Sun Java System Application Server Enterprise Edition 8.2 관리 설명서

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

부품 번호: 820–0849
2007년 3월

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 재산권을 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.가 개발한 구조를 기반으로 하고 있습니다.

OPEN LOOK 및 Sun™ GUI(그래픽 사용자 인터페이스)는 Sun Microsystems, Inc.가 자사의 사용자 및 정식 사용자용으로 개발했습니다. Sun은 컴퓨터 업계를 위한 시각적 또는GUI의 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 제외 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.

070430@17039

목차

그림

표

코드 예

머리말

Sun Java System Application Server Enterprise Edition 8.2 **관리 설명서**에서는 구성, 모니터링, 보안, 자원 관리 및 웹 서비스 관리를 포함한 Application Server의 시스템 관리에 대해 설명합니다.

이 머리말에서는 전체 Sun Java™ System Application Server에 대한 정보 및 규칙에 대해 설명합니다.

Application Server 설명서 모음

Application Server 설명서 모음에서는 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` 및 `http://docs.sun.com/app/docs/coll/1401.2`입니다. Application Server에 대한 소개 내용을 보려면 다음 표에 나열된 설명서를 순서대로 참조하십시오.

표 P–1 Application Server 설명서 모음의 구성

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	Application Server 제품을 시작하는 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	J2EE 구성 요소 및 API용 개방형 Java 표준 모델을 따르는 Application Server에서 실행할 Java 2 Platform, Enterprise Edition(J2EE™ 플랫폼) 응용 프로그램을 만들고 구현하는 방법에 대해 설명합니다. 개발자 도구, 보안, 디버깅, 배포 및 라이프사이클 모듈 생성에 대한 정보를 제공합니다.

표 P–1 Application Server 설명서 모음의 구성 (계속)

설명서 제목	설명
J2EE 1.4 Tutorial	J2EE 1.4 플랫폼 기술 및 API를 사용하여 J2EE 응용 프로그램을 개발하는 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.xml`을 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 않는 결과를 가져올 수 있는 제품 릴리스 및 구성 옵션의 차이점에 대한 설명도 포함되어 있습니다.
Performance Tuning Guide	Application Server를 조정하여 성능을 향상시키는 방법에 대해 설명합니다.
Troubleshooting Guide	Application Server 문제를 해결하는 방법에 대해 설명합니다.
Error Message Reference	Application Server 오류 메시지를 해결하는 방법에 대해 설명합니다.
Reference Manual	Application Server에 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` 및 `http://docs.sun.com/coll/1397.2`의 시스템 설명서를 잘 이해해야 합니다. Java ES 및 해당 구성 요소에 대한 모든 설명서를 볼 수 있는 URL은 `http://docs.sun.com/prod/entsys.5` 및 `http://docs.sun.com/prod/entsys.5?l=ko`입니다.

다른 Sun Java System 서버 설명서를 보려면 다음 설명서를 참조하십시오.

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

또한 다음과 같은 자원도 유용할 수 있습니다.

- J2EE 1.4 Specifications (`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

install-dir	Application Server의 기본 설치 디렉토리를 나타냅니다.	Solaris™ 플랫폼에 Sun Java Enterprise System(Java ES)을 설치한 경우:
		`/opt/SUNWappserver/appserver`
		Linux 플랫폼에 Java ES를 설치한 경우:
		`/opt/sun/appserver/`
		기타 Solaris 및 Linux 설치, 루트가 아닌 사용자의 경우:
		user's home directory`/SUNWappserver`
		기타 Solaris 및 Linux 설치, 루트 사용자의 경우:
		`/opt/SUNWappserver`
		Windows, 모든 설치
		SystemDrive`:\Sun\AppServer`

domain-root-dir	모든 도메인을 포함하는 디렉토리를 나타냅니다.	Solaris 플랫폼에 Java ES를 설치한 경우:
		`/var/opt/SUNWappserver/domains/`
		Linux 플랫폼에 Java ES를 설치한 경우:
		`/var/opt/sun/appserver/domains/`
		다른 모든 설치:
		install-dir`/domains/`

domain-dir	도메인용 디렉토리를 나타냅니다. 구성 파일에서 *domain-dir*이 다음과 같이 표시되어 있을 수 있습니다. `${com.sun.aas.instanceRoot}`	*domain-root-dir*`/`*domain-dir*
instance-dir	서버 인스턴스용 디렉토리를 나타냅니다.	*domain-dir*`/`*instance-dir*

활자체 규약

다음 표에서는 본 설명서에 사용된 활자체 변경 사항에 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 -a`를 사용하여 모든 파일을 나열하십시오. `machine_name% you have mail.`
`AaBbCc123`	컴퓨터 화면 상의 출력과는 달리 사용자가 직접 입력하는 사항입니다.	`machine_name%` **`su`** `Password:`
AaBbCc123	실제 이름이나 값으로 대체되는 자리 표시자입니다.	파일을 삭제하려면 `rm` *filename*을 입력하십시오.
AaBbCc123	책 제목, 새로 나오는 용어 및 강조 표시할 용어입니다(일부 강조된 항목은 온라인상에서 볼드로 표시).	**사용자 설명서**의 6장을 읽으십시오. **캐시**는 로컬에 저장된 복사본입니다. 파일을 저장하지 **마십시오**.

기호 규약

다음 표에서는 본 설명서에 사용된 기호에 대해 설명합니다.

표 P–4 기호 규약

기호	설명	예	의미
`[ ]`	선택적 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.	Control-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 규약 (계속)

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

설명서, 지원 및 교육

Sun 웹 사이트에서는 다음 추가 자원에 대한 정보를 제공합니다.

- 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 보내려면 http://docs.sun.com에서 의견 보내기를 누르십시오. 온라인 양식에서 전체 문서 제목과 부품 번호를 기입해 주십시오. 부품 번호는 해당 설명서의 제목 페이지나 문서 URL에 있으며 일반적으로 7자리 또는 9자리 숫자입니다. 예를 들어, 이 설명서의 부품 번호는 820-0849입니다.

◆ ◆ ◆ 1 장

1 시작하기

이 장에서는 Sun Java System Application Server 관리에 대해 설명합니다. Application Server 관리에는 응용 프로그램 배포와 도메인, 서버 인스턴스, 자원 만들기 및 구성, 도메인 및 서버 인스턴스 제어(시작 및 중지), 성능 모니터링 및 관리, 문제 진단 및 문제 해결과 같은 여러 작업이 포함됩니다. 이 장은 다음 내용으로 구성되어 있습니다.

Sun Java System Application Server 정보

Sun Java System Application Server는 서버측 Java 응용 프로그램 및 웹 서비스를 개발하고 배포하기 위한 Java 2 Platform, Enterprise Edition(J2EE 플랫폼) 1.4 호환 플랫폼을 제공합니다. 주요 기능으로는 확장 가능한 트랜잭션 관리, 컨테이너 관리 지속성 런타임, 성능 웹 서비스, 클러스터링, 고가용성, 보안 및 통합 기능이 있습니다.

Application Server는 다음 개정판에서 사용할 수 있습니다.

- Platform Edition은 무료이며 소프트웨어 개발 및 부서 내 프로덕션 환경을 위한 것입니다.
- Enterprise Edition은 업무에 중요한 서비스 및 대규모 프로덕션 환경을 위해 설계되었습니다. 로드 밸런서 플러그인과의 수평 확장성 및 서비스 연속성과 클러스터 관리를 지원합니다. 또한 Enterprise Edition은 고가용성 데이터베이스(HADB)에 대해 신뢰할 수 있는 세션 상태 관리를 지원합니다.

이 절은 다음 내용으로 구성되어 있습니다.

Application Server란?

Application Server는 웹 게시부터 엔터프라이즈급 트랜잭션 처리까지의 서비스를 지원하는 한편, 개발자가JPS(JavaServer Pages), Java Servlet 및 EJB(Enterprise JavaBeans) 기술을 바탕으로 응용 프로그램을 작성할 수 있도록 해주는 플랫폼입니다.

Application Server Platform Edition은 개발, 프로덕션 배포 및 재배포가 **무료**입니다. 재배포에 대한 자세한 내용은 `http://www.sun.com/software/products/appsrvr/appsrvr_oem.xml`을 참조하십시오.

Application Server Enterprise Edition에서는 고급 클러스터링 및 페일오버 기술을 제공합니다. Application Server 인프라는 여러 유형의 배포된 응용 프로그램에 대한 배포를 지원하며 서비스 지향 아키텍처(SOA)를 기반으로 응용 프로그램을 구축하는 데 이상적인 기초가 됩니다. SOA는 응용 프로그램 서비스의 재사용을 최대화할 목적으로 설계된 방법 체계입니다. 이 기능을 사용하면 확장 가능한 고가용성 J2EE 응용 프로그램을 실행할 수 있습니다.

- **확장성** - 확장성은 클러스터링을 통해 얻을 수 있습니다. 클러스터는 하나의 논리적 엔티티로 함께 작동하는 Application Server 인스턴스 그룹입니다. 클러스터의 모든 Application Server 인스턴스에는 동일한 구성과 동일한 응용 프로그램이 배포됩니다.

  Application Server 인스턴스를 클러스터에 추가하면 시스템 용량이 늘어나서 수평 확장할 수 있습니다. 서비스를 중단하지 않고 Application Server 인스턴스를 클러스터에 추가할 수 있습니다. HTTP, RMI/IIOP 및 JMS 로드 균형 조정 시스템에서는 클러스터 내에서 정상적으로 작동하는 Application Server 인스턴스로 요청을 분산합니다.
- **고가용성** - 가용성은 페일오버 기능을 참조합니다. 하나의 서버 인스턴스가 중지될 경우 클러스터의 다른 서버 인스턴스가 실패한 인스턴스의 세션을 넘겨받아 끊김 없이 클라이언트에 대한 서비스를 계속 수행합니다. 세션 정보는 고가용성 데이터베이스(HADB)에 저장됩니다. HADB는 HTTP 세션 및 Stateful Session Bean의 지속성을 지원합니다.

Application Server 구조

이 절에서는 Application Server의 상위 구조를 보여주는 그림 1–1에 대해 설명합니다.

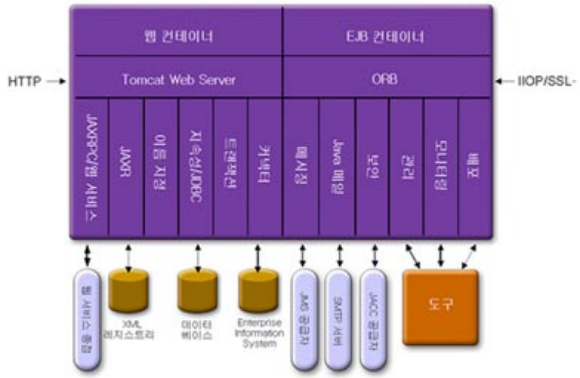

그림 1–1 Application Server 구조

- **컨테이너** - 컨테이너는 보안 및 트랜잭션 관리와 같은 서비스를 J2EE 구성 요소에 제공하는 런타임 환경입니다. 그림 1–1은 두 가지 유형(웹 및 EJB)의 J2EE 컨테이너를 보여줍니다. JSP 페이지와 서블릿 같은 웹 구성 요소는 웹 컨테이너 내에서 실행됩니다. Enterprise Java Bean은 EJB 컨테이너 내에서 실행됩니다.
- **클라이언트 액세스** - 런타임 시 브라우저 클라이언트는 인터넷에서 사용되는 프로토콜인 HTTP를 통해 웹 서버와 통신하여 웹 응용 프로그램에 액세스합니다. HTTPS 프로토콜은 보안 통신을 요구하는 응용 프로그램용입니다. Enterprise Java Bean 클라이언트는 IIOP 또는 IIOP/SSL(보안) 프로토콜을 통해 ORB(Object Request Broker)와 통신합니다. Application Server에는 HTTP, HTTPS, IIOP 및 IIOP/SSL 프로토콜에 대한 별도의 Listener가 있습니다. Listener마다 특정한 포트 번호를 독점적으로 사용합니다.
- **웹 서비스** - J2EE 플랫폼에서는 JAX-RPC(Java API for XML-Based RPC)에서 구현한 웹 서비스를 제공하는 웹 응용 프로그램을 배포할 수 있습니다. J2EE 응용 프로그램이나 구성 요소가 다른 웹 서비스에 대한 클라이언트가 될 수도 있습니다. 응용 프로그램은 JAXR(Java API for XML Registries)을 통해 레지스트리에 액세스합니다.
- **응용 프로그램에 대한 서비스** - J2EE 플랫폼은 컨테이너가 응용 프로그램에 대한 서비스를 제공하도록 설계되었습니다. 그림 1–1은 다음 서비스를 보여줍니다.
  - 이름 지정 - 이름 지정 및 디렉토리서비스는 객체를 이름과 바인드합니다. J2EE 응용 프로그램은 객체의 JNDI 이름을 조회하여 객체를 찾습니다. JNDI는 Java Naming and Directory Interface API의 약자입니다.
  - 보안 - JACC(Java Authorization Contract for Containers)는 J2EE 컨테이너에 정의된 보안 계약의 집합입니다. 컨테이너는 클라이언트의 아이디를 기반으로 컨테이너의 자원과 서비스에 대한 액세스를 제한합니다.
- **트랜잭션 관리** - 트랜잭션은 개별 작업 단위입니다. 예를 들어, 은행 계좌 간에 자금을 이체하는 것이 트랜잭션입니다. 트랜잭션 관리 서비스는 트랜잭션이 완전히 완료되거나 롤백되는 것을 보장합니다.

외부 시스템 액세스

J2EE 플랫폼을 사용하면 응용 프로그램이 외부에 있는 시스템에 액세스할 수 있습니다. 응용 프로그램은 자원이라고 하는 객체를 통해 이 시스템에 연결합니다. 관리자의 책임 중 하나가 자원 구성입니다. J2EE 플랫폼을 사용하면 다음 API 및 구성 요소를 통해 외부 시스템에 액세스할 수 있습니다.

- JDBC - 데이터베이스 관리 시스템(DBMS)은 데이터의 저장, 구성 및 검색에 필요한 기능을 제공합니다. 대부분의 비즈니스 응용 프로그램은 관계형 데이터베이스에 데이터를 저장합니다. 응용 프로그램은 JDBC API를 통해 관계형 데이터베이스에 액세스합니다. 데이터베이스의 정보는 디스크에 저장되고 응용 프로그램이 종료된 후에도 존재하기 때문에 대개 지속성이 있다고 합니다. Application Server 번들에는 Java DB 데이터베이스 관리 시스템이 포함되어 있습니다.
- 메시징 - 메시징은 소프트웨어 구성 요소와 응용 프로그램 간의 통신 수단입니다. 메시징 클라이언트는 다른 클라이언트와 메시지를 주고 받습니다. 응용 프로그램은 JMS(Java Messaging Service) API를 통해 메시징 공급자에 액세스합니다. Application Server에는 JMS 공급자가 포함되어 있습니다.
- 커넥터 - J2EE 커넥터 구조를 사용하면 J2EE 응용 프로그램과 기존 EIS(Enterprise Information Systems) 간의 통합이 가능합니다. 응용 프로그램은 커넥터 또는 자원 어댑터라고 하는 이동 가능한 구성 요소를 통해 EIS에 액세스합니다.
- JavaMail - 응용 프로그램은 JavaMail API를 통해 SMTP 서버에 연결하여 전자 메일을 보내고 받습니다.
- 서버 관리 - 그림 1-1의 오른쪽 아래는 Application Server의 관리 인터페이스를 나타냅니다. 관리 도구는 이 인터페이스를 사용하여 Application Server와 통신합니다.

관리 도구

Sun Java System Application Server를 관리하는 데 다양한 도구와 API를 사용할 수 있습니다.

- 28 페이지 “관리 콘솔”
- 29 페이지 “명령줄 인터페이스(asadmin 유틸리티)”
- 29 페이지 “JConsole”
- 30 페이지 “AMX(Application Server Management Extension)”

관리 콘솔

관리 콘솔은 쉽게 탐색할 수 있는 인터페이스와 온라인 도움말 기능을 제공하는 브라우저 기반의 도구입니다. 관리 콘솔을 사용하려면 관리 서버(Domain Administration Server 또는 DAS라고도 함)를 실행해야 합니다. 관리 콘솔을 시작하려면 관리 서버 호스트 이름 및 포트 번호가 필요합니다. 기본 관리 서버에 대한 기본 관리 서버 포트 번호는 4849입니다. 또한 관리 콘솔에 로그인하려면 관리 사용자 이름 및 비밀번호가 필요합니다. 자세한 내용은 해당 절을 참조하십시오.

관리 콘솔을 시작하려면 웹 브라우저에 다음을 입력합니다.

```
https://hostname:port
```

예를 들면 다음과 같습니다.

```
https://kindness.sun.com:4849
```

관리 서버가 실행 중인 시스템에서 관리 콘솔을 실행할 경우 호스트 이름으로 `localhost`를 지정할 수 있습니다.

Windows의 시작 메뉴에서 Application Server 관리 콘솔을 시작합니다.

명령줄 인터페이스(asadmin 유틸리티)

`asadmin` 유틸리티는 Sun Java System Application Server용 명령줄 인터페이스입니다. 관리 콘솔로 제공되는 관리 작업과 동일한 집합의 관리 작업을 수행할 수 있습니다. `asadmin` 유틸리티는 쉘 명령 프롬프트 또는 다른 스크립트나 프로그램에서 호출할 수 있습니다. `asadmin` 유틸리티는 *install-dir*`/bin` 디렉토리에 설치됩니다. Solaris의 경우 Sun Java System Application Server의 기본 설치 루트 디렉토리는 `/opt/SUNWappserver`입니다.

`asadmin` 유틸리티를 시작하려면 *install-dir*`/bin` 디렉토리로 이동한 후 다음을 입력합니다.

```
$ asadmin
```

`asadmin` 내에서 사용 가능한 명령을 나열하려면 다음을 입력합니다.

```
asadmin> help
```

쉘의 명령 프롬프트에서 `asadmin` 명령을 실행할 수도 있습니다.

```
$ asadmin help
```

명령의 구문과 예를 확인하려면 명령 이름 다음에 `help`를 입력합니다. 예를 들면 다음과 같습니다.

```
asadmin> help create-jdbc-resource
```

지정한 명령에 대한 `asadmin help` 정보는 해당 명령의 Unix 설명서 페이지를 표시합니다. 이 설명서 페이지는 **Sun Java System Application Server Enterprise Edition 8.2 Reference Manual** 웹 사이트에서 HTML 형식으로 사용할 수도 있습니다.

JConsole

Java 2, Platform Standard Edition 5.0에 Java 모니터링 및 관리 콘솔(JConsole)이 도입되었습니다. JConsole은 Sun Java System Application Server를 모니터링하는 데 사용됩니다. Application Server에 연결하는 경우 JConsole 원격 탭 또는 고급 탭을 사용할 수 있습니다.

- 원격 탭: 사용자 이름, 비밀번호, 관리 서버 호스트 및 JMS 포트 번호(기본값: 8686)를 지정하고 연결을 선택합니다.
- 고급 탭: 서비스로 `JMXServiceURL(jmx:rmi:///jndi/rmi://host:jms-port/jmxrmi)`을 지정하고 연결을 선택합니다. `JMXServerURL`은 `server.log` 파일과 도메인 만들기 명령의 명령 창에 출력됩니다.

AMX(Application Server Management Extension)

AMX(**A**pplication Server **M**anagement e**X**tension)는 모든 Application Server 구성 및 모니터링 JMX Managed Bean을 AMX 인터페이스를 구현하는 사용하기 쉬운 클라이언트측 동적 프록시로 제공하는 API입니다.

AMX(Application Server Management Extension) 사용에 대한 자세한 내용은 **Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide**의 16 장, "Using the Java Management Extensions (JMX) API"를 참조하십시오.

Application Server 명령 및 개념

Sun Java System Application Server는 하나 이상의 도메인으로 구성됩니다. 도메인은 관리 경계 또는 컨텍스트로서, 각각 이와 연관된 관리 서버(Domain Administration Server 또는 DAS라고도 함)를 가지며 0개 이상의 독립 실행형 인스턴스 및/또는 클러스터로 구성됩니다. 각 클러스터에는 같은 종류의 서버 인스턴스가 하나 이상 있습니다. 서버 인스턴스는 단일 물리적 시스템에서 Application Server를 실행하는 단일 JVM(Java Virtual Machine)입니다. 도메인의 서버 인스턴스(독립 실행형 인스턴스 또는 클러스터링된 인스턴스)는 서로 다른 물리적 호스트에서 실행될 수 있습니다.

이 절은 다음 내용으로 구성되어 있습니다.

도메인

도메인은 함께 관리되는 인스턴스의 그룹입니다. 그러나 하나의 Application Server 인스턴스는 한 도메인에만 속할 수 있습니다. 도메인은 관리 경계뿐 아니라 서로 다른 관리자가 특정 Application Server 인스턴스 그룹(도메인)을 관리할 수 있는 기본적인 보안 구조도 제공합니다. 서버 인스턴스를 별도의 도메인으로 그룹화하면 서로 다른 조직이나 관리자가 단일 Application Server 설치를 공유할 수 있습니다. 도메인마다 다른 도메인과 독립된 고유한 구성 로그 파일 및 응용 프로그램 배포 영역이 있습니다. 하나의 도메인에 대한 구성을 변경해도 다른 도메인의 구성은 영향을 받지 않습니다.

Sun Java System Application Server 설치 프로그램은 기본 관리 도메인(domain1로 명명)과 이와 연관된 도메인 관리 서버(server로 명명)를 만듭니다. 이때 관리 서버 포트 번호를 입력해야 하며 기본 관리 서버 포트는 4849입니다. 또한 설치 프로그램은 관리 사용자 이름과 비밀번호를 캐리합니다. 설치 후 추가 관리 도메인을 작성할 수 있습니다.

DAS(Domain Administration Server)

도메인마다 고유한 포트 번호를 갖는 자체 DAS(Domain Administration Server)가 있습니다. 관리 콘솔은 특정 DAS와 통신하여 연관된 도메인을 관리합니다. 각 관리 콘솔 세션 중 특정 도메인을 구성하고 관리할 수 있습니다.

DAS(Domain Administration Server)는 관리 응용 프로그램을 호스트하기 위해 특별히 지정된 Application Server 인스턴스입니다. DAS는 관리자를 인증하고 관리 도구의 요청을 승인하며 도메인의 서버 인스턴스와 통신하여 요청을 수행합니다. DAS가 관리 서버 또는 기본 서버로 참조되는 경우도 있습니다. Sun Java System Application Server 설치 시에 만들어지는 유일한 서버 인스턴스이며 배포 시 사용할 수 있기 때문에 기본 서버로 참조되기도 합니다. DAS는 추가 관리 기능이 있는 단순 서버 인스턴스입니다.

각 관리 콘솔 세션 중 단일 도메인을 구성하고 관리할 수 있습니다. 여러 도메인을 만든 경우 다른 도메인을 관리하려면 추가 관리 콘솔 세션을 시작해야 합니다. 관리 콘솔에 대한 URL을 지정할 때 반드시 관리할 도메인과 연관된 DAS의 포트 번호를 사용하십시오.

클러스터

클러스터는 동일한 집합의 응용 프로그램, 자원 및 구성 정보를 공유하는 명명된 서버 인스턴스 모음입니다. 서버 인스턴스는 하나의 클러스터에만 속할 수 있습니다. 클러스터는 여러 시스템에 걸쳐 로드를 분산함으로써 서버 인스턴스의 로드 균형 조정을 용이하게 합니다. 또한 클러스터는 인스턴스 수준의 페일오버를 통해 고가용성을 가능하게 합니다. 관리 관점에서 클러스터는 클러스터 작업(예: 응용 프로그램 배포)이 클러스터를 구성하는 모든 인스턴스상에서 수행되는 가상화된 엔티티를 의미합니다.

노드 에이전트

인스턴스 라이프사이클을 원격으로 용이하게 관리하려면 도메인의 각 노드에 경량 에이전트(예: JMX 런타임만을 호스팅)가 필요합니다. 이 에이전트는 기본적으로 DAS의 지시대로 서버 인스턴스를 시작, 중지 및 생성합니다. 또한 노드 에이전트는 워치독의 역할을 수행하며 실패한 프로세스를 다시 시작합니다. 노드 에이전트는 DAS와 같이 특정 관리 작업에만 사용되어야 하며 고가용성을 예상해서는 안 됩니다. 그러나 노드 에이전트는 "항상 실행"되는 구성 요소로서, 원시 O/S 노드 부트스트랩(예: Solaris/Linux inetd 또는 Windows 서비스)에 의해 시작되도록 구성되어야 합니다. 노드 에이전트는 DAS에 사용되지 않습니다.

서버 인스턴스

서버 인스턴스는 단일 노드에서 J2EE 1.4 Application Server를 호스팅하는 J2EE 호환의 단일 JVM(Java Virtual Machine)입니다. 도메인의 각 서버 인스턴스에는 고유 이름이 있습니다. 클러스터링된 서버 인스턴스는 클러스터의 구성원이며 해당 부모 클러스터에서 모든 응용 프로그램, 자원 및 구성을 상속 받습니다. 따라서 클러스터 내의 모든 인스턴스는 동일한 종류가 됩니다. 클러스터링되지 않은 서버 인스턴스는 클러스터에 속하지 않으며 이에 따라 독립적인 응용 프로그램, 자원 및 구성 집합을 갖습니다.

Application Server 인스턴스는 응용 프로그램 배포의 기본을 구성합니다. 각 인스턴스는 하나의 도메인에 속합니다. DAS를 제외한 모든 서버 인스턴스에는 인스턴스가 상주하는 시스템을 정의하는 노드 에이전트 이름에 대한 참조가 포함되어야 합니다.

토폴로지에 원격 서버 인스턴스(DAS를 제외한 서버 인스턴스)가 포함되어 있으면 노드 에이전트를 만들어 원격 서버 인스턴스를 용이하게 관리합니다. 노드 에이전트의 역할은 서버 인스턴스를 작성, 시작, 중지 및 삭제하는 것입니다. 명령줄 인터페이스 명령을 사용하여 노드 에이전트를 설정합니다. 그림 1–2는 Application Server 인스턴스를 자세히 보여줍니다.

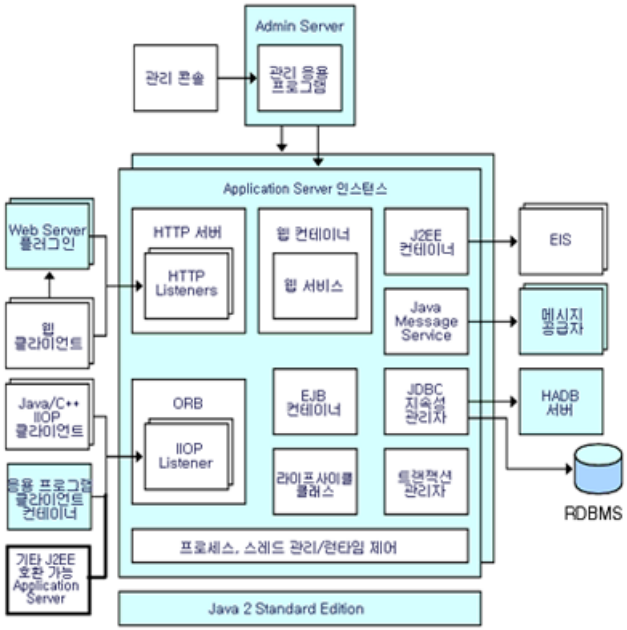

그림 1–2 Application Server 인스턴스

Sun Java System Application Server는 설치 시 server라고 하는 한 개의 Application Server 인스턴스를 만듭니다. 대부분의 경우 Application Server 인스턴스는 하나만 있으면 됩니다. 하지만 환경에 따라 하나 이상의 추가 Application Server 인스턴스를 만들어야 할

수도 있습니다. 예를 들어, 개발 환경에서 다른 Application Server 인스턴스를 사용하여 다른 Application Server 구성을 테스트하거나 다른 응용 프로그램 배포를 비교 및 테스트할 수 있습니다. Application Server 인스턴스는 손쉽게 추가 또는 삭제할 수 있기 때문에 이러한 Application Server 인스턴스를 사용하면 실험에 사용할 임시 샌드 박스 영역을 만들 수 있습니다.

또한 각 Application Server 인스턴스에 대해 가상 서버를 만들 수도 있습니다. 단일 설치 Application Server 인스턴스 내에서 회사나 개인 도메인 이름, IP 주소 및 일부 관리 기능을 제공할 수 있습니다. 사용자의 경우 하드웨어와 기본 서버 유지 관리를 제외하면 고유 웹 서버를 가진 것과 거의 같습니다. 이러한 가상 서버는 여러 Application Server 인스턴스에 걸쳐 있지 않습니다. 가상 서버에 대한 자세한 내용은 12 장을 참조하십시오.

운영상 배포할 때 많은 용도로 여러 Application Server 인스턴스 대신 가상 서버를 사용할 수 있습니다. 그러나 가상 서버가 사용자 요구 사항을 충족시키지 못할 경우 여러 Application Server 인스턴스를 사용할 수도 있습니다. Application Server 인스턴스를 중지하면 Application Server 인스턴스는 더 이상 새 연결을 수락하지 않으며 해결되지 않은 모든 연결이 완료될 때까지 대기합니다. 시스템이 충돌하거나 오프라인이 되면 서버가 종료되므로 서버에서 처리 중이던 요청이 손실될 수 있습니다.

Application Server 명령

Application Server 관리에는 도메인, 클러스터, 노드 에이전트 및 서버 인스턴스의 만들기, 구성, 제어 및 관리와 같은 작업이 포함됩니다. 이 절은 다음 내용으로 구성되어 있습니다.

■ 40 페이지 “관리자 비밀번호 변경”

도메인 만들기

도메인은 create-domain 명령을 사용하여 만듭니다. 다음 예의 명령은 mydomain이라고 하는 도메인을 만듭니다. 관리 서버는 포트 1234에서 수신하고 관리자 이름은 hanan입니다. 관리 비밀번호와 마스터 비밀번호를 묻는 명령 프롬프트가 나타납니다.

```
$ asadmin create-domain --adminport 80 --adminuser hanan mydomain
```

mydomain 도메인의 관리 콘솔을 시작하려면 브라우저에서 다음 URL을 입력합니다.

```
http://hostname:80
```

앞에서 설명한 create-domain 예의 경우, 도메인의 로그 파일, 구성 파일 및 배포된 응용 프로그램은 현재 다음의 디렉토리에 있습니다.

domain-root-dir/mydomain

도메인의 디렉토리를 다른 위치에 만들려면 --domaindir 옵션을 지정합니다. 명령의 전체 구문을 보려면 asadmin help create-domain을 입력합니다.

도메인 삭제

도메인은 asadmin delete-domain 명령을 사용하여 삭제합니다. 도메인을 관리할 수 있는 운영 체제 사용자(또는 루트)만 이 명령을 제대로 실행할 수 있습니다. 예를 들어, mydomain이라는 도메인을 삭제하려면 다음 명령을 입력합니다.

```
$ asadmin delete-domain mydomain
```

도메인 나열

시스템에 만든 도메인은 asadmin list-domains 명령을 사용하여 확인할 수 있습니다. 기본 *domain-root-dir* 디렉토리에 있는 도메인을 나열하려면 다음 명령을 입력합니다.

```
$ asadmin list-domains
```

다른 디렉토리에 만든 도메인을 나열하려면 --domaindir 옵션을 지정합니다.

도메인 시작

도메인을 시작하면 관리 서버와 Application Server 인스턴스가 시작됩니다. Application Server 인스턴스가 시작되면 계속 실행되어 요청을 청취하고 수용합니다. 각 도메인을 별도로 시작해야 합니다.

도메인을 시작하려면 asadmin start-domain 명령을 입력하고 도메인 이름을 지정합니다. 예를 들어, 기본 도메인(domain1)을 시작하려면 다음을 입력합니다.

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

도메인이 하나만 있는 경우, 도메인 이름을 생략합니다. 전체 명령 구문을 보려면 `asadmin help start-domain`을 입력합니다. 비밀번호 데이터를 생략한 경우 비밀번호를 제공하라는 메시지가 표시됩니다.

Windows에서 기본 도메인 시작

Windows 시작 메뉴에서 프로그램 -> Sun Microsystems -> Application Server -> 관리 서버 시작을 선택합니다.

도메인 중지

도메인을 중지하면 관리 서버와 Application Server 인스턴스가 중지됩니다. 도메인을 중지하면 서버 인스턴스가 새로운 연결 승인을 중지하고 모든 진행 중인 연결이 완료될 때까지 기다립니다. 서버 인스턴스가 종료 과정을 완료해야 하기 때문에 이 과정은 시간이 걸립니다. 도메인을 중지하는 동안 관리 콘솔이나 `asadmin` 명령 대부분을 사용할 수 없습니다.

도메인을 중지하려면 `asadmin stop-domain` 명령을 입력하고 도메인 이름을 지정합니다. 예를 들어, 기본 도메인(`domain1`)을 중지하려면 다음을 입력합니다.

```
$ asadmin stop-domain domain1
```

도메인이 하나만 있을 경우 도메인 이름은 선택 사항입니다. 전체 구문을 보려면 `asadmin help stop-domain`을 입력합니다.

Windows에서 기본 도메인 중지

시작 메뉴에서 프로그램 -> Sun Microsystems -> Application Server-> 관리 서버 중지를 선택합니다.

도메인 다시 시작

서버를 다시 시작하는 것은 도메인을 다시 시작하는 것과 같습니다. 도메인이나 서버를 다시 시작하려면 도메인을 중지하고 시작합니다.

클러스터 만들기

`create-cluster` 명령을 사용하여 클러스터를 만듭니다. 다음 예에서는 `mycluster`라는 클러스터를 만듭니다. 관리 서버 호스트는 `myhost`, 서버 포트는 `1234`, 관리 사용자 이름은 `admin`입니다. 명령을 실행하면 관리 비밀번호를 묻는 메시지가 표시됩니다.

```
$ asadmin create-cluster --host myhost --port 1234 --user admin mycluster
```

전체 구문을 보려면 `asadmin help create-cluster`를 입력합니다.

클러스터 시작

start-cluster 명령을 사용하여 클러스터를 시작합니다. 다음 예에서는 mycluster라는 클러스터를 시작합니다. 명령을 실행하면 관리 비밀번호를 묻는 메시지가 표시됩니다.

```
$ asadmin start-cluster --host myhost --port 1234 --user admin mycluster
```

myhost는 관리 서버 호스트, 1234는 관리 포트, admin은 관리 사용자 이름입니다.

전체 구문을 보려면 asadmin help start-cluster를 입력합니다. 클러스터가 시작되면 클러스터 내의 모든 서버 인스턴스가 시작됩니다. 인스턴스가 없는 클러스터는 시작할 수 없습니다.

클러스터 중지

stop-cluster 명령을 사용하여 클러스터를 중지합니다. 다음 예에서는 mycluster라는 클러스터를 중지합니다. 명령을 실행하면 관리 비밀번호를 묻는 메시지가 표시됩니다.

```
$ asadmin stop-cluster --host myhost --port 1234 --user admin mycluster
```

myhost는 관리 서버 호스트, 1234는 관리 포트, admin은 관리 사용자 이름입니다.

전체 구문을 보려면 asadmin help stop-cluster를 입력합니다. 클러스터가 중지되면 클러스터 내의 모든 서버 인스턴스가 중지됩니다. 인스턴스가 없는 클러스터는 중지할 수 없습니다.

노드 에이전트 만들기

create-node-agent 명령을 사용하여 노드 에이전트를 만듭니다. 다음 예에서는 mynodeagent라는 노드 에이전트를 만듭니다. 관리 서버 호스트는 myhost, 관리 서버 포트는 1234, 관리 사용자 이름은 admin입니다. 명령을 실행하면 관리 비밀번호를 묻는 메시지가 표시됩니다.

```
$ asadmin create-node-agent --host myhost --port 1234 --user admin mynodeagent
```

전체 구문을 보려면 asadmin help create-node-agent를 입력합니다.

노드 에이전트 시작

노드 에이전트 이름을 지정하여 start-node-agent 명령을 사용함으로써 노드 에이전트를 시작합니다. 예를 들어, mynodeagent 노드 에이전트를 시작하려면 다음을 입력합니다.

```
$ asadmin start-node-agent --user admin mynodeagent
```

전체 구문을 보려면 asadmin help start-node-agent를 입력합니다.

노드 에이전트 중지

stop-node-agent 명령에서 노드 에이전트 이름을 지정하여 노드 에이전트를 중지합니다. 예를 들어, mynodeagent 노드 에이전트를 중지하려면 다음을 입력합니다.

```
$ asadmin stop-node-agent mynodeagent
```

전체 구문을 보려면 asadmin help stop-node-agent를 입력합니다.

인스턴스 만들기

create-instance 명령을 사용하여 서버 인스턴스를 만듭니다. 다음 예에서는 myinstance라는 인스턴스를 만듭니다. 관리 서버 호스트는 myhost, 관리 서버 포트는 1234, 관리 사용자 이름은 admin입니다. 명령을 실행하면 관리 비밀번호를 묻는 메시지가 표시됩니다.

다음 예에서는 myinstance라는 클러스터링된 서버 인스턴스를 만듭니다. 명령을 실행하면 관리 비밀번호를 묻는 메시지가 표시됩니다.

```
$ asadmin create-instance --host myhost --port 1234
--user admin --cluster mycluster --nodeagent mynodeagent myinstance
```

myhost는 관리 서버 호스트, 1234는 관리 포트, admin은 관리 사용자 이름입니다. 또한 mycluster는 이 서버 인스턴스가 속한 클러스터, mynodeagent는 이 서버 인스턴스를 관리하는 노드 에이전트입니다.

전체 구문을 보려면 asadmin help create-instance를 입력합니다.

독립 실행형 서버 인스턴스를 만들려면 --cluster 옵션을 지정하지 마십시오.

다음 예에서는 mynodeagent라는 노드 에이전트에서 관리되는 myinstance라는 독립 실행형 서버 인스턴스를 만듭니다.

```
$ asadmin create-instance --host myhost --port 1234
--user admin --nodeagent mynodeagent myinstance
```

인스턴스 시작

start-instance 명령을 사용하여 서버 인스턴스를 시작합니다. 다음 예에서는 myinstance라는 서버 인스턴스를 시작합니다. 명령을 실행하면 관리 비밀번호를 묻는 메시지가 표시됩니다.

```
$ asadmin start-instance --host myhost --port 1234 --user admin  myinstance
```

myhost는 관리 서버 호스트, 1234는 관리 포트, admin은 관리 사용자 이름입니다. myinstance 서버 인스턴스는 클러스터링된 인스턴스 또는 독립 실행형 인스턴스일 수 있습니다.

전체 구문을 보려면 asadmin help start-instance를 입력합니다.

인스턴스 중지

stop-instance 명령을 사용하여 서버 인스턴스를 중지합니다. 다음 예에서는 myinstance라는 인스턴스를 중지합니다. 명령을 실행하면 관리 비밀번호를 묻는 메시지가 표시됩니다.

```
$ asadmin stop-instance --host myhost --port 1234 --user admin  myinstance
```

myhost는 관리 서버 호스트, 1234는 관리 포트, admin은 관리 사용자 이름입니다. myinstance 서버 인스턴스는 클러스터링된 인스턴스 또는 독립 실행형 인스턴스일 수 있습니다.

전체 구문을 보려면 asadmin help stop-instance를 입력합니다.

인스턴스 다시 시작

서버 인스턴스를 다시 시작하려면 인스턴스를 중지한 다음 시작합니다.

Domain Administration Server 다시 만들기

미러링을 목적으로 DAS(Domain Administration Server)의 작업 복사본을 제공하려면 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.

▼ DAS를 마이그레이션하는 방법

첫 번째 시스템(machine1)에서 세 번째 시스템(machine3)으로 DAS(Domain Administration Server)를 마이그레이션하려면 다음 단계가 필요합니다.

1 첫 번째 시스템에 설치한 대로 세 번째 시스템에 Application Server를 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.

관리자 비밀번호 변경

관리자 비밀번호를 재설정하려면 도메인의 모든 노드 에이전트를 중지해야 합니다. 노드 에이전트를 중지하면 연관된 모든 서버 인스턴스가 중지됩니다. 모든 서버 인스턴스 및 노드 에이전트가 중지되고 DAS(Domain Administration Server)만 실행 중이라면

다음과 같이 관리 사용자의 비밀번호를 변경할 수 있습니다.

1. 명령줄 인터페이스를 사용하여 관리자 비밀번호를 변경합니다.

   ```
   asadmin update-file-user --authrealmname admin-realm ... --userpassword
   newpassword <admin-user-name>
   ```

2. 관리 콘솔을 사용하여 관리자 비밀번호를 변경합니다.

   관리 서버의 구성 노드 > 보안 > 영역 > 관리 영역 > 파일 영역 사용자 편집을 선택합니다.

   관리자 비밀번호가 성공적으로 변경되었음을 나타내는 메시지가 표시됩니다.

3. 다음과 같이 새 비밀번호로 DAS(Domain Administration Server)를 다시 시작합니다.

   명령줄 인터페이스에 `asadmin start-domain --user admin --password newpassword domain1`을 입력합니다.

   도메인에 두 개의 노드 에이전트(`i1na`, `c1-na`)와 세 개의 인스턴스(`c1i1` 및 `c1i2`는 `c1`이라는 동일한 클러스터에 속해 있고 `i1`은 독립 실행형 서버 인스턴스임)가 있다고 가정합니다.

4. 새 비밀번호로 인스턴스를 시작하지 않고 노드 에이전트를 다시 시작합니다.

   예를 들면 다음과 같습니다.

   ```
   asadmin start-node-agent --user admin --password newpassword
   --startinstances=false i1-na asadmin
   ```

   ```
   asadmin start-node-agent --user admin --password newpassword
   --startinstances=false c1-na
   ```

5. 서버 및 클러스터를 다시 시작합니다.

```
asadmin start-node-agent --user admin --password newpassword ... c1
asadmin start-node-agent --user admin --password newpassword i1
```

Application Server 구성

Sun Java System Application Server 구성은 `domain.xml` 파일에 저장됩니다. `domain.xml`은 Application Server의 구성 상태를 나타내는 문서이며 해당 관리 도메인에 대한 중앙 저장소입니다. 이 문서에는 Application Server 도메인 모델의 XML 표현이 포함됩니다. `domain.xml`의 내용은 도메인 DTD 형식으로 표현된 사양으로 관리됩니다.

이 절은 다음 내용으로 구성되어 있습니다.

- 41 페이지 "Application Server 구성 변경"
- 41 페이지 "Application Server의 포트"
- 42 페이지 "J2SE 소프트웨어 변경"

Application Server 구성 변경

다음 구성 변경 사항을 적용하려면 서버를 다시 시작해야 합니다.

- JVM 옵션 변경
- 포트 번호 변경
- HTTP, IIOP 및 JMS 서비스 관리
- 스레드 풀 관리

자세한 지침은 35 페이지 "도메인 다시 시작"을 참조하십시오.

동적 구성이 활성화된 경우에는 다음 구성 변경 사항을 적용하기 위해 서버를 다시 시작할 필요가 **없습니다**.

- 응용 프로그램 배포 및 배포 해제
- JDBC, JMS, 커넥터 자원 및 풀 추가 또는 제거
- 로깅 수준 변경
- 파일 영역 사용자 추가
- 모니터링 수준 변경
- 자원 및 응용 프로그램 활성화 및 비활성화

`asadmin reconfig` 명령은 더 이상 사용되지 않으므로 필요하지 않습니다. 구성 변경 사항이 서버에 동적으로 적용됩니다.

Application Server의 포트

다음 표는 Application Server의 포트 Listener에 대해 설명합니다.

표 1–1 포트를 사용하는 Application Server Listener

Listener	기본 포트 번호	설명
관리 서버	4849	도메인의 관리 서버는 관리 콘솔과 asadmin 유틸리티에서 액세스합니다. 관리 콘솔의 경우 브라우저의 URL에 포트 번호를 지정합니다. asadmin 명령을 원격으로 실행할 경우 --port 옵션을 사용하여 포트 번호를 지정합니다.
HTTP	8080	웹 서버는 포트에서 HTTP 요청을 수신합니다. 배포된 웹 응용 프로그램에 액세스하기 위해 클라이언트가 이 포트에 연결합니다.
HTTPS	8181	보안 통신을 위해 구성된 웹 응용 프로그램은 별도의 포트에서 수신합니다.
IIOP	3700	Enterprise Bean(EJB 구성 요소)의 원격 클라이언트는 IIOP Listener를 통해 Bean에 액세스합니다.
IIOP, SSL	3820/3890	다른 포트는 보안 통신을 위해 구성된 IIOP Listener에서 사용합니다.
IIOP, SSL 및 상호 인증		다른 포트는 상호(클라이언트 및 서버) 인증을 위해 구성된 IIOP Listener에서 사용합니다.
JMX	8686	다른 포트는 DAS와의 통신을 위해 JMX 커넥터에서 사용합니다.

J2SE 소프트웨어 변경

Application Server는 J2SE(Java 2 Standard Edition) 소프트웨어를 사용합니다. Application Server를 설치한 경우 J2SE 소프트웨어의 디렉토리가 지정됩니다. J2SE 소프트웨어 변경에 대한 자세한 지침은 17 장을 참조하십시오.

◆ ◆ ◆ 2 장

2 응용 프로그램 배포

이 장에서는 Application Server에서 J2EE 응용 프로그램을 배포(설치)하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

배포 라이프사이클

Application Server를 설치하고 도메인을 시작한 후 J2EE 응용 프로그램과 모듈을 배포(설치)할 수 있습니다. 배포 중, 그리고 응용 프로그램이 변경되면 응용 프로그램 또는 모듈은 다음 단계를 거칩니다.

1. 초기 배포

   응용 프로그램 또는 모듈을 배포하기 전에 도메인을 시작합니다.

   특정한 독립 실행형 서버 인스턴스나 클러스터에 응용 프로그램이나 모듈을 배포(설치)합니다. 응용 프로그램과 모듈은 아카이브 파일로 패키지화되므로 배포 중에 아카이브 파일 이름을 지정합니다. 기본값은 기본 서버 인스턴스 server로 배포하는 것입니다.

   서버 인스턴스 또는 클러스터에 배포한 경우 응용 프로그램이나 모듈은 도메인의 중앙 저장소에 있게 되며 배포가 수행된 클러스터 또는 서버 인스턴스가 이를 대상으로 참조합니다.

   관리 콘솔 대신 asadmin deploy 명령을 사용하여 도메인에 배포할 수도 있습니다. 응용 프로그램이나 모듈을 도메인에만 배포할 경우 응용 프로그램이나 모듈은 도메인의 중앙 저장소에 존재하지만 참조를 추가할 때까지는 서버 인스턴스나 클러스터에 의해 참조되지 않습니다.

배포는 동적입니다. 응용 프로그램을 사용하기 위해 응용 프로그램이나 모듈을 배포한 후 서버 인스턴스를 다시 시작할 필요가 없습니다. 다시 시작할 경우 모든 배포된 응용 프로그램과 모듈이 배포되고 사용 가능하게 됩니다.

2. 활성화 또는 비활성화

   기본적으로 배포된 응용 프로그램 또는 모듈은 활성화되어 있습니다. 이는 액세스 가능한 서버 인스턴스나 클러스터에 응용 프로그램을 배포한 경우 이를 실행할 수 있고 클라이언트에서 액세스할 수 있음을 의미합니다. 액세스를 방지하려면 응용 프로그램이나 모듈을 비활성화합니다. 비활성화된 응용 프로그램이나 모듈은 도메인에서 제거되지 않으므로 배포 후 쉽게 활성화할 수 있습니다.

3. 배포된 응용 프로그램이나 모듈의 대상 추가 또는 삭제

   일단 배포되면 응용 프로그램이나 모듈은 중앙 저장소에 있게 되고 여러 서버 인스턴스 및/또는 클러스터가 이를 참조할 수 있습니다. 처음에는 배포 대상인 서버 인스턴스나 클러스터에서 응용 프로그램이나 모듈을 참조합니다.

   응용 프로그램이나 모듈을 배포한 후 이를 참조하는 서버 인스턴스와 클러스터를 변경하려면 관리 콘솔을 사용하여 응용 프로그램이나 모듈의 대상을 변경하거나 `asadmin` 도구를 사용하여 응용 프로그램 참조를 변경합니다. 응용 프로그램이 중앙 저장소에 저장되기 때문에 대상을 추가하거나 삭제하면 다른 대상에 있는 동일한 버전의 응용 프로그램이 추가되거나 삭제됩니다. 그러나 둘 이상의 대상에 배포된 응용 프로그램은 한 대상에서 활성화하고 다른 대상에서는 비활성화할 수 있습니다. 대상에서 응용 프로그램을 참조하더라도 해당 대상에서 응용 프로그램을 활성화하지 않으면 사용자가 사용할 수 없습니다.

4. 재배포

   배포된 응용 프로그램이나 모듈을 대체하려면 다시 배포하십시오. 재배포는 자동으로 이전에 배포된 응용 프로그램이나 모듈의 배포를 취소하고 이를 새 응용 프로그램이나 모듈로 대체합니다.

   관리 콘솔을 통해 재배포하면 재배포된 응용 프로그램이나 모듈은 도메인으로 배포되고 동적 재구성을 활성화한 경우, 이를 참조하는 모든 독립 실행형 또는 클러스터링된 서버 인스턴스는 자동으로 새로운 버전을 수신합니다. `asadmin deploy` 명령을 사용하여 재배포하는 경우 `domain`을 대상으로 지정합니다.

   프로덕션 환경의 경우 서비스를 중단하지 않은 채 응용 프로그램을 업그레이드하는 롤링 업그레이드를 사용합니다.

5. 배포 해제

   응용 프로그램이나 모듈을 제거하려면 배포 해제합니다.

자동 배포

자동 배포 기능을 사용하면 사전에 패키지화되어 있는 응용 프로그램이나 모듈을 *domain-dir*/autodeploy 디렉토리에 복사하는 방법으로 배포할 수 있습니다.

예를 들어, hello.war라는 파일을 *domain-dir* /autodeploy 디렉토리로 복사합니다. 응용 프로그램을 배포 해제하려면 autodeploy 디렉토리에서 hello.war 파일을 제거합니다.

관리 콘솔이나 asadmin 도구를 사용하여 응용 프로그램을 배포 해제할 수도 있습니다. 이러한 경우 아카이브 파일이 그대로 유지됩니다.

주 – 자동 배포는 기본 서버 인스턴스에 대해서만 사용할 수 있습니다.

자동 배포 기능은 개발 환경을 위한 것입니다. 세션 지속성 기능인 프로덕션 환경 기능과 호환되지 않습니다. 자동 배포가 활성화된 경우 세션 지속성을 활성화하지 마십시오.

압축 해제된 응용 프로그램 배포

이 기능은 고급 개발자를 위한 것입니다.

기본 서버 인스턴스(서버)에 배포할 때만 디렉토리 배포를 사용합니다. 클러스터나 독립 실행형 서버 인스턴스에 배포할 때는 사용할 수 없습니다.

압축 해제된 응용 프로그램이나 모듈을 포함하는 디렉토리는 확장된 디렉토리라고 합니다. 디렉토리의 내용은 해당 J2EE 아카이브 파일의 내용과 일치해야 합니다. 예를 들어 디렉토리에서 웹 응용 프로그램을 배포할 경우 디렉토리의 내용은 해당하는 WAR 파일과 동일해야 합니다. 필요한 디렉토리 내용에 대한 정보는 해당 사양을 참조하십시오.

확장된 디렉토리에서 직접 배포 설명자 파일을 변경할 수 있습니다.

동적 재로드를 사용하도록 환경을 구성한 경우 디렉토리에서 배포된 응용 프로그램을 동적으로 재로드할 수도 있습니다. 자세한 내용은 194 페이지 “고급 설정 구성”을 참조하십시오.

배포 계획 사용

이 기능은 고급 개발자를 위한 것입니다.

배포 계획은 Application Server에 관련된 배포 설명자만 포함하는 JAR 파일입니다. 이 배포 설명자(예: sun-application.xml)는 **Application Server Developer's Guide**에 설명되어 있습니다. 배포 계획은 **JSR 88: J2EE 응용 프로그램 배포** 구현의 일부입니다. Application Server에 관련된 배포 설명자를 포함하지 않는 응용 프로그램이나 모듈을 배포하려면 배포 계획을 사용합니다.

배포 계획을 사용하여 배포하려면 asadmin deploy 명령의 --deploymentplan 옵션을 지정합니다. 예를 들어, 다음 명령은 myrosterapp.ear 파일에서 지정한 계획에 따라 mydeployplan.jar 파일에 엔터프라이즈 응용 프로그램을 배포합니다.

```
$ asadmin deploy --user admin ---deploymentplan mydeployplan.jar myrosterapp.ear
```

엔터프라이즈 응용 프로그램(EAR)의 배포 계획 파일에서 sun-application.xml 파일은 루트에 있습니다. 각 모듈의 배포 설명자는 이 구문에 따라 저장됩니다. *module-name.sun-dd-name*, 여기서 *sun-dd-name*은 모듈 유형에 따라 다릅니다. 모듈에 CMP 매핑 파일이 포함된 경우 파일 이름은 *module-name*.sun-cmp-mappings.xml이 됩니다. .dbschema 파일은 슬래시(/) 문자가 파운드 기호(#)로 대체되어 루트 수준에 저장됩니다. 다음 목록에서는 엔터프라이즈 응용 프로그램(EAR)에 대한 배포 계획 파일의 구조를 보여줍니다.

```
$ jar -tvf mydeployplan.jar
420 Thu Mar 13 15:37:48 PST 2003 sun-application.xml
370 Thu Mar 13 15:37:48 PST 2003 RosterClient.war.sun-web.xml
418 Thu Mar 13 15:37:48 PST 2003 roster-ac.jar.sun-application-client.xml
1281 Thu Mar 13 15:37:48 PST 2003 roster-ejb.jar.sun-ejb-jar.xml
2317 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-ejb-jar.xml
3432 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-cmp-mappings.xml
84805 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.RosterSchema.dbschema
```

웹 응용 프로그램이나 모듈 파일의 배포 계획에서 Application Server에 관련된 배포 설명자는 루트 수준에 있습니다. 독립 실행형 EJB 모듈에 CMP Bean이 포함된 경우 배포 계획의 루트 수준에 sun-cmp-mappings.xml 및 .dbschema 파일이 포함됩니다. 다음 목록에서 배포 계획은 CMP Bean을 설명합니다.

```
$ jar r -tvf myotherplan.jar
3603 Thu Mar 13 15:24:20 PST 2003 sun-ejb-jar.xml
3432 Thu Mar 13 15:24:20 PST 2003 sun-cmp-mappings.xml
84805 Thu Mar 13 15:24:20 PST 2003 RosterSchema.dbschema
```

deploytool 유틸리티 사용

소프트웨어 개발자를 위해 설계된 deploytool 유틸리티는 J2EE 응용 프로그램과 모듈을 패키지화하고 배포합니다. deploytool 사용 방법에 대한 지침은 **J2EE 1.4 Tutorial**을 참조하십시오.

J2EE 아카이브 파일의 유형

소프트웨어 공급자가 응용 프로그램이나 모듈을 아카이브 파일로 패키지화합니다. 응용 프로그램이나 모듈을 배포하려면 아카이브 파일 이름을 지정합니다. 아카이브 파일의 내용과 구조는 J2EE 플랫폼의 사양에 의해 정의됩니다. J2EE 아카이브 파일의 유형은 다음과 같습니다.

- WAR(Web Application Archive): WAR 파일은 정적 HTML 페이지, JAR 파일, 태그 라이브러리 및 유틸리티 클래스뿐만 아니라 서블릿 및 JSP 같은 웹 구성 요소로 구성됩니다. WAR 파일 이름은 `.war` 확장자를 갖습니다.
- EJB JAR: EJB JAR 파일에는 EJB 기술에 사용되는 구성 요소인 하나 이상의 Enterprise Bean이 포함되어 있습니다. 또한 EJB JAR 파일에는 Enterprise Bean에 필요한 유틸리티 클래스도 포함되어 있습니다. EJB JAR 파일 이름은 `.jar` 확장자를 갖습니다.
- J2EE 응용 프로그램 클라이언트 JAR: 이 JAR 파일에는 RMI/IIOP를 통해 Enterprise Bean 같은 서버측 구성 요소에 액세스하는 J2EE 응용 프로그램 클라이언트의 코드가 포함되어 있습니다. 관리 콘솔에서는 J2EE 응용 프로그램 클라이언트를 “응용 프로그램 클라이언트”라고 합니다.J2EE 응용 프로그램 클라이언트 JAR 파일 이름은 `.jar` 확장자를 갖습니다.
- RAR(Resource Adapter Archive): RAR 파일에는 자원 어댑터가 있습니다. J2EE Connector Architecture 사양에서 정의한 자원 어댑터는 웹 구성 요소 및 응용 프로그램 클라이언트가 자원 및 외부 엔터프라이즈 시스템에 액세스할 수 있게 해주는 이동 가능한 구성 요소입니다. 자원 어댑터는 커넥터라고 합니다. RAR 파일 이름은 `.rar` 확장자를 갖습니다.
- EAR(Enterprise Application Archive): EAR 파일에는 하나 이상의 WAR, EJB JAR, RAR 또는 J2EE 응용 프로그램 클라이언트 JAR 파일이 있습니다. EAR 파일 이름은 `.ear` 확장자를 갖습니다.

소프트웨어 공급자가 응용 프로그램을 하나의 파일이나 별도의 WAR, EJB JAR 및 응용 프로그램 클라이언트 JAR 파일로 어셈블할 수 있습니다. 관리 도구에서 배포 페이지와 명령은 모든 유형의 파일에 대해 유사합니다.

이름 지정 규약

해당 도메인에서 배포된 응용 프로그램과 모듈의 이름은 고유해야 합니다.

- 관리 콘솔을 사용하여 배포할 경우 응용 프로그램 이름 필드에서 이름을 지정합니다.
- `asadmin deploy` 명령을 사용하여 배포할 경우 응용 프로그램이나 모듈의 기본 이름은 배포할 JAR 파일의 접두어입니다. 예를 들어, `hello.war` 파일의 경우 웹 응용 프로그램 이름은 `hello`입니다. 기본 이름을 대체하려면 `--name` 옵션을 지정합니다.

응용 프로그램 내에서 유형이 다른 모듈은 동일한 이름을 가질 수 있습니다. 응용 프로그램을 배포할 때 개별 모듈이 있는 디렉토리 이름에는 `_jar`, `_war` 및 `_rar` 접미어가 붙습니다. 응용 프로그램 내에서 유형이 같은 모듈은 이름이 고유해야 합니다. 또한 데이터베이스 스키마 파일 이름은 응용 프로그램 내에서 고유해야 합니다.

ejb-jar.xml 파일의 <module-name> 부분에서 볼 수 있는 모듈 파일 이름, EAR 파일 이름, 모듈 이름과 ejb-jar.xml파일의 <ejb-name> 부분에서 볼 수 있는 EJB 이름에는 Java 패키지와 같은 이름 지정 스키마를 사용하는 것이 좋습니다. 이렇게 패키지와 비슷한 이름 지정 스키마를 사용하면 이름 충돌을 방지할 수 있습니다. 이러한 이름 지정의 이점은 Application Server 뿐만 아니라 다른 J2EE 응용 프로그램 서버에도 적용됩니다.

EJB 구성 요소에 대한 JNDI 조회 이름도 고유해야 합니다. 일관된 이름 지정 규약을 설정하는 것이 좋습니다. 예를 들어, EJB 이름에 응용 프로그램 이름과 모듈 이름을 추가하는 것도 고유한 이름을 유지하는 한 가지 방법입니다. 이 경우 mycompany.pkging.pkgingEJB.MyEJB는 응용 프로그램 pkging.ear에 패키지화된 pkgingEJB.jar 모듈의 EJB에 대한 JNDI 이름이 됩니다.

운영 체제에서 사용할 수 없는 공백이나 문자가 패키지 및 파일 이름에 포함되지 않도록 합니다.

◆ ◆ ◆ 3 장

3 JDBC 자원

JDBC 자원(데이터 소스)은 응용 프로그램에 데이터베이스 연결 수단을 제공합니다. 일반적으로 관리자는 도메인에 배포된 응용 프로그램에서 액세스한 데이터베이스마다 JDBC 자원을 만듭니다. 그러나 한 데이터베이스에 대해 JDBC 자원을 여러 개 만들 수 있습니다.

이 장은 다음 내용으로 구성되어 있습니다.

JDBC 자원 만들기

데이터 저장과 구성 및 검색을 위해 대부분의 응용 프로그램은 관계형 데이터베이스를 사용합니다. Java EE 응용 프로그램은 JDBC API를 통해 관계형 데이터베이스에 액세스합니다.

JDBC 자원(데이터 소스)은 응용 프로그램에 데이터베이스 연결 수단을 제공합니다. JDBC 자원을 만들기 전에 먼저 JDBC 연결 풀을 만드십시오.

JDBC 자원을 만들려면 해당 자원을 식별하는 고유한 JNDI 이름을 지정합니다. `java:comp/env/jdbc` 하위 컨텍스트에서 JDBC 자원의 JNDI 이름을 찾을 수 있습니다. 예를 들어, 급여 데이터베이스의 자원에 대한 JNDI 이름은 `java:comp/env/jdbc/payrolldb`일 수 있습니다. 모든 자원 JNDI 이름이 `java:comp/env` 하위 컨텍스트로 되어 있기 때문에 관리 콘솔에서 JDBC 자원의 JNDI 이름을 지정할 때 `jdbc/`*name*만 입력합니다. 예를 들어, 급여 데이터베이스의 경우 `jdbc/payrolldb`를 지정합니다.

관리 콘솔을 사용하여 JDBC 자원을 만들려면 자원 > JDBC 자원을 선택합니다. 다음과 같이 자원 설정을 지정합니다.

- JNDI 이름: 고유 이름을 지정합니다. JNDI 이름은 카드 카탈로그가 도서관 내 책의 위치를 조직화하여 나타내는 것과 유사한 방식으로 분산 컴퓨팅 환경 내에서 구성 요소를 조직화하고 구성 요소의 위치를 찾아냅니다. 이에 따라 JNDI 이름은 JDBC 자원에 액세스하는 중요한 방법이 됩니다. 일반적으로 이 이름은 `jdbc/` 문자열로 시작합니다. 예를 들면 다음과 같습니다. `jdbc/payrolldb`. 슬래시를 반드시 입력합니다.
- 풀 이름: 새 JDBC 자원에 연관시킬 연결 풀을 선택합니다.
- 설명: 자원에 대한 간단한 설명을 입력합니다.
- 상태: 자원을 사용할 수 없게 하려면 사용 확인란을 선택 해제합니다. 기본적으로 자원은 만들자마자 사용(활성화)할 수 있습니다.

명령줄 유틸리티를 사용하여 JDBC 자원을 만들려면 `create-jdbc-resource` 명령을 사용합니다.

JDBC 연결 풀 만들기

JDBC 자원을 만들려면 연관된 연결 풀을 지정하십시오. 여러 JDBC 자원이 단일 연결 풀을 지정할 수 있습니다.

JDBC 연결 풀은 특정 데이터베이스에 대해 재사용 가능한 연결 그룹입니다. 새로운 물리적 연결을 만드는 데 시간이 많이 소모되기 때문에 서버에서는 사용 가능한 연결 풀을 유지 관리하여 성능을 증가시킵니다. 응용 프로그램은 연결을 요청할 때 풀에서 연결을 가져옵니다. 응용 프로그램에서 연결을 닫으면 연결이 풀로 반환됩니다.

연결 풀을 만들 때 특정 데이터베이스에 대한 연결 부분을 실제로 정의합니다. 풀을 만들기 전에 먼저 JDBC 드라이버를 설치 및 통합해야 합니다. 연결 풀의 등록 정보는 데이터베이스 공급업체에 따라 다를 수 있습니다. 일부 공통된 등록 정보는 데이터베이스 이름(URL), 사용자 이름 및 비밀번호입니다.

JDBC 드라이버와 관련된 특정 데이터 및 데이터베이스 공급업체를 입력해야 합니다. 계속하기 전에 다음 정보를 수집합니다.

- 데이터베이스 공급업체 이름
- 자원 유형: `javax.sql.DataSource`(로컬 트랜잭션에만 해당), `javax.sql.XADataSource`(전역 트랜잭션) 등
- 데이터 소스 클래스 이름: JDBC 드라이버에 자원 유형 및 데이터베이스에 대한 데이터 소스 클래스가 있으면 데이터 소스 클래스 이름 필드 값이 필요합니다.
- 필수 등록 정보: 데이터베이스 이름(URL), 사용자 이름 및 비밀번호 등

JDBC 연결 풀은 특정 데이터베이스에 대해 재사용 가능한 연결 그룹입니다. 관리 콘솔에서 풀을 만들 때 관리자가 특정 데이터베이스에 대한 연결 부분을 실제로 정의합니다.

풀을 만들기 전에 먼저 JDBC 드라이버를 설치 및 통합해야 합니다. 연결 풀 만들기 페이지를 구축할 때 JDBC 드라이버와 관련된 특정 데이터와 데이터베이스 공급업체를 입력해야 합니다. 계속하기 전에 다음 정보를 수집합니다.

- 데이터베이스 공급업체 이름
- 자원 유형: `javax.sql.DataSource`(로컬 트랜잭션에만 해당), `javax.sql.XADataSource`(전역 트랜잭션) 등
- 데이터 소스 클래스 이름
- 필수 등록 정보: 데이터베이스 이름(URL), 사용자 이름 및 비밀번호 등

설치한 JDBC 드라이버에서 지정된 대로 일반 설정 값을 정의합니다. 이 설정은 Java 프로그래밍 언어로 된 인터페이스나 클래스의 이름입니다.

매개 변수	설명
데이터 소스 클래스 이름	`DataSource` 및/또는 `XADataSource` API를 구현하는 공급업체별 클래스 이름입니다. 이 클래스는 JDBC 드라이버에 있습니다.
자원 유형	선택 항목에는 `javax.sql.DataSource`(로컬 트랜잭션에만 해당), `javax.sql.XADataSource`(전역 트랜잭션) 및 `java.sql.ConnectionPoolDataSource`(로컬 트랜잭션, 성능 향상 가능)가 포함됩니다.

또한, 풀에 상주하는 물리적 데이터베이스 연결 집합을 정의해야 합니다. 응용 프로그램에서 연결을 요청하면 풀에서 연결이 제거되고 응용 프로그램에서 연결을 해제하면 연결이 풀로 반환됩니다.

매개 변수	설명
초기 및 최소 풀 크기	풀의 최소 연결 수입니다. 풀 값에 따라 풀을 먼저 작성하거나 응용 프로그램 서버를 시작할 때 풀에 있는 연결 수도 결정합니다.
최대 풀 크기	풀의 최대 연결 수입니다.
풀 크기 조정 개수	풀이 최소 풀 크기로 줄어들 경우 일괄적으로 크기가 조정됩니다. 이 값은 일괄적으로 처리할 연결 수를 지정합니다. 이 값을 너무 크게 하면 연결 재순환이 지체되며, 너무 작게 하면 효율성이 떨어집니다.
유휴 시간 초과	풀에서 연결이 유휴 상태로 있을 수 있는 최대 시간(초)입니다. 이 시간이 만료되면 연결이 풀에서 제거됩니다.
최대 대기 시간	연결을 요청한 응용 프로그램이 연결 시간 초과까지 대기하는 시간입니다. 기본 대기 시간이 길기 때문에 응용 프로그램이 무기한 중지될 수 있습니다.

선택에 따라 Application Server는 연결을 응용 프로그램에 전달하기 전에 연결을 검증할 수 있습니다. 이렇게 검증을 하면 네트워크 실패나 데이터베이스 서버 충돌로 인해 데이터베이스를 사용할 수 없는 경우 Application Server가 데이터베이스 연결을 자동으로 다시 설정합니다. 연결 검증을 수행하면 추가 오버헤드가 발생하여 성능이 약간 저하됩니다.

매개 변수	설명
연결 검증	연결 검증을 활성화하려면 필수 확인란을 선택합니다.
검증 방법	Application Server는 데이터베이스 연결을 자동 완결, 메타데이터 및 테이블의 세 가지 방법으로 검증할 수 있습니다. 자동 완결 및 메타데이터 - Application Server는 `con.getAutoCommit()` 및 `con.getMetaData()` 메소드를 호출하여 연결을 검증합니다. 그러나, 많은 JDBC 드라이버에서 이러한 호출의 결과를 캐시하기 때문에 항상 신뢰할 수 있는 검증을 제공하는 것은 아닙니다. 이러한 호출의 캐시 여부를 판단하려면 드라이버 공급업체에 확인합니다. 테이블 - 응용 프로그램에서 지정된 데이터베이스 테이블을 쿼리합니다. 테이블이 반드시 필요하며 액세스할 수 있어야 하지만 행은 없어도 됩니다. 행이 많이 있는 기존 테이블이나 이미 자주 액세스하는 테이블은 사용하지 마십시오.
테이블 이름	검증 방법 콤보 상자에서 테이블을 선택한 경우 여기에서 데이터베이스 테이블 이름을 지정합니다.
실패 시	모든 연결 닫기 확인란을 선택한 경우 연결이 한 번 실패하면 Application Server는 풀의 모든 연결을 닫고 연결을 다시 설정합니다. 이 확인란을 선택하지 않으면 개별 연결을 사용할 경우에만 연결이 다시 설정됩니다.

대개 많은 사용자가 동시에 데이터베이스에 액세스하기 때문에 한 트랜잭션에서 데이터를 읽는 동안 다른 트랜잭션에서 해당 데이터를 업데이트할 수 있습니다. 트랜잭션의 격리 수준은 업데이트되는 데이터를 다른 트랜잭션에 표시하는 정도를 정의합니다. 격리 수준에 대한 자세한 내용은 데이터베이스 공급업체의 설명서를 참조하십시오.

매개 변수	설명
트랜잭션 격리	이 풀의 연결에 대한 트랜잭션 격리 수준을 선택할 수 있게 합니다. 이 확인란을 선택하지 않으면 연결은 JDBC 드라이버에서 제공하는 기본 격리 수준으로 실행됩니다.

매개 변수	설명
격리 수준 보장	격리 수준이 지정된 경우에만 적용할 수 있습니다. 보장 확인란이 선택된 경우 풀에서 가져온 모든 연결은 동일한 격리 수준을 갖게 됩니다. 예를 들어, 마지막으로 사용 시 연결의 격리 수준을 프로그래밍 방식으로 변경한 경우(예: `con.setTransactionIsolation` 사용) 이 메커니즘은 상태를 지정한 격리 수준으로 다시 변경합니다.

JDBC 자원 및 연결 풀을 함께 작업하는 방법

데이터 저장과 구성 및 검색을 위해 대부분의 응용 프로그램은 관계형 데이터베이스를 사용합니다. Java EE 응용 프로그램은 JDBC API를 통해 관계형 데이터베이스에 액세스합니다. 데이터베이스에 액세스하려면 먼저 응용 프로그램에서 연결해야 합니다.

런타임 시 응용 프로그램이 데이터베이스에 연결할 경우 다음과 같은 현상이 발생합니다.

1. 응용 프로그램이 JNDI API를 통해 호출하여 데이터베이스에 연관된 JDBC 자원(데이터 소스)을 가져옵니다.

   자원의 JNDI 이름으로 이름 지정 및 디렉토리 서비스에서 JDBC 자원을 찾습니다. JDBC 자원마다 연결 풀을 지정합니다.
2. JDBC 자원을 통해 응용 프로그램은 데이터베이스 연결을 가져옵니다.

   반면, Application Server는 데이터베이스에 해당하는 연결 풀에서 물리적인 연결을 검색합니다. 풀은 데이터베이스 이름(URL), 사용자 이름 및 비밀번호 같은 연결 속성을 정의합니다.
3. 이제 데이터베이스에 연결되어 응용 프로그램에서 데이터베이스의 데이터를 읽고 수정하며 추가할 수 있습니다.

   응용 프로그램이 JDBC API를 호출하여 데이터베이스에 액세스합니다. JDBC 드라이버가 응용 프로그램의 JDBC 호출을 데이터베이스 서버의 프로토콜로 변환합니다.
4. 데이터베이스 액세스를 완료하면 응용 프로그램에서 연결을 닫습니다.

   Application Server가 연결을 연결 풀로 반환합니다. 연결이 풀로 반환되면 다음 응용 프로그램에 연결을 사용할 수 있습니다.

데이터베이스 액세스 설정

데이터베이스 액세스를 설정하려면 먼저 지원되는 데이터베이스 제품을 설치해야 합니다. Application Server가 지원하는 데이터베이스 제품 목록에 대한 자세한 내용은 **릴리스 노트**를 참조하십시오. 그런 다음 데이터베이스 제품의 JDBC 드라이버를 설치하고 드라이버의 JAR 파일이 도메인의 서버 인스턴스에 액세스할 수 있도록 설정해야 합니다. 이후 데이터베이스와 데이터베이스에 대한 연결 풀 및 연결 풀을 가리키는 JDBC 자원을 만듭니다.

JDBC 드라이버를 설치한 후에는 드라이버가 응용 프로그램의 JDBC 호출을 데이터베이스 서버의 프로토콜로 변환할 수 있도록 드라이버를 통합해야 합니다. JDBC 드라이버를 관리 도메인에 통합하려면 다음 중 하나를 수행합니다.

- 드라이버가 공통 클래스 로더에 액세스할 수 있게 합니다.
  1. 드라이버의 JAR 및 ZIP 파일을 *domain-dir*`/lib` 디렉토리로 복사하거나 해당 클래스 파일을 *domain-dir*`/lib/ext` 디렉토리로 복사합니다.
  2. 도메인을 다시 시작합니다.
- 드라이버가 시스템 클래스 로더에 액세스할 수 있게 합니다.
  1. 관리 콘솔에서 원하는 구성에 대한 JVM 설정을 선택합니다.
  2. 드라이버의 JAR 파일에 대한 정규화된 경로 이름을 지정합니다.
  3. 설정을 저장하고 서버를 다시 시작합니다.

지속성 관리자 자원

이 기능은 역방향 호환성을 위해 필요합니다. Application Server의 버전 7에서 컨테이너 관리 지속성 Bean(EJB 구성 요소 유형)이 있는 응용 프로그램을 실행하기 위해서는 지속성 관리자 자원이 필요했습니다. 대신 JDBC 자원을 사용하는 것이 좋습니다.

관리 콘솔을 사용하여 지속성 관리자 자원을 만들려면 자원 노드를 선택합니다. 지속성 관리자 노드 > 지속성 관리자 페이지로 이동합니다. 명령줄 유틸리티를 사용하려면 `create-persistence-resource` 명령을 사용합니다.

◆ ◆ ◆ 4 장

4 JMS(Java Message Service) 자원 구성

이 장에서는 JMS(Java Message Service) API를 사용하는 응용 프로그램의 자원 구성 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

JMS 자원 정보

Application Server의 JMS 공급자

Application Server는 Sun Java System Message Queue(이전의 Sun ONE Message Queue) 소프트웨어를 Application Server에 통합하여 JMS(Java Message Service) API를 구현합니다. 기본적인 JMS API 관리 작업에는 Application Server 관리 콘솔을 사용합니다. Message Queue 클러스터 관리를 포함한 고급 작업에는 *MQ-install-dir*`/imq/bin` 디렉토리에 제공된 도구를 사용합니다.

Message Queue 관리에 대한 자세한 내용은 **Message Queue 관리 설명서**를 참조하십시오.

JMS 자원

JMS(Java Message Service) API에서는 두 종류의 관리 대상 객체를 사용합니다.

- 연결 팩토리, 응용 프로그램에서 다른 JMS 객체를 프로그래밍 방식으로 만들 수 있게 해주는 객체
- 대상, 메시지를 위한 저장소 역할

이러한 객체는 관리상의 목적으로 만들며 객체를 만드는 방법은 JMS 구현마다 다릅니다. Application Server에서 다음 작업을 수행합니다.

- 연결 팩토리 자원을 만들어 연결 팩토리 만들기
- 다음 두 객체를 만들어 대상 만들기
  - 물리적 대상
  - 물리적 대상을 참조하는 대상 자원

JMS 응용 프로그램에서는 JNDI API를 사용하여 연결 팩토리와 대상 자원에 액세스합니다. JMS 응용 프로그램은 대개 최소한 연결 팩토리 하나와 대상 하나를 사용합니다. 만들 자원을 알아보려면 응용 프로그램을 살펴보거나 응용 프로그램 개발자에게 문의하십시오.

연결 팩토리에는 다음과 같은 세 가지 유형이 있습니다.

- `QueueConnectionFactory` 객체 - 지점간 통신에 사용됩니다.
- `TopicConnectionFactory` 객체 - 게시-가입 통신에 사용됩니다.
- `ConnectionFactory` 객체 - 지점간 통신과 게시-가입 통신 모두에 사용할 수 있으므로 새로운 응용 프로그램에 권장합니다.

대상에는 다음과 같은 두 가지 종류가 있습니다.

- `Queue` 객체 - 지점간 통신에 사용됩니다.
- `Topic` 객체 - 게시-가입 통신에 사용됩니다.

*J2EE 1.4 Tutorial*의 JMS 장에서는 이 두 가지 통신 유형과 JMS의 기타 요소를 자세히 설명합니다(`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html` 참조).

자원을 만드는 순서는 상관이 없습니다.

J2EE 응용 프로그램의 경우 다음과 같이 Application Server 배포 설명자의 연결 팩토리와 대상 자원을 지정합니다.

- `resource-ref` 또는 `mdb-connection-factory` 요소의 연결 팩토리 JNDI 이름을 지정합니다.
- Message-Driven Bean의 `ejb` 요소와 `message-destination` 요소에서 대상 자원 JNDI 이름을 지정합니다.
- Enterprise Bean 배포 설명자의 `message-driven` 요소 또는 `message-destination-ref` 요소 내에서 `message-destination-link` 요소의 물리적 대상 이름을 지정합니다. 이 이름을 `message-destination` 요소에도 지정합니다. `message-destination-ref` 요소는 새 응용 프로그램에서 더 이상 사용되지 않는 `resource-env-ref` 요소를 대체합니다. Application Server 배포 설명자의 `message-destination` 요소에서 물리적 대상 이름을 대상 자원 이름과 연결합니다.

JMS 자원 및 커넥터 자원의 관계

Application Server는 jmsra라고 하는 시스템 자원 어댑터를 사용하여 JMS를 구현합니다. 사용자가 JMS 자원을 만들면 Application Server가 커넥터 자원을 자동으로 만들며 이는 관리 콘솔의 트리 보기에 있는 커넥터 노드에 표시됩니다.

사용자가 만든 JMS 연결 팩토리마다 Application Server가 커넥터 연결 풀과 연결 자원을 만듭니다. 사용자가 만든 JMS 대상마다 Application Server가 관리 객체 자원을 만듭니다. 사용자가 JMS 자원을 삭제하면 Application Server에서 커넥터 자원을 자동으로 삭제합니다.

JMS 자원 노드 대신 관리 콘솔의 커넥터 노드를 사용하여 JMS 시스템 자원 어댑터에 대한 커넥터 자원을 만들 수 있습니다.자세한 내용은 7 장을 참조하십시오.

JMS 연결 팩토리

JMS 연결 팩토리는 응용 프로그램에서 다른 JMS 객체를 프로그래밍 방식으로 만들 수 있게 해주는 객체입니다. 이 관리 대상 객체는 ConnectionFactory, QueueConnectionFactory 및 TopicConnectionFactory 인터페이스를 구현합니다. Application Server 관리 콘솔을 사용하여 JMS 연결 팩토리를 만들고 편집하거나 삭제할 수 있습니다. 새 JMS 연결 팩토리를 만들면 팩토리 및 커넥터 자원에 대한 커넥터 연결 풀도 만들어집니다.

명령줄 유틸리티를 사용하여 JMS 연결 팩토리를 관리하려면 create-jms-resource, list-jms-resources 또는 delete-jms-resource 명령을 사용합니다.

JMS 대상 자원

JMS 대상은 메시지 저장소의 역할을 합니다. 관리 콘솔을 사용하여 JMS 대상 자원을 만들고 수정하거나 삭제할 수 있습니다. 새 JMS 대상 자원을 만들려면 자원 > JMS 자원 > 대상 자원을 선택합니다. 대상 자원 페이지에서 다음 사항을 지정할 수 있습니다.

- 자원의 JNDI 이름. JMS 자원에 대해 이름 지정 하위 컨텍스트 접두어 jms/를 사용하는 것이 좋습니다. 예를 들면 다음과 같습니다. jms/Queue.
- 자원 유형. javax.jms.Topic 또는 javax.jms.Queue일 수 있습니다.
- 대상 자원의 추가 등록 정보. 이러한 모든 설정 및 추가 등록 정보에 대한 자세한 내용은 관리 콘솔 온라인 도움말을 참조하십시오.

명령줄 유틸리티를 사용하여 JMS 대상을 관리하려면 create-jms-resource 또는 delete-jms-resource 명령을 사용합니다.

정보 – asadmin create-jms-resource 명령에 addresslist 등록 정보(host:mqport,host2:mqport,host3:mqport 형식)를 지정하려면 \\를 사용하여 :을 이스케이프 처리합니다. 예를 들면 host1\\:mqport,host2\\:mqport,host3\\:mpqport입니다.

이스케이프 문자 사용에 대한 자세한 내용은 asadmin(8) 설명서 페이지를 참조하십시오.

JMS 물리적 대상

작업 목적을 위해 항상 물리적인 대상을 만듭니다. 그러나 개발 및 테스트 단계 중에는 이 단계가 필요하지 않습니다. 응용 프로그램이 처음 대상 자원에 액세스할 때 Message Queue가 대상 자원의 이름 등록 정보에서 지정한 물리적 대상을 자동으로 만듭니다. 물리적 대상은 임시적이며 Message Queue 구성 등록 정보에서 지정한 기간 후에는 만료됩니다.

관리 콘솔에서 물리적 대상을 만들려면 구성 > 물리적 대상을 선택합니다. 물리적 대상 만들기 페이지에서 물리적 대상의 이름을 지정하고 대상의 유형(topic 또는 queue)을 선택합니다. 물리적 대상 페이지의 필드 및 등록 정보에 대한 자세한 내용은 관리 콘솔 온라인 도움말을 참조하십시오.

작업 목적을 위해 항상 물리적인 대상을 만듭니다. 그러나 개발 및 테스트 단계 중에는 이 단계가 필요하지 않습니다. 응용 프로그램에서 처음 대상 자원에 액세스할 때 대상 자원의 등록 정보에서 지정한 물리적 대상을 Message Queue에서 자동으로 만듭니다. 물리적 대상은 임시적이며 Message Queue 구성 등록 정보에서 지정한 기간 후에는 만료됩니다.

명령줄 유틸리티를 사용하여 JMS 물리적 대상을 관리하려면 create-jmsdest, flush-jmsdest 또는 delete-jmsdest 명령을 사용합니다.

JMS 공급자

JMS 공급자에 대한 일반 등록 정보 구성

관리 콘솔의 JMS 서비스 페이지를 사용하여 모든 JMS 연결에서 사용할 등록 정보를 구성합니다. 관리 콘솔에서 구성 > Java Message Service를 선택합니다. JMS 서비스 페이지에서 다음 일반 JMS 설정을 제어할 수 있습니다.

- Application Server에서 JMS 서비스가 시작될 때까지 대기한 후 서버 시작이 중단되는 시간을 나타내는 시작 시간 초과 기간을 선택합니다.
- JMS 서비스를 관리할 위치(로컬 또는 원격 호스트)를 결정하는 JMS 서비스 유형을 선택합니다.

- 시작 인수를 지정하여 JMS 서비스 시작을 사용자 정의합니다.
- 다시 연결 확인란을 선택하여 연결이 끊어진 경우 JMS 서비스에서 메시지 서버 또는 `AddressList`의 주소 목록에 다시 연결할지 여부를 지정합니다.
- 다시 연결 간격(초)을 지정합니다. `AddressList`의 각 주소에 대한 재연결 시도와 목록의 연속된 주소에 이 값이 적용됩니다. 이 시간 간격이 너무 짧을 경우 브로커가 복구할 시간이 없습니다. 너무 길 경우에는 지연이 지나치게 길게 느껴질 수 있습니다.
- 다시 연결 시도 횟수를 지정합니다. 클라이언트 런타임에서 `AddressList`의 주소에 연결(또는 다시 연결)을 몇 번 시도한 후 해당 목록의 다음 주소로 연결을 시도할지를 필드에 입력합니다.
- 기본 JMS 호스트를 선택합니다.
- 주소 목록 동작 드롭다운 목록에서 `AddressList`의 주소 순서대로(`priority`) 다시 연결을 시도할지 아니면 임의의 순서대로(`random`) 다시 연결을 시도할지 선택합니다.
- 주소 목록 반복 필드에 연결을 설정하거나 재설정할 때 JMS 서비스가 `AddressList`를 반복하는 횟수를 입력합니다.
- 기본값이 아닌 체계나 서비스를 사용할 경우 MQ 체계 및 MQ 서비스 필드에 Message Queue 주소 체계 이름과 Message Queue 연결 서비스 이름을 입력합니다.

위의 모든 등록 정보 값은 런타임 시에도 업데이트할 수 있습니다. 그러나, 등록 정보가 업데이트된 후 만들어진 해당 연결 팩토리만 업데이트된 값을 갖습니다. 기존 연결 팩토리는 계속 원래 등록 정보 값을 갖습니다. 또한, 거의 대부분의 경우 해당 값을 적용하려면 Application Server를 다시 시작해야 합니다. Application Server를 다시 시작하지 않아도 업데이트할 수 있는 등록 정보는 기본 JMS 호스트뿐입니다.

명령줄 유틸리티를 사용하여 JMS 공급자를 관리하려면 `set` 또는 `jms-ping` 명령을 사용합니다.

원격 서버 액세스

공급자와 호스트를 원격 시스템으로 변경하면 모든 JMS 응용 프로그램이 원격 서버에서 실행됩니다. 로컬 서버와 하나 이상의 원격 서버를 모두 사용하려면 원격 서버에 액세스하는 연결을 만드는 AddressList 등록 정보를 사용하여 연결 팩토리 자원을 만듭니다.

외부 JMS 공급자

JMS용 일반 자원 어댑터는 IBM Websphere MQ, Tibco EMS 및 Sonic MQ와 같은 외부 JMS 공급자의 JMS 클라이언트 라이브러리를 래핑할 수 있으며 이에 따라 모든 JMS 공급자를 Sun Java System Application Server와 같은 Java EE 1.4 응용 프로그램 서버에 통합할 수 있는 Java EE Connector 1.5 자원 어댑터입니다. 이 어댑터는 `.rar` 아카이브로서, Java EE 1.4 응용 프로그램 서버의 관리 도구를 사용하여 배포 및 구성할 수 있습니다.

JMS용 일반 자원 어댑터 구성

응용 프로그램 서버의 관리 도구를 사용하여 JMS용 일반 자원 어댑터를 배포하고 구성할 수 있습니다. 이 절에서는 Sun Java System Application Server에 JMS용 일반 자원 어댑터를 구성하는 방법에 대해 설명합니다. 일반적으로 자원 어댑터는 JMS 공급자가 XA를 지원하는지 여부를 나타내도록 구성할 수 있습니다. 또한, JMS 공급자와의 가능한 통합 모드를 나타내도록 구성할 수도 있습니다. 자원 어댑터는 두 가지 통합 모드를 지원합니다. 첫 번째 모드는 JNDI를 통합 수단으로 사용합니다. 이 경우, 관리 대상 객체는 JMS 공급자의 JNDI 트리에 설정되며 일반 자원 어댑터에서 사용할 목적으로 조회됩니다. 해당 모드가 통합에 적합하지 않은 경우 JMS 관리 대상 객체 javabean 클래스의 Java 리플렉션을 통합 모드로 사용할 수도 있습니다. Sun Java System Application Server의 관리 콘솔 또는 CLI를 사용하여 자원 어댑터를 구성할 수 있습니다. 이 방법은 다른 자원 어댑터를 구성하는 방법과 다르지 않습니다.

일반 자원 어댑터 구성

자원 어댑터를 배포하기 전에 JMS 클라이언트 라이브러리를 Application Server에 사용할 수 있어야 합니다. 일부 JMS 공급자의 경우 클라이언트 라이브러리에 원시 라이브러리가 포함되어 있을 수도 있습니다. 이 경우, 원시 라이브러리도 Application Server JVM에 사용할 수 있어야 합니다.

1. 커넥터 모듈을 배포한 것과 동일한 방식으로 일반 자원 어댑터를 배포합니다.

   이에 대한 작업 단계는 관리 콘솔 온라인 도움말을 참조하십시오. 배포 중 `install-dir/lib/addons/resourceadapters/genericjmsra/genericra.rar`을 일반 자원 어댑터의 위치로 지정해야 합니다. 또한, 61 페이지 “자원 어댑터 등록 정보” 절에 설명된 등록 정보를 지정해야 합니다.

2. 커넥터 연결 풀을 만듭니다.

   이에 대한 작업 단계는 관리 콘솔 온라인 도움말을 참조하십시오. 새 커넥터 연결 풀 페이지의 자원 어댑터 콤보 상자에서 `genericra`를 선택하고 연결 정의 콤보 상자에서 `javax.jms.QueueConnectionFactory`를 선택합니다. 또한, 65 페이지 “ManagedConnectionFactory 등록 정보” 절에 설명된 등록 정보를 지정합니다.

3. 커넥터 자원을 만듭니다.

   이 작업에 대한 자세한 절차는 관리 콘솔 온라인 도움말을 참조하십시오. 새 커넥터 자원 페이지에서, 이전 단계에서 만든 풀을 선택합니다.

4. 관리 대상 객체 자원을 만듭니다.

   이 작업에 대한 자세한 절차는 관리 콘솔 온라인 도움말을 참조하십시오. 새 관리 객체 자원 페이지에서 `genericra`를 자원 어댑터로, `javax.jms.Queue`를 자원 유형으로 선택합니다. 다음을 누르고 두 번째 페이지에서 등록 정보 추가를 누릅니다. 추가 등록 정보 테이블에서 `DestinationProperties`라는 새 등록 정보를 `Name\\=clientQueue` 값으로 지정합니다. 다른 등록 정보에 대한 자세한 내용은 65 페이지 “관리 대상 객체 자원 등록 정보” 절을 참조하십시오.

5. Sun Java System Application Server의 보안 정책을 다음과 같이 변경합니다.

- sjsas_home/domains/domain1/config/server.policy를 수정하여 java.util.logging.LoggingPermission "control"을 추가합니다.
- sjsas_home/lib/appclient/client.policy를 수정하여 permission javax.security.auth.PrivateCredentialPermission "javax.resource.spi.security.PasswordCredential * \"*\"","read";를 추가합니다.

자원 어댑터 등록 정보

다음 표는 자원 어댑터를 만들 때 사용할 등록 정보를 나타냅니다.

등록 정보 이름	유효 값	기본값	설명
ProviderIntegrationMode	javabean/jndi	javabean	자원 어댑터와 JMS 클라이언트 간의 통합 모드를 결정합니다.
ConnectionFactory ClassName	Application Server 클래스 경로에 사용할 수 있는 클래스 이름. 예를 들면 다음과 같습니다. com.sun.messaging.ConnectionFactory	없음	JMS 클라이언트의 javax.jms.ConnectionFactory 구현 클래스 이름입니다. ProviderIntegrationMode가 javabean으로 지정된 경우에 사용됩니다.
QueueConnectionFactory ClassName	Application Server 클래스 경로에 사용할 수 있는 클래스 이름. 예를 들면 다음과 같습니다. com.sun.messaging.QueueConnectionFactory	없음	JMS 클라이언트의 javax.jms.QueueConnectionFactory 구현 클래스 이름입니다. ProviderIntegrationMode가 javabean으로 지정된 경우에 사용됩니다.
TopicConnectionFactory ClassName	Application Server 클래스 경로에 사용할 수 있는 클래스 이름. 예를 들면 다음과 같습니다. com.sun.messaging.TopicConnectionFactory	없음	JMS 클라이언트의 javax.jms.TopicConnectionFactory 구현 클래스 이름입니다. ProviderIntegrationMode가 javabean으로 지정된 경우에 사용됩니다.
XAConnectionFactory ClassName	Application Server 클래스 경로에 사용할 수 있는 클래스 이름. 예를 들면 다음과 같습니다. com.sun.messaging.XAConnectionFactory	없음	JMS 클라이언트의 javax.jms.ConnectionFactory 구현 클래스 이름입니다. ProviderIntegrationMode가 javabean으로 지정된 경우에 사용됩니다.

등록 정보 이름	유효 값	기본값	설명
XAQueueConnectionFactory ClassName	Application Server 클래스 경로에 사용할 수 있는 클래스 이름. 예를 들면 다음과 같습니다. com.sun.messaging. XAQueueConnectionFactory	없음	JMS 클라이언트의 javax.jms.XAQueueConnection Factory 구현 클래스 이름입니다. ProviderIntegrationMode가 javabean으로 지정된 경우에 사용됩니다.
XATopicConnectionFactory ClassName	Application Server 클래스 경로에 사용할 수 있는 클래스 이름. 예를 들면 다음과 같습니다. com.sun.messaging. XATopicConnectionFactory	없음	JMS 클라이언트의 javax.jms.XATopicConnection Factory 구현 클래스 이름입니다. ProviderIntegrationMode가 javabean으로 지정된 경우에 사용됩니다.
TopicClassName	Application Server 클래스 경로에 사용할 수 있는 클래스 이름. 예를 들면 다음과 같습니다. com.sun.messaging.Topic	없음	JMS 클라이언트의 javax.jms.Topic 구현 클래스 이름입니다. ProviderIntegrationMode가 javabean으로 지정된 경우에 사용됩니다.
QueueClassName	Application Server 클래스 경로에 사용할 수 있는 클래스 이름. 예를 들면 다음과 같습니다. com.sun.messaging.Queue	없음	JMS 클라이언트의 javax.jms.Queue 구현 클래스 이름입니다. ProviderIntegrationMode가 javabean으로 지정된 경우에 사용됩니다.
SupportsXA	True/false	FALSE	JMS 클라이언트가 XA를 지원할지 여부를 지정합니다.
ConnectionFactory Properties	쉼표로 구분된 이름 값 쌍	없음	javabean 등록 정보 이름 및 JMS 클라이언트의 ConnectionFactory 값을 지정합니다. ProviderIntegrationMode가 javabean으로 지정된 경우에만 사용됩니다.
JndiProperties	쉼표로 구분된 이름 값 쌍	없음	JMS 공급자의 JNDI에 연결하는 데 사용할 JNDI 공급자 등록 정보를 지정합니다. ProviderIntegrationMode가 jndi로 지정된 경우에만 사용됩니다.

등록 정보 이름	유효 값	기본값	설명
CommonSetterMethodName	메소드 이름	없음	일부 JMS 공급업체가 자체 관리 대상 객체의 등록 정보를 설정하는 데 사용하는 일반 setter 메소드 이름을 지정합니다. ProviderIntegrationMode가 javabean으로 지정된 경우에만 사용됩니다. Sun Java System Message Queue의 경우 이 등록 정보의 이름은 setProperty입니다.
UserName	JMS 사용자 이름	없음	JMS 공급자 연결에 사용할 사용자 이름입니다.
Password	JMS 사용자 비밀번호	없음	JMS 공급자 연결에 사용할 비밀번호입니다.

등록 정보 이름	유효 값	기본값	설명
RMPolicy	ProviderManaged 또는 OnePerPhysicalConnection	ProviderManaged	트랜잭션 관리자에서 XAResource의 isSameRM 메소드를 사용하여 두 XAResources가 나타내는 자원 관리자 인스턴스가 동일한지 확인합니다. RMPolicy가 ProviderManaged(기본값)로 설정되면, JMS 공급자는 RMPolicy를 결정하게 되며 일반 자원 어댑터의 XAResource 래퍼는 isSameRM 호출을 메시지 대기열 공급자의 XA 자원 구현에 위임하기만 합니다. 이 기능은 대부분의 메시지 대기열 제품에 대해 완벽하게 작동해야 합니다. IBM MQ 시리즈와 같은 일부 XAResource 구현은 물리적 연결당 하나의 자원 관리자를 사용합니다. 이 경우, 동일한 대기열 관리자에 대한 인바운드 및 아웃바운드 통신이 단일 트랜잭션 내에 있으면(예: MDB가 대상에 응답을 보냄) 문제가 발생합니다. RMPolicy가 OnePerPhysicalConnection으로 설정되면, 일반 자원 어댑터에 있는 XAResource 래퍼 구현의 isSameRM은 래핑된 객체에 위임하기 전에 두 XAResource가 동일한 물리적 연결을 사용하는지 확인합니다. 이 등록 정보에 대한 자세한 내용은 Glassfish 웹 사이트에서 이슈 트래커 데이터베이스의 이슈 번호 5를 참조하십시오.

ManagedConnectionFactory 등록 정보

ManagedConnectionFactory 등록 정보는 connector-connection-pool을 만들 때 지정됩니다. 자원 어댑터를 만들 때 지정된 모든 ManagedConnectionFactory 등록 정보는 대체될 수 있습니다. ManagedConnectionFactory에만 사용할 수 있는 추가 등록 정보는 다음과 같습니다.

등록 정보 이름	유효 값	기본값	설명
ClientId	유효 클라이언트 아이디	없음	JMS 1.1 사양으로 지정된 ClientID입니다.
ConnectionFactoryJndiName	JNDI 이름	없음	JMS 공급자의 JNDI 트리에 바인딩된 연결 팩토리의 JNDI 이름입니다. 관리자는 JMS 공급자에 모든 연결 팩토리 등록 정보(clientID 제외)를 제공해야 합니다. 이 등록 정보 이름은 ProviderIntegrationMode가 jndi로 지정된 경우에만 사용됩니다.
ConnectionValidation Enabled	true/false	false	true로 설정하면 자원 어댑터는 예외 Listener를 사용하여 연결 예외를 포착하고 CONNECTION_ERROR_OCCURED 이벤트를 Application Server에 보냅니다.

관리 대상 객체 자원 등록 정보

이 등록 정보는 관리 대상 객체 자원을 만들 때 지정됩니다. 관리 대상 객체 자원의 모든 자원 어댑터 등록 정보는 대체될 수 있습니다. 관리 대상 객체 자원에만 사용할 수 있는 추가 등록 정보는 다음과 같습니다.

등록 정보 이름	유효 값	기본값	설명
DestinationJndiName	JNDI 이름	없음	JMS 공급자의 JNDI 트리에 바인딩된 대상의 JNDI 이름입니다. 관리자는 JMS 공급자에 모든 등록 정보를 제공해야 합니다. 이 등록 정보 이름은 ProviderIntegrationMode가 jndi로 지정된 경우에만 사용됩니다.
Destination Properties	쉼표로 구분된 이름 값 쌍	없음	이 등록 정보는 javabean 등록 정보 이름 및 JMS 클라이언트의 대상 값을 지정합니다. ProviderIntegrationMode가 javabean으로 지정된 경우에만 사용됩니다.

활성화 사양 등록 정보

이 등록 정보는 MDB의 Sun 특정 배포 설명자에 activation-config-properties로 지정되어 있습니다. 활성화 사양의 모든 자원 어댑터 등록 정보는 대체될 수 있습니다. 활성화 사양에만 사용할 수 있는 추가 등록 정보는 다음과 같습니다.

등록 정보 이름	유효한 값	기본값	설명
MaxPoolSize	정수	8	동시 메시지 전달을 수행하기 위해 자원 어댑터에서 내부적으로 만든 서버 세션 풀의 최대 크기입니다. 이 값은 MDB 객체의 최대 풀 크기와 같아야 합니다.
MaxWaitTime	정수	3	자원 어댑터는 이 등록 정보에 지정된 시간(초) 동안 내부 풀에서 서버 세션을 가져오기 위해 대기합니다. 이 제한이 초과되면 메시지 전달이 실패합니다.
SubscriptionDurability	영구적 또는 비영구적	비영구적	JMS 1.1 사양에서 지정된 SubscriptionDurability입니다.
SubscriptionName		없음	JMS 1.1 사양에서 지정된 SubscriptionName입니다.
MessageSelector	유효 메시지 선택기	없음	JMS 1.1 사양에서 지정된 MessageSelector입니다.

등록 정보 이름	유효한 값	기본값	설명
ClientID	유효 클라이언트 아이디	없음	JMS 1.1 사양에서 지정된 ClientID입니다.
ConnectionFactoryJndiName	유효 JNDI 이름	없음	JMS 공급자에 만들어진 연결 팩토리의 JNDI 이름입니다. 자원 어댑터가 이 연결 팩토리를 사용하여 메시지를 수신하기 위한 연결을 만듭니다. ProviderIntegrationMode가 jndi로 지정된 경우에만 사용됩니다.
DestinationJndiName	유효 JNDI 이름	없음	JMS 공급자에 만들어진 대상의 JNDI 이름입니다. 자원 어댑터가 이 대상을 사용하여 메시지를 수신하기 위한 연결을 만듭니다. ProviderIntegrationMode가 jndi로 지정된 경우에만 사용됩니다.
DestinationType	javax.jms.Queue 또는 javax.jms.Topic	null	MDB가 수신할 대상 유형입니다.
DestinationProperties	쉼표로 구분된 이름 값 쌍	없음	이 등록 정보는 javabean 등록 정보 이름 및 JMS 클라이언트의 대상 값을 지정합니다. ProviderIntegrationMode가 javabean으로 지정된 경우에만 사용됩니다.
RedeliveryAttempts	정수		메시지로 인해 MDB 런타임 예외가 발생한 경우 메시지를 전달할 시간입니다.
RedeliveryInterval	시간(초)		메시지로 인해 MDB 런타임 예외가 발생한 경우의 재전달 시도 간격입니다.
SendBadMessagesToDMD	true/false	false	전달 시도 횟수가 초과된 경우 자원 어댑터가 해당 메시지를 사용 불능 메시지 대상으로 보내야 할지를 나타냅니다.
DeadMessageDestinationJndiName	유효한 JNDI 이름	없음	JMS 공급자에 만들어진 대상의 JNDI 이름입니다. 이 값은 사용 불능 메시지를 위한 목표 대상입니다. ProviderIntegrationMode가 jndi로 지정된 경우에만 사용됩니다.

등록 정보 이름	유효한 값	기본값	설명
DeadMessageDestination ClassName	대상 객체의 클래스 이름	없음	ProviderIntegrationMode가 javabean으로 지정된 경우에 사용됩니다.
DeadMessageDestination Properties	쉼표로 구분된 이름 값 쌍	없음	이 등록 정보는 javabean 등록 정보 이름 및 JMS 클라이언트의 대상 값을 지정합니다. 이는 ProviderIntegrationMode가 javabean으로 지정된 경우에만 사용됩니다.
ReconnectAttempts	정수		연결 시 예외 Listener에서 오류를 포착한 경우 다시 연결을 시도할 시간입니다.
ReconnectInterval	시간(초)		다시 연결 시도 간격입니다.

◆ ◆ ◆ 5 장

5 JavaMail 자원 구성

Application Server에는 JavaMail API가 포함되어 있습니다. JavaMail API는 메일 시스템을 모델링하는 추상 API 집합입니다. API는 메일 및 메시징 응용 프로그램을 빌드할 수 있는 플랫폼과 프로토콜에 독립적인 프레임워크를 제공합니다. JavaMail API는 전자 메시지를 읽고, 작성하고, 보내는 기능을 제공합니다. 서비스 공급자가 특정 프로토콜을 구현합니다. JavaMail API를 사용하면 응용 프로그램에 전자 메일 기능을 추가할 수 있습니다. JavaMail을 사용하면 Java 응용 프로그램이 네트워크 또는 인터넷상의 메일 서버인 IMAP(Internet Message Access Protocol) 및 SMTP(Simple Mail Transfer Protocol)에 액세스할 수 있습니다. JavaMail은 메일 서버 기능을 제공하지 않으므로 JavaMail을 사용하려면 메일 서버에 액세스해야 합니다.

JavaMail API에 대한 자세한 내용은 JavaMail 웹 사이트(`http://java.sun.com/products/javamail/index.html`)를 참조하십시오.

이 장은 다음 내용으로 구성되어 있습니다.

JavaMail 세션 만들기

Application Server에 사용할 수 있도록 JavaMail을 구성하려면 Application Server 관리 콘솔에서 메일 세션을 만듭니다. 그러면 서버측 구성 요소 및 응용 프로그램이 사용자가 할당한 세션 등록 정보를 사용하여 JNDI가 포함된 JavaMail 서비스에 액세스할 수 있습니다. 메일 세션을 만들 때 JavaMail을 사용하는 구성 요소에서 등록 정보를 설정할 필요가 없도록 관리 콘솔에서 메일 호스트, 전송 및 저장소 프로토콜과 기본 메일 사용자를 지정할 수 있습니다. Application Server는 단일 세션 객체를 만들고 JNDI를 통해 필요로 하는 모든 구성 요소에서 사용할 수 있도록 하기 때문에 대규모 전자 메일 사용자를 처리하는 응용 프로그램에 도움이 됩니다.

관리 콘솔을 사용하여 JavaMail 세션을 만들려면 자원 > JavaMail 세션을 선택합니다. 다음과 같이 JavaMail 설정을 지정합니다.

- JNDI 이름: 메일 세션에 대한 고유 이름입니다. JavaMail 자원에 이름 지정 하위 컨텍스트 접두어 `mail/`을 사용합니다. 예를 들면 `mail/MySession`과 같습니다.
- 메일 호스트: 기본 메일 서버의 호스트 이름입니다. 프로토콜 관련 호스트 등록 정보를 제공하지 않으면 저장소 및 전송 객체의 연결 메소드에서 이 값을 사용합니다. 이름을 실제 호스트 이름으로 확인할 수 있어야 합니다.
- 기본 사용자: 메일 서버에 연결할 때 입력할 사용자 이름입니다. 프로토콜 관련 아이디 등록 정보를 제공하지 않으면 저장소 및 전송 객체의 연결 메소드에서 이 값을 사용합니다.
- 기본 반송 주소: 기본 사용자의 전자 메일 주소이며 형식은 *username*@*host.domain*과 같습니다.
- 설명: 구성 요소에 대한 설명을 제공합니다.
- 세션: 이 경우 메일 세션을 활성화하지 않으려면 사용 가능 확인란을 선택 해제합니다.

또한, 메일 공급자가 기본값 이외의 저장소 또는 전송 프로토콜을 사용하도록 재구성된 경우에만 다음 고급 설정을 정의합니다.

- 저장소 프로토콜: 사용할 저장소 객체의 통신 방법을 정의합니다. 기본적으로 저장소 프로토콜은 `imap`로 지정됩니다.
- 저장소 프로토콜 클래스: 사용하려는 저장소 프로토콜을 구현하는 저장소 통신 방법 클래스를 제공합니다. 기본적으로 저장소 프로토콜 클래스는 `com.sun.mail.imap.IMAPStore`로 지정됩니다.
- 전송 프로토콜: 전송 통신 방법을 지정합니다. 기본적으로 전송 프로토콜은 `smtp`로 지정됩니다.
- 전송 프로토콜 클래스: 전송 클래스의 통신 방법을 정의합니다. 기본적으로 전송 프로토콜 클래스는 `com.sun.mail.smtp.SMTPTransport`로 지정됩니다.
- 디버그: 이 메일 세션에 대해 프로토콜 추적을 비롯한 추가 디버깅 출력을 활성화하려면 이 확인란을 선택하십시오. JavaMail 로그 수준을 `FINE` 이상으로 설정한 경우 디버깅 출력이 생성되어 시스템 로그 파일에 포함됩니다. 로그 수준 설정에 대한 자세한 내용은 146 페이지 “사용자 정의 로그 수준 설정”을 참조하십시오.
- 추가 등록 정보: 프로토콜 관련 호스트 또는 사용자 이름 등록 정보와 같은 응용 프로그램에서 필요한 등록 정보를 만듭니다. JavaMail API 설명서에 사용 가능한 등록 정보가 나열되어 있습니다.

◆ ◆ ◆ 6 장

6 JNDI 자원

JNDI(Java Naming and Directory Interface)는 다른 종류의 이름 지정 및 디렉토리 서비스에 액세스하는 데 필요한 API(Application Programming Interface)입니다. Java EE 구성 요소는 JNDI 조회 메소드를 호출하여 객체를 찾습니다.

JNDI는 Java Naming and Directory Interface API의 머리글자입니다. 응용 프로그램은 이 API를 호출하여 자원과 다른 프로그램 객체를 찾습니다. 자원은 데이터베이스 서버나 메시징 시스템 같은 시스템과의 연결을 제공하는 프로그램 객체입니다. JDBC 자원을 데이터 소스라고도 합니다. 모든 자원 객체는 고유하고 사용자에게 친숙한 이름으로 식별되며 JNDI라는 이름으로 부릅니다. 자원 객체와 해당 JNDI 이름은 Application Server에 포함된 이름 지정 및 디렉토리 서비스에 의해 함께 바인딩됩니다. 자원을 새로 만들려면 새로운 이름 객체 바인딩을 JNDI에 입력합니다.

이 장은 다음 내용으로 구성되어 있습니다.

J2EE 이름 지정 서비스

JNDI 이름은 사람들에게 친숙한 객체 이름입니다. 이러한 이름은 J2EE 서버에서 제공하는 이름 지정 및 디렉토리 서비스에 의해 객체에 바인딩됩니다. J2EE 구성 요소가 JNDI API를 통해 이 서비스에 액세스하기 때문에 대개 객체는 해당 JNDI 이름을 사용합니다. 예를 들어, Pointbase 데이터베이스의 JNDI 이름은 `jdbc/Pointbase`입니다. Application Server는 시작할 때 구성 파일로부터 정보를 읽어 JNDI 데이터베이스 이름을 이름 공간에 자동으로 추가합니다.

J2EE 응용 프로그램 클라이언트, Enterprise Bean 및 웹 구성 요소에는 JNDI 이름 지정 환경에 대한 액세스 권한이 필요합니다.

응용 프로그램 구성 요소의 이름 지정 환경은 배포나 어셈블 중에 응용 프로그램 구성 요소 비즈니스 논리의 사용자 정의를 허용하는 메커니즘입니다. 응용 프로그램 구성 요소 환경을 사용하면 응용 프로그램 구성 요소의 소스 코드에 액세스하거나 변경할 필요 없이 응용 프로그램 구성 요소를 사용자 정의할 수 있습니다.

J2EE 컨테이너는 응용 프로그램 구성 요소의 환경을 구현하며 이러한 환경을 응용 프로그램 구성 요소 인스턴스에 JNDI 이름 지정 컨텍스트로 제공합니다. 응용 프로그램 구성 요소의 환경은 다음과 같이 사용됩니다.

- 응용 프로그램 구성 요소의 비즈니스 메소드는 JNDI 인터페이스를 사용하여 환경에 액세스합니다. 응용 프로그램 구성 요소 공급자는 해당 응용 프로그램 구성 요소가 런타임에 자체 환경 내에서 필요로 하는 모든 환경 항목을 배포 설명자에 선언합니다.
- 컨테이너는 응용 프로그램 구성 요소 환경을 저장하는 JNDI 이름 지정 컨텍스트 구현을 제공합니다. 컨테이너는 배포자가 각 응용 프로그램 구성 요소의 환경을 만들고 관리할 수 있게 해주는 도구도 제공합니다.
- 배포자는 컨테이너에서 제공하는 도구를 사용하여 응용 프로그램 구성 요소 배포 설명자에 선언되어 있는 환경 항목을 초기화합니다. 배포자가 환경 항목의 값을 설정 및 수정합니다.
- 컨테이너는 런타임 시 환경 이름 지정 컨텍스트를 응용 프로그램 구성 요소 인스턴스에서 사용할 수 있게 합니다. 응용 프로그램 구성 요소의 인스턴스는 JNDI 인터페이스를 사용하여 환경 항목의 값을 가져옵니다.

각 응용 프로그램 구성 요소는 고유 환경 항목 집합을 정의합니다. 같은 컨테이너 내의 모든 응용 프로그램 구성 요소 인스턴스는 같은 환경 항목을 공유합니다. 응용 프로그램 구성 요소 인스턴스는 런타임 시 환경을 수정할 수 없습니다.

이름 지정 참조 및 바인딩 정보

자원 참조는 자원에 대한 구성 요소의 코드화된 이름을 식별하는 배포 설명자의 요소입니다. 즉, 코드화된 이름은 자원의 연결 팩토리를 참조합니다. 다음 절의 해당 예에서 자원 참조 이름은 `jdbc/SavingsAccountDB`입니다.

자원의 JNDI 이름과 자원 참조의 이름은 같지 않습니다. 이 방법으로 이름을 지정하려면 배포 전에 두 이름을 매핑해야 하지만 자원으로부터 구성 요소를 분리하기도 합니다. 이러한 분리 기능으로 인해 나중에 구성 요소가 다른 자원에 액세스해야 할 경우 이름을 변경할 필요가 없습니다. 또한 이러한 융통성으로 인해 기존의 구성 요소로부터 J2EE 응용 프로그램을 어셈블하기가 쉽습니다.

다음 표에서는 Application Server에서 사용하는 J2EE 자원에 대한 JNDI 조회 및 관련 참조를 나열합니다.

표 6–1 JNDI 조회 및 관련 참조

JNDI 조회 이름	관련 참조
`java:comp/env`	응용 프로그램 환경 항목
`java:comp/env/jdbc`	JDBC 데이터 소스 자원 관리자 연결 팩토리

표 6–1 JNDI 조회 및 관련 참조 (계속)

JNDI 조회 이름	관련 참조
java:comp/env/ejb	EJB 참조
java:comp/UserTransaction	UserTransaction 참조
java:comp/env/mail	JavaMail 세션 연결 팩토리
java:comp/env/url	URL 연결 팩토리
java:comp/env/jms	JMS 연결 팩토리 및 대상
java:comp/ORB	응용 프로그램 구성 요소 간에 공유되는 ORB 인스턴스

사용자 정의 자원 사용

사용자 정의 자원은 로컬 JNDI 저장소를 액세스하고 외부 자원은 외부 JNDI 저장소를 액세스합니다. 두 가지 유형의 자원 모두 사용자 지정 팩토리 클래스 요소, JNDI 이름 속성 등을 필요로 합니다. 이 절에서는 J2EE 자원에 대해 JNDI 연결 팩토리 자원을 구성하는 방법과 이러한 자원에 액세스하는 방법에 대해 설명합니다.

Application Server 내에서 `list-jndi-entities` 뿐만 아니라 자원을 작성, 삭제 및 나열할 수 있습니다.

외부 JNDI 저장소 및 자원 사용

Application Server에서 실행 중인 응용 프로그램에서 외부 JNDI 저장소에 저장된 자원에 액세스해야 할 경우가 자주 있습니다. 예를 들어, LDAP 서버에 일반 Java 객체를 Java 스키마별로 저장할 수 있습니다. 외부 JNDI 자원 요소를 사용하면 이러한 외부 자원 저장소를 구성할 수 있습니다. 외부 JNDI 팩토리는 `javax.naming.spi.InitialContextFactory` 인터페이스를 구현해야 합니다.

다음은 외부 JNDI 자원 사용 예입니다.

```
<resources>
 <!-- external-jndi-resource element specifies how to access J2EE resources
 -- stored in an external JNDI repository. The following example
 -- illustrates how to access a java object stored in LDAP.
 -- factory-class element specifies the JNDI InitialContext factory that
 -- needs to be used to access the resource factory. property element
 -- corresponds to the environment applicable to the external JNDI context
 -- and jndi-lookup-name refers to the JNDI name to lookup to fetch the
 -- designated (in this case the java) object.
 -->
  <external-jndi-resource jndi-name="test/myBean"
```

```
      jndi-lookup-name="cn=myBean"
      res-type="test.myBean"
      factory-class="com.sun.jndi.ldap.LdapCtxFactory">
    <property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
    <property name="SECURITY_AUTHENTICATION" value="simple" />
    <property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
    <property name="SECURITY_CREDENTIALS" value="changeit" />
  </external-jndi-resource>
</resources>
```

◆ ◆ ◆ 7 장

7 커넥터 자원

Java 구성 요소인 커넥터 모듈(자원 어댑터라고도 함)을 사용하면 응용 프로그램이 EIS(Enterprise Information Systems)와 상호 작용할 수 있습니다. EIS 소프트웨어에는 전사적 자원 관리(ERP), 메인프레임 트랜잭션 처리, 비관계형 데이터베이스 등의 다양한 시스템 유형이 포함됩니다. 커넥터 모듈을 설치하려면 다른 Java 모듈에서와 마찬가지로 해당 모듈을 배포합니다.

커넥터 연결 풀은 특정 EIS에 대한 재사용 가능한 연결 그룹입니다. 커넥터 연결 풀을 만들려면 해당 풀과 연관된 커넥터 모듈(자원 어댑터)을 지정합니다.

커넥터 자원은 응용 프로그램과의 EIS 연결을 제공하는 프로그램 객체입니다. 커넥터 자원을 만들려면 해당 JNDI 이름 및 연관된 연결 풀을 지정합니다. 여러 커넥터 자원에서 단일 연결 풀을 지정할 수 있습니다. 응용 프로그램은 해당 JNDI 이름을 조회하여 자원을 찾습니다. EIS에 대한 커넥터 자원의 JNDI 이름은 대개 `java:comp/env/`*eis-specific* 하위 컨텍스트로 되어 있습니다. Application Server 9는 커넥터 모듈(자원 어댑터)을 사용하여 JMS를 구현합니다.

이 장은 다음 내용으로 구성되어 있습니다.

커넥터 연결 풀

다음 표에서는 연결 풀 설정에 대해 설명합니다.

매개 변수	설명
초기 및 최소 풀 크기	풀의 최소 연결 수입니다. 풀 값에 따라 풀을 먼저 작성하거나 응용 프로그램 서버를 시작할 때 풀에 있는 연결 수도 결정합니다.
최대 풀 크기	풀의 최대 연결 수입니다.
풀 크기 조정 개수	풀이 최소 풀 크기로 줄어들 경우 일괄적으로 크기가 조정됩니다. 이 값은 일괄적으로 처리할 연결 수를 지정합니다. 이 값을 너무 크게 하면 연결 재순환이 지연되고, 너무 작게 하면 효율성이 떨어집니다.
유휴 시간 초과	풀에서 연결이 유휴 상태로 있을 수 있는 최대 시간(초)입니다. 이 시간이 만료되면 풀에서 연결이 제거됩니다.
최대 대기 시간	연결을 요청한 응용 프로그램이 연결될 때까지 기다리는 시간으로 이 시간이 지나면 연결 시간 초과가 됩니다. 기본 대기 시간이 길기 때문에 응용 프로그램이 무기한 중지될 수 있습니다.
실패 시	모든 연결 닫기 확인란을 선택한 경우 단일 연결이 실패하면 응용 프로그램 서버는 풀의 모든 연결을 닫은 다음 다시 연결합니다. 확인란을 선택하지 않은 경우 개별 연결을 사용한 경우에만 연결이 다시 설정됩니다
트랜잭션 지원	트랜잭션 지원 목록을 사용하여 연결 풀에 대한 트랜잭션 지원 유형을 선택합니다. 선택한 트랜잭션 지원은 이 연결 풀과 연관된 자원 어댑터의 트랜잭션 지원 속성을 역호환 방식으로 대체합니다. 즉, 자원 어댑터에 지정한 수준보다 낮은 트랜잭션 수준을 지원하거나 자원 어댑터에서 지정한 것과 같은 트랜잭션 수준을 지원할 수 있지만 더 높은 수준은 지정할 수 없습니다. 트랜잭션 지원 메뉴에서 없음을 선택하면 자원 어댑터가 자원 관리자 로컬 또는 JTA 트랜잭션을 지원하지 않고 `XAResource` 또는 `LocalTransaction` 인터페이스를 구현하지 않음을 나타냅니다. JAXR 자원 어댑터의 경우 트랜잭션 지원 메뉴에서 없음을 선택해야 합니다. JAXR 자원 어댑터는 로컬 또는 JTA 트랜잭션을 지원하지 않습니다. 로컬 트랜잭션 지원은 자원 어댑터가 `LocalTransaction` 인터페이스를 구현하여 로컬 트랜잭션을 지원하는 것을 나타냅니다. 로컬 트랜잭션은 자원 관리자 내부로 관리되고 외부 트랜잭션 관리자를 포함시키지 않습니다. XA 트랜잭션 지원은 자원 어댑터가 `LocalTransaction` 및 `XAResource` 인터페이스를 구현하여 자원 관리자 로컬 및 JTA 트랜잭션을 지원함을 나타냅니다. 트랜잭션은 자원 관리자 외부에 있는 트랜잭션 관리자가 제어 및 조정합니다. 로컬 트랜잭션은 자원 관리자 내부로 관리되고 외부 트랜잭션 관리자를 포함시키지 않습니다.
커넥터 검증	응용 프로그램에 연결하기 전에 연결 풀을 검증하려면 사용 가능 확인란을 선택합니다.

연결 풀을 만들기 전에 해당 풀과 연관된 커넥터 모듈(자원 어댑터)을 배포해야 합니다. 관리 콘솔 또는 `asadmin` 명령을 사용하여 커넥터 모듈을 배포할 수 있습니다. `asadmin` 명령에 대한 자세한 내용은 `asadmin`(1M)을 참조하십시오.

연결 풀을 보고, 만들고, 편집하거나 삭제하려면 관리 콘솔에서 자원 > 커넥터 > 커넥터 연결 풀을 누릅니다. 커넥터 연결 풀에 등록 정보(이름 값 쌍)를 추가할 수 있습니다. 또는, 다음 `asadmin` 명령을 사용하여 연결 풀을 만들고 삭제할 수 있습니다.

- `create-connector-connection-pool`(1)
- `delete-connector-connection-pool`(1)

커넥터 자원

커넥터 자원은 응용 프로그램을 EIS(Enterprise Information System)에 연결합니다. 모든 커넥터 자원은 연결 풀과 연관되어 있습니다. 커넥터 자원을 보고, 만들고, 편집하거나 삭제하려면 관리 콘솔에서 자원 > 커넥터 > 커넥터 자원을 누릅니다. 또는, 다음 `asadmin` 명령을 사용하여 연결 자원을 만들고 삭제할 수 있습니다.

- `create-connector-resource`(1)
- `delete-connector-resource`(1)

관리 대상 객체 자원

관리 대상 객체는 응용 프로그램에 전문적 기능을 제공합니다. 예를 들어, 자원 어댑터와 이와 연관된 EIS 특정 구문 분석기에 액세스할 수 있는 기능을 제공합니다. 관리 대상 객체를 보고, 만들고, 편집하거나 삭제하려면 관리 콘솔에서 자원 > 커넥터 > 관리 대상 객체 자원을 누릅니다.

- `create-admin-object`(1)
- `delete-admin-object-1`(1)

◆ ◆ ◆ 8 장

8 J2EE 컨테이너

이 장에서는 서버에 포함된 J2EE 컨테이너를 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

J2EE 컨테이너 유형

J2EE 컨테이너에서는 응용 프로그램 구성 요소에 대한 런타임 지원을 제공합니다. J2EE 응용 프로그램 구성 요소에서는 컨테이너의 프로토콜 및 메소드를 사용하여 서버에서 제공하는 다른 응용 프로그램 구성 요소와 서비스에 액세스합니다. Application Server는 응용 프로그램 클라이언트 컨테이너, 애플릿 컨테이너, 웹 컨테이너 및 EJB 컨테이너를 제공합니다. 컨테이너를 표시하는 다이어그램은 26 페이지 "Application Server 구조" 절을 참조하십시오.

웹 컨테이너

웹 컨테이너는 웹 응용 프로그램을 호스트하는 J2EE 컨테이너입니다. 웹 컨테이너는 개발자에게 서블릿 및 JavaServer Pages(JSP 파일)를 실행하는 환경을 제공함으로써 웹 서버 기능을 확장합니다.

EJB 컨테이너

Enterprise Bean(EJB 구성 요소)은 비즈니스 논리를 포함하는 프로그래밍 언어 서버 구성 요소입니다. EJB 컨테이너는 Enterprise Bean에 대한 로컬 및 원격 액세스를 제공합니다.

Enterprise Bean에는 Session Bean, Entity Bean 및 Message-Driven Bean의 3가지 유형이 있습니다. Session Bean은 임시 객체와 프로세스를 나타내며 대개 단일 클라이언트에서

사용합니다. Entity Bean은 일반적으로 데이터베이스에서 유지 관리되는 지속성 데이터를 나타냅니다. Message-Driven Bean은 응용 프로그램 모듈과 서비스에 비동기적으로 메시지를 전달하기 위해 사용합니다.

컨테이너는 Enterprise Bean을 만들고 다른 응용 프로그램 구성 요소가 Enterprise Bean을 액세스할 수 있도록 이를 이름 지정 서비스에 바인딩하며, 권한이 있는 클라이언트만 Enterprise Bean 메소드에 액세스할 수 있게 하고, Bean의 상태를 영구 저장소에 저장하고 Bean의 상태를 캐싱하고 필요한 경우 Bean을 활성화하거나 비활성화하는 일을 담당합니다.

J2EE 컨테이너 구성

일반 웹 컨테이너 설정 구성

이 릴리스에는 관리 콘솔의 웹 컨테이너에 대한 컨테이너 차원의 설정이 없습니다.

웹 컨테이너 세션 구성

이 절에서는 웹 컨테이너의 HTTP 세션 설정에 대해 설명합니다. HTTP 세션은 영구 저장소에 상태 데이터를 기록하는 고유한 웹 세션입니다.

세션 시간 초과 값 구성

관리 콘솔을 사용하여 HTTP 세션 시간 초과 값을 설정합니다. 세션 시간 초과 값은 HTTP 세션이 유효한 기간을 나타냅니다.

관리 콘솔에서 구성 > 웹 컨테이너 > 세션 등록 정보로 이동하십시오. 세션 시간 초과 필드에 세션이 유효한 시간(초)을 입력합니다.

세션 시간 초과 값 설정에 대한 자세한 지침을 보려면 관리 콘솔에서 도움말을 누르십시오.

관리자 등록 정보 구성

세션 관리자는 세션을 만들고 삭제하는 방법과 세션 상태 저장 위치 및 최대 세션 수를 구성할 수 있는 방법을 제공합니다.

관리 콘솔에서 세션 관리자 설정을 변경하려면 구성 > 웹 컨테이너 > 관리자 등록 정보로 이동하십시오.

- 구성할 인스턴스를 선택합니다.

  특정 인스턴스를 구성하려면 해당 인스턴스 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server에 대해 server-config 노드를 선택합니다.

  모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.

관리자 등록 정보 탭에서 다음 등록 정보를 설정합니다.

- 리프 간격 값. 리프 간격은 비활성 세션 데이터가 저장소에서 삭제되기까지의 시간(초)입니다.
- 최대 세션 값. 최대 세션 필드는 허용된 최대 세션 수입니다.
- 세션 파일 이름 값을 설정합니다. 세션 파일 이름 필드는 세션 데이터를 포함하는 파일입니다.
- 세션 아이디 생성기 클래스 이름 값.

  세션 아이디 생성기 클래스 이름 필드를 사용하면 고유한 세션 아이디를 생성하는데 필요한 사용자 정의 클래스를 지정할 수 있습니다. 서버 인스턴스당 세션 아이디 생성기 클래스 하나만 허용되므로 클러스터의 모든 인스턴스는 세션 키 충돌을 방지하기 위해 동일한 세션 아이디 생성기를 사용해야 합니다.

  사용자 정의 세션 아이디 생성기 클래스는 com.sun.enterprise.util.uuid.UuidGenerator 인터페이스를 구현해야 합니다.

```
package com.sun.enterprise.util.uuid;

public interface UuidGenerator {

    public String generateUuid();
    public String generateUuid(Object obj);  //obj is the session object
}
```

  클래스가 Application Server 클래스 경로에 있어야 합니다.

관리자 등록 정보 설정에 대한 자세한 지침을 보려면 관리 콘솔에서 도움말을 누르십시오.

저장소 등록 정보 구성

세션 저장소 데이터를 저장할 위치를 지정하려면 관리 콘솔에서 구성 > 웹 컨테이너 > 저장소 등록 정보로 이동하십시오.

- 구성할 인스턴스를 선택합니다.

  특정 인스턴스를 구성하려면 해당 인스턴스 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server에 대해 server-config 노드를 선택합니다.

  모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.

- 리프 간격 필드를 설정합니다.
  리프 간격 필드는 비활성 세션 데이터가 저장소에서 삭제되기까지의 시간(초)입니다.
- 세션 데이터를 저장할 디렉토리를 지정합니다.

세션 저장소 등록 정보 설정에 대한 자세한 지침을 보려면 관리 콘솔에서 도움말을 누르십시오.

가상 서버 설정 구성

Application Server를 설치할 때 Application Server 인스턴스의 기본 가상 서버가 만들어집니다. 이 가상 서버의 기본 `docroot`는 *instance-dir* `domains/domain1/docroot`에 만들어지며 *instance_name*`/docroot`와 동기화됩니다. 가상 서버는 사용자가 만든 각 추가 Application Server 인스턴스당 하나씩 만들어집니다.

일반 EJB 설정 구성

이 절에서는 서버의 모든 Enterprise Bean 컨테이너에 적용되는 다음 설정에 대해 설명합니다.

- 82 페이지 “세션 저장 위치”
- 83 페이지 “EJB 풀 설정 구성”
- 83 페이지 “EJB 캐시 설정 구성”

기본값을 컨테이너별로 대체하려면 Enterprise Bean의 `sun-ejb-jar.xml` 파일에서 값을 조정합니다. 자세한 내용은 **Application Server Developer’s Guide**를 참조하십시오.

세션 저장 위치

세션 저장 위치 필드는 비활성화된 Bean과 영구 HTTP 세션을 파일 시스템에 저장하는 디렉토리를 지정합니다.

비활성화된 Bean은 파일 시스템의 파일에 상태를 기록한 Enterprise Bean입니다. 일반적으로 비활성화된 Bean은 일정 기간 동안 유휴 상태에 있으므로 현재 클라이언트가 액세스할 수 없습니다.

비활성화된 Bean과 마찬가지로 영구 HTTP 세션은 파일 시스템의 파일에 자신의 상태를 기록한 개별 웹 세션입니다.

완결 옵션 필드는 컨테이너가 트랜잭션 간의 비활성화된 Entity Bean 인스턴스를 캐시하는 방법을 지정합니다.

옵션 B는 트랜잭션 간의 Entity Bean 인스턴스를 캐시하며 기본적으로 선택되어 있습니다. 옵션 C는 캐싱을 비활성화합니다.

EJB 풀 설정 구성

Bean을 작성하여 생기는 성능 저하 없이 클라이언트 요청에 응답할 수 있도록 컨테이너는 Enterprise Bean 풀을 유지 관리합니다. 이 설정은 Stateless Session Bean 및 Entity Bean에만 적용됩니다.

배포된 Enterprise Bean을 사용하는 응용 프로그램에서 성능 문제가 발생하면 풀을 만들거나 기존 풀이 관리하는 Bean 수를 늘려서 응용 프로그램의 성능을 증가시킬 수 있습니다.

기본적으로 컨테이너는 Enterprise Bean의 풀을 유지 관리합니다.

EJB 캐시 설정 구성

컨테이너는 자주 사용되는 Enterprise Bean에 대해 Enterprise Bean 데이터 캐시를 유지 관리합니다. 이렇게 하면 컨테이너가 다른 응용 프로그램 모듈의 Enterprise Bean 데이터 요청에 빨리 응답할 수 있습니다. 이 절은 Stateful Session Bean과 Entity Bean에만 적용됩니다.

캐시된 Enterprise Bean의 상태는 활성, 유휴 또는 비활성 중 하나입니다. 활성화된 Enterprise Bean은 현재 클라이언트가 액세스하고 있습니다. 유휴 Enterprise Bean의 데이터는 현재 캐시에 있지만 Bean에 액세스하는 클라이언트가 없습니다. 비활성화된 Bean의 데이터는 임시로 저장되지만 클라이언트가 Bean을 요청할 경우 다시 캐시로 읽어 들입니다.

Message-Driven Bean 설정 구성

Message-Driven Bean의 풀은 83 페이지 "EJB 풀 설정 구성"에 설명된 Session Bean의 풀과 유사합니다. 기본적으로 컨테이너는 Message-Driven Bean의 풀을 관리합니다.

이 풀의 구성을 조정하려면 다음 작업을 수행합니다.

EJB 타이머 서비스 설정 구성

타이머 서비스는 Enterprise Bean 컨테이너에서 사용하는 알림이나 이벤트를 예약하기 위해 컨테이너에서 제공하는 영구적인 트랜잭션 알림 서비스입니다. Stateful Session Bean을 제외한 모든 Enterprise Bean은 타이머 서비스를 통해 알림을 받을 수 있습니다. 서비스에 설정된 타이머는 서버가 종료되거나 다시 시작되어도 삭제되지 않습니다.

◆ ◆ ◆ 9 장

9 보안 구성

보안이란 데이터를 보호하는 것이며 데이터를 저장하거나 전송할 때 해당 데이터에 대한 인증 되지않은 액세스나 손상을 방지하는 방법입니다. Application Server에는 J2EE 표준을 기반으로 하는 확장 가능한 동적 보안 구조가 있습니다. 내장된 보안 기능에는 암호화 도구 인증 및 권한 부여 공개 키 인프라가 포함됩니다. Application Server는 시스템이나 사용자에 대한 잠재적인 위험 없이 응용 프로그램을 안전하게 실행할 수 있는 샌드 박스를 사용하는 Java 보안 모델에 구축되어 있습니다. 이 장은 다음 내용으로 구성되어 있습니다.

응용 프로그램 및 시스템 보안 이해

대개, 다음과 같은 두 종류의 응용 프로그램 보안이 있습니다.

- **프로그래밍 방식의 보안**의 경우 개발자가 작성한 응용 프로그램 코드에서 보안 작업을 처리합니다. 관리자는 이 메커니즘에 대한 제어 권한이 없습니다. 일반적으로, J2EE 컨테이너를 통해 구성을 관리하는 대신 응용 프로그램에 보안 구성을 하드 코드하기 때문에 프로그래밍 방식의 보안을 권장하지 않습니다.

- **선언적 보안**의 경우 컨테이너(Application Server)가 응용 프로그램의 배포 설명자를 통해 보안을 처리합니다. 배포 설명자를 직접 편집하거나 `deploytool` 같은 도구를 사용하여 선언적 보안을 제어할 수 있습니다. 응용 프로그램을 개발한 후 배포 설명자를 변경할 수 있기 때문에 선언적 보안을 사용하면 더 많은 유연성이 훨씬 더 허용됩니다.

응용 프로그램 보안 외에 Application Server 시스템의 모든 응용 프로그램에 영향을 미치는 **시스템 보안**도 있습니다.

프로그래밍 방식의 보안은 응용 프로그램 개발자가 제어하므로 이 문서에서 설명하지 않습니다. 선언적 보안의 경우 덜 제어되므로 이 문서에서 가끔 다룹니다. 이 문서는 기본적으로 시스템 관리자를 대상으로 하므로 시스템 보안에 대해 중점적으로 설명합니다.

보안 관리 도구

Application Server에서는 보안을 관리하기 위한 다음 도구를 제공합니다.

- 관리 콘솔은 전체 서버의 보안을 구성하고 사용자 그룹 및 영역을 관리하며 시스템 차원의 다른 보안 작업을 수행하기 위해 사용하는 브라우저 기반의 도구입니다. 관리 콘솔에 대한 일반적인 정보는 28 페이지 "관리 도구"를 참조하십시오. 관리 콘솔에서 수행할 수 있는 보안 작업에 대한 개요는 98 페이지 "관리 콘솔을 사용하여 보안 관리"를 참조하십시오.
- `asadmin`은 관리 콘솔과 동일한 많은 작업을 수행하는 명령줄 도구입니다. 관리 콘솔로 수행할 수 없는 일부 작업은 `asadmin`으로 수행할 수 있습니다. 명령 프롬프트나 스크립트에서 `asadmin` 명령을 수행하여 반복적인 작업을 자동화할 수 있습니다. `asadmin`에 대한 일반적인 정보는 28 페이지 "관리 도구"를 참조하십시오.
- `deploytool`은 그래픽 패키지 및 배포 도구로, 개별 응용 프로그램의 보안을 제어하기 위해 응용 프로그램 배포 설명자를 편집하는 데 사용합니다. `deploytool`은 응용 프로그램 개발자를 대상으로 하기 때문에 이 설명서에서는 자세히 설명하지 않습니다. `deploytool` 사용에 대한 자세한 내용은 도구의 온라인 도움말과 `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`에 있는 **J2EE 1.4 Tutorial**을 참조하십시오.

Java 2 Platform, Standard Edition(J2SE)에서는 보안을 관리하기 위한 다음 두 가지 도구를 제공합니다.

- `keytool`은 디지털 인증서와 키 쌍을 관리하기 위한 명령줄 유틸리티입니다. `keytool`을 사용하여 `certificate` 영역의 사용자를 관리합니다.
- `policytool`은 시스템 차원의 Java 보안 정책을 관리하기 위한 그래픽 유틸리티입니다. 관리자는 `policytool`을 사용할 필요가 거의 없습니다.

`keytool`, `policytool` 및 다른 Java 보안 도구 사용에 대한 자세한 내용은 `http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html#security`에 있는 **Java 2 SDK Tools and Utilities**를 참조하십시오.

Enterprise Edition의 경우 NSS(Network Security Services)를 구현하는 다른 두 개의 도구를 보안 관리에 사용할 수 있습니다. NSS에 대한 자세한 내용을 보려면 http://www.mozilla.org/projects/security/pki/nss/로 이동하십시오. 보안 관리 도구는 다음과 같습니다.

- `certutil`은 인증서 및 키 데이터베이스를 관리하기 위한 명령줄 유틸리티입니다.
- `pk12util`은 인증서/키 데이터베이스와 PKCS12 형식의 파일 간에 키와 인증서를 내보내고 가져오기 위해 사용하는 명령줄 유틸리티입니다.

`certutil`, `pk12util` 및 다른 NSS 보안 도구 사용에 대한 자세한 내용은 http://www.mozilla.org/projects/security/pki/nss/tools에 있는 **NSS Security Tools**를 참조하십시오.

비밀번호 보안 관리

Application Server의 이번 릴리스에서 특정 도메인에 대한 사양을 포함하는 `domain.xml` 파일에는 기본적으로 Sun Java System 메시지 대기열 브로커의 비밀번호가 일반 텍스트로 포함되어 있습니다. 이 비밀번호를 포함하는 `domain.xml` 파일의 요소는 `jms-host` 요소의 `admin-password` 속성입니다. 설치 시, 이 비밀번호를 변경할 수 없기 때문에 보안에 큰 영향을 미치지 않습니다.

그러나 관리 콘솔을 사용하여 사용자와 자원을 추가하고 이 사용자와 자원에 비밀번호를 지정합니다. 예를 들어, 데이터베이스에 액세스하기 위한 비밀번호처럼 이 비밀번호 중 일부는 `domain.xml` 파일에 일반 텍스트로 기록됩니다. 이 비밀번호를 `domain.xml` 파일에 일반 텍스트로 기록하면 보안 위험이 있을 수 있습니다. `admin-password` 속성이나 데이터베이스 비밀번호와 같은 `domain.xml`의 비밀번호를 암호화할 수 있습니다. 보안 비밀번호 관리에 대한 지침은 다음 내용을 참조하십시오.

- 87 페이지 “domain.xml 파일의 비밀번호 암호화”
- 88 페이지 “암호화된 비밀번호를 사용하여 파일 보호”
- 88 페이지 “마스터 비밀번호 변경”
- 89 페이지 “마스터 비밀번호 및 키 저장소 작업”
- 90 페이지 “관리 비밀번호 변경”

domain.xml 파일의 비밀번호 암호화

`domain.xml` 파일의 비밀번호를 암호화하려면 다음 단계를 수행합니다.

1. `domain.xml` 파일이 있는 디렉토리(기본적으로 *domain-dir*`/config`)에서 다음의 `asadmin` 명령을 실행합니다.

   ```
   asadmin create-password-alias --user admin alias-name
   ```

   예를 들면 다음과 같습니다.

```
asadmin create-password-alias --user admin jms-password
```

비밀번호 프롬프트(이 경우 admin)가 표시됩니다. 자세한 내용은 create-password-alias, list-password-aliases, delete-password-alias 명령에 대한 설명서 페이지를 참조하십시오.

2. domain.xml의 비밀번호를 제거하고 바꿉니다. asadmin set 명령을 사용하여 이를 수행합니다. 다음은 이러한 용도로 set 명령을 사용하는 예입니다.

```
asadmin set --user admin server.jms-service.jms-host.
default_JMS_host.admin-password='${ALIAS=jms-password}'
```

주 – 위의 예와 같이 별칭 비밀번호를 작은 따옴표로 묶습니다.

3. 관련 도메인을 위해 Application Server를 다시 시작합니다.

암호화된 비밀번호를 사용하여 파일 보호

일부 파일에는 파일 시스템 권한을 사용하여 보호해야 하는 암호화된 비밀번호가 포함되어 있습니다. 이 파일에는 다음 내용이 포함되어 있습니다.

- *domain-dir*/master-password

  이 파일에는 암호화된 마스터 비밀번호가 포함되어 있고 파일 시스템 권한 600을 사용하여 이 파일을 보호해야 합니다.

- --passwordfile 인수를 사용하는 인수로 asadmin에 전달하기 위해 만든 모든 비밀번호 파일은 파일 시스템 권한 600을 사용하여 보호해야 합니다.

마스터 비밀번호 변경

마스터 비밀번호(MP)는 전체적으로 공유하는 비밀번호입니다. 마스터 비밀번호는 인증에 사용되지 않고 네트워크를 통해 전송되지 않습니다. 이 비밀번호는 전체적인 보안의 억제 지점입니다. 필요할 경우 수동으로 입력하도록 선택하거나 파일에 숨길 수 있습니다. 비밀번호는 시스템에서 가장 중요한 데이터입니다. 이 파일을 제거하여 MP 요구를 강제로 실행할 수 있습니다. 마스터 비밀번호를 변경하면 마스터 비밀번호가 Java JCEKS 유형 키 저장소인 마스터 비밀번호 키 저장소에 다시 저장됩니다.

마스터 비밀번호를 변경하려면 다음 단계를 수행합니다.

1. 도메인의 Application Server를 중지합니다. asadmin change-master-password 명령을 사용하여 이전 비밀번호와 새 비밀번호에 대한 메시지를 표시한 다음 모든 하위 종속 항목을 다시 암호화합니다. 예를 들면 다음과 같습니다.

```
asadmin change-master-password>
Please enter the master password>
Please enter the new master password>
Please enter the the new master password again>
```

2. Application Server를 다시 시작합니다.

주의 – 이 때 실행 중인 서버 인스턴스를 시작해서는 안되며, 해당 노드의 SMP 에이전트가 변경될 때까지 실행 중인 서버 인스턴스를 다시 시작해서는 안 됩니다. SMP를 변경하기 전에 서버 인스턴스를 다시 시작한 경우 서버 인스턴스가 시작되지 않습니다.

3. 노드 에이전트 및 관련된 서버를 한 번에 하나씩 중지합니다. `asadmin change-master-password` 명령을 다시 실행한 다음 노드 에이전트 및 관련된 서버를 다시 시작합니다.
4. 모든 노드 에이전트를 주소 지정할 때까지 다음 노드 에이전트를 계속합니다. 이런 방법으로 변경 사항 롤백이 수행됩니다.

마스터 비밀번호 및 키 저장소 작업

마스터 비밀번호는 보안 키 저장소에 대한 비밀번호입니다. 새 Application Server 도메인을 만들 때 자체 서명된 인증서가 새로 생성되고 마스터 비밀번호를 사용하여 잠근 관련 키 저장소에 저장됩니다. 마스터 비밀번호가 기본값이 아닌 경우 `start-domain` 명령을 실행하면 마스터 비밀번호를 입력하라는 메시지가 표시됩니다. 올바른 마스터 비밀번호를 입력하면 도메인이 시작됩니다.

도메인과 연관된 노드 에이전트를 만들면 노드 에이전트는 데이터를 도메인과 동기화합니다. 그러는 동안 키 저장소도 함께 동기화됩니다. 이 노드 에이전트가 제어하는 모든 서버 인스턴스는 키 저장소를 열어야 합니다. 이 저장소는 기본적으로 도메인 만들기 프로세스에서 만들어진 저장소와 동일하므로 이와 동일한 마스터 비밀번호를 사용해야 열립니다. 그러나 마스터 비밀번호 자체는 동기화되지 않지만, 즉 마스터 비밀번호가 동기화 중 노드 에이전트에 전송되지 않지만 노드 에이전트에 로컬로 사용할 수 있어야 합니다. 이에 따라 노드 에이전트 만들기 및/또는 시작 시 마스터 비밀번호를 입력하라는 메시지가 표시되며, 이때 도메인 만들기/시작 시에 입력한 것과 동일한 비밀번호를 입력해야 합니다. 도메인의 마스터 비밀번호가 변경되면 이 도메인과 연관된 모든 노드 에이전트에 대해 동일한 단계를 수행하여 해당 마스터 비밀번호를 변경해야 합니다.

관리 비밀번호 변경

관리 비밀번호 암호화는 87 페이지 “비밀번호 보안 관리”에 설명되어 있습니다. 관리 비밀번호를 암호화할 것을 강력하게 권장합니다. 암호화하기 전에 관리 비밀번호를 변경하려면 `asadmin set` 명령을 사용합니다. 다음은 이러한 용도로 `set` 명령을 사용하는 예입니다.

```
asadmin set --user admin server.jms-service.jms-host.default_JMS_host.admin-password=new_pwd
```

다음 절차와 같이 관리 콘솔를 사용하여 관리 비밀번호를 변경할 수도 있습니다.

관리 콘솔을 사용하여 관리 비밀번호를 변경하려면 구성 노드 > 구성할 인스턴스 > 보안 노드 > 영역 노드 > **관리 영역** 노드를 선택하고 필요에 맞게 영역 페이지를 편집합니다.

인증 및 권한 부여 정보

인증 및 권한 부여는 응용 프로그램 서버 보안의 핵심 개념입니다. 인증 및 권한 부여와 관련해서 다음 내용을 설명합니다.

- 90 페이지 “엔티티 인증”
- 91 페이지 “사용자 권한 부여”
- 92 페이지 “JACC 공급자 지정”
- 92 페이지 “인증 및 권한 부여 결정 사항 감사”
- 92 페이지 “메시지 보안 구성”

엔티티 인증

인증은 엔티티(사용자, 응용 프로그램 또는 구성 요소)가 다른 엔티티를 확인하는 방법입니다. 엔티티는 **보안 자격 증명**을 사용하여 자신을 인증합니다. 사용자 이름 및 비밀번호 디지털 인증서 등이 자격 증명이 될 수 있습니다.

일반적으로 인증은 사용자 이름과 비밀번호를 사용하여 응용 프로그램에 로그인하는 것을 의미하지만 서버의 자원을 요청할 경우 EJB 공급 보안 자격 증명을 가리킬 수도 있습니다. 대개, 서버나 응용 프로그램에서는 클라이언트의 인증을 요구합니다. 그리고 클라이언트에서는 서버가 자신을 인증할 것을 요구할 수도 있습니다. 인증이 양방향일 경우 이를 상호 인증이라고 합니다.

엔티티가 보호된 자원에 액세스할 경우 Application Server는 해당 자원에 대해 구성된 인증 체계를 사용하여 액세스를 부여할지 여부를 결정합니다. 예를 들어, 사용자는 웹 브라우저에서 사용자 이름과 비밀번호를 입력할 수 있습니다. 응용 프로그램에서 해당 자격 증명을 확인하면 사용자가 인증됩니다. 세션의 나머지 부분에서 사용자는 이 인증된 보안 아이디와 연관되어 있습니다.

Application Server는 90 페이지 “엔티티 인증”에 설명된 대로 네 가지 인증 유형을 지원합니다. 응용 프로그램은 배포 설명자 내에서 사용하는 인증 유형을 지정합니다. `deploytool`을 사용하여 응용 프로그램에 대한 인증 메소드를 구성하는 방법에 대한 자세한 내용은 `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`에 있는 **J2EE 1.4 Tutorial**을 참조하십시오.

표 9–1 Application Server 인증 방법

인증 방법	통신 프로토콜	설명	사용자 자격 증명 암호화
기본	HTTP(SSL 선택 사항)	서버에 내장된 팝업 로그인 대화 상자를 사용합니다.	SSL을 사용하지 않을 경우 없음
양식 기반	HTTP(SSL 선택 사항)	응용 프로그램에서 고유한 사용자 정의 로그인 및 오류 페이지를 제공합니다.	SSL을 사용하지 않을 경우 없음
클라이언트 인증서	HTTPS(SSL에서의 HTTP)	서버는 공개 키 인증서를 사용하여 클라이언트를 인증합니다.	SSL

단일 사인 온(SSO) 검증

단일 사인 온(SSO)을 사용하면 하나의 가상 서버 인스턴스의 여러 응용 프로그램이 사용자 인증 상태를 공유할 수 있습니다. 단일 사인 온(SSO)을 사용할 경우 사용자가 하나의 응용 프로그램에 로그인하면 동일한 인증 정보를 요구하는 다른 응용 프로그램에 암시적으로 로그인됩니다.

단일 사인 온(SSO)은 그룹을 기반으로 합니다. 배포 설명자가 동일한 **그룹**을 정의하고 동일한 인증 방법(기본, 양식, 다이제스트, 인증서)을 사용하는 모든 웹 응용 프로그램은 단일 사인 온(SSO)을 공유합니다.

Application Server에 대해 정의된 가상 서버에는 기본적으로 단일 사인 온(SSO)이 활성화되어 있습니다.

사용자 권한 부여

사용자가 인증되면 **권한 부여**의 수준은 수행할 수 있는 작업을 결정합니다. 사용자의 권한 부여는 **역할**을 기반으로 합니다. 예를 들어, 인사 관리 응용 프로그램은 관리자에게 모든 직원의 사원 정보를 볼 수 있도록 허용하지만 직원들에게는 자신의 개인 정보만 볼 수 있도록 허용할 수 있습니다. 역할에 대한 자세한 내용은 93 페이지 “사용자, 그룹, 역할 및 영역 이해”를 참조하십시오.

JACC 공급자 지정

JACC(Java Authorization Contract for Containers)는 플러그 가능한 권한 부여 공급자의 인터페이스를 정의하는 J2EE 1.4 사양의 일부입니다. JACC를 사용하면 관리자는 타사 플러그인 모듈에서 권한 부여를 수행하도록 설정할 수 있습니다.

기본적으로 Application Server는 JACC 사양과 함께 컴파일되는 단순한 파일 기반의 권한 부여 엔진을 제공합니다. 다른 타사 JACC 공급자를 지정할 수도 있습니다.

JACC 공급자는 JAAS(Java Authentication and Authorization Service) API를 사용합니다. JAAS를 사용하면 서비스에서 사용자에 따라 액세스 제어를 인증하고 강제할 수 있습니다. JAAS는 표준 PAM(Pluggable Authentication Module) 프레임워크의 기술 버전을 구현합니다.

인증 및 권한 부여 결정 사항 감사

Application Server는 감사 모듈을 통해 모든 인증 및 권한 부여 결정 사항의 **감사 추적**을 제공할 수 있습니다. Application Server는 기본 감사 모듈뿐만 아니라 감사 모듈을 사용자 정의할 수 있는 기능도 제공합니다. 사용자 정의 감사 모듈 개발에 대한 자세한 내용은 Application Server *Developer's Guide*의 *Configuring an Audit Module* 절을 참조하십시오.

메시지 보안 구성

메시지 보안을 사용하면 서버는 메시지 계층에서 웹 서비스 호출 및 응답의 종단간 인증을 수행할 수 있습니다. Application Server는 SOAP 계층에서 메시지 보안 공급자를 사용하여 메시지 보안을 구현합니다. 메시지 보안 공급자는 요청 및 응답 메시지에 필요한 인증 유형과 같은 정보를 제공합니다. 지원되는 인증 유형은 다음과 같습니다.

- 사용자 이름-비밀번호 인증을 포함하는 보낸 사람 인증
- XML 디지털 서명을 포함하는 내용 인증

두 개의 메시지 보안 공급자가 이 릴리스에 포함되어 있습니다. SOAP 계층의 인증을 위해 메시지 보안 공급자를 구성할 수 있습니다. 구성할 수 있는 공급자에는 `ClientProvider`와 `ServerProvider`가 포함됩니다.

메시지 계층 보안 지원이 플러그 가능한 인증 모듈 양식으로 클라이언트 컨테이너와 Application Server에 통합됩니다. 기본적으로 메시지 계층 보안은 Application Server에서 비활성화됩니다.

전체 Application Server나 특정 응용 프로그램 또는 메소드에 대해 메시지 수준 보안을 구성할 수 있습니다. Application Server 수준에서 메시지 보안을 구성하는 방법은 10 장에 설명되어 있습니다. 응용 프로그램 수준에서 메시지 보안을 구성하는 방법은 **Developer's Guide**의 *Securing Applications* 장에 설명되어 있습니다.

사용자, 그룹, 역할 및 영역 이해

Application Server는 다음 엔티티에 대해 인증 및 권한 부여 정책을 실행합니다.

- 93 페이지 "사용자": *Application Server*에 **정의된** 개별 아이디. 일반적으로 사용자는 개인, 소프트웨어 구성 요소(예: Enterprise Bean) 또는 서비스일 수도 있습니다. 인증된 사용자를 *principal*이라고도 합니다. 사용자를 **주체**라고도 합니다.
- 94 페이지 "그룹": 공통된 특성으로 분류된 *Application Server*에 **정의된** 사용자 집합
- 94 페이지 "역할": **응용 프로그램에서 정의한** 이름 지정 권한 부여 수준. 잠금을 연 키와 역할을 비교할 수 있습니다. 많은 사용자가 키 사본을 갖고 있을 수 있습니다. 잠금에서는 액세스하는 사용자는 관계없으며 올바른 키를 사용하는지만 관계가 있습니다.
- 94 페이지 "영역": 사용자와 그룹 정보 및 연관된 보안 자격 증명을 포함하는 저장소. 영역을 **보안 정책 도메인**이라고도 합니다.

주 – 사용자와 그룹은 전체 Application Server에 대해 지정되는 반면, 각 응용 프로그램에서는 고유한 역할을 정의합니다. 응용 프로그램을 패키지화 및 배포할 경우 다음 그림에서 설명한 대로 응용 프로그램은 사용자/그룹 및 역할 간의 매핑을 지정합니다.

사용자

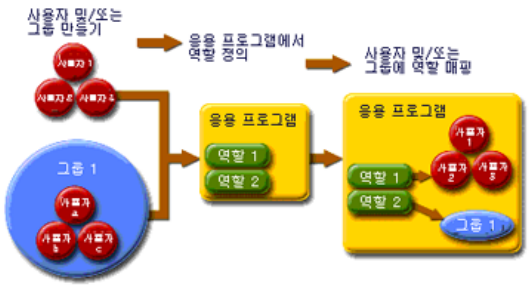

그림 9–1 역할 매핑

사용자는 Application Server에 정의된 개인 또는 응용 프로그램 아이디입니다. 사용자를 그룹과 연관시킬 수 있습니다. Application Server 인증 서비스는 여러 영역의 사용자를 관리할 수 있습니다.

그룹

J2EE **그룹**(또는 단순한 그룹)은 직위나 고객 프로필 같은 공통된 특성에 따라 분류된 사용자 범주입니다. 예를 들어, 전자 상거래 응용 프로그램의 사용자는 `customer` 그룹에 속할 수 있지만, 대량 소비자는 `preferred` 그룹에 속할 수 있습니다. 사용자를 그룹으로 범주화하면 많은 수의 사용자 액세스를 더 쉽게 제어할 수 있습니다.

역할

역할은 응용 프로그램, 각 응용 프로그램에서 사용자가 액세스할 수 있는 부분 및 사용자가 할 수 있는 작업을 정의합니다. 즉, 역할은 사용자의 인증 수준을 결정합니다.

예를 들어, 인사 프로그램에서 모든 직원이 전화 번호와 전자 메일 주소에 액세스할 수 있지만 급여 정보는 관리자만 액세스할 수 있습니다. 응용 프로그램은 다음과 같이 최소한 두 가지 역할을 정의할 수 있습니다. `employee` 및 `manager`. `manager` 역할의 사용자만 급여 정보를 볼 수 있습니다.

역할은 응용 프로그램의 기능을 정의하는 반면, 그룹은 연관된 사용자 집합이라는 점에서 역할과 사용자 그룹은 다릅니다. 예를 들어, 인사 응용 프로그램에는 `full-time`, `part-time` 및 `on-leave` 같은 그룹이 있을 수 있지만, 이 모든 그룹의 사용자는 여전히 `employee` 역할이 됩니다.

역할은 응용 프로그램 배포 설명자에 정의됩니다. 반대로, 그룹은 전체 서버와 영역에 대해 정의됩니다. 응용 프로그램 개발자나 배포자는 배포 설명자의 각 응용 프로그램에 대한 하나 이상의 그룹에 역할을 매핑합니다.

영역

보안 정책 도메인 또는 **보안 도메인**이라고도 하는 **영역**은 서버가 공통 보안 정책을 정의 및 실행하는 범위입니다. 실제적인 면에서, 영역은 서버가 사용자 및 그룹 정보를 저장하는 저장소입니다.

Application Server는 다음의 세 가지 영역으로 미리 구성됩니다. `file`(초기 기본 영역), `certificate` 및 `admin-realm`. `ldap`, `solaris` 또는 사용자 정의 영역을 설정할 수도 있습니다. 응용 프로그램은 배포 설명자에서 사용할 영역을 지정할 수 있습니다. 영역을 지정하지 않을 경우 Application Server는 기본 영역을 사용합니다.

`file` 영역의 경우 서버는 사용자 자격 증명을 `keyfile`이라고 하는 파일에 로컬로 저장합니다. 관리 콘솔을 사용하여 `file` 영역의 사용자를 관리할 수 있습니다..

`certificate` 영역의 경우 서버는 인증서 데이터베이스에 사용자 인증서를 저장합니다. `certificate` 영역을 사용할 경우 서버는 HTTPS 프로토콜과 함께 인증서를 사용하여 웹 클라이언트를 인증합니다. 인증서에 대한 자세한 내용은 95 페이지 “인증서 및 SSL 소개”를 참조하십시오.

admin-realm은 FileRealm이기도 하며 admin-keyfile이라는 파일에 관리자 자격 증명을 로컬로 저장합니다. file 영역에서 사용자를 관리하는 것과 같은 방법으로 관리 콘솔을 사용하여 이 영역의 사용자를 관리합니다.

ldap 영역의 경우 서버는 Sun Java System Directory Server 같은 LDAP(Lightweight Directory Access Protocol) 서버에서 사용자 자격 증명을 가져옵니다. LDAP는 공용 인터넷을 사용하는지 기업 인트라넷을 사용하는지 여부와 상관없이 조직, 개인 및 네트워크의 파일이나 장치 같은 다른 자원을 찾을 수 있게 해주는 프로토콜입니다. ldap 영역에서 사용자 및 그룹 관리에 대한 자세한 내용은 LDAP 서버 설명서를 참조하십시오.

solaris 영역의 경우 서버는 Solaris 운영 체제에서 사용자 자격 증명을 가져옵니다. 이 영역은 Solaris 9 OS 이상에서 지원됩니다. solaris 영역에서 사용자 및 그룹 관리에 대한 자세한 내용은 Solaris 설명서를 참조하십시오.

사용자 정의 영역은 관계형 데이터베이스나 타사 구성 요소 같은 사용자 자격 증명의 다른 저장소입니다. 자세한 내용은 관리 콘솔 온라인 도움말을 참조하십시오.

인증서 및 SSL 소개

이 절에서는 다음 항목에 대해 설명합니다.

디지털 인증서 정보

디지털 인증서(또는 인증서)는 인터넷의 사용자와 자원을 고유하게 식별하는 전자 파일입니다. 인증서는 두 엔티티 간에 안전하고 비밀을 유지하는 통신도 가능하게 해줍니다.

인증서의 종류는 여러 가지가 있는데 개인이 사용하는 개인 인증서와 SSL(Secure Sockets Layer) 기술을 통해 서버와 클라이언트 간에 안전한 세션을 설정하기 위해 사용하는 서버 인증서 등이 있습니다. SSL에 대한 자세한 내용은 97 페이지 “SSL(Secure Sockets Layer) 정보”를 참조하십시오.

인증서는 디지털 **키** 쌍(매우 긴 번호)을 사용하여 대상 수신자만 읽을 수 있도록 정보를 **암호화** 또는 인코딩하는 **공개 키 암호화**를 기반으로 합니다. 수신자는 정보의 비밀번호를 **해독**(디코드)하여 읽습니다.

키 쌍에는 공개 키와 개인 키가 포함됩니다. 소유자는 공개 키를 배포하여 모든 사용자가 사용할 수 있게 합니다. 그러나 소유자는 개인 키는 배포하지 않고 항상 비밀로 합니다. 키가 수학적으로 관련되어 있기 때문에 하나의 키로 암호화된 데이터는 키 쌍의 다른 키로만 해독할 수 있습니다.

인증서는 여권과 같습니다. 인증서는 소유자를 식별하고 기타 중요한 정보를 제공합니다. **인증 기관**(CA)이라고 하는 신뢰할 수 있는 제 3자가 인증서를 발행합니다. CA는 여권 담당 부서와 유사합니다. CA는 인증서가 위조되거나 조작될 수 없도록 소유자의 아이디를 검증하고 인증서에 서명합니다. CA가 인증서에 서명하면 소유자는 신원 증명으로 인증서를 제공하고 암호화된 기밀 통신을 설정할 수 있습니다.

가장 중요한 점은 인증서가 소유자의 공개 키를 소유자의 아이디에 바인드한다는 점입니다. 여권이 소유자에 대한 개인 정보와 사진을 함께 제공하는 것과 같이 인증서는 소유자에 대한 정보에 공개 키를 바인드합니다.

공개 키 외에 인증서에는 대개 다음과 같은 정보가 포함되어 있습니다.

- 소유자의 이름 및 인증서를 사용한 웹 서버의 URL 같은 기타 식별 정보 또는 개인의 전자 메일 주소
- 인증서를 발행한 인증 기관 이름
- 만료일

디지털 인증서는 X.509 형식의 기술 사양으로 관리됩니다. `certificate` 영역에서 사용자의 아이디를 확인하기 위해 인증 서비스는 X.509 인증서의 공통 이름 필드를 principal 이름으로 사용하여 X.509 인증서를 확인합니다.

인증서 체인 정보

웹 브라우저에는 브라우저가 자동으로 신뢰하는 **루트** CA 인증서 집합이 사전 구성되어 있습니다. 모든 인증서에는 자신의 유효성을 검증하기 위한 **인증서 체인**이 함께 제공되어야 합니다. 인증서 체인은 최종적으로 루트 CA 인증서에서 종료되는 연속적인 CA 인증서로 발행된 일련의 인증서입니다.

처음 생성된 인증서는 **자체 서명된** 인증서입니다. 자체 서명된 인증서는 발급자(서명자)가 주제(공개 키를 인증서에서 인증하는 엔티티)와 동일한 인증서입니다. 소유자가 인증서 서명 요청(CSR)을 인증 기관에 전송한 다음 응답을 가져오면 자체 서명된 인증서는 인증서 체인으로 대체됩니다. 체인 맨 아래에는 주제의 공개 키를 인증하는 인증 기관에서 발행한 인증서(응답)가 있습니다. 체인의 다음 인증서는 CA의 공개 키를 인증하는 인증서입니다. 대개는 자체 서명된 인증서, 즉 해당 공개 키를 인증하는 인증 기관의 인증서이고 체인의 마지막 인증서입니다.

다른 경우는 인증서 체인을 반환할 수 있습니다. 이 경우 인증서 체인의 맨 아래 인증서는 동일하지만(키 항목의 공개 키를 인증하는 CA에서 서명한 인증서), 체인의 두 번째 인증서는 CSR을 전송한 CA의 공개 키를 인증하는 다른 CA에서 서명한 인증서입니다. 체인의 다음 인증서는 두 번째 CA의 키를 인증하는 인증서입니다. 자체 서명된 **루트** 인증서에 도달할 때까지 이러한 방식으로 계속됩니다. 체인의 각 인증서(첫째 인증서 제외)는 체인의 이전 인증서의 서명자 공개 키를 인증합니다.

SSL(Secure Sockets Layer) 정보

SSL(Secure Sockets Layer)은 인터넷 통신 및 트랜잭션을 보안하기 위한 가장 일반적인 표준입니다. 웹 응용 프로그램은 서버와 클라이언트간 안전한 기밀 통신을 보장하기 위해 디지털 인증서를 사용하는 HTTPS(SSL 상의 HTTP)를 사용합니다. SSL 연결에서 클라이언트와 서버는 모두 데이터를 전송하기 전에 암호화한 다음 요청 시 해독합니다.

웹 브라우저(클라이언트)에서 보안 사이트에 연결할 경우 *SSL* **핸드셰이크**가 발생합니다.

- 브라우저는 대개 `http` 대신 `https`로 시작하는 URL을 요청하여 보안 세션을 요청하는 네트워크에서 메시지를 전송합니다.
- 서버는 공개 키를 포함하는 인증서를 전송하여 응답합니다.
- 브라우저는 서버의 인증서가 유효한지 그리고 인증서가 브라우저의 데이터베이스에 있고 신뢰할 수 있는 인증 기관에서 서명한 것인지 검증합니다. 그리고 CA 인증서가 만료되지 않았는지 확인합니다.
- 인증서가 유효하면 브라우저는 고유한 일회용 **세션 키**를 생성하고 이를 서버의 공개 키와 함께 암호화합니다. 브라우저는 암호화된 세션 키를 서버에 전송하여 둘 다 복사본을 갖고 있도록 합니다.
- 서버는 개인 키를 사용하여 메시지를 해독하고 세션 키를 복구합니다.

핸드셰이크 후, 클라이언트는 웹 사이트의 아이디를 검증했고 클라이언트와 웹 서버만 세션 키의 사본을 갖고 있습니다. 이 때부터 클라이언트와 서버는 세션 키를 사용하여 서로 간의 모든 통신을 암호화합니다. 따라서 이 통신은 보안이 안전합니다.

SSL 표준의 최신 버전을 TLS(Transport Layer Security)이라고 합니다. Application Server는 SSL(Secure Sockets Layer) 3.0 및 TLS(Transport Layer Security) 1.0 암호화 프로토콜을 지원합니다.

SSL을 사용하려면 Application Server에 외부 인터페이스에 대한 인증서나 보안 연결을 승인하는 IP 주소가 있어야 합니다. 디지털 인증서가 설치되지 않으면 대부분 웹 서버의 HTTPS 서비스가 실행되지 않습니다. 103 페이지 “`keytool` 유틸리티를 사용하여 인증서 생성”에 설명된 절차를 사용하여 웹 서버가 SSL에 대해 사용할 수 있는 디지털 인증서를 설정합니다.

암호화 정보

암호화는 암호화나 해독에 사용되는 암호화 알고리즘입니다. SSL 및 TLS 프로토콜은 서버와 클라이언트가 서로 간에 인증하고 인증서를 전송하며 세션 키를 설정하기 위해 사용한 다양한 암호화를 지원합니다.

일부 비밀번호는 다른 비밀번호보다 더 강력하고 더 안전합니다. 클라이언트와 서버는 서로 다른 암호화 제품군을 지원할 수 있습니다. SSL3 및 TLS 프로토콜에서 암호화를 선택합니다. 세션 연결 중에 클라이언트와 서버는 서로가 통신을 위해 설정한 더 강력한 암호화 사용을 승인하므로 대개 모든 암호화를 충분히 사용할 수 있습니다.

이름 기반의 가상 호스트 사용

보안 응용 프로그램에 대해 이름 기반의 가상 호스트를 사용하는 것은 문제가 될 수 있습니다. 이는 SSL 프로토콜 자체의 설계 한계입니다. 클라이언트 브라우저가 서버 인증서를 승인하는 SSL 핸드셰이크는 HTTP 요청에 액세스하기 전에 발생해야 합니다. 따라서 가상 호스트 이름을 포함하는 요청 정보를 인증 전에 확인할 수 없습니다. 그러므로 단일 IP 주소에 복수 인증서를 지정할 수 없습니다.

단일 IP 주소의 모든 가상 호스트를 동일한 인증서에 대해 인증해야 할 경우 복수 가상 호스트를 추가해도 서버의 정상 SSL 작업을 방해하지 않습니다. 그러나 대부분의 브라우저는 (공식적인 CA 서명된 인증서에 우선적으로 적용할 수 있는 경우) 서버의 도메인 이름을 인증서에 나열된 도메인 이름과 비교합니다. 도메인 이름이 일치하지 않을 경우 이 브라우저에서 경고를 표시합니다. 일반적으로 주소 기반 가상 호스트만 작업 환경에서 SSL과 같이 사용됩니다.

방화벽 정보

방화벽은 둘 이상의 네트워크간 데이터 흐름을 제어하고 네트워크간 링크를 관리합니다. 방화벽은 하드웨어 및 소프트웨어 요소 둘 다로 구성될 수 있습니다. 이 절에서는 일반 방화벽 구조와 방화벽 구성을 설명합니다. 여기에서는 주로 Application Server에 관련된 사항을 다룹니다. 특정 방화벽 기술에 대한 자세한 내용은 방화벽 공급업체의 설명서를 참조하십시오.

일반적으로 클라이언트가 필요한 TCP/IP 포트에 액세스할 수 있도록 방화벽을 구성합니다. 예를 들어 HTTP Listener가 포트 8080에서 작동할 경우 포트 8080에서만 HTTP 요청을 허용하도록 방화벽을 구성합니다. 마찬가지로, 포트 8181에 대한 HTTP 요청을 설정한 경우 포트 8181에서 HTTP 요청을 허용하도록 방화벽을 구성해야 합니다.

인터넷에서 EJB 모듈로 직접 RMI-IIOP(Remote Method Invocations over Internet Inter-ORB Protocol) 액세스가 필요한 경우 RMI-IIOP Listener 포트도 엽니다. 그러나 보안 위험이 생기기 때문에 열지 않는 것이 좋습니다.

이중 방화벽 구조의 경우 HTTP 및 HTTPS 트랜잭션을 허용하도록 외부 방화벽을 구성해야 합니다. 방화벽 뒤에서 Application Server와 통신하려면 HTTP 서버 플러그인을 허용하도록 내부 방화벽을 구성해야 합니다.

관리 콘솔을 사용하여 보안 관리

관리 콘솔은 보안의 다음 측면을 관리하기 위한 수단을 제공합니다.

- 99 페이지 “서버 보안 설정”
- 99 페이지 “영역 및 파일 영역 사용자”
- 99 페이지 “JACC 공급자”
- 99 페이지 “감사 모듈”

- 100 페이지 “메시지 보안”
- 100 페이지 “HTTP 및 IIOP Listener 보안”
- 100 페이지 “관리 서비스 보안”
- 100 페이지 “보안 맵”

서버 보안 설정

보안 설정 페이지에서 기본 영역, 익명 역할, principal 사용자 이름 및 비밀번호 지정을 포함한 전체 서버에 대한 등록 정보를 설정합니다.

영역 및 파일 영역 사용자

영역에 대한 개념은 93 페이지 “사용자, 그룹, 역할 및 영역 이해”에 설명되어 있습니다.

- 새로운 영역 만들기
- 기존 영역 삭제
- 기존 영역의 구성 수정
- `file` 영역의 사용자 추가, 수정 및 삭제
- 기본 영역 설정

JACC 공급자

JACC 공급자에 대한 내용은 92 페이지 “JACC 공급자 지정”에 설명되어 있습니다. 관리 콘솔을 사용하여 다음 작업을 수행할 수 있습니다.

- 새로운 JACC 공급자 추가
- 기존 JACC 공급자 삭제 또는 수정

감사 모듈

감사 모듈에 대한 내용은 92 페이지 “인증 및 권한 부여 결정 사항 감사”에 설명되어 있습니다. 감사는 오류나 보안 위반 같은 중요한 이벤트를 나중에 검토하기 위해 기록하는 방법입니다. 모든 인증 이벤트는 Application Server 로그에 기록됩니다. 전체 액세스 로그는 Application Server 액세스 이벤트의 순차적인 추적을 제공합니다.

관리 콘솔을 사용하여 다음 작업을 수행할 수 있습니다.

- 새로운 감사 모듈 추가
- 기존 감사 모듈 삭제 또는 수정

메시지 보안

메시지 보안에 대한 개념은 92 페이지 "메시지 보안 구성"에 설명되어 있습니다. 관리 콘솔을 사용하여 다음 작업을 수행할 수 있습니다.

- 메시지 보안 사용
- 메시지 보안 공급자 구성
- 기존 메시지 보안 구성 또는 공급자 삭제 또는 구성

이 작업에 대한 자세한 내용은 관리 콘솔 온라인 도움말을 참조하십시오.

HTTP 및 IIOP Listener 보안

HTTP 서비스의 가상 서버마다 하나 이상의 *HTTP Listener*를 통해 네트워크 연결을 제공합니다.

Application Server는 CORBA(Common Object Request Broker Architecture) 객체를 지원합니다. 이 객체는 IIOP(Internet Inter-Orb Protocol)를 사용하여 네트워크에서 통신합니다. *IIOP Listener*는 EJB의 원격 클라이언트와 다른 CORBA 기반 클라이언트에서 들어오는 연결을 받아들입니다. IIOP Listener에 대한 자세한 내용은 140 페이지 "IIOP Listener"를 참조하십시오.

관리 콘솔을 사용하여 다음 작업을 수행합니다.

- 새로운 HTTP 또는 IIOP Listener를 만들고 각각 사용하는 보안을 지정합니다.
- 기존 HTTP 또는 IIOP Listener에 대한 보안 설정을 수정합니다.

관리 서비스 보안

관리 서비스는 서버 인스턴스가 일반 인스턴스 또는 도메인 관리 서버(DAS)인지, 아니면 이 둘의 조합인지 지정합니다. 관리 서비스를 사용하여 원격 서버 인스턴스에 맞게 JSR-160 호환 원격 JMK 커넥터를 구성함으로써 호스트 시스템에서 서버 인스턴스를 관리합니다. 이 커넥터는 도메인 관리 서버와 노드 에이전트 간의 통신을 처리합니다.

관리 콘솔을 사용하여 다음 작업을 수행합니다.

- 관리 서비스 관리
- JMX 커넥터 편집
- JMX 커넥터에 대한 보안 설정 수정

보안 맵

관리 콘솔을 사용하여 다음 보안 매핑 작업을 수행합니다.

- 보안 맵을 기존 커넥터 연결 풀에 추가
- 기존 보안 맵 삭제 또는 구성

인증서 및 SSL 작업

인증서 파일 정보

Application Server를 설치하면 내부 검사에 적합한 JSSE(Java Secure Socket Extension) 또는 NSS(Network Security Services) 형식으로 디지털 인증서가 생성됩니다. 기본적으로 Application Server는 *domain-dir*/config 디렉토리의 인증서 데이터베이스에 인증서 정보를 저장합니다.

- **Keystore 파일**, key3.db에는 개인 키를 비롯하여 Application Server의 인증서가 포함되어 있습니다. Keystore 파일은 비밀번호로 보호됩니다. 비밀번호는 asadmin change-master-password 명령을 사용하여 변경합니다. certutil에 대한 자세한 내용은 106 페이지 “certutil 유틸리티 사용”을 참조하십시오.

  모든 keystore 항목에는 고유한 별칭이 있습니다. 설치 후 Application Server keystore는 별칭이 slas인 단일 항목을 가집니다.

- **Truststore 파일**, cert8.db에는 다른 엔티티에 대한 공개 키를 비롯하여 Application Server의 신뢰할 수 있는 인증서가 포함되어 있습니다. 신뢰할 수 있는 인증서의 경우 서버에서 인증서의 공개 키가 인증서 소유자에게 속하는지를 확인합니다. 일반적으로 신뢰할 수 있는 인증서에는 인증 기관의 인증서(CA)가 포함됩니다.

  Platform Edition의 경우 Application Server는 서버측에서 keytool을 사용하여 인증서와 keystore를 관리하는 JSSE 형식을 사용합니다. Enterprise Edition의 경우 Application Server는 서버측에서 certutil을 사용하여 개인 키와 인증서를 저장하는 NSS 데이터베이스를 관리하는 NSS를 사용합니다. 두 경우 모두 클라이언트측(응용 프로그램 클라이언트 또는 독립 실행형)에서는 JSSE 형식을 사용합니다.

  기본적으로 Application Server에는 예 응용 프로그램과 함께 작동하고 개발 용도로 사용되는 keystore 및 truststore가 구성되어 있습니다. 작업을 위해 인증서 별칭을 변경하거나 다른 인증서를 truststore에 추가하거나 keystore 및 truststore 파일의 이름 및/또는 위치를 변경할 수 있습니다.

인증서 파일 위치 변경

개발용으로 제공된 keystore 및 truststore 파일은 *domain-dir*/config 디렉토리에 저장됩니다.

관리 콘솔에서 server-config 노드 > JVM 설정 > JVM 옵션 탭을 확장하여 인증서 파일의 새 위치 값을 추가하거나 수정합니다.

```
-Dcom.sun.appserv.nss.db=${com.sun.aas.instanceRoot}/NSS-database-directory
```

여기서 *NSS-database-directory*는 NSS 데이터베이스의 위치입니다.

JSSE(Java Secure Socket Extension) 도구 사용

keytool을 사용하여 JSSE(Java Secure Socket Extension) 디지털 인증서를 설정하고 작업합니다. Platform Edition과 Enterprise Edition 모두의 경우 클라이언트측(응용 프로그램 클라이언트 또는 독립 실행형)에서는 JSSE 형식을 사용합니다.

J2SE SDK에는 keytool이 함께 제공되므로 관리자가 공개/개인 키 쌍 및 연관된 인증서를 관리할 수 있습니다. 사용자가 통신 피어의 공개 키를 인증서 양식으로 캐시할 수도 있습니다.

keytool을 실행하려면 J2SE /bin 디렉토리가 경로에 있도록 쉘 환경을 구성하거나 도구에 대한 전체 경로를 명령줄에서 제공해야 합니다. keytool에 대한 자세한 내용은 http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html에 있는 keytool 설명서를 참조하십시오.

keytool 유틸리티 사용

다음 예는 JSSE 도구를 사용한 인증서 처리와 관련된 사용법을 보여줍니다.

- 자체 서명된 인증서를 RSA 키 알고리즘을 사용하여 JKS 유형의 keystore로 만듭니다. RSA는 RSA Data Security, Inc.에서 개발한 공개 키 암호화 기술입니다. RSA는 기술 발명자인 Rivest, Shamir, Adelman의 약자입니다.

  ```
  keytool  -genkey -noprompt -trustcacerts -keyalg RSA -alias ${cert.alias}
  -dname  ${dn.name} -keypass ${key.pass} -keystore ${keystore.file}
  -storepass ${keystore.pass}
  ```

  인증서를 만드는 다른 예는 103 페이지 “keytool 유틸리티를 사용하여 인증서 생성”에 있습니다.
- 자체 서명된 인증서를 기본 키 알고리즘을 사용하여 JKS 유형의 keystore로 만듭니다.

  ```
  keytool -genkey -noprompt -trustcacerts -alias ${cert.alias} -dname
  ${dn.name} -keypass ${key.pass} -keystore ${keystore.file} -storepass
  ${keystore.pass}
  ```

  인증서에 서명하는 예는 104 페이지 “keytool 유틸리티를 사용하여 디지털 인증서 서명”에 있습니다.
- JKS 유형의 keystore에서 사용할 수 있는 인증서를 표시합니다.

  ```
  keytool -list -v  -keystore ${keystore.file} -storepass ${keystore.pass}
  ```
- JKS 유형의 keystore에서 인증서 정보를 표시합니다.

  ```
  keytool  -list -v  -alias ${cert.alias} -keystore ${keystore.file}
  -storepass ${keystore.pass}
  ```

- RFC/텍스트 형식 인증서를 JKS 저장소로 가져옵니다. 인증서는 대체로 바이너리 인코딩 대신 인터넷 RFC(Request for Comments) 1421 표준으로 정의한 인쇄 가능한 인코딩 형식으로 저장됩니다. *Base 64* **인코딩**이라고도 하는 이 인증서 형식은 인증서를 다른 응용 프로그램으로 전자 메일이나 몇 가지 다른 메커니즘을 통해 간편하게 내보낼 수 있게 해줍니다.

  ```
  keytool -import -noprompt -trustcacerts -alias ${cert.alias} -file
  ${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
  ```

- JKS 유형의 keystore에서 PKCS7 형식으로 인증서를 내보냅니다. Public Key Cryptography Standards(공개 키 암호화 표준) #7과 Cryptographic Message Syntax Standard(암호화 메시지 구문 표준)로 정의한 응답 형식에는 발행된 인증서 외에 지원 인증서 체인이 포함됩니다.

  ```
  keytool  -export -noprompt  -alias ${cert.alias} -file ${cert.file}
  -keystore ${keystore.file} -storepass ${keystore.pass}
  ```

- JKS 유형의 keystore에서 PKCS7 형식으로 인증서를 내보냅니다.

  ```
  keytool  -export -noprompt -rfc  -alias ${cert.alias} -file
  ${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
  ```

- JKS 유형의 keystore에서 인증서를 삭제합니다.

  ```
  keytool  -delete -noprompt -alias ${cert.alias}  -keystore ${keystore.file}
  -storepass ${keystore.pass}
  ```

  keystore에서 인증서를 삭제하는 다른 예는 105 페이지 “keytool 유틸리티를 사용하여 인증서를 삭제하는 방법”에 있습니다.

keytool 유틸리티를 사용하여 인증서 생성

keytool을 사용하여 인증서를 생성하고 가져오며 내보냅니다. 기본적으로 keytool은 실행될 디렉토리에 keystore 파일을 만듭니다.

1. 인증서가 실행될 디렉토리로 변경합니다.

   인증서는 반드시 keystore와 truststore 파일이 있는 디렉토리(기본은 *domain-dir*/config)에 생성하십시오. 이 파일의 위치를 변경하는 방법에 대한 자세한 내용은 101 페이지 “인증서 파일 위치 변경”을 참조하십시오.

2. 다음의 keytool 명령을 입력하여 keystore 파일 keystore.jks에 인증서를 생성합니다.

   ```
   keytool -genkey -alias keyAlias-keyalg RSA
    -keypass changeit
    -storepass changeit
   -keystore keystore.jks
   ```

   *keyAlias*와 같이 고유한 이름을 사용합니다. keystore 또는 개인 키 비밀번호의 기본값을 변경한 경우 위 명령의 changeit을 새 비밀번호로 대체합니다.

이름, 조직 및 keytool이 인증서를 생성하는 데 사용할 기타 정보를 묻는 메시지가 표시됩니다.

3. 다음의 keytool 명령을 입력하여 생성된 인증서를 server.cer(아니면 client.cer) 파일로 내보냅니다.

```
keytool -export -alias keyAlias-storepass changeit
 -file server.cer
 -keystore keystore.jks
```

4. 인증 기관에서 서명한 인증서가 필요한 경우에는 104 페이지 "keytool 유틸리티를 사용하여 디지털 인증서 서명"을 참조하십시오.
5. cacerts.jks truststore 파일을 만들고 인증서를 truststore에 추가하려면 다음의 keytool 명령을 입력합니다.

```
keytool -import -v -trustcacerts
-alias keyAlias
 -file server.cer
-keystore cacerts.jks
 -keypass changeit
```

6. keystore 또는 개인 키 비밀번호의 기본값을 변경한 경우 위 명령의 changeit을 새 비밀번호로 대체합니다.

   도구는 인증서에 대한 정보를 표시하고 해당 인증서에 대한 신뢰 여부를 묻습니다.
7. yes를 입력하고 Enter 키를 누릅니다.

   그러면 keytool은 다음과 같은 메시지를 표시합니다.

```
Certificate was added to keystore
[Saving cacerts.jks]
```

8. Application Server를 다시 시작합니다.

keytool 유틸리티를 사용하여 디지털 인증서 서명

디지털 인증서를 만든 후 소유자는 위조를 막기 위해 인증서에 서명해야 합니다. 전자 상거래 사이트나 아이디 인증이 중요한 사이트는 잘 알려진 인증 기관(CA)에서 인증서를 구매할 수 있습니다. 인증이 중요하지 않은 경우, 예를 들어 개인적인 보안 통신만 필요한 경우에는 인증서를 얻는데 필요한 시간과 경비를 절약하고 자체 서명된 인증서를 사용합니다.

1. CA의 웹 사이트에서 지침을 수행하여 인증서 키 쌍을 생성합니다.
2. 생성된 인증서 키 쌍을 다운로드합니다.

   인증서는 반드시 keystore와 truststore 파일이 있는 디렉토리(기본은 *domain-dir*/config 디렉토리)에 저장하십시오. 101 페이지 "인증서 파일 위치 변경"을 참조하십시오.
3. 쉘에서 인증서가 있는 디렉토리로 변경합니다.

4. keytool을 사용하여 인증서를 로컬 keystore로 가져오고, 필요한 경우 로컬 truststore로도 가져옵니다.

   ```
   keytool -import -v -trustcacerts
   -alias keyAlias
    -file server.cer
   -keystore cacerts.jks
    -keypass changeit
   -storepass changeit
   ```

   keystore 또는 개인 키 비밀번호의 기본값을 변경한 경우 위 명령의 changeit을 새 비밀번호로 대체합니다.

5. Application Server를 다시 시작합니다.

keytool 유틸리티를 사용하여 인증서를 삭제하는 방법

기존의 인증서를 삭제하려면 다음 예와 같이 keytool -delete 명령을 사용합니다.

```
keytool -delete
 -alias keyAlias
 -keystore keystore-name
 -storepass password
```

NSS(Network Security Services) 도구 사용

Enterprise Edition의 경우 서버측에서 NSS(Network Security Services) 디지털 인증서를 사용하여 개인 키와 인증서를 저장하는 데이터베이스를 관리합니다.
클라이언트측(응용 프로그램 클라이언트 또는 독립 실행형)의 경우 102 페이지 “JSSE(Java Secure Socket Extension) 도구 사용”에서 설명한 대로 JSSE 형식을 사용합니다.

NSS(Network Security Services)를 사용하여 보안을 관리하는 도구는 다음 사항을 포함합니다.

- certutil은 인증서 및 키 데이터베이스를 관리하기 위한 명령줄 유틸리티입니다. certutil 유틸리티를 사용한 몇 가지 예는 106 페이지 “certutil 유틸리티 사용”에 있습니다.
- pk12util은 인증서/키 데이터베이스와 PKCS12 형식의 파일 간에 키와 인증서를 내보내고 가져오기 위해 사용하는 명령줄 유틸리티입니다. pk12util 유틸리티를 사용한 몇 가지 예는 107 페이지 “pk12util 유틸리티를 사용하여 인증서 가져오기 및 내보내기”에 있습니다.
- modutil은 secmod.db 파일 또는 하드웨어 토큰 내에서 PKCS #11 모듈 정보를 관리하기 위해 사용하는 명령줄 유틸리티입니다. modutil 유틸리티를 사용한 몇 가지 예는 108 페이지 “modutil을 사용하여 PKCS11 모듈 추가 및 삭제”에 있습니다.

도구는 *install-dir*/lib/ 디렉토리에 있습니다. 다음 환경 변수는 NSS 보안 도구의 위치를 나타내는 데 사용됩니다.

- LD_LIBRARY_PATH =${*install-dir*}/lib
- ${*os.nss.path*}

예에서 인증서 공통 이름(CN)은 클라이언트나 서버의 이름입니다. CN은 SSL 핸드셰이크 중에 인증서 이름과 인증서를 만든 호스트 이름을 비교하는 데도 사용됩니다. 인증서 이름과 호스트 이름이 일치하지 않으면 SSL 핸드셰이크 중에 경고나 예외가 생성됩니다. 몇 가지 예에서 편의상 인증서 공통 이름 CN=localhost가 사용되는데 모든 사용자는 실제 호스트 이름을 사용하여 새 인증서를 만드는 대신 이 인증서를 사용할 수 있습니다.

다음 예는 NSS 도구를 사용한 인증서 처리와 관련된 사용법을 보여줍니다.

- 106 페이지 “certutil 유틸리티 사용”
- 107 페이지 “pk12util 유틸리티를 사용하여 인증서 가져오기 및 내보내기”
- 108 페이지 “modutil을 사용하여 PKCS11 모듈 추가 및 삭제”

certutil 유틸리티 사용

certutil을 실행하기 전에 LD_LIBRARY_PATH가 이 유틸리티를 실행하는 데 필요한 라이브러리의 위치를 가리키도록 합니다. 이 위치는 asenv.conf(제품 차원의 구성 파일)의 AS_NSS_LIB 값에서 식별할 수 있습니다.

인증서 데이터베이스 도구인 certutil은 Netscape Communicator cert8.db 및 key3.db 데이터베이스 파일을 작성 및 수정할 수 있는 명령줄 유틸리티입니다. 또한 cert8.db 파일 내에 인증서를 나열, 생성, 수정 또는 삭제하거나, 비밀번호를 작성 또는 변경하거나 새로운 공개 및 개인 키 쌍을 생성하거나, 키 데이터베이스 내용을 표시하거나 key3.db 파일 내의 키 쌍을 삭제할 수도 있습니다.

키와 인증서 관리 프로세스는 대개 키 데이터베이스에 키를 만든 다음 인증서 데이터베이스에 인증서를 생성 및 관리하는 것으로 시작됩니다. 다음의 설명서에서는 certutil 유틸리티의 구문을 포함하여 NSS를 사용한 인증서 및 키 데이터베이스 관리에 대해 설명합니다.

http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html

아래 목록의 각 항목은 NSS 및 JSSE 보안 도구를 사용하여 인증서를 만들거나 관리하는 예를 보여줍니다.

- 자체 서명된 서버와 클라이언트 인증서를 생성합니다. 이 예에서 CN 형식은 hostname.domain.[com|org|net|...]입니다.

  이 예의 경우 *domain-dir*/config입니다. serverseed.txt와 clientseed.txt 파일은 임의의 텍스트를 포함할 수 있습니다. 임의의 텍스트는 키 쌍을 생성하는 데 사용됩니다.

```
certutil -S -n $SERVER_CERT_NAME -x -t "u,u,u"
 -s "CN=$HOSTNAME.$HOSTDOMAIN, OU=Java Software, O=Sun Microsystems Inc.,
    L=Santa Clara, ST=CA, C=US"
 -m 25001 -o  $CERT_DB_DIR/Server.crt
 -d $CERT_DB_DIR -f passfile &lt;$CERT_UTIL_DIR/serverseed.txt
```

클라이언트 인증서를 생성합니다. 이 인증서도 자체 서명된 인증서입니다.

```
certutil -S -n $CLIENT_CERT_NAME  -x -t "u,u,u"
 -s "CN=MyClient, OU=Java Software, O=Sun Microsystems Inc.,
    L=Santa Clara, ST=CA, C=US"
 -m 25002 -o  $CERT_DB_DIR/Client.crt
 -d $CERT_DB_DIR -f passfile &lt;$CERT_UTIL_DIR/clientseed.txt
```

- 이전의 글머리표 기호에서 인증서가 생성되었는지 확인합니다.

```
certutil -V -u V -n $SERVER_CERT_NAME -d  $CERT_DB_DIR
certutil -V -u C -n $CLIENT_CERT_NAME -d  $CERT_DB_DIR
```

- 사용할 수 있는 인증서를 표시합니다.

```
certutil -L -d  $CERT_DB_DIR
```

- RFC 텍스트 형식 인증서를 NSS 인증서 데이터베이스로 가져옵니다.

```
certutil -A -a -n ${cert.nickname} -t ${cert.trust.options}
-f ${pass.file} -i ${cert.rfc.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- 인증서를 NSS 인증서 데이터베이스에서 RFC 형식으로 내보냅니다.

```
certutil -L -a -n ${cert.nickname} -f ${pass.file}
-d ${admin.domain.dir}/${admin.domain}/config > cert.rfc
```

- 인증서를 NSS 인증서 데이터베이스에서 삭제합니다.

```
certutil -D -n ${cert.nickname} -f ${pass.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- 인증서를 NSS 데이터베이스에서 JKS 형식으로 이동합니다.

```
certutil -L -a -n ${cert.nickname}
-d ${admin.domain.dir}/${admin.domain}/config > cert.rfc
keytool  -import -noprompt -trustcacerts -keystore ${keystore.file}
-storepass ${keystore.pass} -alias ${cert.alias} -file cert.rfc
```

pk12util 유틸리티를 사용하여 인증서 가져오기 및 내보내기

pk12util은 PKCS12 형식의 인증서/키 데이터베이스 및 파일 간에 키와 인증서를 가져오고 내보내는 데 사용하는 명령줄 유틸리티입니다. PKCS12는 Personal Information Exchange Syntax Standard(개인 정보 상호 교환 구문 표준)의 PKCS(Public-Key Cryptography Standards) #12입니다. pk12util 유틸리티에 대한 자세한 내용은 http://www.mozilla.org/projects/security/pki/nss/tools/pk12util.html을 참조하십시오.

- PKCS12 형식의 인증서를 NSS 인증 데이터베이스로 가져옵니다.

```
pk12util -i ${cert.pkcs12.file} -k ${certdb.pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- PKCS12 형식의 인증서를 NSS 인증 데이터베이스 토큰 모듈로 가져옵니다.

```
pk12util -i ${cert.pkcs12.file} -h ${token.name} -k ${certdb.pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- 인증서를 NSS 인증서 데이터베이스에서 PKCS12 형식으로 내보냅니다.

```
pk12util -o -n ${cert.nickname} -k ${pass.file} -w${cert.pass.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- 인증서를 NSS 인증서 데이터베이스 토큰 모듈에서 PKCS12 형식(하드웨어 가속기 구성에 유용)으로 내보냅니다.

```
pk12util -o -n ${cert.nickname} -h ${token.name} -k ${pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- PKCS12 인증서를 Java 소스를 필요로 하는 JKS 형식으로 변환합니다.

```
&lt;target name="convert-pkcs12-to-jks" depends="init-common">
   &lt;delete file="${jks.file}" failonerror="false"/>
   &lt;java classname="com.sun.enterprise.security.KeyTool">
      &lt;arg line="-pkcs12"/>
      &lt;arg line="-pkcsFile ${pkcs12.file}"/>
      &lt;arg line="-pkcsKeyStorePass ${pkcs12.pass}"/>
      &lt;arg line="-pkcsKeyPass ${pkcs12.pass}"/>
      &lt;arg line="-jksFile ${jks.file}"/>
      &lt;arg line="-jksKeyStorePass ${jks.pass}"/>
      &lt;classpath>
         &lt;pathelement path="${s1as.classpath}"/>
         &lt;pathelement path="${env.JAVA_HOME}/jre/lib/jsse.jar"/>
      &lt;/classpath>
   &lt;/java>
&lt;/target>
```

modutil을 사용하여 PKCS11 모듈 추가 및 삭제

보안 모듈 데이터베이스 도구인 `modutil`은 `secmod.db` 파일이나 하드웨어 토큰 내에서 PKCS #11(Cryptographic Token Interface Standard, 암호화 토큰 인터페이스 표준) 모듈 정보를 관리하는 명령줄 유틸리티입니다. 이 도구를 사용하여 PKCS #11 모듈을 추가 및 삭제하고, 비밀번호를 변경하며 기본값을 설정하고, 모듈 내용을 나열하고 슬롯을 활성화하거나 비활성화하며, FIPS-140-1 호환을 활성화하거나 비활성화하고 기본 공급자를 암호화 작업에 할당할 수 있습니다. 이 도구는 `key3.db`, `cert7.db` 및 `secmod.db` 보안 데이터베이스 파일도 만들 수 있습니다. 이 도구에 대한 자세한 내용은 `http://www.mozilla.org/projects/security/pki/nss/tools/modutil.html`을 참조하십시오.

- 새 PKCS11 모듈이나 토큰을 추가합니다.

```
modutil -add ${token.module.name} -nocertdb -force -mechanisms RSA:DSA:RC4:DES
-libfile ${SCA.lib.path} -dbdir ${admin.domain.dir}/${admin.domain}/config
```

- NSS 저장소에서 PKCS11 모듈을 삭제합니다.

```
modutil -delete ${token.module.name} -nocertdb -force -mechanisms RSA:DSA:RC4:DES
-libfile ${SCA.lib.path} -dbdir ${admin.domain.dir}/${admin.domain}/config
```

- NSS 저장소의 사용 가능한 토큰 모듈을 나열합니다.

```
modutil -list  -dbdir ${admin.domain.dir}/${admin.domain}/config
```

Application Server에 하드웨어 암호화 가속기 사용

하드웨어 가속기 토큰을 사용하면 암호화 성능을 향상시킬 수 있으며 보안 키 저장소 기능을 제공할 수 있습니다. 또한, 스마트 카드를 통해 최종 사용자에게 이동식 보안 키 저장소를 제공할 수도 있습니다.

Sun Java System Application Server 8.1 및 8.2 Standard Edition 또는 Enterprise Edition은 Java 2 Platform, Standard Edition(J2SE 플랫폼) 5.0에서 실행되는 경우, SSL 또는 TLS 통신에 대한 PKCS#11 토큰 사용 및 키와 PKCS#11 토큰을 관리하는 데 필요한 NSS(Network Security Services) 도구를 지원합니다. 이 절에서는 Application Server가 해당 지원을 제공하는 방법에 대해 설명하며 관련 구성에 대한 절차를 안내합니다.

J2SE 5.0 PKCS#11 공급자는 Application Server 런타임과 쉽게 통합할 수 있습니다. 이 공급자를 통해 Application Server에서 하드웨어 가속기 및 다른 PKCS#11 토큰을 사용함으로써 빠른 성능을 얻을 수 있으며 SSL 또는 TLS 통신의 고유 개인 키를 보호할 수 있습니다.

이 절은 다음 내용으로 구성되어 있습니다.

하드웨어 암호화 가속기 구성 정보

Sun Java System Application Server 8.1 및 8.2 Standard Edition 또는 Enterprise Edition은 Sun Crypto Accelerator 1000(SCA-1000) 및 SCA-4000 테스트를 거쳤습니다.

Application Server는 J2SE 5.0과 함께 사용하는 경우 PKCS#11 토큰과 통신할 수 있습니다. NSS PKCS#11 토큰 라이브러리(NSS 내부 PKCS#11 모듈의 경우 일반적으로 NSS 소프트 토큰으로 알려져 있음) 및 NSS 명령줄 관리 도구는 Application Server에 패키지화되어 있습니다. 자세한 내용은 105 페이지 “NSS(Network Security Services) 도구 사용”을 참조하십시오.

런타임 시 토큰 키와 인증서에 액세스할 수 있도록 NSS 도구를 사용하여 PKCS#11 토큰 및 J2SE PKCS#11 공급자에 키와 인증서를 만듭니다. PKCS#11 공급자는 원시 PKCS#11 라이브러리에 대한 래퍼 역할을 하는 암호화 서비스 공급자입니다. PKCS#11 토큰은 일반적으로 원시 PKCS#11 인터페이스를 가진 모든 하드웨어 및 소프트웨어 토큰을 나타냅니다. 하드웨어 토큰은 하드웨어 가속기 및 스마트 카드와 같은 물리적 장치에서 구현되는 PKCS#11 토큰입니다. 소프트웨어 토큰은 전적으로 소프트웨어에서 구현되는 PKCS#11 토큰입니다.

주 – J2SE 1.4.x 플랫폼에서 Application Server를 실행하면 하나의 PKCS#11 토큰(NSS 소프트 토큰)만 지원됩니다.

Microsoft Windows 환경의 경우, NSS 라이브러리 *AS_NSS*의 위치와 NSS 도구 디렉토리 *AS_NSS_BIN*의 위치를 PATH 환경 변수에 추가합니다. 작업상 편의를 위해 이 절의 절차에서는 UNIX 명령만 사용합니다. 적절한 위치에서 UNIX 변수를 Windows 변수로 바꿔야 합니다.

하드웨어 암호화 가속기를 구성하려면 다음의 두 가지 주요 절차를 수행합니다.

- 110 페이지 “PKCS#11 토큰 구성”
- 113 페이지 “J2SE 5.0 PKCS#11 공급자 구성”

PKCS#11 토큰 구성

이 절에서는 NSS 보안 도구 `modutil`을 사용하여 PKCS#11 토큰을 구성하는 방법에 대해 설명합니다. 다음 절차를 사용하여 PKCS#11 토큰을 구성합니다.

다음 명령을 모두 한 줄로 입력합니다.

```
modutil -dbdir AS_NSS_DB -nocertdb -force -add moduleName -libfile
 absolute_path_of_pkcs11_library -mechanisms list_of_security_mechanisms
```

여기서, *AS_NSS_DB*는 NSS 데이터베이스 디렉토리입니다(DAS(Domain Administration Server)를 사용하는 경우 `AS_DOMAIN_CONFIG`와 동일함).

예를 들어, 하드웨어 가속기 토큰을 구성하려면 다음을 모두 한 줄로 입력합니다.

```
modutil -dbdir AS_NSS_DB -nocertdb -force -add "Sun Crypto Accelerator" -libfile
 /opt/SUNWconn/crypto/lib/libpkcs11.so -mechanisms RSA:DSA:RC4:DES
```

이 예에서 하드웨어 가속기는 SCA–1000 암호화 가속기입니다. 해당 PKCS#11 라이브러리는 기본적으로 `/opt/SUNWconn/crypto/lib/libpkcs11.so`에 있습니다.

`mechanisms`는 해당 토큰에 사용할 수 있는 암호화 메커니즘에 대한 전체 목록이어야 합니다. 몇 개의 암호화 메커니즘만 사용하려면 113 페이지 “J2SE 5.0 PKCS#11 공급자 구성”을 참조하십시오. 지원되는 모든 메커니즘 목록을 보려면 NSS 보안 도구 사이트(`http://www.mozilla.org/projects/security/pki/nss/tools`)의 `modutil` 설명서를 참조하십시오.

다음 예에서는 토큰 설치 시 지정한 토큰 이름이 mytoken이라고 가정합니다.

하드웨어 가속기가 제대로 구성되었는지 확인하려면 다음 명령을 입력합니다.

```
modutil -list -dbdir AS_NSS_DB
```

표준 출력은 다음과 유사합니다.

```
Using database directory /var/opt/SUNWappserver/domains/domain1/config ...

Listing of PKCS#11 Modules
-----------------------------------------------------------
  1. NSS Internal PKCS#11 Module
         slots: 2 slots attached
        status: loaded

         slot: NSS Internal Cryptographic Services
        token: NSS Generic Crypto Services

         slot: NSS User Private Key and Certificate Services
        token: NSS Certificate DB

  2. Sun Crypto Accelerator
        library name: /opt/SUNWconn/crypto/lib/libpkcs11.so
         slots: 1 slot attached
        status: loaded

         slot: Sun Crypto Accelerator:mytoken
        token: mytoken
-----------------------------------------------------------
```

키 및 인증서 관리

이 절에서는 certutil 및 pk12util을 사용하여 키 및 인증서를 만들고 관리하는 몇 가지 일반 절차에 대해 설명합니다. certutil 및 pk12util에 대한 자세한 내용은 105 페이지 “NSS(Network Security Services) 도구 사용” 및 NSS 보안 도구 사이트(http://www.mozilla.org/projects/security/pki/nss/tools)의 설명서를 참조하십시오.

주 – 또한 java.security 등록 정보 파일(Java 런타임의 *JAVA_HOME*/jre/lib/security 디렉토리에 있음)에 PKCS#11 공급자를 구성하면 J2SE keytool 유틸리티를 사용하여 키와 인증서를 관리할 수 있습니다. keytool 사용에 대한 자세한 내용은 http://java.sun.com/j2se/1.5.0/docs/guide/secuirty/p11guide.html의 Java PKCS#11 Reference Guide를 참조하십시오.

이 절은 다음 내용으로 구성되어 있습니다.

키 및 인증서 나열

- 구성된 PKCS#11 토큰의 키와 인증서를 나열하려면 다음 명령을 실행합니다.

  ```
  certutil -L -d AS_NSS_DB [-h tokenname]
  ```

  예를 들어, 기본 NSS 소프트 토큰의 내용을 나열하려면 다음을 입력합니다.

  ```
  certutil -L -d AS_NSS_DB
  ```

  표준 출력은 다음과 유사합니다.

  ```
  verisignc1g1                 T,c,c
  verisignc1g2                 T,c,c
  verisignc1g3                 T,c,c
  verisignc2g3                 T,c,c
  verisignsecureserver         T,c,c
  verisignc2g1                 T,c,c
  verisignc2g2                 T,c,c
  verisignc3g1                 T,c,c
  verisignc3g2                 T,c,c
  verisignc3g3                 T,c,c
  s1as                         u,u,u
  ```

  출력의 왼쪽 열에는 토큰의 이름이 표시되고 오른쪽 열에는 세 가지 트러스트 속성 집합이 표시됩니다. Application Server 인증서의 경우 대부분 `T,c,c`로 출력됩니다. 한 가지 수준의 트러스트만 포함된 J2SE `java.security.KeyStore` API와는 달리, NSS 기술에는 여러 수준의 트러스트가 포함되어 있습니다. Application Server는 기본적으로 이 토큰이 SSL을 사용하는 방법을 설명하는 첫 번째 트러스트 속성을 사용합니다. 이 속성에 대한 설명은 다음과 같습니다.

  `T`는 클라이언트 인증서 발급을 위한 인증 기관(CA)이 트러스트되었음을 나타냅니다.
  `u`는 인증서(및 키)를 인증 또는 서명에 사용할 수 있음을 나타냅니다.
  `u,u,u` 속성 조합은 개인 키가 데이터베이스에 있음을 나타냅니다.

- 하드웨어 토큰 `mytoken`의 내용을 나열하려면 다음 명령을 실행합니다.

  ```
  certutil -L -d AS_NSS_DB -h mytoken
  ```

  하드웨어 토큰의 비밀번호를 입력하라는 메시지가 표시됩니다. 표준 출력은 다음과 유사합니다.

```
Enter Password or Pin for "mytoken":
mytoken:Server-Cert                                          &#9;u,u,u
```

개인 키 및 인증서 작업

certutil을 사용하여 자체 서명된 인증서를 만들고 인증서를 가져오거나 내보냅니다. 개인 키를 가져오거나 내보내려면 pk12util 유틸리티를 사용합니다. 자세한 내용은 105 페이지 “NSS(Network Security Services) 도구 사용”을 참조하십시오.

주의 – Application Server에서 NSS 도구 certutil 및 modutil을 사용하여 NSS 비밀번호를 직접 수정하지 마십시오. NSS 비밀번호를 직접 수정하면 Application Server의 보안 데이터가 손상될 수 있습니다.

J2SE 5.0 PKCS#11 공급자 구성

Application Server는 J2SE PKCS#11 공급자를 사용하여 런타임 시 PKCS#11 토큰에 있는 키 및 인증서에 액세스합니다. 기본적으로 Application Server는 NSS 소프트 토큰에 J2SE PKCS#11 공급자를 구성합니다. 이 절에서는 J2SE PKCS#11 공급자의 기본 구성을 대체하는 방법에 대해 설명합니다.

Application Server에서 각 PKCS#11 토큰에 다음과 같은 기본 PKCS#11 구성 매개 변수가 생성됩니다.

- 기본 NSS 소프트 토큰에 대한 구성:

```
name=internal
library=${com.sun.enterprise.nss.softokenLib}
nssArgs="configdir='${com.sun.appserv.nss.db}'
 certPrefix='' keyPrefix='' secmod='secmod.db'"
slot=2
omitInitialize = true
```

- SCA 1000 하드웨어 가속기에 대한 구성:

```
name=HW1000
library=/opt/SUNWconn/crypto/lib/libpkcs11.so
slotListIndex=0
omitInitialize=true
```

이 구성은 Java PKCS#11 Reference Guide에 설명된 구문을 준수합니다.

주 – 이름 매개 변수는 고유해야 하며, 이외의 다른 요구 사항은 없습니다. J2SE 5.0과 같은 이전의 특정 버전은 영숫자 문자만 지원합니다.

사용자 정의 구성 파일을 만들어 기본 구성 매개 변수를 대체할 수 있습니다. 예를 들어, SCA–1000의 RSA 암호화 및 RSA 키 쌍 생성기를 명시적으로 비활성화할 수 있습니다. RSA 암호화 및 RSA 키 쌍 생성기를 비활성화하는 방법에 대한 자세한 내용은 http://www.mozilla.org/projects/security/pki/nss/tools를 참조하십시오.

사용자 정의 구성 파일을 만들려면 다음을 수행합니다.

1. 다음 코드를 사용하여 *install-dir*/mypkcs11.cfg라는 구성 파일을 만들고 저장합니다.

```
name=HW1000
library=/opt/SUNWconn/crypto/lib/libpkcs11.so
slotListIndex=0
disabledMechanisms = {
&#9;CKM_RSA_PKCS
&#9;CKM_RSA_PKCS_KEY_PAIR_GEN
}
omitInitialize=true
```

2. 필요한 경우 NSS 데이터베이스를 업데이트합니다. 여기서는 RSA를 비활성화하기 위해 NSS 데이터베이스를 업데이트합니다.

   다음 명령을 실행합니다.

```
modutil -undefault "Sun Crypto Accelerator" -dbdir AS_NSS_DB -mechanisms RSA
```

   mechanisms 목록의 알고리즘 이름은 기본 구성의 알고리즘 이름과 다릅니다. NSS의 유효한 mechanisms 목록을 보려면 NSS 보안 도구 사이트(http://www.mozilla.org/projects/security/pki/nss/tools)의 modutil 설명서를 참조하십시오.

3. 다음과 같이 등록 정보를 해당 위치에 추가하여 서버에 이 변경 사항을 업데이트합니다.

```
&lt;property name="mytoken" value="&InstallDir;/mypkcs11.cfg"/>
```

   등록 정보의 위치는 다음 중 하나일 수 있습니다.

   - 공급자가 DAS 또는 서버 인스턴스용인 경우, 연관된 <security-service> 아래에 등록 정보를 추가합니다.
   - 공급자가 노드 에이전트용인 경우, domain.xml 파일에서 연관된 <node-agent> 요소 아래에 등록 정보를 추가합니다.

4. Application Server를 다시 시작합니다.

   사용자 정의한 구성을 적용하려면 서버를 다시 시작합니다.

추가 정보

- Java 2 Standard Edition의 보안 설명은 http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html에서 볼 수 있습니다.
- *J2EE 1.4 Tutorial*의 *Security* 장은 http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html에서 볼 수 있습니다.
- **관리 설명서**의 10 장
- **Developer's Guide**의 *Securing Applications* 장

◆ ◆ ◆ 10장

10 메시지 보안 구성

이 장의 일부 내용은 보안 및 웹 서비스 개념에 대한 기본적인 이해가 필요합니다. 이 장을 시작하기 전에 이 개념에 대한 자세한 내용을 보려면 115 페이지 “추가 정보”에 나열된 자원을 살펴봅니다.

이 장에서는 Application Server의 웹 서비스에 대한 메시지 계층 보안 구성에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

메시지 보안 개요

메시지 보안의 경우 보안 정보가 메시지에 삽입되므로 네트워킹 계층을 지나 메시지와 함께 메시지 대상에 도달합니다. 메시지 보안은 메시지 보안을 사용하여 메시지 전송에서 메시지 보호를 분리하여 전송 후에도 메시지가 보호되도록 하는 전송 계층 보안(**J2EE 1.4 Tutorial**의 **Security** 장에서 설명)과 다릅니다.

웹 서비스 보안: SOAP 메시지 보안(WS-Security)은 Sun Microsystems를 포함한 모든 웹 서비스 기술 주요 제공업체가 OASIS로 공동 개발한 상호 운영 가능한 웹 서비스의 국제적인 표준입니다. WS-Security는 XML 암호화 및 XML 디지털 서명을 사용하여 SOAP에서 전송된 웹 서비스 메시지를 보안 처리하는 메시지 보안 체계입니다. WS-Security 사양에서는 SOAP 웹 서비스 메시지를 암호화하기 위한 X.509 인증서, SAML 명제 및 사용자 이름/비밀번호 토큰 등을 포함한 다양한 보안 토큰 사용을 정의합니다.

WS-Security 사양은 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf에서 볼 수 있습니다.

Application Server의 메시지 보안 이해

Application Server는 서버의 웹 서비스 클라이언트와 서버측 컨테이너의 WS-Security 표준에 대해 통합 지원을 제공합니다. 이 기능은 Application Server의 컨테이너가 응용 프로그램을 대신하여 웹 서비스 보안을 실행하고 응용 프로그램의 구현을 변경하지 않고 웹 서비스 응용 프로그램을 보호하는 데 적용할 수 있도록 통합되어 있습니다. Application Server에서는 SOAP 계층 메시지 보안 공급자와 메시지 보호 정책을 컨테이너와 컨테이너에 배포된 응용 프로그램에 바인드하는 기능을 제공하여 이러한 효과를 달성합니다.

메시지 보안 책임 지정

Application Server에서는 118 페이지 "시스템 관리자"와 119 페이지 "응용 프로그램 배포자" 역할에 메시지 보안을 구성하는 기본 책임이 있다고 예상합니다. 일반적인 경우에는 다른 역할에서 개발자의 관여 없이 기존 응용 프로그램의 구현을 변경하지 않고도 기존 응용 프로그램의 보안을 확보할 수 있지만 119 페이지 "응용 프로그램 개발자"가 기여하는 경우도 있습니다. 다음 절에는 다양한 역할의 책임이 정의되어 있습니다.

시스템 관리자

시스템 관리자의 책임은 다음과 같습니다.

- Application Server에서 메시지 보안 공급자 구성
- 사용자 데이터베이스 관리
- keystore 및 truststore 파일 관리
- 암호화를 사용하고 Java SDK 1.5.0 이전 버전을 실행할 경우 Java Cryptography Extension(JCE) 공급자 구성
- 샘플 서버 설치. `xms` 샘플 응용 프로그램을 사용하여 메시지 계층 웹 서비스 보안 사용을 보여줄 경우에만 수행합니다.

시스템 관리자는 관리 콘솔을 사용하여 서버 보안 설정을 관리하고 명령줄 도구를 사용하여 인증서 데이터베이스를 관리합니다. Platform Edition의 경우 인증서 및 개인 키는 키 저장소에 저장되고 `keytool`을 사용하여 관리합니다. Standard Edition 및 Enterprise Edition은 인증서와 개인 키를 NSS 데이터베이스에 저장하고 `certutil`을 사용하여 관리합니다. 이 문서는 기본적으로 시스템 관리자를 대상으로 합니다. 메시지 보안 작업에 대한 개요는 123 페이지 "메시지 보안을 위한 Application Server 구성 "을 참조하십시오.

응용 프로그램 배포자

응용 프로그램 배포자의 책임은 다음과 같습니다.

- 상위 스트림 역할 개발자나 어셈블러에서 응용 프로그램 관련 메시지 보호 정책을 지정하지 않은 경우 응용 프로그램 어셈블리에서 이러한 필수 정책 지정
- 웹 서비스 종점 및 서비스 참조에 대한 응용 프로그램 관련 메시지 보호 정책 정보(message-security-binding 요소)를 지정하기 위해 Sun 특정 배포 설명자 수정

이 보안 작업은 **Developers' Guide**의 **Securing Applications** 장에 설명되어 있습니다. 이 장에 대한 링크는 115 페이지 "추가 정보"를 참조하십시오.

응용 프로그램 개발자

응용 프로그램 개발자는 메시지 보안을 설정할 수 있지만 그럴 책임은 없습니다. 모든 웹 서비스가 보안 처리되도록 시스템 관리자가 메시지 보안을 설정하거나 응용 프로그램에 바인드된 공급자나 보호 정책이 컨테이너에 바인드된 것과 달라야 하는 경우 응용 프로그램 개발자가 메시지 보안을 설정할 수 있습니다.

응용 프로그램 개발자나 어셈블러의 책임은 다음과 같습니다.

- 응용 프로그램에서 응용 프로그램 관련 메시지 보호 정책을 요구하는지 여부를 결정합니다. 요구할 경우, 응용 프로그램 배포자와 통신하여 수행할 수 있는 응용 프로그램 어셈블리에서 필요한 정책을 지정합니다.

보안 토큰 및 보안 체계 정보

WS-Security 사양에서는 보안 토큰을 사용하여 SOAP 웹 서비스 메시지를 인증 및 암호화하는 것과 관련해서 확장 가능한 체계를 제공합니다. Application Server와 함께 설치된 SOAP 계층 메시지 보안 공급자는 사용자 이름/비밀번호 및 X.509 인증서 보안 토큰을 사용하여 SOAP 웹 서비스 메시지를 인증하고 암호화하는 데 사용될 수 있습니다. SAML 명제를 포함하여 다른 보안 토큰을 사용하는 추가 공급자가 Application Server의 후속 릴리스와 함께 설치됩니다.

사용자 이름 토큰 정보

Application Server는 SOAP 메시지의 **사용자 이름 토큰**을 사용하여 메시지 **보낸 사람**의 인증 아이디를 설정합니다. 포함된 비밀번호 내에 사용자 이름 토큰이 포함된 메시지의 수신자는 보낸 사람이 사용자의 비밀(비밀번호)을 알고 있는지 확인하여 토큰에서 식별된 사용자 역할을 할 수 있는 권한이 있는지 확인합니다.

사용자 이름 토큰을 사용할 경우 Application Server에 유효한 사용자 데이터베이스를 구성해야 합니다.

디지털 서명 정보

Application Server는 XML 디지털 서명을 사용하여 인증 아이디를 메시지 **내용**에 바인드합니다. 클라이언트는 디지털 서명을 사용하여 전송 계층 보안을 사용할 경우 동일한 작업을 수행하기 위해 기본 인증 또는 SSL 클라이언트 인증서 인증을 사용한 것과 유사하게 호출자 아이디를 설정합니다. 디지털 서명은 메시지 발신자가 다를 수 있는 메시지 내용의 원본을 인증하기 위해 메시지 수신자가 확인합니다.)

디지털 서명을 사용할 경우 Application Server에 유효한 keystore 및 truststore 파일을 구성해야 합니다. 이 항목에 대한 자세한 내용은 101 페이지 “인증서 파일 정보”를 참조하십시오.

암호화 정보

암호화의 목적은 대상만 이해할 수 있도록 데이터를 수정하는 것입니다. 원본 내용을 암호화된 요소로 대체하여 수정합니다. 공개 키 암호화 도구에서 서술한 경우 암호화를 사용하여 메시지를 읽을 수 있는 당사자의 아이디를 설정할 수 있습니다.

암호화를 사용할 경우 암호화를 지원하는 JCE 공급자를 설치해야 합니다. 이 항목에 대한 자세한 내용은 125 페이지 “JCE 공급자 구성”을 참조하십시오.

메시지 보호 정책 정보

요청 메시지 처리와 응답 메시지 처리에 대해 메시지 보호 정책이 정의되고 원본 또는 수신자 인증을 위한 요구 사항과 관련하여 메시지 보호 정책을 표시합니다. 원본 인증 정책은 메시지를 보냈거나 메시지 내용을 정의한 엔티티의 아이디를 메시지에 설정하여 메시지 수신자가 이를 인증할 수 있도록 하는 요구 사항을 나타냅니다. 수신자 인증 정책은 메시지를 수신할 수 있는 엔티티의 아이디를 메시지 보낸 사람이 설정할 수 있도록 하는 요구 사항을 나타냅니다. SOAP 웹 서비스 메시지 측면에서 메시지 보호 정책이 실현되도록 공급자는 특정 메시지 보안 체계를 적용합니다.

공급자를 컨테이너에 구성할 경우 요청 및 응답 메시지 보호 정책이 정의됩니다. 응용 프로그램 또는 응용 프로그램 클라이언트의 Sun 특정 배포 설명자 내에 웹 서비스 포트나 작업의 단위로 응용 프로그램 관련 메시지 보호 정책을 구성할 수도 있습니다. 어떤 경우든 메시지 보호 정책이 정의된 경우 클라이언트의 요청 및 응답 메시지 보호 정책은 서버의 요청 및 응답 메시지 보호 정책과 일치해야 합니다. 응용 프로그램 관련 메시지 보호 정책 정의에 대한 자세한 내용은 **Developers' Guide**의 **Securing Applications** 장을 참조하십시오.

메시지 보안 용어의 용어집

이 문서에서 사용한 용어는 아래에서 설명합니다. 개념에 대해서는 123 페이지 “메시지 보안을 위한 Application Server 구성 ”에도 설명되어 있습니다.

- 인증 계층

인증 계층은 인증 처리를 수행해야 하는 메시지 계층입니다. Application Server는 SOAP 계층에서 웹 서비스 메시지 보안을 적용합니다.

- 인증 공급자

  이번 Application Server 릴리스에서 Application Server는 SOAP 메시지 계층 보안을 처리하기 위해 **인증 공급자**를 호출합니다.

  - **클라이언트측** 공급자는 서명 또는 사용자 이름/비밀번호로 요청 메시지의 원본 아이디를 설정하거나 암호화로 요청 메시지를 대상 수신자만 볼 수 있도록 보호합니다. 클라이언트측 공급자는 수신한 응답을 해독하여 해당 컨테이너를 수신한 응답의 인증된 수신자로 설정하고 응답의 비밀번호나 서명을 검증하여 응답과 연관된 원본 아이디를 인증합니다. Application Server에 구성된 클라이언트측 공급자를 사용하여 다른 서비스의 클라이언트 역할을 하는 서버측 구성 요소(서블릿 및 EJB 구성 요소)에서 전송한 요청 메시지와 수신한 응답 메시지를 보호할 수 있습니다.
  - **서버측** 공급자는 수신한 요청을 해독하여 해당 컨테이너를 수신한 요청의 인증된 수신자로 설정하고 요청의 비밀번호나 서명을 검증하여 요청과 연관된 원본 아이디를 인증합니다. 서버측 공급자는 서명 또는 사용자 이름/비밀번호로 응답 메시지의 원본 아이디를 설정하고 암호화로 응답 메시지를 대상 수신자만 볼 수 있도록 보호합니다. **서버측 공급자**는 서버측 컨테이너에서만 호출합니다.

- 기본 서버 공급자

  기본 서버 공급자는 특정 서버 공급자가 바인드되지 않은 응용 프로그램에 대해 호출할 서버 공급자를 식별하기 위해 사용합니다. **기본 서버 공급자**는 **기본 공급자**라고도 합니다.

- 기본 클라이언트 공급자

  기본 클라이언트 공급자는 특정 클라이언트 공급자가 바인드되지 않은 응용 프로그램에 대해 호출할 클라이언트 공급자를 식별하기 위해 사용합니다.

- 요청 정책

  요청 정책은 인증 공급자가 수행한 요청 처리와 연관된 인증 정책 요구 사항을 정의합니다. 정책은 메시지 수신자가 해당 서명을 확인하기 전에 메시지를 해독해야 함을 의미하는 내용이 표시된 이후에 암호화가 발생하는 요구 사항과 같은 메시지 보낸 사람 순서에 표시됩니다.

- 응답 정책

  응답 정책은 인증 공급자가 수행한 응답 처리와 연관된 인증 정책 요구 사항을 정의합니다. 정책은 메시지 수신자가 해당 서명을 확인하기 전에 메시지를 해독해야 함을 의미하는 내용이 표시된 이후에 암호화가 발생하는 요구 사항과 같은 메시지 보낸 사람 순서에 표시됩니다.

웹 서비스 보안

Application Server에 배포된 웹 서비스는 SOAP 계층 메시지 보안 공급자와 메시지 보호 정책을 응용 프로그램이 배포된 컨테이너나 응용 프로그램에서 처리한 웹 서비스 종점에 바인드하는 작업을 통해 보호됩니다. SOAP 계층 메시지 보안 기능은 SOAP 계층 메시지 보안 공급자와 메시지 보호 정책을 클라이언트 컨테이너나 클라이언트 응용 프로그램에서 선언한 이식 가능한 서비스 참조에 바인드하는 작업을 통해 Application Server의 클라이언트측 컨테이너에 구성됩니다.

Application Server가 설치될 때 SOAP 계층 메시지 보안 공급자는 Application Server의 클라이언트 및 서버측 컨테이너에 구성됩니다. 이들 공급자는 컨테이너 또는 컨테이너에 배포된 개별 응용 프로그램이나 클라이언트에서 사용할 바인딩에 사용할 수 있습니다. 설치 중 공급자는 간단한 메시지 보호 정책으로 구성됩니다. 즉, 컨테이너나 컨테이너의 응용 프로그램 또는 클라이언트에 바인드된 경우 모든 요청 및 응답 메시지의 내용 원본을 디지털 서명으로 인증하게 됩니다.

Application Server의 관리 인터페이스를 사용하여 Application Server의 서버측 컨테이너에서 사용하기 위해 기존 공급자를 바인드하거나 공급자가 실행한 메시지 보호 정책을 수정하거나 대체 메시지 보호 정책으로 새로운 공급자 구성을 만들 수 있습니다. 128 페이지 “응용 프로그램 클라이언트를 위한 메시지 보안 활성화”에 정의된 대로 비슷한 관리 작업을 응용 프로그램 클라이언트 컨테이너의 SOAP 메시지 계층 보안 구성에서 수행할 수 있습니다.

기본적으로 메시지 계층 보안은 Application Server에서 비활성화됩니다. Application Server에 대한 메시지 계층 보안을 구성하려면 123 페이지 “메시지 보안을 위한 Application Server 구성 ”에 설명된 단계를 수행합니다. Application Server에 배포된 모든 웹 서비스 응용 프로그램을 보호하기 위해 웹 서비스 보안을 사용하려면 127 페이지 “메시지 보안을 위한 공급자 활성화”의 단계를 수행합니다.

Application Server를 다시 시작해야 할 수도 있는 위 단계를 완료하면 웹 서비스 보안이 Application Server에 배포된 모든 웹 서비스 응용 프로그램에 적용됩니다.

응용 프로그램 관련 웹 서비스 보안 구성

응용 프로그램의 Sun 특정 배포 설명자에 요소를 정의하여 응용 프로그램 어셈블리 시 응용 프로그램 관련 웹 서비스 보안 기능이 구성됩니다. 이 요소를 사용하여 특정 공급자나 메시지 보호 정책을 웹 서비스 종점이나 서비스 참조와 연관시킬 수 있고 해당하는 종점이나 참조 대상 서비스의 특정 포트나 메소드에 적용할 수 있습니다.

응용 프로그램 관련 메시지 보호 정책 정의에 대한 자세한 내용은 *Developers' Guide*의 **Securing Applications** 장을 참조하십시오. 이 장에 대한 링크는 115 페이지 “추가 정보”를 참조하십시오.

샘플 응용 프로그램 보안

Application Server는 xms라는 샘플 응용 프로그램과 함께 제공됩니다. xms 응용 프로그램은 J2EE EJB 종점 및 Java Servlet 종점 모두에서 구현되는 간단한 웹 서비스 기능을 합니다. 두 종점은 동일한 서비스 종점 인터페이스를 공유합니다. 서비스 종점 인터페이스는 문자열 인수를 취하는 단일 작업 sayHello를 정의하고 사전 보류 중인 Hello가 구성한 String을 호출 인수에 반환합니다.

xms 샘플 응용 프로그램은 Application Server의 WS-Security 기능을 사용하여 기존의 웹 서비스 응용 프로그램을 보안하는 방법을 보여주기 위해 제공됩니다. 샘플과 함께 제공된 지침에서는 xms 응용 프로그램을 보안하기 위해 사용한 Application Server의 WS-Security 기능을 활성화하는 방법에 대해 설명합니다. 샘플은 122 페이지 "응용 프로그램 관련 웹 서비스 보안 구성" 응용 프로그램에 설명된 대로 WS-Security 기능을 응용 프로그램에 직접 바인딩하는 방법도 보여줍니다.

xms 샘플 응용 프로그램은 다음 디렉토리에 설치됩니다.
install-dir/samples/webservices/security/ejb/apps/xms/

xms 샘플 응용 프로그램의 컴파일, 패키지화 및 실행에 대한 자세한 내용은 **Developers' Guide**의 **Securing Applications** 장을 참조하십시오.

메시지 보안을 위한 Application Server 구성

요청 및 응답 정책 구성 작업

다음 표에서는 메시지 보호 정책 구성과 해당 구성의 WS-Security SOAP 메시지 보안 공급자가 수행한 메시지 보안 작업 결과를 보여줍니다.

표 10–1 메시지 보호 정책과 WS-Security SOAP 메시지 보안 작업 매핑

메시지 보호 정책	WS-Security SOAP 메시지 보호 작업 결과
auth-source="sender"	메시지에는 wsse:UsernameToken(비밀번호 포함)을 포함하는 wsse:Security 헤더가 들어 있습니다.
auth-source="content"	SOAP 메시지 본문의 내용을 서명합니다. 메시지에는 ds:Signature로 표시되는 메시지 본문 서명을 포함하는 wsse:Security 헤더가 들어 있습니다.Signature.

표 10–1 메시지 보호 정책과 WS-Security SOAP 메시지 보안 작업 매핑 *(계속)*

메시지 보호 정책	WS-Security SOAP 메시지 보호 작업 결과
auth-source="sender" auth-recipient="before-content" 또는 auth-recipient="after-content"	SOAP 메시지 본문의 내용이 암호화되고 xend:EncryptedData 결과와 대체됩니다. 메시지에는 wsse:UsernameToken(**비밀번호 포함**) 및 xenc:EncryptedKey가 포함된 wsse:Security 헤더가 들어 있습니다. xenc:EncryptedKey는 SOAP 메시지 본문을 암호화하는 데 사용한 키를 포함합니다. 키는 수신자의 공개 키에 암호화되어 있습니다.
auth-source="content" auth-recipient="before-content"	SOAP 메시지 본문의 내용이 암호화되고 xend:EncryptedData 결과와 대체됩니다. xenc:EncryptedData가 서명됩니다. 메시지에는 xenc:EncryptedKey 및 ds:Signature가 포함된 wsse:Security 헤더가 들어 있습니다. xenc:EncryptedKey는 SOAP 메시지 본문을 암호화하는 데 사용한 키를 포함합니다. 키는 수신자의 공개 키에 암호화되어 있습니다.
auth-source="content" auth-recipient="after-content"	SOAP 메시지 본문의 내용이 서명 및 암호화되고 xend:EncryptedData 결과와 대체됩니다. 메시지에는 xenc:EncryptedKey 및 ds:Signature가 포함된 wsse:Security 헤더가 들어 있습니다. xenc:EncryptedKey는 SOAP 메시지 본문을 암호화하는 데 사용한 키를 포함합니다. 키는 수신자의 공개 키에 암호화되어 있습니다.
auth-recipient="before-content" 또는 auth-recipient="after-content"	SOAP 메시지 본문의 내용이 암호화되고 xend:EncryptedData 결과와 대체됩니다. 메시지에는 xenc:EncryptedKey가 포함된 wsse:Security 헤더가 들어 있습니다. xenc:EncryptedKey는 SOAP 메시지 본문을 암호화하는 데 사용한 키를 포함합니다. 키는 수신자의 공개 키에 암호화되어 있습니다.
지정된 정책이 없습니다.	모듈에서 보안 작업을 수행하지 않습니다.

다른 보안 기능 구성

Application Server는 SOAP 처리 계층에 통합된 메시지 보안 공급자를 사용하여 메시지 보안을 구현합니다. 메시지 보안 공급자는 Application Server의 다른 보안 기능에 따라 달라집니다.

1. Java SDK 1.5.0 이전의 버전을 사용하고 암호화 기술을 사용할 경우 JCE 공급자를 구성합니다.
2. JCE 공급자 구성은 125 페이지 “JCE 공급자 구성”에 설명되어 있습니다.
3. 사용자 이름 토큰을 사용할 경우 필요하면 사용자 데이터베이스를 구성합니다. 토큰을 사용할 경우 해당하는 영역을 구성하고 영역에 해당하는 사용자 데이터베이스를 구성해야 합니다.

4. 필요한 경우 인증서 및 개인 키를 관리합니다.

작업 완료 후

Application Server의 기능이 메시지 보안 공급자가 사용할 수 있도록 구성되면 Application Server와 함께 설치된 공급자가 127 페이지 "메시지 보안을 위한 공급자 활성화"에 설명된 대로 활성화될 수 있습니다.

JCE 공급자 구성

J2SE 1.4.x가 포함된 JCE(Java Cryptography Extension) 공급자는 RSA 암호화를 지원하지 않습니다. WS-Security에서 정의한 XML 암호화는 대개 RSA 암호화를 기반으로 하기 때문에 WS-Security를 사용하여 SOAP 메시지를 암호화하려면 RSA 암호화를 지원하는 JCE 공급자를 다운로드 및 설치해야 합니다.

주 – RSA는 RSA Data Security, Inc.에서 개발한 공개 키 암호화 기술입니다. RSA는 기술 발명자인 Rivest, Shamir, Adelman의 약자입니다.

Application Server를 1.5 버전의 Java SDK에서 실행할 경우 JCE 공급자는 이미 올바르게 구성되어 있습니다. Application Server를 1.4.x 버전의 Java SDK에서 실행할 경우 다음과 같이 JCE 공급자를 정적으로 JDK 환경의 일부로 추가할 수 있습니다.

1. JCE 공급자 JAR(Java ARchive) 파일을 다운로드하여 설치합니다.

   다음 URL에서는 RSA 암호화를 지원하는 JCE 공급자 목록을 제공합니다.
   `http://java.sun.com/products/jce/jce14_providers.html`
2. JCE 공급자 JAR 파일을 *java-home*`/jre/lib/ext/`에 복사합니다.
3. Application Server를 중지합니다.

   Application Server를 중지하지 않고 나중에 이 프로세스에서 다시 시작하면 Application Server가 JCE 공급자를 인식하지 않습니다.
4. 텍스트 편집기에서 *java-home*`/jre/lib/security/java.security` 등록 정보 파일을 편집합니다. 다운로드한 JCE 공급자를 이 파일에 추가합니다.

   `java.security` 파일에는 이 공급자를 추가하기 위한 자세한 지침이 포함되어 있습니다. 기본적으로 유사한 등록 정보를 가진 위치에 다음 형식의 행을 추가해야 합니다.

   `security.provider.`*n*`=`*provider-class-name*

   이 예에서 *n*은 보안 공급자를 평가할 때 Application Server에서 사용하는 기본 설정 순서입니다. 추가한 JCE 공급자에 대해 *n*을 2로 설정합니다.

   예를 들어, Legion of the Bouncy Castle JCE 공급자를 다운로드한 경우 다음 행을 추가합니다.

```
security.provider.2=org.bouncycastle.jce.provider.
   BouncyCastleProvider
```

Sun 보안 공급자가 값 1을 가진 가장 높은 기본 설정을 유지하는지 확인합니다.

```
security.provider.1=sun.security.provider.Sun
```

다른 보안 공급자의 수준을 하향 조정하여 각 수준에 하나의 보안 공급자만 있도록 합니다.

다음은 필요한 JCE 공급자를 제공하고 기존 공급자를 올바른 위치에 보관하는 `java.security` 파일의 예입니다.

```
security.provider.1=sun.security.provider.Sun
security.provider.2=org.bouncycastle.jce.provider.
   BouncyCastleProvider
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.rsajca.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
```

5. 파일을 저장하고 닫습니다.
6. Application Server를 다시 시작합니다.

메시지 보안 설정

메시지 보안을 사용하기 위해 Application Server를 설정하는 대부분의 단계는 관리 콘솔이나 `asadmin` 명령줄 도구를 사용하여 수행하거나 시스템 파일을 수동으로 편집하여 수행할 수 있습니다. 일반적으로 시스템 파일을 편집하는 것은 의도하지 않은 변경으로 인해 Application Server가 제대로 실행되지 못하게 할 수 있기 때문에 권장하지 않습니다. 따라서 가능한 한 관리 콘솔을 사용하여 Application Server를 구성하는 단계가 먼저 표시되고 `asadmin` 도구 명령을 사용한 단계는 나중에 표시됩니다. 관리 콘솔이나 해당 `asadmin` 명령이 없을 경우에만 시스템 파일을 수동으로 편집하는 단계가 표시됩니다.

메시지 계층 보안 지원이 플러그 가능한 인증 모듈 양식으로 클라이언트 컨테이너와 Application Server에 통합됩니다. 기본적으로 메시지 계층 보안은 Application Server에서 비활성화됩니다. 다음 절에서는 메시지 보안 구성 및 공급자를 활성화 작성 편집 및 삭제하는 것과 관련한 세부 사항을 제공합니다.

- 127 페이지 “메시지 보안을 위한 공급자 활성화”
- 127 페이지 “메시지 보안 공급자 구성”
- 128 페이지 “메시지 보안 공급자 만들기”
- 128 페이지 “응용 프로그램 클라이언트를 위한 메시지 보안 활성화”
- 129 페이지 “응용 프로그램 클라이언트 구성에 대한 요청 및 응답 정책 설정”
- 130 페이지 “추가 정보”

대부분의 경우 위에 나열된 관리 작업을 수행한 후 Application Server를 다시 시작해야 합니다. 특히 작업이 수행된 시점에 Application Server에 이미 배포된 응용 프로그램에 관리 변경 사항을 적용하려는 경우 이 작업을 수행합니다.

메시지 보안을 위한 공급자 활성화

Application Server에 배포된 웹 서비스 종점에 대한 메시지 보안을 활성화하려면 서버측에서 기본적으로 사용되는 공급자를 지정해야 합니다. 메시지 보안을 위한 기본 공급자를 활성화한 경우 Application Server에 배포된 웹 서비스의 클라이언트에서 사용할 공급자도 활성화해야 합니다. 클라이언트가 사용하는 공급자를 활성화하는 방법에 대한 자세한 내용은 128 페이지 “응용 프로그램 클라이언트를 위한 메시지 보안 활성화”를 참조하십시오.

배포된 종점에서 발생한 웹 서비스 호출에 대해 메시지 보안을 활성화하려면 기본 클라이언트 공급자를 지정해야 합니다. Application Server에 대한 기본 클라이언트 공급자를 활성화한 경우 Application Server에 배포된 종점에서 호출한 모든 서비스가 메시지 계층 보안과 호환 가능하게 구성되도록 해야 합니다.

다음과 같이 명령줄 유틸리티를 사용합니다.

- 기본 서버 공급자를 지정하려면 다음 작업을 수행합니다.

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_provider=ServerProvider
```

- 기본 클라이언트 공급자를 지정하려면 다음 작업을 수행합니다.

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_client_provider=ClientProvider
```

메시지 보안 공급자 구성

일반적으로 공급자 유형 구현 클래스 및 공급자별 구성 등록 정보를 수정할 수 있더라도 메시지 보호 정책을 수정하기 위해 공급자가 다시 구성됩니다.

명령줄 유틸리티를 사용하여 응답 정책을 설정하려면 다음 명령에서 단어 `request`를 `response`로 바꿉니다.

- 요청 정책을 클라이언트에 추가하고 인증 소스를 설정합니다.

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_source=
sender | content
```

- 요청 정책을 서버에 추가하고 인증 소스를 설정합니다.

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_source=
sender | content
```

- 요청 정책을 클라이언트에 추가하고 인증 수신자를 설정합니다.

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_recipient=
before-content | after-content
```

- 요청 정책을 서버에 추가하고 인증 수신자를 설정합니다.

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_recipient=
before-content | after-content
```

메시지 보안 공급자 만들기

관리 콘솔을 사용하여 기존 공급자를 구성하려면 구성 노드 > 구성할 인스턴스 > 보안 노드 > 메시지 보안 노드 > SOAP 노드 > 공급자 탭을 선택합니다.

메시지 보안 공급자 만들기에 대한 자세한 내용은 관리 콘솔 온라인 도움말을 참조하십시오.

응용 프로그램 클라이언트를 위한 메시지 보안 활성화

클라이언트 공급자의 메시지 보호 정책이 클라이언트 공급자가 상호 작용하는 서버측 공급자의 메시지 보호 정책과 동등하도록 구성해야 합니다. Application Server가 설치될 때 구성되었지만 활성화되지 않은 공급자의 경우 이미 이 상태로 되어 있습니다.

클라이언트 응용 프로그램에 대한 메시지 보안을 활성화하려면 응용 프로그램 클라이언트 컨테이너의 Application Server 관련 구성을 수정합니다.

응용 프로그램 클라이언트 구성에 대한 요청 및 응답 정책 설정

요청 및 응답 정책에서는 인증 공급자가 수행한 요청 및 응답 처리와 연관된 인증 정책 요구 사항을 정의합니다. 정책은 메시지 수신자가 해당 서명을 확인하기 전에 메시지를 해독해야 함을 의미하는 내용이 표시된 이후에 암호화가 발생하는 요구 사항과 같은 메시지 보낸 사람 순서에 표시됩니다.

메시지 보안을 달성하려면 서버와 클라이언트 모두에서 요청 및 응답 정책을 활성화해야 합니다. 클라이언트와 서버에 정책을 구성할 경우 응용 프로그램 수준 메시지 바인딩에서 요청 응답 보호를 위해 서버 정책과 클라이언트 정책이 일치해야 합니다.

응용 프로그램 클라이언트 구성에 대한 요청 정책을 설정하려면 128 페이지 “응용 프로그램 클라이언트를 위한 메시지 보안 활성화”에 설명된 대로 응용 프로그램 클라이언트 컨테이너에 대한 Application Server 관련 구성을 수정합니다. 응용 프로그램 클라이언트 구성 파일에 다음과 같이 `request-policy` 및 `response-policy`를 추가하여 요청 정책을 설정합니다.

참조를 위해 다른 코드가 제공됩니다. 다른 코드는 설치에 따라 약간씩 달라집니다. 이것을 변경하지 마십시오.

```
<client-container>
  <target-server name="your-host" address="your-host"
      port="your-port"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
      default-client-provider="ClientProvider">
    <provider-config
        class-name="com.sun.enterprise.security.jauth.ClientAuthModule"
        provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
       <property name="security.config"
           value="install-dir/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

`auth-source`의 유효한 값에 `sender`와 `content`가 포함됩니다. `auth-recipient`의 유효한 값에 `before-content` 및 `after-content`가 포함됩니다. 이 값이 다양하게 결합된 결과를 설명하는 표는 123 페이지 “요청 및 응답 정책 구성 작업”에 있습니다.

요청이나 응답 정책을 지정하지 않으려면 예를 들어 다음 요소를 비워둡니다.

```
<response-policy/>
```

추가 정보

- Java 2 Standard Edition 보안 설명은 `http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html`에서 볼 수 있습니다.
- *J2EE 1.4 Tutorial*의 *Security* 장은 `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`에서 볼 수 있습니다.
- **관리 설명서**의 9 장
- **Developer's Guide**의 **Securing Applications** 장
- Oasis 웹 서비스 보안: SOAP 메시지 보안(WS-Security) 사양은 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf에서 볼 수 있습니다.
- OASIS Web Services Security Username Token Profile 1.0은 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf에서 볼 수 있습니다.
- OASIS Web Services Security X.509 Certificate Token Profile 1.0은 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf에서 볼 수 있습니다.
- **XML-Signature Syntax and Processing** 설명서는 `http://www.w3.org/TR/xmldsig-core/`에서 볼 수 있습니다.
- **XML Encryption Syntax and Processing** 설명서는 `http://www.w3.org/TR/xmlenc-core/`에서 볼 수 있습니다.

◆ ◆ ◆ 11장

11 트랜잭션

분할할 수 없는 작업 단위에 하나 이상의 단계를 포함시켜 트랜잭션에서 데이터 무결성과 일관성을 보장합니다. 이 장은 다음 내용으로 구성되어 있습니다.

트랜잭션

트랜잭션은 응용 프로그램에서 모두 성공적으로 완료하지 않으면 각 작업의 변경 사항이 철회되는 일련의 작업입니다. 예를 들어 당좌 계좌의 자금을 저축 계좌로 대체하는 것은 다음 단계로 구성된 트랜잭션입니다.

1. 당좌 계좌에 대체에 충분한 자금이 있는지 확인합니다.
2. 당좌 계좌에 충분한 자금이 있을 경우 당좌 계좌의 금액을 차변에 기입합니다.
3. 저축 계좌의 대변에 자금을 기입합니다.
4. 당좌 계좌 로그에 대체를 기록합니다.
5. 저축 계좌 로그에 대체를 기록합니다.

이 단계 중 어느 한 단계라도 실패할 경우 이전 단계의 모든 변경 사항이 철회되고 당좌 계좌와 저축 계좌는 트랜잭션 시작 전과 동일한 상태가 되어야 합니다. 이 이벤트를 **롤백**이라고 합니다. 모든 단계가 성공적으로 완료되면 트랜잭션은 **완결된** 상태에 있습니다. 트랜잭션은 완결 또는 롤백 상태로 종료됩니다.

J2EE 기술의 트랜잭션

J2EE 기술에서 트랜잭션을 처리하는 데는 다음 5명의 참가자가 관련됩니다.

- 트랜잭션 관리자
- Application Server
- 자원 관리자
- 자원 어댑터
- 사용자 응용 프로그램

각 엔티티는 다음에서 설명하는 여러 API 및 기능을 구현하여 신뢰할 수 있는 방법으로 트랜잭션을 처리합니다.

- 트랜잭션 관리자는 트랜잭션 구분, 트랜잭션 자원 관리, 동기화 및 트랜잭션 컨텍스트 전파 지원에 필요한 서비스 및 관리 기능을 제공합니다.
- Application Server는 트랜잭션 상태 관리 등의 응용 프로그램 런타임 환경 지원에 필요한 인프라를 제공합니다.
- 자원 관리자(자원 어댑터를 통한)는 응용 프로그램에게 자원에 대한 액세스를 제공합니다. 자원 관리자는 트랜잭션 관리자가 트랜잭션 연결, 트랜잭션 완료 및 복구 작업과 통신할 때 사용하는 트랜잭션 자원 인터페이스를 구현하여 분산된 트랜잭션에 참여합니다. 이러한 자원 관리자의 예로 관계형 데이터베이스 서버를 들 수 있습니다.
- 자원 어댑터는 Application Server 또는 클라이언트가 자원 관리자와 연결할 때 사용하는 시스템 수준 소프트웨어 라이브러리입니다. 자원 어댑터는 일반적으로 자원 관리자마다 따로 지정됩니다. 자원 어댑터는 라이브러리로 사용 가능하며 이것을 사용하는 클라이언트 주소 공간 내에서 사용됩니다. 이러한 자원 어댑터의 예는 JDBC 드라이버입니다.
- Application Server 환경에서 실행되도록 개발된 트랜잭션 사용자 응용 프로그램은 JNDI를 사용하여 트랜잭션 데이터 소스를 조회할 뿐만 아니라 필요에 따라 트랜잭션 관리자도 조회합니다. 응용 프로그램은 Enterprise Bean에 대한 선언적 트랜잭션 속성 설정 또는 명시적 프로그래밍 방식의 트랜잭션 경계를 사용할 수 있습니다.

트랜잭션 복구

서버나 자원 관리자의 문제로 인해 트랜잭션이 완료되지 않을 수 있습니다. 이 경우, 문제가 있는 트랜잭션을 완료하고 실패를 복구해야 합니다. Application Server는 이 실패를 복구하고 서버 시작 시 트랜잭션을 완료하도록 설계되었습니다.

복구를 수행하는 중 일부 자원에 연결할 수 없을 경우 트랜잭션 복구를 시도하기 때문에 서버 재시작이 지연될 수 있습니다.

트랜잭션이 여러 서버에 걸쳐 있는 경우 트랜잭션을 시작한 서버는 다른 서버에 연결하여 트랜잭션의 결과를 가져올 수 있습니다. 다른 서버에 연결할 수 없는 경우 트랜잭션은 발견적 판단 필드를 사용하여 결과를 확인합니다.

관리 콘솔을 사용하여 Application Server가 트랜잭션을 복구하도록 구성할 수 있습니다. 이 작업에 대한 자세한 절차는 관리 콘솔 온라인 도움말을 참조하십시오.

트랜잭션 시간 초과 값

기본적으로 서버는 트랜잭션을 시간 초과하지 않습니다. 즉, 서버는 트랜잭션이 완료될 때까지 무기한 기다립니다. 트랜잭션의 시간 초과 값을 설정하면 트랜잭션이 구성된 시간 내에 완료되지 않을 경우 Application Server에서 트랜잭션을 롤백합니다. 이 작업에 대한 자세한 절차는 관리 콘솔 온라인 도움말을 참조하십시오.

트랜잭션 로그

트랜잭션 로그는 관련된 자원의 데이터 무결성을 유지하고 오류를 복구하기 위해 각 트랜잭션에 대한 정보를 기록합니다. 트랜잭션 로그 위치 필드에서 지정한 디렉토리의 `tx` 하위 디렉토리에 트랜잭션 로그가 저장됩니다. 이 로그는 사람이 읽을 수 없습니다.

키포인트 간격

키 포인트 작업은 트랜잭션 로그 파일을 압축합니다. 키 포인트 간격은 로그 상의 키 포인트 작업 간 트랜잭션 수입니다. 키 포인트 작업으로 트랜잭션 로그 파일 크기가 줄어들 수 있습니다. 키 포인트 간격이 크면(예: 2048) 트랜잭션 로그 파일이 더 커지고 키 포인트 작업이 적어질수록 성능이 더 좋아질 가능성이 있습니다. 키포인트 간격이 작아지면(예: 256) 로그 파일이 작아지지만, 키 포인트 작업 빈도가 높아져서 성능이 약간 낮아질 수 있습니다.

◆ ◆ ◆ 12 장

12 서비스 구성

HTTP 서비스는 웹 응용 프로그램을 배포하고 배포된 웹 응용 프로그램을 HTTP 클라이언트에서 액세스할 수 있게 하는 기능을 제공하는 Application Server의 구성 요소입니다이 기능은 가상 서버와 HTTP Listener라는 두 종류의 관련된 객체 수단으로 제공됩니다.

이 장은 다음 내용으로 구성되어 있습니다.

가상 서버

가상 호스트라고도 하는 가상 서버는 동일한 물리적 서버에서 여러 개의 인터넷 도메인 이름을 호스트할 때 사용할 수 있는 객체입니다. 동일한 물리적 서버에서 호스트되는 모든 가상 서버는 해당 서버의 인터넷 프로토콜(IP) 주소를 공유합니다. 가상 서버는 서버의 도메인 이름(예: `www.aaa.com`)을 Application Server가 실행 중인 특정 서버와 연결합니다.

주 – 인터넷 도메인을 Application Server의 관리 도메인과 혼동하지 마십시오.

예를 들어, 물리적 서버에서 다음 도메인을 호스트하려고 합니다.

```
www.aaa.com
www.bbb.com
www.ccc.com
```

`www.aaa.com`, `www.bbb.com` 및 `www.ccc.com`에 각각 연관된 `web1`, `web2` 및 `web3` 웹 모듈이 있다고 가정합니다.

이러한 모든 URL을 물리적 서버에서 처리한다는 것을 의미합니다.

```
http://www.aaa.com:8080/web1
http://www.bbb.com:8080/web2
http://www.ccc.com:8080/web3
```

첫 번째 URL은 `www.aaa.com` 가상 호스트에 매핑되고 두 번째 URL은 `www.bbb.com` 가상 호스트에 매핑되며, 세 번째 URL은 `www.ccc.com` 가상 호스트에 매핑됩니다.

반면, 다음의 URL에서는 `web3`이 `www.bbb.com`에 등록되지 않았으므로 404 반환 코드가 반환됩니다.

```
http://www.bbb.com:8080/web3
```

이 매핑이 작동하려면 `www.aaa.com`, `www.bbb.com` 및 `www.ccc.com` 모두 물리적 서버의 IP 주소로 확인되어야 합니다. 이들을 네트워크의 DNS 서버에 등록해야 합니다. 또한 UNIX 시스템의 경우 이 도메인을 `/etc/hosts` 파일에 추가합니다(`/etc/nsswitch.conf` 파일의 `hosts`에 대한 설정에 `files`가 포함된 경우).

Application Server를 시작하면 다음과 같은 가상 서버를 자동으로 시작합니다.

- 모든 사용자 정의 웹 모듈을 호스트하는 `server`라는 가상 서버
- 모든 관리 관련 웹 모듈(특히 관리 콘솔)을 호스트하는 `__asadmin`이라는 가상 서버. 이 서버는 제한적이므로 이 가상 서버에 웹 모듈을 배포할 수 없습니다.

프로덕션 이외의 환경에서 웹 서비스를 개발, 테스트 및 배포하기 위해서는 대개 `server`만이 유일하게 필요한 가상 서버입니다. 프로덕션 환경에서는 물리적 서버가 하나만 있더라도 고유한 웹 서버를 가진 것으로 표시되도록 추가 가상 서버에서 사용자와 고객을 위한 호스팅 기능을 제공합니다.

HTTP Listener

가상 서버마다 하나 이상의 HTTP Listener를 통해 서버와 클라이언트 간의 연결을 제공합니다. 각 HTTP Listener는 IP 주소, 포트 번호, 서버 이름 및 기본 가상 서버를 가진 수신 소켓입니다.

HTTP Listener의 포트 번호와 IP 주소의 조합은 고유해야 합니다. 예를 들어, IP 주소를 0.0.0.0으로 지정하면 HTTP Listener를 시스템의 지정된 포트에 구성된 모든 IP 주소에서 수신할 수 있습니다. 또는 HTTP Listener에서 각 Listener에 대해 고유한 IP 주소를 지정할 수 있지만 동일한 포트를 사용합니다.

HTTP Listener는 IP 주소와 포트 번호의 조합이므로 동일한 IP 주소와 다른 포트 번호(예: 1.1.1.1:8081 및 1.1.1.1:8082)를 갖거나 다른 IP 주소와 동일한 포트 번호(예: 1.1.1.1:8081 및 1.2.3.4:8081, 시스템이 해당 주소에 모두 응답하도록 구성되어 있는 경우)를 갖는 여러 HTTP Listener가 있을 수 있습니다.

그러나 특정 포트에서 모든 IP 주소에서 수신하는 0.0.0.0 IP 주소를 HTTP Listener에서 사용할 경우, 같은 포트에서 특정한 IP 주소를 수신하는 추가 IP 주소를 위한 HTTP

Listener를 만들 수 없습니다. 예를 들어, 한 개의 HTTP Listener가 0.0.0.0:8080(포트 8080의 모든 IP 주소)을 사용하면 다른 HTTP Listener는 1.2.3.4:8080을 사용할 수 없습니다.

Application Server를 실행 중인 시스템은 대개 하나의 IP 주소만 액세스하기 때문에 HTTP Listener는 대개 0.0.0.0 IP 주소와 다른 포트 번호를 사용합니다. 각 포트 번호는 다른 용도에 사용됩니다. 시스템이 둘 이상의 IP 주소에 액세스할 경우 각 주소를 다른 용도로 사용할 수 있습니다.

기본적으로 Application Server가 시작되면 다음과 같은 HTTP Listener를 갖고 있습니다.

- `server`라는 가상 서버와 연관된 `http-listener-1` 및 `http-listener-2`라는 두 개의 HTTP Listener. `http-listener-1`이라고 하는 Listener에는 보안이 활성화되지 않고, `http-listener-2`에는 보안이 활성화되어 있습니다.
- `__asadmin`이라는 가상 서버와 연관된 `admin-listener`라고 하는 HTTP Listener에는 보안이 활성화되어 있습니다.

이러한 모든 Listener는 IP 주소 0.0.0.0과 Application Server 설치 중에 HTTP 서버 포트 번호로 지정한 포트 번호를 사용합니다. Application Server가 기본 포트 번호 값을 사용하면 `http-listener-1`은 포트 8080을 사용하고 `http-listener-2`는 포트 8181을 사용하며 `admin-listener`는 포트 4849를 사용합니다.

각 HTTP Listener에는 기본 가상 서버가 있습니다. 기본 가상 서버는 호스트 구성 요소가 HTTP Listener와 연결된 가상 서버와 일치하지 않는 모든 요청 URL을 HTTP Listener가 라우팅하는 서버입니다. 가상 서버는 `http-listeners` 속성에 HTTP Listener를 나열하여 HTTP Listener와 연결합니다.

또한 HTTP Listener에 억셉터 스레드 수도 지정합니다. 억셉트 스레드는 연결을 기다리는 스레드입니다. 스레드는 연결을 승인하고 이 연결을 작업자 스레드에서 선택할 수 있도록 연결 대기열이라고 하는 대기열에 둡니다. 새로운 요청이 있을 때 항상 사용할 수 있도록 억셉터 스레드를 충분히 구성합니다. 그러나 시스템에 많은 부담을 주지 않도록 적당히 구성합니다.연결 대기열은 억셉터 스레드가 방금 수락한 새 연결과 연결 유지 연결 관리 하위 시스템이 관리하는 지속적인 연결을 모두 포함합니다.

요청 처리 스레드 집합은 연결 대기열에서 들어오는 HTTP 요청을 검색하여 처리합니다. 이 스레드는 헤더를 구문 분석하고 해당하는 가상 서버를 선택한 다음 요청 처리 엔진을 실행하여 요청을 서비스합니다. 처리할 요청이 더 이상 없지만 연결에서 `HTTP/1.1`을 사용하거나 `Connection: keep-alive` 헤더를 전송하여 지속성을 유지할 수 있는 경우 요청 처리 스레드에서는 연결이 유휴 상태인 것으로 가정하고 그 연결을 연결 유지 연결 관리 하위 시스템으로 전달합니다.

연결 유지 하위 시스템은 정기적으로 유휴 연결을 폴링하여 나중에 처리할 수 있도록 작업이 있는 연결을 연결 대기열에 넣어둡니다. 여기에서 요청 처리 스레드는 다시 연결을 검색하고 요청을 처리합니다. 연결 유지 하위 시스템은 수 만개의 연결을 관리할 수 있기 때문에 다중 스레딩되어 있습니다. 연결 개수를 더 작은 하위 집합으로 나누는 효율적인 폴링 기술을 사용하여 요청이 준비된 연결과 닫힌 것으로 간주해도 될 정도로 허용 가능한 최대 연결 유지 시간을 초과한 것으로 충분히 유휴 상태였던 연결을 확인합니다.

HTTP Listener의 서버 이름은 서버가 리디렉션의 일부로 클라이언트에 전송한 URL에 표시되는 호스트 이름입니다. 이 속성은 서버에서 자동으로 생성하는 URL에 영향을 주지만, 서버에 저장된 디렉토리 및 파일의 URL에는 영향을 주지 않습니다. 서버에서 별칭을 사용하는 경우 이 이름은 대개 별칭 이름입니다. 클라이언트가 `Host:` 헤더를 보내면 호스트 이름이 리디렉션에서 HTTP Listener의 서버 이름 값을 대체합니다.

원래 요청에 지정된 것과는 다른 포트 번호를 사용하도록 리디렉션 포트를 지정합니다. 다음 상황 중 하나에서 **리디렉션**이 발생합니다.

- 클라이언트가 지정한 URL에 더 이상 존재하지 않는 자원(즉, 다른 위치로 이동한 자원)에 액세스할 경우 서버는 404를 반환하는 대신 지정한 응답 코드를 반환하고 응답의 위치 헤더에 새로운 위치를 포함시켜서 클라이언트를 새로운 위치로 리디렉션합니다.
- 클라이언트가 정규 HTTP 포트에서 보호된 자원(예: SSL)에 액세스할 경우 서버는 이 요청을 SSL 가능 포트로 리디렉션합니다. 이 경우 서버는 원래 비보안 포트가 SSL 가능 포트로 대체된 위치 응답 헤더에 새로운 URL을 반환합니다. 그러면 클라이언트는 새 URL에 연결됩니다.

HTTP Listener에 대한 보안 활성화 여부 및 사용한 보안 종류(예: SSL 프로토콜 및 암호화)도 지정합니다.

Application Server에 배포된 웹 응용 프로그램에 액세스하려면 웹 응용 프로그램에 지정된 컨텍스트 루트와 함께 URL `http://localhost:8080/`(보안 처리된 응용 프로그램일 경우 `http://localhost:8080/`)을 사용합니다. 관리 콘솔에 액세스하려면 `https://localhost:4849/` 또는 `https://localhost:4849/asadmin/`(기본 컨텍스트 루트)을 사용합니다.

가상 서버에서 기존 HTTP Listener를 지정해야 하고 다른 가상 서버에서 이미 사용 중인 HTTP Listener는 지정할 수 없기 때문에 새로운 가상 서버를 만들기 전에 최소한 하나의 HTTP Listener를 만들어야 합니다.

◆ ◆ ◆ 13장

13 ORB(Object Request Broker) 구성

이 장에서는 ORB(Object Request Broker) 및 IIOP Listener를 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다

CORBA

Application Server는 표준 프로토콜 및 형식 집합을 지원하여 상호 운용성을 보장합니다. 이 프로토콜 중 일부는 CORBA에서 정의합니다.

CORBA(Common Object Request Broker Architecture) 모델은 원격 메소드 요청의 형태로 객체에 요청을 발행하여 잘 정의된 인터페이스를 통해 분산 객체나 서버에서 서비스를 요청하는 클라이언트를 기반으로 합니다. 원격 메소드 요청은 호출된 메소드에 대한 서비스 공급자와 매개 변수의 객체 이름(객체 참조라고 함)을 포함하여 수행해야 하는 작업에 대한 정보를 전송합니다. CORBA는 객체 등록, 객체 위치 지정, 객체 활성화, 요청 멀티플렉싱 해제, 오류 처리, 마샬 및 작업 디스패치 등과 같은 많은 네트워킹 프로그래밍 작업을 자동으로 처리합니다.

ORB

ORB(Object Request Broker)는 CORBA의 핵심 구성 요소입니다. ORB는 객체를 식별하여 찾고 연결 관리를 처리하며 데이터를 전달하고 통신을 요청하는 데 필요한 인프라를 제공합니다.

CORBA 객체는 서로 직접 통신하지 않습니다. 대신, 객체는 로컬 시스템에서 실행 중인 ORB에 대한 원격 스텁을 통해 요청합니다. 그러면 로컬 ORB는 IIOP(Internet Inter-Orb

Protocol)를 사용하여 다른 시스템의 ORB에 이 요청을 전달합니다. 원격 ORB는 적절한 객체를 찾고 요청을 처리하며, 결과를 반환합니다.

IIOP는 RMI-IIOP를 사용하여 응용 프로그램이나 객체에서 RMI(Remote Method Invocation) 프로토콜로 사용할 수 있습니다. Enterprise Bean(EJB 모듈)의 원격 클라이언트는 RMI-IIOP를 통해 Application Server와 통신합니다.

IIOP Listener

IIOP Listener는 Enterprise Bean의 원격 클라이언트와 다른 CORBA 기반 클라이언트에서 들어오는 연결을 받아들이는 수신 소켓입니다. Application Server에 대해 여러 IIOP Listener를 구성할 수 있습니다. 각 Listener에 대해 포트 번호와 네트워크 주소, 그리고 필요에 따라 보안 속성을 지정합니다.

ORB 작업

IIOP Listener를 만들려면 관리 콘솔에서 구성 > ORB > IIOP Listener를 선택하고 새로 만들기를 누릅니다. 또는 다음과 같은 asadmin 명령을 사용하여 IIOP Listener를 만들 수 있습니다. `create-iiop-listener`(1) 및 `create-ssl`(1).

IIOP Listener를 편집하려면 관리 콘솔에서 구성 > ORB > IIOP Listener를 선택하고 수정할 Listener를 선택합니다. 설정을 수정합니다. 포트 번호를 변경한 경우에는 서버를 다시 시작해야 합니다. ORB는 스레드 풀을 사용하여 RMI-IIOP를 통해 통신하는 다른 클라이언트와 Enterprise Bean의 원격 클라이언트에서 들어오는 요청에 응답합니다.

IIOP Listener를 삭제하려면 관리 콘솔에서 구성 > ORB > IIOP Listener를 선택하고 삭제할 Listener를 선택합니다. 또는 `delete-iiop-listener`(1) 명령을 사용할 수도 있습니다.

`ORBCommunicationsRetryTimeout` 등록 정보는 ORB 클라이언트가 연결할 수 없는 ORB 백엔드에 연결 구성을 시도할 시간(초)을 지정합니다. 기본값은 `60`초입니다. 이 기본값으로 설정되어 있는 경우 ORB 백엔드에 연결할 수 없으면 네트워크 사용량이 많아질 뿐만 아니라 로그에 수많은 CORBA 예외가 기록될 수 있습니다.

이 경우, `ORBCommunicationsRetryTimeout`을 낮은 값으로 설정합니다.

타사 ORB

Sun Java System Application Server는 타사 ORB 소프트웨어와 함께 사용할 수 있습니다. 타사 ORB를 지원하려면 서버측 설정을 수정해야 합니다.

Application Server에 타사 ORB에 대한 지원을 구현하려면 `domain.xml` 및 `server.policy` 파일을 편집해야 합니다. 타사 ORB 샘플을 구성하는 방법에 대한 자세한 내용은

Configuring Sun Java System Application Server for Third-Party ORBs (http://developers.sun.com/prodtech/appserver/reference/techart/orb.html)를 참조하십시오.

◆ ◆ ◆ 14장

14 스레드 풀

Java Virtual Machine(JVM)에서는 한번에 여러 스레드 실행을 지원할 수 있습니다. 성능 향상을 위해 Application Server에서는 하나 이상의 스레드 풀을 유지합니다. 특정 스레드 풀을 커넥터 모듈과 ORB에 할당할 수 있습니다.

하나의 스레드 풀이 여러 커넥터 모듈과 Enterprise Bean을 처리할 수 있습니다. 요청 스레드는 응용 프로그램 구성 요소에 대한 사용자 요청을 처리합니다. 서버는 요청을 받으면 요청을 스레드 풀의 여유 스레드에 할당합니다. 스레드는 클라이언트의 요청을 실행하여 결과를 반환합니다. 예를 들어, 요청이 현재 작업 중인 시스템 자원을 사용해야 하는 경우 스레드는 자원의 작업이 끝날 때까지 기다린 후 요청이 해당 자원을 사용할 수 있도록 합니다.

응용 프로그램의 요청에 예약된 스레드의 최소 수와 최대 수를 지정합니다. 스레드 풀은 이러한 두 값 사이에서 동적으로 조절됩니다. 지정한 최소 스레드 풀 크기는 응용 프로그램 요청에 대한 예약에 적어도 그 수만큼의 스레드를 할당하라는 신호를 서버에게 보냅니다. 이 수는 지정한 최대 스레드 풀 크기만큼 증가됩니다.

프로세스가 사용할 수 있는 스레드의 수를 늘리면 프로세스는 더 많은 응용 프로그램 요청에 동시에 응답할 수 있습니다.

한 개의 자원 어댑터나 응용 프로그램이 Application Server의 모든 스레드를 차지하는 곳에서는 Application Server의 스레드를 서로 다른 스레드 풀로 나누어 스레드 고갈을 방지합니다.

이 장은 다음 내용으로 구성되어 있습니다.

스레드 풀 구성

관리 콘솔을 사용하여 스레드 풀을 만들려면 구성 > 스레드 풀 > 현재 풀 > 새로 만들기로 이동합니다.

- 스레드 풀 아이디 필드에 스레드 풀 이름을 입력합니다.
- 최소 스레드 풀 크기 필드에 이 대기열에서 요청을 처리하는 스레드 풀의 최소 스레드 수를 입력합니다.

  이 스레드 풀을 인스턴스화할 경우 이 스레드가 위에 만들어집니다.
- 최대 스레드 풀 크기 필드에 이 대기열에서 요청을 처리하는 스레드 풀의 최대 스레드 수를 입력합니다.

  스레드 풀에 존재하는 스레드 수에 대한 상한 값입니다.
- 유휴 시간 초과 필드에 풀에서 유휴 스레드를 제거하는 시간(초)을 입력합니다.
- 작업 대기열 수 필드에 이 스레드 풀이 처리하는 총 작업 대기열 수를 입력합니다.
- Application Server를 다시 시작합니다.

스레드 풀 만들기에 대한 자세한 내용을 보려면 관리 콘솔의 도움말을 누르십시오.

또한 명령줄에서 `asadmin` 명령 `create-threadpool`(1)을 사용하여 스레드 풀을 만들 수도 있습니다.

관리 콘솔을 사용하여 스레드 풀의 설정을 편집하려면 구성 > 스레드 풀 > 현재 풀로 이동하고 구성할 풀을 선택합니다. 선택한 스레드 풀의 값을 수정하고 저장한 후 Application Server를 다시 시작합니다.

스레드 풀 편집에 대한 자세한 내용을 보려면 관리 콘솔의 도움말을 누르십시오.

관리 콘솔을 사용하여 스레드 풀을 삭제하려면 구성 > 스레드 풀 > 현재 풀로 이동합니다. 삭제할 스레드 풀 이름을 선택하고 삭제를 누릅니다.

Application Server를 다시 시작합니다.

또한 명령줄에서 `asadmin` 명령 `delete-threadpool`(1)을 사용하여 스레드 풀을 삭제할 수도 있습니다.

◆ ◆ ◆ 15장

15 로깅 구성

이 장에서는 로깅을 구성하고 서버 로그를 보는 방법에 대해 간단하게 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

로그 레코드

Application Server는 JSR 047에 지정된 Java 2 플랫폼 로깅 API를 사용합니다. 일반적으로 Application Server 로깅 메시지는 *domain-dir*`/logs/server.log`에 위치한 서버 로그에 기록됩니다.

domain-dir`/logs` 디렉토리에는 서버 로그와 두 가지 다른 종류의 로그가 있습니다. `access` 하위 디렉토리에는 HTTP 서비스 액세스 로그가 있고, `tx` 하위 디렉토리에는 트랜잭션 서비스 로그가 있습니다. 이 로그에 대한 자세한 내용은 관리 콘솔 온라인 도움말을 참조하십시오.

Application Server 구성 요소는 로깅 출력을 생성합니다. 응용 프로그램 구성 요소에서도 로깅 출력을 생성할 수 있습니다.

응용 프로그램 구성 요소에서 Apache Commons Logging Library를 사용하여 메시지를 로그할 수 있습니다. 그러나 더 좋은 로그 구성을 위해서는 플랫폼 표준 JSR 047 API를 권장합니다.

로그 레코드는 다음과 같은 일관된 형식을 따릅니다.

```
[#|yyyy-mm-ddThh:mm:ss.SSS-Z|Log Level|ProductName-Version|LoggerName|Key Value Pairs|Message|#]
```

예를 들면 다음과 같습니다.

```
[#|2004-10-21T13:25:53.852-0400|INFO|sun-appserver-e8.1|javax.enterprise.
system.core|_ThreadID=13;|CORE5004: Resource Deployed:
[cr:jms/DurableConnectionFactory].|#]
```

이 예에서

- `[#` 및 `#]`은 레코드의 시작과 끝을 표시합니다.
- 세로 막대(`|`)는 레코드 필드를 구분합니다.
- `2004-10-21T13:25:53.852-0400`은 날짜와 시간을 지정합니다.
- *Log Level*은 `INFO`입니다. 이 수준은 다음 값 중 하나일 수 있습니다. `SEVERE`, `WARNING`, `INFO`, `CONFIG`, `FINE`, `FINER` 및 `FINEST`
- *ProductName-Version*은 `sun-appserver-ee8.1`입니다.
- *LoggerName*은 로그 모듈의 소스를 식별하는 계층적 로거 이름 공간이며, 이 경우에는 `javax.enterprise.system.core`입니다.
- *Key Value Pairs*는 키 이름과 값입니다. 대개는 `_ThreadID=14;` 같은 스레드 아이디입니다.
- *Message*는 로그 메시지의 텍스트입니다. 모든 Application Server `SEVERE` 및 `WARNING` 메시지와 대부분의 `INFO` 메시지는 모듈 코드와 숫자 값(이 경우 `CORE5004`)으로 구성된 메시지 아이디로 시작합니다.

이후 릴리스에서 로그 레코드 형식이 변경되거나 향상될 수 있습니다.

사용자 정의 로그 수준 설정

이 절에서는 `java.util.logging` 패키지를 사용하고 Application Server의 로깅 하위 시스템에 액세스하는 응용 프로그램에 대한 사용자 정의 로깅 수준을 구성하는 방법에 대해 설명합니다.

`java.util.logging` 패키지는 로거 인스턴스를 만들 수 있는 계층적 이름 공간을 제공합니다. 인스턴스의 로그 파일은 특정 로깅 레코드가 Application Server에 출력되는지 여부에 상관없이 로그 레코드의 로그 수준 및 지정된 로그 수준에 따라 달라집니다.

Application Server 로거 설정 구성은 Application Server의 내부 로깅에 대한 미세 제어를 가능하게 하는 20개 이상의 로깅 모듈을 제공합니다. 또한, 모듈 이름 및 모듈이 사용해야 하는 로깅 수준을 지정함으로써 사용자 정의 추가 로그 모듈을 만들 수 있는 옵션도 있습니다.

여기서 중요한 점은 로거는 정적 이름이며 상속은 제공되지 않는다는 점입니다. 따라서, 이름이 `com.someorg.app`인 사용자 정의 로거가 구성되고 응용 프로그램에서 `com.someorg.app.submodule` 로거를 조회하는 경우 `com.someorg.app.submodule`에는 `com.someorg.app`의 설정을 상속하는 로거가 제공되지 않습니다. 대신, `com.someorg.app.submodule`은 INFO 이상의 수준으로 기록하도록 설정된 기본 로거를 갖습니다.

응용 프로그램에서 로거 상속을 사용해야 하는 경우에는 Application Server를 실행하는 데 사용하는 Java 런타임의 `logging.properties` 파일을 편집하여 이 `com.someorg.app.submodule`을 구성할 수 있습니다. 예를 들어, 다음 항목을 `logging.properties` 파일에 추가하면 `com.someorg.app.submodule`은 만들어질 때와 동일한 FINE 수준을 상속합니다.

```
com.someorg.app.level = FINE
```

Java 로깅 API에 대한 자세한 내용은 다른 `java.util.logging` 클래스 및 `http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/package-summary.html`의 Java 설명서를 참조하십시오.

로거 이름 공간 계층

Application Server는 각 모듈에 대한 로거를 제공합니다. 다음 표는 모듈 이름과 각 로거에 대한 이름 공간이 관리 콘솔의 로그 수준 페이지에 표시되는 알파벳 순서로 나열합니다. 표의 마지막 세 모듈은 로그 수준 페이지에 표시되지 않습니다.

표 15–1 Application Server 로거 이름 공간

모듈 이름	이름 공간
관리	`javax.enterprise.system.tools.admin`
클래스 로더	`javax.enterprise.system.core.classloading`
CMP	`javax.enterprise.system.container.cmp`
구성	`javax.enterprise.system.core.config`
커넥터	`javax.enterprise.resource.resourceadapter`
CORBA	`javax.enterprise.resource.corba`
배포	`javax.enterprise.system.tools.deployment`
EJB 컨테이너	`javax.enterprise.system.container.ejb`
JavaMail	`javax.enterprise.resource.javamail`
JAXR	`javax.enterprise.resource.webservices.registry`
JAX-RPC	`javax.enterprise.resource.webservices.rpc`
JDO	`javax.enterprise.resource.jdo`
JMS	`javax.enterprise.resource.jms`
JTA	`javax.enterprise.resource.jta`

표 15–1 Application Server 로거 이름 공간 *(계속)*

모듈 이름	이름 공간
JTS	javax.enterprise.system.core.transaction
MDB 컨테이너	javax.enterprise.system.container.ejb.mdb
이름 지정	javax.enterprise.system.core.naming
노드 에이전트(Enterprise Edition에만 해당)	javax.ee.enterprise.system.nodeagent
루트	javax.enterprise
SAAJ	javax.enterprise.resource.webservices.saaj
보안	javax.enterprise.system.core.security
서버	javax.enterprise.system
동기화(Enterprise Edition에만 해당)	javax.ee.enterprise.system.tools.synchronization
Util	javax.enterprise.system.util
검증자	javax.enterprise.system.tools.verifier
웹 컨테이너	javax.enterprise.system.container.web
코어	javax.enterprise.system.core
시스템 출력(System.out.println)	javax.enterprise.system.stream.out
시스템 오류(System.err.println)	javax.enterprise.system.stream.err

◆ ◆ ◆ 16장

16 구성 요소 및 서비스 모니터링

Application Server의 서버 인스턴스에 배포된 다양한 구성 요소와 서비스의 런타임 상태를 확인하려면 모니터링을 사용합니다. 런타임 구성 요소와 프로세스의 상태에 대한 정보를 사용하면 성능 조정을 목적으로 성능 병목 현상을 확인하고 용량 계획을 지원하며, 실패를 예상하고 실패 시 근본적인 원인 분석을 수행하고 모든 항목이 예상대로 기능하도록 할 수 있습니다.

모니터링을 설정하면 오버헤드가 증가하여 성능이 저하됩니다.

이 장은 다음 내용으로 구성되어 있습니다.

모니터링 개요

Application Server를 모니터하려면 다음 단계를 수행합니다.

1. 관리 콘솔이나 `asadmin` 도구를 사용하여 특정 서비스와 구성 요소의 모니터링을 활성화합니다.

   이 단계에 대한 자세한 내용은 173 페이지 "모니터링 활성화 및 비활성화"를 참조하십시오.

2. 관리 콘솔이나 `asadmin` 도구를 사용하여 특정 서비스 또는 구성 요소의 모니터링 데이터를 확인합니다.

   이 단계에 대한 자세한 내용은 174 페이지 "모니터링 데이터 보기"를 참조하십시오.

모니터링 가능한 객체의 트리 구조 정보

Application Server는 트리 구조를 사용하여 모니터링 가능한 객체를 추적합니다. 모니터링 객체의 트리가 동적이기 때문에 인스턴스에 구성 요소가 추가 업데이트 또는 제거되면 트리도 변경됩니다. 트리의 루트 객체는 서버 인스턴스 이름(예: server)입니다. Platform Edition에서는 한 개의 서버 인스턴스만 허용됩니다.

다음 명령은 트리의 최상위 수준을 표시합니다.

```
asadmin> list --user adminuser --monitor server
server.applications
server.http-service
server.connector-service
server.jms-service
server.jvm
server.orb
server.resources
server.thread-pools
```

다음 절에서는 이 하위 트리를 설명합니다.

- 150 페이지 “응용 프로그램 트리”
- 151 페이지 “HTTP 서비스 트리”
- 152 페이지 “자원 트리”
- 152 페이지 “커넥터 서비스 트리”
- 152 페이지 “JMS 서비스 트리”
- 153 페이지 “ORB 트리”
- 153 페이지 “스레드 풀 트리”

응용 프로그램 트리

다음 계통도에서는 엔터프라이즈 응용 프로그램의 다양한 구성 요소에 대한 상위 노드와 하위 노드를 표시합니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다. 자세한 내용은 154 페이지 “EJB 컨테이너 통계”를 참조하십시오.

예 16–1 응용 프로그램 노드 트리 구조

```
applications
   |--- application1
   |    |--- ejb-module-1
   |    |        |--- ejb1 *
   |    |                 |--- cache (for entity/sfsb) *
   |    |                 |--- pool (for slsb/mdb/entity) *
   |    |                 |--- methods
   |    |                      |---method1 *
   |    |                      |---method2 *
   |    |                 |--- stateful-session-store (for sfsb)*
```

예 16–1 응용 프로그램 노드 트리 구조 *(계속)*

```
|    |                 |--- timers (for s1sb/entity/mdb) *
|    |--- web-module-1
|    |        |--- virtual-server-1 *
|    |                        |---servlet1  *
|    |                        |---servlet2  *
|--- standalone-web-module-1
|    |        |----- virtual-server-2 *
|    |                        |---servlet3 *
|    |                        |---servlet4 *
|    |        |----- virtual-server-3 *
|    |                        |---servlet3 *(same servlet on different vs)
|    |                        |---servlet5 *
|--- standalone-ejb-module-1
|    |        |--- ejb2 *
|    |                 |--- cache (for entity/sfsb) *
|    |                 |--- pool (for slsb/mdb/entity) *
|    |                 |--- methods
|    |                        |--- method1 *
|    |                        |--- method2 *
|--- application2
```

HTTP 서비스 트리

다음 계통도에는 HTTP 서비스 노드가 표시됩니다. 모니터링 정보를 사용할 수 있는 노드에는 별표(*)가 표시됩니다. 159 페이지 “HTTP 서비스 통계”를 참조하십시오.

예 16–2 HTTP 서비스 계통도(Platform Edition 버전)

```
http-service
         |--- virtual-server-1
              |--- http-listener-1 *
              |--- http-listener-2 *
         |--- virtual-server-2
              |--- http-listener-1 *
              |--- http-listener-2 *
```

예 16–3 HTTP 서비스 계통도(Enterprise Edition 버전)

```
http-service *
             |---connection-queue *
             |---dns *
             |---file-cache *
             |---keep-alive *
             |---pwc-thread-pool *
             |---virtual-server-1*
```

예 16–3 HTTP 서비스 계통도(Enterprise Edition 버전) *(계속)*

```
|        |--- request *
|---virtual-server-2*
|        |--- request *
```

자원 트리

자원 노드에는 풀(예: JDBC 연결 풀이나 커넥터 연결 풀)에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 다양한 자원 구성 요소에 대한 상위 노드와 하위 노드를 보여줍니다모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다.
160 페이지 “JDBC 연결 풀 통계”를 참조하십시오.

예 16–4 자원 계통도

```
resources
        |---connection-pool1(either connector-connection-pool or jdbc)*
        |---connection-pool2(either connector-connection-pool or jdbc)*
```

커넥터 서비스 트리

커넥터 서비스 노드에는 풀(예: 커넥터 연결 풀)에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 다양한 커넥터 서비스 구성 요소에 대한 상위 노드와 하위 노드를 보여줍니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다.
161 페이지 “JMS/커넥터 서비스 통계”를 참조하십시오.

예 16–5 커넥터 서비스 계통도

```
connector-service
        |--- resource-adapter-1
        |        |-- connection-pools
        |        |        |-- pool-1 (All pool stats for this pool)
        |        |-- work-management (All work mgmt stats for this RA)
```

JMS 서비스 트리

JMS 서비스 노드에는 풀(예: 커넥터 연결 풀)에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 다양한 서비스 구성 요소에 대한 상위 노드와 하위 노드를 보여줍니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다.

예 16–6 서비스 계통도

```
jms-service
        |-- connection-factories [AKA conn. pools in the RA world]
        |        |-- connection-factory-1 (All CF stats for this CF)
        |-- work-management (All work mgmt stats for the MQ-RA)
```

ORB 트리

ORB 노드에는 연결 관리자에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 ORB 구성 요소에 대한 상위 노드와 하위 노드를 보여줍니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다. 162 페이지 “ORB의 연결 관리자용 통계”를 참조하십시오.

예 16–7 ORB 계통도

```
orb
        |--- connection-managers
        |        |--- connection-manager-1 *
        |        |--- connection-manager-1 *
```

스레드 풀 트리

스레드 풀 노드에는 연결 관리자에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 ORB 구성 요소에 대한 상위 노드와 하위 노드를 보여줍니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다. 162 페이지 “스레드 풀 통계”를 참조하십시오.

예 16–8 스레드 풀 계통도

```
thread-pools
        |    |--- thread-pool-1 *
        |    |--- thread-pool-2 *
```

모니터된 구성 요소 및 서비스에 대한 통계

이 절에서는 사용 가능한 모니터링 통계에 대해 설명합니다.

EJB 컨테이너 통계

EJB 통계는 다음 표에 설명되어 있습니다.

표 16–1 EJB 통계

속성 이름	데이터 유형	설명
createcount	CountStatistic	EJB의 create 메소드를 호출한 횟수입니다.
removecount	CountStatistic	EJB의 remove 메소드를 호출한 횟수입니다.
pooledcount	RangeStatistic	풀링된 상태의 Entity Bean 수입니다.
readycount	RangeStatistic	준비 상태의 Entity Bean 수입니다.
messagecount	CountStatistic	Message-Driven Bean에 대해 수신한 메시지 수입니다.
methodreadycount	RangeStatistic	MethodReady 상태의 Stateful 또는 Stateless Session Bean 수입니다.
passivecount	RangeStatistic	Passive 상태의 Stateful Session Bean 수입니다.

EJB 메소드 호출에 사용할 수 있는 통계는 다음 표에 나열되어 있습니다.

표 16–2 EJB 메소드 통계

속성 이름	데이터 유형	설명
methodstatistic	TimeStatistic	작업을 호출한 횟수와 호출에 소요된 전체 시간 등입니다.
totalnumerrors	CountStatistic	메소드 실행 결과 예외가 발생한 횟수입니다. EJB 컨테이너에 대해 모니터링을 사용하는 경우 Stateless 및 Stateful Session Bean 또는 Entity Bean에 대해 이 통계를 수집합니다.
totalnumsuccess	CountStatistic	메소드가 성공적으로 실행된 횟수입니다. EJB 컨테이너에 대해 모니터링을 사용하는 경우 Stateless 및 Stateful Session Bean 또는 Entity Bean에 대해 이 통계를 수집합니다.

표 16–2 EJB 메소드 통계 *(계속)*

속성 이름	데이터 유형	설명
executiontime	CountStatistic	작업을 실행하기 위한 마지막 성공 실패 시도에 대해 메소드를 실행하는 데 걸린 시간(밀리초)입니다. EJB 컨테이너에 대해 모니터링을 사용하는 경우 Stateless 및 Stateful Session Bean 또는 Entity Bean에 대해 이 통계를 수집합니다.

EJB 세션 저장소에 대한 통계는 다음 표에 나열되어 있습니다.

표 16–3 EJB 세션 저장소 통계

속성 이름	데이터 유형	설명
currentSize	RangeStatistic	현재 저장소에 있는 비활성화된 세션이나 검사점이 지정된 세션 수입니다.
activationCount	CountStatistic	저장소에서 활성화된 세션 수입니다.
activationSuccessCount	CountStatistic	저장소에서 성공적으로 활성화된 세션 수입니다.
activationErrorCount	CountStatistic	작업을 실행하기 위한 마지막 성공 실패 시도에 대해 메소드를 실행하는 데 걸린 시간(밀리초)입니다. EJB 컨테이너에 대해 모니터링을 사용하는 경우 Stateless 및 Stateful Session Bean 또는 Entity Bean에 대해 이 통계를 수집합니다.
passivationCount	CountStatistic	이 저장소를 사용하여 비활성화된 세션 수입니다.
passivationSuccessCount	CountStatistic	이 저장소를 사용하여 성공적으로 비활성화된 세션 수입니다.
passivationErrorCount	CountStatistic	이 저장소를 사용하여 비활성화할 수 없는 세션 수입니다.
expiredSessionCount	CountStatistic	이 저장소에서 제거한 만료된 세션 수입니다.
passivatedBeanSize	CountStatistic	전체, 최소 및 최대를 포함하여 이 저장소에서 비활성화된 전체 바이트 수입니다.

표 16–3 EJB 세션 저장소 통계 *(계속)*

속성 이름	데이터 유형	설명
passivationTime	CountStatistic	전체, 최소 및 최대를 포함하여 Bean을 저장소로 비활성화하는 데 걸린 시간입니다.
checkpointCount (EE 전용)	CountStatistic	이 저장소를 사용하여 검사점이 지정된 세션 수입니다.
checkpointSuccessCount (EE 전용)	CountStatistic	검사점이 성공적으로 지정된 세션 수입니다.
checkpointErrorCount (EE 전용)	CountStatistic	검사점을 지정할 수 없는 세션 수입니다.
checkpointedBeanSize (EE 전용)	ValueStatistic	저장소에서 검사점을 지정한 Bean의 총 개수입니다.
checkpointTime (EE 전용)	TimeStatistic	Bean을 저장소로 검사점을 지정하는 데 걸린 시간입니다.

EJB 풀에 사용할 수 있는 통계는 다음 표에 나열되어 있습니다.

표 16–4 EJB 풀 통계

속성 이름	데이터 유형	설명
numbeansinpool	BoundedRangeStatistic	풀 변경 방법에 대한 정보를 제공하는 연관된 풀의 EJB 수입니다.
numthreadswaiting	BoundedRangeStatistic	사용 가능한 Bean을 기다리는 스레드 수로 요청이 정체될 수 있음을 나타냅니다.
totalbeanscreated	CountStatistic	데이터 수집이 시작된 후 연관된 풀에서 만들어진 Bean 수입니다.
totalbeansdestroyed	CountStatistic	데이터 수집이 시작된 후 연관된 풀에서 삭제된 수입니다.
jmsmaxmessagesload	CountStatistic	Message-driven Bean을 위해 JMS 세션에 한 번에 로드하는 최대 메시지 수입니다. 기본값은 1이며, Message-driven Bean의 풀에만 적용됩니다.

EJB 캐시에 사용할 수 있는 통계는 다음 표에 나열되어 있습니다.

표 16–5 EJB 캐시 통계

속성 이름	데이터 유형	설명
cachemisses	BoundedRangeStatistic	사용자 요청 시 캐시에서 Bean을 찾지 못한 횟수입니다.
cachehits	BoundedRangeStatistic	사용자 요청 시 캐시에서 항목을 찾은 횟수입니다.
numbeansincache	BoundedRangeStatistic	캐시에 있는 Bean의 수입니다. 이 값은 캐시의 현재 크기입니다.
numpassivations	CountStatistic	비활성화된 Bean의 수입니다. Stateful Session Bean에만 적용됩니다.
numpassivationerrors	CountStatistic	비활성화 처리를 수행하는 동안 발생한 오류 횟수입니다. Stateful Session Bean에만 적용됩니다.
numexpiredsessionsremoved	CountStatistic	정리 스레드로 제거된 만료된 세션의 수입니다. Stateful Session Bean에만 적용됩니다.
numpassivationsuccess	CountStatistic	비활성화 처리가 성공적으로 완료된 횟수입니다. Stateful Session Bean에만 적용됩니다.

타이머에 사용할 수 있는 통계는 다음 표에 나열되어 있습니다.

표 16–6 타이머 통계

통계	데이터 유형	설명
numtimerscreated	CountStatistic	시스템에서 만들어진 타이머 수입니다.
numtimersdelivered	CountStatistic	시스템에서 전달한 타이머 수입니다.
numtimersremoved	CountStatistic	시스템에서 제거한 타이머 수입니다.

웹 컨테이너 통계

웹 컨테이너는 150 페이지 “응용 프로그램 트리”에 제시된 객체 트리와 일치합니다. 모든 개별 웹 응용 프로그램에 대한 웹 컨테이너 통계가 표시됩니다. 서블릿의 웹 컨테이너에 사용할 수 있는 통계는 157 페이지 “웹 컨테이너 통계”에 표시되고 웹 모듈에 사용할 수 있는 통계는 157 페이지 “웹 컨테이너 통계”에 표시됩니다.

표 16–7 웹 컨테이너(서블릿) 통계

통계	단위	데이터 유형	설명
errorcount	수	CountStatistic	응답 코드가 400보다 크거나 같은 경우의 누적 수입니다.
maxtime	밀리초	CountStatistic	웹 컨테이너가 요청을 기다리는 최대 시간입니다.
processingtime	밀리초	CountStatistic	각 요청을 처리하는 데 필요한 시간의 누적 값입니다. 처리 시간은 요청 처리 시간을 요청 수로 나눈 평균 값입니다.
requestcount	수	CountStatistic	지금까지 처리한 총 요청 수입니다.

웹 모듈에 사용할 수 있는 통계는 157 페이지 “웹 컨테이너 통계”에 표시됩니다.

표 16–8 웹 컨테이너(웹 모듈) 통계

통계	데이터 유형	설명
jspcount	CountStatistic	웹 모듈에 로드된 JSP 페이지 수입니다.
jspreloadcount	CountStatistic	웹 모듈에 다시 로드된 JSP 페이지 수입니다.
sessionstotal	CountStatistic	웹 모듈에 대해 만들어진 총 세션 수입니다.
activesessionscurrent	CountStatistic	웹 모듈에 대해 현재 활성화된 세션 수입니다.
activesessionshigh	CountStatistic	웹 모듈에 대해 동시에 활성화된 최대 세션 수입니다.
rejectedsessionstotal	CountStatistic	웹 모듈에 대해 거부된 총 세션 수입니다. 허용된 최대 세션 수가 활성화되었기 때문에 만들어지지 않은 세션 수입니다.
expiredsessionstotal	CountStatistic	웹 모듈에 대해 만료된 총 세션 수입니다.
sessionsize (EE 전용)	AverageRangeStatistic	웹 모듈에 대한 세션 크기입니다. 값은 높음, 낮음 또는 평균이거나 일련화된 세션의 경우 바이트입니다.
containerlatency (EE 전용)	AverageRangeStatistic	전체 대기 시간 요청 중에서 웹 컨테이너 부분에 대한 대기 시간입니다. 값은 높음, 낮음 또는 평균입니다.

표 16–8 웹 컨테이너(웹 모듈) 통계 (계속)

통계	데이터 유형	설명
sessionpersisttime (EE 전용)	AverageRangeStatistic	웹 모듈의 백엔드 저장소에 HTTP 세션 상태를 지속시키는 데 걸린 시간(낮음, 높음 또는 평균)입니다.
cachedsessionscurrent (EE 전용)	CountStatistic	웹 모듈의 메모리에 현재 캐시된 세션 수입니다.
passivatedsessionscurrent (EE 전용)	CountStatistic	웹 모듈에 대해 현재 비활성화된 세션 수입니다.

HTTP 서비스 통계

HTTP 서비스에 사용할 수 있는 통계는 159 페이지 “HTTP 서비스 통계”에 표시됩니다. 이 통계는 Platform Edition에만 적용됩니다. Enterprise Edition의 HTTP 서비스에 대한 통계는 167 페이지 “PWC(Production Web Container) 통계”를 참조하십시오.

표 16–9 HTTP 서비스 통계(Platform Edition에만 적용)

통계	단위	데이터 유형	설명
bytesreceived	바이트	CountStatistic	각 요청 프로세서에서 수신한 누적 바이트 값입니다.
bytessent	바이트	CountStatistic	각 요청 프로세서에서 전송한 누적 바이트 값입니다.
currentthreadcount	수	CountStatistic	Listener 스레드 풀에서 현재 처리 중인 스레드 수입니다.
currentthreadsbusy	수	CountStatistic	요청을 처리하는 Listener 스레드 풀에서 현재 사용 중인 요청 처리 스레드 수입니다.
errorcount	수	CountStatistic	오류 누적 수는 응답 코드가 400보다 크거나 같은 경우의 수를 나타냅니다.
maxsparethreads	수	CountStatistic	존재할 수 있는 사용하지 않은 응답 처리 스레드의 최대 수입니다.
minsparethreads	수	CountStatistic	존재할 수 있는 사용하지 않은 응답 처리 스레드의 최소 수입니다.
maxthreads	수	CountStatistic	Listener에서 만든 요청 처리 스레드의 최대 수입니다.
maxtime	밀리초	CountStatistic	처리 스레드에 대한 최대 시간입니다.

표 16–9 HTTP 서비스 통계(Platform Edition에만 적용) *(계속)*

통계	단위	데이터 유형	설명
processing-time	밀리초	CountStatistic	각 요청을 처리하는 데 걸린 시간의 누적 값입니다. 처리 시간은 요청 처리 시간을 요청 수로 나눈 평균 값입니다.
request-count	수	CountStatistic	지금까지 처리한 총 요청 수입니다.

JDBC 연결 풀 통계

성능을 측정하고 런타임 시 JDBC 자원 사용을 수집하기 위해 자원을 모니터합니다. JDBC 연결을 만들면 부담이 크고 응용 프로그램의 성능 병목 상태를 자주 일으키기 때문에 JDBC 연결 풀에서 새로운 연결을 해제 및 작성하는 방법과 많은 스레드가 특정 풀에서 연결을 검색하기 위해 대기하는 방법을 모니터하는 것이 중요합니다.

JDBC 연결 풀에 사용할 수 있는 통계는 다음 표에 표시됩니다.

표 16–10 JDBC 연결 풀 통계

통계	단위	데이터 유형	설명
numconnfailedvalidation	수	CountStatistic	시작 시간 이후 마지막 샘플 시간까지 연결 풀에서 검증에 실패한 총 연결 수입니다.
numconnused	수	RangeStatistic	연결 사용 통계를 제공합니다. 현재 사용하고 있는 총 연결 수 외에 사용한 최대 연결 수(고수위 표시)에 대한 정보도 제공합니다.
numconnfree	수	RangeStatistic	마지막 샘플링 시에 풀에서 사용 가능한 총 연결 수입니다.
numconntimedout	수	BoundedRangeStatistic	시작 시간과 마지막 샘플 시간 사이에 시간 초과된 풀의 총 연결 수입니다.
averageconnwaittime	수	CountStatistic	커넥터 연결 풀에 대한 연결 요청의 평균 연결 대기 시간을 나타냅니다.
waitqueuelength	수	CountStatistic	대기열에서 처리를 기다리는 연결 요청 수입니다.
connectionrequestwaittime		RangeStatistic	연결 요청의 가장 긴 대기 시간과 가장 짧은 대기 시간입니다. 현재 값은 풀에서 처리된 마지막 요청의 대기 시간을 나타냅니다.
numconncreated	밀리초	CountStatistic	마지막 재설정 후 만들어진 물리적 연결 수입니다.

표 16–10 JDBC 연결 풀 통계 *(계속)*

통계	단위	데이터 유형	설명
numconndestroyed	수	CountStatistic	마지막 재설정 후 삭제된 물리적 연결 수입니다.
numconnacquired	수	CountStatistic	풀에서 얻은 논리적 연결 수입니다.
numconnreleased	수	CountStatistic	풀에 해제된 논리적 연결 수입니다.

JMS/커넥터 서비스 통계

커넥터 연결 풀에 사용할 수 있는 통계는 161 페이지 “JMS/커넥터 서비스 통계”에 표시됩니다. 커넥터 작업 관리에 대한 통계는 161 페이지 “JMS/커넥터 서비스 통계”에 표시됩니다.

표 16–11 커넥터 연결 풀 통계

통계	단위	데이터 유형	설명
numconnfailedvalidation	수	CountStatistic	시작 시간 이후 마지막 샘플 시간까지 연결 풀에서 검증에 실패한 총 연결 수입니다.
numconnused	수	RangeStatistic	연결 사용 통계를 제공합니다. 현재 사용하고 있는 총 연결 수 외에 사용한 최대 연결 수(고수위 표시)에 대한 정보도 제공합니다.
numconnfree	수	RangeStatistic	마지막 샘플링 시에 풀에서 사용 가능한 총 연결 수입니다.
numconntimedout	수	CountStatistic	시작 시간과 마지막 샘플 시간 사이에 시간 초과된 풀의 총 연결 수입니다.
averageconnwaittime	수	CountStatistic	연결 풀에서 처리할 때까지 연결의 평균 대기 시간입니다.
waitqueuelenght	수	CountStatistic	대기열에서 처리를 기다리는 연결 요청 수입니다.
connectionrequestwaittime		RangeStatistic	연결 요청의 가장 긴 대기 시간과 가장 짧은 대기 시간입니다. 현재 값은 풀에서 처리된 마지막 요청의 대기 시간을 나타냅니다.
numconncreated	밀리초	CountStatistic	마지막 재설정 후 만들어진 물리적 연결 수입니다.

표 16–11 커넥터 연결 풀 통계 *(계속)*

통계	단위	데이터 유형	설명
numconndestroyed	수	CountStatistic	마지막 재설정 후 삭제된 물리적 연결 수입니다.
numconnacquired	수	CountStatistic	풀에서 얻은 논리적 연결 수입니다.
numconnreleased	수	CountStatistic	풀에 해제된 논리적 연결 수입니다.

커넥터 작업 관리에 사용할 수 있는 통계는 161 페이지 “JMS/커넥터 서비스 통계”에 나열되어 있습니다.

표 16–12 커넥터 작업 관리 통계

통계	데이터 유형	설명
activeworkcount	RangeStatistic	커넥터에서 실행한 작업 객체 수입니다.
waitqueuelength	RangeStatistic	실행하기 전에 대기열에서 대기 중인 작업 객체 수입니다.
workrequestwaittime	RangeStatistic	실행되기 전 작업 객체의 가장 긴 대기 및 가장 짧은 대기 시간입니다.
submittedworkcount	CountStatistic	커넥터 모듈에서 제출한 작업 객체 수입니다.
rejectedworkcount	CountStatistic	Application Server에서 거부한 작업 객체 수입니다.
completedworkcount	CountStatistic	완료한 작업 객체 수입니다.

ORB의 연결 관리자용 통계

ORB의 연결 관리자에 사용할 수 있는 통계는 162 페이지 “ORB의 연결 관리자용 통계”에 나열되어 있습니다.

표 16–13 ORB의 연결 관리자 통계

통계	단위	데이터 유형	설명
connectionsidle	수	CountStatistic	ORB에 대해 유휴인 총 연결 수를 제공합니다.
connectionsinuse	수	CountStatistic	ORB에 대해 사용 중인 총 연결 수를 제공합니다.
totalconnections	수	BoundedRangeStatistic	ORB에 대한 총 연결 수입니다.

스레드 풀 통계

스레드 풀에 사용할 수 있는 통계는 다음 표에 표시됩니다.

표 16–14 스레드 풀 통계

통계	단위	데이터 유형	설명
averagetimeinqueue	밀리초	RangeStatistic	처리하기 전에 대기열에서 요청이 대기한 평균 시간(밀리초)입니다.
averageworkcompletion-time	밀리초	RangeStatistic	할당을 완료하는 데 걸린 평균 시간(밀리초)입니다.
currentnumberofthreads	수	BoundedRangeStatistic	현재 스레드를 처리하는 요청 수입니다.
numberofavailablethreads	수	CountStatistic	사용 가능한 스레드 수입니다.
numberofbusythreads	수	CountStatistic	사용 중인 스레드 수입니다.
totalworkitemsadded	수	CountStatistic	지금까지 작업 대기열에 추가된 총 작업 항목 수입니다.

트랜잭션 서비스 통계

클라이언트는 트랜잭션 서비스를 통해 트랜잭션 하위 시스템을 중단하여 트랜잭션을 롤백하고 중단 시에 처리 중인 트랜잭션을 확인할 수 있습니다. 트랜잭션 서비스에 사용할 수 있는 통계는 다음 표에 표시됩니다.

표 16–15 트랜잭션 서비스 통계

통계	데이터 유형	설명
activecount	CountStatistic	현재 활성화된 트랜잭션 수입니다.
activeids	StringStatistic	현재 활성화된 트랜잭션의 아이디입니다. 이러한 각 트랜잭 트랜잭션 서비스를 고정한 후 롤백할 수 있습니다.
committedcount	CountStatistic	완결된 트랜잭션 수입니다.
rolledbackcount	CountStatistic	롤백된 트랜잭션 수입니다.
state	StringStatistic	트랜잭션이 고정되었는지 여부를 나타냅니다.

Java Virtual Machine(JVM) 통계

JVM에는 항상 활성화되어 있는 모니터링 가능한 속성이 있습니다. JVM에 사용할 수 있는 통계는 다음 표에 표시됩니다.

표 16–16 JVM 통계

통계	데이터 유형	설명
heapsize	BoundedRangeStatistic	상주 메모리 범위는 JVM 메모리 힙 크기의 상한 및 하한입니다.
uptime	CountStatistic	JVM이 실행되고 있는 시간입니다.

J2SE 5.0의 JVM 통계

Application Server를 J2SE 5.0 버전 이상에서 실행되도록 구성한 경우 JVM에서 추가 모니터링 정보를 얻을 수 있습니다. 이 추가 정보 표시를 활성화하려면 모니터링 수준을 낮음으로 설정합니다. 시스템의 라이브 스레드에 관련된 정보도 표시하려면 모니터링 수준을 높음으로 설정합니다. J2SE 5.0에서 사용 가능한 추가 모니터링 기능에 대한 자세한 내용은 http://java.sun.com/j2se/1.5.0/docs/guide/management/의 *Monitoring and Management for the Java Platform* 설명서를 참조하십시오.

J2SE 5.0 모니터링 도구는 http://java.sun.com/j2se/1.5.0/docs/tooldocs/#manage에서 설명합니다.

J2SE 5.0의 JVM에서 로딩한 클래스에 사용할 수 있는 통계는 164 페이지 “J2SE 5.0의 JVM 통계”에 표시됩니다.

표 16–17 J2SE 5.0용 JVM 통계 - 클래스 로딩

통계	데이터 유형	설명
loadedclasscount	CountStatistic	현재 JVM에 로드된 클래스 수입니다.
totalloadedclasscount	CountStatistic	JVM에서 실행을 시작한 후 로드된 총 클래스 수입니다.
unloadedclasscount	CountStatistic	JVM에서 실행을 시작한 후 JVM에서 언로드된 클래스 수입니다.

J2SE 5.0의 JVM에서 컴파일에 사용할 수 있는 통계는 164 페이지 “J2SE 5.0의 JVM 통계”에 표시됩니다.

표 16–18 J2SE 5.0용 JVM 통계 - 컴파일

통계	데이터 유형	설명
totalcompilationtime	CountStatistic	컴파일에 소비된 누적 시간(밀리초)입니다.

J2SE 5.0의 JVM에서 가비지 컬렉션에 사용할 수 있는 통계는 164 페이지 “J2SE 5.0의 JVM 통계”에 표시됩니다.

표 16–19 J2SE 5.0용 JVM 통계 - 가비지 컬렉션

통계	데이터 유형	설명
collectioncount	CountStatistic	발생한 총 모음 수입니다.
collectiontime	CountStatistic	누적된 모음 시간(밀리초)입니다.

J2SE 5.0의 JVM에서 메모리에 사용할 수 있는 통계는 164 페이지 “J2SE 5.0의 JVM 통계”에 표시됩니다.

표 16–20 J2SE 5.0용 JVM 통계 - 메모리

통계	데이터 유형	설명
objectpending finalizationcount	CountStatistic	완성을 보류 중인 대략적인 객체 수입니다.
initheapsize	CountStatistic	JVM에서 처음 요청한 힙 크기입니다.
usedheapsize	CountStatistic	현재 사용 중인 힙 크기입니다.
maxheapsize	CountStatistic	메모리 관리를 위해 사용할 수 있는 최대 메모리 양(바이트)이
committedheapsize	CountStatistic	JVM에서 사용하기 위해 완결한 메모리 양(바이트)입니다.
initnonheapsize	CountStatistic	JVM에서 처음 요청한 힙이 아닌 영역의 크기입니다.
usednonheapsize	CountStatistic	현재 사용 중인 힙이 아닌 영역의 크기입니다.
maxnonheapsize	CountStatistic	메모리 관리를 위해 사용할 수 있는 최대 메모리 양(바이트)이
committednonheapsize	CountStatistic	JVM에서 사용하기 위해 완결한 메모리 양(바이트)입니다.

J2SE 5.0의 JVM에서 운영 체제에 사용할 수 있는 통계는 164 페이지 “J2SE 5.0의 JVM 통계”에 표시됩니다.

표 16–21 J2SE 5.0용 JVM 통계 - 운영 체제

통계	데이터 유형	설명
arch	StringStatistic	운영 체제 구조
availableprocessors	CountStatistic	JVM에 사용 가능한 프로세서 수입니다.
name	StringStatistic	운영 체제 이름입니다.
version	StringStatistic	운영 체제 버전입니다.

J2SE 5.0의 JVM에서 런타임에 사용할 수 있는 통계는 164 페이지 “J2SE 5.0의 JVM 통계”에 표시됩니다.

표 16–22 J2SE 5.0용 JVM 통계 - 런타임

통계	데이터 유형	설명
name	StringStatistic	실행 중인 JVM을 나타내는 이름입니다.
vmname	StringStatistic	JVM 구현 이름입니다.
vmvendor	StringStatistic	JVM 구현 공급업체입니다.
vmversion	StringStatistic	JVM 구현 버전입니다.
specname	StringStatistic	JVM 사양 이름입니다.
specvendor	StringStatistic	JVM 사양 공급업체입니다.
specversion	StringStatistic	JVM 사양 버전입니다.
management specversion	StringStatistic	관리 사양입니다. JVM에서 구현되는 버전입니다.
classpath	StringStatistic	클래스 파일을 검색하기 위해 시스템 클래스 로더에서 사용하는 클래스 경로입니다.
librarypath	StringStatistic	Java 라이브러리 경로입니다.
bootclasspath	StringStatistic	클래스 파일을 검색하기 위해 부트스트랩 클래스 로더에서 사용하는 클래스 경로입니다.
inputarguments	StringStatistic	JVM에 전달된 입력 인수입니다. main 메소드에 대한 인수를 포함하지 않습니다.
uptime	CountStatistic	JVM의 가동 시간(밀리초)입니다.

J2SE 5.0의 JVM에서 ThreadInfo에 사용할 수 있는 통계는 164 페이지 “J2SE 5.0의 JVM 통계”에 표시됩니다.

표 16–23 J2SE 5.0용 JVM 통계 - 스레드 정보

통계	데이터 유형	설명
threadid	CountStatistic	스레드의 아이디입니다.
threadname	StringStatistic	스레드의 이름입니다.
threadstate	StringStatistic	스레드의 상태입니다.
blockedtime	CountStatistic	스레드가 BLOCKED 상태가 된 후 경과된 시간(밀리초)입니다. 스레드 경합 모니터링이 비활성화된 경우 -1을 반환합니다.
blockedcount	CountStatistic	스레드가 BLOCKED 상태가 된 총 횟수입니다.
waitedtime	CountStatistic	스레드가 WAITING 상태를 지속한 경과 시간(밀리초)입니다. 스레드 경합 모니터링이 비활성화된 경우 -1을 반환합니다.

표 16–23 J2SE 5.0용 JVM 통계 - 스레드 정보 (계속)

통계	데이터 유형	설명
waitedcount	CountStatistic	스레드가 WAITING 또는 TIMED_WAITING 상태가 된 총 횟수입니
lockname	StringStatistic	스레드의 입력이 차단되거나 Object.wait 메소드를 통해 통지 기다리는 모니터 잠금을 나타내는 문자열입니다.
lockownerid	CountStatistic	이 스레드가 차단되는 객체의 모니터 잠금을 수용하는 스레드 아이디입니다.
lockownername	StringStatistic	이 스레드가 차단되는 객체의 모니터 잠금을 수용하는 스레드 이름입니다.
stacktrace	StringStatistic	이 스레드와 연관된 스택 추적입니다.

J2SE 5.0의 JVM에서 스레드에 사용할 수 있는 통계는 164 페이지 “J2SE 5.0의 JVM 통계”에 표시됩니다.

표 16–24 J2SE 5.0용 JVM 통계 - 스레드

통계	데이터 유형	설명
threadcount	CountStatistic	라이브 데몬과 데몬이 아닌 스레드의 현재 수입니다.
peakthreadcount	CountStatistic	JVM이 시작되거나 최대값을 다시 설정한 후 라이브 스레드 수
totalstartedthreadcount	CountStatistic	JVM을 시작한 후 만들거나 시작한 총 스레드 수입니다.
daemonthreadcount	CountStatistic	라이브 데몬 스레드의 현재 수입니다.
allthreadids	StringStatistic	모든 라이브 스레드 아이디 목록입니다.
currentthreadcputime	CountStatistic	CPU 시간 측정을 활성화한 경우 현재 스레드의 CPU 시간(나노초)입니다. CPU 시간 측정을 비활성화한 경우 -1을 반환합니다.
monitordeadlockedthreads	StringStatistic	모니터가 교착 상태인 스레드 아이디 목록입니다.

PWC(Production Web Container) 통계

통계는 Application Server의 EE(Enterprise Edition)에 있는 다음의 PWC 구성 요소와 서비스에 사용할 수 있습니다.

- 167 페이지 “PWC(Production Web Container) 통계”, PWC 가상 서버
- 167 페이지 “PWC(Production Web Container) 통계”, PWC 요청
- 167 페이지 “PWC(Production Web Container) 통계”, PWC 파일 캐시
- 167 페이지 “PWC(Production Web Container) 통계”, PWC 연결 유지
- 167 페이지 “PWC(Production Web Container) 통계”, PWC DNS

- 167 페이지 “PWC(Production Web Container) 통계”, PWC 스레드 풀
- 167 페이지 “PWC(Production Web Container) 통계”, PWC 연결 대기열
- 167 페이지 “PWC(Production Web Container) 통계”, PWC HTTP 서비스

PWC 가상 서버에 대한 통계는 167 페이지 “PWC(Production Web Container) 통계”에 나열되어 있습니다.

표 16–25 PWC 가상 서버 통계(EE 전용)

속성 이름	데이터 유형	설명
id	StringStatistic	가상 서버의 아이디입니다.
mode	StringStatistic	가상 서버의 모드입니다. 옵션에는 unknown 또는 active가 있습니다.
hosts	StringStatistic	이 가상 서버에서 처리한 호스트 이름입니다.
interfaces	StringStatistic	가상 서버가 구성된 인터페이스(Listener) 유형입니다.

PWC 요청에 사용할 수 있는 통계는 다음 표에 나열되어 있습니다.

표 16–26 PWC 요청 통계(EE 전용)

속성 이름	데이터 유형	설명
method	StringStatistic	요청에 사용되는 메소드입니다.
uri	StringStatistic	사용한 마지막 URI입니다.
countrequests	CountStatistic	사용한 요청 수입니다.
countbytestransmitted	CountStatistic	전송된 바이트 수입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
countbytesreceived	CountStatistic	수신된 바이트 수입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
ratebytesreceived	CountStatistic	서버에서 정의한 시간 간격 동안 데이터가 수신된 속도입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
maxbytestransmissionrate	CountStatistic	서버에서 정의한 시간 간격 동안 데이터가 전송된 최대 속도입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
countopenconnections	CountStatistic	현재 열려 있는 연결의 수입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.

표 16–26 PWC 요청 통계(EE 전용) *(계속)*

속성 이름	데이터 유형	설명
maxopenconnections	CountStatistic	동시에 열려 있는 연결의 최대 수입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
count2xx	CountStatistic	코드 2XX의 총 응답 수입니다.
count3xx	CountStatistic	코드 3XX의 총 응답 수입니다.
count4xx	CountStatistic	코드 4XX의 총 응답 수입니다.
count5xx	CountStatistic	코드 5XX의 총 응답 수입니다.
countother	CountStatistic	다른 응답 코드를 가진 총 응답 수입니다.
count200	CountStatistic	코드 200의 총 응답 수입니다.
count302	CountStatistic	코드 302의 총 응답 수입니다.
count304	CountStatistic	코드 304의 총 응답 수입니다.
count400	CountStatistic	코드 400의 총 응답 수입니다.
count401	CountStatistic	코드 401의 총 응답 수입니다.
count403	CountStatistic	코드 403의 총 응답 수입니다.
count404	CountStatistic	코드 404의 총 응답 수입니다.
count503	CountStatistic	코드 503의 총 응답 수입니다.

캐시 정보 절에서는 파일 캐시 사용 방법에 대한 정보를 제공합니다. PWC 파일 캐시에 대한 통계는 다음 표에 나열되어 있습니다.

표 16–27 PWC 파일 캐시 통계(EE 전용)

속성 이름	데이터 유형	설명
flagenabled	CountStatistic	파일 캐시가 활성화되어 있는지 여부를 나타냅니다. 유효한 값은 n 0 또는 yes에 대해 1입니다.
secondsmaxage	CountStatistic	유효한 캐시 항목의 최대 사용 시간(초)입니다.
countentries	CountStatistic	현재 캐시 항목 수입니다. 단일 캐시 항목은 단일 URI를 나타냅니다
maxentries	CountStatistic	동시 캐시 항목의 최대 수입니다.
countopenentries	CountStatistic	열려 있는 파일과 관련된 항목 수입니다.
maxopenentries	CountStatistic	열려있는 파일과 관련된 동시 캐시 항목의 최대 수입니다.

표 16–27 PWC 파일 캐시 통계(EE 전용) *(계속)*

속성 이름	데이터 유형	설명
sizeheapcache	CountStatistic	캐시 내용에 사용되는 힙 공간입니다.
maxheapcachesize	CountStatistic	캐시 파일 내용에 사용되는 최대 힙 공간입니다.
sizemmapcache	CountStatistic	메모리 매핑된 파일 내용에서 사용하는 주소 공간 양입니다.
maxmmapcachesize	CountStatistic	파일 캐시에서 메모리 매핑된 파일 내용에 사용하는 최대 주소 공간 양입니다.
counthits	CountStatistic	성공한 캐시 조회 수입니다.
countmisses	CountStatistic	실패한 캐시 조회 수입니다.
countinfohits	CountStatistic	성공한 파일 정보 조회 횟수입니다.
countinfomisses	CountStatistic	캐시된 파일 정보의 오류 수입니다.
countcontenthits	CountStatistic	캐시된 파일 내용의 적중 횟수입니다.
countcontentmisses	CountStatistic	파일 정보 조회가 실패한 횟수입니다.

이 절에서는 서버의 HTTP 수준 연결 유지 시스템에 대한 정보를 제공합니다. PWC 연결 유지에 사용할 수 있는 통계는 다음 표에 나열되어 있습니다.

표 16–28 PWC 연결 유지 통계(EE 전용)

속성 이름	데이터 유형	설명
countconnections	CountStatistic	연결 유지 모드의 연결 수입니다.
maxconnections	CountStatistic	연결 유지 모드에서 동시에 허용되는 최대 연결 수입니다.
counthits	CountStatistic	후속적으로 유효한 요청이 이루어진 연결 유지 모드의 총 연결 횟수입니다
countflushes	CountStatistic	서버가 연결 유지 연결을 닫은 횟수입니다.
countrefusals	CountStatistic	너무 많은 지속성 연결 때문에 연결 유지 스레드에 서버가 연결을 분배할 없는 횟수입니다.
counttimeouts	CountStatistic	클라이언트 연결이 작업 없이 시간 만료되었기 때문에 서버에서 연결 유지 연결을 종료한 횟수입니다.
secondstimeout	CountStatistic	유휴 연결 유지 연결을 닫기 전의 시간(초)입니다.

DNS 캐시가 IP 주소와 DNS 이름을 캐시합니다. 서버의 DNS 캐시는 기본적으로 비활성화되어 있습니다. 단일 캐시 항목은 단일 IP 주소나 DNS 이름 조회를 나타냅니다. PWC DNS에 사용할 수 있는 통계는 다음 표에 나열되어 있습니다.

표 16–29 PWC DNS 통계(EE 전용)

속성 이름	데이터 유형	설명
flagcacheenabled	CountStatistic	DNS 캐시가 활성화되어 있는지 여부를 표시합니다. 0(해제) 또는 1(설정) 중 하나입니다.
countcacheentries	CountStatistic	현재 캐시에 있는 DNS 항목의 수입니다.
maxcacheentries	CountStatistic	캐시에서 수용할 수 있는 최대 DNS 항목 수입니다.
countcachehits	CountStatistic	DNS 캐시 조회가 성공한 횟수입니다.
countcachemisses	CountStatistic	DNS 캐시 조회가 실패한 횟수입니다.
flagasyncenabled	CountStatistic	비동기 DNS 조회 활성화(on) 여부를 나타냅니다. 0(해제) 또는 1(설정) 중 하나입니다.
countasyncnamelookups	CountStatistic	비동기 DNS 이름 조회 총 수입니다.
countasyncaddrlookups	CountStatistic	비동기 DNS 주소 조회 총 수입니다.
countasynclookupsinprogress	CountStatistic	처리 중인 비동기 조회 수입니다.

PWC 스레드 풀에 대한 통계는 다음 표에 나열되어 있습니다.

표 16–30 PWC 스레드 풀 통계(EE 전용)

속성 이름	데이터 유형	설명
id	StringStatistic	스레드 풀의 아이디입니다.
countthreadsidle	CountStatistic	현재 유휴 상태인 요청 처리 스레드의 수입니다.
countthreads	CountStatistic	현재 요청 처리 스레드의 수입니다.
maxthreads	CountStatistic	동시에 존재할 수 있는 요청 처리 스레드의 최대 수입니다.
countqueued	CountStatistic	이 스레드 풀에서 처리하기 위해 대기열에 있는 요청 수입니다.
peakqueued	CountStatistic	대기열에 동시에 존재하는 최대 요청 수입니다.
maxqueued	CountStatistic	대기열에 한 번에 있을 수 있는 최대 요청 수입니다.

연결 대기열은 요청이 처리되기 전에 대기하는 대기열입니다. 연결 대기열에 대한 통계에서는 대기열의 세션 수 연결이 승인되기 전의 평균 지연을 나타냅니다. PWC 연결 대기열에 대한 통계는 다음 표에 나열되어 있습니다.

표 16–31 PWC 연결 대기열 통계(EE 전용)

속성 이름	데이터 유형	설명
id	StringStatistic	연결 대기열의 아이디입니다.
counttotalconnections	CountStatistic	승인된 총 연결 수입니다.
countqueued	CountStatistic	현재 대기열에 있는 연결 수입니다.
peakqueued	CountStatistic	대기열에 동시에 존재하는 최대 연결 수입니다.
maxqueued	CountStatistic	연결 대기열의 최대 크기입니다.
countoverflows	CountStatistic	대기열이 가득 차서 연결을 수용할 수 없었던 횟수입니다.
counttotalqueued	CountStatistic	대기열에 있던 연결의 총 수입니다. 특정 연결이 여러 번 대기열에 있을 수 있으므로 counttotalqueued 값은 counttotalconnections보다 크거나 같을 수 있습니다.
tickstotalqueued	CountStatistic	연결이 대기열에서 보낸 시간의 총 눈금 수입니다. 눈금은 시스템 종속 시간 단위입니다.
countqueued1minuteaverage	CountStatistic	마지막 1분 동안 대기열에 있던 연결의 평균 수입니다.
countqueued5minuteaverage	CountStatistic	마지막 5분 동안 대기열에 있던 연결의 평균 수입니다.
countqueued15minuteaverage	CountStatistic	마지막 15분 동안 대기열에 있던 평균 연결 수입니다.

PWC HTTP 서비스에 대한 통계는 다음 표에 나열되어 있습니다.

표 16–32 PWC HTTP 서비스 통계(EE 전용)

속성 이름	데이터 유형	설명
id	StringStatistic	HTTP 서비스의 인스턴스 이름입니다.
versionserver	StringStatistic	HTTP 서비스의 버전 번호입니다.
timestarted	StringStatistic	HTTP 서비스가 시작된 시간(GMT)입니다.
secondsrunning	CountStatistic	HTTP 서비스가 시작된 이후의 시간(초)입니다.
maxthreads	CountStatistic	각 인스턴스에 있는 최대 작업자 스레드 수입니다.

표 16–32 PWC HTTP 서비스 통계(EE 전용) *(계속)*

속성 이름	데이터 유형	설명
maxvirtualservers	CountStatistic	각 인스턴스에 구성할 수 있는 최대 가상 서버 수입니다.
flagprofilingenabled	CountStatistic	HTTP 서비스 성능 프로필의 활성화 여부입니다. 유효한 값은 0 또는
flagvirtualserver overflow	CountStatistic	maxvirtualservers 이상의 항목이 구성되었는지 여부를 나타냅니다 속성이 1로 설정되면 모든 가상 서버에 대한 통계가 추적되지 않습니
load1minuteaverage	CountStatistic	마지막 1분 동안의 요청에 대한 평균 로드입니다.
load5minuteaverage	CountStatistic	마지막 5분 동안의 요청에 대한 평균 로드입니다.
load15minuteaverage	CountStatistic	마지막 15분 동안의 요청에 대한 평균 로드입니다.
ratebytestransmitted	CountStatistic	서버가 정의한 간격 동안 데이터가 전송된 속도입니다. 이 정보를 사 없을 경우 결과는 0입니다.
ratebytesreceived	CountStatistic	서버가 정의한 간격 동안 데이터가 수신된 속도입니다. 이 정보를 사 없을 경우 결과는 0입니다.

모니터링 활성화 및 비활성화

관리 콘솔에서 구성 노드 > 서버 인스턴스 노드 > 모니터링 페이지를 선택하여 모니터링 수준을 구성하고, 모니터링 수준을 변경할 서비스의 값을 선택할 수 있습니다. 기본적으로 모든 구성 요소 및 서비스에 대한 모니터링은 해제되어 있습니다.

모니터링 수준을 구성하는 방법에 대한 자세한 단계는 관리 콘솔 온라인 도움말을 참조하십시오.

명령줄에서 asadmin set을 사용하여 다양한 Application Server 구성 요소에 대한 모니터링을 활성화합니다. 예를 들어, HTTP 서비스에 대한 모니터링을 활성화하려면 다음 명령을 실행합니다.

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=HIGH
```

get 명령을 사용하여 현재 모니터링이 활성화된 서비스와 구성 요소를 확인합니다.

```
asadmin> get --user admin-user server.monitoring-service.module-monitoring-levels.*
```

반환 결과:

```
server.monitoring-service.module-monitoring-levels.
connector-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.
connector-service = OFF
server.monitoring-service.module-monitoring-levels.ejb-container = OFF
```

```
server.monitoring-service.module-monitoring-levels.http-service = HIGH
server.monitoring-service.module-monitoring-levels.jdbc-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.jms-service = OFF
server.monitoring-service.module-monitoring-levels.jvm = OFF
server.monitoring-service.module-monitoring-levels.orb = OFF
server.monitoring-service.module-monitoring-levels.thread-pool = OFF
server.monitoring-service.module-monitoring-levels.transaction-service = OFF
server.monitoring-service.module-monitoring-levels.web-container = OFF
```

set 명령을 사용하여 모니터링을 비활성화합니다.

예를 들어, HTTP 서비스에 대한 모니터링을 비활성화하려면 다음 명령을 실행합니다.

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=OFF
```

마찬가지로 다른 구성 요소에 대한 모니터링을 비활성화하려면 set 명령을 사용하고 이때 모니터링 수준을 OFF로 지정합니다.

모니터링 데이터 보기

관리 콘솔에서 독립 실행형 서버 인스턴스에 대한 모니터링 페이지로 이동한 후 모니터링이 활성화된 서버 인스턴스에 배포된 구성 요소 또는 서비스를 선택하여 모니터링 데이터를 볼 수 있습니다. 보기 필드 아래에 선택한 구성 요소나 서비스의 모니터링 데이터가 표시됩니다. 모니터링 가능한 등록 정보에 대한 자세한 내용은 다음을 참조하십시오.

관리 콘솔을 사용하여 원격 응용 프로그램 및 인스턴스를 모니터합니다. 이를 위해서는 원격 인스턴스가 실행 중이고 구성이 설정되어야 합니다. 모니터링 데이터를 보는 방법에 대한 자세한 단계는 관리 콘솔 온라인 도움말을 참조하십시오.

명령줄 유틸리티를 사용하여 모니터링 데이터를 보려면 다음과 같이 모니터할 수 있는 객체의 점으로 구분된 이름이 뒤에 오는 asadmin list 및 asadmin get 명령을 사용합니다.

1. 모니터링할 수 있는 객체 이름을 보려면 asadmin list 명령을 사용합니다.

   예를 들어, 서버 인스턴스에 대한 모니터링을 활성화한 응용 프로그램 구성 요소와 하위 시스템 목록을 보려면 터미널 창에 다음 명령을 입력합니다.

   ```
   asadmin> list --user adminuser --monitor server
   ```

   예를 들어, 이전 명령에서는 모니터링이 활성화된 응용 프로그램 구성 요소와 하위 시스템 목록을 반환합니다.

   ```
   server.resources
   server.connector-service
   server.orb
   ```

```
server.jms-service
server.jvm
server.applications
server.http-service
server.thread-pools
```

2. 모니터링이 활성화된 응용 프로그램 구성 요소나 하위 시스템에 대한 모니터링 통계를 표시하려면 asadmin get 명령을 사용합니다.

   통계를 구하려면 단말기 창에서 asadmin get 명령을 입력하고 이전 단계의 list 명령에서 표시한 이름을 지정합니다. 다음 예에서는 특정 객체의 하위 시스템에서 모든 속성을 가져옵니다.

   ```
   asadmin> get --user adminuser --monitor server.jvm.*
   ```

   명령은 다음 속성과 데이터를 반환합니다.

   ```
   server.jvm.dotted-name = server.jvm
   server.jvm.heapsize-current = 21241856
   server.jvm.heapsize-description = Provides statistical information about
      the JVM's memory heap size.
   server.jvm.heapsize-highwatermark = 21241856
   server.jvm.heapsize-lastsampletime = 1080232913938
   server.jvm.heapsize-lowerbound = 0
   server.jvm.heapsize-lowwatermark = 0
   server.jvm.heapsize-name = JvmHeapSize
   server.jvm.heapsize-starttime = 1080234457308
   server.jvm.heapsize-unit = bytes
   server.jvm.heapsize-upperbound = 518979584
   server.jvm.uptime-count = 1080234457308
   server.jvm.uptime-description = Provides the amount of time the JVM has
      been running.
   server.jvm.uptime-lastsampletime = 1080234457308
   server.jvm.uptime-name = JvmUpTime
   server.jvm.uptime-starttime = 1080232913928
   server.jvm.uptime-unit = milliseconds
   ```

점으로 구분된 이름 이해 및 지정

asadmin list 및 get 명령에서 모니터할 수 있는 객체의 점으로 구분된 이름을 지정합니다. 모든 하위 객체는 점(.) 문자를 분리자로 사용하여 표시되므로 **점으로 구분된 이름**이라고 합니다. 자식 노드가 싱글톤 유형인 경우에는 그 객체를 나타내는 데 모니터링 객체 유형만 필요하고 그렇지 않은 경우에는 type.name 형태의 이름이 필요합니다.

예를 들어, http-service는 유효한 모니터 가능 객체 유형 중 하나이며 싱글톤입니다. server 인스턴스의 http-service를 나타내는 싱글톤 하위 노드를 표시하려면 다음의 점으로 구분된 이름을 사용합니다.

```
server.http-service
```

다른 예를 들어보면, application은 유효한 모니터 가능 객체 유형이며 싱글톤이 아닙니다. PetStore 응용 프로그램을 나타내는데 싱글톤이 아닌 자식 노드를 표시하려면 다음과 같은 점으로 구분된 이름을 사용합니다.

```
server.applications.petstore
```

점으로 구분된 이름을 사용하여 모니터할 수 있는 객체의 특정 속성을 나타낼 수도 있습니다. 예를 들어, http-service는 모니터할 수 있는 bytereceived-lastsampletime이라는 속성을 가집니다. 다음 이름은 bytesreceived 속성을 나타냅니다.

```
server.http-service.server.http-listener-1.
  bytesreceived-lastsampletime
```

관리자는 asadmin list 및 get 명령에 유효한 점으로 구분된 이름을 알지 못합니다. list 명령을 사용하여 모니터할 수 있는 객체를 표시하는 한편 get 명령을 와일드카드 매개변수와 함께 사용하여 모니터할 수 있는 객체의 모든 사용 가능한 속성을 검사할 수 있습니다.

점으로 구분된 이름과 함께 list 및 get 명령을 사용할 때 다음 가정을 전제로 합니다.

- 점으로 구분된 이름 다음에 와일드 카드(*)가 **없는** list 명령은 현재 노드의 직계 하위를 반환합니다. 예를 들어, list --user *adminuser* --monitor server는 server 노드에 속하는 모든 직계 하위를 나열합니다.
- 점으로 구분된 이름 다음에 .* 양식의 와일드카드가 있는 list 명령은 현재 노드에서 하위 노드의 계층 트리를 만듭니다. 예를 들어, list --user *adminuser* --monitor server.applications.*는 applications의 모든 하위와 그들의 후속 하위 노드 등을 나열합니다.
- 점으로 구분된 이름이 앞에 있거나 **dotted name** * 양식의 와일드카드가 뒤에 나오는 list 명령은 제공한 비교 패턴으로 만들어진 정규 표현식과 일치하는 모든 노드와 하위를 표시합니다.
- 뒤에 .* 또는 *가 나오는 get 명령은 비교할 현재 노드에 속하는 속성 및 속성 값 세트를 표시합니다.

자세한 내용은 182 페이지 “모든 수준의 list 및 get 명령에 대한 예상 출력”을 참조하십시오.

list 명령 예

list 명령은 지정된 서버 인스턴스 이름에 대해 현재 모니터되고 있는 응용 프로그램 구성 요소 및 하위 시스템에 대한 정보를 제공합니다. 이 명령을 사용하여 서버

인스턴스에 대해 모니터할 수 있는 구성 요소와 하위 구성 요소를 볼 수 있습니다. list 예의 전체 목록에 대한 자세한 내용은 182 페이지 "모든 수준의 list 및 get 명령에 대한 예상 출력"을 참조하십시오.

예 1

```
asadmin> list --user admin-user --monitor server
```

예를 들어, 이전 명령에서는 모니터링이 활성화된 응용 프로그램 구성 요소와 하위 시스템 목록을 반환합니다.

```
server.resources
server.orb
server.jvm
server.jms-service
server.connector-service
server.applications
server.http-service
server.thread-pools
```

지정된 서버 인스턴스에서 현재 모니터되는 응용 프로그램을 나열할 수도 있습니다. 이 목록은 응용 프로그램에서 get 명령을 사용하여 특정 모니터링 통계를 얻으려고 할 때 유용할 수 있습니다.

예 2

```
asadmin> list --user admin-user --monitor server.applications
```

반환 결과:

```
server.applications.adminapp
 server.applications.admingui
server.applications.myApp
```

get 명령 예

get 명령은 다음과 같은 모니터된 정보를 검색합니다.

- 구성 요소 또는 하위 시스템 내에서 모니터된 모든 속성
- 구성 요소 또는 하위 시스템 내에서 모니터된 특정 속성

  특정 구성 요소나 하위 시스템에 존재하지 않는 속성을 요청한 경우에는 오류가 반환됩니다. 마찬가지로 구성 요소나 하위 시스템에 대해 활성화되어 있지 않은 특정 속성을 요청한 경우에도 오류가 반환됩니다.

  get 명령 사용에 대한 자세한 내용은 182 페이지 "모든 수준의 list 및 get 명령에 대한 예상 출력"을 참조하십시오.

예 1

하위 시스템에서 특정 객체에 대한 모든 속성을 검색하려는 경우:

```
asadmin> get --user admin-user --monitor server.jvm.*
```

반환 결과:

```
server.jvm.dotted-name= server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information about
   the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
   been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

예 2

J2EE 응용 프로그램에서 모든 속성을 검색하려는 경우:

```
asadmin> get --user admin-user --monitor server.applications.myJ2eeApp.*
```

반환 결과:

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

J2EE 응용 프로그램 수준에 모니터할 수 있는 속성이 없기 때문에 이 응답이 표시됩니다.

예 3

하위 시스템에서 특정 속성을 검색하려는 경우:

```
asadmin> get --user admin-user --monitor server.jvm.uptime-lastsampletime
```

반환 결과:

```
server.jvm.uptime-lastsampletime = 1093215374813
```

예 4

하위 시스템 속성 내에서 알 수 없는 속성을 가져오려는 경우:

```
asadmin> get --user admin-user --monitor server.jvm.badname
```

반환 결과:

```
No such attribute found from reflecting the corresponding Stats
interface: [badname]
CLI137 Command get failed.
```

PetStore 예 사용

다음 예에서는 모니터링을 위해 asadmin 도구를 사용하는 방법에 대해 설명합니다.

샘플 Petstore 응용 프로그램이 Application Server에 배포된 후 이 응용 프로그램의 메소드가 호출된 횟수를 검사하려고 합니다. 응용 프로그램이 배포된 인스턴스 이름은 server입니다. list 명령과 get 명령을 조합하여 메소드에 대한 원하는 통계에 액세스합니다.

1. Application Server 및 asadmin 도구를 시작합니다.
2. 다음과 같이 몇 가지 유용한 환경 변수를 설정하여 모든 명령에 일일이 입력하지 않도록 합니다.

   ```
   asadmin> export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123
   asadmin>export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=4849
   ```

3. server 인스턴스에 대해 모니터할 수 있는 구성 요소를 나열합니다.

   ```
   asadmin> list --user adminuser --monitor server*
   ```

   반환 결과(다음과 유사한 출력):

   ```
   server
   server.applications
   server.applications.CometEJB
   server.applications.ConverterApp
   server.applications.petstore
   server.http-service
   server.resources
   server.thread-pools
   ```

   모니터할 수 있는 구성 요소 목록에는 thread-pools, http-service, resources와 모든 배포 및 활성화된 applications이 포함됩니다.

4. PetStore 응용 프로그램의 모니터할 수 있는 하위 구성 요소를 나열합니다(--monitor 대신 -m을 사용할 수 있음).

   ```
   asadmin> list -m server.applications.petstore
   ```

   반환 결과:

   ```
   server.applications.petstore.signon-ejb_jar
   server.applications.petstore.catalog-ejb_jar
   server.applications.petstore.uidgen-ejb_jar
   server.applications.petstore.customer-ejb_jar
   server.applications.petstore.petstore-ejb_jar
   server.applications.petstore.petstore\.war
   server.applications.petstore.AsyncSenderJAR_jar
   server.applications.petstore.cart-ejb_jar
   ```

5. PetStore 응용 프로그램의 EJB 모듈 signon-ejb_jar에 있는 모니터할 수 있는 하위 구성 요소를 나열합니다.

   ```
   asadmin> list -m server.applications.petstore.signon-ejb_jar
   ```

   반환 결과:

   ```
   server.applications.petstore.signon-ejb_jar.SignOnEJB
   server.applications.petstore.signon-ejb_jar.UserEJB
   ```

6. PetStore 응용 프로그램의 EJB 모듈 signon-ejb_jar에 대한 Entity Bean UserEJB에서 모니터할 수 있는 하위 구성 요소를 나열합니다.

   ```
   asadmin> list -m server.applications.petstore.signon-ejb_jar.UserEJB
   ```

   반환 결과(공간을 고려하여 제거한 점으로 구분된 이름이 있음):

   ```
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-cache
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-pool
   ```

7. PetStore 응용 프로그램의 EJB 모듈 signon-ejb_jar에 있는 Entity Bean UserEJB의 getUserName 메소드에서 모니터할 수 있는 하위 구성 요소를 나열합니다.

   ```
   asadmin> list -m
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName
   ```

   반환 결과:

   ```
   Nothing to list at server.applications.petstore.signon-ejb_jar.
   UserEJB.bean-methods.getUserName. To get the valid names beginning with a
   string, use the wildcard "*" character. For example, to list all names
   that begin with "server", use "list server*".
   ```

8. 메소드에 대해 모니터할 수 있는 하위 구성 요소가 없습니다. getUserName 메소드에 대해 모니터할 수 있는 모든 통계를 가져옵니다.

```
asadmin> get -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName.*
```

반환 결과:

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-description = Provides the time in milliseconds
   spent during the last successful/unsuccessful attempt to execute the
   operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-lastsampletime = 1079981809259
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-description = Provides the number of times an
   operation was called, the total time that was spent during the
   invocation and so on.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-lastsampletime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-maxtime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-mintime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-totaltime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-unit =
 server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-description = Provides the total number of errors
   that occured during invocation or execution of an operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
```

```
getUserName.totalnumerrors-lastsampletime = 1079981809273
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-name = TotalNumErrors
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-description = Provides the total number of
   successful invocations of the method.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-lastsampletime = 1079981809255
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-name = TotalNumSuccess
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-unit = count
```

9. 실행 시간과 같은 특정 통계를 얻으려면 다음 명령을 사용합니다.

```
asadmin> get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.executiontime-count
```

반환 결과:

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 1
```

모든 수준의 list 및 get 명령에 대한 예상 출력

다음 표에서는 트리의 각 수준에서의 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 16–33 최상위 수준

명령	점으로 구분된 이름	출력:
`list -m`	`server`	`server.applicationsserver.thread-poolsserver.resourcesserver.http-serviceserver.transaction-serviceserver.orb.connection-managersserver.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnectionsserver.jvm`
`list -m`	`server.*`	이 노드 아래의 자식 노드 계층입니다.

표 16–33 최상위 수준 *(계속)*

명령	점으로 구분된 이름	출력:
get -m	server.*	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다

다음 표에서는 응용 프로그램 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 16–34 응용 프로그램 수준

명령	점으로 구분된 이름	출력:
list -m	server.applications 또는 *applications	appl1app2web-module1_warejb-module2_jar...
list -m	server.applications.* 또는 *applications.*	이 노드 아래의 자식 노드 계층입니다.
get -m	server.applications.* 또는 *applications.*	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시

다음 표에서는 응용 프로그램 수준의 독립 실행형 모듈 및 엔터프라이즈 응용 프로그램에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 16–35 응용 프로그램 - 엔터프라이즈 응용 프로그램 및 독립 실행형 모듈

명령	점으로 구분된 이름	출력:
list -m	server.applications.app1 또는 *app1 참고: 엔터프라이즈 응용 프로그램이 배포된 경우에만 이 수준을 적용할 수 있습니다. 독립 실행형 모듈이 배포된 경우에는 적용할 수 없습니다.	ejb-module1_jarweb-module2_warejb-module3_jarweb-module3
list -m	server.applications.app1.* 또는 *app1.*	이 노드 아래의 자식 노드 계층입니다.

표 16–35 응용 프로그램 - 엔터프라이즈 응용 프로그램 및 독립 실행형 모듈 *(계속)*

명령	점으로 구분된 이름	출력:

`get -m`	`server.applications.app1.*`
	또는
	`*app1.*`
		이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.

`list -m`	`server.applications.app1.ejb-module1_jar`
	또는
	`*ejb-module1_jar`
	또는
	`server.applications.ejb-module1_jar`
		`bean1bean2bean3...`

`list -m`	`server.applications.app1.ejb-module1_jar`
	또는
	`*ejb-module1_jar`
	또는
	`server.applications.ejb-module1_jar`
		이 노드 아래의 자식 노드 계층입니다.

`get -m`	`server.applications.app1.ejb-module1_jar.*`
	또는
	`*ejb-module1_jar.*`
	또는
	`server.applications.ejb-module1_jar.*`
		이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.

`list -m`	`server.applications.app1.ejb-module1_jar.bean1` 참고: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 `app1`)이 포함된 노드는 표시되지 않습니다.	하위 노드 목록: `bean-poolbean-cachebean-method`
`list -m`	`server.applications.app1.ejb-module1_jar.bean1` 참고: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 `app1`)이 포함된 노드는 표시되지 않습니다.	하위 노드의 계층 구조 및 이 노드와 모든 후속 하위 노드에 대한 모든 속성 목록입니다.

표 16–35 응용 프로그램 - 엔터프라이즈 응용 프로그램 및 독립 실행형 모듈 *(계속)*

명령	점으로 구분된 이름	출력:

`get -m`	`server.applications.app1.ejb-module1_jar.bean1.*`
	참고: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 `app1`)이 포함된 노드는 표시되지 않습니다.
		속성 및 해당하는 연관된 값은 다음과 같습니다.
		`CreateCount_CountCreateCount_`
		`DescriptionCreateCount_`
		`LastSampleTimeCreateCount_`
		`NameCreateCount_`
		`StartTimeCreateCount_`
		`UnitMethodReadyCount_`
		`CurrentMethodReadyCount_`
		`DescriptionMethodReadyCount_`
		`HighWaterMarkMethodReadyCount_`
		`LastSampleTimeMethodReadyCount_`
		`LowWaterMarkMethodReadyCount_`
		`NameMethodReadyCount_`
		`StartTimeMethodReadyCount_`
		`UnitRemoveCount_CountRemoveCount_`
		`DescriptionRemoveCount_`
		`LastSampleTimeRemoveCount_`
		`NameRemoveCount_StartTimeAttribute`
		`RemoveCount_Unit`

`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-pool` 참고: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 `app1`)이 포함된 노드는 표시되지 않습니다.	해당 속성은 없지만 다음과 같은 메시지가 표시됩니다. `server.applications.appl.ejb-module1_jar.bean1-cache`에서 나열할 사항이 없습니다. 문자열로 시작하는 유효한 이름을 얻으려면 와일드카드(*) 문자를 사용하십시오. 예를 들어, `server`로 시작하는 모든 이름을 나열하려면 `list server*`를 사용합니다.
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-pool.*` 참고: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 `app1`)이 포함된 노드는 표시되지 않습니다.	표 1-4에서 설명한 EJB 풀 속성에 해당하는 속성 및 값 목록입니다.
`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-cache` 참고: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 `app1`)이 포함된 노드는 표시되지 않습니다.	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-cache.*` 참고: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 `app1`)이 포함된 노드는 표시되지 않습니다.	표 1-5에서 설명한 EJB 캐시 속성에 해당하는 속성 및 값 목록입니다.

표 16–35 응용 프로그램 - 엔터프라이즈 응용 프로그램 및 독립 실행형 모듈 *(계속)*

명령	점으로 구분된 이름	출력:
`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-method.method1` 참고: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 `app1`)이 포함된 노드는 표시되지 않습니다.	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-method.method1.*` 참고: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 `app1`)이 포함된 노드는 표시되지 않습니다.	표 1-2에서 설명한 EJB 메소드 속성에 해당하는 속성 및 값 목록입니다.
`list -m`	`server.applications.app1.web-module1_war`	모듈에 할당된 가상 서버를 표시합니다.
`get -m`	`server.applications.app1.web-module1_war.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.applications.app1.web-module1_war.virtual_server`	등록된 서블릿 목록을 표시합니다.
`get -m`	`server.applications.app1.web-module1_war.virtual_server.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.applications.app1.web-module1_war.virtual_server.servlet1`	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values 메시지가 표시됩니다.
`get -m`	`server.applications.app1.web-module1_war.virtual_server.servlet1.*`	표 1-7에서 설명한 웹 컨테이너(서블릿) 속성에 해당하는 속성 및 값 목록입니다.

다음 표에서는 HTTP 서비스 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 16–36 HTTP-서비스 수준

명령	점으로 구분된 이름	출력:
`list -m`	`server.http-service`	가상 서버 목록입니다.
`get -m`	`server.http-service.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.http-service.server`	HTTP Listener 목록입니다.
`get -m`	`server.http-service.server.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.

표 16–36 HTTP-서비스 수준 (계속)

명령	점으로 구분된 이름	출력:
list -m	server.http-service.server.http-listener1	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
get -m	server.http-service.server.*	HTTP 서비스 속성에 해당하는 속성 및 값 목록입니다.

다음 표에서는 스레드 풀 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 16–37 스레드 풀 수준

명령	점으로 구분된 이름	출력:
list -m	server.thread-pools	thread-pool 이름 목록입니다.
get -m	server.thread-pools.*	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
list -m	server.thread-pools.orb\.threadpool \.thread-pool-1	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values." 메시지가 표시됩니다.
get -m	server.thread-pools..orb\.threadpool \.thread-pool-1.*	스레드 풀 속성에 해당하는 속성 및 값 목록입니다.

다음 표에서는 자원 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 16–38 자원 수준

명령	점으로 구분된 이름	출력:
list -m	server.resources	풀 이름 목록입니다.
get -m	server.resources.*	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
list -m	server.resources.jdbc-connection-pool-pool. connection-pool1	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.

표 16–38 자원 수준 *(계속)*

명령	점으로 구분된 이름	출력:
`get -m`	`server.resources.jdbc-connection-pool-pool.connection-pool1.*`	연결 풀 속성에 해당하는 속성 및 값 목록입니다.

다음 표에서는 트랜잭션 서비스 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 16–39 트랜잭션 서비스 수준

명령	점으로 구분된 이름	출력:
`list -m`	`server.transaction-service`	해당 속성은 없지만 "Use get command with the `--monitor` option to view this node's attributes and values." 메시지가 표시됩니다.
`get -m`	`server.transaction-service.*`	트랜잭션 서비스 속성에 해당하는 속성 및 값 목록입니다.

다음 표에서는 ORB 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 16–40 ORB 수준

명령	점으로 구분된 이름	출력:
`list -m`	`server.orb`	`server-orb.connection-managers`
`get -m`	`server.orb.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.orb.connection-managers`	ORB 연결 관리자의 이름입니다.
`get -m`	`server.orb.connection-managers.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections`	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
`get -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections.*`	ORB 연결 관리자 속성에 해당하는 속성 및 값 목록입니다.

다음 표에서는 JVM 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 16–41 JVM 수준

명령	점으로 구분된 이름	출력:
`list -m`	`server.jvm`	해당 속성은 없지만 "Use get command with the `--monitor` option to view this node's attributes and values."와 유사한 메시지가 표시됩니다.
`get -m`	`server.jvm.*`	JVM 속성에 해당하는 속성 및 값 목록입니다.

Jconsole 사용

이 절은 다음 내용으로 구성되어 있습니다.

Application Server 관리(관리 및 모니터링)는 JMX를 기반으로 합니다. 이는 관리 대상 구성 요소가 MBean으로 표시된다는 것을 의미합니다. Java 2 Standard Edition(J2SE) 5.0을 통해 JVM을 모니터하고 JVM MBean을 확인하여 작업 상황을 이해할 수 있습니다. Application Server는 이 JVM을 제공하기 위해 `System JMX Connector Server`라는 표준 JMX Connector Server의 구성을 제공합니다. Application Server를 시작하면 이 JMX Connector Server의 인스턴스가 시작되며 신뢰할 수 있는 클라이언트에 이 JVM이 제공됩니다.

Java 모니터링 및 관리 콘솔(JConsole)은 JMX 백엔드를 관리할 수 있는 일반적인 JMX 커넥터입니다.
JConsole(`http://java.sun.com/j2se/1.5.0/docs/tooldocs/share/jconsole.html`)은 J2SE 5.0을 시작할 때 표준 JDK 배포의 일부로 사용할 수 있습니다. JConsole에 대한 자세한 내용은
`http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html`을 참조하십시오.

Application Server에 JConsole을 구성할 때 Application Server는 JMX 커넥터의 서버측이 되고 JConsole은 JMX 커넥터의 기본 클라이언트측이 됩니다.

Application Server 연결에 대한 JConsole 보안

연결에 대한 전송 계층 보안을 기반으로 Application Server 또는 JMX 커넥터 서버측에 연결하는 방법에는 미묘한 차이가 있습니다. 서버측이 보안 설정(전송 계층 보안 보장)되면 클라이언트측에서 수행할 구성 작업이 조금 많아집니다.

- Application Server의 Platform Edition에서는 기본적으로 시스템 JMX 커넥터 서버측 보안이 활성화되지 않습니다.
- Application Server의 Enterprise Edition에서는 기본적으로 시스템 JMX 커넥터 서버측 보안이 활성화됩니다.
- 통신에 사용되는 프로토콜은 RMI/JRMP입니다. JMX 커넥터에 대한 보안이 활성화되면 RMI/JRMP 프로토콜이 SSL을 통해 사용됩니다.

주 – SSL을 통해 RMI를 사용하면 클라이언트가 대상 서버와 통신하는지 확인하는 추가 검사가 수행되지 않습니다. 따라서, JConsole 사용 중 항상 사용자 이름 및 비밀번호를 악성 호스트에 보낼 가능성이 있습니다. 보안을 손상시키지 않으려면 전적으로 관리자의 노력이 필요합니다.

Platform Edition 도메인을 `appserver.sun.com`과 같은 시스템에 설치할 때 DAS(Domain Administration Server, 관리 서버 또는 도메인이라고도 함)의 `domain.xml`에 다음 항목이 표시됩니다.

```
<!- – The JSR 160 "system-jmx-connector" – –><jmx-connector accept-all="false"
address="0.0.0.0" auth-realm-name="admin-realm" enabled="true" name="system"
port="8686" protocol="rmi_jrmp" security-enabled="false"/> <!- – The JSR 160
"system-jmx-connector" – –>
```

JMX 커넥터에 대한 `security-enabled` 플래그는 *false*로 설정되어 있습니다. Enterprise Edition을 실행 중인 경우 또는 Platform Edition에서 JMX 커넥터에 대한 보안을 활성화한 경우에는 이 플래그가 *true*로 설정됩니다.

```
<!- – The JSR 160 "system-jmx-connector" – –><jmx-connector accept-all="false"
address="0.0.0.0" auth-realm-name="admin-realm" enabled="true" name="system"
port="8686" protocol="rmi_jrmp" security-enabled="true"/>
...</jmx-connector><!- – The JSR 160 "system-jmx-connector" – –>
```

JConsole을 Application Server에 연결하는 데 필요한 필수 조건

JConsole 설정에는 서버측 및 클라이언트측의 두 가지 부분이 있습니다. Application Server 도메인은 강력한 Solaris 서버인 `appserver.sun.com`이라고 하는 시스템에 설치되며 이 부분이 서버측이 됩니다.

클라이언트측에도 Application Server가 설치됩니다. 클라이언트측은 Windows 시스템에 Java SE 5.0 및 Application Server가 설치되었다고 가정합니다.

주 – Application Server 도메인에 대한 보안이 원격 시스템에서 활성화된 경우에만(Enterprise Edition 기본값) 클라이언트측에 Application Server를 설치해야 합니다. 위의 Solaris 시스템에서 Application Server Platform Edition 도메인만 관리하려는 경우 이 클라이언트 시스템에 Application Server를 설치할 필요가 없습니다.

서버측과 클라이언트측이 동일한 시스템에 있는 경우 `localhost`를 사용하여 호스트 이름을 지정할 수 있습니다.

Application Server에 JConsole 연결

이 절에서는 JMX 커넥터에 대한 보안을 활성화하지 않고 JConsole을 Application Server에 연결하는 방법에 대해 설명합니다. Application Server Platform Edition에서는 기본적으로 보안이 활성화되지 않습니다.

1. `appserver.sun.com`에서 도메인을 시작합니다.
2. *JDK_HOME*`/bin/jconsole`을 실행하여 JConsole을 시작합니다.
3. JConsole의 에이전트에 연결 탭에서 사용자 이름, 비밀번호, 호스트 이름 및 포트(기본값: 8686)를 입력합니다.

   사용자 이름은 관리 사용자 이름을 참조하고 비밀번호는 도메인의 관리 비밀번호를 참조합니다.
4. 연결을 누릅니다.

   JConsole 창의 여러 탭에 모든 MBean, VM 정보 등이 표시됩니다.

Application Server에 JConsole 보안 연결

이 절에서는 JMX 커넥터에 대한 보안을 활성화하고 JConsole을 Application Server에 연결하는 방법에 대해 설명합니다. Application Server Enterprise Edition에서는 기본적으로 보안이 활성화되어 있습니다. Platform Edition에서 JMX 커넥터에 대한 보안을 활성화한 경우 이 절차를 사용합니다.

1. 클라이언트 시스템(JConsole이 설치됨)에 Application Server를 설치합니다.

   이 작업을 수행하는 유일한 목적은 Domain Administration Server의 신뢰할 수 있는 서버 인증서가 있는 위치를 JConsole에 알리기 위한 것입니다. 해당 인증서를 얻으려면 **원격** `asadmin` 명령을 하나 이상 호출해야 하며 Application Server를 로컬로 설치해야 합니다.
2. `appserver.sun.com`에서 Application Server Enterprise Edition을 시작합니다.

   이 도메인은 Enterprise Edition 도메인이므로 System JMX Connector Server 보안이 활성화됩니다.

3. 로컬 Application Server 설치에서 *install-dir*`/bin/asadmin list --user admin --secure=true --host appserver.sun.com --port 4849`(여기서 4849는 서버 관리 포트임)를 실행합니다.

   이 예에서는 `asadmin list` 명령이 선택되었지만 모든 원격 `asadmin` 명령을 실행할 수 있습니다. 이제 `appserver.sun.com`의 DAS에서 전송된 인증서를 수락하라는 메시지가 표시됩니다.

4. `y`를 입력하여 `appserver.sun.com`의 Domain Administration Server에서 전송된 인증서를 수락합니다.

   서버 인증서는 클라이언트 시스템의 홈 디렉토리에 있는 `.asadmintruststore`라는 파일에 저장됩니다.

주 – 서버 시스템 및 클라이언트 시스템이 동일한 경우에는 이 단계가 필요하지 않습니다. 즉, `appserver.sun.com`에서 JConsole도 실행 중인 경우 이 단계를 생략합니다.

---

5. 다음 JConsole 명령을 사용하여 JConsole에 DAS의 트러스트 저장소 위치를 알립니다.

   JDK-dir`/bin/jconsole.exe -J-Djavax.net.ssl.trustStore="C:/Documents and Settings/`*user*`/.asadmintruststore"`

   이 인증서는 이제 JConsole에서 자동으로 트러스트됩니다.

6. *JDK_HOME*`/bin/jconsole`을 실행하여 JConsole을 시작합니다.

7. JConsole의 에이전트에 연결 탭에서 사용자 이름, 비밀번호, 호스트 이름 및 포트(기본값: 8686)를 입력합니다.

   사용자 이름은 관리 사용자 이름을 참조하고, 비밀번호는 도메인의 관리 비밀번호를 참조합니다.

8. 연결을 누릅니다.

   JConsole 창의 여러 탭에 모든 MBean, VM 정보 등이 표시됩니다.

◆ ◆ ◆ 17장

17 JVM(Java Virtual Machine) 및 고급 설정

JVM(Java Virtual Machine)은 컴파일된 Java 프로그램에서 바이트 코드를 실행하는 데 필요한 해석적 컴퓨팅 엔진입니다. JVM은 Java 바이트 코드를 호스트 시스템의 원시 명령으로 변환합니다. Java 프로세스인 Application Server에는 서버에서 실행되는 Java 응용 프로그램을 실행하고 지원하기 위해 JVM이 필요합니다. JVM 설정은 Application Server 구성의 일부입니다.

이 장에서는 JVM(Java Virtual Machine) 및 다른 고급 설정을 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

JVM 설정 조정

Application Server 구성 작업의 일부로 JVM(Java Virtual Machine) 사용을 향상시킬 수 있는 설정을 정의합니다. 관리 콘솔을 사용하여 JVM 구성을 변경하려면 Application Server > JVM 설정 탭을 선택하고 다음과 같이 일반 JVM 설정을 정의합니다.

- Java 홈: Java 소프트웨어의 설치 디렉토리 이름을 입력합니다. Application Server는 Java SE 소프트웨어를 사용합니다.

  주 – 존재하지 않는 디렉토리 이름을 입력하거나 지원되지 않는 Java EE 소프트웨어 버전의 설치 디렉토리 이름을 입력하면 Application Server가 시작되지 않습니다.

- Javac 옵션: Java 프로그래밍 언어 컴파일러에 대한 명령줄 옵션을 입력합니다. EJB 구성 요소가 배포되면 Application Server에서 컴파일러를 실행합니다.
- 디버그: JPDA(Java Platform Debugger Architecture)로 디버깅하도록 설정하려면 이 사용 가능 확인란을 선택합니다.

  JPDA는 응용 프로그램 개발자가 사용합니다.

- 디버그 옵션: 디버깅이 활성화될 때 JVM에 전달되는 JPDA 옵션을 지정합니다.
- RMI 컴파일 옵션: `rmic` 컴파일러에 대한 명령줄 옵션을 입력합니다. EJB 구성 요소가 배포되면 Application Server가 `rmic` 컴파일러를 실행합니다.
- 바이트 코드 선행 프로세서: 쉼표로 구분된 클래스 이름 목록을 입력합니다. 클래스마다 `com.sun.appserv.BytecodePreprocessor` 인터페이스를 구현해야 합니다. 지정한 순서대로 클래스가 호출됩니다.

  프로필러와 같은 도구를 사용하려면 바이트 코드 선행 프로세서 필드 항목이 필요할 수 있습니다. 프로필러는 서버 성능을 분석하는 데 필요한 정보를 생성합니다.

고급 설정 구성

관리 콘솔을 사용하여 고급 응용 프로그램 구성을 설정하려면 Application Server > 고급 탭 > 응용 프로그램 구성 탭을 선택하고 다음과 같이 응용 프로그램 구성을 설정합니다.

- 재로드: 응용 프로그램의 동적 재로드를 활성화하려면 이 확인란을 선택합니다.

  동적 재로드가 활성화되면(기본값) 해당 코드 또는 배포 설명자를 변경할 때 응용 프로그램 또는 모듈을 재배포할 필요가 없습니다. 이 경우 변경된 JSP 또는 클래스 파일을 응용 프로그램 또는 모듈의 배포 디렉토리에 복사만 하면 됩니다. 서버는 변경 사항을 정기적으로 확인하고 변경 사항과 함께 응용 프로그램을 동적으로(자동으로) 재배포합니다. 이 기능은 코드 변경을 빠르게 테스트할 수 있기 때문에 개발 환경에서 유용합니다. 그러나 프로덕션 환경에서는 동적 재로드가 성능을 저하시킬 수 있습니다. 또한 재로드가 수행될 때마다 해당 전송 시간의 세션이 무효화됩니다. 클라이언트가 세션을 다시 시작해야 합니다.
- 재로드 폴링 간격: 응용 프로그램 및 모듈에서 코드 변경 사항을 검사하여 동적으로 재로드할 간격을 정의합니다. 기본값은 2입니다.
- 관리 세션 시간 초과: 관리 세션 시간 초과의 기준이 되는 비활성 시간(분)을 지정합니다.

또한 다음과 같이 배포 설정을 정의합니다.

- 자동 배포: 응용 프로그램의 자동 배포를 활성화하려면 이 확인란을 선택합니다.

  자동 배포에서는 응용 프로그램 또는 모듈 파일(JAR, WAR, RAR 또는 EAR)을 특수 디렉토리에 복사해야 합니다. 여기서 이 파일은 Application Server에 의해 자동으로 배포됩니다.
- 자동 배포 폴링 간격: 응용 프로그램 및 모듈에서 코드 변경 사항을 검사하여 동적으로 재로드할 간격을 정의합니다. 기본값은 2입니다.
- 검증자: 배포 설명자 파일을 검증하려면 검증자 사용 가능 확인란을 선택합니다. 이 기능은 선택 사항입니다.
- 사전 컴파일: JSP 파일을 사전 컴파일하려면 사전 컴파일 사용 가능 확인란을 선택합니다.

◆ ◆ ◆ 18장

18 domain.xml의 점으로 구분된 이름 속성

이 부록에서는 MBean과 해당 속성을 지정하는 데 사용할 수 있는 점으로 구분된 이름 속성에 대해 설명합니다. `domain.xml` 파일의 모든 요소에는 해당하는 MBean이 있습니다. 이 이름을 사용하는 구문에서는 점으로 이름을 구분해야 하기 때문에 이 이름을 **점으로 구분된 이름**이라고 합니다.

*(별표)는 위치와 상관없이 점으로 구분된 이름의 일부로 사용할 수 있으며 정규 표현식에서 와일드카드 문자의 역할을 합니다. 와일드카드 문자를 사용하면 점으로 구분된 이름의 일부분을 축약할 수 있는 이점을 얻을 수 있습니다. 예를 들어, `this.is.really.long.hierarchy`와 같이 점으로 구분된 이름이 긴 경우 `th*.hierarchy`로 축약할 수 있습니다. 하지만 .은 항상 이름을 구분하는 역할을 합니다. *를 사용하면 점으로 구분된 이름의 전체 목록이 표시됩니다.

이 부록은 다음 내용으로 구성되어 있습니다.

최상위 수준 요소

`domain.xml` 파일의 모든 최상위 수준 요소에 대해서 다음 조건을 준수해야 합니다.

- 모든 서버, 구성, 클러스터 또는 노드 에이전트에는 고유한 이름이 있어야 합니다.
- 서버, 구성, 클러스터 또는 노드 에이전트 이름을 `domain`으로 지정할 수 없습니다.
- 서버 인스턴스 이름을 `agent`로 지정할 수 없습니다.

다음 표에서는 최상위 수준 요소와 해당하는 점으로 구분된 이름 접두어를 설명합니다.

요소 이름	점으로 구분된 이름 접두어
applications	`domain.applications`

요소 이름	점으로 구분된 이름 접두어
resources	`domain.resources`
configurations	`domain.configs`
servers	`domain.servers` 이 요소에 포함된 모든 서버는 *server-name*으로 액세스할 수 있습니다. 여기에서 *server-name*은 서버 하위 요소에 대한 이름 속성 값입니다.
clusters	`domain.clusters` 이 요소에 포함된 모든 클러스터는 *cluster-name* 으로 액세스할 수 있습니다. 여기에서 *cluster-name* 은 클러스터 하위 요소의 이름 속성 값입니다.
node-agents	`domain.node-agents`
lb-configs	`domain.lb-configs`
system-property	`domain.system-property`

두 가지 수준의 별칭을 사용할 수 있습니다.

1. 첫 번째 수준의 별칭을 사용하여 `domain.servers` 또는 `domain.clusters` 접두어를 검색하지 않고 서버 인스턴스 또는 클러스터의 속성에 액세스할 수 있습니다. 따라서, `server1` 양식과 같이 점으로 구분된 이름은 점으로 구분된 이름 `domain.servers.server1`로 매핑되며 여기서 `server1`은 서버 인스턴스입니다.
2. 두 번째 수준의 별칭을 사용하여 클러스터나 독립 실행형 서버 인스턴스 대상의 구성 응용 프로그램 및 자원을 참조합니다.

다음 표에서는 도메인의 최상위 수준 이름에 대한 별칭인 서버 이름이나 클러스터 이름으로 시작하는 점으로 구분된 이름을 설명합니다.

점으로 구분된 이름	별칭	설명
target`.applications.*`	`domain.applications.*`	별칭은 *target*에서만 참조하는 응용 프로그램으로 변환됩니다.
target`.resources.*`	`domain.resources.*`	별칭은 모든 `jdbc-connection-pool`, `connector-connection-pool`, `resource-adapter-config` 및 *target*에서 참조되는 다른 모든 자원으로 변환됩니다.

다음 표에서는 서버나 클러스터에서 참조하는 구성 내의 최상위 수준 이름으로 별칭 지정된 서버 이름이나 클러스터 이름으로 시작하는 점으로 구분된 이름을 설명합니다.

점으로 구분된 이름	별칭
target.http-service	*config-name*.http-service
target.iiop-service	*config-name*.iiop-service
target.admin-service	*config-name*.admin-service
target.web-container	*config-name*.web-container
target.ejb-container	*config-name*.ejb-container
target.mdb-container	*config-name*.mdb-container
target.jms-service	*config-name*.jms-service
target.log-service	*config-name*.log-service
target.security-service	*config-name*.security-service
target.transaction-service	*config-name*.transaction-service
target.monitoring-service	*config-name*.monitoring-service
target.java-config	*config-name*.java-config
target.availability-service	*config-name*.availability-service
target.thread-pools	*config-name*.thread-pools

별칭을 지정하지 않는 요소

클러스터화된 인스턴스는 별칭을 지정해서는 안 됩니다. 클러스터화된 인스턴스의 시스템 등록 정보를 가져올 때 점으로 구분된 이름 속성으로 *clustered-instance-name*.*system-property*가 아닌 domain.servers.*clustered-instance-name*.*system-property*를 사용해야 합니다.

asadmin 명령

asadmin get, set 및 list 명령을 차례대로 실행하면 Application Server의 추상 계층에 대한 탐색 메커니즘이 제공됩니다. 계층에는 구성 및 모니터링의 두 가지 계층이 있으며 이 명령은 두 가지 계층 모두에서 작동합니다. list 명령은 읽기 전용 또는 수정 가능한 속성을 갖는 관리 구성 요소의 점으로 구분된 정규화된 이름을 제공합니다.

구성 계층은 수정 가능한 속성을 제공하지만, 모니터링 계층은 읽기 전용의 관리 구성 요소 속성만을 제공합니다. 구성 계층은 어느 정도 도메인의 스키마 문서를 기반으로 합니다. list 명령을 사용하면 원하는 계층의 특정 관리 구성 요소에 연결할 수 있습니다. 그런 다음 get 및 set 명령을 호출하면 이름 및 값을 얻거나 관리 구성 요소의 속성 값을

설정할 수 있습니다. 와일드카드(*) 옵션을 사용하면 점으로 구분된 정규화된 해당 이름과 일치하는 모든 항목을 불러올 수 있습니다. `get`, `set` 및 `list` 명령의 사용 예는 다음 설명서 페이지를 참조하십시오.

`get`(1)
`set`(1)
`list`(1)

◆ ◆ ◆ 19장

19 asadmin 유틸리티

Application Server에는 asadmin으로 알려진 명령줄 관리 유틸리티가 포함되어 있습니다. asadmin 유틸리티는 사용자, 자원 및 응용 프로그램 관리뿐 아니라 Application Server 시작 및 중지에도 사용됩니다.

이 장은 다음 내용으로 구성되어 있습니다.

asadmin 명령 사용법

asadmin 유틸리티를 사용하면 Application Server의 모든 관리 작업을 수행할 수 있습니다. 관리 콘솔을 사용하는 대신 이 asadmin 유틸리티를 사용할 수 있습니다.

asadmin 유틸리티는 수행할 작업을 식별하는 명령을 호출합니다. 이 명령은 대소문자를 구분합니다. 짧은 옵션 인수에는 한 개의 대시(-)를 붙이고 긴 옵션 인수에는 두 개의 대시(--)를 붙입니다. 옵션은 유틸리티가 명령을 수행하는 방법을 제어합니다. 옵션도 대소문자를 구분합니다. 기능의 ON, OFF를 전환하는 부울 옵션을 제외한 대부분의 옵션에는 인수 값이 필요합니다. 피연산자는 인수 값 뒤에 붙이며 공백, 탭 또는 이중 대시(--)로 구분됩니다. asadmin 유틸리티는 옵션 및 해당 값 뒤에 나오는 모든 부분을 피연산자로 간주합니다.

예 19–1 구문 예

```
asadmin command [-short_option] [short_option_argument]* [--long_option
[long_option_argument]* [operand]*

asadmin create-profiler -u admin --passwordfile password.txt myprofiler
```

Solaris 플랫폼에서 Application Server asadmin 유틸리티 명령에 대한 설명서 페이지에 액세스하려면 MANPATH 환경 변수에 $AS_INSTALL/man을 추가하십시오.

--help 옵션을 호출하면 모든 asadmin 유틸리티 명령에 대한 전체 사용법 정보를 얻을 수 있습니다. 명령을 지정하면 해당 명령에 대한 사용법 정보가 표시됩니다. 명령을 지정하지 않고 --help 옵션을 사용하면 사용 가능한 모든 명령 목록이 표시됩니다.

예 19–2 help 명령 예

asadmin --help - 일반 도움말을 표시합니다.

asadmin *command* --help - 지정된 명령에 대한 도움말을 표시합니다.

이 절은 다음 내용으로 구성되어 있습니다.

다중 모드 및 대화식 모드

asadmin 유틸리티는 명령 쉘 호출 또는 다중 명령 모드(multimode 명령으로 알려져 있음)로 사용할 수 있습니다. 명령 쉘 호출의 경우 명령 쉘에서 asadmin 유틸리티를 호출합니다. asadmin은 명령을 실행한 다음 종료됩니다. 다중 명령 모드의 경우

asadmin이 한 번 호출되면 asadmin을 종료하여 정상 명령 쉘 호출로 돌아가기 전까지 여러 번의 명령 실행이 허용됩니다. 다중 명령 모드에서 설정된 환경 변수는 multimode를 종료하기 전까지 모든 후속 명령에 사용됩니다. 파일 또는 표준 입력(파이프)에서 이전에 준비된 명령 목록을 전달하여 명령을 입력할 수 있습니다. 다중 모드 세션 중에 multimode를 호출할 수 있으며 이때 두 번째 다중 모드 환경을 종료하면 원래 다중 모드 환경으로 돌아갑니다.

asadmin 유틸리티를 대화식 또는 비대화식 모드로 실행할 수도 있습니다. 기본적으로 대화식 모드 옵션이 활성화됩니다. 대화식 모드 옵션을 활성화할 때에는 필수 인수를 입력하라는 메시지가 표시됩니다. 모든 환경에서 명령 쉘 호출로 대화식 모드 옵션을 사용할 수 있습니다. 명령 프롬프트에서 한 번에 하나의 명령을 실행하고 파일에서 명령을 multimode로 실행하는 경우 multimode로 대화식 모드 옵션을 사용할 수 있습니다. 입력 스트림에서 파이프된 경우 multimode의 명령 및 다른 프로그램에서 호출된 명령은 대화식 모드로 실행되지 않습니다.

로컬 명령

로컬 명령은 관리 서버가 없는 상태에서 실행할 수 있습니다. 그러나 사용자는 명령을 실행하고 설치 및 도메인 디렉토리에 대한 액세스 권한을 갖기 위해 도메인을 호스팅하는 시스템에 로그인해야 합니다.

해당 환경 또는 명령줄에서 로컬 또는 원격으로 실행할 수 있는 명령에 --host, --port, --user 및 --passwordfile 옵션 중 하나를 설정하면 이 명령은 원격 모드로 실행됩니다. 명령줄 또는 해당 환경에서 로컬 옵션을 설정하지 않으면 기본적으로 이 명령은 로컬로 실행됩니다.

원격 명령

원격 명령은 항상 관리 서버에 연결된 상태로 해당 관리 서버에서 실행됩니다. 관리 서버가 실행 중이어야 합니다. 모든 원격 명령에는 다음 명령 옵션이 필요합니다.

표 19–1 원격 명령의 필수 옵션

짧은 옵션	옵션	정의
-H	--host	도메인 관리 서버가 실행 중인 시스템 이름입니다. 기본값은 localhost입니다.
-p	--port	관리용 HTTP/S 포트입니다. 이 포트는 도메인을 관리할 수 있도록 사용자의 브라우저를 가리켜야합니다. 예를 들면 다음과 같습니다. http://localhost:4848. Platform Edition의 기본 포트 번호는 4848입니다.

표 19–1 원격 명령의 필수 옵션 *(계속)*

짧은 옵션	옵션	정의
-u	--user	인증된 도메인 관리 서버 관리 사용자 이름입니다. asadmin login 명령을 사용하여 도메인에 인증한 경우 이 특정 도메인에 대해서는 후속 작업 시 --user 옵션을 지정할 필요가 없습니다.
	--passwordfile	--passwordfile 옵션은 특정 형식의 비밀번호 항목이 포함된 파일의 이름을 지정합니다. 비밀번호 항목에는 AS_ADMIN_ 접두어가 있어야 하며 뒤에 대문자로 된 비밀번호 이름이 나와야 합니다. 예를 들어, 도메인 관리 서버 비밀번호를 지정하려면 다음 형식의 항목을 사용합니다. AS_ADMIN_PASSWORD=*password*. 여기서, *password*는 실제 관리자 비밀번호입니다. 지정할 수 있는 다른 비밀번호에는 AS_ADMIN_PASSWORD, AS_ADMIN_USERPASSWORD 및 AS_ADMIN_ALIASPASSWORD, AS_ADMIN_MAPPEDPASSWORD가 있습니다. 모든 원격 명령 실행 시에는 --passwordfile이나 asadmin login을 사용하거나 명령 프롬프트에서 대화식 방법을 사용하여 관리 비밀번호를 지정함으로써 도메인 관리 서버에 인증해야 합니다. asadmin login 명령은 관리 비밀번호를 지정하는 경우에만 사용할 수 있습니다. 원격 명령에 다른 비밀번호를 지정해야 하는 경우에는 --passwordfile을 사용하거나 명령 프롬프트에서 해당 비밀번호를 입력합니다. asadmin login 명령을 사용하여 도메인에 인증한 경우 이 특정 도메인에 대해서는 후속 작업 시 --passwordfile 옵션을 통해 관리 비밀번호를 지정할 필요가 없습니다. 그러나 이 절차는 AS_ADMIN_PASSWORD 옵션에만 적용됩니다. 하지만 update-file-user와 같은 개별 명령에서 요구하는 경우 AS_ADMIN_USERPASSWORD와 같은 다른 비밀번호를 입력해야 합니다. 보안상 환경 변수로 지정된 비밀번호는 asadmin으로 읽을 수 없습니다.
-s	--secure	true로 설정되면 SSL/TLS를 사용하여 도메인 관리 서버와 통신합니다.
-I	--interactive	true로 설정되면(기본값) 필수 비밀번호와 사용자 옵션을 입력하라는 메시지만 표시됩니다.
-t	--terse	일반적으로 서술형 문장을 배제하고 스크립트 사용에 적합한 형식이 되도록 출력 데이터를 간결화합니다. 기본값은 false입니다.
-e	--echo	true로 설정되면 명령줄 명령이 표준 출력에 표시됩니다. 기본값은 false입니다.
-h	--help	명령에 대한 도움말 텍스트를 표시합니다.

비밀번호 파일

보안상의 목적으로 명령줄에서 비밀번호를 입력하는 대신 명령에 대한 비밀번호를 파일에서 설정할 수 있습니다. `--passwordfile` 옵션을 사용하면 비밀번호가 포함된 파일이 사용됩니다. 파일의 유효한 내용은 다음과 같습니다.

예 19–3 passwordfile 내용

```
AS_ADMIN_PASSWORD=value
AS_ADMIN_ADMINPASSWORD=value
AS_ADMIN_USERPASSWORD=value
AS_ADMIN_MASTERPASSWORD=value
```

다중 모드 명령

`multimode` 명령을 사용하여 `asadmin` 명령을 처리할 수 있습니다. 명령줄 인터페이스에서 명령을 입력하라는 메시지를 표시하고, 입력된 명령을 실행하며, 명령에 대한 결과를 표시합니다. 그런 다음 다시 다음 명령을 입력하라는 메시지를 표시합니다. 또한, 이 모드에서 설정된 모든 `asadmin` 옵션 이름은 이에 대한 모든 후속 명령에 사용됩니다. 사용자 환경을 설정할 수 있으며 "exit" 또는 "quit"를 입력하여 `multimode`를 종료하기 전까지 명령을 실행할 수 있습니다.파일 또는 표준 입력(파이프)에서 이전에 준비한 명령 목록을 전달하여 명령을 제공할 수도 있습니다. **다중 모드** 세션 내에서 `multimode`를 호출할 수 있으며, 두 번째 **다중 모드** 환경을 종료하면 원래 **다중 모드** 환경으로 돌아갑니다.

다중 모드를 호출하려면 `asadmin multimode`를 입력합니다.

나열, 가져오기 및 설정 명령

`asadmin list`, `get` 및 `set` 명령을 함께 사용하면 이름이 점으로 구분된 Application Server의 계층에 대한 탐색 메커니즘이 제공됩니다. 다음과 같은 두 가지 계층이 있으며 이러한 명령은 두 계층 모두에서 작동합니다. **구성** 및 **모니터링**. `list` 명령은 읽기 전용의 또는 수정 가능한 속성을 갖는 관리 구성 요소의 점으로 구분된 정규화된 이름을 제공합니다.

구성 계층은 수정 가능한 속성을 제공하지만, **모니터링** 계층은 읽기 전용의 관리 구성 요소 속성만을 제공합니다. **구성** 계층은 어느 정도 도메인의 스키마 문서를 기반으로 하지만, **모니터링** 계층은 조금 다릅니다.

`list` 명령을 사용하면 원하는 계층의 특정 관리 구성 요소에 연결할 수 있습니다. 그런 다음 `get` 및 `set` 명령을 호출하면 이름 및 값을 얻거나 관리 구성 요소의 속성 값을 직접 설정할 수 있습니다. 와일드카드(*) 옵션을 사용하면 점으로 구분된 정규화된 해당 이름과 일치하는 모든 항목을 불러올 수 있습니다.

Application Server의 점으로 구분된 이름은 “.” (마침표)를 구분자로 사용하여 전체 이름을 구성하는 부분을 구분합니다. 이는 UNIX 파일 시스템에서 파일에 대한 절대 경로 이름의 수준을 구분하는 데 “/” 문자를 사용하는 것과 비슷합니다. get, set 및 list 명령에서 허용되는 점으로 구분된 이름을 구성할 때 다음 규칙이 적용됩니다. 특정 명령에는 일부 추가 어휘가 적용됩니다.

- .(마침표)는 항상 이름의 순차적인 두 부분을 구분합니다.
- 이름을 구성하는 부분은 대개 응용 프로그램 서버 하위 시스템 및/또는 특정 인스턴스를 식별합니다. 예를 들면 다음과 같습니다. web-container, log-service, thread-pool-1 등.
- 이름을 구성하는 부분 자체에 .(마침표)가 포함되어 있는 경우에는 “.” 가 구분자로 사용되지 않도록 앞에 \(백슬래시)를 붙여 이스케이프 처리해야 합니다.
- *(별표)는 위치와 상관없이 점으로 구분된 이름의 일부로 사용할 수 있으며 정규 표현식에서 와일드카드 문자의 역할을 합니다. 또한 *는 점으로 구분된 이름의 모든 부분을 축약할 수 있습니다. 예를 들어, "<classname>this.is.really.long.hierarchy</classname>"과 같이 길이가 긴 점으로 구분된 이름을 "<classname>th*.hierarchy</classname>"으로 축약할 수 있습니다.하지만 .는 항상 이름을 구성하는 부분을 구분하는 역할을 합니다.
- Solaris의 경우 *가 포함된 명령을 옵션 값 또는 피연산자로 실행할 때 따옴표가 필요합니다.
- 점으로 구분된 이름에 대한 최상위 수준의 스위치는 --monitor 또는 -m이며 해당 명령줄에 별도로 지정됩니다. 이 스위치 존재의 유무는 응용 프로그램 서버 관리를 위해 모니터링 및 구성 계층 중 하나를 선택했다는 것을 의미합니다.
- 점으로 구분된 전체 이름을 정확히 알고 있어 와일드카드 문자를 사용하지 않는 경우에는 list 및 get/set에 어휘상 약간의 차이점이 있습니다.
  - list 명령의 경우 이 점으로 구분된 전체 이름은 계층에 있는 부모 노드의 전체 이름으로 간주됩니다. 이 이름을 list 명령에 사용하면 해당 수준에서 직계 자식의 이름이 반환됩니다. 예를 들어, list server.applications.web-module을 사용하면 도메인 또는 기본 서버에 배포된 모든 웹 모듈이 나열됩니다.
  - get 및 set 명령의 경우 이 점으로 구분된 전체 이름은 노드(이에 대한 점으로 구분된 이름은 이 점으로 구분된 이름의 마지막 부분을 제거하여 얻어진 이름임) 속성의 정규화된 이름으로 간주되며, 이 명령을 실행하면 해당 속성 값을 가져오고 설정할 수 있습니다. 이러한 속성이 있는 경우 이 설정은 true입니다. 계층에 있는 특정 노드의 속성 이름을 찾기 위해서는 와일드카드 문자 *를 사용해야 하기 때문에 이 경우에는 시작할 수 없습니다. 예를 들어, server.applications.web-module.JSPWiki.context-root*는 도메인 또는 기본 서버에 배포된 웹 응용 프로그램의 컨텍스트 루트를 반환합니다.

list 명령은 이 세 명령의 탐색 기능의 기반입니다. 특정 응용 프로그램 서버 하위 시스템의 속성을 set 또는 get하려면 점으로 구분된 해당 이름을 알고 있어야 합니다. list는 해당 하위 시스템의 점으로 구분된 이름을 찾을 수 있도록 해주는 명령입니다. 예를 들어, 대용량 파일 시스템의 특정 파일에서 /로 시작하는 수정된 날짜(속성)를 찾으려는 경우 이 명령을 사용할 수 있습니다. 먼저 파일 시스템에서 해당 파일의 위치를

찾은 다음 해당 속성을 확인할 수 있으므로 Application Server의 계층을 인식하는 첫 번째 두 명령은 `* list "*"` 및 `<command>* list * --monitor`가 됩니다. 이 명령에 대한 정렬된 출력을 식별하려면 `get`, `set` 또는 `list` 명령 설명서 페이지를 참조하십시오.

서버 라이프사이클 명령

서버 라이프사이클 명령은 도메인, 서비스(DAS) 또는 인스턴스를 만들고 삭제하거나 시작하고 중지하는 명령입니다.

표 19–2 서버 라이프사이클 명령

명령	정의
`create-service`	자동 부트 시 DAS 시작을 구성합니다. Solaris 10의 경우 이 명령은 SMF(Service Management Facility)를 사용합니다. 이 명령은 로컬 명령으로 수퍼유저 권한을 가진 OS 수준의 사용자로 실행해야 합니다. 이 명령은 Solaris 10에서만 사용할 수 있습니다. 서비스가 만들어지면 사용자는 이 서비스를 시작, 활성화, 비활성화, 삭제 또는 중지해야 합니다. DAS는 수퍼유저가 액세스할 수 있는 폴더에 저장되어야 합니다. 구성은 네트워크 파일 시스템에 저장할 수 없습니다. DAS의 구성이 있는 폴더를 소유한 OS 수준의 사용자가 제어할 수 있도록 서비스가 만들어집니다. 이 명령을 실행하려면 `solaris.smf.*` 권한이 있어야 합니다.
`create-domain`	도메인 구성을 만듭니다. 도메인은 관리 이름 공간입니다. 모든 도메인에는 구성이 있으며 이 구성은 파일 집합에 저장되어 있습니다. Application Server의 해당 설치에 각 고유 관리 아이디가 있는 도메인을 얼마든지 만들 수 있습니다. 도메인은 다른 도메인과 독립적으로 존재할 수 있습니다. 해당 시스템의 `asadmin` 유틸리티에 액세스할 수 있는 사용자는 도메인을 만들 수 있으며 해당 구성을 선택한 폴더에 저장할 수 있습니다. 기본적으로 도메인 구성은 *install_dir*`/domains` 디렉토리에 만들어집니다. 이 위치를 대체하여 구성을 다른 곳에 저장할 수 있습니다.
`delete-domain`	명명된 도메인을 삭제합니다. 도메인이 이미 있어야 하며 중지해야 합니다.
`start-domain`	도메인을 시작합니다. 도메인 디렉토리가 지정되지 않은 경우에는 기본 *install_dir*`/domains` 디렉토리의 도메인이 시작됩니다. 도메인이 두 개 이상 있으면 *domain_name* 피연산자를 지정해야 합니다.
`stop-domain`	지정된 도메인의 Domain Administration Server를 중지합니다.
`restore-domain`	백업 디렉토리에서 도메인의 파일을 복원합니다.
`list-domains`	도메인을 나열합니다. 도메인 디렉토리가 지정되지 않은 경우에는 기본 *install_dir*`/domains` 디렉토리의 도메인이 나열됩니다. 도메인이 두 개 이상 있으면 *domain_name* 피연산자를 지정해야 합니다.
`backup-domain`	명명된 도메인의 파일을 백업합니다.

표 19–2 서버 라이프사이클 명령 *(계속)*

명령	정의
list-backups	백업 저장소의 모든 백업에 대한 상태 정보를 표시합니다.
shutdown	관리 서버 및 실행 중인 모든 인스턴스를 적절하게 중지합니다. 관리 서버를 다시 시작하려면 수동으로 시작해야 합니다.

List 및 Status 명령

List 및 Status 명령은 배포된 구성 요소의 상태를 표시합니다.

표 19–3 List 및 Status 명령

명령	정의
show-component-status	배포된 구성 요소의 상태를 가져옵니다. 상태는 서버에서 반환된 문자열 표현입니다. 상태 문자열에는 "*app-name*의 상태: enabled" 또는 "*app-name*의 상태: disabled"를 사용할 수 있습니다.
list-components	배포된 모든 Java EE 5 구성 요소를 나열합니다. --type 옵션이 지정되지 않은 경우 모든 구성 요소가 나열됩니다.
list-sub-components	배포된 모듈 또는 배포된 응용 프로그램의 모듈에 있는 EJB 또는 서블릿을 나열합니다. 모듈이 식별되지 않으면 모든 모듈이 나열됩니다.

배포 명령

배포 명령은 응용 프로그램을 배포하거나 클라이언트 스텁을 가져옵니다.

표 19–4 배포 명령

명령	정의
deploy	엔터프라이즈 응용 프로그램, 웹 응용 프로그램, EJB 모듈, 커넥터 모듈 또는 응용 프로그램 클라이언트 모듈을 배포합니다. 구성 요소가 이미 배포되었거나 존재하는 경우 --force 옵션을 true로 설정하면 구성 요소가 강제로 재배포됩니다.
deploydir	응용 프로그램을 개발 디렉토리에서 직접 배포합니다. Java EE 사양을 준수하는 적합한 디렉토리 계층 및 배포 설명자가 배포 디렉토리에 있어야 합니다.

표 19–4 배포 명령 (계속)

명령	정의
get-client-stubs	AppClient 독립 실행형 모듈 또는 AppClient 모듈이 포함된 응용 프로그램의 클라이언트 스텁 JAR 파일을 서버 시스템에서 로컬 디렉토리로 가져옵니다. 이 명령을 실행하기 전에 응용 프로그램 또는 모듈을 배포해야 합니다. --retrieve 옵션을 사용하여 deploy 명령의 일부로 클라이언트 스텁을 가져올 수도 있습니다.
undeploy	배포된 지정 구성 요소를 제거합니다.

Message Queue 관리 명령

Message Queue 관리 명령을 사용하면 JMS 대상을 관리할 수 있습니다.

표 19–5 Message Queue 명령

명령	정의
create-jmsdest	JMS 물리적 대상을 만듭니다. 물리적 대상과 함께 create-jms-resource 명령을 사용하여 물리적 대상을 지정하는 Name 등록 정보가 있는 JMS 대상 자원을 만들 수 있습니다.
delete-jmsdest	지정된 JMS 대상을 제거합니다.
flush-jmsdest	지정된 대상의 JMS 서비스 구성에서 물리적 대상의 메시지를 제거합니다.
list-jmsdest	JMS 물리적 대상을 나열합니다.
jms-ping	JMS 서비스(JMS 공급자로도 알려져 있음)가 실행 중인지 확인합니다. Application Server를 시작할 때 기본적으로 JMS 서비스가 시작됩니다. 이 명령은 JMS 서비스 내의 기본 JMS 호스트만 ping합니다. 기본 제공 JMS 서비스를 ping할 수 없는 경우 오류 메시지가 표시됩니다.

자원 관리 명령

자원 명령을 사용하면 응용 프로그램에 사용되는 다양한 자원을 관리할 수 있습니다.

표 19–6 자원 관리 명령

명령	정의
create-jdbc-connection-pool	지정된 JDBC 연결 풀 이름을 사용하여 새 JDBC 연결 풀을 등록합니다.

표 19–6 자원 관리 명령 *(계속)*

명령	정의
`delete-jdbc-connection-pool`	JDBC 연결 풀을 삭제합니다. 피연산자가 삭제할 JDBC 연결 풀을 식별합니다.
`list-jdbc-connection-pools`	만들어진 JDBC 연결 풀을 가져옵니다.
`create-jdbc-resource`	새 JDBC 자원을 만듭니다.
`delete-jdbc-resource`	지정된 JNDI 이름을 가진 JDBC 자원을 제거합니다.
`list-jdbc-resources`	만들어진 JDBC 자원 목록을 표시합니다.
`create-jms-resource`	JMS(Java Message Service) 연결 팩토리 자원 또는 JMS 대상 자원을 만듭니다.
`delete-jms-resource`	지정된 JMS 자원을 제거합니다.
`list-jms-resources`	기존 JMS 자원(대상 및 연결 팩토리 자원)을 나열합니다.
`create-jndi-resource`	JNDI 자원을 등록합니다.
`delete-jndi-resource`	지정된 JNDI 이름을 가진 JNDI 자원을 제거합니다.
`list-jndi-resources`	기존의 모든 JNDI 자원을 확인합니다.
`list-jndi-entries`	JNDI 트리를 찾아 쿼리합니다.
`create-javamail-resource`	JavaMail 세션 자원을 만듭니다.
`delete-javamail-resource`	지정된 JavaMail 세션 자원을 제거합니다.
`list-javamail-resources`	기존 JavaMail 세션 자원을 나열합니다.
`create-persistence-resource`	지속성 자원을 등록합니다.
`delete-persistence-resource`	지속성 자원을 제거합니다. 지속성 자원을 삭제할 때 이 명령은 `create-persistence-resource` 명령을 사용하여 만든 JDBC 자원도 제거합니다.
`list-persistence-resources`	모든 지속성 자원을 표시합니다.
`create-custom-resource`	사용자 정의 자원을 만듭니다. 사용자 정의 자원은 `javax.naming.spi.ObjectFactory` 인터페이스를 구현하는 사용자 정의 서버측 자원 객체 팩토리를 지정합니다.
`delete-custom-resource`	사용자 정의 자원을 제거합니다.
`list-custom-resources`	사용자 정의 자원을 나열합니다.
`create-connector-connection-pool`	지정된 연결 풀 이름을 사용하여 새 커넥터 연결 풀을 추가합니다.
`delete-connector-connection-pool`	피연산자 `connector_connection_pool_name`을 사용하여 지정한 커넥터 연결 풀을 제거합니다.

표 19–6 자원 관리 명령 *(계속)*

명령	정의
`list-connector-connection-pools`	만들어진 커넥터 연결 풀을 나열합니다.
`create-connector-resource`	지정된 JNDI 이름을 가진 커넥터 자원을 등록합니다.
`delete-connector-resource`	지정된 JNDI 이름을 가진 커넥터 자원을 제거합니다.
`list-connector-resources`	모든 커넥터 자원을 가져옵니다.
`create-admin-object`	지정한 JNDI 이름을 가진 관리 대상 객체를 추가합니다.
`delete-admin-object`	지정된 JNDI 이름을 가진 관리 대상 객체를 제거합니다.
`list-admin-objects`	모든 관리 대상 객체를 나열합니다.
`create-resource-adapter-config`	커넥터 모듈에 대한 구성 정보를 만듭니다.
`delete-resource-adapter-config`	`domain.xml`에 만들어진 커넥터 모듈에 대한 구성 정보를 삭제합니다.
`list-resource-adapter-configs`	`domain.xml`의 커넥터 모듈에 대한 구성 정보를 나열합니다.
`add-resources`	지정된 XML 파일에 명명된 자원을 만듭니다. *xml_file_path*는 생성할 자원이 포함될 XML 파일 경로입니다. `resources.xml` 파일에서 `DOCTYPE`은 *install_dir*`/lib/dtds/sun-resources_1_2.dtd`로 지정되어야 합니다.
`ping-connection-pool`	연결 풀을 JDBC 연결 풀 및 커넥터 연결 풀 모두에 사용할 수 있는지 테스트합니다. 예를 들어, 나중에 배포될 것으로 예상되는 응용 프로그램에 새 JDBC 연결 풀을 만든 경우 응용 프로그램을 배포하기 전에 JDBC 풀이 이 명령으로 테스트됩니다. 연결 풀을 ping하기 전에 인증이 있는 연결 풀을 만들어야 하며 Application Server 또는 데이터베이스가 시작되었는지 확인해야 합니다.

Application Server 구성 명령

구성 명령을 사용하면 Application Server 작업을 구성할 수 있습니다. 이 절은 다음 내용으로 구성되어 있습니다.

일반 구성 명령

이 명령을 사용하면 Application Server 구성 요소의 구성을 관리할 수 있습니다.

표 19–7 일반 구성 명령

명령	정의
`enable`	지정된 구성 요소를 활성화합니다. 구성 요소가 이미 활성화되어 있는 경우 구성 요소가 다시 활성화됩니다. 구성 요소를 활성화하려면 배치되어 있어야 합니다. 구성 요소가 배포되어 있지 않은 경우에는 오류 메시지가 반환됩니다.
`disable`	명명된 구성 요소를 즉시 비활성화합니다. 구성 요소가 배포되어 있어야 합니다. 구성 요소가 배포되어 있지 않은 경우에는 오류 메시지가 반환됩니다.
`export`	후속 명령 환경을 위한 자동 내보내기 변수 이름을 표시합니다. 모든 후속 명령은 명령을 `unset`하거나 `multimode`를 종료하지 않는 한 지정된 변수 이름 값을 사용합니다.
`get`	속성 이름 및 값을 가져옵니다.
`set`	구성 가능한 속성 값을 한 개 이상 설정합니다.
`list`	구성 가능한 요소를 나열합니다. Solaris의 경우 *가 포함된 명령을 옵션 값 또는 피연산자로 실행할 때 따옴표가 필요합니다.
`unset`	다중 모드 환경에서 설정한 변수를 한 개 이상 제거합니다. 변수 및 연관된 값이 환경에서 제거됩니다.

HTTP, IIOP 및 SSL Listener 명령

HTTP 및 IIOP Listener 명령은 Listener 관리 작업을 도와줍니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–8 IIOP Listener 명령

명령	정의
`create-http-listener`	새 HTTP Listener를 추가합니다.
`delete-http-listener`	지정된 HTTP Listener를 제거합니다.
`list-http-listeners`	기존 HTTP Listener를 나열합니다.
`create-iiop-listener`	IIOP Listener를 만듭니다.
`delete-iiop-listener`	지정된 IIOP Listener를 제거합니다.

표 19–8 IIOP Listener 명령 *(계속)*

명령	정의
`list-iiop-listeners`	기존 IIOP Listener를 나열합니다.
`create-ssl`	선택한 HTTP Listener, IIOP Listener 또는 IIOP 서비스에 SSL 요소를 만들고 구성하여 해당 Listener/서비스의 보안 통신을 활성화합니다.
`delete-ssl`	선택한 HTTP Listener, IIOP Listener 또는 IIOP 서비스의 SSL 요소를 삭제합니다.

라이프사이클 및 감사 모듈 명령

라이프사이클 및 감사 모듈 명령은 라이프사이클 모듈 및 감사 기능을 구현하는 선택 사항의 플러그인 모듈을 제어하는 데 도움을 줍니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–9 라이프사이클 모듈 명령

명령	정의
`create-lifecycle-module`	라이프사이클 모듈을 만듭니다. 라이프사이클 모듈은 Application Server 환경 내에서 기간이 짧거나 긴 Java 기반의 작업을 실행하는 수단을 제공합니다.
`delete-lifecycle-module`	지정된 라이프사이클 모듈을 제거합니다.
`list-lifecycle-modules`	기존 라이프사이클 모듈을 나열합니다.
`create-audit-module`	감사 기능을 구현하는 플러그인 모듈에 명명된 감사 모듈을 추가합니다.
`delete-audit-module`	명명된 감사 모듈을 제거합니다.
`list-audit-modules`	모든 감사 모듈을 나열합니다.

프로필러 및 JVM 옵션 명령

프로필러 및 JVM 옵션 명령을 사용하면 프로필러를 관리하고 해당 요소를 제어할 수 있습니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–10 프로필러 및 JVM 옵션 명령

명령	정의
`create-profiler`	프로필러 요소를 만듭니다. Java 구성의 프로필러 요소는 서버 인스턴스를 특정 프로필러와 연결합니다. 프로필러를 변경하려면 서버를 다시 시작해야 합니다.

표 19–10 프로필러 및 JVM 옵션 명령 (계속)

명령	정의
delete-profiler	지정된 프로필러 요소를 삭제합니다. Java 구성의 프로필러 요소는 서버 인스턴스를 특정 프로필러로 묶습니다. 프로필러를 변경하려면 서버를 다시 시작해야 합니다.
create-jvm-option	Java 구성의 JVM 옵션 또는 domain.xml 파일의 프로필러 요소를 만듭니다. 프로필러에 만든 JVM 옵션은 특정 프로필러를 실행하는 데 필요한 설정을 기록할 때 사용합니다. 새로 만든 JVM 옵션을 적용하려면 서버를 다시 시작해야 합니다.
delete-jvm-option	Java 구성의 JVM 옵션 또는 domain.xml 파일의 프로필러 요소를 제거합니다.

가상 서버 명령

가상 서버 명령을 사용하면 해당 요소를 제어할 수 있습니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–11 가상 서버 명령

명령	정의
create-virtual-server	명명된 가상 서버를 만듭니다. Application Server를 가상화하면 여러 호스트 주소를 수신하는 단일 HTTP 서버 프로세스를 통해 여러 URL 도메인을 서비스할 수 있습니다. 응용 프로그램이 두 개의 가상 서버에서 사용 가능한 경우 이 가상 서버는 계속해서 동일한 물리적 자원 풀을 공유합니다.
delete-virtual-server	지정된 가상 서버 아이디를 가진 가상 서버를 제거합니다.
list-virtual-server	기존 가상 서버를 나열합니다.

스레드 풀 명령

스레드 풀 명령을 사용하면 해당 요소를 제어할 수 있습니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–12 스레드 풀 명령

명령	정의
`create-threadpool`	지정된 이름을 사용하여 스레드 풀을 만듭니다. 풀 내 스레드의 최대 및 최소 개수, 작업 대기열 수 및 스레드의 유휴 시간 초과를 지정할 수 있습니다. 만들어진 스레드 풀은 IIOP 요청을 서비스할 때 및 자원 어댑터가 작업 관리 요청을 서비스할 때 사용할 수 있습니다. 만들어진 스레드 풀은 여러 자원 어댑터에 사용할 수 있습니다.
`delete-threadpool`	명명된 아이디를 가진 스레드 풀을 제거합니다.
`list-threadpools`	모든 스레드 풀을 나열합니다.

트랜잭션 및 타이머 명령

트랜잭션 및 타이머 명령을 사용하면 트랜잭션 및 타이머 하위 시스템을 제어할 수 있으며 실행 중인 트랜잭션을 일시 중지할 수도 있습니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–13 트랜잭션 명령

명령	정의
`freeze-transaction`	실행 중인 모든 트랜잭션이 일시 중지되도록 트랜잭션 하위 시스템을 중단합니다. 실행 중인 트랜잭션을 롤백하기 전에 이 명령을 호출합니다. 이미 중단된 트랜잭션 하위 시스템에서 이 명령을 호출하는 경우 아무런 영향을 주지 않습니다.
`unfreeze-transaction`	일시 중지된 모든 실행 트랜잭션을 다시 시작합니다. 이미 중단된 트랜잭션에서 이 명령을 호출합니다.
`recover-transactions`	보류 중인 트랜잭션을 수동으로 복구합니다.
`rollback-transaction`	명명된 트랜잭션을 롤백합니다.
`list-timers`	특정 서버 인스턴스가 소유한 타이머를 나열합니다.

사용자 관리 명령

이 사용자 명령을 사용하면 파일 영역 인증에서 지원되는 사용자를 관리할 수 있습니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–14 사용자 관리 명령

명령	정의
`create-file-user`	지정된 사용자 이름, 비밀번호 및 그룹을 사용하여 키 파일에 항목을 만듭니다. 콜론(:)으로 그룹을 구분하여 여러 그룹을 만들 수 있습니다.
`delete-file-user`	키 파일에서 지정된 사용자 이름을 가진 항목을 삭제합니다.
`update-file-user`	지정된 `user_name`, `user_password` 및 그룹을 사용하여 키 파일의 기존 항목을 업데이트합니다. 콜론(:)으로 그룹을 구분하여 여러 그룹을 입력할 수 있습니다.
`list-file-users`	파일 영역 인증에서 지원되는 파일 사용자 목록을 만듭니다.
`list-file-groups`	파일 영역 인증에서 지원되는 사용자 및 그룹을 나열합니다. 이 명령은 파일 사용자가 사용할 수 있는 그룹을 나열합니다.

모니터링 데이터 명령

모니터링 데이터 명령을 사용하면 서버를 모니터할 수 있습니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–15 모니터링 데이터 명령

명령	정의
`start-callflow-monitoring`	웹 컨테이너, EJB 컨테이너 및 JDBC에서 데이터를 수집하고 상관 관계를 지정하여 요청의 전체 호출 흐름/경로를 제공합니다. `callflow-monitoring`이 ON으로 지정된 경우에만 데이터를 수집할 수 있습니다.
`stop-callflow-monitoring`	요청의 호출 흐름 정보 수집을 비활성화합니다.

규칙 명령

규칙 명령을 사용하면 서버의 규칙을 관리할 수 있습니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–16 규칙 명령

명령	정의
`create-management-rule`	새 관리 규칙을 만들어 Application Server 설치 및 배포된 응용 프로그램을 지능적으로 자체 관리합니다.
`delete-management-rule`	지정된 관리 규칙을 제거합니다.

표 19–16 규칙 명령 *(계속)*

명령	정의
list-management-rules	사용 가능한 관리 규칙을 나열합니다.

데이터베이스 명령

데이터베이스 명령을 사용하면 Java DB 데이터베이스(Apache Derby를 기반으로 함)를 시작하고 중지할 수 있습니다. 이 명령은 로컬 모드에서만 지원됩니다.

표 19–17 데이터베이스 명령

명령	정의
start-database	Application Server와 함께 사용할 수 있는 Java DB 서버를 시작합니다. Application Server에 배포된 응용 프로그램 작업을 수행하는 경우에만 이 명령을 사용합니다.
stop-database	Java DB 서버 프로세스를 중지합니다. Java DB 서버는 Application Server와 함께 사용할 수 있습니다.

진단 및 로깅 명령

진단 및 로깅 명령은 Application Server의 문제를 해결하는 데 도움을 줍니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–18 진단 및 로깅 명령

명령	정의
generate-diagnostic-report	응용 프로그램 서버 인스턴스의 구성 세부 정보, 로깅 세부 정보 또는 프로세스별 정보와 같은 응용 프로그램 서버 설치 세부 정보에 대한 탐색 링크나 포인터가 포함된 HTML 보고서를 생성합니다.
generate-jvm-report	Domain Administration Service를 포함한 해당 대상 인스턴스의 스레드(스택 추적 덤프), 클래스 및 메모리를 표시합니다. 이 명령은 응용 프로그램 서버 인스턴스 프로세스에만 사용됩니다. 이 명령은 응용 프로그램 서버 프로세스에 ctrl+break 또는 kill -3 신호를 보내는 것과 같은 이전 기술을 대체합니다. 대상 서버 인스턴스가 실행 중이 아니면 이 명령은 작동하지 않습니다.
display-error-statistics	마지막으로 서버가 다시 시작된 후 server.log에 기록된 심각도 및 경고에 대한 요약 목록을 표시합니다.
display-error-distribution	모듈 수준에서 인스턴스 오류 server.log의 배포를 표시합니다.™

표 19–18 진단 및 로깅 명령 *(계속)*

명령	정의
display-log-records	지정된 모듈에 대한 모든 오류 메시지를 지정된 타임스탬프에 표시합니다.

웹 서비스 명령

웹 서비스 명령을 사용하면 배포된 웹 서비스를 모니터하고 변환 규칙을 관리할 수 있습니다.

표 19–19 웹 서비스 명령

명령	정의
configure-webservice-management	배포된 웹 서비스 종점의 maxhistorysize 속성 또는 모니터링을 구성합니다.
create-transformation-rule	웹 서비스 작업에 적용할 수 있는 XSLT 변환 규칙을 만듭니다. 이 규칙은 요청 및 응답에 적용할 수 있습니다.
delete-transformation-rule	지정된 웹 서비스의 XSLT 변환 규칙을 삭제합니다.
list-transformation-rules	지정된 웹 서비스의 모든 변환 규칙을 적용된 순서대로 나열합니다.
publish-to-registry	웹 서비스 아티팩트를 레지스트리 서버에 게시합니다.
unpublish-from-registry	레지스트리 서버의 웹 서비스 아티팩트 게시를 취소합니다.
list-registry-locations	구성된 웹 서비스 항목의 액세스 지점에 대한 목록을 표시합니다.

보안 서비스 명령

이 보안 명령을 사용하면 커넥터 연결 풀에 대한 보안 매핑을 제어할 수 있습니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–20 보안 명령

명령	정의
`create-connector-security-map`	지정된 커넥터 연결 풀에 대한 보안 맵을 만듭니다. 보안 맵이 없는 경우 새 맵이 만들어집니다. 또한, 이 명령을 사용하여 응용 프로그램(principal 또는 사용자 그룹)의 호출자 아이디를 컨테이너 관리 트랜잭션 기반 시나리오의 적절한 EIS(Enterprise Information System) principal로 매핑합니다. 한 개 이상의 명명된 보안 맵을 커넥터 연결 풀에 연관시킬 수 있습니다. 커넥터 보안 맵 구성은 모든 사용자 또는 사용자 그룹을 나타낼 수 있도록 와일드카드 문자(*) 사용을 지원합니다. 이 명령을 성공적으로 실행하려면 먼저 커넥터 연결 풀을 만들어야 합니다. EIS는 조직 데이터를 보관하는 시스템입니다. 이는 메인프레임, 메시징 시스템, 데이터베이스 시스템 또는 응용 프로그램일 수 있습니다.
`delete-connector-security-map`	지정된 커넥터 연결 풀에 대한 보안 맵을 삭제합니다.
`update-connector-security-map`	지정된 커넥터 연결 풀에 대한 보안 맵을 수정합니다.
`list-connector-security-maps`	지정된 커넥터 연결 풀에 속한 보안 맵을 나열합니다.
`create-message-security-provider`	관리자가 지정된 메시지 계층에 대한 `provider-config` 하위 요소를 만들 수 있도록 활성화합니다(Application Server에 대한 매개 변수 및 등록 정보를 지정하는 `domain.xml` 파일의 `message-security-config` 요소).
`delete-message-security-provide`	관리자가 지정된 메시지 계층에 대한 `provider-config` 하위 요소를 삭제할 수 있도록 활성화합니다(Application Server에 대한 매개 변수 및 등록 정보를 지정하는 `domain.xml` 파일의 `message-security-config` 요소).
`list-message-security-providers`	관리자가 지정된 메시지 계층에 대한 모든 보안 메시지 공급자(`provider-config` 하위 요소)를 나열할 수 있도록 활성화합니다(`domain.xml`의 `message-security-config` 요소).
`create-auth-realm`	명명된 인증 영역을 추가합니다.
`delete-auth-realm`	명명된 인증 영역을 제거합니다.
`list-auth-realms`	기존 인증 영역을 나열합니다.

비밀번호 명령

비밀번호 명령을 사용하면 비밀번호를 관리할 수 있으며 Application Server의 보안을 확보할 수 있습니다.

표 19–21 비밀번호 명령

명령	정의
create-password-alias	비밀번호의 별칭을 만들고 domain.xml에 저장합니다. 별칭은 ${ALIAS=password-alias-password} 형식의 토큰입니다. 별칭 이름에 해당하는 비밀번호는 암호화된 형식으로 저장됩니다. 이 명령에는 대화식 보안 형식(모든 정보에 대해 입력 요청 메시지가 표시됨) 및 스크립트 형식(비밀번호가 명령줄에서 전파됨)이 모두 사용됩니다.
delete-password-alias	비밀번호 별칭을 삭제합니다.
update-password-alias	명명된 대상의 비밀번호 별칭 아이디를 업데이트합니다.
list-password-aliases	모든 비밀번호 별칭을 나열합니다.
change-admin-password	이 원격 명령은 관리 비밀번호를 수정합니다. 이 명령은 대화식으로 실행되며 이전 관리 비밀번호 및 새 관리 비밀번호(비밀번호 확인 포함)를 입력하라는 메시지를 표시합니다.
change-master-password	이 로컬 명령은 마스터 비밀번호를 수정합니다. 이 명령은 대화식으로 실행되며 이전 마스터 비밀번호 및 새 마스터 비밀번호를 입력하라는 메시지를 표시합니다. 서버가 중지되지 않으면 이 명령이 작동하지 않습니다.

domain.xml 검증 명령

XML 검증 명령은 domain.xml 파일의 내용을 검증합니다.

표 19–22 domain.xml 검증 명령

명령	정의
verify-domain-xml	domain.xml 파일의 내용을 검증합니다.

사용자 정의 MBean 명령

MBean 명령을 사용하면 사용자 정의 MBean을 관리하고 등록할 수 있습니다. 이 명령은 원격 모드에서만 지원됩니다.

표 19–23 사용자 정의 MBean 명령

명령	정의
create-mbean	사용자 정의 MBean을 만들고 등록합니다. 대상 MBeanServer(DAS)가 실행 중이 아니면 MBean이 등록되지 않습니다.

표 19–23 사용자 정의 MBean 명령 (계속)

명령	정의
delete-mbean	사용자 정의 MBean을 삭제합니다. 대상 MBeanServer가 실행 중인지 확인합니다.
list-mbeans	지정된 대상의 사용자 정의 MBean을 나열합니다.

기타 명령

기타 명령을 사용하면 Application Server의 다양한 측면을 관리할 수 있습니다.

표 19–24 기타 명령

명령	정의
login	도메인에 로그인합니다. 여러 시스템에 다양한 응용 프로그램 서버 도메인이 로컬로 만들어진 경우 한 시스템에서 asadmin을 호출하여 다른 곳에 있는 도메인을 원격으로 관리할 수 있습니다. 특히, 특정 시스템이 관리 클라이언트로 선택되어 여러 도메인 및 서버를 관리할 때 편리합니다. 다른 곳에 있는 도메인을 관리하는 데 사용되는 asadmin 명령을 원격 명령이라고 합니다. asadmin login 명령을 사용하면 이러한 원격 도메인을 쉽게 관리할 수 있습니다. login 명령은 대화식 모드에서만 실행됩니다. 이 명령을 실행하면 관리 사용자 이름과 비밀번호를 입력하라는 메시지가 표시됩니다. 로그인에 성공하면 .asadminpass 파일이 사용자의 홈 디렉토리에 만들어집니다. 이 파일은 --savelogin 옵션을 사용한 create-domain 명령으로 수정된 파일과 동일합니다. 이 명령을 실행하려면 도메인이 실행 중이어야 합니다.
version	버전 정보를 표시합니다. 지정된 사용자/비밀번호 및 호스트/포트를 사용하여 명령이 응용 프로그램 서버와 통신할 수 없는 경우 해당 명령은 로컬에서 버전을 검색하고 경고 메시지를 표시합니다.
help	모든 asadmin 유틸리티 명령에 대한 목록을 표시합니다. 명령을 지정하면 해당 명령에 대한 사용법 정보가 표시됩니다.
install-license	Application Server의 인증되지 않은 사용을 방지합니다. 라이센스 파일을 설치합니다.

색인