터에 대한 응용 프로그램 구성”
 - 150 페이지 “클러스터에 대한 자원 구성”
 - 151 페이지 “클러스터 삭제”
 - 152 페이지 “서비스 손실 없이 구성 요소 업그레이드”
 - 149 페이지 “클러스터의 서버 인스턴스 구성”

▼ 클러스터 구성

1 트리 구성 요소에서 클러스터 노드를 확장합니다.

2 원하는 클러스터 노드를 선택합니다.

일반 정보 페이지에서 다음 작업을 수행할 수 있습니다.

- 인스턴스 시작을 눌러 클러스터링된 서버 인스턴스를 시작합니다.
- 인스턴스 중지를 눌러 클러스터링된 서버 인스턴스를 중지합니다.
- EJB 타이머 마이그레이션을 눌러 EJB 타이머를 중지된 서버 인스턴스에서 클러스터 내 다른 서버 인스턴스로 마이그레이션합니다.

자세한 정보 **해당 asadmin 명령**

start-cluster, stop-cluster, migrate-timers

- 참조
- 146 페이지 “클러스터 만들기”
 - 147 페이지 “클러스터에 대한 서버 인스턴스 만들기”
 - 150 페이지 “클러스터에 대한 응용 프로그램 구성”
 - 150 페이지 “클러스터에 대한 자원 구성”
 - 151 페이지 “클러스터 삭제”
 - 152 페이지 “서비스 손실 없이 구성 요소 업그레이드”
 - 151 페이지 “EJB 타이머 마이그레이션”

▼ 클러스터링된 인스턴스 시작, 중지 및 삭제

- 1 트리 구성 요소에서 클러스터 노드를 확장합니다.
- 2 서버 인스턴스를 포함하는 클러스터의 노드를 확장합니다.
- 3 인스턴스 탭을 눌러 클러스터링된 서버 인스턴스 페이지를 표시합니다.
이 페이지에서 다음을 수행할 수 있습니다.
 - 인스턴스에 대한 확인란을 선택하고 삭제, 시작 또는 중지를 눌러 지정된 모든 서버 인스턴스에 대해 선택한 작업을 수행합니다.
 - 인스턴스 이름을 눌러 일반 정보 페이지를 표시합니다.

▼ 클러스터의 서버 인스턴스 구성

- 1 트리 구성 요소에서 클러스터 노드를 확장합니다.
- 2 서버 인스턴스를 포함하는 클러스터의 노드를 확장합니다.
- 3 서버 인스턴스 노드를 선택합니다.
- 4 일반 정보 페이지에서 다음을 수행할 수 있습니다.
 - 인스턴스 시작을 눌러 인스턴스를 시작합니다.
 - 인스턴스 중지를 눌러 실행 중인 인스턴스를 중지합니다.
 - JNDI 찾아보기를 눌러 실행 중인 인스턴스의 JNDI 트리를 검색합니다.
 - 로그 파일 보기를 눌러 서버 로그 뷰어를 엽니다.
 - 로그 파일 회전을 눌러 인스턴스의 로그 파일을 회전합니다. 이 작업을 수행하면 회전할 로그 파일이 예약됩니다. 다음 번에 로그 파일에 항목이 기록될 때 실제 회전이 발생합니다.
 - 트랜잭션 복구를 눌러 불완전한 트랜잭션을 복구합니다.
 - 등록 정보 탭을 눌러 인스턴스의 포트 번호를 수정합니다.
 - 모니터 탭을 눌러 모니터링 등록 정보를 변경합니다.

- 참조
- [146 페이지 “클러스터 만들기”](#)
 - [148 페이지 “클러스터 구성”](#)
 - [147 페이지 “클러스터에 대한 서버 인스턴스 만들기”](#)
 - [150 페이지 “클러스터에 대한 응용 프로그램 구성”](#)

- 150 페이지 “클러스터에 대한 자원 구성”
- 151 페이지 “클러스터 삭제”
- 152 페이지 “서비스 손실 없이 구성 요소 업그레이드”
- Sun Java System Application Server Enterprise Edition 8.2 관리 설명서의 “트랜잭션 복구”

▼ 클러스터에 대한 응용 프로그램 구성

- 1 트리 구성 요소에서 클러스터 노드를 확장합니다.
- 2 원하는 클러스터 노드를 선택합니다.
- 3 응용 프로그램 탭을 눌러 응용 프로그램 페이지를 표시합니다.
이 페이지에서 다음을 수행할 수 있습니다.
 - 배포 드롭다운 목록에서 배포할 응용 프로그램 유형을 선택합니다. 표시되는 배포 페이지에서 응용 프로그램을 지정합니다.
 - 필터 드롭다운 목록에서 목록에 표시할 응용 프로그램 유형을 선택합니다.
 - 응용 프로그램을 편집하려면 응용 프로그램 이름을 누릅니다.
 - 응용 프로그램 옆에 있는 확인란을 선택하고 사용 가능이나 비활성화를 선택하여 클러스터에 대해 해당 응용 프로그램을 활성화하거나 비활성화합니다.

- 참조
- 146 페이지 “클러스터 만들기”
 - 148 페이지 “클러스터 구성”
 - 147 페이지 “클러스터에 대한 서버 인스턴스 만들기”
 - 150 페이지 “클러스터에 대한 자원 구성”
 - 151 페이지 “클러스터 삭제”
 - 152 페이지 “서비스 손실 없이 구성 요소 업그레이드”

▼ 클러스터에 대한 자원 구성

- 1 트리 구성 요소에서 클러스터 노드를 확장합니다.
- 2 원하는 클러스터 노드를 선택합니다.
- 3 자원 탭을 눌러 자원 페이지를 표시합니다.
이 페이지에서 다음을 수행할 수 있습니다.
 - 클러스터에 대한 새 자원 만들기: 새로 만들기 드롭다운 목록에서 만들 자원 유형을 선택합니다. 자원을 만들 때 클러스터를 대상으로 지정해야 합니다.

- 전역으로 자원 활성화 또는 비활성화: 자원 옆의 확인란을 선택하고 활성화 또는 비활성화를 누릅니다. 이렇게 해도 자원이 제거되지 않습니다.
- 특정 유형의 자원만 표시: 필터 드롭다운 목록에서 목록에 표시할 자원 유형을 선택합니다.
- 자원 편집: 자원 이름을 누릅니다.

- 참조
- 146 페이지 “클러스터 만들기”
 - 148 페이지 “클러스터 구성”
 - 147 페이지 “클러스터에 대한 서버 인스턴스 만들기”
 - 150 페이지 “클러스터에 대한 응용 프로그램 구성”
 - 151 페이지 “클러스터 삭제”

▼ 클러스터 삭제

- 1 트리 구성 요소에서 클러스터 노드를 선택합니다.
- 2 클러스터 페이지에서 클러스터 이름 옆의 확인란을 선택합니다.
- 3 삭제를 누릅니다.

자세한 정보 **해당 asadmin 명령**

`delete-cluster`

- 참조
- 146 페이지 “클러스터 만들기”
 - 148 페이지 “클러스터 구성”
 - 147 페이지 “클러스터에 대한 서버 인스턴스 만들기”
 - 150 페이지 “클러스터에 대한 응용 프로그램 구성”
 - 150 페이지 “클러스터에 대한 자원 구성”
 - 152 페이지 “서비스 손실 없이 구성 요소 업그레이드”

▼ EJB 타이머 마이그레이션

서버 인스턴스가 비정상적으로 또는 예기치 않게 중단되었을 경우 해당 서버 인스턴스에 설치된 EJB 타이머를 클러스터 내 실행 중인 다른 서버 인스턴스로 이동해야 할 수도 있습니다. 그렇게 하려면 다음 단계를 수행합니다.

- 1 트리 구성 요소에서 클러스터 노드를 확장합니다.
- 2 원하는 클러스터 노드를 선택합니다.

- 3 일반 정보 페이지에서 EJB 타이머 마이그레이션을 누릅니다.
- 4 EJB 타이머 마이그레이션 페이지에서 다음을 수행합니다.
 - a. 소스 드롭다운 목록에서 타이머를 마이그레이션할 중지된 서버 인스턴스를 선택합니다.
 - b. (옵션) 대상 드롭다운 목록에서 타이머를 마이그레이션할 실행 중인 서버 인스턴스를 선택합니다.
이 필드를 비워 두면 실행 중인 서버 인스턴스가 무작위로 선택됩니다.
 - c. 확인을 누릅니다.
- 5 대상 서버 인스턴스를 중지했다가 다시 시작합니다.
소스 서버 인스턴스가 실행 중이거나 대상 서버 인스턴스가 실행되고 있지 않으면 관리 콘솔에 오류 메시지가 표시됩니다.

자세한 정보 해당 asadmin 명령

migrate-timers

- 참조
- 148 페이지 “클러스터 구성”
 - Sun Java System Application Server Enterprise Edition 8.2 관리 설명서의 “EJB 타이머 서비스 설정 구성”

▼ 서비스 손실 없이 구성 요소 업그레이드

로드 밸런서와 여러 클러스터를 사용하여 서비스 손실 없이 Application Server 내에서 구성 요소를 업그레이드할 수 있습니다. HADB(High-Availability Database)의 스키마를 변경한 경우에는 이 방법을 사용할 수 없습니다. 자세한 내용은 3 장을 참조하십시오.

예를 들어, 구성 요소로는 JVM, Application Server 또는 웹 응용 프로그램 등이 있습니다. 응용 프로그램 업그레이드에 대한 자세한 내용은 134 페이지 “가용성 손실 없이 응용 프로그램 업그레이드”를 참조하십시오.



주의 - 클러스터의 모든 서버 인스턴스를 함께 업그레이드합니다. 그렇지 않으면 인스턴스에 실행 중인 다른 버전의 구성 요소가 있을 경우 서버간에 세션이 페일오버됨으로써 버전 불일치가 발생할 위험이 있습니다.

- 1 클러스터의 일반 정보 페이지에 있는 클러스터 중지 버튼을 사용하여 클러스터 중 하나를 중지합니다.

- 2 해당 클러스터의 구성 요소를 업그레이드합니다.
- 3 클러스터의 일반 정보 페이지에 있는 클러스터 시작 버튼을 사용하여 클러스터를 시작합니다.
- 4 다른 클러스터에 대해서도 차례로 과정을 반복합니다.

한 클러스터 내 세션이 다른 클러스터 내 세션으로 페일오버되지 않기 때문에 한 가지 버전의 구성 요소를 실행 중인 서버 인스턴스에서 다른 버전의 구성 요소를 실행 중인 다른 클러스터 내 서버 인스턴스로 세션이 페일오버됨으로 버전이 불일치할 위험은 없습니다. 따라서 클러스터는 클러스터 내 서버 인스턴스의 세션 페일오버에 대한 안전한 경계 역할을 합니다.

- 참조
- 146 페이지 “클러스터 만들기”
 - 148 페이지 “클러스터 구성”
 - 147 페이지 “클러스터에 대한 서버 인스턴스 만들기”
 - 150 페이지 “클러스터에 대한 응용 프로그램 구성”
 - 150 페이지 “클러스터에 대한 자원 구성”
 - 151 페이지 “클러스터 삭제”

구성 관리

이 장에서는 Application Server의 명명된 서버 구성 추가, 변경 및 사용에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- 155 페이지 “구성 사용”
- 158 페이지 “명명된 구성 작업”

구성 사용

- 155 페이지 “구성”
- 156 페이지 “default-config 구성”
- 156 페이지 “인스턴스 또는 클러스터에 대한 구성”
- 157 페이지 “고유한 포트 번호 및 구성”

구성

구성은 HTTP 수신기, ORB/IIOP 수신기, JMS 브로커, EJB 컨테이너, 보안, 로깅 및 모니터링 등에 대한 설정을 포함하는 서버 구성 정보 집합입니다. 명명된 구성에는 응용 프로그램과 자원이 정의되지 않습니다.

구성은 관리 도메인에 존재합니다. 같은 도메인의 클러스터나 여러 서버 인스턴스는 동일한 구성을 참조할 수도 있고 별도의 구성을 가질 수도 있습니다.

클러스터의 경우 같은 클러스터의 모든 서버 인스턴스는 클러스터의 구성을 상속하므로 같은 클러스터에 있는 인스턴스에는 같은 환경이 보장됩니다.

구성에는 많은 필수 설정이 포함되기 때문에 기존의 명명된 구성을 복사하여 새로운 구성을 작성하는 것이 좋습니다. 새로 작성된 구성은 해당 구성 설정을 변경할 때까지 복사한 구성과 동일합니다.

클러스터나 인스턴스는 다음의 세 가지 방식으로 구성을 사용합니다.

- **독립 실행형:** 독립 실행형 서버 인스턴스나 클러스터는 구성을 다른 서버 인스턴스나 클러스터와 공유하지 않습니다. 즉, 그 명명된 구성을 참조하는 다른 서버 인스턴스나 클러스터가 없습니다. 기존 구성을 복사하고 이름을 변경하여 독립 실행형 인스턴스나 클러스터를 만듭니다.
- **공유:** 공유 서버 인스턴스나 클러스터는 다른 서버 인스턴스나 클러스터와 구성을 공유합니다. 즉, 동일한 명명된 구성을 참조하는 인스턴스나 클러스터가 여러 개 있습니다. 기존 구성을 복사하지 않고 참조하여 공유 서버 인스턴스나 클러스터를 만듭니다.
- **클러스터링:** 클러스터링된 서버 인스턴스는 클러스터의 구성을 상속합니다.
참고 항목:
 - [156 페이지 “default-config 구성”](#)
 - [156 페이지 “인스턴스 또는 클러스터에 대한 구성”](#)
 - [157 페이지 “고유한 포트 번호 및 구성”](#)
 - [158 페이지 “명명된 구성 만들기”](#)
 - [158 페이지 “명명된 구성의 등록 정보 편집”](#)

default-config 구성

default-config 구성은 독립 실행형 서버 인스턴스나 독립 실행형 클러스터 구성을 만들기 위한 템플릿의 역할을 하는 특수 구성입니다. 클러스터 및 개별 서버 인스턴스는 default-config를 참조할 수 없습니다. 이 구성은 새 구성을 만들기 위해 복사만 할 수 있습니다. 복사한 새로운 구성에 올바른 초기 설정이 있도록 기본 구성을 편집합니다.

자세한 내용은 다음을 참조하십시오.

- [156 페이지 “인스턴스 또는 클러스터에 대한 구성”](#)
- [155 페이지 “구성”](#)
- [158 페이지 “명명된 구성 만들기”](#)
- [158 페이지 “명명된 구성의 등록 정보 편집”](#)
- [160 페이지 “구성을 참조하는 인스턴스의 포트 번호 편집 방법”](#)

인스턴스 또는 클러스터에 대한 구성

새로운 서버 인스턴스나 새로운 클러스터를 작성할 경우 다음 중 하나를 수행합니다.

- 기존 구성을 참조합니다. 새로운 구성이 추가되지 않습니다.
- 기존 구성을 복사합니다. 서버 인스턴스나 클러스터를 추가하면 새로운 구성이 추가됩니다.

기본적으로 새로운 클러스터나 인스턴스는 `default-config` 구성에서 복사한 구성으로 만들어집니다. 다른 구성에서 복사하려면 새로운 인스턴스나 클러스터를 작성할 때 지정하십시오.

서버 인스턴스의 경우 새 구성의 이름이 `instance_name-config`로 지정됩니다. 클러스터의 경우 새 구성의 이름은 `cluster-name-config`로 지정됩니다.

자세한 내용은 다음을 참조하십시오.

- 156 페이지 “default-config 구성”
- 155 페이지 “구성”
- 158 페이지 “명명된 구성 만들기”
- 158 페이지 “명명된 구성의 등록 정보 편집”

클러스터링된 구성 동기화

클러스터링된 구성을 만들 경우 Application Server는 도메인 관리 서버의 `domain-root/domain-dir/config/cluster-config`에서 클러스터 구성 디렉토리를 만듭니다. 이 디렉토리는 클러스터의 모든 인스턴스에 대한 구성을 동기화하는 데 사용됩니다.

고유한 포트 번호 및 구성

동일한 호스트 시스템에 있는 여러 인스턴스가 동일한 구성을 참조하면 각 인스턴스는 고유한 포트 번호를 수신해야 합니다. 예를 들어, 두 개의 서버 인스턴스가 포트 80의 HTTP Listener를 사용하여 명명된 구성을 참조할 경우 포트 충돌로 인해 서버 인스턴스 중 하나를 시작할 수 없습니다. 개별 서버 인스턴스가 수신하는 포트 번호를 정의하는 등록 정보를 고유한 포트 번호를 사용하도록 변경합니다.

포트 번호에는 다음 원칙이 적용됩니다.

- 개별 서버 인스턴스의 포트 번호는 처음에 구성으로부터 상속됩니다.
- 서버 인스턴스를 작성할 때 포트가 이미 사용 중이면 포트 충돌을 방지하기 위해 상속된 기본값을 인스턴스 수준에서 대체합니다.
- 인스턴스에서 구성을 공유하는 것으로 가정합니다. 구성에 포트 번호 n 이 있습니다. 동일한 구성을 사용하는 시스템에서 새로운 인스턴스를 만들 경우 새로운 인스턴스에는 포트 번호 $n+1$ 이 지정됩니다(사용 가능한 경우). 이 포트 번호를 사용할 수 없으면 $n+1$ 다음의 사용 가능한 포트가 선택됩니다.
- 구성의 포트 번호를 변경하면 그 포트 번호를 상속하는 서버 인스턴스는 변경된 포트 번호를 자동으로 상속합니다.
- 인스턴스의 포트 번호를 변경하고 계속해서 구성의 포트 번호를 변경하면 인스턴스의 포트 번호는 바뀌지 않은 채 남아 있습니다.

자세한 내용은 다음을 참조하십시오.

- 160 페이지 “구성을 참조하는 인스턴스의 포트 번호 편집 방법”

- 158 페이지 “명명된 구성의 등록 정보 편집”
- 155 페이지 “구성”

명명된 구성 작업

- 158 페이지 “명명된 구성 만들기”
- 158 페이지 “명명된 구성의 등록 정보 편집”
- 160 페이지 “구성을 참조하는 인스턴스의 포트 번호 편집 방법”
- 160 페이지 “명명된 구성의 대상 보기”
- 161 페이지 “명명된 구성 삭제”

▼ 명명된 구성 만들기

- 1 트리 구성 요소에서 구성 노드를 선택합니다.
- 2 구성 페이지에서 새로 만들기를 누릅니다.
- 3 구성 만들기 페이지에서 구성의 고유한 이름을 입력합니다.
- 4 복사할 구성을 선택합니다.

default-config 구성이 독립 실행형 인스턴스나 독립 실행형 클러스터를 만들 때 사용되는 기본 구성입니다.

자세한 정보 해당 asadmin 명령

copy-config

- 참조
- 155 페이지 “구성”
 - 156 페이지 “default-config 구성”
 - 158 페이지 “명명된 구성의 등록 정보 편집”
 - 160 페이지 “구성을 참조하는 인스턴스의 포트 번호 편집 방법”
 - 160 페이지 “명명된 구성의 대상 보기”
 - 161 페이지 “명명된 구성 삭제”

명명된 구성의 등록 정보 편집

다음 표에서는 구성에 대해 미리 정의된 등록 정보를 나타냅니다.

미리 정의된 등록 정보는 포트 번호입니다. 유효한 포트 값은 1-65535입니다. UNIX의 경우 포트 1-1024에서 수신하는 소켓을 만들려면 슈퍼유저 권한이 있어야 합니다. 시스템에 서버 인스턴스가 여러 개 있을 경우 포트 번호는 고유해야 합니다.

등록 정보 이름	설명
HTTP_LISTENER_PORT	http-listener-1에 대한 포트 번호
HTTP_SSL_LISTENER_PORT	http-listener-2에 대한 포트 번호
IIOP_SSL_LISTENER_PORT	IIOP Listener SSL이 수신하는 IIOP 연결에 대한 ORB Listener 포트
IIOP_LISTENER_PORT	orb-listener-1이 수신하는 IIOP 연결에 대한 ORB Listener 포트
JMX_SYSTEM_CONNECTOR_PORT	JMX 커넥터가 수신하는 포트 번호
IIOP_SSL_MUTUALAUTH_PORT	IIOP Listener SSL_MUTUALAUTH가 수신하는 IIOP 연결에 대한 ORB Listener 포트

▼ 명명된 구성의 등록 정보 편집

- 1 트리 구성 요소에서 구성 노드를 확장합니다.
- 2 명명된 구성의 노드를 선택합니다.
- 3 구성 시스템 등록 정보 페이지에서 동적 재구성을 사용 가능하게 할 것인지 여부를 선택합니다.
사용 가능하게 하면 서버를 다시 시작하지 않고도 구성에 대한 변경 사항이 서버 인스턴스에 적용됩니다.
- 4 필요에 맞게 등록 정보를 추가, 삭제 또는 수정합니다.
- 5 구성과 연관된 모든 인스턴스에 대한 등록 정보의 현재 값을 편집하려면 인스턴스 값을 누릅니다.

자세한 정보 해당 asadmin 명령

set

- 참조
- 155 페이지 “구성”
 - 158 페이지 “명명된 구성 만들기”
 - 160 페이지 “명명된 구성의 대상 보기”
 - 161 페이지 “명명된 구성 삭제”

▼ 구성을 참조하는 인스턴스의 포트 번호 편집 방법

명명된 구성을 참조하는 모든 인스턴스마다 우선 그 구성에서 포트 번호를 상속합니다. 시스템에서 포트 번호가 고유해야 하기 때문에 상속된 포트 번호를 대체해야 할 수도 있습니다.

- 1 트리 구성 요소에서 구성 노드를 확장합니다.
- 2 명명된 구성의 노드를 선택합니다.
관리 콘솔에는 구성 시스템 등록 정보 페이지가 표시됩니다.
- 3 편집하려는 인스턴스 변수 옆의 인스턴스 값을 누릅니다.
예를 들어, HTTP-LISTENER-PORT 인스턴스 변수 옆의 인스턴스 값을 누르면 해당 구성을 참조하는 모든 서버 인스턴스에 대한 HTTP-LISTENER-PORT 값이 표시됩니다.
- 4 필요에 맞게 값을 변경하고 저장을 누릅니다.

자세한 정보 해당 asadmin 명령

set

- 참조
- 157 페이지 “고유한 포트 번호 및 구성”
 - 155 페이지 “구성”
 - 158 페이지 “명명된 구성의 등록 정보 편집”

▼ 명명된 구성의 대상 보기

구성 시스템 등록 정보 페이지에 그 구성을 사용하는 모든 대상 목록이 표시됩니다. 클러스터 구성의 경우 대상은 클러스터입니다. 인스턴스 구성의 경우 대상은 인스턴스입니다.

- 1 트리 구성 요소에서 구성 노드를 확장합니다.
- 2 명명된 구성의 노드를 선택합니다.

- 참조
- 157 페이지 “고유한 포트 번호 및 구성”
 - 155 페이지 “구성”
 - 158 페이지 “명명된 구성 만들기”
 - 158 페이지 “명명된 구성의 등록 정보 편집”
 - 161 페이지 “명명된 구성 삭제”

▼ 명명된 구성 삭제

- 1 트리 구성 요소에서 구성 노드를 선택합니다.
- 2 구성 페이지에서 삭제할 명명된 구성의 확인란을 선택합니다.
default-config 구성은 삭제할 수 없습니다.
- 3 삭제를 누릅니다.

자세한 정보 **해당 asadmin 명령**

delete-config

- 참조
- [155 페이지 “구성”](#)
 - [158 페이지 “명명된 구성 만들기”](#)
 - [158 페이지 “명명된 구성의 등록 정보 편집”](#)
 - [160 페이지 “명명된 구성의 대상 보기”](#)

노드 에이전트 구성

이 장에서는 Application Server의 노드 에이전트에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

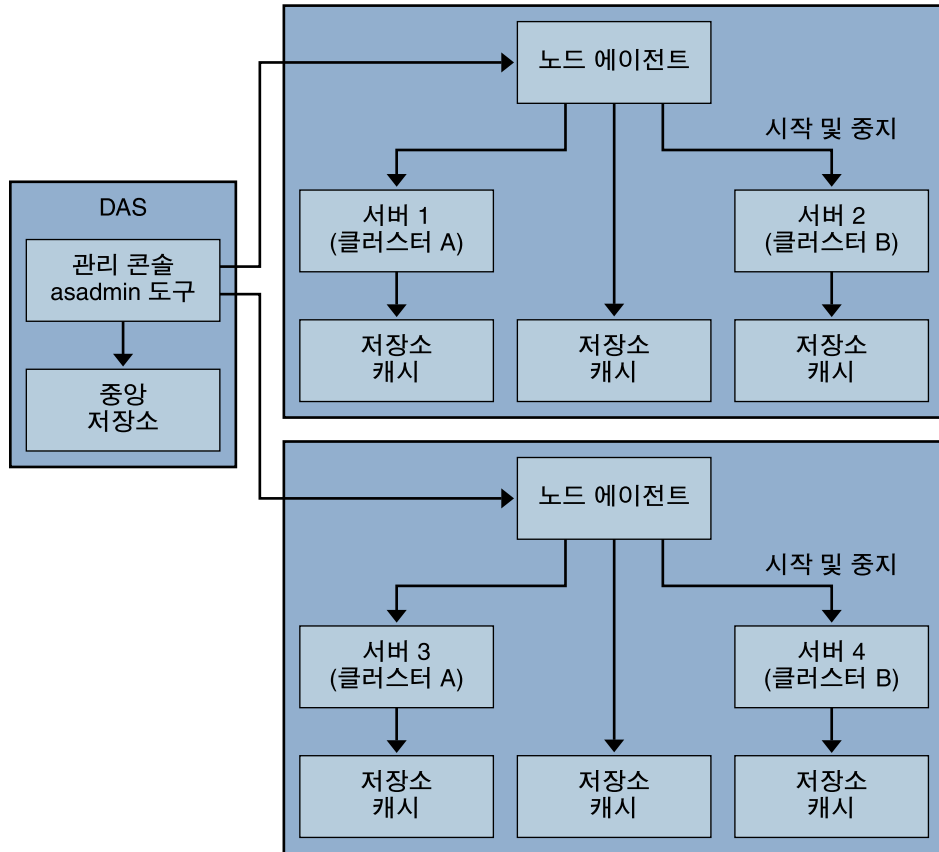
- 163 페이지 “노드 에이전트란?”
- 165 페이지 “노드 에이전트 배포”
- 167 페이지 “노드 에이전트 및 도메인 관리 서버 동기화”
- 171 페이지 “노드 에이전트 로그 보기”
- 171 페이지 “노드 에이전트 작업”

노드 에이전트란?

노드 에이전트는 DAS(Domain Administration Server)를 호스트하는 시스템을 비롯한 서버 인스턴스를 호스트하는 모든 시스템에 필요한 경량 프로세스입니다. 노드 에이전트는 다음과 같습니다.

- DAS(Domain Administration Server)에서 지시하는 대로 서버 인스턴스를 시작, 중지, 작성 및 삭제합니다.
- 실패한 서버 인스턴스를 다시 시작합니다.
- 실패한 서버의 로그 파일 보기를 제공합니다.
- 각 서버 인스턴스의 로컬 구성 저장소를 DAS의 중앙 저장소와 동기화합니다. 각 로컬 저장소에는 해당 서버 인스턴스나 노드 에이전트에 관련된 정보만 포함되어 있습니다.

다음 그림에서는 전반적인 노드 에이전트 구조를 설명합니다.



Application Server를 설치하면 기본적으로 시스템의 호스트 이름으로 노드 에이전트가 만들어집니다. 이 노드 에이전트는 로컬 시스템에서 수동으로 시작해야 작동합니다.

노드 에이전트를 실행하지 않더라도 서버 인스턴스를 만들고 삭제할 수 있습니다. 그러나, 노드 에이전트를 실행해야 노드 에이전트를 사용하여 서버 인스턴스를 시작 및 중지할 수 있습니다.

노드 에이전트 하나는 한 도메인만 지원합니다. 한 시스템이 여러 도메인에서 실행하는 인스턴스를 호스트할 경우 복수 노드 에이전트를 실행해야 합니다.

노드 에이전트 배포

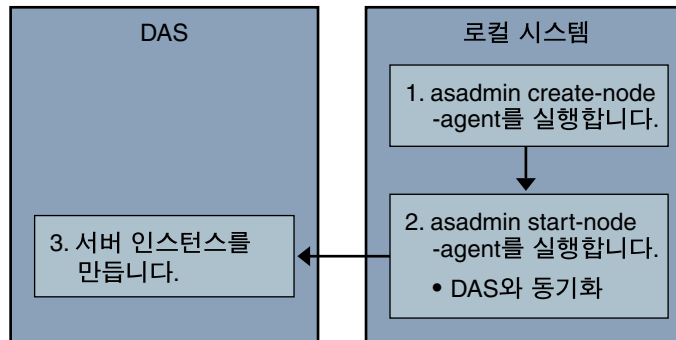
다음의 두 가지 방법으로 노드 에이전트를 구성하고 배포할 수 있습니다.

- **온라인 배포** - 토폴로지를 알고 있으며 도메인에 맞는 하드웨어가 이미 준비된 경우
- **오프라인 배포** - 전체 환경을 설정하기 전에 도메인 및 서버 인스턴스를 구성할 경우

▼ 노드 에이전트를 온라인으로 배포

도메인 토폴로지를 이미 알고 있고 도메인에 맞는 하드웨어가 준비된 경우 온라인 배포를 사용합니다.

다음 그림에서는 노드 에이전트의 온라인 배포를 요약하여 설명합니다.



시작하기 전에 Domain Administration Server를 설치 및 시작합니다. DAS를 실행하고 온라인 또는 오프라인 배포를 시작합니다.

1 서버 인스턴스를 호스트할 모든 시스템에 노드 에이전트를 설치합니다.

설치 프로그램 또는 `asadmin create-node-agent` 명령을 사용합니다. 시스템에 두 개 이상의 노드 에이전트가 필요한 경우 `asadmin create-node-agent` 명령을 사용하여 노드 에이전트를 만듭니다.

자세한 내용은 [173 페이지 “노드 에이전트 만들기”](#)를 참조하십시오.

2 `asadmin start-node-agent` 명령을 사용하여 노드 에이전트를 시작합니다.

노드 에이전트를 시작하면 노드 에이전트는 DAS(Domain Administration Server)와 통신합니다. 노드 에이전트가 DAS에 연결되면 노드 에이전트의 구성이 DAS에 만들어집니다. 구성이 존재하면 관리 콘솔에서 그 노드 에이전트를 볼 수 있습니다.

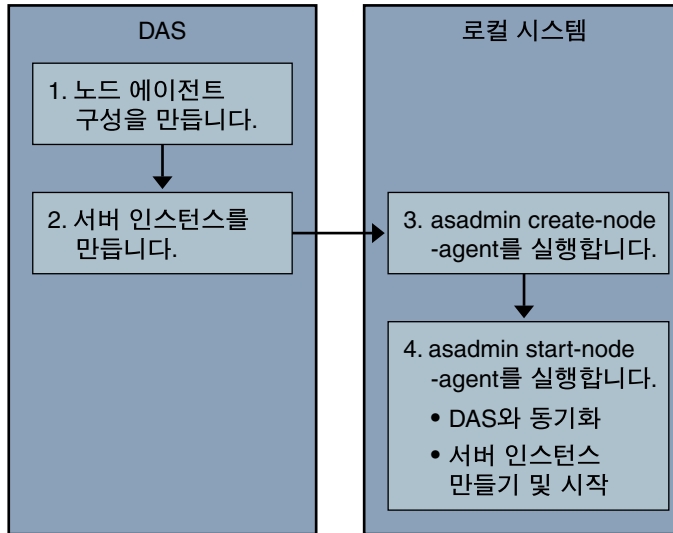
자세한 내용은 [174 페이지 “노드 에이전트 시작”](#)을 참조하십시오.

3 도메인 구성: 서버 인스턴스와 클러스터를 만들고 응용 프로그램을 배포합니다.

▼ 노드 에이전트를 오프라인으로 배포

개별 로컬 시스템을 구성하기 전에 도메인에 노드 에이전트를 배포하려면 오프라인 배포를 사용합니다.

다음 그림에서는 오프라인 배포를 요약하여 설명합니다.



시작하기 전에 Domain Administration Server를 설치 및 시작합니다. DAS를 실행하고 온라인 또는 오프라인 배포를 시작합니다.

1 DAS에 자리 표시자 노드 에이전트를 만듭니다.

자세한 내용은 [172 페이지 “노드 에이전트 자리 표시자 만들기”](#)를 참조하십시오.

2 서버 인스턴스와 클러스터를 만들고 응용 프로그램을 배포합니다.

서버 인스턴스를 만들 때는 아직 사용되고 있지 않은 포트 번호를 지정해야 합니다. 구성이 오프라인 상태에서 수행되므로 도메인은 작성 시에 포트 충돌을 확인할 수 없습니다.

3 서버 인스턴스를 호스트할 모든 시스템에 노드 에이전트를 설치합니다.

설치 프로그램 또는 `asadmin create-node-agent` 명령을 사용합니다. 노드 에이전트의 이름은 이전에 만든 자리 표시자 노드 에이전트와 동일해야 합니다.

자세한 내용은 [173 페이지 “노드 에이전트 만들기”](#)를 참조하십시오.

4 asadmin start-node-agent 명령을 사용하여 노드 에이전트를 시작합니다.

노드 에이전트가 시작되면 DAS에 바인딩되고 이전에 연관된 모든 서버 인스턴스를 만듭니다.

자세한 내용은 174 페이지 “노드 에이전트 시작”을 참조하십시오.

노드 에이전트 및 도메인 관리 서버 동기화

구성 데이터가 Domain Administration Server의 저장소(중앙 저장소)에 저장되고 노드 에이전트의 로컬 시스템에도 캐시되므로 두 위치를 동기화해야 합니다. 캐시 동기화는 항상 관리 도구를 통한 명시적인 사용자 작업에서 수행됩니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 167 페이지 “노드 에이전트 동기화”
- 168 페이지 “서버 인스턴스 동기화”
- 169 페이지 “라이브러리 파일 동기화”
- 169 페이지 “고유한 설정 및 구성 관리”
- 170 페이지 “대용량 응용 프로그램 동기화”

노드 에이전트 동기화

노드 에이전트를 처음 시작하면 중앙 저장소의 최신 정보에 대한 요청을 DAS에 전송합니다. DAS에 연결하여 구성 정보를 가져오면 노드 에이전트는 그 DAS에 바인딩됩니다.

주 - `asadmin start-node-agent` 명령은 DAS와 동기화되지 않고 원격 서버 인스턴스를 자동으로 시작합니다. `asadmin start-instance` 명령 다음에 `--startinstances=false` 옵션을 사용하여 DAS가 관리하는 중앙 저장소와 동기화되는 원격 서버 인스턴스를 시작합니다.

DAS에 자리 표시자 노드 에이전트를 만든 경우 노드 에이전트를 처음 시작하면 DAS의 중앙 저장소에서 구성을 가져옵니다. 초기 시작 중에 DAS가 실행되지 않아서 노드 에이전트가 DAS에 연결할 수 없는 경우 노드 에이전트가 중지되고 바인딩되지 않습니다.

도메인에서 노드 에이전트 구성을 변경하면 노드 에이전트를 실행 중일 때 로컬 시스템의 노드 에이전트와 자동으로 통신합니다.

DAS에서 노드 에이전트 구성을 삭제하면 다음에 노드 에이전트가 동기화할 때 스스로 중지하고 삭제 대기로 표시합니다. 로컬 `asadmin delete-node-agent` 명령을 사용하여 수동으로 삭제합니다.

서버 인스턴스 동기화

관리 콘솔이나 `asadmin` 도구를 사용하여 서버 인스턴스를 명시적으로 시작하면 서버 인스턴스가 중앙 저장소와 동기화됩니다. 이 동기화가 실패하면 서버 인스턴스가 시작되지 않습니다.

관리 콘솔이나 `asadmin` 도구를 통한 명시적인 요청 없이 노드 에이전트가 서버 인스턴스를 시작하면 서버 인스턴의 저장소 캐시가 동기화되지 않습니다. 서버 인스턴스는 해당 캐시에 저장된 구성으로 실행됩니다. 원격 서버 인스턴스의 캐시에서 파일을 추가하거나 제거해서는 안 됩니다.

원격 서버 인스턴스의 구성은 캐시(`nodeagents/na1/server1` 아래의 모든 파일)로 처리되며 **Application Server**가 소유합니다. 사용자가 원격 서버 인스턴스의 모든 파일을 제거하고 노드 에이전트를 다시 시작하는 극단적인 경우에는 원격 서버 인스턴스(예: `server1`)가 다시 만들어지고 필요한 모든 파일이 동기화됩니다.

다음 파일과 디렉토리는 **Application Server**에 의해 동기화된 상태로 유지됩니다.

표 8-1 원격 서버 인스턴스 사이에서 동기화되는 파일과 디렉토리

파일 또는 디렉토리	설명
<code>applications</code>	배포된 모든 응용 프로그램입니다. 이 디렉토리 및 하위 디렉토리에서 동기화되는 부분은 서버 인스턴스에서 참조되는 응용 프로그램에 따라 다릅니다. 노드 에이전트는 응용 프로그램을 참조하지 않기 때문에 어떠한 응용 프로그램도 동기화하지 않습니다.
<code>config</code>	전체 도메인에 대한 구성 파일을 포함합니다. <code>admch</code> , <code>admsn</code> , <code>secure.seed</code> , <code>timestamp</code> 및 <code>__timer_service_shutdown__.dat</code> 등과 같은 런타임 임시 파일을 제외한 이 디렉토리의 모든 파일이 동기화됩니다.
<code>config/config_name</code>	<code>config_name</code> 이라는 구성을 사용하여 모든 인스턴스에서 공유할 파일을 저장하는 디렉토리입니다. 이러한 디렉토리는 <code>domain.xml</code> 에 정의된 모든 구성에 대해 하나만 존재합니다. 이 디렉토리의 모든 파일은 <code>config_name</code> 을 사용하는 서버 인스턴스에 동기화됩니다.
<code>config/config_name/lib/ext</code>	Java 확장 클래스(예: <code>zip</code> 또는 <code>jar</code> 아카이브)를 드롭할 수 있는 폴더입니다. <code>config_name</code> 이라는 구성을 사용하여 서버 인스턴스에 배포된 응용 프로그램에 사용됩니다. 이러한 <code>jar</code> 파일은 Java 확장 메커니즘을 사용하여 로드됩니다.
<code>docroot</code>	HTTP 문서 루트입니다. 기본 구성에서 도메인의 모든 서버 인스턴스는 동일한 <code>docroot</code> 를 사용합니다. 가상 서버의 <code>docroot</code> 등록 정보를 구성하여 서버 인스턴스에서 다른 <code>docroot</code> 를 사용하게 해야 합니다.
<code>generated</code>	EJB 스텝, 컴파일된 JSP 클래스 및 보안 정책 파일과 같은 Java EE 응용 프로그램 및 모듈용으로 생성된 파일입니다. 이 디렉토리는 응용 프로그램 디렉토리와 동기화됩니다. 따라서 서버 인스턴스에서 참조하는 응용 프로그램에 해당하는 디렉토리만 동기화됩니다.

표 8-1 원격 서버 인스턴스 사이에서 동기화되는 파일과 디렉토리 (계속)

파일 또는 디렉토리	설명
lib, lib/classes	전체 도메인에 배포된 응용 프로그램에 사용되는 일반 Java 클래스 파일이나 jar 및 zip 아카이브를 드롭할 수 있는 폴더입니다. 이러한 클래스는 Application Server의 클래스 로더를 사용하여 로드됩니다. 클래스 로더에서의 로드 순서는 lib/classes, lib/*.jar, lib/*.zip입니다.
lib/ext	전체 도메인에 배포된 응용 프로그램에 사용되는 일반 Java 확장 클래스(jar 및 zip 아카이브)를 드롭할 수 있는 폴더입니다. 이러한 jar 파일은 Java 확장 메커니즘을 사용하여 로드됩니다.

라이브러리 파일 동기화

응용 프로그램에 대한 `--libraries` 배포 시간 속성을 사용하여 응용 프로그램의 런타임 종속성을 지정할 수 있습니다.

전체 도메인에서 라이브러리를 사용할 수 있게 하려면 `domain-dir/lib` 또는 `domain-dir/lib/classes`에 JAR 파일을 배치할 수 있습니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide**의 “Using the Common Classloader”를 참조하십시오. 이 작업은 도메인의 모든 응용 프로그램에서 공유하는 JDBC 드라이버 및 기타 유틸리티 라이브러리에 일반적으로 적용됩니다.

클러스터 전체 또는 독립 실행형 서버 전체에서 사용하는 경우 jar를 `domain-dir/domain1/config/xyz-config/lib` 디렉토리에 복사합니다. 그런 다음 jar를 `xyz-config`의 `classpath-suffix` 또는 `classpath-prefix` 요소에 추가합니다. 이렇게 하면 `xyz-config`를 사용하는 모든 서버 인스턴스에 대해 jar가 동기화됩니다.

요약하면 다음과 같습니다.

- `domains/domain1/lib` - 도메인 전체 범위, 일반 클래스 로더, jar를 자동으로 추가합니다.
- `domains/domain1/config/cluster1, config/lib` - 구성 전체 범위, `classpath-prefix` 또는 `classpath-suffix`를 업데이트합니다.
- `domains/domain1/config/cluster1, config/lib/ext - java.ext.dirs`
(<http://java.sun.com/j2se/1.5.0/docs/guide/extensions/extensions.html>)에 자동으로 추가됩니다.

고유한 설정 및 구성 관리

구성 파일(`domains/domain1/config`에 있음)은 도메인에서 동기화됩니다. 독립 실행형 서버 인스턴스(`server1`)에 사용되는 `server1-config`에 대한 `server.policy` 파일을 사용자 정의하려면 수정된 `server.policy` 파일을 `domains/domain1/config/server1-config` 디렉토리에 배치합니다.

수정된 이 `server.policy` 파일은 독립 실행형 서버 인스턴스인 `server1`에 대해서만 동기화됩니다. `jvm-option`을 업데이트해야 합니다. 예를 들면 다음과 같습니다.

```
<java-config> ...
<jvm-options>-Djava.security.policy=${com.sun.aas.instanceRoot}/config
/server1-config/server.policy</jvm-options></java-config>
```

대용량 응용 프로그램 동기화

환경에 동기화할 대용량 응용 프로그램이 포함되거나 사용 가능한 메모리가 제한된 경우 JVM 옵션을 조정하여 메모리 사용을 제한할 수 있습니다. 이렇게 조정하면 메모리 부족 오류 발생 가능성이 줄어듭니다. 인스턴스 동기화 JVM은 기본 설정을 사용하지만 JVM 옵션을 구성하여 변경할 수 있습니다.

INSTANCE-SYNC-JVM-OPTIONS 등록 정보를 사용하여 JVM 옵션을 설정합니다. 등록 정보를 설정하는 명령은 다음과 같습니다.

```
asadmin set
domain.node-agent.node_agent_name.property.INSTANCE-SYNC-JVM-OPTIONS="JVM_options"
```

예를 들면 다음과 같습니다.

```
asadmin set
domain.node-agent.node0.property.INSTANCE-SYNC-JVM-OPTIONS="-Xmx32m -Xss2m"
```

이 예에서 노드 에이전트는 `node0`이고 JVM 옵션은 `-Xmx32m -Xss2m`입니다.

자세한 내용은 <http://java.sun.com/docs/hotspot/VMOptions.html>을 참조하십시오.

주 - 구성에서 등록 정보를 추가하거나 변경한 경우 노드 에이전트가 자동으로 동기화되지 않으므로 INSTANCE-SYNC-JVM-OPTIONS 등록 정보를 변경한 후 노드 에이전트를 다시 시작합니다.

doNotRemoveList 플래그 사용

응용 프로그램이 Application Server가 동기화하는 디렉토리(applications, generated, docroot, config, lib)에서 파일을 저장하고 읽어야 하는 경우 `doNotRemoveList` 플래그를 사용합니다. 이 속성은 컴포로 구분된 파일 또는 디렉토리 목록을 사용합니다. 응용 프로그램 종속 파일은 DAS가 관리하는 중앙 저장소에 없는 경우에도 서버 시작 도중에 제거되지 않습니다. 중앙 저장소에 동일한 파일이 있는 경우 동기화 중 덮어씁니다.

INSTANCE-SYNC-JVM-OPTIONS 등록 정보를 사용하여 `doNotRemoveList` 속성을 전달합니다.

예를 들면 다음과 같습니다.

```
<node-agent name="na1" ...>
```

```
...
<property name="INSTANCE-SYNC-JVM-OPTIONS"
value="-Dcom.sun.appserv.doNotRemoveList=applications/j2ee-modules
/<webapp_context>/logs,generated/mylogdir"/>

</node-agent>
```

노드 에이전트 로그 보기

각 노드 에이전트에는 고유한 로그 파일이 있습니다. 노드 에이전트에 문제가 생기면 다음 위치에 있는 로그 파일을 참조하십시오.

node_agent_dir/node_agent_name/agent/logs/server.log.

노드 에이전트 로그에서 서버의 로그를 확인하여 문제점에 대한 자세한 메시지를 확인할 것을 권장할 때도 있습니다.

서버 로그는 다음 위치에 있습니다.

node_agent_dir/node_agent_name/server_name/logs/server.log

*node_agent_dir*의 기본 위치는 *install_dir/nodeagents*입니다.

노드 에이전트 작업

- 171 페이지 “노드 에이전트 작업을 수행하는 방법”
- 172 페이지 “노드 에이전트 자리 표시자”
- 172 페이지 “노드 에이전트 자리 표시자 만들기”
- 173 페이지 “노드 에이전트 만들기”
- 174 페이지 “노드 에이전트 시작”
- 174 페이지 “노드 에이전트 중지”
- 175 페이지 “노드 에이전트 삭제”
- 175 페이지 “일반 노드 에이전트 정보 보기”
- 176 페이지 “노드 에이전트 구성 삭제”
- 176 페이지 “노드 에이전트 구성 편집”
- 177 페이지 “노드 에이전트 영역 편집”
- 177 페이지 “JMX에 대해 노드 에이전트 Listener 편집”

노드 에이전트 작업을 수행하는 방법

일부 노드 에이전트 작업에서는 노드 에이전트가 실행되는 시스템에서 **asadmin** 도구를 로컬로 사용해야 합니다. 다른 작업은 관리 콘솔 또는 **asadmin**을 사용하여 원격으로 수행할 수 있습니다.

다음 표에서는 작업과 작업이 실행되는 위치를 요약하여 설명합니다.

표 8-2 노드 에이전트 작업을 수행하는 방법

작업	관리 콘솔	asadmin 명령
Domain Administration Serve에 노드 에이전트 자리 표시자 만들기	노드 에이전트 자리 표시자 만들기 페이지	create-node-agent-config
노드 에이전트 만들기	사용할 수 없음	create-node-agent
노드 에이전트 시작	사용할 수 없음	start-node-agent
노드 에이전트 중지	사용할 수 없음	stop-node agent
DAS에서 노드 에이전트 구성 삭제	노드 에이전트 페이지	delete-node-agent-config
로컬 시스템에서 노드 에이전트 삭제	사용할 수 없음	delete-node-agent
노드 에이전트 구성 편집	노드 에이전트 페이지	set
노드 에이전트 나열	노드 에이전트 페이지	list-node-agents

노드 에이전트 자리 표시자

노드 에이전트 자리 표시자를 사용하여 기존 노드 에이전트 없이도 서버 인스턴스를 만들고 삭제할 수 있습니다. 노드 에이전트 자리 표시자는 노드 에이전트의 로컬 시스템에 노드 에이전트를 만들기 전에 도메인 관리 서버(DAS)에 만들어집니다.

노드 에이전트 자리 표시자를 만드는 방법에 대한 자세한 내용은 [172 페이지 “노드 에이전트 자리 표시자 만들기”](#)를 참조하십시오.

주 - 자리 표시자 노드 에이전트를 만들었으면 이를 사용하여 해당 도메인에 인스턴스를 만들 수 있습니다. 그러나 인스턴스를 시작하기 전에 asadmin 명령을 사용하여 인스턴스가 상주하는 시스템에서 로컬로 실제 노드 에이전트를 만들고 시작해야 합니다. [173 페이지 “노드 에이전트 만들기”](#) 및 [174 페이지 “노드 에이전트 시작”](#)을 참조하십시오.

▼ 노드 에이전트 자리 표시자 만들기

노드 에이전트는 노드 에이전트를 호스트하는 시스템에서 로컬로 만들어져야 하므로 관리 콘솔을 통해 노드 에이전트에 대한 자리 표시자만 만들 수 있습니다. 이 자리 표시자는 노드 에이전트가 아직 없는 노드 에이전트 구성입니다.

자리 표시자를 만든 후 노드 에이전트를 호스트하는 시스템에서 asadmin 명령 create-node-agent를 사용하여 작성을 완료합니다. 자세한 내용은 [173 페이지 “노드 에이전트 만들기”](#)를 참조하십시오.

노드 에이전트 작성 및 사용과 관련된 작업 단계를 보려면 [165 페이지](#) “노드 에이전트 배포”를 참조하십시오.

- 1 트리 구성 요소에서 노드 에이전트 노드를 선택합니다.
- 2 노드 에이전트 페이지에서 새로 만들기를 누릅니다.
- 3 현재 노드 에이전트 자리 표시자 페이지에서 새로운 노드 에이전트의 이름을 입력합니다.
도메인의 모든 노드 에이전트 이름, 서버 인스턴스 이름, 클러스터 이름 및 구성 이름에서 이름이 고유해야 합니다.
- 4 확인을 누릅니다.
노드 에이전트 페이지에 새로운 노드 에이전트의 자리 표시자가 나열됩니다.

자세한 정보 해당 **asadmin** 명령

`create-node-agent-config`

노드 에이전트 만들기

노드 에이전트를 만들려면 노드 에이전트가 실행되는 시스템에서 **asadmin** 명령 `create-node-agent`를 로컬로 실행합니다.

노드 에이전트의 기본 이름은 노드 에이전트가 만들어지는 호스트 이름입니다.

노드 에이전트 자리 표시자를 이미 만든 경우 노드 에이전트 자리 표시자와 동일한 이름을 사용하여 연관된 노드 에이전트를 만듭니다. 노드 에이전트 자리 표시자를 만들지 않았지만 DAS를 실행 중이고 연결할 수 있는 경우 `create-node-agent` 명령으로도 DAS에 노드 에이전트 구성(자리 표시자)을 만들 수 있습니다.

명령 구문에 대한 자세한 설명은 명령에 대한 온라인 도움말을 참조하십시오.

예 8-1 노드 에이전트 작성 예

다음 명령을 사용하면 노드 에이전트가 만들어집니다.

```
asadmin create-node-agent --host myhost --port 4849 ---user admin nodeagent1
```

여기서 *myhost*는 DAS(Domain Administration Server) 호스트 이름이고, 4849는 DAS 포트 번호이고, *admin*은 DAS 사용자이고, *nodeagent1*은 작성할 노드 에이전트의 이름입니다.

주 - 다음 상황에서는 DNS에 연결 가능한 호스트 이름을 지정해야 합니다.

- 도메인이 서브넷 경계에 걸쳐 있을 경우(즉, 노드 에이전트와 DAS가 서로 다른 도메인(예: sun.com 및 java.com)에 있을 경우)
- DNS에 등록되지 않은 호스트 이름의 DHCP 시스템을 사용할 경우

도메인과 노드 에이전트를 만들 때 이들에 대한 호스트 이름을 명시적으로 지정하여 DNS 연결 가능한 호스트 이름을 지정합니다.

```
create-domain --domainproperties domain.hostName=DAS-host-name
create-node-agent --hostDAS-host-name
--agentproperties remoteclientaddress=node-agent-host-name
```

다른 방법은 플랫폼에 관련된 hosts hostname/IP 결정 파일을 업데이트하여 호스트 이름이 올바른 IP 주소로 변환될 수 있도록 하는 것입니다. 그러나 DHCP를 사용하여 다시 연결할 경우 다른 IP 주소가 지정될 수 있습니다. 그럴 경우 각 서버에서 호스트 결정 파일을 업데이트해야 합니다.

노드 에이전트 시작

노드 에이전트로 서버 인스턴스를 관리하려면 노드 에이전트가 실행되고 있어야 합니다. 노드 에이전트가 상주하는 시스템에서 `asadmin` 명령 `start-node-agent`를 로컬로 실행하여 노드 에이전트를 시작합니다.

명령 구문에 대한 자세한 설명은 명령에 대한 온라인 도움말을 참조하십시오.

예를 들면 다음과 같습니다.

```
asadmin start-node-agent --user admin --startinstances=false nodeagent1
```

여기서 `admin`은 관리 사용자이고 `nodeagent1`은 시작할 노드 에이전트입니다. 그런 다음 `asadmin start-instance` 명령을 사용하여 서버 인스턴스를 시작합니다.

노드 에이전트 중지

노드 에이전트가 상주하는 시스템에서 `asadmin` 명령 `stop-node-agent`를 실행하여 실행 중인 노드 에이전트를 중지합니다. `stop-node-agent` 명령은 노드 에이전트가 관리하는 모든 서버 인스턴스를 중지합니다.

명령 구문에 대한 자세한 설명은 명령에 대한 온라인 도움말을 참조하십시오.

예를 들면 다음과 같습니다.

```
asadmin stop-node-agent nodeagent1
```

여기서 `nodeagent1`은 노드 에이전트의 이름입니다.

노드 에이전트 삭제

노드 에이전트를 삭제하기 전에 노드 에이전트를 중지해야 합니다. 시작하지 않았거나 DAS에 연결하지 못했던(즉 바인딩되지 않은) 노드 에이전트를 삭제할 수도 있습니다.

노드 에이전트가 상주하는 시스템에서 `asadmin delete-node-agent`를 실행하여 노드 에이전트 파일을 삭제합니다.

명령 구문에 대한 자세한 설명은 명령에 대한 온라인 도움말을 참조하십시오.

예를 들면 다음과 같습니다.

```
asadmin delete-node-agent nodeagent1
```

여기서 `nodeagent1`은 노드 에이전트입니다.

노드 에이전트를 삭제할 경우 관리 콘솔이나 `asadmin delete-node-agent-config` 명령을 사용하여 DAS에서 노드 에이전트의 구성도 삭제해야 합니다.

▼ 일반 노드 에이전트 정보 보기

1 트리 구성 요소에서 노드 에이전트 노드를 선택합니다.

2 노드 에이전트 이름을 누릅니다.

노드 에이전트가 이미 존재하지만 여기에 나타나지 않으면 `asadmin start-node-agent`를 사용하여 노드 에이전트의 호스트 시스템에서 노드 에이전트를 시작합니다. [174 페이지 “노드 에이전트 시작”](#)을 참조하십시오.

3 노드 에이전트의 호스트 이름을 확인합니다.

호스트 이름이 알 수 없는 호스트일 경우 노드 에이전트가 DAS와 초기 연결하지 않았기 때문입니다.

4 노드 에이전트 상태를 확인합니다.

상태는 다음과 같을 수 있습니다.

- **실행 중:** 노드 에이전트를 제대로 만들었고 현재 실행 중입니다.
- **실행 중이 아님:** 노드 에이전트가 로컬 시스템에서 만들어졌으나 시작되지 않았거나 노드 에이전트가 시작되었으나 중지되었습니다.
- **랑데부를 기다리는 중:** 노드 에이전트가 로컬 시스템에 작성된 적이 없는 자리 표시자입니다.

173 페이지 “노드 에이전트 만들기” 및 174 페이지 “노드 에이전트 시작”을 참조하십시오.

5 시작 시 인스턴스를 시작할지 여부를 선택합니다.

노드 에이전트를 시작할 때 노드 에이전트와 연관된 서버 인스턴스를 자동으로 시작하려면 예를 선택합니다. 인스턴스를 수동으로 시작하려면 아니요를 선택합니다.

6 노드 에이전트가 DAS와 연결했는지 여부를 확인합니다.

노드 에이전트가 DAS에 연결하지 않았으면 성공적으로 시작되지 않습니다.

7 노드 에이전트와 연결된 서버 인스턴스를 관리합니다.

노드 에이전트가 실행 중일 경우 인스턴스 이름 옆에 있는 확인란을 선택하고 시작이나 중지를 눌러 인스턴스를 시작하거나 중지합니다.

▼ 노드 에이전트 구성 삭제

관리 콘솔을 통해 도메인에서 노드 에이전트 구성만 삭제할 수 있습니다. 실제 노드 에이전트는 삭제할 수 없습니다. 노드 에이전트 자체를 삭제하려면 노드 에이전트의 로컬 시스템에서 `asadmin delete-node-agent`를 실행합니다. 자세한 내용은 175 페이지 “노드 에이전트 삭제”를 참조하십시오.

노드 에이전트 구성을 삭제하기 전에 노드 에이전트를 중지해야 하고 연관된 인스턴스가 없어야 합니다. 노드 에이전트를 중지하려면 `asadmin stop-node-agent`를 사용합니다. 자세한 내용은 174 페이지 “노드 에이전트 중지”를 참조하십시오.

1 트리 구성 요소에서 노드 에이전트 노드를 선택합니다.

2 노드 에이전트 페이지에서 삭제할 노드 에이전트 옆에 있는 확인란을 선택합니다.

3 삭제를 누릅니다.

자세한 정보 **해당 asadmin 명령**

`delete-node-agent-config`

▼ 노드 에이전트 구성 편집

1 트리 구성 요소에서 노드 에이전트 노드를 확장합니다.

2 편집하려면 노드 에이전트 구성을 누릅니다.

- 3 시작 시 인스턴스 시작을 선택하여 에이전트가 시작될 때 에이전트의 서버 인스턴스를 시작합니다.

이 페이지에서 인스턴스를 수동으로 시작 및 중지할 수도 있습니다.

이 구성이 자리 표시자 노드 에이전트를 위한 것일 경우 `asadmin create-node-agent`를 사용하여 실제 노드 에이전트를 만들 때 이 구성을 사용합니다. 노드 에이전트 작성에 대한 자세한 내용은 [173 페이지 “노드 에이전트 만들기”](#)를 참조하십시오.

이 구성이 기존 노드 에이전트를 위한 것일 경우 노드 에이전트 구성 정보가 자동으로 동기화됩니다.

▼ 노드 에이전트 영역 편집

노드 에이전트에 연결하는 사용자에게 인증 영역을 설정해야 합니다. 관리 사용자만 노드 에이전트에 액세스해야 합니다.

- 1 트리 구성 요소에서 노드 에이전트 노드를 확장합니다.
- 2 편집하려면 노드 에이전트 구성을 누릅니다.
- 3 인증 영역 탭을 누릅니다.
- 4 노드 에이전트 영역 편집 페이지에서 영역을 입력합니다.
노드 에이전트를 만들 때 만들어진 `admin-realm`이 기본값입니다. 다른 영역을 사용하려면 도메인에서 제어하는 모든 구성 요소의 영역을 대체합니다. 그렇지 않으면 구성 요소가 제대로 통신하지 못합니다.
- 5 클래스 이름 필드에서 영역을 구현하는 Java 클래스를 지정합니다.
- 6 필수 등록 정보를 모두 추가합니다.
인증 영역에는 공급자 관련 등록 정보가 필요한데, 이는 특정 구현에서 요구하는 내용에 따라 다릅니다.

▼ JMX에 대해 노드 에이전트 Listener 편집

노드 에이전트는 JMX를 사용하여 Domain Administration Server와 통신합니다. 따라서 JMX 요청을 수신하려면 포트와 다른 listener 정보가 있어야 합니다.

- 1 트리 구성 요소에서 노드 에이전트 노드를 확장합니다.
- 2 편집하려면 노드 에이전트 구성을 누릅니다.
- 3 JMX 탭을 누릅니다.

- 4 주소 필드에 IP 주소 또는 호스트 이름을 입력합니다.
Listener가 고유한 포트 값을 사용하여 서버에 대한 모든 IP 주소에서 수신할 경우 주소 필드에 0.0.0.0을 입력합니다. 그렇지 않으면 서버에 대한 유효한 IP 주소를 입력합니다.
- 5 포트 필드에서 노드 에이전트의 JMX 커넥터가 수신하는 포트를 입력합니다.
IP 주소가 0.0.0.0일 경우 포트 번호가 고유해야 합니다.
- 6 JMX 프로토콜 필드에서 JMX 커넥터가 지원하는 프로토콜을 입력합니다.
기본값은 rmi_jrmp입니다.
- 7 모든 IP 주소에 대한 연결을 허용하려면 모든 주소 허용 옆에 있는 확인란을 선택합니다.
노드 에이전트는 네트워크 카드에 연결된 특정 IP 주소에서 수신하거나 모든 IP 주소에서 수신합니다. 모든 주소를 허용하면 "listening host address" 등록 정보에 0.0.0.0 값이 입력됩니다.
- 8 영역 이름 필드에서 Listener에 대한 인증을 처리하는 영역 이름을 입력합니다.
이 페이지의 보안 절에서 SSL, TLS 또는 SSL 및 TLS 보안 둘 다를 사용하도록 Listener를 구성합니다.
보안 listener를 설정하려면 다음 작업을 수행합니다.
- 9 보안 필드에서 사용 가능 확인란을 선택합니다.
보안은 기본적으로 활성화되어 있습니다.
- 10 클라이언트 인증을 설정합니다.
이 Listener를 사용할 경우 클라이언트가 자신을 서버에 인증하도록 하려면 클라이언트 인증 필드에서 사용 가능 확인란을 선택합니다.
- 11 인증서 별명을 입력합니다.
인증서 별명 필드에 기존 서버 키 쌍 및 인증서의 이름을 입력합니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 “인증서 및 SSL 작업”을 참조하십시오.
- 12 SSL3/TLS 섹션:
 - a. Listener에서 활성화할 보안 프로토콜을 선택합니다.
SSL3, TLS 또는 두 가지 프로토콜을 모두 선택해야 합니다.
 - b. 프로토콜이 사용하는 암호 제품군을 선택합니다.
모든 암호 제품군을 사용하려면 지원되는 모든 암호화 제품군을 선택합니다.
- 13 저장을 누릅니다.

고가용성 세션 지속성 및 페일오버 구성

이 장에서는 고가용성 세션 지속성을 활성화 및 구성하는 방법을 설명합니다.

- 179 페이지 “세션 지속성 및 페일오버 개요”
- 181 페이지 “고가용성 세션 지속성 설정”
- 183 페이지 “HTTP 세션 페일오버”
- 187 페이지 “Stateful Session Bean 페일오버”

세션 지속성 및 페일오버 개요

Application Server에서는 HTTP 세션 데이터 및 Stateful Session Bean(SFSB) 세션 데이터의 **페일오버**를 통해 고가용성 세션 지속성을 제공합니다. 페일오버는 서버 인스턴스 또는 하드웨어 오류가 발생했을 때 또 다른 서버 인스턴스가 분산된 세션을 인계 받음을 의미합니다.

요구 사항

다음 경우에 분산 세션이 여러 Sun Java System Application Server 인스턴스에서 실행될 수 있습니다.

- 각 서버 인스턴스가 동일한 HADB(고가용성 데이터베이스)에 액세스할 수 있는 경우. 이 데이터베이스를 활성화하는 방법에 대한 자세한 내용은 `configure-ha-cluster(1)`를 참조하십시오.
- 각 서버 인스턴스에 동일한 분산 가능 웹 응용 프로그램이 배포되어 있는 경우. `web.xml` 배포 설명자 파일의 `web-app` 요소에 `distributable` 요소가 포함되어야 합니다.
- 웹 응용 프로그램은 고가용성 세션 지속성을 사용하는 경우. 분산 불가능 웹 응용 프로그램이 고가용성 세션 지속성을 사용하도록 구성되면 서버는 로그 파일에 오류를 씁니다.

- `--availabilityenabled` 옵션을 `true`로 설정하고 `deploy` 또는 `deploydir` 명령을 사용하여 웹 응용 프로그램을 배포하는 경우. 이러한 명령에 대한 자세한 내용은 `deploy(1)` 및 `deploydir(1)`을 참조하십시오.

제한 사항

세션이 페일오버되면 열린 파일 또는 네트워크 연결에 대한 모든 참조가 손실됩니다. 이러한 제한 사항을 고려하면서 응용 프로그램을 코딩해야 합니다.

페일오버를 지원하는 분산 세션에만 특정 객체를 바인딩할 수 있습니다. Servlet 2.4 사양과는 반대로, 페일오버에 대해 지원되지 않는 객체 유형이 분산 세션에 바인딩되면 Sun Java System Application Server에서는 `IllegalArgumentException`을 발생시킵니다.

페일오버를 지원하는 분산 세션으로 다음 객체를 바인딩할 수 있습니다.

- 모든 EJB 구성 요소에 대한 로컬 홈 및 객체 참조
- 같은 시스템에 있는 Entity Bean, Stateful Session Bean 및 분산 Entity Bean 원격 홈 참조, 원격 참조
- 분산 세션 Bean 원격 홈 및 원격 참조
- `InitialContext` 및 `java:comp/env`에 대한 JNDI 컨텍스트
- `UserTransaction` 객체. 그러나 장애가 발생한 인스턴스가 다시 시작되지 않으면 준비된 모든 전역 트랜잭션이 손실되고 제대로 롤백 또는 완결되지 못할 수 있습니다.
- 일련화 가능 Java 유형

다음 객체 유형은 페일오버를 지원하는 세션에 바인딩할 수 없습니다.

- `JDBC DataSource`
- `JMS(Java Message Service) ConnectionFactory` 및 `Destination` 객체
- `JavaMail™` 세션
- 연결 팩토리
- 관리 객체
- 웹 서비스 참조

일반적으로 이러한 객체의 경우 페일오버가 작동되지 않습니다. 그러나 객체가 일련화 가능한 경우 등 일부 경우에 페일오버가 작동될 수 있습니다.

샘플 응용 프로그램

다음 디렉토리에는 세션 지속성을 설명하는 샘플 응용 프로그램이 포함되어 있습니다.

`install_dir/samples/ee-samples/highavailability`

`install_dir/samples/ee-samples/failover`

다음 예제 응용 프로그램은 SFSB 세션 지속성을 보여 줍니다.

`install_dir/samples/ee-samples/failover/apps/sfsbfailover`

고가용성 세션 지속성 설정

이 절에서는 고가용성 세션 지속성을 설정하는 방법을 설명하며 다음 내용으로 구성되어 있습니다.

- 181 페이지 “고가용성 세션 지속성 설정”
- 182 페이지 “세션 가용성 활성화”

▼ 고가용성 세션 지속성 설정

시작하기 전에 고가용성 세션 지속성은 동적 배포, 동적 다시 로드 및 자동 배포와 호환되지 않습니다. 이러한 기능은 프로덕션 환경이 아닌 개발용이므로 HA 세션 지속성을 활성화하기 전에 이러한 기능을 비활성화해야 합니다. 이러한 기능을 비활성화하는 방법에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 2 장, “응용 프로그램 배포”를 참조하십시오.

1 Application Server 클러스터를 만듭니다.

자세한 내용은 146 페이지 “클러스터 만들기”를 참조하십시오.

2 클러스터용 HADB 데이터베이스를 만듭니다.

자세한 내용은 `configure-ha-cluster(1)`를 참조하십시오.

3 클러스터에 대한 HTTP 로드 균형 조정을 설정합니다.

자세한 내용은 119 페이지 “HTTP 로드 균형 조정 설정”을 참조하십시오.

4 원하는 응용 프로그램 서버 인스턴스 및 웹 또는 EJB 컨테이너에 대한 가용성을 활성화합니다.

그런 후 세션 지속성 설정을 구성합니다. 다음 중 하나를 선택하십시오.

- 관리 콘솔을 사용합니다. 183 페이지 “서버 인스턴스에 대한 가용성 활성화”를 참조하십시오.
- `asadmin` 명령줄 유틸리티를 사용합니다. `set(1)` 및 `configure-ha-persistence(1)`를 참조하십시오.

5 클러스터의 모든 서버 인스턴스를 다시 시작합니다.

인스턴스에서 현재 요청을 처리할 경우 인스턴스가 처리 중인 요청을 처리할 수 있는 충분한 시간을 가질 수 있도록 인스턴스를 다시 시작하기 전에 인스턴스를 중지합니다. 자세한 내용은 127 페이지 “서버 인스턴스 또는 클러스터 비활성화(정지)”를 참조하십시오.

6 가용성이 필요한 모든 특정 SFSB에 대해 가용성을 활성화합니다.

세션 상태에 필요한 검사점 지정 방법을 선택합니다. [190 페이지 “개별 Bean에 대한 가용성 구성”](#)을 참조하십시오.

7 각 웹 모듈의 고가용성을 유지하려면 웹 모듈을 분산 가능 상태로 만듭니다.**8 배포 중에 개별 응용 프로그램, 웹 모듈 또는 EJB 모듈에 대한 가용성을 활성화합니다.**

[190 페이지 “개별 응용 프로그램 또는 EJB 모듈에 대한 가용성 구성”](#)을 참조하십시오.

관리 콘솔에서 가용성 사용 가능 확인란을 선택하거나 `--availabilityenabled` 옵션을 `true`로 설정하여 `asadmin deploy` 명령을 사용합니다.

세션 가용성 활성화

다음과 같은 5가지 범위(최고 수준에서 최저 수준까지)로 세션 가용성을 활성화할 수 있습니다.

1. 기본적으로 활성화된 서버 인스턴스. 서버 인스턴스에 대한 세션 가용성 활성화는 서버 인스턴스에서 실행 중인 모든 응용 프로그램이 고가용성 세션 지속성을 가질 수 있다는 것을 의미합니다. 자세한 내용은 다음 섹션 [183 페이지 “서버 인스턴스에 대한 가용성 활성화”](#)를 참조하십시오.
2. 기본적으로 활성화된 컨테이너(웹 또는 EJB). 컨테이너 수준에서 가용성을 활성화하는 방법에 대한 자세한 내용은 다음을 참조하십시오.
 - [183 페이지 “웹 컨테이너에 대한 가용성 구성”](#)
 - [188 페이지 “EJB 컨테이너에 대한 가용성 구성”](#)
3. 기본적으로 비활성화된 응용 프로그램
4. 기본적으로 비활성화된 독립 실행형 웹 또는 EJB 모듈
5. 기본적으로 비활성화된 개별 SFSB

지정된 범위에서 가용성을 활성화하려면 모든 상위 수준에서 가용성을 활성화해야 합니다. 예를 들어, 응용 프로그램 수준에서 가용성을 활성화하려면 서버 인스턴스 및 컨테이너 수준에서도 활성화해야 합니다.

지정한 수준의 기본값은 다음으로 높은 수준의 설정값입니다. 예를 들어, 컨테이너 수준에서 가용성을 활성화한 경우 기본적으로 응용 프로그램 수준에서도 가용성이 활성화됩니다.

서버 인스턴스 수준에서 가용성을 비활성화한 경우 다른 수준에서 활성화해도 적용되지 않습니다. 서버 인스턴스 수준에서 가용성을 활성화한 경우 명시적으로 비활성화하지 않으면 모든 수준에서 가용성이 활성화됩니다.

서버 인스턴스에 대한 가용성 활성화

서버 인스턴스에 대한 가용성을 활성화하려면 `asadmin set` 명령을 사용하여 구성의 `availability-service.availability-enabled` 등록 정보를 `true`로 설정합니다.

예를 들어, `config1`이 구성 이름이면 다음 명령을 사용합니다.

```
asadmin set --user admin --passwordfile password.txt
--host localhost
--port 4849
config1.availability-service.availability-enabled="true"
```

▼ 관리 콘솔을 사용하여 서버 인스턴스에 대한 가용성 활성화

- 1 트리 구성 요소에서 구성 노드를 확장합니다.
- 2 편집할 구성의 노드를 확장합니다.
- 3 가용성 서비스 노드를 선택합니다.
- 4 가용성 서비스 페이지에서 가용성 서비스 확인란을 선택하여 인스턴스 수준의 가용성을 활성화합니다.
비활성화하려면 확인란을 선택 해제합니다.
또는 세션 지속성을 위해 HADB 연결에 사용한 JDBC 자원을 변경한 경우 저장소 풀 이름을 변경할 수 있습니다. 자세한 내용은 `configure-ha-cluster(1)`를 참조하십시오.
- 5 저장 버튼을 누릅니다.
- 6 서버 인스턴스를 중지했다가 다시 시작합니다.

HTTP 세션 페일오버

J2EE 응용 프로그램은 일반적으로 많은 양의 세션 상태 데이터를 포함하게 됩니다. 웹 장바구니가 세션 상태의 일반적인 예에 해당합니다. 또한 응용 프로그램은 자주 필요한 데이터를 세션 객체에 캐시할 수 있습니다. 실제로 사용자 상호 작용이 자주 발생하는 거의 모든 응용 프로그램에서는 세션 상태가 유지되어야 합니다.

웹 컨테이너에 대한 가용성 구성

`asadmin`을 사용하여 웹 컨테이너 가용성을 활성화 및 구성하려면 `configure-ha-persistence(1)`를 참조하십시오.

또는 `asadmin set` 명령을 사용하여 구성의

`availability-service.web-container-availability.availability-enabled` 등록 정보를 `true`로 설정한 후 `configure-ha-persistence`를 사용하여 해당 등록 정보를 설정합니다.

예를 들어, 다음과 같이 `set` 명령을 사용합니다. 여기서 `config1`은 구성 이름입니다.

```
asadmin set --user admin --passwordfile password.txt
--host localhost --port 4849
config1.availability-service.web-container-availability.availability-enabled="true"
asadmin configure-ha-persistence --user admin --passwordfile secret.txt
--type ha
--frequency web-method
--scope modified-session
--store jdbc/hastore
--property maxSessions=1000:reapIntervalSeconds=60 cluster1
```

▼ 관리 콘솔을 사용하여 웹 컨테이너에 대한 가용성 활성화

1 트리 구성 요소에서 원하는 구성을 선택합니다.

2 가용성 서비스를 누릅니다.

3 웹 컨테이너 가용성 탭을 선택합니다.

가용성 서비스 확인란을 선택하여 가용성을 활성화합니다. 비활성화하려면 확인란을 선택 해제합니다.

4 다음 섹션 **184 페이지** “가용성 설정”에 설명된 대로 다른 설정을 변경합니다.

5 서버 인스턴스를 다시 시작합니다.

가용성 설정

가용성 서비스의 웹 컨테이너 가용성 탭을 사용하여 다음과 같은 가용성 설정을 변경할 수 있습니다.

지속성 유형: 가용성이 활성화된 웹 응용 프로그램에 대한 세션 지속성 메커니즘을 지정합니다. 허용되는 값은 `memory`(지속성 없음), `file`(파일 시스템) 및 `ha`(HADB)입니다.

`ha` 세션 지속성을 사용하려면 먼저 HADB를 구성하고 활성화해야 합니다. 구성에 대한 자세한 내용은 `configure-ha-cluster(1)`를 참조하십시오.

웹 컨테이너 가용성이 활성화되면 기본값은 `ha`입니다. 그렇지 않은 경우 기본값은 `memory`입니다. 세션 지속성이 필요한 프로덕션 환경의 경우 `ha`를 사용합니다. 두 유형, `memory` 및 `file` 지속성은 고가용성 세션 지속성을 제공하지 않습니다.

지속성 빈도: 세션 상태가 저장되는 빈도를 지정합니다. 지속성 유형이 `ha`일 경우에만 해당됩니다. 허용되는 값은 다음과 같습니다.

- `web-method` - 응답을 다시 클라이언트에 전송하기 전에 각 웹 요청 끝에 세션 상태가 저장됩니다. 이 모드는 오류 시 세션 상태의 완벽한 업데이트를 가장 확실하게 보장합니다. 이 값이 기본값입니다.
- `time-based-reapIntervalSeconds` 저장소 등록 정보에서 설정한 빈도로 세션 상태가 백그라운드로 저장됩니다. 이 모드는 세션 상태의 완벽한 업데이트를 확실하게 보장하지 못합니다. 그러나 요청 후마다 상태를 저장하지 않기 때문에 성능이 크게 향상됩니다.

지속성 범위: 저장되는 세션 객체의 양과 세션 상태가 저장되는 빈도를 지정합니다. 지속성 유형이 `ha`일 경우에만 해당됩니다. 허용되는 값은 다음과 같습니다.

- `session` - 항상 전체 세션 상태가 저장됩니다. 이 모드는 분산 가능한 웹 응용 프로그램의 경우 세션 데이터의 정확한 저장을 가장 확실하게 보장합니다. 이 값이 기본값입니다.
- `modified-session` - 수정하면 전체 세션 상태가 저장됩니다.
`HttpSession.setAttribute()` 또는 `HttpSession.removeAttribute()`를 호출한 경우 세션이 수정된 것으로 간주됩니다. 속성을 변경할 때마다 `setAttribute()`를 호출하도록 해야 합니다. 이는 J2EE 사양 요구 사항이 아니지만 이 모드가 제대로 작동하려면 필요합니다.
- `modified-attribute` - 수정된 세션 속성만 저장됩니다. 이 모드가 제대로 작동하려면 몇 가지 지침을 수행해야 합니다.
 - 세션 상태가 수정될 때마다 `setAttribute()`를 호출합니다.
 - 속성 간에는 상호 참조가 없어야 합니다. 별개 속성 키의 객체 그래프는 별도로 일련화 및 저장됩니다. 별도 키의 객체 간에 객체 상호 참조가 있을 경우 제대로 일련화 및 일련화 해제되지 않습니다.
 - 여러 속성에서 세션 상태를 분배하거나 최소한 읽기 전용 속성 및 수정 가능한 속성 간에 세션 상태를 분배합니다.

단일 사인 온 상태: 단일 사인 온 상태의 지속성을 활성화하려면 이 확인란을 선택합니다. 비활성화하려면 확인란을 선택 해제합니다. 자세한 내용은 [186 페이지 “세션 페일오버와 함께 단일 사인 온 사용”](#)을 참조하십시오.

HTTP 세션 저장소: 세션 지속성을 위해 HADB 연결에 사용한 JDBC 자원을 변경한 경우 HTTP 세션 저장소를 변경할 수 있습니다. 자세한 내용은 `configure-ha-cluster(1)`를 참조하십시오.

개별 웹 응용 프로그램에 대한 가용성 구성

개별 웹 응용 프로그램에 대한 가용성을 활성화하고 구성하려면 응용 프로그램 배포 설명자 파일 `sun-web.xml`을 편집합니다. 응용 프로그램 배포 설명자의 설정은 웹 컨테이너의 가용성 설정을 대체합니다.

session-manager 요소의 persistence-type 속성은 응용 프로그램이 사용하는 세션 지속성 유형을 결정합니다.고가용성 세션 지속성을 활성화하려면 이 속성을 ha로 설정해야 합니다.

sun-web.xml 파일에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide**의 “The sun-web.xml File”을 참조하십시오.

예

```
<sun-web-app> ...
  <session-config>
    <session-manager persistence-type=ha>
      <manager-properties>
        <property name=persistenceFrequency value=web-method />
      </manager-properties>
      <store-properties>
        <property name=persistenceScope value=session />
      </store-properties>
    </session-manager> ...
  </session-config> ...
```

세션 페일오버와 함께 단일 사인 온 사용

단일 응용 프로그램 서버 인스턴스에서 응용 프로그램에 의해 인증된 사용자는 동일한 인터페이스에서 실행되는 다른 응용 프로그램에서 개별적으로 다시 인증을 받을 필요가 없습니다. 이를 **단일 사인 온**이라고 합니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide**의 “User Authentication for Single Sign-on”을 참조하십시오.

클러스터의 다른 인스턴스에 HTTP 세션이 페일오버되더라도 이 기능이 계속 작동하려면 단일 사인 온 정보가 HADB에 지속되어야 합니다. 단일 사인 온 정보를 지속하려면 먼저 서버 인스턴스 및 웹 컨테이너의 가용성을 활성화한 후 단일 사인 온 상태 페일오버를 활성화합니다.

관리 콘솔을 사용하면 [183 페이지 “웹 컨테이너에 대한 가용성 구성”](#)에 설명된 대로 가용성 서비스의 웹 컨테이너 가용성 탭에서 단일 사인 온 상태 페일오버를 활성화할 수 있습니다. 또한 asadmin set 명령을 사용하여 구성의 availability-service.web-container-availability.sso-failover-enabled 등록 정보를 true로 설정합니다.

예를 들어, 다음과 같이 set 명령을 사용합니다. 여기서 config1은 구성 이름입니다.

```
asadmin set --user admin --passwordfile password.txt
--host localhost --port 4849
config1.availability-service.web-container-availability.
sso-failover-enabled="true"
```

단일 사인 온 그룹

단일 이름 및 비밀번호 조합을 통해 액세스할 수 있는 응용 프로그램이 **단일 사인 온 그룹**을 구성합니다. 단일 사인 온 그룹의 일부인 응용 프로그램에 해당하는 HTTP 세션의 경우 세션 중 하나가 시간 초과되더라도 다른 세션은 무효화되지 않고 계속 사용할 수 있습니다. 이는 한 세션의 시간 초과가 다른 세션의 가용성에 영향을 미치지 않기 때문입니다.

이 동작의 결과, 세션이 시간 초과되고 세션을 실행 중인 동일한 브라우저 창에서 해당하는 응용 프로그램에 액세스할 경우 다시 인증할 필요가 없습니다. 그러나 새로운 세션이 만들어집니다.

두 개의 다른 응용 프로그램을 사용하는 단일 사인 온 그룹의 일부인 장바구니 응용 프로그램을 예로 듭니다. 다른 두 응용 프로그램의 세션 시간 초과 값이 장바구니 응용 프로그램의 세션 초과 값보다 더 높은 것으로 가정합니다. 장바구니 응용 프로그램의 세션이 시간 초과되고 세션을 실행 중인 동일한 브라우저 창에서 장바구니 응용 프로그램을 실행할 경우 다시 인증할 필요가 없습니다. 그러나 이전 장바구니는 손실되고 새로운 장바구니를 만들어야 합니다. 장바구니 응용 프로그램을 실행하는 세션이 시간 초과되더라도 다른 두 응용 프로그램은 평소대로 계속 실행됩니다.

마찬가지로 다른 두 응용 프로그램에 해당하는 세션이 시간 초과되는 것으로 가정합니다. 세션을 실행 중인 동일한 브라우저 창에서 응용 프로그램에 연결하고 있는 동안에는 다시 인증할 필요가 없습니다.

주 - 세션이 시간 초과된 경우에만 이 동작이 적용됩니다. 단일 사인 온이 활성화되고 `HttpSession.invalidate()`를 사용하여 세션 중 하나를 무효화한 경우 단일 사인 온 그룹에 속하는 모든 응용 프로그램의 세션이 무효화됩니다. 단일 사인 온 그룹에 속하는 응용 프로그램에 액세스할 경우 다시 인증을 받아야 하고 응용 프로그램에 액세스하는 클라이언트에 대해 새로운 세션이 만들어집니다.

Stateful Session Bean 페일오버

SFSB(Stateful Session Bean)에는 클라이언트 특정 상태가 포함됩니다. 클라이언트와 Stateful Session Bean 간에는 일대일 관계가 있습니다. SFSB 작성 시 EJB 컨테이너는 SFSB를 클라이언트에 바인딩하는 고유한 세션 ID를 각 SFSB에 지정합니다.

서버 인스턴스에 장애가 발생할 경우 SFSB의 상태는 지속형 저장소에 저장될 수 있습니다. SFSB의 상태는 수명 주기 중 미리 정의된 시점에 지속형 저장소에 저장됩니다. 이것을 **검사점 지정**이라고 합니다. 검사점을 활성화하면 트랜잭션이 롤백된 경우에도 일반적으로 Bean이 트랜잭션을 완료한 후에 검사점이 지정됩니다.

그러나 SFSB가 Bean 관리 트랜잭션에 참여할 경우 Bean 메소드 실행 중간에 트랜잭션이 완결될 수 있습니다. 메소드 호출의 결과로 Bean의 상태가 전이 중일 수 있으므로 Bean의 상태에 검사점을 지정할 적절한 시간이 없습니다. Bean이 메소드가 끝날 때 EJB

컨테이너가 다른 트랜잭션의 범위에 있지 않는다고 가정할 경우, 해당 메소드 끝에 Bean의 상태에 검사점을 지정합니다. Bean 관리 트랜잭션이 여러 메소드에 걸쳐 있는 경우 후속 메소드 끝에 활성 트랜잭션이 나오지 않을 때까지 검사점 지정이 지연됩니다.

SFSB의 상태가 반드시 트랜잭션인 것은 아니며 비트랜잭션 비즈니스 메소드의 결과로 크게 바뀔 수 있습니다. 이 경우 [191 페이지 “검사점을 지정할 메소드 지정”](#)에 설명된 대로 검사점이 지정된 메소드 목록을 지정할 수 있습니다.

분산 가능 웹 응용 프로그램이 SFSB를 참조하고 웹 응용 프로그램의 세션이 페일오버되면 EJB 참조 또한 페일오버됩니다.

Application Server 인스턴스가 중지된 동안 세션 지속성을 사용하는 SFSB의 배포가 해제되면 지속성 저장소의 세션 데이터가 지워지지 않을 수 있습니다. 이를 방지하려면 Application Server 인스턴스가 실행 중인 동안 SFSB의 배포를 해제합니다.

EJB 컨테이너에 대한 가용성 구성

▼ EJB 컨테이너에 대한 가용성 활성화

- 1 EJB 컨테이너 가용성 탭을 선택합니다.
- 2 가용성 서비스 상자를 선택합니다.
가용성을 비활성화하려면 이 상자를 선택하지 마십시오.
- 3 [189 페이지 “가용성 설정”](#)에 설명된 대로 다른 설정을 변경합니다.
- 4 저장 버튼을 누릅니다.
- 5 서버 인스턴스를 다시 시작합니다.

자세한 정보 해당 asadmin 명령

EJB 컨테이너에 대한 가용성을 활성화하려면 `asadmin set` 명령을 사용하여 구성에 대해 다음의 세 가지 등록 정보를 설정합니다.

- `availability-service.ejb-container-availability.
availability-enabled`
- `availability-service.ejb-container-availability.
sfsb-persistence-type`

■

```
availability-service.ejb-container-availability.  
sfsb-ha-persistence-type
```

예를 들어, config1이 구성 이름이면 다음 명령을 사용합니다.

```
asadmin set --user admin --passwordfile password.txt --host localhost --port  
4849config1.availability-service.  
ejb-container-availability.availability-enabled="true"
```

```
asadmin set --user admin --passwordfile password.txt --host localhost --port  
4849config1.availability-service.  
ejb-container-availability.sfsb-persistence-type="file"
```

```
asadmin set --user admin --passwordfile password.txt --host localhost --port  
4849config1.availability-service.  
ejb-container-availability.sfsb-ha-persistence-type="ha"
```

가용성 설정

가용성 서비스에 대한 EJB 컨테이너 가용성 탭을 사용하여 다음 설정을 변경할 수 있습니다.

HA 지속성 유형: 가용성이 설정된 SFSB에 대해 세션 지속성 및 비활성화 메커니즘을 지정합니다. 허용되는 값은 file(파일 시스템) 및 ha(HADB)입니다. 세션 지속성이 필요한 프로덕션 환경의 경우 기본값인 ha를 사용합니다.

SFSB 지속성 유형: 가용성이 활성화되지 않은 SFSB에 대해 비활성화 메커니즘을 지정합니다. 허용되는 값은 file(파일 시스템) 및 ha입니다.

지속성 유형을 file로 설정한 경우 EJB 컨테이너는 비활성화된 Session Bean 상태가 저장되는 파일 시스템 위치를 지정합니다. 파일 시스템에 검사점을 지정하는 것은 테스트에는 유용하지만 프로덕션 환경에는 도움이 되지 않습니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 “저장소 등록 정보 구성”을 참조하십시오.

HA 지속성을 사용하면 서버 인스턴스에 장애가 발생할 경우 서버 인스턴스 클러스터가 SFSB 상태를 복구할 수 있습니다. HADB는 비활성 및 활성 저장소로 사용되기도 합니다. 이 옵션은 SFSB 상태 지속성이 필요한 프로덕션 환경에서 사용하십시오. 자세한 내용은 configure-ha-cluster(1)를 참조하십시오.

SFSB 저장소 풀 이름: 세션 지속성을 위해 HADB 연결에 사용한 JDBC 자원을 변경한 경우 SFSB 저장소 풀 이름을 변경할 수 있습니다. 자세한 내용은 configure-ha-cluster(1)를 참조하십시오.

가용성이 비활성화된 경우 SFSB 세션 저장소 구성

가용성이 비활성화되면 로컬 파일 시스템은 SFSB 지속성이 아닌 상태 비활성화에 사용됩니다. SFSB 상태가 저장되는 장소를 변경하려면 EJB 컨테이너의 세션 저장소 위치 설정을 변경합니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 “저장소 등록 정보 구성”을 참조하십시오.

개별 응용 프로그램 또는 EJB 모듈에 대한 가용성 구성

배포 중에 개별 응용 프로그램 또는 EJB 모듈에 대한 SFSB 가용성을 활성화할 수 있습니다.

- 관리 콘솔을 사용하여 배포를 수행할 경우 가용성 사용 가능 확인란을 선택합니다.
- `asadmin deploy` 또는 `asadmin deploydir` 명령을 사용하여 배포를 수행할 경우 `--availabilityenabled` 옵션을 `true`로 설정합니다. 자세한 내용은 `deploy(1)` 및 `deploydir(1)`을 참조하십시오.

개별 Bean에 대한 가용성 구성

개별 SFSB에 대해 가용성을 활성화하고 검사점을 지정할 메소드를 선택하려면 `sun-ejb-jar.xml` 배포 설명자 파일을 사용합니다.

고가용성 세션 지속성을 활성화하려면 `ejb` 요소에서 `availability-enabled="true"`를 설정합니다. SFSB 캐시의 크기 및 동작을 제어하려면 다음 요소를 사용합니다.

- `max-cache-size`: 캐시에 보관되는 세션 Bean의 최대 수를 지정합니다. 캐시가 오버플로될 경우(Beans의 수가 `max-cache-size` 초과) 컨테이너는 일부 Bean을 비활성화하거나 Bean의 일련화된 상태를 파일에 씁니다. 파일이 만들어진 디렉토리는 구성 API를 사용하여 EJB 컨테이너에서 가져옵니다.
- `resize-quantity`
- `cache-idle-timeout-in-seconds`
- `removal-timeout-in-seconds`
- `victim-selection-policy`

`sun-ejb-jar.xml`에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide**의 “The `sun-ejb-jar.xml` File”을 참조하십시오.

예 9-1 가용성이 활성화된 EJB 배포 설명자의 예

```
<sun-ejb-jar>
...
<enterprise-beans>
```

예 9-1 가용성이 활성화된 EJB 배포 설명자의 예 (계속)

```

...
<ejb availability-enabled="true">
  <ejb-name>MySFSB</ejb-name>
</ejb>
...
</enterprise-beans>
</sun-ejb-jar>

```

검사점을 지정할 메소드 지정

검사점을 활성화하면 트랜잭션이 롤백된 경우에도 일반적으로 Bean이 트랜잭션을 완료한 후에 검사점이 지정됩니다. Bean의 상태를 크게 바꾸는 비트랜잭션 비즈니스 메소드의 끝에 SFSB의 선택적 검사점을 지정하려면 sun-ejb-jar.xml 배포 설명자 파일의 ejb 요소에 있는 checkpoint-at-end-of-method 요소를 사용합니다.

checkpoint-at-end-of-method 요소의 비트랜잭션 메소드는 다음과 같을 수 있습니다.

- 작성 직후에 SFSB의 초기 상태에 검사점을 지정하려는 경우 SFSB의 홈 인터페이스에 정의된 create() 메소드
- 컨테이너 관리 트랜잭션만 사용하는 SFSB의 경우, 트랜잭션 속성 TX_NOT_SUPPORTED 또는 TX_NEVER로 표시된 Bean의 원격 인터페이스에 있는 메소드
- Bean 관리 트랜잭션만 사용하는 SFSB의 경우, Bean 관리 트랜잭션이 시작되지도 완결되지도 않은 메소드

이 목록에 언급된 다른 모든 메소드는 무시됩니다. 이러한 각 메소드의 호출이 끝나면 EJB 컨테이너는 SFSB의 상태를 지속형 저장소에 저장합니다.

주 - SFSB가 트랜잭션에 참여하지 않고 어떠한 메소드도 checkpoint-at-end-of-method 요소에 명시적으로 지정되지 않았으면 이 Bean에 대해 availability-enabled="true"인 경우에도 Bean의 상태는 검사되지 않습니다.

더 나은 성능을 위해 적은 수의 메소드를 지정하십시오. 이러한 메소드는 많은 양의 작업을 수행하거나 Bean의 상태를 크게 바꿉니다.

예 9-2 메소드 검사점을 지정하는 EJB 배포 설명자의 예

```

<sun-ejb-jar>
...
<enterprise-beans>
...
  <ejb availability-enabled="true">

```

예 9-2 메소드 검사점을 지정하는 EJB 배포 설명자의 예 (계속)

```
<ejb-name>ShoppingCartEJB</ejb-name>
<checkpoint-at-end-of-method>
  <method>
    <method-name>addToCart</method-name>
  </method>
</checkpoint-at-end-of-method>
</ejb>
...
</enterprise-beans>
</sun-ejb-jar>
```


Java Message Service 로드 균형 조정 및 페일오버

이 장에서는 Application Server에서 사용할 수 있도록 JMS(Java Message Service)의 로드 균형 조정 및 페일오버를 구성하는 방법을 설명합니다. 다음 항목으로 구성됩니다.

- 193 페이지 “Java Message Service 개요”
- 194 페이지 “Java Message Service 구성”
- 197 페이지 “연결 풀링 및 페일오버”
- 198 페이지 “Application Server에서 MQ 클러스터 사용”

Java Message Service 개요

Java Message Service(JMS) API는 J2EE 응용 프로그램 및 구성 요소가 메시지를 작성하고, 보내고, 받고, 읽을 수 있도록 하는 메시징 표준입니다. 또한 느슨하게 결합되고 안정적인 비동기식 분산 통신을 가능하게 합니다. JMS를 구현하는 Sun Java System Message Queue(MQ)는 Application Server와 긴밀하게 통합되어 MDB(Message-Driven Bean)와 같은 구성 요소를 만들 수 있도록 합니다.

MQ는 J2EE Connector Architecture Specification 1.5에 의해 정의된 자원 어댑터라고도 하는 커넥터 모듈을 사용하여 Application Server와 통합됩니다. Application Server로 배포된 J2EE 구성 요소는 커넥터 모듈을 통해 통합된 JMS 공급자를 사용하여 JMS 메시지를 교환합니다. Application Server에서 JMS 자원을 만들면 백그라운드에서 커넥터 자원이 만들어집니다. 따라서 각 JMS 작업은 커넥터 런타임을 호출하며 백그라운드에서 MQ 자원 어댑터를 사용합니다.

관리 콘솔이나 `asadmin` 명령줄 유틸리티를 사용하여 Java Message Service를 관리할 수 있습니다.

샘플 응용 프로그램

`mqfailover` 샘플 응용 프로그램은 JMS 항목에서 들어오는 메시지를 받는 Message Driven Bean이 있는 MQ 페일오버를 보여 줍니다. 이 샘플에는 MDB와 응용 프로그램

클라이언트가 포함되어 있습니다. Application Server는 MDB의 가용성을 높입니다. 한 브로커가 다운되면 대화 상태(MDB에 수신된 메시지)가 클러스터의 다른 가용 브로커 인스턴스로 투명하게 마이그레이션됩니다.

이 샘플은 다음 위치에 설치됩니다.

`install_dir/samples/ee-samples/failover/apps/mqfailover`

추가 정보

JMS 자원 구성에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 4 장, “JMS(Java Message Service) 자원 구성”을 참조하십시오. JMS에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer’s Guide**의 14 장, “Using the Java Message Service”를 참조하십시오. 커넥터(자원 어댑터)에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer’s Guide**의 9 장, “Developing Connectors”.

Sun Java System Message Queue에 대한 자세한 내용은 Message Queue 설명서(http://docs.sun.com/app/docs/coll/MessageQueue_05q1 및 http://docs.sun.com/app/docs/coll/MessageQueue_05q1_ko)를 참조하십시오. JMS API에 대한 일반 정보는 **JMS 웹 페이지** (<http://java.sun.com/products/jms/index.html>)를 참조하십시오.

Java Message Service 구성

Java Message Service 구성은 Sun Java System Application Server 클러스터 또는 인스턴스에 대한 모든 인바운드 및 아웃바운드 연결에서 사용할 수 있습니다. 다음에서 Java Message Service를 구성할 수 있습니다.

- 관리 콘솔. 적절한 구성을 사용하여 Java Message Service 구성 요소를 엽니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 4 장, “JMS(Java Message Service) 자원 구성”을 참조하십시오.
- `asadmin set` 명령. 다음 속성을 설정할 수 있습니다.

```
server.jms-service.init-timeout-in-seconds = 60
server.jms-service.type = LOCAL
server.jms-service.start-args =
server.jms-service.default-jms-host = default_JMS_host
server.jms-service.reconnect-interval-in-seconds = 60
server.jms-service.reconnect-attempts = 3
server.jms-service.reconnect-enabled = true
server.jms-service.addresslist-behavior = random
server.jms-service.addresslist-iterations = 3
server.jms-service.mq-scheme = mq
server.jms-service.mq-service = jms
```

다음 등록 정보도 설정할 수 있습니다.

```
server.jms-service.property.instance-name = imqbroker
server.jms-service.property.instance-name-suffix =
server.jms-service.property.append-version = false
```

Java Message Service 속성 및 등록 정보를 모두 나열하려면 `asadmin get` 명령을 사용합니다. `asadmin get`에 대한 자세한 내용은 `get(1)`을 참조하십시오. `asadmin set`에 대한 자세한 내용은 `set(1)`을 참조하십시오.

JMS 연결 팩토리 설정을 사용하여 Java Message Service 구성을 대체할 수 있습니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 “JMS 연결 팩토리”를 참조하십시오.

주 - Java Message Service의 구성을 변경한 후에 Application Server 인스턴스를 다시 시작해야 합니다.

JMS 관리에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 4 장, “JMS(Java Message Service) 자원 구성”을 참조하십시오.

Java Message Service 통합

MQ는 관리 콘솔의 Java Message Service Type 속성에 표시되는 LOCAL 및 REMOTE의 두 가지 방법으로 Application Server에 통합할 수 있습니다.

LOCAL Java Message Service

Type 속성이 LOCAL(독립 실행형 Application Server 인스턴스의 기본값)이면 Application Server는 기본 JMS 호스트로 지정된 MQ 브로커를 시작했다가 중지합니다. 독립 실행형 Application Server 인스턴스에는 LOCAL 유형이 가장 적절합니다.

Application Server 인스턴스와 Message Queue 브로커 사이에 일대일 관계를 만들려면 이 유형을 LOCAL로 설정하고 각 Application Server 인스턴스에 다른 기본 JMS 호스트를 부여합니다. 클러스터가 Application Server에 정의되는지 또는 MQ에 정의되는지에 관계없이 이 작업을 수행할 수 있습니다.

LOCAL 유형을 사용할 경우 Start Arguments 속성을 사용하여 MQ 브로커 시작 매개 변수를 지정합니다.

REMOTE Java Message Service

Type 속성이 REMOTE이면 MQ 브로커를 따로 시작해야 합니다. 이 값은 클러스터가 Application Server에 정의된 경우 기본값입니다. 브로커 시작에 대한 자세한 내용은 **Sun Java System Message Queue Administration Guide**를 참조하십시오.

이 경우 Application Server에서는 외부에 구성된 브로커나 브로커 클러스터를 사용합니다. 또한 Application Server와는 별도로 MQ 브로커를 시작 및 중지해야 하며 MQ 도구를 사용하여 브로커 또는 브로커 클러스터를 구성하고 조정해야 합니다. REMOTE 유형은 Application Server 클러스터에 가장 적합합니다.

REMOTE 유형을 사용할 경우 MQ 도구를 사용하여 MQ 브로커 시작 매개 변수를 지정해야 합니다. Start Arguments 속성은 무시됩니다.

JMS 호스트 목록

JMS 호스트는 MQ 브로커를 나타냅니다. Java Message Service에는 Application Server가 사용하는 모든 JMS 호스트가 들어 있는 **JMS 호스트 목록**(AddressList라고도 함)이 포함되어 있습니다.

JMS 호스트 목록은 지정된 MQ 브로커의 호스트 및 포트로 채워지며 JMS 호스트 구성이 변경될 때마다 업데이트됩니다. JMS 자원을 만들거나 MDB를 배포하면 해당 항목에 JMS 호스트 목록이 상속됩니다.

주 - Sun Java System Message Queue 소프트웨어에서 AddressList 등록 정보는 imqAddressList로 지정됩니다

기본 JMS 호스트

JMS 호스트 목록의 호스트 중 하나가 Default_JMS_host라는 기본 JMS 호스트로 지정됩니다. Java Message Service 유형이 LOCAL로 구성되면 Application Server 인스턴스가 기본 JMS 호스트를 시작합니다.

Sun Java System Message Queue 소프트웨어에서 다중 브로커 클러스터를 만든 경우 기본 JMS 호스트를 삭제하고 Message Queue 클러스터의 브로커를 JMS 호스트로 추가합니다. 이 경우 기본 JMS 호스트는 JMS 호스트 목록의 첫 번째 호스트가 됩니다.

Application Server는 Message Queue 클러스터를 사용할 때 기본 JMS 호스트에서 Message Queue 특정 명령을 실행합니다. 예를 들어, 세 개의 브로커로 구성된 Message Queue 클러스터에 대해 하나의 물리적 대상이 만들어질 때 물리적 대상을 만드는 명령은 기본 JMS 호스트에서 실행되지만 클러스터의 세 브로커 모두에서 해당 물리적 대상이 사용됩니다.

JMS 호스트 만들기

다음 방법으로 추가 JMS 호스트를 만들 수 있습니다.

- 관리 콘솔을 사용합니다. 적절한 구성에서 Java Message Service 구성 요소를 열고 JMS 호스트 구성 요소를 선택한 후 새로 만들기를 누릅니다. 자세한 내용은 관리 콘솔 온라인 도움말을 참조하십시오.

- `asadmin create-jms-host` 명령을 사용합니다. 자세한 내용은 `create-jms-host(1)`를 참조하십시오.

JMS 호스트 목록은 JMS 호스트 구성이 변경될 때마다 업데이트됩니다.

연결 풀링 및 페일오버

Application Server는 JMS 연결 풀링 및 페일오버를 지원합니다. Sun Java System Application Server는 JMS 연결을 자동으로 풀링합니다. 주소 목록 동작 속성이 `random`(기본값)이면 Application Server는 JMS 호스트 목록에서 무작위로 기본 브로커를 선택합니다. 페일오버가 발생하면 MQ는 로드를 투명하게 다른 브로커로 전송하고 JMS 의미를 유지 관리합니다.

To specify whether the Application Server가 연결이 끊어졌을 때 기본 브로커에 다시 연결하려고 시도할지 여부를 지정하려면 다시 연결 확인란을 선택합니다. 이 옵션을 선택한 상태에서 기본 브로커가 다운되면 Application Server는 JMS 호스트 목록의 다른 브로커로 다시 연결하려고 시도합니다.

다시 연결을 선택한 경우 다음 속성도 지정하십시오.

- **주소 목록 동작:** 연결 시도가 JMS 주소 목록에 있는 주소 순서대로 수행될지(priority) 또는 무작위로 수행될지(random) 여부. Priority로 설정하면 Java Message Service는 JMS 호스트 목록에 지정된 첫 번째 MQ 브로커에 연결하려고 시도한 후 첫 번째 브로커를 사용할 수 없는 경우에만 다른 브로커를 사용합니다. Random으로 설정하면 Java Message Service는 JMS 호스트 목록에서 무작위로 MQ 브로커를 선택합니다. 동일한 연결 팩토리를 사용하여 연결을 시도하는 클라이언트가 많을 경우 이 설정을 사용하여 클라이언트가 모두 동일한 주소에 연결을 시도하지 못하도록 합니다.
- **주소 목록 반복:** Java Message Service가 연결을 설정하거나 다시 설정하기 위해 JMS 호스트 목록 전체를 반복하는 횟수. -1 값은 횟수에 제한이 없음을 나타냅니다.
- **다시 연결 시도:** 클라이언트 런타임에서 JMS 호스트 목록의 다음 주소로 연결을 시도하기 전에 각 주소에 연결 또는 다시 연결을 시도하는 횟수. -1 값은 재연결 시도 횟수에 제한이 없음을 나타냅니다. 클라이언트 런타임은 성공할 때까지 첫 번째 주소에 연결을 시도합니다.
- **다시 연결 간격:** 다시 연결 시도 간격(초). JMS 호스트 목록의 각 주소에 대한 다시 연결 시도와 목록의 후속 주소에 이 값이 적용됩니다. 이 시간 간격이 너무 짧을 경우 브로커에게 복구할 시간이 없습니다. 너무 길 경우에는 지연이 지나치게 길게 느껴질 수 있습니다.

JMS 연결 팩토리 설정을 사용하여 이러한 설정을 대체할 수 있습니다. 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 관리 설명서**의 “JMS 연결 팩토리”를 참조하십시오.

로드 균형 조정된 메시지 흐름

Application Server에서는 동일한 ClientID를 갖는 MDB에 무작위로 메시지를 전달합니다. 영구 가입자에게는 ClientID가 필요합니다.

ClientID가 구성되지 않은 비영구 가입자의 경우 동일한 항목에 가입된 특정 MDB의 모든 인스턴스가 동일한 것으로 간주됩니다. MDB가 Application Server의 여러 인스턴스에 배포되면 MDB 중 하나만 메시지를 받습니다. 여러 고유 MDB가 동일한 항목에 가입하면 각 MDB의 한 인스턴스가 메시지 복사본을 받습니다.

동일한 대기열을 사용하는 여러 사용자를 지원하려면 물리적 대상의 `maxNumActiveConsumers` 등록 정보를 큰 값으로 설정하십시오. 이 등록 정보를 설정하면 MQ는 지정된 최대 수의 MDB가 동일한 대기열의 메시지를 사용할 수 있도록 허용합니다. 메시지는 MDB에 무작위로 배달됩니다. `maxNumActiveConsumers`가 -1로 설정되면 사용자 수에 제한이 없습니다.

Application Server에서 MQ 클러스터 사용

MQ Enterprise Edition은 브로커 클러스터로 알려져 있는 상호 연결된 여러 브로커 인스턴스를 지원합니다. 브로커 클러스터를 사용하면 클라이언트 연결이 클러스터에 있는 모든 브로커 간에 분산됩니다. 클러스터링은 수평적 확장을 제공하며 가용성을 향상시킵니다.

이 절에서는 고가용성 Sun Java System Message Queue 클러스터를 사용하도록 Application Server를 구성하는 방법을 설명합니다. 또한 Message Queue 클러스터를 시작하고 구성하는 방법을 설명합니다.

Application Server 토폴로지 및 MQ 배포에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Deployment Planning Guide**의 “Planning Message Queue Broker Deployment”를 참조하십시오.

▼ Application Server 클러스터가 있는 MQ 클러스터 활성화

1 Application Server 클러스터가 아직 없으면 만듭니다.

클러스터를 만드는 방법에 대한 자세한 내용은 [146 페이지 “클러스터 만들기”](#)를 참조하십시오.

2 MQ 브로커 클러스터를 만듭니다.

먼저 도메인 관리 서버에 의해 시작된 브로커를 참조하는 기본 JMS 호스트를 삭제한 후 MQ 브로커 클러스터에 포함될 세 개의 외부 브로커(JMS 호스트)를 만듭니다.

관리 콘솔이나 `asadmin` 명령줄 유틸리티에서 JMS 호스트를 만듭니다.

`asadmin`을 사용하려면 다음과 같이 명령을 지정합니다.

```
asadmin delete-jms-host --target cluster1 default_JMS_host
asadmin create-jms-host --target cluster1
    --mqhost myhost1 --mqport 6769
    --mquser admin --mqpassword admin broker1
asadmin create-jms-host --target cluster1
    --mqhost myhost2 --mqport 6770
    --mquser admin --mqpassword admin broker2
asadmin create-jms-host --target cluster1
    --mqhost myhost3 --mqport 6771
    --mquser admin --mqpassword admin broker3
```

관리 콘솔을 사용하여 호스트 만들기

- a. JMS 호스트 노드(구성 > *config-name* > Java Message Service > JMS 호스트)로 이동합니다.
- b. 기본 브로커(`default_JMS_host`)를 삭제합니다.
해당 브로커 옆의 확인란을 선택한 후 삭제를 누릅니다.
- c. 새로 만들기를 눌러 각 JMS 호스트를 만들고 해당 등록 정보 값을 입력합니다.
호스트 이름, DNS 이름 또는 IP 주소, 포트 번호, 관리 사용자 이름 및 암호 값을 채웁니다.

3 마스터 MQ 브로커 및 기타 MQ 브로커를 시작합니다.

JMS 호스트 시스템에서 세 개의 외부 브로커를 시작하고 임의의 시스템에서 하나의 마스터 브로커를 시작합니다. 이 마스터 브로커는 브로커 클러스터에 속할 필요가 없습니다. 예를 들면 다음과 같습니다.

```
/usr/bin/imqbrokerd -tty -name broker1 -port 6772
    -cluster myhost1:6769,myhost2:6770,myhost2:6772,myhost3:6771
    -D"imq.cluster.masterbroker=myhost2:6772"
```

4 클러스터의 Application Server 인스턴스를 시작합니다.

5 클러스터에 JMS 자원을 만듭니다.

a. JMS 물리적 대상을 만듭니다.

예를 들어, `asadmin`을 사용하려면 다음과 같이 지정합니다.

```
asadmin create-jmsdest --desttype queue --target cluster1 MyQueue
asadmin create-jmsdest --desttype queue --target cluster1 MyQueue1
```


관리 콘솔을 사용하려면

i. **JMS 호스트 페이지**(구성 > *config-name* > **Java Message Service** > 물리적 대상)로 이동합니다.

ii. 새로 만들기를 눌러 각 JMS 물리적 대상을 만듭니다.

iii. 각 대상에 대해 이름 및 유형(queue)을 입력합니다.

b. JMS 연결 팩토리를 만듭니다.

예를 들어, *asadmin*을 사용하려면 다음과 같이 지정합니다.

```
asadmin create-jms-resource --target cluster1
--restype javax.jms.QueueConnectionFactory jms/MyQcf
asadmin create-jms-resource --target cluster1
--restype javax.jms.QueueConnectionFactory jms/MyQcf1
```

관리 콘솔을 사용하려면

i. **JMS 연결 팩토리 페이지**(자원 > JMS 자원 > 연결 팩토리)로 이동합니다.

ii. 각 연결 팩토리를 만들려면 새로 만들기를 누릅니다.

JMS 연결 팩토리 만들기 페이지가 열립니다.

iii. 각 연결 팩토리에 대해 JNDI 이름(예: *jms/MyQcf*) 및 유형 *javax.jms.QueueConnectionFactory*를 입력합니다.

iv. 페이지 맨 아래에 나오는 사용 가능한 대상 목록에서 클러스터를 선택하고 추가를 누릅니다.

v. 확인을 눌러 연결 팩토리를 만듭니다.

c. JMS 대상 자원을 만듭니다.

예를 들어, *asadmin*을 사용하려면 다음과 같이 지정합니다.

```
asadmin create-jms-resource --target cluster1
--restype javax.jms.Queue
--property imqDestinationName=MyQueue jms/MyQueue
asadmin create-jms-resource --target cluster1
--restype javax.jms.Queue
--property imqDestinationName=MyQueue1 jms/MyQueue1
```

관리 콘솔을 사용하려면

i. **JMS 대상 자원 페이지**(자원 > JMS 자원 > 연결 팩토리)로 이동합니다.

ii. 각 대상 자원을 만들려면 새로 만들기를 누릅니다.

JMS 대상 자원 만들기 페이지가 열립니다.

iii. 각 대상 자원에 대해 JNDI 이름(예: jms/MyQueue) 및 유형 javax.jms.Queue를 입력합니다.

iv. 페이지 맨 아래에 나오는 사용 가능한 대상 목록에서 클러스터를 선택하고 추가를 누릅니다.

v. 확인을 눌러 대상 자원을 만듭니다.

6 -retrieve 옵션을 지정하여 응용 프로그램 클라이언트에 응용 프로그램을 배포합니다. 예를 들면 다음과 같습니다.

```
asadmin deploy --target cluster1
--retrieve /opt/work/MQapp/mdb-simple3.ear
```

7 응용 프로그램에 액세스한 후 예상대로 동작하는지 테스트합니다.

8 Application Server를 기본 JMS 구성으로 되돌리려면 만든 JMS 호스트를 삭제하고 기본 JMS 호스트를 다시 만듭니다. 예를 들면 다음과 같습니다.

```
asadmin delete-jms-host --target cluster1 broker1
asadmin delete-jms-host --target cluster1 broker2
asadmin delete-jms-host --target cluster1 broker3
asadmin create-jms-host --target cluster1
--mqhost myhost1 --mqport 7676
--mquser admin --mqpassword admin
default_JMS_host
```

관리 콘솔에서도 동일한 작업을 수행할 수 있습니다.

일반 오류 문제가 발생하면 다음을 참조하십시오.

- Application Server 로그 파일을 검토합니다. 로그 파일에 MQ 브로커가 메시지에 응답하지 않는 것으로 나오면 브로커를 중지했다가 다시 시작합니다.
- 항상 MQ 브로커를 시작한 후 Application Server 인스턴스를 시작합니다.
- Java Message Service에서 기본값을 지정할 경우 모든 MQ 브로커가 다운되면 Application Server가 다운되거나 다시 작동되는 데 30분 정도 소요됩니다. Java Message Service에서 이 시간 초과 값을 적절히 조정하십시오. 예를 들면 다음과 같습니다.

```
asadmin set --user admin --password administrator
cluster1.jms-service.reconnect-interval-in-seconds=5
```


RMI-IIOP 로드 균형 조정 및 페일오버

이 장에서는 RMI-IIOP를 통해 원격 EJB 참조 및 JNDI 객체에 대해 Sun Java System Application Server의 고가용성 기능을 사용하는 방법을 설명합니다.

- 203 페이지 “개요”
- 205 페이지 “RMI-IIOP 로드 균형 조정 및 페일오버 설정”

개요

RMI-IIOP 로드 균형 조정 기능을 사용하면 IIOP 클라이언트 요청이 여러 다른 서버 인스턴스나 이름 서버에 분산됩니다. 목표는 클러스터 내에서 로드를 균일하게 분산시켜 확장성을 제공하는 것입니다. IIOP 로드 균형 조정 기능과 EJB 클러스터링 및 가용성을 함께 사용하면 EJB 페일오버가 구현됩니다.

클라이언트가 객체에 대해 JNDI 조회를 수행하면 이름 지정 서비스는 특정 서버 인스턴스와 관련된 `InitialContext(IC)` 객체를 만듭니다. 그러면 해당 IC 객체를 사용하여 수행된 모든 조회 요청은 동일한 서버 인스턴스로 보내집니다. 해당 `InitialContext`를 사용하여 조회된 모든 `EJBHome` 객체는 동일한 대상 서버에 호스트됩니다. 이후에 가져온 모든 `Bean` 참조 또한 동일한 대상 호스트에서 만들어집니다. 이 경우 `InitialContext` 객체를 만들 때 모든 클라이언트가 활성 대상 서버 목록을 임의화하므로 로드 균형 조정이 효과적으로 구현될 수 있습니다. 대상 서버 인스턴스가 작동 중단되면 조회 또는 EJB 메소드 호출은 다른 서버 인스턴스로 페일오버됩니다.

IIOP 로드 균형 조정 및 페일오버는 투명하게 발생합니다. 응용 프로그램 배포 중에 특별한 단계가 필요하지는 않습니다. 그러나 클러스터에서 새 인스턴스를 추가 또는 삭제해도 클러스터의 기존 클라이언트 뷰가 업데이트되지 않습니다. 이렇게 하려면 클라이언트의 종점 목록을 수동으로 업데이트해야 합니다.

요구 사항

Sun Java System Application Server Enterprise Edition에서는 다음이 모두 적용될 경우 RMI-IIOP를 통해 원격 EJB 참조 및 `NameService` 객체의 고가용성을 제공합니다.

- 배포에 적어도 두 개 이상의 응용 프로그램 서버 인스턴스로 구성된 클러스터가 있어야 합니다.
- 로드 균형 조정에 참여하는 모든 `Application Server` 인스턴스와 클러스터에 J2EE 응용 프로그램이 배포되어 있어야 합니다.
- RMI-IIOP 클라이언트 응용 프로그램이 로드 균형 조정을 사용하도록 되어 있어야 합니다.

`Application Server`에서는 `Application Server`에 배포된 EJB 구성 요소에 액세스하는 다음 RMI-IIOP 클라이언트에 대해 로드 균형 조정을 지원합니다.

- ACC(`Application Client Container`)에서 실행되는 Java 응용 프로그램. 205 페이지 “`Application Client Container`에 대해 RMI-IIOP 로드 균형 조정 설정”을 참조하십시오.
- ACC에서 실행되지 않는 Java 응용 프로그램. 207 페이지 “독립 실행형 클라이언트에 대해 RMI-IIOP 로드 균형 조정 및 페일오버 설정”을 참조하십시오.

주 - `Application Server`에서는 SSL(`Secure Sockets Layer`)을 통한 RMI-IIOP 로드 균형 조정 및 페일오버를 지원하지 않습니다.

알고리즘

`Application Server`에서는 RMI-IIOP 로드 균형 조정 및 페일오버에 대해 임의화 및 라운드 로빈 알고리즘을 사용합니다.

RMI-IIOP 클라이언트가 새로운 `InitialContext` 객체를 처음 만들면 해당 클라이언트에 대해 사용 가능한 `Application Server` IIOP 종점 목록이 임의화됩니다. `InitialContext` 객체의 경우 로드 밸런서는 조회 요청과 다른 `InitialContext` 작업을 목록의 첫 번째 종점으로 지정합니다. 첫 번째 종점을 사용할 수 없는 경우 목록의 두 번째 종점을 사용하는 식으로 계속합니다.

클라이언트가 계속해서 새로운 `InitialContext` 객체를 만들 때마다 종점 목록이 회전하므로 `InitialContext` 작업에 대해 다른 IIOP 종점이 사용됩니다.

`InitialContext` 객체에서 얻은 참조에서 Bean을 구하거나 만들 경우 해당 Bean은 `InitialContext` 객체에 할당된 IIOP 종점 역할을 하는 `Application Server` 인스턴스에 만들어집니다. 해당 Bean에 대한 참조에는 클러스터의 모든 `Application Server` 인스턴스의 IIOP 종점 주소가 포함됩니다.

기본 종점은 Bean을 조회하거나 만드는 데 사용된 `InitialContext` 종점에 해당하는 Bean 종점입니다. 클러스터의 다른 IIOP 종점은 **대체 종점**으로 지정됩니다. Bean의 기본 종점을 사용할 수 없게 되면 해당 Bean의 추가 요청이 대체 종점 중 하나로 페일오버됩니다.

You can configure ACC에서 실행되는 응용 프로그램 및 독립 실행형 Java 클라이언트에서 작동되도록 RMI-IIOP 로드 균형 조정 및 페일오버를 구성할 수 있습니다.

샘플 응용 프로그램

다음 디렉토리에는 ACC가 있거나 없는 RMI-IIOP 페일오버를 사용하는 것을 설명하는 샘플 응용 프로그램이 포함되어 있습니다.

`install_dir/samples/ee-samples/sfsbfailover`

ACC가 있거나 없는 응용 프로그램을 실행하는 방법에 대한 자세한 내용은 이 샘플과 함께 제공된 `index.html` 파일을 참조하십시오. `ee-samples` 디렉토리에는 샘플을 실행할 수 있도록 환경을 설정하는 데 필요한 정보도 포함되어 있습니다.

RMI-IIOP 로드 균형 조정 및 페일오버 설정

ACC(Application Client Container)에서 실행되는 응용 프로그램 및 독립 실행형 클라이언트 응용 프로그램에 대해 RMI-IIOP 로드 균형 조정 및 페일오버를 설정할 수 있습니다.

▼ Application Client Container에 대해 RMI-IIOP 로드 균형 조정 설정

이 절차에서는 ACC(Application Client Container)와 함께 RMI-IIOP 로드 균형 조정 및 페일오버를 사용하는 데 필요한 단계를 개략적으로 설명합니다. ACC에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide**의 “Developing Clients Using the ACC”를 참조하십시오.

- 1 `install_dir/bin` 디렉토리로 이동합니다.
- 2 `package-appclient`를 실행합니다.
이 유틸리티는 `appclient.jar` 파일을 생성합니다. `package-appclient`에 대한 자세한 내용은 `package-appclient(1M)`를 참조하십시오.
- 3 클라이언트에서 압축을 풀 시스템으로 `appclient.jar` 파일을 복사합니다.

- 4 해당 시스템의 올바른 디렉토리 값을 참조하도록 `asenv.conf` 또는 `asenv.bat` 경로 변수를 편집합니다.

파일은 `appclient-install-dir/config/`에 있습니다.

업데이트할 경로 변수 목록을 보려면 `package-appclient(1M)`를 참조하십시오.

- 5 필요하면 `appclient` 스크립트를 실행 파일로 만듭니다.

예를 들어, UNIX에서 `chmod 700`을 사용합니다.

- 6 클러스터의 인스턴스에 대한 IIOP Listener 포트 번호를 찾습니다.

IIOP Listener를 종점으로 지정하여 요청을 받을 IIOP Listener를 결정합니다. 관리 콘솔에 IIOP Listener 표시

a. 관리 콘솔의 트리 구성 요소에서 클러스터 노드를 확장합니다.

b. 클러스터를 확장합니다.

c. 클러스터의 인스턴스를 선택합니다.

d. 오른쪽 창에서 등록 정보 탭을 누릅니다.

특정 인스턴스에 대한 IIOP Listener 포트를 기록합니다.

e. 모든 인스턴스에 대해 이 프로세스를 반복합니다.

- 7 종점 값으로 `sun-acc.xml`을 입력합니다.

이전 단계의 IIOP Listener를 사용하여 다음 형식으로 종점 값을 만듭니다.

`machine1:instance1-iiop-port, machine2:instance2-iiop-port`

예를 들면 다음과 같습니다.

```
<property name="com.sun.appserv.iiop.endpoints"
value="host1.sun.com:3335,host2.sun.com:3333,host3.sun.com:3334">
```

- 8 `--retrieve` 옵션을 통해 클라이언트 응용 프로그램을 배포하여 클라이언트 jar 파일을 가져옵니다.

클라이언트 jar 파일을 클라이언트 시스템에 보관합니다.

예를 들면 다음과 같습니다.

```
asadmin deploy --user admin --passwordfile pw.txt --retrieve /my_dir myapp
```

- 9 다음과 같이 응용 프로그램 클라이언트를 실행합니다.

```
appclient -client clientjar -name appname
```

다음 순서 페일오버를 테스트하려면 클러스터의 한 인스턴스를 중지하고 응용 프로그램이 제대로 작동하는지 확인합니다. 클라이언트 응용 프로그램에 중단점(또는 일시 정지)이 있을 수도 있습니다.

로드 균형 조정 기능을 테스트하려면 여러 클라이언트를 사용하고 모든 종점에 로드가 분산되는 방식을 확인하십시오.

▼ 독립 실행형 클라이언트에 대해 RMI-IIOP 로드 균형 조정 및 페일오버 설정

- 1 `--retrieve` 옵션을 통해 응용 프로그램을 배포하여 클라이언트 jar 파일을 가져옵니다. 클라이언트 jar 파일을 클라이언트 시스템에 보관합니다.

예를 들면 다음과 같습니다.

```
asadmin deploy --user admin --passwordfile pw.txt --retrieve /my_dir myapp
```

- 2 종점 및 InitialContext를 `-D` 값으로 지정하여 클라이언트 jar 및 필수 jar 파일을 실행합니다.

예를 들면 다음과 같습니다.

```
java -Dcom.sun.appserv.iiope.endpoints=
host1.sun.com:33700,host2.sun.com:33700,host3.sun.com:33700
samples.rmiiopclient.client.Standalone_Client
```

다음 순서 페일오버를 테스트하려면 클러스터의 한 인스턴스를 중지한 후 응용 프로그램이 정상적으로 작동하는지 확인합니다. 클라이언트 응용 프로그램에 중단점(또는 일시 정지)이 있을 수도 있습니다.

로드 균형 조정 기능을 테스트하려면 여러 클라이언트를 사용하고 모든 종점에 로드가 분산되는 방식을 확인하십시오.

색인

A

- active-healthcheck-enabled, 125
- AddressList, 기본 JMS 호스트, 196
- Apache Web Server
 - Application Server 설치 프로그램에서 수정한 사항, 110
 - 로드 밸런서 포함, 104
 - 로드 밸런서 플러그인에서 수정한 사항, 107
 - 보안 파일, 111
- asadmin create-jms-host 명령, 197
- asadmin get 명령, 195
- asadmin set 명령, 194
- authPassthroughEnabled, 130

C

- cacheDatabaseMetaData 등록 정보, 77
- checkpoint-at-end-of-method 요소, 191
- Connection Validation Required 설정, 76
- ConnectionTrace 속성, 71
- CoreFile 속성, 72
- create-http-lb-config 명령, 121
- create-http-lb-ref 명령, 123
- create-node-agent 명령, 173

D

- Data Source Enabled 설정, 78
- Database Vendor 설정, 76
- DatabaseName 속성, 72

- databuf 옵션, 97
- DataBufferPoolSize 속성, 72
- DataDeviceSize 속성, 72, 87
- DataSource Classname 설정, 76
- dbpassword option, 61
- dbpasswordfile option, 61
- dbpasswordfile 옵션, 61
- default-config 구성, 156
- delete-http-lb-ref 명령, 123
- delete-node-agent 명령, 175
- DevicePath 속성, 72, 90
- devicepath 옵션, 66
- devicesize 옵션, 66
- disable-http-lb-application 명령, 127
- disable-http-lb-server 명령, 127
- Domain Administration Server
 - 노드 에이전트 동기화, 167
 - 서버 인스턴스 동기화, 168

E

- EagerSessionThreshold 속성, 72
- EagerSessionTimeout 속성, 72
- EJB 컨테이너, 가용성, 188-189
- eliminateRedundantEndTransaction 등록 정보, 77
- enable-http-lb-application 명령, 123
- enable-http-lb-server 명령, 123
- EventBufferSize 속성, 73
- export-http-lb-config 명령, 125

F

Fail All Connections 설정, 76
fast 옵션, 85

G

Global Transaction Support 설정, 76
Guarantee Isolation Level 설정, 76

H**HADB**

JDBC URL 가져오기, 75-76
nodes, 93
구성, 64-78
내역 파일, 101
노드 시작, 80
노드 재시작, 81
노드 중지, 81
노드 추가, 88
노드 확장, 87-88
데이터 손상, 86
데이터베이스 나열, 84
데이터베이스 시작, 82
데이터베이스 이름, 66
데이터베이스 재시작, 83
데이터베이스 제거, 85
데이터베이스 중지, 83
데이터베이스 지우기, 84
모니터링, 92-99
상태 가져오기, 92-94
설정 속성, 68
속성 설정, 70
시스템 유지 관리, 99
시스템 추가, 88
연결 풀 등록 정보, 76-77
연결 풀 설정, 76
이기종 장치 경로, 69
이중 네트워크, 36-37
자원 정보 가져오기, 96-99
장치 정보 가져오기, 95
재조각화, 90
포트 할당, 69

HADB (계속)

환경 변수, 62
HADB 관리 에이전트, 시작, 43, 51-59
HADB 구성
네트워크 구성, 34-37
노드 수퍼바이저 프로세스, 41-42
시간 동기화, 39-40
HADB 설정, 33
hadbm addnodes 명령, 88
hadbm clear 명령, 84
hadbm clearhistory 명령, 101
hadbm command, 59-64
hadbm create command, 65
hadbm delete 명령, 85
hadbm deviceinfo 명령, 95
hadbm get 명령, 70
hadbm list 명령, 84
hadbm refragment 명령, 90
hadbm resourceinfo 명령, 96-99
hadbm restart 명령, 83
hadbm restartnode 명령, 81
hadbm start 명령, 82
hadbm startnode 명령, 80
hadbm status 명령, 92-94
hadbm stop 명령, 83
hadbm stopnode 명령, 81
HistoryPath 속성, 73
historypath 옵션, 67
hosts 옵션, 67, 90
HTTP
HTTPS 라우팅, 128
세션 페일오버, 128-129
HTTP_LISTENER_PORT 등록 정보, 159
HTTP_SSL_LISTENER_PORT 등록 정보, 159
HTTP 세션, 21
분산, 179-180
HTTPS
라우팅, 128-129
세션 페일오버, 128-129
HTTPS 라우팅, 128-129

I

IIOPLISTENER_PORT 등록 정보, 159

IOP_SSL_MUTUALAUTH_PORT 등록 정보, 159
 InternalLogbufferSize 속성, 73
 IOP_SSL_LISTENER_PORT 등록 정보, 159

J

JdbcUrl 속성, 73
 JMS
 구성, 194
 연결 페일오버, 197
 연결 풀링, 22, 197
 호스트 만들기, 196
 JMS 호스트 목록, 연결, 196
 JMX Listener, 노드 에이전트, 177
 JMX_SYSTEM_CONNECTOR_PORT 등록
 정보, 159
 JNDI Name 설정, 78

L

loadbalancer.xml 파일, 125
 locks 옵션, 97
 logbuf 옵션, 97
 LogbufferSize 속성, 73

M

magnus.conf 파일, 웹 서버, 103
 maxStatement 등록 정보, 77
 MaxTables 속성, 73
 Microsoft 인터넷 정보 서비스(IIS), 로드 균형 조정을
 위한 수정 사항, 113

N

Name 설정, 76
 nilogbuf 옵션, 97
 no-refragment option, 89
 no-repair option, 81
 nodes 옵션, 93
 number-healthcheck-retries, 125

NumberOfDatadevices 속성, 73
 NumberOfLocks 속성, 73
 NumberOfSessions 속성, 73

O

obj.conf 파일, 웹 서버, 104

P

password 등록 정보, 76
 Pool Name 설정, 78
 Portbase 속성, 73
 portbase 옵션, 67

R

RelalgdeviceSize 속성, 73
 rewrite-location property, 132

S

saveto 옵션, 101
 serverList 등록 정보, 77
 SessionTimeout 속성, 73
 set 옵션, 68, 69
 spares 옵션, 67, 85, 90
 SQLTraceMode 속성, 74
 start-node-agent 명령, 174
 startlevel option, 82
 startlevel 옵션, 80
 StartRepairDelay 속성, 74
 Stateful Session Bean, 187
 세션 지속성, 187, 190
 Stateful Session Bean 상태의 검사점 지정, 182
 StatInterval 속성, 74
 Steady Pool Size 설정, 76
 stop-node-agent 명령, 174
 sun-ejb-jar.xml 파일, 191
 Sun Java System Message Queue, 커넥터, 194

Sun Java System Web Server, 로드 밸런서에서 수정한 사항, 103

sun-passthrough.properties 파일, 로그 수준, 143

sun-passthrough 등록 정보, 115

SyslogFacility 속성, 74

SysLogging 속성, 74

SysLogLevel 속성, 75

SyslogPrefix 속성, 75

T

Table Name 설정, 76

TakeoverTime 속성, 75

Transaction Isolation 설정, 76

U

username 등록 정보, 76

V

Validation Method 설정, 76

가

가용성

EJB 컨테이너 수준, 190-191

Stateful Session Bean, 187

수준, 182

웹 모듈, 179-180

활성화 및 비활성화, 182

검

검사점 지정, 187

메소드 선택, 187, 191

경

경로 쿠키, 122

고

고정 라운드 로빈 로드 균형 조정, 118

관

관리 콘솔

JMS 서비스 구성에 사용, 194

JMS 호스트 작성에 사용, 196

구

구성, 참조 명명된 구성

기

기본 종점, RMI-IIOP 페일오버, 205

네

네트워크 구성 요구 사항, 34-37

노

노드 슈퍼바이저 프로세스 및 고가용성, 41-42

노드 에이전트

Domain Administration Server와 동기화, 167

JMX Listener, 177

로그, 171

만들기, 173

배포, 165

삭제, 175, 176

설치, 166

시작, 174

인증 영역, 177

자리 표시자, 172

노드 에이전트 (계속)

정보, 163
중지, 174

다

다시 로드 간격, 122

단

단일 사인 온, 세션 지속성, 186-187

대

대상, 로드 밸런서 구성, 122
대체 종점, RMI-IIOP 페일오버, 205

데

데이터베이스 옵션, 66

동

동적 재구성, 로드 밸런서, 126

라

라운드 로빈 로드 균형 조정, 고정, 118

로

로그
노드 에이전트 로그 보기, 171
로드 밸런서, 140
로드 균형 조정
Apache Web Server, 104
HTTP, 정보, 117
HTTP 알고리즘, 118

로드 균형 조정 (계속)

Microsoft IIS, 113
RMI-IIOP 요구 사항, 204
Sun Java System Web Server, 103
고정 라운드 로빈, 118
구성 변경, 126
구성 파일 내보내기, 125
다중 웹 서버 인스턴스, 133
동적 재구성, 126
로그 메시지, 140
로드 밸런서 구성 만들기, 121
멥등원(Idempotent) URL, 133
상태 검사기, 124
서버 인스턴스 또는 클러스터 정지, 127
서버 인스턴스 활성화, 123
설정, 120
세션 페일오버, 128-129
역 프록시 플러그인으로 사용, 120
응용 프로그램 정지, 127
응용 프로그램 활성화, 123
참조 만들기, 123
할당된 요청, 118

롤

롤링 업그레이드, 134

멥

멥등원(Idempotent) URL, 133

명

명명된 구성
default-config, 156
공유, 156
기본 이름, 157
정보, 155
포트 번호, 157

배

배포, 배포 중에 가용성 설정, 182

분

분산 가능 웹 응용 프로그램, 182

분산된 HTTP 세션, 179-180

비

비정상 서버 인스턴스, 124

상

상태 검사기, 124

서

서버, 클러스터, 145

서버 인스턴스

로드 균형 조정을 위해 활성화, 123

정지, 127

세

세마포, 37

세션

HTTP, 21

지속성, 21

세션 저장소

HTTP 세션, 184

Stateful Session Bean, 189, 190

세션 지속성

Stateful Session Bean, 187, 190

단일 사인 온, 186-187

웹 모듈, 179-180

세션 페일오버, HTTP 및 HTTPS, 128-129

시

시간 동기화, 39-40

알

알고리즘

HTTP 로드 균형 조정, 118

RMI-IIOP 페일오버, 204

역

역 프록시 플러그인, 120

연

연결 풀

HADB에 대한 등록 정보, 76-77

HADB에 대해 설정, 76

영

영역, 노드 에이전트 인증, 177

웹

웹 서버, 다중 인스턴스 및 로드 균형 조정, 133

웹 응용 프로그램, 분산 가능, 182

웹 컨테이너, 가용성, 183

응

응답 시간 초과, 122

응용 프로그램

가용성 손실 없이 업그레이드, 134

로드 균형 조정을 위해 활성화, 123

정지, 127

인

인증 영역, 노드 에이전트, 177

정

정지

서버 인스턴스 또는 클러스터, 127

응용 프로그램, 127

종

종점, RMI-IIOP 페일오버, 205

중

중앙 저장소, 노드 에이전트 동기화, 167

지

지속성, 세션, 21

지속성 저장소, Stateful Session Bean 상태, 187

쿠

쿠키 기반 세션 고정, 118

클

클러스터, 145

공유, 25-26

독립 실행형, 25-26

정지, 127

클러스터링된 서버 인스턴스, 구성, 156

통

통과 플러그인, 120

트

트랜잭션

세션 지속성, 187, 191

페

페일오버

HTTP 정보, 117

JMS 연결, 197

RMI-IIOP 요구 사항, 204

Stateful Session Bean 상태, 187

웹 모듈 세션, 179-180

포

포트 번호, 구성, 157

할

할당되지 않은 요청, 118

할당된 요청, 118

