Sun Java Enterprise System 5
모니터링 설명서

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

부품 번호: 820–1591
2007년 3월

070330@17039

목차

머리말

이 설명서에서는 Sun Java™ Enterprise System 5(Java ES)의 새 모니터링 기능에 대해 설명합니다. 모니터링은 Sun Java System Monitoring Framework 2.0 및 Sun Java System Monitoring Console 1.0에 의해 구현됩니다.

이 설명서의 절차에서는 설치된 각 구성 요소에 대해 Monitoring Framework를 구성 및 활성화하는 방법과 Monitoring Console에서 모든 모니터링된 데이터를 보는 방법을 보여줍니다. 로그 파일이나 이 프레임워크를 벗어난 개별 구성 요소의 다른 모니터링 메커니즘은 이 설명서에서 다루지 않습니다.

대상

이 설명서의 대상은 다음과 같습니다.

- Java ES 배포를 위한 유지 관리 계획을 설계해야 하는 소프트웨어 설계자
- Java ES 설치 및 구성을 수행하는 시스템 관리자
- Java ES 배포를 모니터링 및 유지 관리하는 시스템 관리자 및 기술자

이 설명서를 읽기 전에

다음 절에 설명된 Java ES 설명서 세트의 문서에 익숙해야 합니다. 또한 모니터링할 Java ES 구성 요소의 디자인과 작동에 익숙해야 합니다.

이외에도 모니터링 구성 요소를 설치 및 구성하려는 경우 먼저 다른 모든 구성 요소의 설치를 완료해야 합니다. 설치나 구성을 수행하기 전에 **Sun Java Enterprise System 5 UNIX용 릴리스 노트**를 참조해야 합니다.

Java ES 설명서 세트

Java ES 설명서 세트는 배포 계획 및 시스템 설치를 설명합니다. 시스템 설명서에 대한 URL은 http://docs.sun.com/coll/1286.2 및 http://docs.sun.com/coll/1397.2입니다. Java ES에 대한 소개는 다음 표에 나열된 순서대로 설명서를 참조하십시오.

표 P–1 Java Enterprise System 설명서

문서 제목	목차
Sun Java Enterprise System 5 UNIX용 릴리스 노트 **Sun Java Enterprise System 5 Release Notes for Microsoft Windows**	알려진 문제점을 포함하여 Java ES에 대한 최신 정보를 제공합니다. 또한 구성 요소에는 릴리스 노트 모음(http://docs.sun.com/coll/1315.2)에 나열된 고유한 릴리스 노트가 있습니다.
Sun Java Enterprise System 5 기술 개요	Java ES의 기술 및 개념적 기반에 대해 소개합니다. 구성 요소, 구조, 과정 및 기능을 설명합니다.
Sun Java Enterprise System Deployment Planning Guide	Java ES을 기반으로 하는 엔터프라이즈 배포 솔루션의 계획 및 설계를 소개합니다. 배포 계획 및 설계의 기본 개념 및 원칙을 소개하고 솔루션 라이프 사이클을 설명하며 Java ES 기반 솔루션 계획 시 사용할 수 있는 고급 예 및 전략을 제공합니다.
Sun Java Enterprise System 5 설치 계획 설명서	하드웨어 구현 사양, 운영 체제 및 Java ES 배포 네트워크 기능을 개발할 수 있도록 도와줍니다. 설치 및 구성 계획에서 설명하는 구성 요소 종속성과 같은 문제들에 대해 설명합니다.
Sun Java Enterprise System 5 UNIX용 설치 설명서 **Sun Java Enterprise System 5 Installation Guide for Microsoft Windows**	Java ES를 설치하는 과정에 대해 설명합니다. 또한 설치 후에 구성 요소를 구성하고 올바르게 작동하는지 확인하는 방법을 설명합니다.
Sun Java Enterprise System 5 UNIX용 설치 참조 설명서	구성 매개 변수에 대한 추가 정보 및 구성 계획에 사용하는 워크시트를 제공하며, Solaris 운영 체제 및 Linux 운영 환경에서의 기본 디렉토리 및 포트 번호와 같은 참조 자료를 나열합니다.
Sun Java Enterprise System 5 UNIX용 업그레이드 설명서 **Sun Java Enterprise System 5 Upgrade Guide for Microsoft Windows**	이전에 설치된 버전에서 Java ES 5로 업그레이드하기 위한 지침을 제공합니다.
Sun Java Enterprise System 5 모니터링 설명서	각 제품 구성 요소의 Monitoring Framework를 설정하고 Monitoring Console을 사용하여 실시간 데이터를 확인하고 모니터링 규칙을 만드는 방법을 설명합니다.
Sun Java Enterprise System Glossary	Java ES 설명서에 사용되는 용어를 정의합니다.

기본 경로 및 파일 이름

다음 표에서는 모니터링을 구현하는 Java ES 구성 요소의 기본 경로 및 파일 이름에 대해 설명합니다.

표 P–2 기본 경로 및 파일 이름

자리 표시자	설명	기본값
mfwk-base	Monitoring Framework 공유 구성 요소가 자동으로 설치되는 디렉토리를 나타냅니다. 이 경로는 또한 구성 디렉토리의 일부로 사용됩니다.	Solaris 시스템: `/opt/SUNWmfwk` Linux 시스템: `/opt/sun/mfwk`
MConsole-base	Monitoring Console에 대해 선택된 설치 디렉토리를 나타냅니다.	Solaris 시스템: `/opt/SUNWjesmc` Linux 시스템: `/opt/sun/jesmc`
WebConsole-base	Web Console 공유 구성 요소가 자동으로 설치되는 디렉토리를 나타냅니다.	Solaris 시스템: `/etc/webconsole/console` Linux 시스템: `/etc/opt/webconsole/console`
AccessMgr-base	Sun Java System Access Manager에 대해 선택된 설치 디렉토리를 나타냅니다.	Solaris 시스템: `/opt/SUNWam` Linux 시스템: `/opt/sun/identity`
AppServer-base	Sun Java System Application Server에 대해 선택된 설치 디렉토리를 나타냅니다.	Solaris 시스템: `/opt/SUNWappserver/appserver` Linux 시스템: `/opt/sun/appserver`
CalServ-base	Sun Java System Calendar Server에 대해 선택된 설치 디렉토리를 나타냅니다.	Solaris 시스템: `/opt/SUNWics5` Linux 시스템: `/opt/sun/calendar`
DirServ-base	Sun Java System Directory Server에 대해 선택된 설치 디렉토리를 나타냅니다.	Solaris 시스템: `/opt/SUNWdsee/ds6` Linux 시스템: `/opt/sun/ds6`
IM-base	Sun Java System Instant Messaging에 대해 선택된 설치 디렉토리를 나타냅니다.	Solaris 시스템: `/opt/SUNWiim` Linux 시스템: `/opt/sun/im`
MsgServ-base	Sun Java System Messaging Server에 대해 선택된 설치 디렉토리를 나타냅니다.	Solaris 시스템: `/opt/SUNWmsgsr` Linux 시스템: `/opt/sun/messaging`
Portal-base	Sun Java System Portal Server에 대해 선택된 설치 디렉토리를 나타냅니다.	Solaris 시스템: `/opt/SUNWportal` Linux 시스템: `/opt/sun/portal`

표 P–2 기본 경로 및 파일 이름 *(계속)*

자리 표시자	설명	기본값
WebServer-base	Sun Java System Web Server에 대해 선택된 설치 디렉토리를 나타냅니다.	Solaris 시스템: `/opt/SUNWwbsvr7` Linux 시스템: `/opt/sun/webserver`

활자체 규약

다음 표에는 이 책에 사용된 활자체 규칙 변경 사항이 나와 있습니다.

표 P–3 활자체 규약

서체	의미	예
`AaBbCc123`	명령 이름, 파일, 디렉토리 및 화면 상의 컴퓨터 출력	`.login` 파일을 편집합니다. 모든 파일을 나열하려면 `ls -a`를 사용합니다. `machine_name% you have mail.`
`AaBbCc123`	화면 상의 컴퓨터 출력과 반대로 사용자가 직접 입력하는 내용입니다.	`machine_name%` **`su`** **비밀번호**:
AaBbCc123	실제 이름 또는 값으로 대체되는 자리 표시자	파일을 제거하는 명령은 `rm` *filename*입니다.
AaBbCc123	설명서 제목, 새 용어 및 강조된 용어(일부 강조된 항목은 온라인에 굵은 글씨로 표시됨)	6장의 **사용 설명서**를 읽어 보십시오. **캐시**는 로컬로 저장된 복사본입니다. 파일을 저장하지 **마십시오**.

명령 예제의 쉘 프롬프트

다음 표에는 기본 시스템 프롬프트 및 수퍼유저 프롬프트가 표시됩니다.

표 P–4 쉘 프롬프트

쉘	프롬프트
UNIX 및 Linux 시스템의 C 쉘	`machine_name%`
UNIX 및 Linux 시스템의 C 쉘 수퍼유저	`machine_name#`
UNIX 및 Linux 시스템의 Bourne 쉘 및 Korn 쉘	`$`
UNIX 및 Linux 시스템의 Bourne 쉘 및 Korn 쉘 수퍼유저	`#`

표 P–4 쉘 프롬프트 (계속)

쉘	프롬프트
Microsoft Windows 명령줄	C:\

기호 규칙

다음 표는 이 설명서에 사용될 수 있는 기호에 대해 설명합니다.

표 P–5 기호 규칙

기호	설명	예	의미
[ ]	선택 인자 및 명령 옵션을 포함합니다.	ls [-l]	-l 옵션은 선택 옵션입니다.
{ | }	선택할 수 있는 필수 명령 옵션 집합이 포함됩니다.	-d {y|n}	-d 옵션에는 y 인수 또는 n 인수를 사용해야 합니다.
${ }	변수 참조를 나타냅니다.	${com.sun.javaRoot}	com.sun.javaRoot 변수의 값 참조입니다.
-	동시에 눌러야 하는 여러 키를 결합합니다.	Control-A	Control 키를 누른 상태에서 A 키를 누릅니다.
+	연속적으로 눌러야 하는 여러 키를 결합합니다.	Ctrl+A+N	Control 키를 눌렀다 놓고 나서 나머지 키를 순서대로 누릅니다.
→	그래픽 사용자 인터페이스에서 메뉴 항목 선택을 나타냅니다.	파일 → 새로 만들기 → 템플리트	파일 메뉴에서 새로 만들기를 선택합니다. 새로 만들기 하위 메뉴에서 템플리트를 선택합니다.

문서, 지원 및 교육

Sun 웹 사이트에서는 다음과 같은 추가 자원에 관한 정보가 제공됩니다.

- 문서(http://www.sun.com/documentation/)
- 지원(http://kr.sun.com/support/)
- 교육(http://kr.sun.com/korea/)

Sun 제품 설명서 검색

docs.sun.com℠ 웹 사이트에서 Sun 제품 설명서를 검색할 뿐만 아니라 검색 필드에 다음 구문을 입력하여 검색 엔진을 사용할 수도 있습니다.

search-term `site:docs.sun.com`

예를 들어 "broker"를 검색하려면 다음과 같이 입력합니다.

```
broker site:docs.sun.com
```

다른 Sun 웹 사이트(예: java.sun.com, www.sun.com 및 developers.sun.com)를 검색 대상에 포함하려면 검색 필드에서 `docs.sun.com` 대신 `sun.com`을 사용합니다.

타사 웹사이트

본 설명서에서는 타사 URL을 참조하여 추가 관련 정보를 제공합니다.

주 – Sun은 본 설명서에서 언급된 타사 웹 사이트의 가용성 여부에 대해 책임을 지지 않습니다. 또한 해당 사이트나 리소스를 통해 제공되는 내용, 광고, 제품 및 기타 자료에 대해 어떠한 보증도 하지 않으며 그에 대한 책임도 지지 않습니다. 따라서 타사 웹 사이트의 내용, 제품 또는 리소스의 사용으로 인해 발생한 실제 또는 주장된 손상이나 피해에 대해서도 책임을 지지 않습니다.

Sun은 여러분의 의견을 환영합니다.

Sun은 설명서의 내용 개선에 노력을 기울이고 있으며, 여러분의 의견과 제안을 환영합니다. `http://docs.sun.com`에서 의견 보내기를 눌러 여러분의 의견을 제출하여 주십시오. 해당 필드에 전체 설명서 제목과 부품 번호를 입력해 주십시오. 부품 번호는 해당 설명서의 제목 페이지나 문서 맨 위 또는 URL에 있으며 7자리 또는 9자리 숫자입니다. 예를 들어, 본 설명서의 부품 번호는 820-1591입니다. 사용자 의견을 제출할 때 해당 양식에 영문 설명서 제목과 부품 번호를 입력해야 할 수도 있습니다. 본 설명서의 영문 부품 번호와 제목은 819-5081, Sun Java Enterprise System5 Monitoring Guide입니다.

◆ ◆ ◆ 1 장

1 Java ES 모니터링 개요

이 설명서에서는 Sun Java™ Enterprise System(Java ES)의 Monitoring Framework 2.0 및 Monitoring Console 1.0 구성 요소에 대해 설명합니다. 이러한 구성 요소는 함께 사용되어 릴리스 5에서 소개된 새 모니터링 기능을 구현합니다.

이 설명서의 절차는 설치된 각 구성 요소에 대해 Monitoring Framework를 활성화한 다음 Monitoring Console의 모든 모니터링된 데이터를 보는 방법을 보여줍니다. 로그 파일 또는 오류 메시지나 개별 구성 요소가 프레임워크 외부에서 구현할 수 있는 기타 모니터링 메커니즘은 이 설명서에서 다루지 않습니다. Monitoring Framework 및 Monitoring Console은 모니터링된 구성 요소를 위한 관리 또는 운영 기능을 제공하지 않습니다. 구성 요소의 관리에 대한 자세한 내용은 해당 제품의 설명서를 참조하십시오.

이 장에서는 모니터링 개념을 소개하고 Monitoring Framework의 아키텍처를 제공합니다.

이 장의 내용은 다음과 같습니다.

Monitoring Framework 및 Monitoring Console 구성 요소

Sun Java System Monitoring Framework는 구성 요소를 계측하고 관찰을 위해 해당 속성을 표시하는 인프라를 제공하며 업계 표준 CIM(Common Information Model) 사양에 기초하여 *CMM(Common Monitoring Model)*이라는 모니터링 객체의 계층을 정의합니다. 각 제품 구성 요소는 관찰 가능한 속성을 나타내는 객체를 표시하고 **노드 에이전트**는 하나의 호스트에서 여러 구성 요소의 보기를 집계합니다. 또한 Monitoring Framework는 작업 통계를 수집하고 사용자 정의 임계값에 기초하여 경보를 정의하는 메커니즘을 제공합니다.

Sun Java System Monitoring Console은 Java ES 구성 요소를 모니터링하기 위한 그래픽 인터페이스이며 Java ES 배포의 모든 노드 에이전트에 연결되는 **마스터 에이전트**를

포함합니다. Monitoring Console은 HTTP를 통해 모든 곳에서 액세스할 수 있도록 Sun Java System 웹 콘솔에 의존하는 웹 기반 응용 프로그램입니다. 기본 화면에서는 트리거된 경보를 비롯하여 활성화된 모든 구성 요소의 요약 상태가 제공됩니다. 그런 다음 각 구성 요소에 있는 모니터링된 객체의 계층에 액세스하고 모든 모니터링된 속성의 자세한 상태와 실시간 값을 볼 수 있습니다. Monitoring Console 인터페이스를 사용하면 모든 경보의 세부 정보를 표시하거나 경보를 확인하거나 모든 속성에 기초한 새 모니터링 규칙을 만들 수 있습니다.

Java ES 모니터링 작동 방법

모니터링은 시스템 관리자가 성능을 평가하고 경보에 대한 알림을 받을 수 있도록 런타임 데이터를 수집하고 표시하고 서비스 품질 기준을 계산하는 전체 프로세스입니다. 런타임 작업 도중 관리자는 Monitoring Console과 상호 작용하여 성능 통계를 표시하고 자동 모니터링을 위한 규칙을 만들고 경고를 확인하면 됩니다. 그러나 구성, 문제 해결 및 고급 모니터링의 경우 Monitoring Framework의 아키텍처와 Monitoring Console에 연결되는 방법을 이해하면 도움이 됩니다.

Java ES의 모니터링은 다음 개념에 기반을 둡니다.

- CMM(Common Monitoring Model)은 모든 Java ES 구성 요소가 비교할 수 있는 속성에 대한 일관된 객체와 값을 표시하도록 합니다.
- CMM 인터페이스에 의해 정의되는 Java 객체는 제품 구성 요소를 위한 표준화된 계측을 제공합니다.
- 노드 에이전트는 시스템에 설치된 모든 구성 요소에 대한 모든 모니터링된 객체를 표시하고 이러한 객체에 대한 통계, 규칙 및 경보를 관리합니다.
- 별개의 호스트에 있는 마스터 에이전트는 모든 노드 에이전트에서 모든 모니터링된 객체를 집계하고 Monitoring Console에서 데이터를 사용할 수 있게 합니다.

다음 절에서는 모니터링 아키텍처의 이러한 각 개념에 대해 자세히 설명합니다.

CMM(Common Monitoring Model)

표준화된 모니터링 메커니즘의 기초는 모니터링되는 객체를 정의하고 모든 모니터링된 구성 요소에서 이러한 객체를 채택하는 것입니다. 이를 위해 모니터링 아키텍처는 DMTF(Distributed Management Task Force)가 유지 관리하는 CIM(Common Information Model)의 확장으로 CMM(Common Monitoring Model)을 정의합니다. CMM은 컴퓨터, 응용 프로그램과 같은 모니터링된 객체를 지정하는 정보 모델이고 작업 상태 값과 같은 일관된 값을 지정하는 데이터 모델입니다. 또한 정보 모델의 일부로서 CMM은 객체의 속성(예: 서비스가 처리하는 요청 수)과 객체 사이의 관계(예: 서비스가 특정 컴퓨터에서 호스팅된다는 사실)를 정의합니다.

기본 구현이 다른 경우에도 CMM 덕분에 응용 프로그램, 서비스, 액세스 지점 등과 같은 개념은 모든 제품 구성 요소에서 동일합니다. 예를 들어, Web Server는 HTTP 요청을

처리하는 서비스를 표시할 수 있고 Directory Server는 LDAP 요청을 처리하는 서비스를 표시할 수 있습니다. 그러나 표준 객체는 처리된 요청 수를 측정하는 기능, 일정 기간 동안 요청에 응답하는 데 걸린 평균 시간 등과 같은 이러한 두 기능에 공통된 사항을 파악합니다.

게다가 특정 데이터 값이 표준화되므로 전체 시스템에서 해당 의미가 일관됩니다. 예를 들어, 작업 상태 `DEGRADED`은 모니터링하고 있는 제품 구성 요소에 상관 없이 서비스를 여전히 사용할 수 있지만 성능이 크게 저하되었다는 것을 항상 의미합니다.

부록 A에 설명된 계측에 사용되는 Java 인터페이스 및 클래스에서 CMM 사양이 구현됩니다.

CMM 계측

Monitoring Framework에서 계측은 CMM 정의를 구현하는 Java 인터페이스 및 클래스 집합입니다. Java ES의 새 모니터링 기능을 위해 제품 구성 요소는 모니터링된 객체의 속성을 통해 CMM 객체를 인스턴스화하고 런타임 값을 표시하도록 코드를 계측했습니다. 각 구성 요소에 의해 구현되는 CMM 객체는 모니터링할 수 있는 항목을 결정합니다. 이러한 이유 때문에 일부 구성 요소는 다른 구성 요소보다 적은 속성을 표시합니다. 모니터링을 위해 각 제품 구성 요소에 의해 표시되는 객체 및 속성 목록은 부록 B에서 제공됩니다.

노드 에이전트

모니터링 용어에서 노드는 고유한 정규화된 도메인 이름이나 IP 주소로 식별되는 단일 논리 호스트입니다. 노드는 가상 시스템으로 구성된 Solaris 영역이나 전체 시스템이 될 수 있습니다. 노드 에이전트는 해당 호스트의 모든 계측된 구성 요소와 통신하고 모든 모니터링된 객체를 표시합니다. 또한 노드 에이전트는 성능 통계를 수집하기 위한 모든 논리를 관리하고 규칙에 정의된 임계값을 모니터링하며 포함된 모니터링된 객체에 대한 경보를 생성합니다.

다음 다이어그램은 Java ES 제품 구성 요소 세 개의 인스턴스가 있는 단일 호스트의 노드 에이전트에 포함된 내용을 나타냅니다. 또한 제품 구성 요소에 의해 제공된 값을 표시하기 위해 계측이 노드 에이전트에서 인스턴스화되는 방법을 보여줍니다.

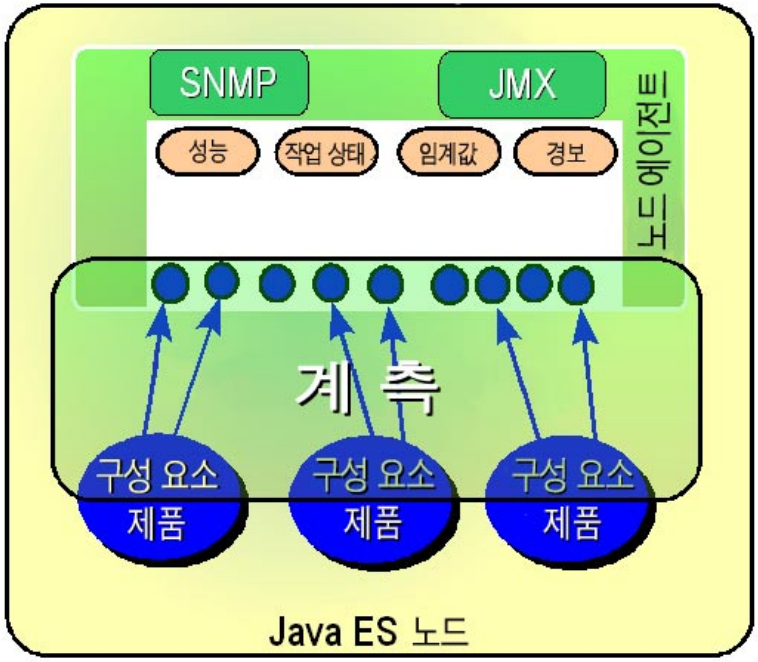


그림 1–1 노드 에이전트 다이어그램

노드 에이전트는 그 자체가 Java Virtual Machine인 Common Agent Container에 로드되는 모듈로 구현됩니다. 노드 에이전트 구현은 모니터링 및 원격 관리를 위한 표준 Java 확장인 JMX(Java Management Extensions)에 기반을 둡니다. CMM을 이해하는 모든 JMX 사용 모니터링 응용 프로그램은 노드 에이전트의 모니터링된 객체에 액세스할 수 있습니다. 또한 JMX 기능을 사용하여 노드 에이전트는 SNMP(Simple Network Monitoring Protocol)를 통해 모니터링된 특정 객체를 표시할 수 있습니다.

마스터 에이전트

마스터 에이전트는 Monitoring Console 설치의 일부로 별개의 시스템에 배포됩니다. 마스터 에이전트는 모든 노드의 이름이나 주소를 사용하여 구성되므로 모든 노드 에이전트에서 모니터링된 객체를 집계할 수 있습니다. 또한 마스터 에이전트는 노드 에이전트와 통신하기 위해 사용되는 JMX에 의존하며 해당 로컬 Common Agent Container에도 로드됩니다.

다음 다이어그램은 두 개의 노드에 연결된 마스터 에이전트를 나타냅니다. Monitoring Console은 마스터 에이전트에 연결되어 각 노드에서 세 개의 구성 요소를 모니터링합니다. 마스터 에이전트가 SNMP 속성을 집계하지 않으므로 SNMP를

모니터링에 사용하려면 각 노드에 별도로 연결해야 합니다. 마스터 에이전트는 단지 Monitoring Console과 함께 사용하도록 설계되었으며 다른 모니터링 응용 프로그램에서 액세스할 수 없습니다.

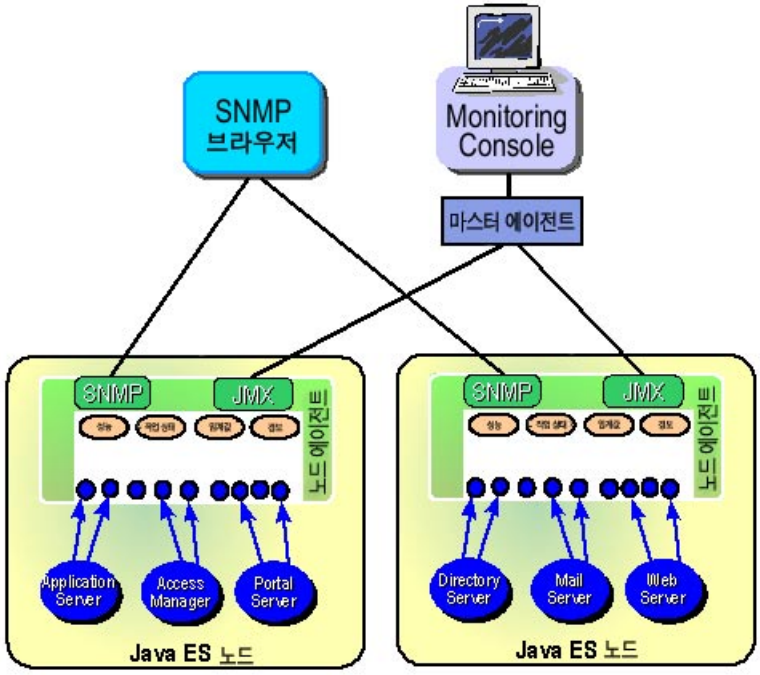


그림 1–2 전체 모니터링 아키텍처의 다이어그램

제안된 설치 순서

Java ES의 모니터링 기능을 평가하거나 배포하려는 경우 가장 쉬운 방법은 다음 순서로 설치를 수행하는 것입니다.

1. **Sun Java Enterprise System 5 UNIX용 설치 설명서**의 권장 사항 및 지침에 따라 배포의 모든 구성 요소를 설치 및 구성합니다.
2. 2 장에 설명된 대로 모든 모니터링된 구성 요소에 대해 Monitoring Framework를 활성화 및 구성합니다.
3. 3 장에 설명된 대로 Monitoring Console을 별개의 호스트에 설치하고 마스터 에이전트를 시작한 다음 Web Server를 시작합니다. 그런 다음 모든 모니터링된 구성 요소가 Monitoring Console에서 표시되고 활발하게 모니터링되어야 합니다.

주 – 이 릴리스에서 노드 에이전트와 마스터 에이전트의 비호환성으로 인해 Monitoring Console은 다른 Java ES 구성 요소를 포함하지 않는 호스트에 설치해야 합니다. 자세한 내용은 62 페이지 “Monitoring Console 문제 해결”을 참조하십시오.

모니터링을 활성화한 후에 배포된 구성 요소를 수정할 때마다 27 페이지 “Monitoring Framework 문제 해결”에 설명된 대로 마스터 에이전트에 대한 컨테이너와 Monitoring Console에 대한 Web Server를 다시 시작해야 합니다.

◆ ◆ ◆ 2 장

2 Monitoring Framework 활성화 및 구성

14 페이지 “Java ES 모니터링 작동 방법”에 설명된 대로 Monitoring Framework는 모니터링된 모든 구성 요소에 필요한 계측과 노드 에이전트를 제공합니다. 따라서 Monitoring Framework는 Java Enterprise System 설치 프로그램을 사용하여 모니터링된 구성 요소를 설치할 때마다 자동으로 설치되는 공유 구성 요소입니다.

그러나 대부분의 모니터링된 구성 요소는 기본적으로 모니터링이 활성화되어 있지 않으며 일부는 노드 에이전트에서 표시하기 위한 추가 구성이 필요합니다. 설치한 각각의 제품 구성 요소에 대해 이 장의 절차를 따릅니다.

주 – 이 장의 절차를 수행하기 전에 지정된 호스트에서 실행하려는 모든 제품 구성 요소를 설치 및 구성하는 것이 좋습니다. 설치나 구성을 수행하기 전에 **Sun Java Enterprise System 5 UNIX용 릴리스 노트**를 참조해야 합니다.

이러한 절차에서는 `mfwksetup` 명령이 사용되는데 이 명령은 일반적으로 필요하지 않으므로 설명되어 있지 않습니다.

이 장의 내용은 다음과 같습니다.

설치된 디렉토리 레이아웃

공유 구성 요소로서 Monitoring Framework는 필요할 때마다 자동으로 설치됩니다. 운영 체제에 설치되는 패키지의 이름은 **Sun Java Enterprise System 5 UNIX용 설치 참조 설명서**의 5 장, "설치 패키지 목록"을 참조하십시오. 다음 표에서는 Monitoring Framework 패키지의 디렉토리에 대해 설명합니다. 9 페이지 "기본 경로 및 파일 이름"에 설명된 대로 기본 설치 디렉토리 *mfwk-base*에는 다음 의미가 있습니다.

- Solaris 시스템: `/opt/SUNWmfwk`
- Linux 시스템: `/opt/sun/mfwk`

표 2–1 Monitoring Framework에 사용되는 디렉토리

경로	내용 설명
mfwk-base`/config`	구성 파일용 템플릿
Solaris 시스템: *mfwk-base*`/lib` Linux 시스템: *mfwk-base*`/share/lib`	Java 아카이브(`.jar`) 파일
Solaris 시스템: *mfwk-base*`/lib` Linux 시스템: *mfwk-base*`/share/lib`	32비트 런타임 라이브러리 파일(`.so`)
Solaris SPARC® 시스템: *mfwk-base*`/lib/sparcv9` Solaris x86 시스템: *mfwk-base*`/amd64` Linux 시스템: *mfwk-base*`/lib64`	64비트 런타임 라이브러리 파일(`.so`)
mfwk-base`/bin`	공용 스크립트 및 전용 바이너리
mfwk-base`/mib`	Monitoring Framework에서 지원하는 텍스트 버전의 SNMP MIB
mfwk-base`/xml`	에이전트 및 마스터 에이전트용 Common Agent Container 설명자 템플릿(`mfwksetup` 명령에 의해 배포됨)
mfwk-base`/dtd`	OSS/J 기능용 DTD 파일
`/etc/`*mfwk-base*`/config`	보안 관련 파일을 포함한 구성 파일
`/etc/`*mfwk-base*`/xml`	에이전트용 Common Agent Container 설명자 및 예제
`/var/`*mfwk-base*`/logs`	Monitoring Framework의 로그 파일
`/var/`*mfwk-base*`/reports`	규칙 보고서를 모니터링하기 위한 기본 디렉토리
`/var/`*mfwk-base*`/alarms`	경보 파일용 저장소

Access Manager에서 Monitoring Framework 사용

기본적으로 Access Manager에서 모니터링이 활성화되지만 모니터링된 객체가 Monitoring Console에 표시될 수 없는 제한 사항이 있습니다.

모니터링할 수 있는 객체 및 속성 목록은 67 페이지 "Access Manager의 계측"을 참조하십시오.

▼ Access Manager에서 모니터링을 활성화하는 방법

1 다음 명령을 사용하여 Access Manager에서 일시적으로 모니터링을 비활성화합니다.

```
cacaoadm unregister-module com.sun.cmm.am.xml
cacaoadm restart
```

2 편집하기 위해 Access Manager XML 설명자 파일을 엽니다.

```
vi /etc/AccessMgr-base/config/com.sun.cmm.am.xml
```

3 다음을 포함하는 줄을 찾습니다.

```
<param-name>Product Name</param-name>
<param-value>Access Manager</param-value>
```

두 번째 줄을 다음과 같이 수정합니다.

```
<param-value>Java ES Access Manager</param-value>
```

파일을 저장하고 편집기를 종료합니다.

4 수정된 XML 모듈을 등록합니다.

```
mfwk-base/bin/mfwksetup -u /etc/AccessMgr-base/config/com.sun.cmm.am.xml
mfwk-base/bin/mfwksetup -r /etc/AccessMgr-base/config/com.sun.cmm.am.xml
```

5 Common Agent Container를 다시 시작합니다.

```
cacaoadm restart
```

일반 오류 타사 웹 컨테이너에서의 테스트되지 않은 동작으로 인해 Access Manager가 Websphere 또는 Weblogic에 배포될 때 모니터링이 기본적으로 비활성화됩니다. 지원되지 않는 구성이기는 하지만 54 페이지 "모니터링을 선택적으로 비활성화하고 다시 활성화하는 방법"에 설명된 대로 모니터링을 활성화할 수 있습니다.

Application Server에서 Monitoring Framework 사용

모니터링할 수 있는 객체 및 속성 목록은 67 페이지 “Application Server의 계측”을 참조하십시오.

▼ Application Server에서 모니터링을 활성화하는 방법

1 **/var/*AppServer-base*/domains/domain1/config/domain.xml 파일을 편집하여 모든 module-monitoring-level 설정을 OFF에서 HIGH로 변경합니다. 또는 다음을 수행합니다.**

   a. **https:// *hostname*:4849의 Application Server 관리 콘솔에 로그온합니다.**

   b. **구성을 선택한 다음 server-config(관리 구성)를 선택합니다.**

   c. **모니터링 값을 HIGH로 설정합니다.**

   d. **다른 모든 값을 HIGH로 설정합니다.**

2 **다음 명령을 사용하여 Application Server를 다시 시작합니다.**

```
cd AppServer-base/appserv/bin
asadmin stop-domain domain1
asadmin start-domain user myUser domain1
```

   메시지가 표시되면 *myUser*에 대한 비밀번호를 입력합니다.

3 **Application Server와 함께 Portal Server 인스턴스를 배포 및 모니터링한 경우 Application Server를 다시 시작하는 프로세스가 Portal Server 모니터링과 충돌합니다. Portal Server 인스턴스가 Monitoring Console에 표시되게 하려면 브라우저에서 포털 페이지를 방문해야 합니다. 예를 들어, portalserv.example.com의 모니터링을 허용하도록 http://portalserv.example.com:8080/portal 페이지를 로드합니다.**

일반 오류 Application Server가 중단되거나 다운될 경우 Application Server에 대한 모니터링된 객체가 Monitoring Framework에서 제거되는 제한 사항이 있습니다. 이 경우 Application Server가 Monitoring Console에서 사라지며 더 이상 모니터링할 수 없습니다.

Calendar Server에 서 Monitoring Framework 사용

모니터링할 수 있는 객체 및 속성 목록은 67 페이지 “Calendar Server의 계측”을 참조하십시오.

▼ Calendar Server에 서 모니터링을 활성화하는 방법

1 **ics.conf 파일을 편집합니다.**

```
vi CalServ-base/cal/config/ics.conf
```

2 **다음 줄을 추가합니다.**

```
local.mfagent.enable="yes"
```

3 **Calendar Server XML 모듈을 등록합니다.**

```
mfwk-base/bin/mfwksetup -r /opt/SUNWics5/cal/lib/com.sun.cmm.cs.xml
```

4 **LD_LIBRARY_PATH 환경 변수를 다음과 같이 설정합니다.**

```
LD_LIBRARY_PATH=mfwk-base/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

5 **Calendar Server를 다시 시작합니다.**

```
cd CalServ-base/cal/sbin/
./stop-cal
./start-cal
```

6 **Common Agent Container를 다시 시작합니다.**

```
cacaoadm restart
```

Directory Server에 서 Monitoring Framework 사용

모니터링할 수 있는 객체 및 속성 목록은 68 페이지 “Directory Server의 계측”을 참조하십시오.

▼ Directory Server에 서 모니터링을 활성화하는 방법

1 **임시 비밀번호 파일을 만듭니다.**

```
echo –n password > /tmp/pwd
```

2 **다음 명령을 사용하여 모니터링 플러그인을 활성화합니다.**

```
DirServ-base/ds6/bin/dscfg enable-plugin -e -p 389 -w /tmp/pwd "Monitoring Plugin"
```

3 **Directory Server를 다시 시작합니다.**

```
cd DirServ-base/ds6/bin
./dsadm restart /var/DirServ-base/DSinstance/
```

Instant Messaging에서 Monitoring Framework 사용

모니터링할 수 있는 객체 및 속성 목록은 68 페이지 "Instant Messaging의 계측"을 참조하십시오.

▼ Instant Messaging에서 모니터링을 활성화하는 방법

1 **편집하기 위해 Instant Messaging XML 설명자를 엽니다.**

```
vi /etc/IM-base/default/com.sun.cmm.im.xml
```

2 **설치 위치를 *IM-base*에서 `/etc/`*IM-base*`/default`로 변경합니다.**

3 **수정된 Instant Messaging XML 설명자를 등록합니다.**

```
mfwk-base/bin/mfwksetup -r /etc/IM-base/default/com.sun.cmm.im.xml
```

4 **다음 줄을 *IM-base*`/config/iim.conf` 파일에 추가하여 계측을 활성화합니다.**

```
iim_server.monitor.enable = true
```

5 **다음 명령을 사용하여 Instant Messaging을 다시 시작합니다.**

```
cd IM-base/sbin
./imadmin stop
./imadmin start
```

6 **Common Agent Container를 다시 시작합니다.**

```
cacaoadm restart
```

Messaging Server에서 Monitoring Framework 사용

모니터링할 수 있는 객체 및 속성 목록은 68 페이지 "Messaging Server의 계측"을 참조하십시오.

▼ Messaging Server에서 모니터링을 활성화하는 방법

1 **다음 명령을 사용하여 계측을 활성화합니다.**

```
MsgServ-base/sbin/configutil -o local.mfagent.enable -v 1
```

2 **Messaging Server XML 모듈을 등록합니다.**

mfwk-base/bin/mfwksetup -r *MsgServ-base*/lib/com.sun.cmm.ms.xml

3 **Messaging Server를 다시 시작합니다.**

```
cd MsgServ-base/sbin
./stop-msg
./start-msg
```

4 **Common Agent Container를 다시 시작합니다.**

```
cacaoadm restart
```

Portal Server에서 Monitoring Framework 사용

모니터링할 수 있는 객체 및 속성 목록은 68 페이지 "Portal Server의 계측"을 참조하십시오.

▼ Portal Server에서 모니터링을 활성화하는 방법

- **Portal Server를 활성화하려면 로그온해야 합니다.**

  http://*FullHostname*:8080/portal/dt

  이렇게 하면 포털 JSP가 컴파일되어 모니터링 준비가 되어 있는 포털 인스턴스가 만들어집니다.

일반 오류 Portal Server를 호스팅하는 Application Server가 다시 시작될 때마다 이 절차에 따라 모니터링을 수동으로 다시 활성화해야 합니다.

Web Server에서 Monitoring Framework 사용

모니터링할 수 있는 객체 및 속성 목록은 68 페이지 "Web Server의 계측"을 참조하십시오.

▼ Web Server에서 모니터링을 활성화하는 방법

1 **다음 명령을 사용하여 Web Server를 시작합니다.**

```
cd /var/WebServer-base/https-FullHostname/bin
./startserv
```

2 **Administration Server를 시작합니다.**

```
cd /var/WebServer-base/admin-server/bin
./startserv
```

Common Agent Container 설정

Common Agent Container는 또 다른 공유 구성 요소이며 Monitoring Framework는 이 구성 요소에 의존하여 노드 에이전트를 실행합니다. 설치 순서에 따라 Common Agent Container를 중지했다가 다시 시작해야 할 수 있습니다. 또한 Common Agent Container는 계측되어 있으며 모니터링할 수도 있습니다. 모니터링된 객체에 대한 자세한 내용은 67 페이지 "Common Agent Container의 계측"을 참조하십시오.

Common Agent Container 및 노드 에이전트가 이미 시작되었는지 확인하려면 다음 명령을 실행합니다.

```
cacaoadm status
```

다음과 비슷한 메시지가 표시되면 노드 에이전트가 실행 중입니다.

```
default instance is DISABLED at system startup.
Smf monitoring process:
26996
Uptime: 0 day(s), 0:57
```

다음과 비슷한 메시지가 표시되면 노드 에이전트가 실행 중이 아닙니다.

```
default instance is DISABLED at system startup.
default instance is not running.
```

▼ Common Agent Container의 모니터링을 활성화하는 방법

Common Agent Container는 모니터링을 허용하기 위한 계측이 있는 공유 구성 요소입니다. 15 페이지 "노드 에이전트"에 설명된 대로 호스트 또는 영역의 모든 Java ES 구성 요소는 Common Agent Container 및 노드 에이전트를 공유합니다. Common Agent Container를 모니터링하려는 배포의 모든 논리 호스트에서 이 작업을 루트로 수행합니다.

1 **Common Agent Container가 실행 중인 경우 다음 명령을 사용하여 중지합니다.**

```
cacaoadm stop
```

2 **컨테이너 자체의 계측을 활성화합니다.**

```
cacaoadm set-param enable-instrumentation=true
```

3 **방금 설정한 매개 변수 값을 확인하고 Common Agent Container를 다시 시작합니다.**

```
cacaoadm get-param enable-instrumentation
cacaoadm start
```

4 **키 비밀번호를 만듭니다.**

```
echo –n password > /etc/mfwk-base/config/security/password.cacao
```

5 **키를 생성합니다.**

```
mfwk-base/bin/cpgenkey -n cacao -p /etc/mfwk-base/config/security/password.cacao
```

6 **Common Agent Container의 고유한 모니터링 모듈을 등록합니다.**

```
cacaoadm register-module /usr/lib/cacao/ext/instrum/config/com.sun.cacao.instrum.xml
cacaoadm register-module /usr/lib/cacao/ext/instrum_jesmf/config/com.sun.cacao.instrum.jesmf.xml
cacaoadm register-module /usr/lib/cacao/ext/instrum_jesmf/config/com.sun.cacao.cmm.xml
```

Monitoring Framework 문제 해결

Sun Java Enterprise System 5 UNIX용 릴리스 노트에 나열된 알려진 문제도 참조하십시오.

HP-UX 플랫폼에서 Monitoring Framework 사용

HP-UX의 JVM(Java Virtual Machine)은 기본적으로 Monitoring Framework에 필요한 작업 집중적 처리에 맞게 조정되지 않으며 이로 인해 `OutOfMemory` 예외가 발생할 수 있습니다. JVM을 구성하려면 다음 위치에서 `HPjconfig` 도구를 다운로드하여 실행합니다.`http://h21007.ww2.hp.com/dspp/tech/tech_TechDocumentDetailPage_IDX/1,1701,1620,00.html`.

Microsoft Windows에서 Monitoring Framework 사용

몇 가지 차이점이 있기는 하지만 Windows 플랫폼에서 Monitoring Framework를 통해 Java ES 구성 요소를 모니터링하는 것이 완전히 지원됩니다. 예를 들어, 알려진 특정 문제를 방지하기 위해 Java 1.5 이상으로 업그레이드해야 합니다. 다른 알려진 문제는 **Sun Java Enterprise System 5 Release Notes for Microsoft Windows**를 참조하십시오.

▼ 노드 에이전트를 다시 시작하는 방법

노드 에이전트를 호스팅하는 Common Agent Container를 다시 시작해야 할 경우 다음 절차를 수행할 때까지 해당 노드 에이전트를 통해 모니터링되는 구성 요소를 Monitoring Console에서 볼 수 없습니다.

1. **노드 에이전트를 호스팅하는 Common Agent Container를 다시 시작합니다.**

   ```
   cacaoadm restart
   ```

2. **마스터 에이전트를 호스팅하는 Common Agent Container를 다시 시작합니다. 마스터 에이전트는 Monitoring Console이 설치된 영역이나 호스트의 Monitoring Framework에서 실행됩니다.**

   ```
   cacaoadm restart
   ```

   마스터 에이전트는 이전에 모니터링하던 모든 노드 에이전트와 자동으로 다시 연결됩니다.

3. **Monitoring Console을 호스팅하는 웹 서버를 다시 시작합니다.**

   ```
   /usr/sbin/smcwebserver restart
   ```

mfwkadm 명령

이 절에서는 설명서 페이지 섹션 1M의 유지 관리 명령인 mfwkadm 명령의 설명서 페이지를 재현합니다. 이 명령을 사용하여 모니터링하는 구성 요소에 대한 모든 모듈 및 노드에 정의한 작업이라고도 알려진 모든 모니터링 규칙을 비롯한 노드 에이전트의 내용을 관리합니다. 설명서 페이지의 일부 용어와 설명은 이 문서에 사용된 내용과 일치하도록 여기에서 수정되었습니다.

개요

mfwkadm --help

mfwkadm start

mfwkadm stop

mfwkadm restart

mfwkadm list-params

mfwkadm list-modules

mfwkadm info *runningInstance*

성능 모니터링

mfwkadm pm-job observable-classes

mfwkadm pm-job observable-objects [class=*objectClass*] [domain= *objectDomain*]

mfwkadm pm-job observable-attributes class=*objectClass*

mfwkadm pm-job list

mfwkadm pm-job info *jobName*

mfwkadm pm-job create *jobName* granularity=*integerValue* object=*objectName* [object=*objectName* ...]

mfwkadm pm-job delete *jobName*

mfwkadm pm-job suspend *jobName*

mfwkadm pm-job resume *jobName*

작업 상태 모니터링

mfwkadm opstat-job observable-classes

mfwkadm opstat-job observable-objects [class=*objectClass*] [domain= *objectDomain*]

mfwkadm opstat-job observable-attributes class= *objectClass*

mfwkadm opstat-job list

mfwkadm opstat-job info *jobName*

mfwkadm opstat-job create *jobName* granularity= *integerValue* object=*objectName* [object=*objectName* ...]

mfwkadm opstat-job delete *jobName*

mfwkadm opstat-job suspend *jobName*

mfwkadm opstat-job resume *jobName*

임계값 모니터링

mfwkadm thrsh-job observable-classes

mfwkadm thrsh-job observable-objects [class=*objectClass*] [domain= *objectDomain*]

mfwkadm thrsh-job observable-attributes class=*objectClass*

mfwkadm thrsh-job list

mfwkadm thrsh-job info *jobName*

mfwkadm thrsh-job create *jobName* granularity=*integerValue* attributeName=*attributeName* attributeType=*attributeType* thresholdValue=*thresholdValue* thresholdOffset=*offsetValue* thresholdDirection=[RISING | FALLING] object=*objectName*

mfwkadm thrsh-job delete *jobName*

mfwkadm thrsh-job suspend *jobName*

mfwkadm thrsh-job resume *jobName*

설명

mfwkadm 유틸리티는 노드 에이전트라고도 하는 Monitoring Framework 에이전트를 관리하기 위한 명령줄 인터페이스입니다. 노드 에이전트는 Common Agent Container 내에서 실행됩니다. mfwkadm 유틸리티를 사용하여 노드 에이전트를 중지 및 재시작하고 수행하는 모니터링 작업을 관리할 수 있습니다. 이 명령은 노드 에이전트가 실행 중인 동일한 호스트에서 실행해야 합니다. 여기에 제공된 이 명령의 인수 순서를 유지해야 합니다.

출력 메시지의 언어를 변경하려면 LC_MESSAGE 환경 변수를 해당 **로켈**로 설정합니다. mfwkadm 명령은 lib/resources 디렉토리에 있는 JesmfMessages_ *locale*.pm이라는 파일에 포함된 메시지를 사용합니다. 해당 메시지 파일이 로켈에 없거나 로켈이 지정되지 않은 경우 mfwkadm 명령은 JesmfMessages.pm 파일에 있는 기본 메시지 집합을 사용합니다.

mfwkadm 유틸리티에는 다음 하위 명령이 있습니다. 별표(*)가 표시된 경우에는 Common Agent Container가 실행 중이어야 하고 노드 에이전트가 로드되어야 합니다.

- start
- stop
- restart
- list-params(*)
- list-modules(*)
- info(*)
- pm-job(*)
- opstat-job(*)
- thrsh-job(*)

로드할 Common Agent Container 모듈 수에 따라서 노드 에이전트를 시작하는 시간과 mfwkadm 유틸리티를 사용할 수 있을 때까지의 시간 사이에 몇 초 또는 몇 분의 시간 지연이 발생할 수 있습니다. 이 기간 동안 명령은 명시적 메시지와 함께 실패합니다.

옵션

다음 옵션이 지원됩니다.

--help　　사용법 요약을 표시합니다.

하위 명령

start

Common Agent Container를 중지하지 않고 Monitoring Framework 노드 에이전트 및 연관된 구성 요소 제품 모듈을 시작합니다.

이 작업은 먼저 노드 에이전트를 배포한 다음 Common Agent Container의 연관된 구성 요소 제품 모듈을 배포합니다. 이 기능은 cacaoadm 유틸리티의 lock 및 undeploy 하위 명령 위에 있는 래퍼입니다.

start 하위 명령은 Monitoring Framework와 연관된 노드 에이전트 및 Java ES 구성 요소 모듈만 시작합니다. 구성 요소 모듈에는 접두어 com.sun.cmm이 있습니다.

보안: start 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다. 다른 사용자가 실행할 경우 다음과 같은 오류 메시지가 표시됩니다.

```
Error occured in mfwkadm
Problem running /usr/sbin/cacaoadm unlock com.sun.mfwk 2>&1.
Stdout/Stderr: This command must be run by user: [root].
```

stop

Common Agent Container에 있는 Monitoring Framework 노드 에이전트 및 연관된 Java ES 구성 요소 모듈을 중지합니다.

이 작업은 먼저 Common Agent Container에 배포된 모든 Java ES 구성 요소의 모듈을 중지한 다음 노드 에이전트를 중지합니다. 이 기능은 cacaoadm lock 및 unlock 하위 명령 위에 있는 래퍼입니다.

stop 하위 명령은 Monitoring Framework와 연관된 Java ES 구성 요소 모듈만 중지한 다음 노드 에이전트 자체를 중지합니다. 구성 요소 모듈에는 접두어 com.sun.cmm이 있습니다.

보안: stop 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다. 다른 사용자가 실행할 경우 다음과 같은 오류 메시지가 표시됩니다.

```
Error occured in mfwkadm
Problem running /usr/sbin/cacaoadm unlock com.sun.mfwk 2>&1.
Stdout/Stderr: This command must be run by user: [root].
```

restart

Common Agent Container에 있는 Monitoring Framework 노드 에이전트 및 연관된 Java ES 구성 요소 모듈을 다시 시작합니다.

이 작업은 stop 및 start 하위 명령과 같은 방법으로 Common Agent Container에 있는 노드 에이전트 및 연관된 모듈을 중지한 다음 시작합니다.

보안: restart 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다. 다른 사용자가 실행할 경우 다음과 같은 오류 메시지가 표시됩니다.

```
Error occured in mfwkadm
Problem running //usr/sbin/cacaoadm unlock com.sun.mfwk 2>&1.
Stdout/Stderr: This command must be run by user: [root].
```

`list-params`
Monitoring Framework 노드 에이전트와 관련된 모든 구성 매개 변수를 나열합니다.

보안: 이 명령에 대한 사용자 제한은 없습니다.

`list-modules`
CMM(Common Monitoring Model)을 구현하고 Common Agent Container에 로드되는 구성 요소 제품 모듈의 목록을 표시합니다. 또한 이 하위 명령은 설치된 각 Java ES 구성 요소의 모든 실행 중인 인스턴스를 나열합니다. 각 구성 요소는 0개, 1개 또는 그 이상의 실행 중인 인스턴스를 가질 수 있습니다.

보안: Common Agent Container를 시작한 사용자가 아닌 경우 설치된 Java ES 구성 요소의 목록에 구성 요소 인스턴스가 포함되지 않습니다.

`info` *runningInstance*
명명된 *runningInstance*에 대한 정보를 표시합니다. *runningInstance*는 `list-modules` 하위 명령의 출력에 나열된 실행 중인 인스턴스와 일치해야 합니다. 다음과 같은 정보가 표시됩니다.

- 각 유형의 모니터링 작업에 대한 클래스 이름으로 정렬된 실행 중인 인스턴스와 연관된 관찰 가능한 모든 객체. 관찰 가능한 객체는 각각 `pm-job`, `opstat-job` 또는 `thrsh-job` 하위 명령을 사용하여 성능 모니터링 작업, 작업 상태 작업 또는 임계값 모니터링 작업을 만들 수 있는 객체입니다.
- 관찰 가능한 객체의 각 클래스에 대한 각각의 이름 및 유형을 포함하는 관찰 가능한 모든 속성

보안: Common Agent Container를 시작한 사용자가 아닌 경우 어떠한 정보도 표시되지 않습니다.

성능 모니터링

`pm-job observable-classes`
성능 모니터링 작업을 만들 수 있는 객체의 현재 관찰 가능한 모든 클래스의 목록을 표시합니다.

`pm-job observable-objects` [class= *objectC lass*] [domain=*objectDomain*]
성능 모니터링 작업을 만들 수 있는 현재 관찰 가능한 모든 객체의 목록을 표시합니다. 기본적으로 관찰 가능한 모든 클래스 및 모든 도메인에 있는 모든 객체가 나열됩니다. 객체 목록은 클래스 이름으로 정렬됩니다.

class=*objectClass*
선택적 *objectClass*를 지정하면 해당 특정 클래스의 관찰 가능한 객체로 출력이 제한됩니다. *objectClass*는 `pm-job observable-classes` 하위 명령이 나열하는 클래스 중 하나여야 합니다.

domain=*objectDomain*
선택적 *objectDomain*을 지정하면 해당 도메인의 관찰 가능한 객체로 출력이 제한됩니다. 객체의 도메인은 객체 이름에서 콜론(":") 문자 앞에 있는 문자열입니다.

`pm-job observable-attributes class=`*objectClass*
지정된 *objectClass*에 있는 모든 관찰 가능한 속성의 목록을 표시합니다. 속성은 이름 및 유형과 함께 표시됩니다. *objectClass*는 `pm-job observable-classes` 하위 명령이 나열하는 성능 모니터링 작업을 지원하는 클래스 중 하나여야 합니다.

`pm-job list`
현재 정의된 모든 성능 모니터링 작업의 목록을 표시합니다. 정의된 성능 작업이 있는 각 객체에 대한 작업이 나열되며 객체는 해당 클래스 이름으로 정렬됩니다. 각 작업에 대해 표시되는 정보는 `pm-job info` 하위 명령이 표시하는 정보와 동일합니다.

보안: Common Agent Container를 시작한 사용자가 아닌 경우 어떠한 작업도 표시되지 않습니다.

`pm-job info` *jobName*
*jobName*이라는 성능 모니터링 작업에 대한 세부 정보를 표시합니다. *jobName*은 `pm-job list` 하위 명령이 표시하는 이름이어야 합니다. 이 하위 명령은 다음 정보를 표시합니다.

- 성능 모니터링 작업의 이름
- 성능 모니터링 작업의 유형인 "객체별" 또는 "클래스별".객체별 작업은 하나 이상의 명명된 객체 인스턴스를 모니터링하고 클래스별 작업은 객체 클래스의 모든 인스턴스를 모니터링합니다. `mfwkadm` 유틸리티를 사용하여 클래스별 작업을 만들 수 없습니다.
- 성능 모니터링 작업의 상태: 활성 작동, 활성 비작동 또는 일시 중지됨. 활성 작동 작업은 실행하도록 예약되어 있고 데이터를 수집하는 중입니다. 활성 비작동 작업은 현재 시간이 작업 일정을 벗어났기 때문에 실행 중이지만 데이터를 수집하지는 않습니다. 일시 중지된 작업은 실행 중이거나 데이터를 수집하는 중이 아닙니다. `pm-job suspend` 및 `pm-job resume` 하위 명령을 사용하여 성능 모니터링 작업의 실행 상태를 변경합니다.
- 성능 모니터링 작업의 세부 기간(초). 이 작업의 데이터 수집 간격입니다.
- 모니터링 작업의 보고 기간. 보고 기간에 세부 기간을 곱하면 알림 빈도가 됩니다. 예를 들어, 세부 기간이 10초이고 보고 기간이 6인 경우 이벤트에 의한 작업 보고는 10초마다 데이터를 수집하고 60(10*6)초마다 6개의 보고서를 포함한 알림을 보냅니다. 작업이 또한 파일로 보고되는 경우 생성된 6개 파일의 위치를 포함하는 이벤트가 60초마다 보내집니다.
- 성능 모니터링 작업이 이벤트에 의해 보고되고 있는지 여부. 이는 성능 모니터링 작업의 결과가 등록된 클라이언트에게 알림으로 보내진다는 것을 의미합니다.
- 성능 모니터링 작업이 파일로 보고되고 있는지 여부. 이는 성능 모니터링 작업의 보고서가 로컬 파일에 기록되고 파일 이름을 포함하는 알림이 등록된 클라이언트에게 보내진다는 것을 의미합니다.
- 항상 XML인 성능 모니터링 작업의 보고서 형식

- 성능 모니터링 작업의 일정. 일정은 작업이 활성 작동 또는 활성 비작동(각각 데이터를 수집 중이거나 수집하지 않는 중)인 날짜와 시간을 지정합니다. 그런 다음 객체별 작업에 대한 다음 정보가 표시됩니다.
- 이름별로 순서가 지정된 관찰된 객체 목록
- 관찰 가능한 속성의 하위 집합만 지정된 경우 관찰된 객체의 관찰된 속성이 이름 및 유형별로 나열됩니다. 또한 클래스별 작업에 대한 다음 정보가 표시됩니다.
- 이름별로 순서가 지정된 관찰된 클래스 목록
- 관찰 가능한 속성의 하위 집합만 지정된 경우 관찰된 클래스의 관찰된 속성이 이름 및 유형별로 나열됩니다. 이러한 속성은 모든 클래스에 공통됩니다.

보안: Common Agent Container를 시작한 사용자가 아닌 경우 어떠한 정보도 표시되지 않습니다.

`pm-job create` *jobName* granularity= *integerValue* object=*objectName* [object= *objectName* ...]
하나 이상의 객체에서 새 성능 모니터링 작업을 만듭니다. `mfwkadm` 명령은 클래스별 작업을 만들 수 없습니다. 성능 모니터링 작업을 만들 경우 다음 매개 변수를 설정할 수 있습니다.

jobName
성능 모니터링 작업을 고유하게 식별하는 문자열. *jobName*은 다른 성능 모니터링 작업에서 이미 사용 중일 수 없습니다.

granularity=*integerValue*
작업이 활성 작동인 도중에 두 개의 연속된 측정 데이터 수집이 시작되는 시간 간격(초). 예를 들어, 세부 기간은 300초(5분), 900초(15분), 1800초(30분), 3600초(1시간)일 수 있습니다. 대부분의 경우 세부 기간이 300초면 충분합니다. 일부 측정의 경우 더 큰 세부 기간으로 데이터를 수집하는 것이 더 의미 있을 수 있습니다.

object=*objectName* [object= *objectName* ...]
성능 모니터링 작업이 데이터를 수집하고 보고하는 하나 이상의 관찰 가능한 객체. *objectName*은 `pm-job list` 또는 `pm-job observable-objects` 하위 명령이 표시하는 이름이어야 합니다. 여러 object= *objectName* 매개 변수를 지정할 경우 여러 객체를 모니터링하는 단일 성능 모니터링 작업이 만들어집니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

`pm-job delete` *jobName*
*jobName*이라는 성능 모니터링 작업을 삭제합니다. *jobName*은 `pm-job list` 하위 명령이 표시하는 이름이어야 합니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

`pm-job suspend` *jobName*
*jobName*이라는 성능 모니터링 작업을 일시 중지합니다. 일시 중지된 작업은 활성 상태가 아니며 일정에 상관 없이 더 이상 데이터를 수집하지 않습니다. 그러나 작업은 정의된 상태로 유지되며 `pm-job resume` 하위 명령을 사용하여 다시 활성화할 수 있습니다. *jobName*은 `pm-job list` 하위 명령이 표시하는 이름이어야 합니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

`pm-job resume` *jobName*
*jobName*이라는 성능 모니터링 작업을 다시 시작합니다. 다시 시작된 작업은 데이터 수집을 시작하고 일정에 따라 보고서를 보냅니다. *jobName*은 `pm-job list` 하위 명령이 표시하는 이름이어야 합니다. 이 하위 명령은 `pm-job suspend` 하위 명령에 대응합니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

작업 상태 모니터링

`opstat-job observable-classes`
작업 상태 모니터링 작업을 만들 수 있는 객체의 현재 관찰 가능한 모든 클래스의 목록을 표시합니다.

`opstat-job observable-objects` [class= *objectClass*] [domain=*objectDomain*]
작업 상태 모니터링 작업을 만들 수 있는 현재 관찰 가능한 모든 객체의 목록을 표시합니다. 기본적으로 관찰 가능한 모든 클래스 및 모든 도메인에 있는 모든 객체가 나열됩니다. 객체 목록은 클래스 이름으로 정렬됩니다.

class=*objectClass*
선택적 *objectClass*를 지정하면 해당 특정 클래스의 관찰 가능한 객체로 출력이 제한됩니다. *objectClass*는 `opstat-job observable-classes` 하위 명령이 나열하는 클래스 중 하나여야 합니다.

domain=*objectDomain*
선택적 *objectDomain*을 지정하면 해당 도메인의 관찰 가능한 객체로 출력이 제한됩니다. 객체의 도메인은 객체 이름에서 콜론(“:”) 문자 앞에 있는 문자열입니다.

`opstat-job observable-attributes class=` *objectClass*
지정된 *objectClass*에 있는 모든 관찰 가능한 속성의 목록을 표시합니다. 속성은 이름 및 유형과 함께 표시됩니다. *objectClass*는 `opstat-job observable-classes` 하위 명령이 나열하는 클래스 중 하나여야 합니다.

`opstat-job list`
현재 정의된 모든 작업 상태 모니터링 작업의 목록을 표시합니다. 정의된 작업 상태 작업이 있는 각 객체에 대한 작업이 나열되며 객체는 해당 클래스 이름으로 정렬됩니다. 각 작업에 대해 표시되는 정보는 `opstat-job info` 하위 명령이 표시하는 정보와 동일합니다.

보안: Common Agent Container를 시작한 사용자가 아닌 경우 어떠한 작업도 표시되지 않습니다.

`opstat-job info` *jobName*
*jobName*이라는 작업 상태 모니터링 작업에 대한 세부 정보를 표시합니다. *jobName*은 `opstat-job list` 하위 명령이 표시하는 이름이어야 합니다. 이 하위 명령은 다음 정보를 표시합니다.

- 작업 상태 모니터링 작업의 이름
- 작업 상태 모니터링 작업의 유형인 "객체별" 또는 "클래스별".객체별 작업은 명명된 객체 인스턴스를 모니터링하고 클래스별 작업은 객체 클래스의 모든 인스턴스를 모니터링합니다. `mfwkadm` 유틸리티를 사용하여 클래스별 작업을 만들 수 없습니다.
- 작업 상태 모니터링 작업의 상태: 활성 작동, 활성 비작동 또는 일시 중지됨. 활성 작동 작업은 실행하도록 예약되어 있고 데이터를 수집하는 중입니다. 활성 비작동 작업은 현재 시간이 작업 일정을 벗어났기 때문에 실행 중이지만 데이터를 수집하지는 않습니다. 일시 중지된 작업은 실행 중이거나 데이터를 수집하는 중이 아닙니다. `opstat-job suspend` 및 `opstat-job resume` 하위 명령을 사용하여 작업 상태 모니터링 작업의 실행 상태를 변경합니다.
- 작업 상태 모니터링 작업의 세부 기간(초). 이 작업의 데이터 수집 간격입니다.
- 작업 상태 모니터링 작업이 이벤트에 의해 보고되고 있는지 여부. 이는 작업 상태 모니터링 작업의 결과가 등록된 클라이언트에게 알림으로 보내진다는 것을 의미합니다.
- 작업 상태 모니터링 작업이 파일로 보고되고 있는지 여부. 이는 작업 상태 모니터링 작업의 보고서가 로컬 파일에 기록되고 파일 이름을 포함하는 알림이 등록된 클라이언트에게 보내진다는 것을 의미합니다.
- 항상 `XML`인 작업 상태 모니터링 작업의 보고서 형식
- 작업 상태 모니터링 작업의 일정. 일정은 작업이 활성 작동 또는 활성 비작동(각각 데이터를 수집 중이거나 수집하지 않는 중)인 날짜와 시간을 지정합니다.
- 객체별 작업의 경우 이름별로 순서가 지정된 관찰된 객체 목록
- 클래스별 작업의 경우 이름별로 순서가 지정된 관찰된 클래스 목록

보안: Common Agent Container를 시작한 사용자가 아닌 경우 어떠한 정보도 표시되지 않습니다.

`opstat-job create` *jobName* granularity= *integerValue* object=*objectName* [object= *objectName* ...]
하나 이상의 객체에서 새 작업 상태 모니터링 작업을 만듭니다. `mfwkadm` 명령은 클래스별 작업을 만들 수 없습니다. 성능 모니터링 작업을 만들 경우 다음 매개 변수를 설정할 수 있습니다.

jobName
작업 상태 모니터링 작업을 고유하게 식별하는 문자열. *jobName*은 다른 작업 상태 모니터링 작업에서 이미 사용 중일 수 없습니다.

granularity=*integerValue*
작업이 활성 작동인 도중에 두 개의 연속된 측정 데이터 수집이 시작되는 시간 간격(초).

object=*objectName* [object= *objectName* ...]
작업 상태 모니터링 작업이 데이터를 수집하고 보고하는 하나 이상의 관찰 가능한 객체. *objectName*은 `opstat-job list` 또는 `opstat-job observable-objects` 하위 명령이 표시하는 이름이어야 합니다. 여러 object= *objectName* 매개 변수를 지정할 경우 여러 객체를 모니터링하는 단일 작업 상태 작업이 만들어집니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

`opstat-job delete` *jobName*
*jobName*이라는 작업 상태 모니터링 작업을 삭제합니다. *jobName*은 `opstat-job list` 하위 명령이 표시하는 이름이어야 합니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

`opstat-job suspend` *jobName*
*jobName*이라는 작업 상태 모니터링 작업을 일시 중지합니다. 일시 중지된 작업은 활성 상태가 아니며 일정에 상관 없이 더 이상 데이터를 수집하지 않습니다. 그러나 작업은 정의된 상태로 유지되며 `opstat-job resume` 하위 명령을 사용하여 다시 활성화할 수 있습니다. *jobName*은 `opstat-job list` 하위 명령이 표시하는 이름이어야 합니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

`opstat-job resume` *jobName*
*jobName*이라는 작업 상태 모니터링 작업을 다시 시작합니다. 다시 시작된 작업은 데이터 수집을 시작하고 일정에 따라 보고서를 보냅니다. *jobName*은 `opstat-job list` 하위 명령이 표시하는 이름이어야 합니다. 이 하위 명령은 `opstat-job suspend` 하위 명령에 대응합니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

임계값 모니터링

`thrsh-job observable-classes`
임계값 모니터링 작업을 만들 수 있는 객체의 현재 관찰 가능한 모든 클래스의 목록을 표시합니다.

`thrsh-job observable-objects` [class= *objectClass*] [domain=*objectDomain*]
임계값 모니터링 작업을 만들 수 있는 현재 관찰 가능한 모든 객체의 목록을 표시합니다. 기본적으로 관찰 가능한 모든 클래스 및 모든 도메인에 있는 모든 객체가 나열됩니다. 객체 목록은 클래스 이름으로 정렬됩니다.

class=*objectClass*
선택적 *objectClass*를 지정하면 해당 특정 클래스의 관찰 가능한 객체로 출력이 제한됩니다. *objectClass*는 `thrsh-job observable-classes` 하위 명령이 나열하는 클래스 중 하나여야 합니다.

domain=*objectDomain*
선택적 *objectDomain*을 지정하면 해당 도메인의 관찰 가능한 객체로 출력이 제한됩니다. 객체의 도메인은 객체 이름에서 콜론(“:”) 문자 앞에 있는 문자열입니다.

`thrsh-job observable-attributes class=`*objectClass*
지정된 *objectClass*에 있는 모든 관찰 가능한 속성의 목록을 표시합니다. 속성은 이름 및 유형과 함께 표시됩니다. *objectClass*는 `thrsh-job observable-classes` 하위 명령이 나열하는 클래스 중 하나여야 합니다.

`thrsh-job list`
현재 정의된 모든 임계값 모니터링 작업의 목록을 표시합니다. 정의된 임계값 작업이 있는 각 객체에 대한 작업이 나열되며 객체는 해당 클래스 이름으로 정렬됩니다. 각 작업에 대해 표시되는 정보는 `thrsh-job info` 하위 명령이 표시하는 정보와 동일합니다.

보안: Common Agent Container를 시작한 사용자가 아닌 경우 어떠한 작업도 표시되지 않습니다.

`thrsh-job info` *jobName*
*jobName*이라는 임계값 모니터링 작업에 대한 세부 정보를 표시합니다. *jobName*은 `thrsh-job list` 하위 명령이 표시하는 이름이어야 합니다. 이 하위 명령은 다음 정보를 표시합니다.

- 임계값 모니터링 작업의 이름
- 임계값 모니터링 작업의 다중성. 이 릴리스에서는 하나의 객체에서 하나의 속성을 모니터링하는 간단한 임계값 작업만 가능합니다.
- 임계값 모니터링 작업의 상태: 활성 작동, 활성 비작동 또는 일시 중지됨. 활성 작동 작업은 실행하도록 예약되어 있고 데이터를 수집하는 중입니다. 활성 비작동 작업은 현재 시간이 작업 일정을 벗어났기 때문에 실행 중이지만 데이터를 수집하지는 않습니다. 일시 중지된 작업은 실행 중이거나 데이터를 수집하는 중이

아닙니다. thrsh-job suspend 및 thrsh-job resume 하위 명령을 사용하여 임계값 모니터링 작업의 실행 상태를 변경합니다.

- 임계값 모니터링 작업의 세부 기간(초). 이 작업의 데이터 수집 간격입니다.
- 임계값 모니터링 작업의 일정. 일정은 작업이 활성 작동 또는 활성 비작동(각각 데이터를 수집 중이거나 수집하지 않는 중)인 날짜와 시간을 지정합니다.
- 임계값 모니터링 작업의 경보 구성. 모니터링된 속성의 관찰된 값이 정의된 임계값을 초과할 경우 트리거되는 경보입니다. 경보의 유형과 심각도가 표시됩니다.
- 임계값 모니터링 작업의 관찰된 객체
- 임계값이 적용되는 속성 이름
- 경보를 트리거하는 임계값
- 임계값에서 경보를 트리거하는 값의 진행 방향인 RISING 또는 FALLING
- 임계값의 허용치 오프셋. 방향이 RISING인 경우 관찰된 속성이 *thresholdValue-offsetValue*보다 작을 때까지 경보가 다시 트리거되지 않습니다. 방향이 FALLING인 경우 관찰된 속성이 *thresholdValue+offsetValue*보다 클 때까지 경보가 다시 트리거되지 않습니다. 오프셋이 0인 경우에도 이 동작이 적용됩니다.

보안: Common Agent Container를 시작한 사용자가 아닌 경우 어떠한 정보도 표시되지 않습니다.

thrsh-job create *jobName* object= *objectName* granularity=*integerValue* attributeName= *attributeName* attributeType=*attributeType* thresholdValue= *thresholdValue* thresholdOffset=*offsetValue* thresholdDirection= [RISING|FALLING]

단일 객체에서 하나의 속성을 모니터링하는 새 임계값 모니터링 작업을 만듭니다. 임계값 작업을 만들 경우 다음 매개 변수를 설정할 수 있습니다.

jobName
: 임계값 모니터링 작업을 고유하게 식별하는 문자열. *jobName*은 다른 임계값 모니터링 작업에서 이미 사용 중일 수 없습니다.

object=*objectName*
: 임계값 모니터링 작업이 임계값에 대해 비교하기 위해 속성 값을 수집하는 관찰 가능한 객체 *objectName*은 thrsh-job list 또는 thrsh-job observable-objects 하위 명령이 표시하는 이름이어야 합니다.

granularity=*integerValue*
: 작업이 활성 작동인 도중에 두 개의 연속된 속성 값 관찰이 시작되는 시간 간격(초).

attributeName=*attributeName*
: 임계값 모니터링 작업이 임계값에 비교하기 위해 값을 수집하는 속성의 이름 *attributeName*은 thrsh-job info 또는 thrsh-job observable-attributes 하위 명령에 의해 나열되어야 합니다.

attributeType=*attributeType*
모니터링할 관찰 가능한 속성의 유형. *attributeType*은 `thrsh-job info` 또는 `thrsh-job observable-attributes` 하위 명령에 의해 나열되어야 합니다.

thresholdValue=*thresholdValue*
`thresholdDirection`에 지정된 방향으로 초과할 경우 이 임계값 작업이 경보를 트리거하게 하는 모니터링된 속성의 값

thresholdOffset=*offsetValue*
*offsetValue*는 연속된 경보를 트리거하는 임계값 작업의 허용치를 결정합니다. *offsetValue*는 0 또는 양수 값이어야 합니다. 경보 이벤트가 트리거된 후 모니터링된 속성 값이 *offsetValue* 및 `thresholdDirection`에 정의된 범위를 초과할 때까지 새 경보 이벤트가 트리거되지 않습니다.

thresholdDirection=`[RISING|FALLING]`
방향이 `RISING`인 경우 관찰된 속성 값이 *thresholdValue-offsetValue*보다 작을 때까지 경보 이벤트가 다시 트리거되지 않습니다. 방향이 `FALLING`인 경우 관찰된 속성 값이 *thresholdValue+offsetValue*보다 클 때까지 경보 이벤트가 다시 트리거되지 않습니다. *offsetValue*가 0인 경우에도 이 동작이 적용됩니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

`thrsh-job delete` *jobName*
*jobName*이라는 임계값 모니터링 작업을 삭제합니다. *jobName*은 `thrsh-job list` 하위 명령이 표시하는 이름이어야 합니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

`thrsh-job suspend` *jobName*
*jobName*이라는 임계값 모니터링 작업을 일시 중지합니다. 일시 중지된 작업은 활성 상태가 아니며 일정에 상관 없이 더 이상 데이터를 수집하지 않습니다. 그러나 작업은 정의된 상태로 유지되며 `thrsh-job resume` 하위 명령을 사용하여 다시 활성화할 수 있습니다. *jobName*은 `thrsh-job list` 하위 명령이 표시하는 이름이어야 합니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

`thrsh-job resume` *jobName*
*jobName*이라는 임계값 모니터링 작업을 다시 시작합니다. 다시 시작된 작업은 데이터 수집을 시작하고 일정에 따라 보고서를 보냅니다. *jobName*은 `thrsh-job list` 하위 명령이 표시하는 이름이어야 합니다. 이 하위 명령은 `thrsh-job suspend` 하위 명령에 대응합니다.

보안: 이 하위 명령은 Common Agent Container를 시작한 사용자만 실행할 수 있습니다.

예

다음 가상 시나리오는 mfwkadm 유틸리티를 해당 옵션 및 하위 명령과 함께 사용하는 방법을 보여줍니다.

list-modules 하위 명령은 현재 호스트에 있는 Java ES 구성 요소 인스턴스와 Common Agent Container에 있는 해당 모듈을 보여줍니다. 다음 예에서는 설치된 두 개의 구성 요소, 즉 실행 중인 인스턴스가 없는 Directory Server와 실행 중인 하나의 인스턴스가 있는 Web Server를 나열합니다.

```
$ mfwkadm list-modules

Installed products and their running instances:
===============================================

Instances for installed product: com.sun.cmm.ds:collectionID=/opt/SUNWdsee/ds6,
name=Sun Java(TM) System Directory Server,type=CMM_InstalledProduct
------------------------------

No instance.

Instances for installed product: com.sun.cmm.ws:collectionID=/var/opt/SUNWwbsvr7,
name=WebServer,type=CMM_InstalledProduct
------------------------------

/wsPrefix/com.sun.cmm.ws:name=https-hostname.example.com,type=CMM_ApplicationSystem
```

다음 info 하위 명령은 Web Server 인스턴스의 관찰 가능한 객체를 각 작업 유형에 대한 해당 클래스 및 속성과 함께 표시합니다.

```
$ mfwkadm info /wsPrefix/com.sun.cmm.ws:name=https-hostname.example.com,\\
type=CMM_ApplicationSystem

Information about running instance: /wsPrefix/com.sun.cmm.ws:
name=https-hostname.example.com,type=CMM_ApplicationSystem
==================================

Observable objects for performance jobs:
----------------------------------------

+ Objects of class: com.sun.cmm.settings.CMM_ApplicationSystemSetting

        /wsPrefix/com.sun.cmm.ws:name=https-hostname.example.com-setting,
type=CMM_ApplicationSystemSetting

        Observable attributes:

        Caption [STRING]
```

```
        ConfigurationDirectory [STRING]
        CreationClassName [STRING]
        Description [STRING]
        DirectoryName [STRING]
        ElementName [STRING]
        InstanceID [STRING]
        Name [STRING]
        URL [STRING]

+ Objects of class: com.sun.cmm.settings.CMM_KeepAliveSetting

        /wsPrefix/com.sun.cmm.ws:name=process-1-keepalive-setting,
type=CMM_KeepAliveSetting

        Observable attributes:

        AllocationUnit [STRING]
        Caption [STRING]
        ConnectionsUpperBound [LONG]
        CreationClassName [STRING]
        Description [STRING]
        ElementName [STRING]
        InputUnit [STRING]
        InstanceID [STRING]
        LowerAllocationLimit [LONG]
        LowerInputLimit [LONG]
        LowerOutputLimit [LONG]
        Name [STRING]
        OtherAllocationUnit [STRING]
        OtherInputUnit [STRING]
        OtherLowerAllocationLimit [LONG]
        OtherLowerInputLimit [LONG]
        OtherLowerOutputLimit [LONG]
        OtherOutputUnit [STRING]
        OtherUpperAllocationLimit [LONG]
        OtherUpperInputLimit [LONG]
        OtherUpperOutputLimit [LONG]
        OutputUnit [STRING]
        QueuedUpperBound [LONG]
        SecondsTimeout [LONG]
        TimeoutUpperBound [LONG]
        UpperAllocationLimit [LONG]
        UpperInputLimit [LONG]
        UpperOutputLimit [LONG]
        ...
```

다음 명령은 정의된 성능 모니터링 작업의 목록을 보여줍니다. 이 예에는 하나의 객체를 모니터링하는 `myPerfJob`이라는 하나의 성능 작업이 있습니다.

```
$ mfwkadm pm-job list

BY_OBJECTS performance jobs:
============================

Performance job information for: myPerfJob
------------------------------

Type:                BY_OBJECTS
State:               ACTIVE_ON_DUTY
Granularity period:  30
Reporting period:    1
By event:            EVENT_SINGLE
By file:             EVENT_SINGLE
Report format:       XML
Schedule:
        Global start time: Immediately
        Global stop time: Forever
        Weekly schedule: Everyday
        Daily schedule: All day
Observed objects:
                /wsPrefix/com.sun.cmm.ws:name=virtualServer-hostname.example.com-
webApp-/-stats,type=CMM_VirtualServerWebModuleStats
Observed attributes:
                All available

BY_CLASSES performance jobs:
============================

No jobs found.
```

다음 명령은 opstat-job info 또는 opstat-job observable-objects 하위 명령에서 얻은 두 개의 관찰 가능한 객체와 관련된 작업 상태 모니터링 작업을 만듭니다.

```
$ mfwkadm opstat-job create myOpStatJob granularity=60 \\
object=/wsPrefix/com.sun.cmm.ws:name=process-1,type=CMM_UnixProcess \\
object=/wsPrefix/com.sun.cmm.ws:name=process-1-DNSCache1,type=CMM_DnsCache
```

다음 명령은 위에서 만든 작업을 일시 중지합니다.

```
$ mfwkadm opstat-job suspend myOpStatJob
```

다음 명령은 잠재적 임계값 모니터링 작업에 대한 관찰 가능한 클래스를 보여줍니다.

```
$ mfwkadm thrsh-job observable-classes

Threshold jobs observable classes:
==================================
```

```
com.sun.cmm.cim.CIM_ScopedSettingData
com.sun.cmm.cim.CIM_SettingData
com.sun.cmm.cim.CIM_StatisticalData
com.sun.cmm.cim.statistics.CIM_EthernetPortStatistics
com.sun.cmm.cim.statistics.CIM_NetworkPortStatistics
com.sun.cmm.cim.statistics.j2ee.CIM_J2eeJVMStats
com.sun.cmm.cim.statistics.j2ee.CIM_J2eeStatistic
com.sun.cmm.settings.CMM_ApplicationSystemSetting
com.sun.cmm.settings.CMM_KeepAliveSetting
com.sun.cmm.settings.CMM_QueueTimeoutSetting
com.sun.cmm.settings.CMM_RFC2788ApplicationSystemSetting
com.sun.cmm.settings.CMM_ScopedSettingData
com.sun.cmm.settings.CMM_SoftwareResourceSetting
com.sun.cmm.settings.CMM_SWRBufferSetting
com.sun.cmm.settings.CMM_SWRLimitSetting
com.sun.cmm.settings.CMM_SWRQueueSetting
com.sun.cmm.settings.CMM_VirtualServerSetting
com.sun.cmm.statistics.CMM_ApplicationSystemStats
com.sun.cmm.statistics.CMM_ApplicationSystemWatchdogStats
com.sun.cmm.statistics.CMM_ConnectionQueueStats
com.sun.cmm.statistics.CMM_DnsCacheStats
com.sun.cmm.statistics.CMM_EthernetPortStats
com.sun.cmm.statistics.CMM_FileCacheStats
com.sun.cmm.statistics.CMM_HTTPResponsesStats
com.sun.cmm.statistics.CMM_JVMJSR174ExtStats
com.sun.cmm.statistics.CMM_JVMJSR174Stats
com.sun.cmm.statistics.CMM_JVMStats
com.sun.cmm.statistics.CMM_NetworkPortStats
com.sun.cmm.statistics.CMM_OperatingSystemStats
com.sun.cmm.statistics.CMM_ProcessorStats
com.sun.cmm.statistics.CMM_ProcessStats
com.sun.cmm.statistics.CMM_QueueTimeoutStats
com.sun.cmm.statistics.CMM_RFC2788ApplicationTableStats
com.sun.cmm.statistics.CMM_ServiceStats
com.sun.cmm.statistics.CMM_SoftwareResourceStats
com.sun.cmm.statistics.CMM_SolarisEthernetPortStats
com.sun.cmm.statistics.CMM_SolarisNetworkPortStats
com.sun.cmm.statistics.CMM_SolarisOperatingSystemStats
com.sun.cmm.statistics.CMM_SolarisProcessorStats
com.sun.cmm.statistics.CMM_SolarisProcessorSysinfoStats
com.sun.cmm.statistics.CMM_SolarisProcessorVmStats
com.sun.cmm.statistics.CMM_Statistic
com.sun.cmm.statistics.CMM_SWRBufferStats
com.sun.cmm.statistics.CMM_SWRCacheStats
com.sun.cmm.statistics.CMM_SWRLimitStats
com.sun.cmm.statistics.CMM_SWRQueueStats
com.sun.cmm.statistics.CMM_UnixOperatingSystemStats
```

```
com.sun.cmm.statistics.CMM_UnixProcessStats
com.sun.cmm.statistics.CMM_VirtualServerWebModuleStats
com.sun.cmm.statistics.CMM_WebModuleStats
```

다음 명령은 이전 예에서 발견된 com.sun.cmm.statistics.CMM_SWRQueueStats 클래스의 객체를 모니터링하는 임계값 작업에 대한 관찰 가능한 속성을 보여줍니다.

```
$ mfwkadm thrsh-job observable-attributes \\
class=com.sun.cmm.statistics.CMM_SWRQueueStats

Threshold jobs observable attributes:
=====================================

Class: com.sun.cmm.statistics.CMM_SWRQueueStats

Attributes:

BufferSize [LONG]
EntriesCount [LONG]
EntriesHighWaterMark [LONG]
EntriesLowWaterMark [LONG]
EntriesTotal [LONG]
ErrorCount [INTEGER]
FailedOperations [LONG]
LowerLimit [LONG]
OperationsCount [LONG]
OtherLowerLimit [LONG]
OtherUpperLimit [LONG]
OverflowsCount [LONG]
QueuedCount [LONG]
QueuedHighWater [LONG]
SampleInterval [LONG]
TotalQueuedCount [LONG]
UpperLimit [LONG]
```

다음 명령은 작업 작성의 또 다른 예이고 여기서는 임계값 작업이 포함됩니다.

```
$ mfwkadm thrsh-job create myThreshJob granularity=30 \\
object=/wsPrefix/com.sun.cmm.ws:name=process-1-threadPool-NativePool-stats,\\
type=CMM_SWRQueueStats attributeName=EntriesCount attributeType=LONG \\
thresholdValue=1000 thresholdOffset=10 thresholdDirection=RISING
```

다음 예는 이전 예에서 만들어진 임계값 모니터링 작업에 대한 thrsh-job info 하위 명령의 출력을 보여줍니다.

```
$ mfwkadm thrsh-job info myThreshJob

Threshold job information for: myThreshJob
```

```
----------------------------

Type:                SIMPLE
State:               ACTIVE_ON_DUTY
Granularity period:  30
Schedule:
        Global start time: Immediately
        Global stop time: Forever
        Weekly schedule: Everyday
        Daily schedule: All day
Alarm configuration:
                Type: QualityOfServiceAlarm
                Severity: INDETERMINATE
Threshold definition(s):
                Object: /wsPrefix/com.sun.cmm.ws:name=process-1-threadPool-
NativePool-stats,type=CMM_SWRQueueStats
                        Attribute: EntriesCount [LONG]
                        Value: 1000
                        Direction: RISING
                        Offset: 10
```

종료 상태

다음 종료 값이 반환됩니다.

0 성공적으로 완료됨

1 오류가 발생함

속성

속성 유형	속성 값
가용성	SUNWmfwk
인터페이스 안정성	민간 계약

참조

cacao.5, cacaoadm.1m

◆ ◆ ◆ 3 장

3 Monitoring Console 설치 및 사용

Monitoring Console은 계측에 의해 수집된 모든 모니터링 데이터를 표시하는 웹 기반 응용 프로그램입니다. 이 응용 프로그램은 마스터 에이전트에 의존하여 각 노드 에이전트의 모든 값과 경보 알림을 집계합니다.

Monitoring Console이 설치되고 나면 모든 호스트의 간단한 브라우저 창에서 안전하게 액세스할 수 있습니다. 방화벽에서 허용하도록 구성된 경우에는 인터넷을 통해 액세스할 수도 있습니다. 그런 다음 그래픽 인터페이스를 사용하여 모니터링된 값을 실시간으로 표시하고 경보를 확인하고 사용자 정의 경보를 트리거하기 위한 규칙을 만들 수 있습니다.

주 – 설치 또는 구성을 수행하기 전에 **Sun Java Enterprise System 5 UNIX용 릴리스 노트**를 참조해야 합니다.

이 장의 내용은 다음과 같습니다.

Monitoring Console 설치

이 릴리스에서는 마스터 에이전트의 제한 사항으로 인해 노드 에이전트와 동일한 호스트에 마스터 에이전트를 가질 수 없습니다. 결과적으로 Monitoring Console을 Java ES의 다른 모니터링된 구성 요소와 동일한 호스트에 설치할 수 없습니다. Solaris 영역을 구성하지 않은 경우 고유한 호스트에 설치해야 합니다. 자세한 내용은 아래의 49 페이지 “Solaris 영역에 Monitoring Console을 설치하는 방법”를 참조하십시오.

Monitoring Console을 설치하면 또한 Monitoring Framework가 공유 구성 요소 종속성으로 설치됩니다. 콘솔에서는 마스터 에이전트를 로드하기 위해 프레임워크와 Common Agent Container가 필요하지만 노드 에이전트와 달리 마스터 에이전트는 사용자가 구성할 수 없습니다. 특히 Monitoring Console을 설치하는 호스트나 영역에서 `mfwkadm` 명령을 사용하지 않아야 합니다.

▼ Java ES 설치 프로그램을 사용하여 Monitoring Console을 설치하는 방법

이 베타 릴리스의 제한 사항으로 인해 다른 Java ES 구성 요소가 설치되지 않은 호스트나 Solaris 영역에 Monitoring Console을 설치해야 합니다. 결과적으로 Monitoring Console은 이 절차에서 설치하는 유일한 구성 요소입니다.

이 절차에서는 설치 프로그램의 그래픽 인터페이스를 사용합니다. 다른 모드에서 설치 프로그램을 실행하는 방법에 대한 자세한 내용은 **Sun Java Enterprise System 5 UNIX용 설치 설명서**의 4 장, "텍스트 기반 인터페이스를 사용하여 설치" 및 **Sun Java Enterprise System 5 UNIX용 설치 설명서**의 5 장, "자동 모드로 설치"를 참조하십시오.

1. **Java ES 릴리스의 플랫폼에 해당하는 디렉토리에서 `installer` 응용 프로그램을 시작합니다. 자세한 내용은 Sun Java Enterprise System 5 UNIX용 설치 설명서의 "설치를 시작하려면"을 참조하십시오.**

2. **시작 화면을 지나서 라이센스에 동의하고 업그레이드 또는 설치를 선택한 후 새 소프트웨어 설치를 선택한 후 다음을 누릅니다.**

3. **구성 요소 선택 화면에서 Sun Java System Monitoring Console만 설치하도록 선택합니다. 다음을 누릅니다.**

4. **설치 프로그램은 공유 구성 요소에 필요한 업그레이드를 확인합니다. 작업이 끝나면 다음을 누릅니다.**

5. **이제 설치 프로그램은 시스템 요구 사항을 확인합니다. 운영 체제에 패치가 필요한 경우 설치를 취소하고 필요한 패치를 시스템에 추가한 후 이 절차를 다시 시작합니다. 그렇지 않은 경우 다음을 누릅니다.**

6. **구성 유형 선택 화면에서 지금 구성을 선택하고 다음 사용자 정의 구성 화면에서 다음을 누릅니다.**

7. **설치 프로그램에서 Monitoring Console을 설치할 준비가 되었으므로 다음을 눌러 설치를 시작합니다. 설치 도중에 Java ES 배포를 아직 등록하지 않은 경우 제품 등록 창을 열 수 있습니다.**

8. **설치가 완료되면 설치 요약 및 로그를 검토한 다음 설치 완료를 눌러 설치 프로그램을 종료합니다.**

다음 순서 이제 49 페이지 "Monitoring Console을 구성하는 방법"을 진행해야 합니다.

▼ Solaris 영역에 Monitoring Console을 설치하는 방법

Solaris 영역을 사용하여 Java ES의 다른 구성 요소와 동일한 물리적 호스트에 Monitoring Console을 설치할 수 있습니다. 이러한 구성 요소는 전역 영역에 존재하며 사용자는 Monitoring Console에 대한 논리 호스트가 되도록 스파스 루트 로컬 영역을 만듭니다. 다음 순서대로 진행합니다.

1. **전역 영역에서 Monitoring Console을 제외한 모든 Java ES 구성 요소를 설치 및 구성합니다. 모든 서버 인스턴스가 실행 중이 되도록 전역 영역에서 선택한 구성 요소의 모든 설치 후 구성을 완료합니다.**

2. **전역 영역에서 설치의 일부로 Monitoring Framework는 전역 영역에 공유 구성 요소로 설치됩니다. 설치된 구성 요소에 적용할 수 있는 2 장의 모든 절차를 수행합니다.**

3. **동일한 호스트에서 Monitoring Console에 대한 논리적 호스트로서 스파스 루트 로컬 영역을 만듭니다. 이 영역이 스파스 루트 영역이므로 *mfwk-base*에 설치된 Monitoring Framework를 볼 수 있어야 합니다(9 페이지 "기본 경로 및 파일 이름" 참조).**

4. **48 페이지 "Java ES 설치 프로그램을 사용하여 Monitoring Console을 설치하는 방법"의 절차에 따라 스파스 루트 로컬 영역에 Monitoring Console을 설치합니다.**

5. **다음 명령을 사용하여 스파스 루트 영역에서 Monitoring Framework를 구성합니다.**

   ```
   cd mfwk-base/bin
   ./mfwksetup -i
   ```

   전역 영역의 파일을 사용하여 이 명령은 로컬 영역에서 필요한 Monitoring Framework 구성 파일을 만듭니다.

다음 순서 이제 49 페이지 "Monitoring Console을 구성하는 방법"을 진행해야 합니다.

▼ Monitoring Console을 구성하는 방법

이 절차에서는 별개의 물리적 호스트에서 Monitoring Console을 구성하는 방법에 대해 설명합니다. Solaris 영역에 의해 작성된 논리적 호스트에 Monitoring Console을 설치한 경우 명령은 동일하지만 해당 영역의 파일 시스템 내에서 명령을 실행해야 합니다.

1. **다음 명령으로 Monitoring Framework를 사용하여 마스터 에이전트를 초기화합니다.**

```
cd mfwk-base/bin
./masetup -i
```

2. **다음 명령을 사용하여 Common Agent Container(cacao)를 다시 시작합니다.**

```
cacaoadm restart
```

▼ Monitoring Console 구성을 해제하는 방법

다른 구성 요소를 설치하려는 호스트에 Monitoring Console을 설치하여 구성한 경우 Monitoring Console의 충돌로 인해 이러한 구성 요소를 모니터링할 수 없습니다. 노드 에이전트를 사용하여 새 구성 요소를 모니터링하려면 Monitoring Console의 마스터 에이전트를 구성 해제해야 합니다.

- **루트로 다음 명령을 실행하여 Monitoring Console을 구성 해제합니다.**

```
cacaoadm stop
cacaoadm unregister-module com.sun.mfwk.masteragent.xml
cacaoadm register-module /etc/mfwk-base/xml/com.sun.mfwk.xml
cacaoadm restart
```

설치된 디렉토리 레이아웃

운영 체제에 설치되는 패키지의 이름은 **Sun Java Enterprise System 5 UNIX용 설치 참조 설명서**의 5 장, "설치 패키지 목록"을 참조하십시오. 다음 표에서는 Monitoring Console 패키지의 디렉토리에 대해 설명합니다. 9 페이지 "기본 경로 및 파일 이름"에 설명된 대로 기본 설치 디렉토리 *MConsole-base*에는 다음 의미가 있습니다.

- Solaris 시스템: `/opt/SUNWjesmc`
- Linux 시스템: `/opt/sun/jesmc`

표 3–1 Monitoring Console에 사용되는 디렉토리 및 파일

경로	내용 설명
MConsole-base/WEB-INF/classes	웹 응용 프로그램 서블릿 클래스
MConsole-base/WEB-INF/lib	웹 응용 프로그램 JAR 종속성
MConsole-base/WEB-INF/*.xml	웹 응용 프로그램 설명자
MConsole-base/css	스타일시트 파일
MConsole-base/html	HTML 파일

표 3–1 Monitoring Console에 사용되는 디렉토리 및 파일 *(계속)*

경로	내용 설명
MConsole-base/images	사용자 인터페이스에서 사용되는 GIF 이미지 파일
MConsole-base/js	JavaSript™ 파일
MConsole-base/*.jsp	JavaServer Pages™ 파일
WebConsole-base/prereg/jesmc/*.reg	Monitoring Console용 Web Console 파일

Monitoring Console 시작

Monitoring Console은 설치되어 있는 호스트에 연결할 수 있는 모든 브라우저를 통해 사용할 수 있는 웹 응용 프로그램입니다. 동일한 호스트에 자동으로 설치되는 Web Console을 통해 Monitoring Console에 액세스합니다. 다음 절차에서는 Monitoring Console에 액세스하여 모니터링된 구성 요소를 보는 방법에 대해 설명합니다.

▼ Monitoring Console을 시작하는 방법

1 **먼저 Web Console에 대한 Web Server를 다시 시작해야 합니다. Monitoring Console을 설치한 호스트나 영역에서 다음 명령을 실행합니다.**

```
/usr/sbin/smcwebserver restart
```

2 **Web Console이 시작되기를 기다립니다. 다음 명령을 사용하여 준비가 되었는지 확인합니다.**

```
/usr/sbin/smcwebserver status
```

다음 메시지가 나타날 때까지 이 명령을 여러 번 실행해야 할 수 있습니다.

```
Sun Java(TM) Web Console is running.
```

3 **Monitoring Console 호스트에 연결할 수 있는 모든 브라우저에서 다음 URL을 사용하여 Web Console을 엽니다. Solaris 영역에 설치한 경우 *MC-host*는 해당 영역에 제공한 논리적 호스트 이름입니다.**

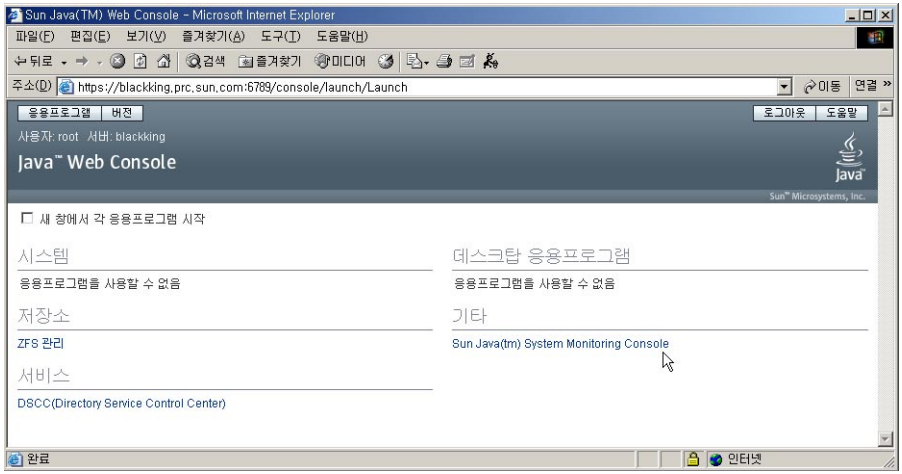
```
https://MC-host.domain:6789
```

4 **브라우저가 구성된 방법에 따라 신뢰할 수 없는 인증서에 대한 메시지가 표시될 수 있습니다. Web Console에 액세스하려면 인증서를 신뢰해야 합니다.**

5 **메시지가 나타나면 Monitoring Console 호스트의 루트 비밀번호를 사용하여 `root`로 Web Console에 로그인합니다.**

로그인하면 Web Console은 제공되는 모든 서비스를 나열합니다.

6 **Monitoring Console의 주 창을 열려면 다음 화면 캡처와 같이 "기타"라는 머리글 아래의 Sun Java System Monitoring Console을 누릅니다.**



▼ 노드 에이전트에 연결하는 방법

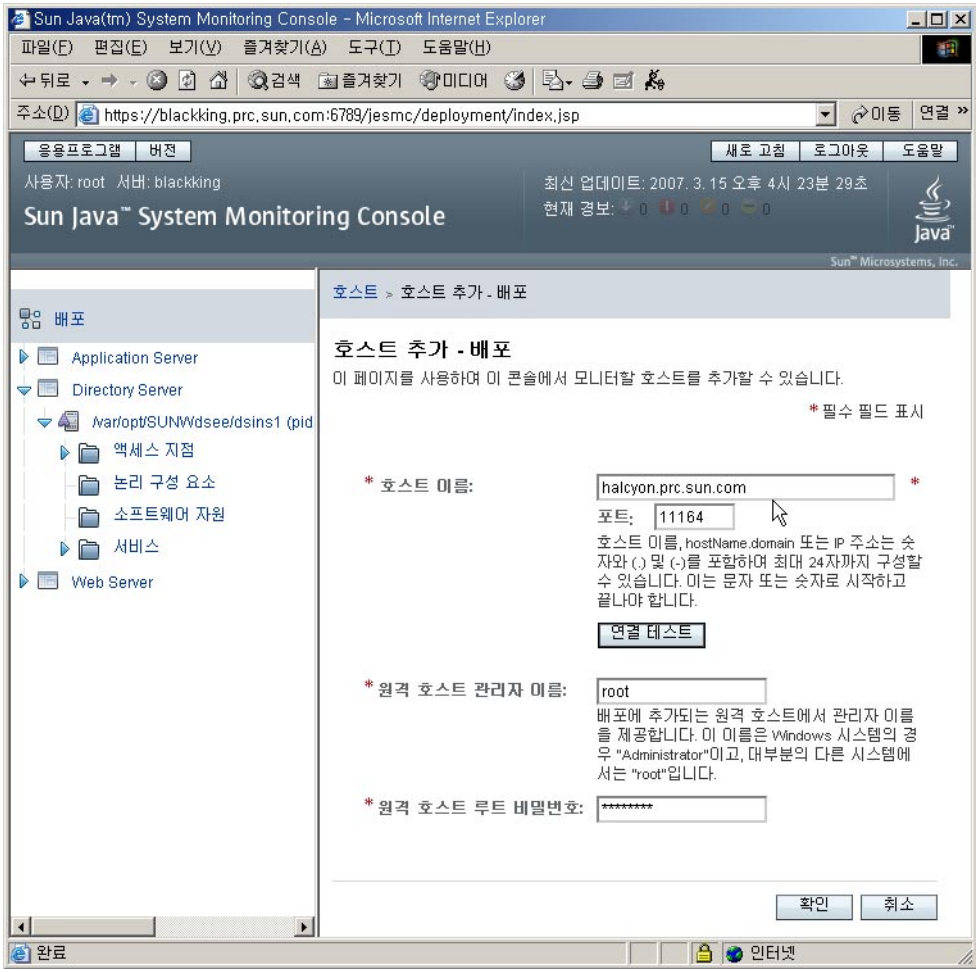
Monitoring Console이 처음 시작될 경우 모니터링된 구성 요소가 호스팅되는 위치를 나타내야 합니다. Java ES 배포에서 각 노드 에이전트의 위치를 지정하면 콘솔은 각 노드 에이전트의 모든 구성 요소를 자동으로 표시합니다. 또한 나중에 Java ES 구성 요소를 새 호스트에 설치하여 배포에 추가할 경우 이 절차를 반복해야 합니다.

노드 에이전트가 추가되고 나면 명시적으로 제거될 때까지 Monitoring Console은 사용자가 콘솔에 액세스할 때마다 노드 에이전트에 다시 연결됩니다. 이전에 추가한 노드 에이전트가 응답하지 않을 경우 27 페이지 "노드 에이전트를 다시 시작하는 방법" 절차를 따릅니다.

1 **Monitoring Console이 설치되는 논리적 호스트와 모니터링할 노드 에이전트 및 Java ES 구성 요소를 포함하는 호스트 간에 날짜와 시간을 동기화합니다. 동기화가 자동인지 아니면 수동인지 여부에 상관 없이 각 호스트에서의 시간은 서로 약 10분 이내여야 합니다.**

2 **필요한 경우 Monitoring Console의 왼쪽에 있는 계층의 루트에서 배포 링크를 눌러 배포 수준 표시로 이동합니다. 이제 오른쪽 창에서 호스트 탭을 선택하고 추가를 누릅니다.**

3 **호스트 추가 대화 상자가 표시되면 다음 화면 캡처와 같이 필요한 정보를 입력합니다.**



- 호스트 이름: 모니터링된 구성 요소를 구성한 노드 에이전트의 정규화된 호스트 이름을 입력합니다.
- 포트: 11164(노드 에이전트가 상주하는 호스트에서 Monitoring Framework를 다르게 구성하지 않은 경우)
- 원격 호스트 루트 비밀번호: 노드 에이전트가 상주하는 시스템에 대한 루트 비밀번호를 입력합니다.

4 연결 테스트를 누릅니다.

연결 정보가 올바르고 호스트 에이전트가 구성되어 실행 중이면 이제 연결되었다는 것이 대화 상자에 표시됩니다.

5 **확인을 눌러 호스트 추가 대화 상자를 끝내면 호스트 목록에 새 이름이 나타납니다. 이제 호스트의 노드 에이전트에 있는 모니터링된 모든 구성 요소가 왼쪽 열에도 표시됩니다.**

6 **모니터링된 구성 요소가 설치된 Java ES의 모든 호스트에 대해 이 절차를 반복합니다.**

다음 순서 이제 왼쪽 열에 나열된 구성 요소를 탐색하여 해당 작업 상태, 표시되는 모니터링된 속성, 트리거된 모든 경보 등을 확인합니다.

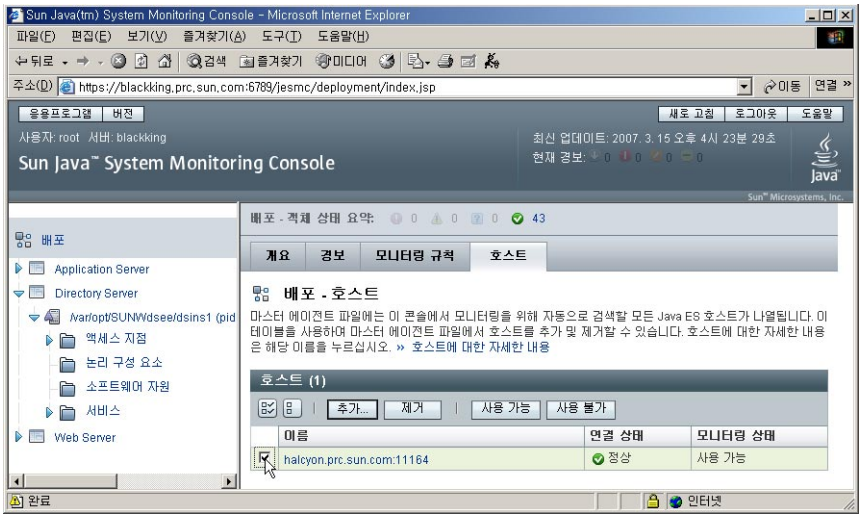
Monitoring Console 사용

이 절의 절차에서는 Monitoring Console과 상호 작용하는 방법에 대해 설명합니다.

▼ 모니터링을 선택적으로 비활성화하고 다시 활성화하는 방법

Java ES 모니터링 메커니즘은 프로덕션 시스템의 성능에 영향을 주지 않도록 하기 위해 경량급으로 설계되었습니다. 그러나 계측이 성능에 거의 영향을 주지 않도록 모니터링 값 수집을 중지하는 것이 바람직한 경우가 있습니다. Monitoring Console은 다음 절차에 설명된 대로 호스트 단위로 이 작업을 수행하는 방법을 제공합니다.

1 **필요한 경우 Monitoring Console의 왼쪽에 있는 계층의 루트에서 배포 링크를 눌러 배포 수준 표시로 이동합니다. 그런 다음 오른쪽 창에 있는 호스트 탭을 누릅니다.**



호스트 탭의 테이블은 Monitoring Console에 의해 모니터링되는 Java ES 구성 요소를 포함하는 모든 호스트를 나열합니다.

2 **테이블의 왼쪽 열에 있는 확인란을 사용하여 모니터링을 중지할 모든 호스트를 선택합니다. 호스트 테이블의 맨 위에 있는 비활성화를 누릅니다.**

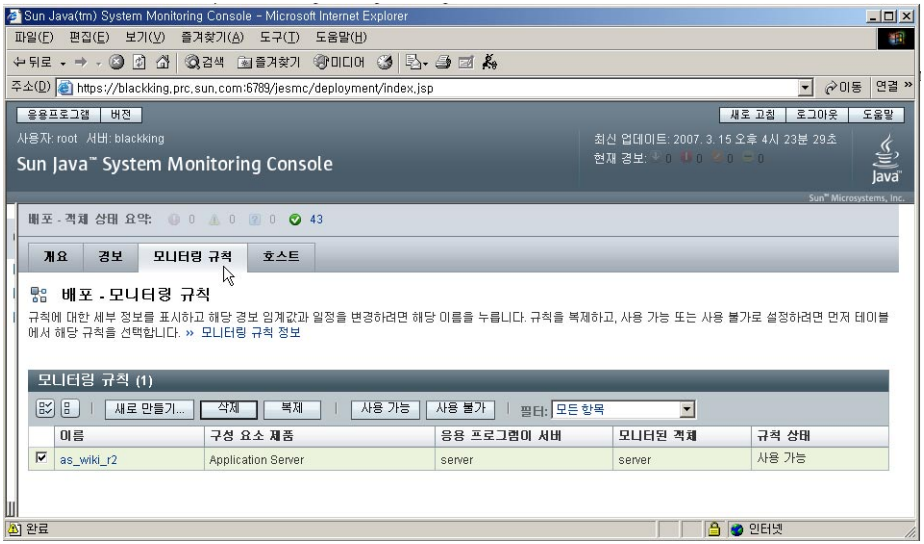
자세한 정보 **결과**

호스트의 모니터링이 비활성화될 경우 계층에서 해당 호스트 아래의 모든 모니터링 객체가 비활성화됩니다. 비활성화된 상태에서 모니터링된 객체는 여전히 마지막 값을 포함할 수 있지만 더 이상 업데이트되지 않습니다. 비활성화된 객체에 의존하는 모니터링 규칙은 일시 중지됩니다. 비활성화된 호스트를 활성화하려면 호스트 테이블의 맨 위에 있는 활성화 버튼을 사용하여 이 절차를 수행합니다.

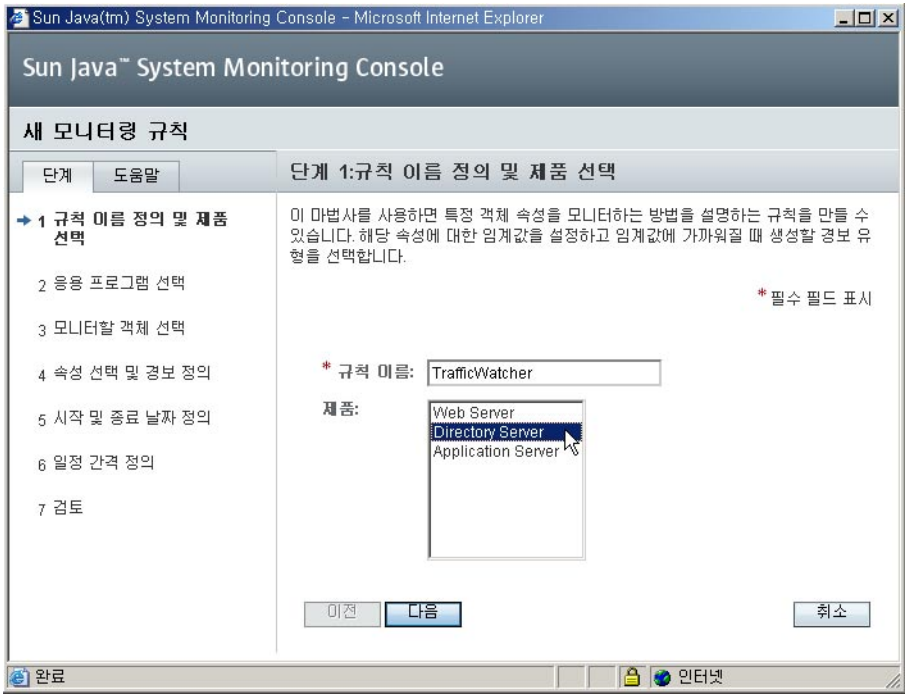
▼ 새 모니터링 규칙을 만드는 방법

모니터링 작업이라고도 하는 모니터링 규칙은 경보를 트리거하기 위해 사용자가 정의되는 모니터링된 값의 조건 집합입니다. Monitoring Console의 모니터링 규칙 마법사를 사용하면 모니터링할 조건을 정의할 수 있습니다.

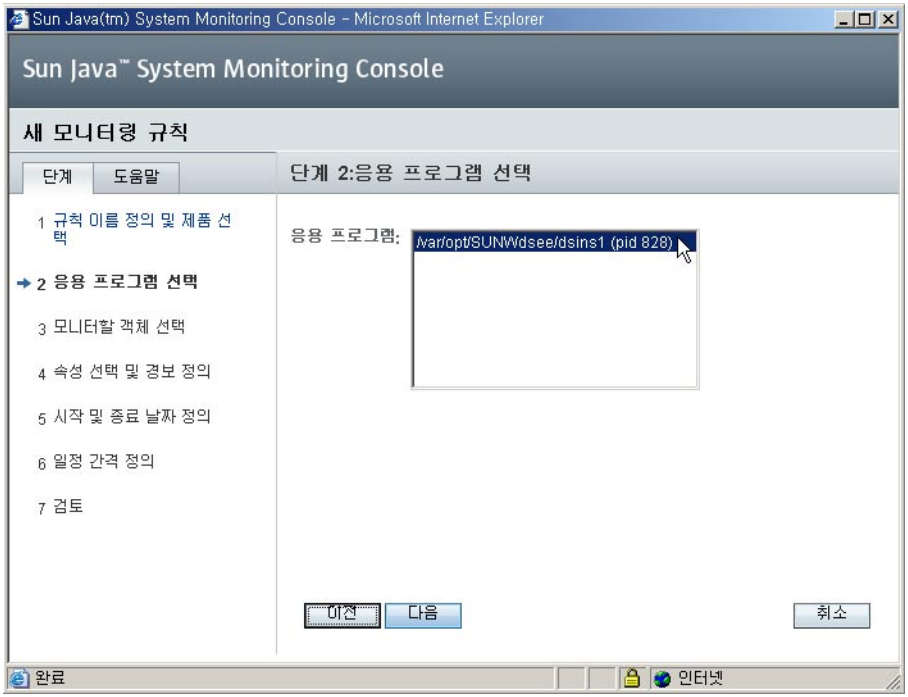
1 **필요한 경우 Monitoring Console의 왼쪽에 있는 계층의 루트에서 배포 링크를 눌러 배포 수준 표시로 이동합니다. 이제 다음 화면 캡처와 같이 오른쪽 창의 규칙 탭을 선택하고 모니터링 규칙 테이블에서 새로 만들기를 누릅니다.**



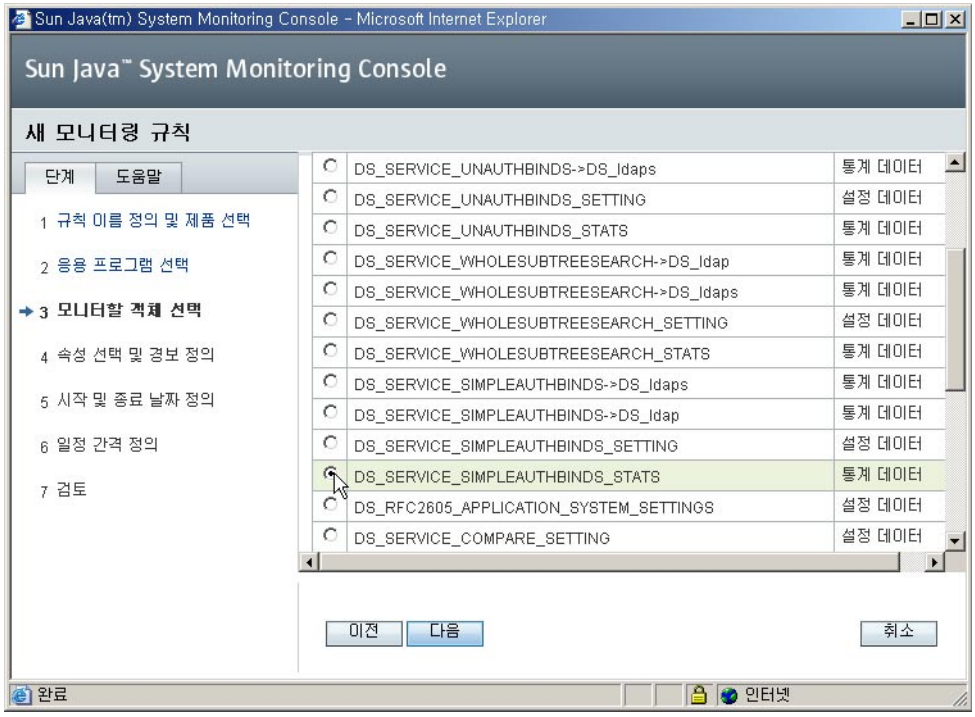
2 **새 모니터링 규칙에 이름을 제공하고 모니터링할 서버 유형을 선택합니다.**



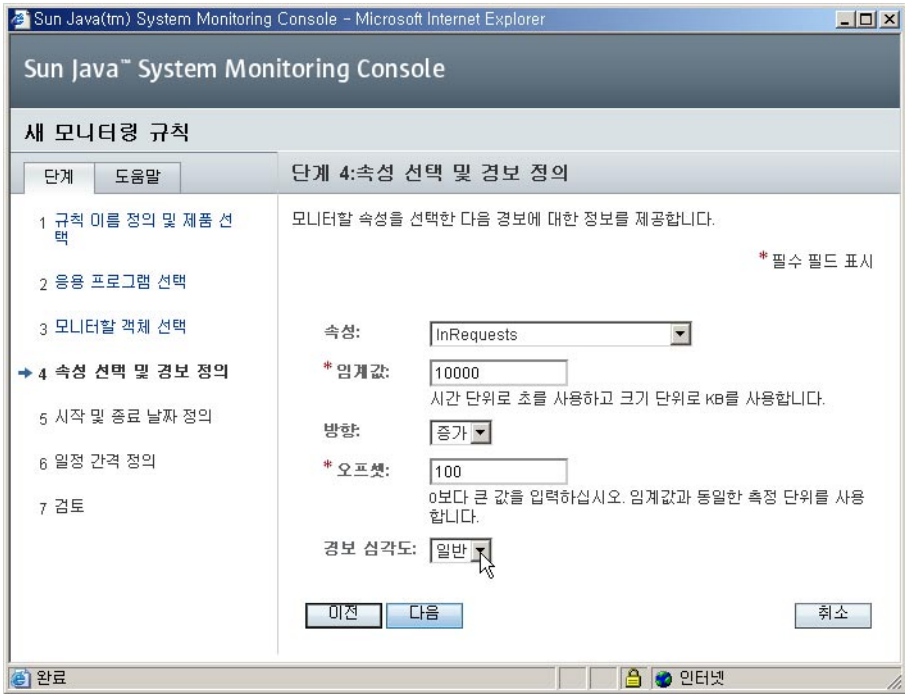
3 모니터링할 구성 요소 제품의 인스턴스를 선택합니다. 별개의 호스트에 설치된 동일한 제품의 두 인스턴스가 있는 경우 일부 인스턴스는 이 테이블에서 동일한 이름을 가질 수 있습니다. 이 경우 인스턴스는 왼쪽 창에서 계층에 표시되는 것과 동일한 순서일 수 있지만 이를 알 수 있는 확실한 방법은 없습니다. 규칙이 정의되도록 두 인스턴스에서 동일한 모니터링 규칙을 만들어야 할 수 있습니다.



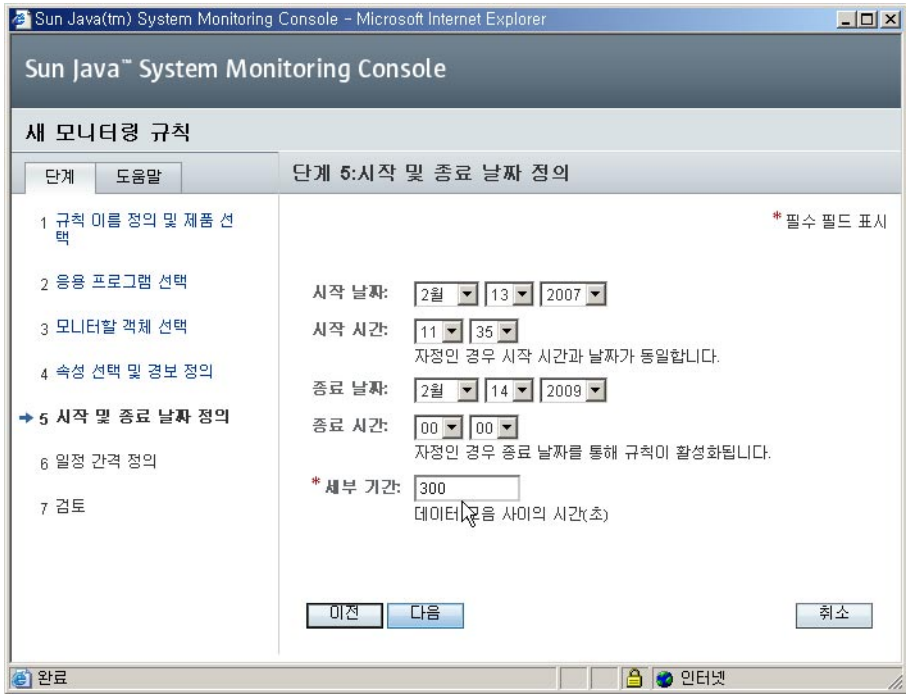
4 모니터링할 속성이 포함된 객체를 선택합니다.



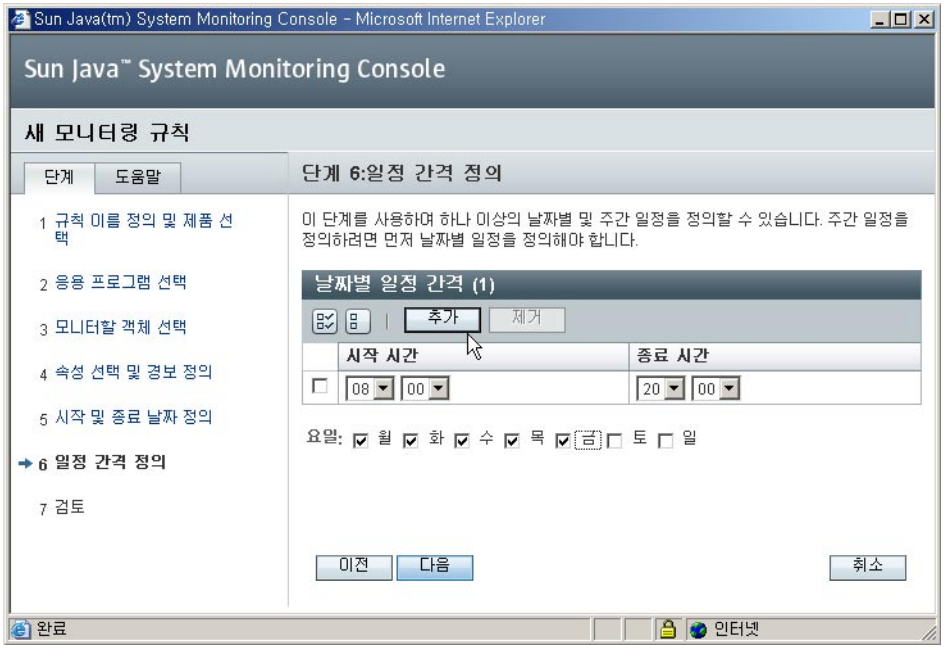
5 이제 마지막으로 경보를 생성하는 값과 함께 모니터링할 속성을 지정할 수 있습니다.



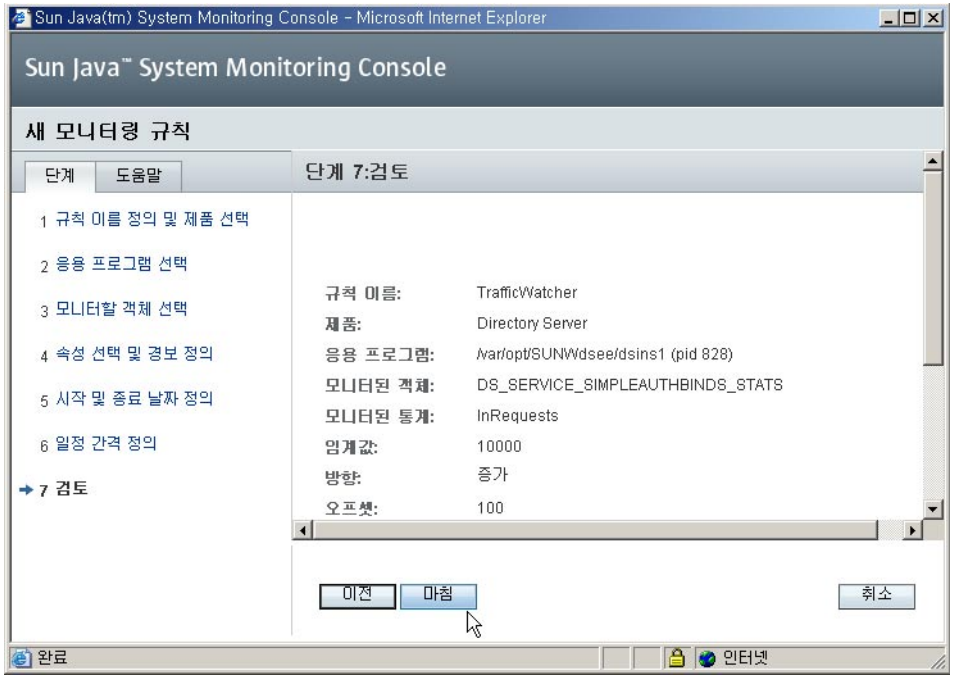
6 규칙의 시작 및 종료 날짜를 입력합니다. 규칙의 활성 또는 비활성화를 결정하는 일정과 달리 시작 및 종료 날짜는 규칙이 종료하도록 정의된 시간을 결정합니다. 시작 날짜가 과거이면 기본값에서 항상 그런 것처럼 이 규칙과 연관된 모니터링이 즉시 시작됩니다.



7 원하는 경우 컨트롤을 사용하여 규칙이 활성 모니터링 상태가 되는 하나 이상의 시간 범위를 만듭니다. 주간 일정을 만들기 위해 요일을 선택할 수도 있습니다.



8 **규칙 마법사의 마지막 단계에서는 입력을 검토하고 마침을 눌러 새 규칙을 만듭니다.**



규칙 마법사가 끝나면 규칙 테이블에 새 규칙이 있는 모니터링 규칙 탭을 다시 확인해야 합니다.

Monitoring Console 문제 해결

Sun Java Enterprise System 5 UNIX용 릴리스 노트에 나열된 알려진 문제도 참조하십시오.

마스터 에이전트가 노드 에이전트와 충돌할 경우 다음 조건을 확인합니다.

- Solaris 영역을 사용하는 경우 Monitoring Console을 스파스 루트 로컬 영역에 설치했는지 확인합니다.
- 모니터링된 구성 요소의 이전 설치가 호스트나 영역에 남아 있지 않는지 확인합니다.
- 위 경우에 해당한다면 Monitoring Console과 모든 구성 요소를 제거하고 문제를 해결한 다음 Monitoring Console 설치를 다시 실행해야 합니다.

Monitoring Console을 제거하고 동일한 호스트에 다시 설치할 경우 초기화에 실패하고 Web Console에 표시되지 않습니다. 이 경우에는 Monitoring Console 호스트에서 `masetup -i` 명령을 실행하여 마스터 에이전트를 초기화합니다. 그런 다음 51 페이지 “Monitoring Console을 시작하는 방법” 절차를 따릅니다.

모니터링 규칙에는 활성 상태인 규칙만 비활성화할 수 있다는 제한 사항이 있습니다. 현재 시점에 규칙을 비활성화하는 일정이 있는 규칙을 비활성화하려면 규칙을 활성화하도록 해당 일정을 변경하거나 규칙을 완전히 제거해야 합니다.

Windows 플랫폼의 제한 사항으로 인해 호스트 통계의 `handleCount` 및 `threadCount` 값은 항상 0입니다.

◆ ◆ ◆ 부 록 A

CMM 객체 참조

CMM(Common Monitoring Model)은 Java 프로그래밍 언어에서 구현되는 CIM(Common Information Model)의 확장입니다. CIM은 com.sun.cmm.cim.* 패키지의 Java 인터페이스에서 구현되고 CMM은 CIM 인터페이스를 확장하는 `com.sun.cmm.*` 패키지의 인터페이스에서 구현됩니다. 모니터링된 객체는 CMM 인터페이스를 구현하는 클래스에 의해 노드 에이전트에서 나타납니다. 다음 표에서는 각 객체 클래스에 대해 모니터링할 수 있는 속성을 보여줍니다.

CMM 객체 개요

CMM은 해당 유형의 모니터링된 객체가 표시할 수 있는 속성을 정의하는 핵심 인터페이스의 제한된 집합에 기반을 둡니다. 다음 목록에서는 CMM에 의해 정의되는 광범위한 유형의 모니터링된 객체를 나타내는 클래스와 몇 가지 해당 주요 속성을 보여줍니다.

`CMM_InstalledProduct` 전체적인 관점에서의 Java ES 구성 요소 제품입니다. 예를 들어, Java ES Directory Server입니다.

`CMM_ApplicationSystem` Java ES 구성 요소 제품의 설치 및 구성된 인스턴스입니다. 이 인스턴스는 실행 중이거나 실행 중이 아닐 수 있습니다. 이 객체의 일반적인 속성은 관리자의 연락처 정보, 시스템의 작업 상태, 응용 프로그램 시작 또는 중지 시간입니다.

`CMM_Service` 구성 요소 제품의 특정 기능(예: Java ES Directory Server 인증 서비스)입니다. 일반적인 속성은 서비스의 작업 상태입니다.

`CMM_SoftwareResource` 환경에 있는 소프트웨어 엔티티(예: 캐시, 스레드 풀 등)를 나타냅니다. 일반적인 속성은 캐시 크기입니다.

CMM_LogicalComponent	서비스의 의해 조작되고 최종 사용자가 볼 수 있지만 실제 물리적 자원이나 소프트웨어 기능을 나타내지 않는 엔티티입니다. 예를 들어, 인스턴스 그 자체가 아닌 소프트웨어 인스턴스에 대한 구성 매개 변수의 세트일 수 있습니다.
CMM_ServiceAccessURI	서비스를 사용할 수 있게 되는 지점입니다. 일반적인 속성은 포트 번호 또는 URI(Uniform Resource Identifier)입니다.
CMM_RemoteServiceAccessPoint	원격 연결을 위한 액세스 및 주소 지정 정보입니다. 일반적인 속성은 URI 또는 연결의 작업 상태(열림 또는 닫힘)입니다.
CMM_Process	실행 프로그램의 단일 인스턴스입니다. 일반적인 속성은 메모리 또는 CPU 사용입니다.
CMM_UnitaryComputerSystem	Java ES 배포에 사용되는 단일 호스트(예: 데스크탑 시스템 또는 서버)입니다. 일반적인 속성은 사용 가능한 프로세서 수 또는 물리적 메모리 양입니다.
CMM_OperatingSystem	호스트 시스템의 하드웨어를 사용할 수 있게 만드는 소프트웨어나 펌웨어입니다. 일반적인 속성은 시스템에서 사용할 수 있는 가상 메모리 양입니다.
CMM_JVM	Java ES 서버에 사용되는 Java Virtual Machine입니다. 예제 속성은 Java Virtual Machine의 버전 번호입니다.
CMM_DatabaseService	데이터베이스를 대신하여 수행되는 작업입니다(예: 사용자 액세스 제공). 일반적인 속성은 데이터베이스 허용되는 최대 연결 수입니다.
CMM_CommonDatabase	지정된 유형의 데이터베이스에서 공통된 등록 정보입니다. 일반적인 속성은 최근 백업의 날짜입니다.

◆ ◆ ◆ 부 록 B

각 구성 요소가 표시하는 모니터링된 객체

이 부록의 절에서는 모니터링을 지원하는 각 제품 구성 요소에서 계측된 CMM 객체를 나열합니다. 객체 속성의 하위 집합만 계측된 경우 이러한 속성도 나열됩니다.

Common Agent Container의 계측

아직 문서화되지 않았습니다.

Access Manager의 계측

아직 문서화되지 않았습니다.

Application Server의 계측

아직 문서화되지 않았습니다.

Calendar Server의 계측

아직 문서화되지 않았습니다.

Directory Server의 계측

아직 문서화되지 않았습니다.

Instant Messaging의 계측

아직 문서화되지 않았습니다.

Messaging Server의 계측

아직 문서화되지 않았습니다.

Portal Server의 계측

아직 문서화되지 않았습니다.

Web Server의 계측

아직 문서화되지 않았습니다.

색인