Sun Fire™ B1600용 Sun™ SNMP 관리 에이전트 안내서

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

부품 번호 817-2502-10
2003 년 4 월 , 개정판 01

이 문서에 대한 의견은 다음 주소로 보내 주십시오. http://www.sun.com/hwdocs/feedback

목차

그림

표

코드 예

머리말

이 안내서는 Sun Fire B1600 플랫폼을 위한 Sun SNMP 관리 에이전트에 대해 설명하는데, 이것은 단순 네트워크 관리 프로토콜(SNMP)을 사용한 플랫폼 하드웨어의 관리를 지원합니다.

관리 에이전트는 재고명세, 구성 및 환경과 결함 보고의 *모니터링*을 제공합니다. 또한 서비스 표시기, 전원 및 프로세서 블레이드의 대기 및 재설정의 *제어* 및 *모니터링*을 제공합니다.

이 안내서는 숙련된 엔터프라이즈 관리자와 전문 개발자를 위한 것입니다.

이 안내서는 다음 두 파트로 나뉘어집니다.

- 1부(1 장 - 8 장)는 SNMP 관리 에이전트를 소개하고 그의 기능을 설명합니다.
- 2부(9 장 - 13 장)는 소프트웨어의 설치 및 구성 방법을 설명합니다.

이 책의 구성

이 안내서는 다음 장들로 구성되어 있습니다.

1 부

1 장은 Sun SNMP 관리 에이전트 소프트웨어의 구성요소를 설명합니다.

2 장은 SNMP(단순 네트워크 관리 프로토콜)의 본질적인 기능에 대한 간략한 소개를 제공합니다.

3 장에서는 SNMPv3 마스터 에이전트의 기능과 특징에 대해 설명합니다.

4 장은 SNMP가 Sun Fire B1600을 모델화하는 방법의 개요를 제공합니다.

5 장에서는 SNMP 인터페이스가 Sun Fire B1600 관리 개체 및 그들의 관계를 제공하는 방법에 대해 설명합니다.

6 장에서는 sunPlat 물리적 클래스 계층과 sunPlat 모델에서 정의되는 관리 물리적 개체 클래스가 SUN-PLATFORM-MIB에 의해 표시되는 방법에 대해 설명합니다.

7 장에서는 sunPlat 논리적 클래스 계층과 sunPlat 모델에서 정의되는 관리 개체 클래스가 SUN-PLATFORM-MIB에 의해 표시되는 방법에 대해 설명합니다.

8 장에서는 SUN–PLATFORM–MIB에 정의된 것처럼 SunPlat 통지 클래스 및 속성에 대해 설명합니다.

2 부

9 장은 Sun Fire B1600을 위한 관리 소프트웨어를 구성하는 구성요소를 설명하고 SNMP 소프트웨어를 설치하기 전에 수행해야 하는 시스템 점검을 나열합니다.

10 장은
Sun Fire B1600에 관리 소프트웨어를 설치하는 방법에 대해 설명합니다.

11 장은 사용자가 구성할 수 있는 파일에 대한 정보를 제공합니다.

12 장에서는 설치 후의 기본 구성에 대해 설명하고 구성 파일을 수정하는 방법을 설명합니다.

13 장에서는 소프트웨어 설치 제거 방법에 대해 설명합니다.

14 장은 소프트웨어 문제 해결에 대한 도움말을 제공합니다.

부록 A는 J2RE를 J2SE와 공존하도록 설치하는 방법과 설치를 찾기 위해 시작 스크립트를 수정하는 방법에 대해 설명합니다.

문서 규약

활자체 또는 기호	의미	보기
`AaBbCc123`	명령어, 파일, 디렉토리의 이름; 화면 출력	`.login` 파일을 편집하십시오. 모든 파일을 나열하려면 `ls -a`를 사용하십시오. `% You have mail.`
`AaBbCc123`	화면 출력에 대해 사용자가 입력하는 내용	`% su` `Password:`
AaBbCc123	책 제목, 새 단어 및 용어, 강조하는 단어. 명령줄 변수를 실제 이름이나 변수로 대체.	*사용 설명서*의 6 장을 읽어 보십시오. 이러한 옵션을 *class* 옵션이라고 합니다. 파일을 삭제하려면 **`rm`** *파일이름*을 입력하십시오.

쉘 프롬프트

쉘	프롬프트
C 쉘	*시스템이름*%
C 쉘 수퍼유저	*시스템이름*#
Bourne 쉘 및 Korn 쉘	$
Bourne 쉘 및 Korn 쉘 수퍼유저	#

관련 문서

적용	제목	부품 번호
SunMC	*Sun Fire B1600용 SunMC 3.0 부록*	817-2539
릴리스 노트	*SNMP Release Notes for the Sun Fire B1600*	817-1006
Sun Fire B1600 플랫폼	*Sun Fire B1600 Blade System Chassis Hardware Installation Guide*	816-7614
	Sun Fire B1600 Blade System Chassis Software Setup Guide	816-3361
	Sun Fire B1600 Blade System Chassis Administration Guide	816-4765
	Sun Fire B1600 Blade System Chassis Switch Administration Guide	816-3365

온라인 Sun 문서 사용

다음 웹 사이트에서 번역된 버전을 포함하여 다양한 종류의 Sun 설명서를 보고 인쇄하고 구매할 수 있습니다.

`http://www.sun.com/documentation`

Sun 기술 지원 문의

이 문서에서 해답을 찾을 수 없는 제품에 관한 기술적 의문이 있는 경우, 다음 웹 사이트를 방문하십시오.

`http://www.sun.com/service/contacting`

고객의 의견

Sun은 설명서 개선을 위해 노력하고 있으며 고객의 의견과 제안을 환영합니다. 다음 웹 사이트에 방문하여 의견을 제출할 수 있습니다.

`http://www.sun.com/hwdocs/feedback`

피드백에 다음 문서의 제목과 부품 번호를 포함시키십시오.

Sun Fire B1600용 Sun SNMP 관리자 에이전트 안내서, 부품 번호 817-2502-10

I 부 기술적 설명과 기능

1 장

Sun SNMP 관리 에이전트 부록

이 Sun™ SNMP 관리 에이전트 릴리스는 Sun Fire™ B1600 선반 및 Sun Fire B100 블레이드의 모니터링 및 제어를 제공합니다.

플랫폼 유형에 따라서 다음을 채택할 수 있습니다.

- 도메인 에이전트, Sun Fire B100 블레이드에서 실행(도메인 하드웨어 모니터링)

  소프트웨어는 모니터되는 서버에 로컬로 설치되고 해당 서버만을 모니터할 수 있습니다. Sun Fire B1600의 경우, 각 블레이드가 개별적으로 모니터됩니다.

  Sun Fire B100 블레이드에 대한 도메인 하드웨어 모니터링의 범위는 블레이드만의 하드웨어로 제한됩니다. 서비스 표시기, PSU, SSC 및 선반 자체의 ID와 같은 다른 선반 구성요소는 포함되지 않습니다.

- 플랫폼 에이전트, 시스템 컨트롤러를 통해 프록시됨(플랫폼 하드웨어 모니터링)

  소프트웨어는 시스템 컨트롤러를 통해 플랫폼 계측에 액세스하는 원격(플랫폼 에이전트) 서버에 설치됩니다. 플랫폼 에이전트를 사용하면 시스템 컨트롤러가 관리하는 모든 하드웨어를 모니터할 수 있습니다.

  Sun Fire B1600에 대한 플랫폼 하드웨어 모니터링의 범위는 선반, 그의 ID, 서비스 표시기 및 그의 모든 FRU(현장 대체 가능 장치)를 포함합니다. 그 밖에 도메인 하드웨어 모니터링을 사용하여 사용 가능하지 않은 Sun Fire B100 블레이드에 대한 일부 하드웨어 정보(특히 전압 모니터링)를 사용할 수 있습니다.

  그림 1-1은 두 유형 모두의 하드웨어 모니터링 예를 보여줍니다. Sun Fire B1600 선반 A와 B가 플랫폼 에이전트 서버를 통해 NMS(네트워크 관리 스테이션)에 연결됩니다(플랫폼 하드웨어 모니터링). Sun Fire B1600 선반 C의 경우, Sun Fire B100 블레이드가 NMS(네트워크 관리 스테이션)에 직접 연결됩니다(도메인 하드웨어 모니터링).

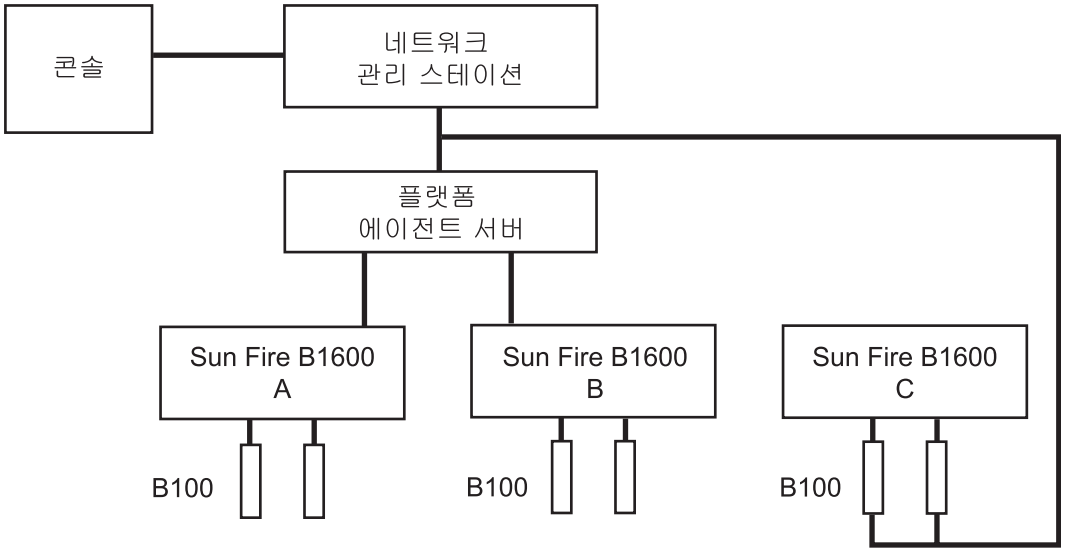

그림 1-1 도메인 및 플랫폼 하드웨어 모니터링의 예

이 소프트웨어는 다음 기능을 제공하는 많은 패키지로 구성됩니다.

- SNMP 서브에이전트

  기본적으로 SNMP 서브에이전트는 Solaris 마스터 에이전트인 snmpdx의 서브에이전트로서 등록됩니다. 서브에이전트를 *SNMP 조정자*라고도 합니다.

- SNMPv3 마스터 에이전트

  SNMPv3 마스터 에이전트는 플랫폼에 상주하는 SNMP 조정자가 액세스할 수 있는 하나의 보안된 존재점을 제공합니다. 마스터 에이전트는 `snmpdx`로 요청을 전달하여 프록시로서 작용합니다.

- Sun Fire B1600 및 Sun Fire B100 계측

  이들 패키지는 도메인 기반 또는 플랫폼 기반 모니터링이 채택되는지 여부에 따라서 필요할 때 설치됩니다.

2 장

SNMP 소개

이 장은 SNMP(단순 네트워크 관리 프로토콜)의 본질적인 기능에 대한 간략한 소개를 제공합니다. 포괄적인 내용이며 Sun Fire™ B1600 시스템과 특별히 관련되는 문제를 다룹니다.

이 장에는 다음 절이 들어있습니다.

SNMP 버전

SNMP는 네트워크 장치(시스템) 관리를 위한 개방형 인터넷 표준입니다. SNMP는 다른 인터넷 표준과 같이 IETF(Internet Engineering Task Force)가 발간한 많은 RFC(Requests for Comment)에 의해 정의됩니다.

승인된 표준을 정의하는 다음 세 가지 버전의 SNMP가 있습니다.

- SNMPv1
- SNMPv2(SNMPv2c라고도 하며, 이 문서에서 참조됨)
- SNMPv3

SNMPv1은 1988년에 처음 정의되었습니다. SNMPv2는 1993년에 도입되었으며 추가 프로토콜 작업과 데이터 유형을 추가하고 보안을 제공하여 SNMPv1의 일부 결점을 보완하려 했습니다. 보안 모델에서의 한계가 지금은 SNMPv2c 표준으로 승인되었으며, 이 표준은 새로운 보안 기반 기능을 제공했습니다. SNMPv2usec 및 SNMPv2*로 알려진 실험 버전도 현재 나오긴 했지만, 널리 채택되지 않았으며 아직 실험 단계에 있습니다.

1999년에 소개된 SNMPv3는 보안을 포함하여 플러그 가능한 구성요소를 지원하는 SNMP 관리 프레임워크를 정의합니다.

이들 표준에 대한 추가 정보는 IETF 웹 사이트(`http://www.ietf.org/rfc.html`)에 있는 다음 RFC를 참조하십시오.

- SNMPv1: RFC1155, RFC1157, RFC1212, RFC1215
- SNMPv2: RFC2578, RFC2579, RFC2580, RFC3416
- SNMPv3: RFC3410, RFC3411, RFC3412, RFC3413, RFC3414, RFC3415
- 표준 사이의 공존성: RFC2576

SNMP 관리자와 에이전트

SNMP는 일반적으로 *관리자*라고 부르는 NMS(네트워크 관리 스테이션)가 장치를 원격으로 관리할 수 있게 하는 네트워크 프로토콜입니다.

장치가 관리되려면 그와 연관된 SNMP *에이전트*(SNMP 조정자라고 함)를 가져야 합니다. 조정자의 목적은 다음을 수행하는 것입니다.

- 관리자로부터 장치의 상태를 표시하는 데이터에 대한 요청 수신 및 적당한 응답 제공
- 장치 상태의 제어가 가능하도록 관리자의 데이터 승인
- 장치와 관련된 중요한 이벤트를 신호하기 위해 하나 이상의 선택된 관리자에게 전송되는 요구받지 않은 메시지인 SNMP *트랩* 생성

SNMP Management Information Base

장치를 관리하고 모니터하기 위해서는 장치의 특성이 에이전트와 관리자 모두에게 알려진 형식을 사용하여 표현되어야 합니다. 이들 특성은 팬 속도 같은 물리적 특성이나 라우팅 테이블 같은 서비스를 표현할 수 있습니다. 이들 특성을 정의하는 데이터 구조를 MIB*(Management Information Base)*라고 합니다. 이 데이터 모델은 보통 테이블 형식으로 구성되지만 간단한 값을 포함할 수도 있습니다. 전자의 예가 라우팅 테이블이며, 후자의 예는 에이전트가 시작된 시간을 표시하는 시간소인입니다.

MIB는 SNMP를 통해 액세스할 수 있는 가상 데이터 스토어에 대한 정의입니다. 다음과 같이 관리자에서 `get` 및 `set` 작업을 사용하여 MIB 내용에 액세스할 수 있습니다.

- `get` 작업에 대한 응답에서, 조정자는 로컬로 유지되거나 아니면 관리되는 장치에서 직접 오는 데이터를 제공합니다.
- `set` 작업에 대한 응답에서, 에이전트가 보통 에이전트 자신 또는 관리되는 장치의 상태에 영향을 주는 어떤 조치를 수행합니다.

NMS가 그의 에이전트를 통해 장치를 관리할 수 있으려면, 에이전트가 제공하는 데이터에 대응하는 MIB가 관리자에 로드되어야 합니다. 이 작업을 수행하기 위한 메커니즘은 네트워크 관리 소프트웨어의 구현에 따라 다릅니다. 이 작업은 관리자에게 에이전트가 제공하는 데이터 모델을 다루고 올바르게 해석하기 위해 필요한 정보를 제공합니다.

참고 – MIB가 다른 MIB의 정의를 참조할 수 있으므로, 주어진 MIB를 사용하려면 다른 MIB를 로드해야 할 수 있습니다.

이 가상 데이터 스토어의 내용을 주소지정하기 위해, MIB는 OID*(Object Identifier)*의 형식으로 정의됩니다. OID는 고유한 이름 공간을 정의하는 정수의 계층 구조로 배열된 순서로 구성됩니다. 각 지정된 정수가 연관된 텍스트 이름을 갖습니다. 예를 들어, OID `1.3.6.1`은 OID 이름 `iso.org.dod.internet`에 해당하며 `1.3.6.1.4`는 OID 이름 `iso.org.dod.internet.private`에 해당합니다.

숫자 양식은 SNMP 프로토콜 트랜잭션 내에서 사용되는 반면, 텍스트 양식은 사용자 인터페이스에서 이해를 돕기 위해 사용됩니다. 이런 OID로 표시되는 개체는 일반적으로 축약형 양식으로서 이름의 마지막 구성요소에 의해 참조됩니다. 이 관례에서 발생하는 혼동을 피하기 위해, 그 안에 정의되는 모든 개체 이름에 *sunPlat* 같은 MIB에 고유한 접두어를 적용하여 그러한 모든 ID를 전역적으로 고유하게 하는 것이 표준입니다.

참고 – MIB는 ASN.1이라고 부르는 언어를 사용하여 정의되는데, 이에 대한 논의는 이 문서의 범위를 벗어납니다. 참고를 위해, SNMPv2c에 대한 MIB의 구조는 RFC2578에 정의된 SMI(Structure of Management Information)에 의해 정의됩니다. 이것은 MIB에 사용 가능한 구문 및 기본 데이터 유형을 정의합니다. RFC2579에 정의된 Textual Conventions(유형 정의)이 추가 데이터 유형 및 열거를 정의합니다.

MIB 테이블

MIB에 의해 정의되는 많은 데이터 내용은 표 형식으로 되어 있고 각각이 고유한 OID를 갖는 일련의 개체로 구성되는 항목으로 이루어집니다. 예를 들어, 팬 특성 표는 팬당 하나의 많은 행으로 구성되며, 각 행에는 현재 속도, 예상 속도 및 최소 허용 가능한 속도에 대응하는 열이 들어있습니다.

테이블 안에서 행의 주소지정은 다음과 같을 수 있습니다.

- 단순한 1차원 색인(테이블 안에서의 행 번호, 예를 들면 '6')
- IP 주소와 포트 번호 같은 더 복잡한 다차원 인스턴스 지정자 (예를 들면, `27.0.0.1, 1234`)

MIB 내의 각 테이블 정의에는 주어진 항목을 선택하는 데 사용할 인스턴스 지정자를 정의하는 INDEX 절이 있습니다. 어느 경우에나 필수 행에 대한 색인을 정의하는 데 사용되는 개체는 MIB 안에서 스스로 정의되어야 합니다. 따라서 단순한 1차원 색인은 보통 테이블의 INDEX 절이 참조하는 색인 열을 갖습니다. 그러면 테이블의 특정 데이터 항목은 그의 열 접두어를 제공하는 OID를 지정하여 주소지정됩니다.

예를 들어, 접미어 인스턴스 지정자(예: 앞의 예에 있는 `127.0.0.1.1234`)를 갖는 `myFanTable.myFanEntry.myCurrentFanSpeed`는 `myFanTable.myFanEntry.myCurrentFanSpeed.127.0.0.1.1234`가 됩니다.

MIB 구문을 정의하는 SMI는 효과적으로 테이블에 여분의 열을 추가하여 추가 항목을 추가하는 테이블 확장을 위한 중요한 기능을 제공합니다. 이것은 확장될 테이블 색인 절의 중복된 색인 절을 포함한 테이블을 정의하여 이루어집니다.

또한 그 안에 들어있는 개체가 아니라 잠재적으로 다른 MIB에 정의되는 다른 테이블에서 *반입된* 개체에 의해 색인화되는 MIB 테이블을 정의할 수 있습니다. 이 구성은 사실상 열이 기존 테이블에 추가될 수 있게 합니다.

참고 – SUN-PLATFORM-MIB는 이 메커니즘을 널리 사용하여 ENTITY-MIB(5 장 참조)로 정의되는 테이블을 확장합니다.

액세스 제어

MIB에 정의되는 모든 주소 지정 가능한 개체는 연관된 최대 액세스 권리(예: *read–only* 또는 *read–write*)를 갖습니다. 이들 권한이 에이전트가 지원할 수 있는 최대 액세스를 판별하며, 관리자가 오퍼레이터가 시도하도록 허용할 작업을 제한하는 데 사용할 수 있습니다. 에이전트는 필요한 경우 더 낮은 액세스 권리를 적용할 수 있습니다. 즉, 읽기–쓰기로 간주되는 개체에 대한 쓰기를 거부할 수 있습니다. 이 거부는 다음을 기초로 할 수 있습니다.

- 작업이 주소 지정될 개체에 적용 가능한 방법(예를 들어, MIB로 정의되는 개체가 특정 트랜잭션만이 합법적인 상태 시스템을 표시하는 경우)
- 특정 작업을 제한된 관리자 세트로 제한하는 보안 제한사항

SMMPv1에서 보안 액세스 권리를 통신하는 데 사용되는 메커니즘은 *커뮤니티 문자열*입니다. 커뮤니티 문자열은 단순히 각 SNMP 데이터 요청과 함께 전달되는 *private* 및 *public* 같은 텍스트 문자열입니다. SNMPv1 및 SNMPv2 요청이 암호화되지 않기 때문에 이것은 안전한 것으로 간주해서는 안됩니다. 에이전트가 응답할 커뮤니티 문자열과 발신 관리자를 정의하는 데 사용되는 메커니즘은 에이전트의 구현에 달려있지만, 보통 적용 가능한 액세스 권한을 기술하는 파일인 액세스 제어 목록(ACL)을 바탕으로 합니다.

ACL 구성 방법의 설명에 대해서는 11 장을 참조하십시오.

SNMP 마스터 에이전트

관리자는 에이전트가 실행 중인 시스템의 잘 알려진 포트(161)로 (UDP) 패킷을 전송하여 에이전트와 통신합니다. 주어진 시스템에서 각기 서로 다른 장치를 관리하는 여러 에이전트가 실행 중이면 포트 자원 사용에 대한 충돌이 있을 수 있습니다. 한 가지 가능한 해결책은 각 에이전트에 대해 상이한 비표준 포트 번호를 사용하는 것입니다. 또 다른 방법은 주어진 시스템에서 실행 중인 모든 에이전트를 대신하여 SNMP 요청을 승인하고 이들 요청을 적절하게 전송하는 *마스터 에이전트*의 개념을 도입하는 것입니다. 이것은 관리자가 일관된 방식으로 모든 SNMP 에이전트에 액세스할 수 있게 하는 장점이 있습니다. Sun Fire B1600은 두 접근 방식을 모두 지원합니다.

마스터 에이전트에 대한 자세한 정보는 3 장을 참조하십시오.

SNMP 조정자 및 snmpdx

`snmpdx`는 표준 Solaris™ SNMP 에이전트이며 SNMPv1 마스터 에이전트입니다.

기본적으로 SNMP 조정자는 `snmpdx`의 서브에이전트로 등록됩니다. 이 구성에서는, SNMPv2c 통지가 발행됨에도 불구하고 SNMPv1 `get` 및 `set` 요청만이 지원됩니다.

`snmpdx`와 SNMP 조정자 사이의 관계는 10 장 및 11 장에서 자세히 설명됩니다.

매뉴얼 페이지 `snmpdx`(1M)도 참조하십시오.

3 장

마스터 에이전트

이 장에서는 SNMPv3 마스터 에이전트의 기능과 특징에 대해 설명합니다.

이 장에는 다음 절이 들어있습니다.

기능

SNMPv3 마스터 에이전트는 SNMP 관리 정보에 액세스할 수 있는 보안된 존재점을 제공합니다.

SNMPv3 마스터 에이전트는 SNMP 서비스 포트(기본값 161)에 바인드하고 모든 요청을 `snmpdx`에 전달하는데, 이것은 표준 Solaris 마스터 에이전트로서 이들 요청을 다시 적절한 등록된 서브에이전트에 전달합니다. `snmpdx`는 표준 Solaris 분배의 일부로 제공되지만, SNMPv1만을 지원하므로 SNMPv3이 제공하는 보안을 직접 제공하지 않습니다. 마스터 에이전트가 SNMPv1, v2c 또는 v3일 수 있는 모든 요청을 SNMPv1으로 변환하므로, `snmpdx`가 이들을 처리할 수 있어야 합니다.

사실상, 마스터 에이전트는 기존의 모든 서브에이전트에 대한 보안 액세스를 제공하여 SNMP 방화벽으로 작용합니다.

구성 개요

이 절은 마스터 에이전트 기능을 포함하도록 SNMP를 구성하는 방법을 소개합니다. 이 주제는 11 장에서 자세히 다루어지는데, 이 절의 나머지에서 참조되는 구성 파일의 모든 설명과 예가 포함됩니다.

SNMP 조정자는 자동으로 할당되는 포트 번호를 사용하여 snmpdx의 서브에이전트로 등록됩니다.

참고 – 이 안내서에서는 SNMP 에이전트를 SNMP 조정자로 부릅니다.

마스터 에이전트가 사용 가능할 때, `snmpdx` 자동 시작이 사용 불가능하며 마스터 에이전트가 포트 161에 등록됩니다. 새 포트 번호가 `snmpdx`에 지정되는데, 이것은 마스터 에이전트 시작 파일의 제어하에 시작됩니다.

구성은 다음을 사용하여 수행됩니다.

- SNMPv3 보안 파일 `spama.uacl` 및 `spama.security`
- SNMPv1/v2 액세스 제어 목록(ACL) 파일 `spama.acl`
- 구성 파일 `spama.conf`
- 시작 스크립트, `spama`

마스터 에이전트를 실행하는 중에 동적으로 구성하는 것은 불가능합니다.

SNMP 조정자도 구성이 필요하며 11 장에 설명되어 있습니다.

4 장

플랫폼 관리 모델

이 장에서는 SNMP가 Sun 플랫폼 SNMP 모델(sunPlat)을 사용하여 Sun Fire B1600 시스템을 모델화하는 방법의 개요를 설명합니다.

이 장에는 다음 절이 들어있습니다.

Sun Fire B1600 플랫폼 모델링

Sun Fire B1600은 섀시 안에 내포된 *하드웨어 자원*의 콜렉션으로 표현됩니다. 마더보드와 같은 일부 자원은 섀시 안에 직접 내포될 수 있습니다. 다른 것들은 다른 자원 안에 내포됩니다. 예를 들어, 마더보드가 프로세서를 포함할 수 있습니다. 이러한 관계는 섀시 안에서부터 확장하여 각각이 격납하는 *상위* 안에 물리적으로 포함되는 하드웨어 자원의 *계층 구조*를 형성합니다. 이 계층 구조는 하드웨어 자원을 표현하는 *관리 개체* 사이의 *관계*를 사용하여 모델화됩니다.

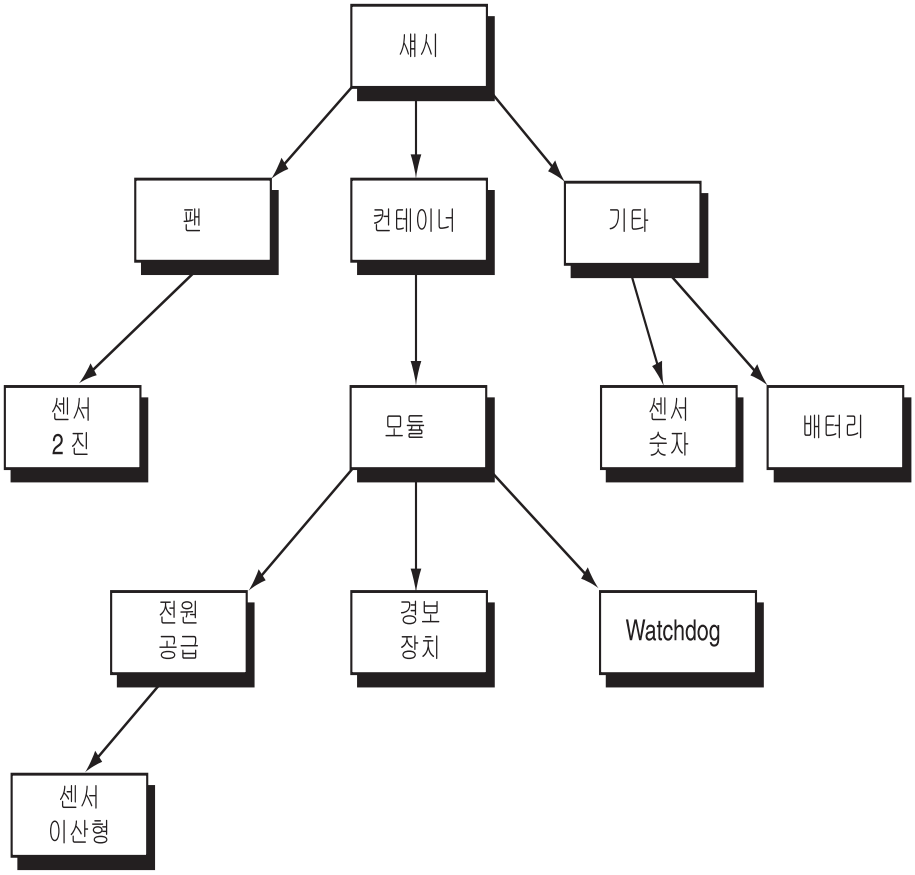

그림 4-1 하드웨어 자원 계층 구조 예

관리 개체

sunPlat 모델은 기본 하드웨어 자원을 표현하는 공통된 플랫폼 구성 단위의 유용한 세트를 제공합니다. 이들 플랫폼 구성 단위의 인스턴스를 *관리 개체*라고 부릅니다. 하드웨어 자원은 모니터할 수 있거나 유용한 구성 정보를 제공하는 경우 관리 개체로 표시됩니다.

추가 관리 개체가 관리 인터페이스의 다른 기능을 표시하는 데 사용됩니다. 예를 들어, 하드웨어 자원은 문제점(*경보*) 또는 구성 변경(*이벤트*)에 대한 응답으로 비동기 상태 보고서(*통지*)를 발행할 수 있습니다.

관리 개체는 관리 개체 *클래스*를 사용하여 정의됩니다. 자원의 특성은 관리 개체의 *등록정보*로 표시됩니다. 기존 클래스를 사용하여 *서브클래스*라고 부르는 새 클래스가 정의됩니다. 서브클래스는 그의 *수퍼클래스*의 모든 특성을 *상속*하지만, 새 등록정보를 추가하여 고유한 특성을 표시합니다.

그림 4-2는 sunPlat 모델에 의해 정의되는 하드웨어 구성 단위의 클래스 상속 계층 구조를 보여줍니다.

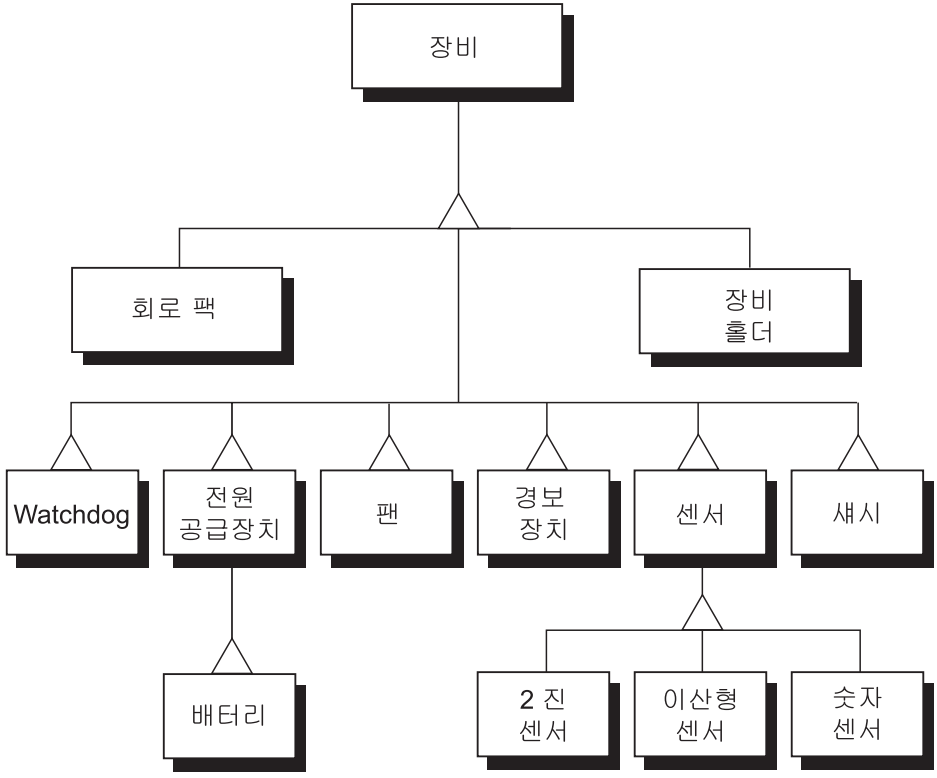

그림 4-2 sunPlat 관리 개체 클래스 상속 도표

sunPlat 클래스의 변종

sunPlat 클래스는 산업 표준의 관리 개념을 기반으로 합니다. Sun Fire B1600 시스템은 하드웨어 기반구조의 표시를 위해 선택된 ITU-T Generic Network Information Model의 서브세트를 사용합니다. 이것은 TMN(Telecommunications Management Network)에서 균일한 결함 및 구성 관리를 지원하는 강력하고 확장성있는 프레임워크를 제공합니다.

DMTF(Distributed Management Task Force) CIM(Common Information Model) Schema는 물리적 환경과 이벤트 정의 및 처리를 모델화하고, 공통 모델에 대한 시스템 고유 확장을 제공합니다.

5 장

Sun Fire B1600 MIB

이 장에서는 SNMP 인터페이스가 Sun Fire B1600 관리 개체 및 그들의 관계를 제공하는 방법에 대해 설명합니다.

이 장에는 다음 절이 들어있습니다.

모델의 SNMP 표시

SNMP 조정자는 검사 및 이벤트 기반 관리를 모두 지원합니다. Sun Fire B1600 시스템의 물리적 구성요소와 그 안에 있는 관리 도메인의 논리 표시는 RFC 2737에 의해 정의되고 SUN-PLATFORM-MIB로 확장되는 ENTITY-MIB에 의해 제공됩니다.

참고 – MIB에 정의되는 많은 개체는 `read-write`의 `MAX-ACCESS`를 갖지만, 이들 개체는 작업이 모델화되는 구성요소에 적합한 경우에만 쓰기 가능합니다.

ENTITY-MIB에는 다음 그룹이 들어있는데, 이것은 관리되는 시스템의 물리적 및 논리적 요소를 설명합니다.

entityPhysical 그룹

`entityPhysical` 그룹은 에이전트가 관리하는 식별 가능한 물리적 자원(예: 섀시, 전원 공급장치, 센서 등)인 물리적 엔티티를 기술합니다. 이들 엔티티는 *entPhysicalTable*에서 행으로 표시됩니다.

entityLogical 그룹

`entityLogical` 그룹은 에이전트가 관리하는 논리적 엔티티를 기술합니다. 이들은 더 높은 수준의 관리가 관리해야 하는 서비스의 추상적 개념을 제공하는 *고가의* 논리적 엔티티의 표시입니다. 이들은 주로 플랫폼 하드웨어 관리와 관련되며 OS 재부트, 하드웨어 재설정 및 전원 제어 같은 기능을 포함합니다. 일반적으로 Solaris 도메인이나 서비스 컨트롤러 같은 관리 도메인에 해당합니다.

entityMapping 그룹

`entityMapping` 그룹은 `entityPhysical` 그룹과 `entityLogical` 그룹간의 관계를 식별합니다. 이 기능은 내부적으로 SNMP 조정자에 의해 처리됩니다.

entityGeneral 그룹

`entityGeneral` 그룹은 물리적 엔티티 테이블 또는 물리적 맵핑 테이블의 임의의 엔티티가 변경되는 시간에 대한 마지막 기회 시간 소인을 제공합니다.

entityMIBTraps 그룹

`entityMIBTraps` 그룹은 ENTITY-MIB의 임의의 개체에 대한 변경을 신호하는 데 사용되는 *entPhysicalChange* 통지를 정의합니다.

2 장에 SNMP의 일반 요소가 Sun Platform SNMP 모델을 표시하는 방법의 개요가 제공되어 있습니다.

물리적 모델

sunPlat 물리적 모델은 ENTITY-MIB를 사용하여 하드웨어 엔티티의 포함 계층 구조를 제공합니다. 각 엔티티는 ENTITY-MIB의 entPhysicalTable의 개별 행으로 모델화됩니다.

그림 5-1 은 물리적 포함 계층 구조의 예를 보여줍니다 . 우측 하단 모서리에 있는 숫자는 *entPhysicalTable* (표 5-1 참조) 의 대응하는 행에 대한 색인을 제공합니다 .

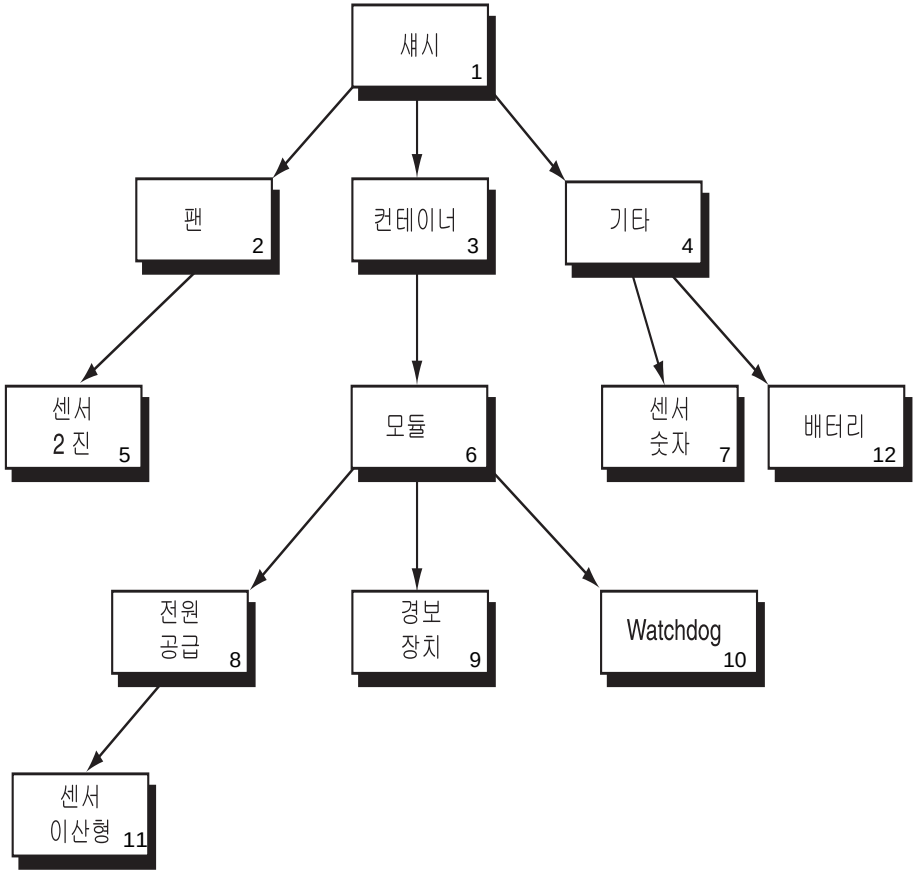

그림 5-1 하드웨어 자원 계층 구조 예

이 정보는 다음 SNMP 테이블을 사용하여 표시됩니다.

- 물리적 엔티티 테이블(*entPhysicalTable*)

  이 테이블은 하드웨어 엔티티당 한 행을 제공합니다. 이들 행을 *항목*이라고 부르며 특정 행을 *인스턴스*로 참조합니다. 각 항목은 다음을 포함합니다.

  - 물리적 클래스(*entPhysicalClass*)
  - 하드웨어 엔티티의 공통 특성
  - 고유한 색인(*entPhysicalIndex*)
  - 이 자원에 대한 *컨테이너*로서 작용하는 하드웨어 엔티티의 행을 가리키는 참조(*entPhysicalContainedIn*). 다른 컨테이너 안에 물리적으로 포함되지 않는 구성요소(예: 새시)의 경우 이것은 0입니다.

- 물리적 맵핑 테이블(*entPhysicalContainsTable*)

  이 테이블은 물리적 엔티티 테이블에 표시되는 하드웨어 자원의 계층 구조의 가상 사본을 제공합니다. 이 테이블은 2차원으로, 먼저 포함하는 항목의 *entPhysicalIndex*에 의해 색인화되고, 두 번째로 각 포함된 항목의 *entPhysicalIndex*에 의해 색인화됩니다.

표 5-1은 위의 수치가 바탕으로 하는 *entPhysicalTable*을 보여주며, 표 5-2는 물리적 맵핑을 표시합니다.

표 5-1 물리적 엔티티 테이블

entPhysicalIndex	entPhysicalClass	entPhysicalContainedIn
1	새시	0
2	팬	1
3	컨테이너(예를 들면, FRU를 포함하는 슬롯)	1
4	기타	1
5	센서(2진)	2
6	모듈(예: 플러그 가능한 FRU)	3
7	센서(숫자)	4
8	전원 공급장치	6
9	경보 장치	6
10	watchdog	6
11	센서(이산형)	8
12	전원 공급장치(배터리)	4

표 5-2 물리적 맵핑 테이블

entPhysicalIndex	entPhysicalChildIndex
1	2
1	3
1	4
2	5
3	6
4	7
4	12
6	8
6	9
6	10
8	11

클래스

`entPhysicalClass`는 특정 물리적 엔티티의 일반 하드웨어 유형의 표시를 제공하는 열거된 값으로, 각각 *entPhysicalTable*에 있는 한 행으로 표시됩니다.

다음 열거는 Sun Fire B1600 플랫폼에 적용됩니다(그림 5-1 및 표 5-1을 참조하십시오).

- `other(1)`

열거 `other`는 물리적 엔티티가 다음 중 하나로 분류될 수 없는 경우에 적용됩니다.

- `chassis(3)`

`chassis` 클래스는 장비에 대한 전체 컨테이너를 표시합니다. 물리적 엔티티의 모든 클래스를 섀시 안에서 얻을 수 있습니다.

- `container(5)`

`container` 클래스는 동일한 또는 상이한 유형의 하나 이상의 제거 가능한 물리적 엔티티를 포함할 수 있는 물리적 엔티티에 적용됩니다. 예를 들어, 섀시의 비었거나 가득 찬 각 슬롯이 컨테이너로 모델화됩니다. 전원 공급장치나 팬 같은 FRU(현장 대체 가능 장치)는 컨테이너 엔티티 내의 모듈로 모델화됩니다.

- `powerSupply(6)`

`power supply` 클래스는 전원을 공급할 수 있는 장비에 적용됩니다.

- `fan(7)`

`fan` 클래스는 물리적 엔티티가 팬이나 기타 냉각 장치인 경우에 적용됩니다.

■ `sensor(8)`

`sensor` 클래스는 어떤 물리적 등록정보를 측정할 수 있는 물리적 엔티티에 적용됩니다.

■ `module(9)`

`module` 클래스는 자체 포함 서브시스템에 적용되며, `chassis` 또는 다른 `module` 같은 다른 물리적 엔티티 내에 모델화됩니다. 이 엔티티는 항상 `container` 내에 모델화됩니다.

논리적 모델

sunPlat 논리적 모델은 ENTITY-MIB를 사용하여 고가의 논리적 엔티티를 제공합니다. 각 엔티티는 ENTITY-MIB의 *entLogicalTable*에서 개별 행으로 표시됩니다. 물리적 모델과는 달리 논리적 모델은 계층 구조가 아니라 평면 구조임을 유의하십시오.

ENTITY-MIB는 물리적 개체의 경우와는 달리 논리적 개체의 여러 클래스를 구별하지 않습니다. SUN-PLATFORM-MIB가 논리적 개체에 대한 클래스 계층을 제공하며 7 장에서 설명합니다.

*entLogicalTable*의 정보를 사용하여 상이한 이름 지정 문맥을 사용하는 다중 범위 지정을 지원할 수 있습니다. 그러나 이 기능은 본 제품에서 채택되지 않습니다. 특정 값의 정보는 *entLogicalDescription* 및 *entLogicalTAddress*로서, 후자는 논리적 엔티티를 액세스할 수 있는 IP 주소를 제공합니다.

논리적 및 물리적 계층 구조 맵핑

ENTITY-MIB는 논리적 개체와 논리적 개체가 구성하는 물리적 개체 사이의 맵핑을 제공합니다. 이 맵핑은 *entLPMappingTable*에 의해 달성되는데, 이 테이블은 2차원 테이블(*entPhysicalContainsTable*와 유사함)이며 주어진 논리적 엔티티를 실현하는 물리적 엔티티를 식별합니다. 이들 물리적 엔티티는 *entLPPhysicalIndex*로 식별되는데, 이것은 *entPhysicalIndex*와 동등합니다.

이 테이블이 잠재적으로 주어진 논리적 엔티티와 연관된 모든 물리적 엔티티를 표시할 수 있지만, 관례에 따라 격납하는 물리적 엔티티만이 참조됩니다. 예를 들어, 물리적 모듈에 의해 실현되는 논리적 엔티티의 경우 맵핑은 그 안에 들어있는 모든 물리적 엔티티가 아니라 모듈만을 참조합니다.

이벤트 및 경보 모델

ENTITY-MIB는 하나의 SNMP 통지, *entConfigChange*를 제공하는데, MIB에 있는 임의의 테이블에 대한 변경을 신호하는 데 사용됩니다. 5초마다 최대 하나의 트랩을 제공하도록 설정됩니다.

SUN-PLATFORM-MIB는 보다 특정한 통지를 정의하며 8 장에서 설명합니다.

SUN-PLATFORM-MIB

SUN-PLATFORM-MIB는 다음을 수행합니다.

- 구성요소의 새 클래스를 표시하기 위해 물리적 엔티티 테이블을 확장합니다.
- 고가의 플랫폼 및 서버 개체를 표시하기 위해 논리적 엔티티 테이블을 확장합니다.

참고 – SUN-PLATFORM-MIB의 모든 개체는 글로벌하게 고유하게 만들기 위해 접두어 *sunPlat*을 갖습니다.

물리적 모델 테이블 확장

SUN-PLATFORM-MIB는 물리적 엔티티 테이블에서 표시되지 않는 클래스의 추가 속성을 제공합니다. 아래의 드문드문하게 점유되는 테이블 확장을 추가하여 물리적 엔티티 테이블을 확장합니다.

- 장비 테이블 확장

  이것은 Equipment 클래스의 관리 개체에 대한 추가 정보를 제공하기 위해 물리적 엔티티 테이블을 증대시킵니다. 이 클래스는 모든 Sun Fire B1600 하드웨어 자원에 적용 가능합니다. Equipment 클래스의 서브클래스는 추가 테이블 확장에 의해 표시됩니다.

  - 장비 홀더 테이블 확장

    이것은 장비 테이블 확장을 확장합니다. `container(5)` entPhysicalClass의 관리 개체와 관련된 추가 정보를 제공합니다.

  - 회로 팩 테이블 확장

    이것은 장비 테이블 확장을 확장합니다. `module(9)` entPhysical 클래스의 관리 개체와 관련된 추가 정보를 제공합니다.

- 물리적 테이블 확장

  이것은 물리적 엔티티 테이블을 확장합니다. 물리적 엔티티 테이블의 entPhysicalClass 열을 보충하는 데 사용됩니다. 자원이 `other(1)`의 entPhysicalClass를 갖지만 sunPlat에 의해 모델화되는 클래스(즉, Watchdog 또는 AlarmDevice 클래스)인 경우, 이 테이블이 그의 sunPlatPhysicalClass를 식별합니다.

- 센서 테이블 확장

  이것은 장비 테이블 확장을 확장합니다. entPhysicalClass `sensor(8)` 관리 개체와 관련된 추가 정보를 제공합니다. Sensor 클래스의 서브클래스는 추가 테이블 확장에 의해 표시되며 sunPlatSensorClass를 사용하는 이 테이블에 의해 식별됩니다.

  - 2진 센서 테이블 확장

    이것은 센서 테이블 확장을 확장합니다. entPhysicalClass `sensor(8)` 및 sunPlatSensorClass `binary(1)`의 관리 개체와 관련된 추가 정보를 제공합니다.

  - 숫자 센서 테이블 확장

    이것은 센서 테이블 확장을 확장합니다. entPhysicalClass `sensor(8)` 및 sunPlatSensorClass `numeric(2)`의 관리 개체와 관련된 추가 정보를 제공합니다.

  - 이산형 센서 테이블 확장

    이것은 센서 테이블 확장을 확장합니다. entPhysicalClass `sensor(8)` 및 sunPlatSensorClass `discrete(3)`의 관리 개체와 관련된 추가 정보를 제공합니다.

- 팬 테이블 확장

  이것은 장비 테이블 확장을 확장합니다. entPhysicalClass `fan(7)` 관리 개체와 관련된 추가 정보를 제공합니다.

- 경보 테이블 확장

  이것은 장비 테이블 확장을 확장합니다. entPhysicalClass `other(1)` 및 sunPlatPhysicalClass `alarm(8)`의 관리 개체와 관련된 추가 정보를 제공합니다.

- Watchdog 테이블 확장

  이것은 장비 테이블 확장을 확장합니다. entPhysicalClass `other(1)` 및 보통 서비스 표시기 LED를 표시하는 sunPlatPhysicalClass `watchdog(3)`의 관리 개체와 관련된 추가 정보를 제공합니다.

- 전원 공급장치 테이블 확장

  이것은 장비 테이블 확장을 확장합니다. entPhysicalClass `powerSupply(6)`의 관리 개체와 관련된 추가 정보를 제공합니다.

표 5-3은 물리적 엔티티 테이블에 대한 테이블 확장의 예를 보여줍니다. entPhysicalIndex(이 테이블의 열 1)는 그림 5-1에 표시된 하드웨어 자원 계층 구조 예를 바탕으로 합니다.

ENTITY-MIB			SUN-PLATFORM-MIB											
entPhysicalIndex	entPhysicalClass				sunPlatPhysicalClass		sunPlatFanClass		sunPlatSensorClass					sunPlatPowerSupplyClass
1	새시													
3	컨테이너													
8	전원 공급 장치													G
12	전원 공급 장치													H
2	팬						A							
	팬						B							
	팬						C							
5	센서								D					
7	센서								E					
11	센서								F					
6	모듈													
9	기타				경보									
10	기타				watch dog									
4	기타				기타									
entPhysicalTable		sunPlatEquipmentTable	sunPlatEquipmentHolderTable	sunPlatCircuitPackTable	sunPlatPhysicalTable	sunPlatWatchdogTable	sunPlatFanTable	sunPlatAlarmTable	sunPlatSensorTable	sunPlatBinarySensorTable	sunPlatNumericSensorTable	sunPlatDiscreteSensorTable	sunPlatDiscreteSensorStatusTable	sunPlatPowerSupplyTable

표 5-3 물리적 엔티티 테이블 확장

표 5-4 물리적 엔티티 테이블 확장에 대한 키(표 5-3)

참조	설명
A	팬
B	냉각
C	탈열기
D	2진
E	숫자
F	이산형
G	전원 공급 장치
H	배터리

논리적 모델 테이블 확장

SUN-PLATFORM-MIB는 논리적 엔티티 테이블에서 지원되지 않는 클래스의 추가 속성을 제공합니다. 아래의 드물게 점유되는 테이블 확장을 추가하여 논리적 엔티티 테이블을 확장합니다.

- 논리적 클래스 확장 테이블

이것은 entLogicalTable을 확장하여 논리적 엔티티의 클래스, *SunPlatLogicalClass* 및 그의 상태 *sunPlatLogicalStatus*를 정의합니다. sunPlatLogicalTable은 entLogicalTable의 모든 항목에 대해 유효합니다. Logical 클래스의 Computer System 서브클래스가 추가 테이블 확장에 의해 표시됩니다.

- 컴퓨터 시스템 테이블 확장

  이 테이블은 컴퓨터 시스템의 인스턴스에 공통인 속성을 제공하기 위해 entLogicalTable을 확장합니다. sunPlatUnitaryComputerSystemTable은 `computerSystem(2)`의 sunPlatLogicalClass를 갖는 entLogicalTable의 해당 행에 대해 유효합니다.

  로드 정보 테이블(*sunPlatInitialLoadInfoTable*)의 항목 세트가 각 컴퓨터 시스템 논리적 엔티티와 연관됩니다. 이 세트가 컴퓨터 시스템의 부트 설정을 제어하는 데 사용되는 매개변수를 구성합니다.

이벤트 및 경보 로그 테이블

SNMP 트랩은 전달될 것으로 보증되지 않습니다. 이 관점에서 플랫폼 경보의 현재 상태를 정확하게 추적하는 관리 응용프로그램 능력을 지원하기 위해 MIB는 각 관리 개체에 대한 현재 문제점 목록을 유지합니다. 이것은 각 개체에 대한 미해결 경보의 테이블입니다. 이들은 경보 조건이 지워질 때 자동으로 지워집니다.

SUN-PLATFORM-MIB는 이벤트나 경보 유형별로 또는 영향을 받는 엔티티별로 그룹화되는 이벤트 또는 경보를 기록하는 데 사용할 수 있는 로그를 정의합니다. Sun Fire B1600을 위한 구현은 이들 로그를 채택하여 앞의 문단에서 명시된 이유 때문에 모니터되는 모든 엔티티에 대한 미해결 경보의 목록을 유지합니다.

경보가 생성될 수 있는 MIB의 각 엔티티(즉, 모든 물리적 및 논리적 엔티티)는 로그 테이블(sunPlatLogTable)에 항목을 갖습니다. 이 테이블은 현재 문제점 목록에 대한 관리 상태와 제어를 제공합니다. 로그는 영구적으로 사용 가능하며 크기에 제한이 없습니다.

로그 테이블의 각 항목에 대응하는 0개 이상의 항목이 로그 레코드 테이블에 존재할 수 있습니다. 이들 항목은 8 장에서 자세히 설명됩니다.

이벤트 레코드

이들 레코드가 sunPlat 트랩 통지의 일부를 형성합니다. 모델에서의 변경사항은 SNMP 트랩의 두 범주, 이벤트 및 경보를 사용하여 관리 응용프로그램에 알려집니다.

이벤트

- 개체 작성 레코드

  이 레코드는 자원이 개체 모델에 추가되었음을 표시합니다.
- 개체 삭제 레코드

  이 레코드는 자원이 개체 모델에서 제거되었음을 표시합니다.
- 상태 변경 레코드

  이 레코드는 자원의 상태가 변경되었음을 표시합니다.
- 정수 속성 값 변경 레코드

  이 레코드는 유형 `INTEGER`의 속성에 의해 모델화되는 자원의 특성 변화를 표시합니다. 정수는 영향을 받는 개체에 따라서 부호있는 또는 부호없는 정수일 수 있습니다.
- 문자열 속성 값 변경 레코드

  이 레코드는 유형 `OCTET STRING`의 속성에 의해 모델화되는 자원의 특성 변화를 표시합니다.
- OID 속성 값 변경 레코드

  이 레코드는 유형 `OBJECT IDENTIFIER`의 개체 ID 속성의 변경을 표시합니다.

경보

- 통신 경보 레코드

  이 레코드는 자원이 지원하는 통신 서비스에서 장애가 발생했음을 표시합니다.

- 환경 경보 레코드

  이 레코드는 자원과 관련된 환경 조건을 표시합니다.
- 장비 경보 레코드

  이 레코드는 자원이 고장났음을 표시합니다.
- 처리 오류 경보 레코드

  이 레코드는 자원에 연관된 소프트웨어 또는 처리 결함이 있음을 표시합니다.
- 서비스 품질 경보 레코드

  이 레코드는 서비스 품질 경보가 발생했음을 표시합니다.
- 불확정 경보 레코드

  이 레코드는 알 수 없는 유형의 경보가 발생했음을 표시합니다.

6 장

물리적 모델

이 장에서는 sunPlat 물리적 클래스 계층과 sunPlat 모델에서 정의되는 관리 물리적 개체 클래스가 SUN-PLATFORM-MIB에 의해 표시되는 방법에 대해 설명합니다.

이 장에는 다음 절이 들어있습니다.

sunPlat 물리적 클래스 계층

그림 6-1은 Sun Fire B1600 내의 하드웨어 자원을 모델화하는 데 사용되는 sunPlat 클래스의 상속 계층을 보여줍니다.

물리적 엔티티 수퍼클래스는 관리 개체 사이의 관계 정의를 위한 속성을 제공합니다. 또한 Equipment 클래스의 속성에 대응하는 표준 SNMP 속성을 제공합니다.

sunPlat Equipment 클래스는 물리적 엔티티 수퍼클래스에서 파생되어 결함 모니터링에 적용할 수 있는 대응하는 클래스에 정의되는 추가 속성을 제공합니다.

sunPlat 장비 홀더 및 sunPlat 회로 팩 클래스는 sunPlat 장비 수퍼클래스에서 파생되어 각각 해당 자원에 꽂는 컨센트와 구성요소를 표시합니다.

그런 다음 DMTF 파생 클래스를 제공하기 위해 sunPlat 장비 클래스가 더욱 전문화됩니다.

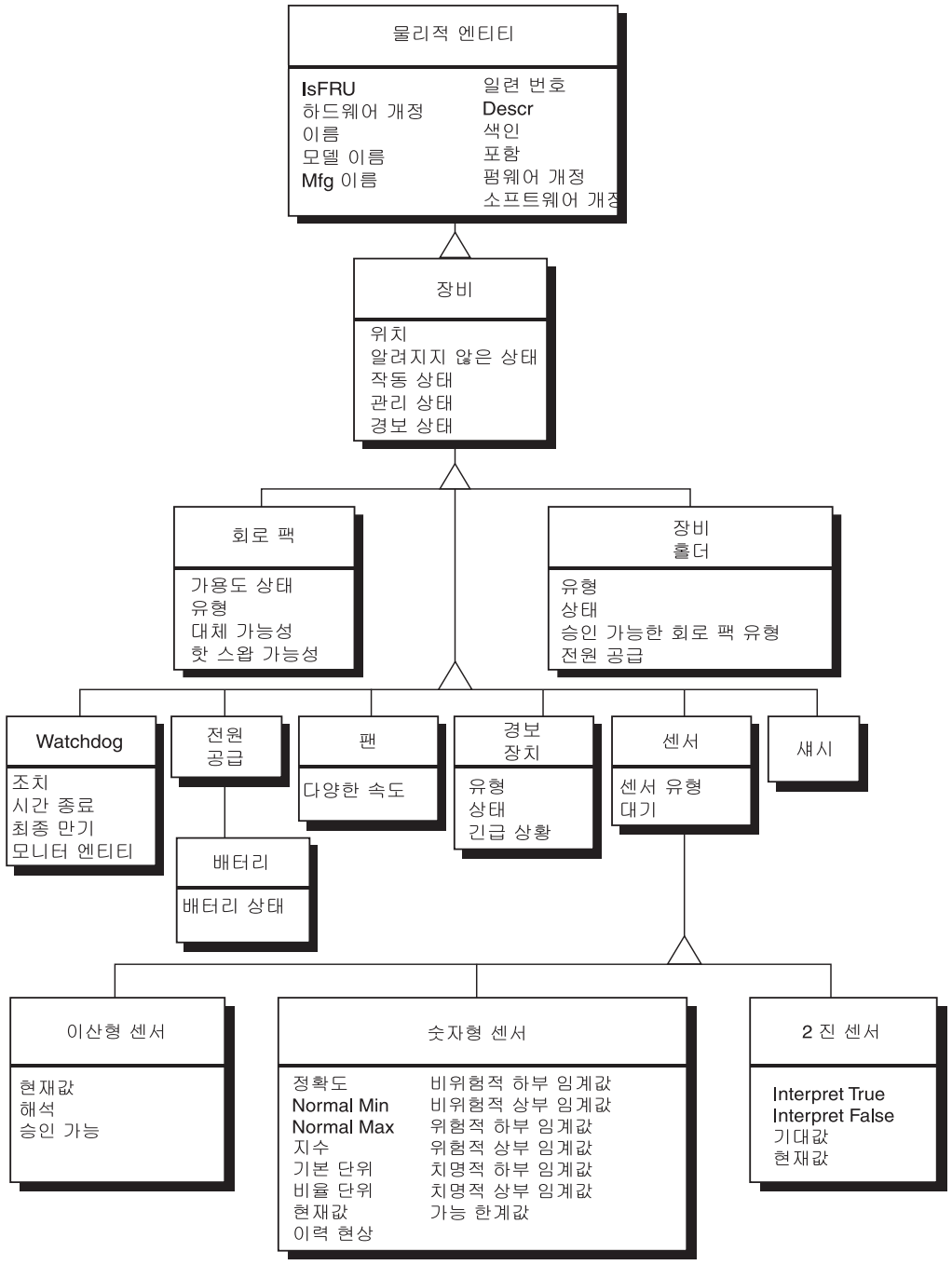

그림 6-1 sunPlat 물리적 자원 상속 클래스 도표

sunPlat 클래스 정의

sunPlat 클래스의 속성이 하드웨어 자원의 특성을 표시하는 데 사용됩니다. 관리자에 대한 자원의 가용성과 조작성은 관리 개체의 *상태*로 표시됩니다. 여러 sunPlat 클래스가 관리 개체의 상태 측면을 표현하는 다양한 속성을 갖습니다.

물리적 엔티티

물리적 엔티티 수퍼클래스는 모든 자원에 공통적인 특성을 표시하는 데 사용됩니다.

참고 – 명확성을 위해 다음 속성 이름에서 *entPhysical* 접두어가 생략되었습니다.

- ***Descr***
  이것은 자원에 대한 알려진 이름을 포함하는 텍스트 문자열입니다. 이 이름은 보통 제품 문서, 제품 범례 또는 가능한 경우 펌웨어에 저장되는 이름에서 자원을 기술하는 데 사용되는 이름입니다.
- ***Is FRU***
  이것은 자원이 현장 대체 가능 장치인지 여부를 표시하는 부울입니다. *sunPlatCircuitPack* 클래스의 하드웨어 자원만이 FRU인 것으로 간주됩니다.
- ***하드웨어 개정***
  이것은 자원에 대한 제조업체의 하드웨어 개정 정보를 포함하는 텍스트 문자열입니다. 모든 하드웨어 자원이 연관된 하드웨어 개정 정보를 갖지는 않습니다.
- ***이름***
  이것은 자원이 운영 체제 및 연관된 유틸리티에 알려지는 논리적 이름을 포함하는 텍스트 문자열입니다. 이 이름은 장치 노드 또는 적용 가능한 경우 시스템 유틸리티가 사용하는 정의된 이름일 수 있습니다. 모든 자원이 장치 이름을 갖지는 않습니다.
- ***모델 이름***
  이것은 제조업체의 고객이 볼 수 있는 부품 번호 또는 부품 정의를 포함하는 텍스트 문자열입니다. 모든 하드웨어 자원이 연관된 부품 번호나 정의를 갖지는 않습니다.
- ***일련 번호***
  이것은 자원에 대한 제조업체의 일련 번호를 포함하는 텍스트 문자열입니다. 모든 하드웨어 자원이 연관된 일련 번호를 갖지는 않습니다.
- ***Mfg 이름***
  이것은 자원에 대한 제조업체 이름을 포함하는 텍스트 문자열입니다. 모든 하드웨어 자원이 연관된 제조업체 이름을 갖지는 않습니다.

물리적 엔티티 수퍼클래스는 또한 하드웨어 자원의 계층을 설명하는 데 사용되는 속성을 포함합니다.

■ *클래스*
이 열거된 유형은 특정 물리적 자원의 일반 하드웨어 유형의 표시를 포함합니다. 이 클래스의 지원되는 값은 ENTITY-MIB에 의해 정의됩니다. 이 속성을 관리 개체에 대한 관련 테이블 확장의 표시로 사용할 수 있습니다. ENTITY-MIB 클래스와 sunPlat 클래스 사이의 맵핑은 표 6-1에 표시된 것과 같습니다.

표 6-1 물리적 엔티티 수퍼클래스 ëClassí 속성 맵핑

entPhysicalClass	**sunPlat 클래스**
chassis(3)	*sunPlat 새시*
backplane(4)	구현되지 않음
container(5)	*sunPlat 장비 홀더*
powerSupply(6)	*sunPlat 전원 공급장치*
fan(7)	*sunPlat 팬*
sensor(8)	*sunPlat 센서*, 서브클래스 포함
module(9)	*sunPlat 회로 팩*
port(10)	구현되지 않음
stack(11)	구현되지 않음
other(1)	*sunPlat 장비*, 서브클래스 포함
unknown(2)	구현되지 않음

■ *색인*
이 정수는 관리 개체를 식별하는 물리적 엔티티 테이블의 항목을 고유하게 식별합니다. 값은 사전 할당되지 않으며 에이전트의 각 호출 시 달라질 수 있습니다.

■ *색인 속성*
이 정수는 이 관리 개체를 포함하는 관리 개체의 *색인* 속성을 표시합니다. 그러므로 이 속성은 관리 개체 사이의 관계를 모델화합니다.

참고 – 물리적 포함 계층의 루트에 있는 개체(보통 새시)는 테이블에 표시되는 다른 엔티티 안에 물리적으로 포함되지 않습니다. 이를 표시하기 위해, *entPhysicalContainedIn* 값이 `0`으로 설정됩니다.

■ *펌웨어 개정*
이것은 자원에 대한 제조업체의 펌웨어 개정 정보를 포함하는 텍스트 문자열입니다. 모든 하드웨어 자원이 연관된 펌웨어 개정 정보를 갖지는 않습니다.

■ *소프트웨어 개정*
이것은 자원에 대한 제조업체의 소프트웨어 개정 정보를 포함하는 텍스트 문자열입니다. 모든 하드웨어 자원이 연관된 소프트웨어 개정 정보를 갖지는 않습니다.

sunPlat 장비 클래스

sunPlat 장비 클래스는 모든 자원에 공통적인 특성을 표시하는 데 사용됩니다. 이 클래스는 구성 및 일반 작동 상태 정보를 표시하는 속성을 포함합니다. 이 클래스는 더욱 서브클래스화되어 특정 유형의 자원에 대한 보다 상세한 구성 정보 및 모니터링 데이터를 제공합니다.

entPhysicalClass는 표시되는 서브클래스에 의존합니다.

sunPlat 장비 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *entPlatEquipment* 접두어가 생략되었습니다.

- ***관리 상태***
  이 읽기-쓰기 속성은 자원의 현재 관리 상태를 표시하는 다음 열거된 값 중 하나를 갖습니다.
  - `locked(1)`
  - `unlocked(2)`
  - `shuttingDown(3)`
- ***작동 상태***
  이 읽기 전용 속성은 자원이 실제로 설치되고 서비스를 제공할 수 있는지 여부를 표시하는 열거된 유형입니다. 이 속성이 관리 개체의 *상태*에 기여하며 표 6-2에 표시된 값을 가질 수 있습니다.

표 6-2 작동 상태 속성 값

속성 값	설명
`disabled(1)`	자원이 전적으로 작동 불가능하며 사용자에게 서비스를 제공할 수 없습니다.
`enabled(2)`	자원이 부분적으로 또는 완전히 작동 가능하며 사용할 수 있습니다.

- ***경보 상태***
  이 읽기 전용 속성은 자원의 현재 경보 상태를 표시하는 열거된 값을 갖습니다. 관리 개체에서 미해결인 모든 경보의 가장 높은 심각도를 표시합니다. 이 속성은 다음 값을 가질 수 있습니다.
  - `critical(1),`
  - `major(2),`
  - `minor(3),`
  - `indeterminate(4),`
  - `warning(5),`
  - `pending(6),`

- `cleared(7)`

- *알려지지 않은 상태*
  이 읽기 전용 속성은 다른 상태 속성이 자원의 실제 속성을 반영하지 않을 수 있는지를 표시합니다. 이 속성은 관리 개체가 자원에 대한 결함을 정확하게 보고할 수 있는지 여부를 표시하는 부울 값을 갖습니다. 자원이 절대적으로 *상태*를 반영할 수 없는 경우, 이 속성은 `true`로 설정됩니다.

- *위치 이름*
  이 읽기 전용 속성은 자원에 대한 위치 지정자를 포함합니다. 새시 안에 직접 들어있는 자원의 경우, 이 속성은 슬롯 및 제품 문서의 범례와 관련되거나 새시 안에서 자원의 지리학적 위치 표시를 제공합니다. 다른 하드웨어 자원은 보통 해당 자원이 들어있는 자원에 대한 관리 개체의 *이름*에 해당하는 *위치*를 갖습니다.

sunPlat 회로 팩 클래스

sunPlat 회로 팩 클래스는 대체 가능 자원 또는 FRU에 일반적인 특성을 표시하는 데 사용됩니다. 대체 가능 자원은 그의 목적이 내부 하드웨어 구성요소를 인식되는 폼 팩터에 패키지하는 것인 하드웨어 모듈로서 정의됩니다. 일반적으로 FRU는 정의된 폼 팩터와 물리적 외양을 갖습니다. 커넥터에 꽂는 접속 가능한 제거 가능 장치이거나, 베이 안에 영구적으로 설치할 수 있거나, 드로어, 랙 또는 선반에 연결할 수 있습니다.

이 클래스는 entPhysicalClass `module(9)`을 갖습니다.

sunPlat 회로 팩 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatCircuitPack* 접두어가 생략되었습니다.

- *유형*
  이 읽기 전용 속성은 자원의 컨테이너 호환성을 평가하는 데 사용되는 텍스트 문자열입니다. 이 속성은 자원의 기능과 폼 팩터 특성을 식별할 수 있습니다.

■ *가용성 상태*
이 읽기 전용 속성은 관리 개체의 *작동 상태*를 규정합니다. BITS 구문을 사용하는 개체이며, 표 6-3에 표시된 값 세트의 0개 이상을 취할 수 있습니다. 모든 값이 관리 개체의 모든 클래스에 적용될 수 있는 것은 아닙니다. 이 속성은 관리 개체의 *상태*에 기여합니다.

표 6-3 가용성 상태 속성 값

속성 값	비트 번호	16진	설명
inTest(0)	0	80	자원이 테스트 절차를 진행 중입니다.
failed(1)	1	40	자원에 작동하지 못하게 하는 내부 결함이 있습니다. *작동 상태*는 disabled(1)입니다.
powerOff(2)	2	20	자원에 전원이 공급되어야 하며 켜지지 않았습니다.
offLine(3)	3	10	자원이 온라인으로 만들고 사용 가능하게 하기 위한 루틴 작업이 수행되어야 합니다. *작동 상태*는 disabled(1)입니다.
offDuty(4)	4	08	자원이 내부 제어 프로세스에 의해 비활성화되었습니다.
dependency(5)	5	04	자원이 의존하는 일부 다른 자원이 사용 불가능하기 때문에 동작할 수 없습니다. *작동 상태*는 disabled(1)입니다.
degraded(6)	6	02	자원에서 사용 가능한 서비스가 속도 또는 동작 능력과 같은 일부 측면에서 저하됩니다. 그러나 자원은 계속 서비스에 사용 가능합니다. *작동 상태*는 enabled(2)입니다.
notInstalled(7)	7	01	관리 개체에 의해 표시되는 자원이 존재하지 않거나 불완전합니다. *작동 상태*는 disabled(1)입니다.

■ *대체 가능*
이 읽기 전용 속성은 자원이 대체 가능 장치인지 여부를 표시하는 부울 값을 갖습니다.

■ *핫 스왑 가능*
이 읽기 전용 속성은 대체 가능 자원이 핫 스왑 가능한지 여부를 표시하는 부울 값을 갖습니다.

sunPlat 장비 홀더

sunPlat 장비 홀더 클래스는 제거 가능한 하드웨어 자원을 보유할 수 있는 하드웨어 자원의 특성을 표시하는 데 사용됩니다.

이 클래스는 entPhysicalClass `container(5)`를 갖습니다.

sunPlat 장비 홀더 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *entPlatEquipmentHolder* 접두어가 생략되었습니다.

■ *유형*
이 읽기 전용 속성은 표 6-4에 표시된 것처럼 자원의 홀더 유형을 표시하는 열거된 유형입니다.

표 6-4 장비 홀더 유형 속성 값

속성 값	설명
`bay(1)`	베이는 보통 원격통신 장비를 보유하기 위한 선반이나 드로어를 포함하는 랙 안에 있는 수직 공간의 한 단위입니다. SunPlat은 새시 내에서의 베이 사용을 신호 연결을 위해 케이블이 필요한 물리적 콘센트로서 해석합니다.
`shelf(2)`	랙 안에서 원격통신 장비를 보유하기 위한 수평 지지대 또는 서브랙.
`drawer(3)`	랙 안에서 원격통신 장비를 보유하기 위한 수평 격납장치.
`slot(4)`	제거 가능 장비를 위한 신호 연결용 통합 커넥터를 갖는 물리적 콘센트.
`rack(5)`	랙은 자체 포함 격납장치 안에 원격통신 장비, 홀더 및 케이블 관리 시스템을 보유하기 위한 지지 하부구조입니다.

■ *가능 자원 유형*
이 읽기 전용 속성은 홀더가 지지하는 제거 가능 자원(회로 팩)의 유형을 표시하는 텍스트 문자열의 목록입니다. 이들 유형은 제거 가능 자원의 *Type* 속성과의 호환성이 테스트됩니다.

■ *상태*
이 읽기 전용 속성은 표 6-5에 표시된 것처럼 자원이 포함할 수 있는 모든 대체 가능 하드웨어 자원(회로 팩)과 관해서 홀더의 상태를 표시하는 열거된 유형입니다.

표 6-5 장비 홀더 상태 속성 값

속성 값	설명
`holderEmpty(1)`	홀더에 제거 가능 자원이 없습니다.

표 6-5 장비 홀더 상태 속성 값

`inTheAcceptableList(2)`	홀더에 *승인 가능한 회로 팩 유형* 목록에 있는 유형 중 하나인 제거 가능 자원이 들어있습니다.
`notInTheAcceptableList(3)`	홀더에 네트워크 요소가 인식할 수 있는 제거 가능 자원이 들어있지만, *승인 가능한 회로 팩 유형* 목록에 있는 유형 중 하나가 아닙니다.
`unknownType(4)`	홀더에 인식할 수 없는 제거 가능 자원이 들어있습니다.

- ***전원***
  이 읽기-쓰기 속성은 자원의 전원 상태를 표시하는 열거된 유형입니다. 가능한 값은 다음과 같습니다.
  - `other(1)`
  - `unknown(2)`
  - `powerOff(3)`
  - `powerOn(4)`

sunPlat 전원 공급장치

sunPlat 전원 공급장치 클래스는 전원 공급장치를 표시하는 데 사용됩니다. sunPlat 장비 클래스의 특성을 확장하지 않습니다. 전원 공급장치는 보통 모니터되는 등록정보(예: 전압, 전류 및 온도)를 표시하는 센서를 포함합니다. 또한 팬과 같은 다른 하드웨어 자원도 포함할 수 있습니다. 이것은 관리 개체 사이의 관계를 사용하여 모델화됩니다.

전원 공급장치가 제거 가능 자원인 경우, sunPlat 회로 팩 클래스의 관리 개체 안에 모델화됩니다.

이 클래스는 entPhysicalClass `powerSupply(6)`를 갖습니다.

sunPlat 전원 공급장치 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatPowerSupply* 접두어가 생략되었습니다.

- ***클래스***

  이 읽기 전용 속성은 전원 공급장치의 클래스를 표시하는 열거된 유형이며, 다음 값을 취합니다.
  - `other(1)`
  - `powerSupply(2)`
  - `battery(3)`

sunPlat 배터리

sunPlat 배터리 클래스는 배터리로부터 전원을 공급하는 전원 공급장치를 표시하는 데 사용됩니다.

이 클래스는 entPhysicalClass `powerSupply(6)` 및 SunPlatPowerSupplyClass `battery(3)`를 갖습니다.

sunPlat 배터리 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatBattery* 접두어가 생략되었습니다.

- *상태*
  이 읽기 전용 속성은 배터리의 상태를 표시하는 열거된 유형이며, 다음 값을 취합니다.
  - `other(1)`
  - `unknown(2)`
  - `fullyCharged(3)`
  - `low(4)`
  - `critical(5)`
  - `charging(6)`
  - `chargingAndHigh(7)`
  - `chargingAndLow(8)`
  - `chargingAndCritical(9)`
  - `undefined(10)`
  - `partiallyCharged(11)`

sunPlat Watchdog

sunPlat Watchdog 클래스는 하드웨어가 운영 체제 또는 응용프로그램의 상태를 모니터할 수 있게 하는 타이머 하드웨어 자원의 특성을 표시하는 데 사용됩니다.

이 클래스는 entPhysicalClass `other(1)` 및 sunPlatPhysicalClass `watchdog(3)`을 갖습니다.

sunPlat Watchdog 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatWatchdog* 접두어가 생략되었습니다.

- *시간 종료*
  이 읽기 전용 속성은 재설정되지 않는 경우 watchdog이 시간 종료할 간격을 밀리초 단위로 표시하는 정수입니다.
- *조치*
  이 읽기 전용 속성은 *시간종료*에 의해 지정된 기간 안에 재설정되지 않는 경우 watchdog이 취하는 조치를 표시하는 열거된 유형입니다. 가능한 값이 표 6-6에 표시됩니다.

표 6-6 Watchdog 조치 속성 값

조치	설명
`statusOnly(1)`	Watchdog이 소프트웨어에 의해 읽을 수 있지만 어떤 조치도 취하지 않습니다.
`systemInterrupt(2)`	Watchdog이 모니터되는 시스템에 하드웨어 인터럽트를 생성합니다.
`systemReset(3)`	Watchdog이 모니터되는 시스템을 재설정합니다.
`systemPowerOff(4)`	Watchdog이 모니터되는 시스템의 전원을 끕니다.
`systemPowerCycle(5)`	Watchdog이 모니터되는 시스템의 전원을 끈 후 다시 켭니다.

- *만기일*
  이 읽기 전용 속성은 watchdog이 마지막으로 만기된 날짜 및 시간을 표시합니다.
- *모니터 가능한 엔티티*
  이 읽기 전용 속성은 watchdog이 모니터할 수 있는 엔티티를 표시하는 열거된 유형입니다. 가능한 값은 다음과 같습니다.
  - `unknown(1)`
  - `other(2)`
  - `operatingSystem(3)`
  - `operatingSystemBootProcess(4)`
  - `operatingSystemShutdownProcess(5)`
  - `firmwareBootProcess(6)`
  - `biosBootProcess(7)`
  - `application(8)`
  - `serviceProcessors(9)`

sunPlat 경보

sunPlat 경보 클래스는 문제점 상황과 관련된 표시(예: 경고음, LED, 릴레이, 진동자 및 소프트웨어 경보)를 발령하는 하드웨어 자원의 특성을 표시하는 데 사용됩니다.

이 클래스는 entPhysicalClass `other(1)` 및 sunPlatPhysicalClass `alarm(2)`을 갖습니다.

sunPlat 경보 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatAlarm* 접두어가 생략되었습니다.

- ***유형***
  이 읽기 전용 속성은 경보 조건이 통신되는 방법을 표시하는 열거된 유형입니다. 가능한 값이 표 6-7에 표시됩니다.

표 6-7 경보 유형 속성 값

속성 값	설명
`other(1)`	경보 장치 유형이 다음 중 하나가 아닙니다.
`audible(2)`	경보 장치가 장치에서의 청취 가능한 변화입니다.
`visible(3)`	경보가 장치에서 시각적인 변화를 유발합니다.
`motion(4)`	경보가 장치의 움직임을 유발합니다.
`switch(5)`	경보가 전기적 신호 변경을 유발합니다.

- ***상태***
  이 읽기-쓰기 속성은 경보의 상태를 표시하는 열거된 유형입니다. 가능한 값이 표 6-8에 표시됩니다.

표 6-8 경보 상태 속성 값

속성 값	설명
`unknown(1)`	경보의 상태가 정의되지 않았거나 관측 불가능합니다.
`off(2)`	경보가 비활성입니다.
`steady(3)`	경보가 활성입니다.
`alternating(4)`	경보가 비활성 상태와 활성 상태 사이에서 순환 중입니다.

- ***긴급***
  이 읽기-쓰기 속성은 경보가 깜박이거나, 진동하거나 가청음을 발령하는 상대적 빈도를 표시하는 열거된 유형입니다. 가능한 값은 다음과 같습니다.
  - `other(1)`
  - `unknown(2)`
  - `notSupported(3)`
  - `informational(4)`
  - `nonCritical(5)`

- `critical(6)`
- `unrecoverable(7)`

sunPlat 팬

sunPlat 팬 클래스는 활성 냉각 장치의 특성을 표시하는 데 사용됩니다. 팬은 일반적으로 회전 속도를 표시하는 센서를 포함합니다. 이것은 sunPlat 팬 관리 개체와 sunPlat 센서 클래스의 유속계 관리 개체 사이의 물리적 포함 관계를 사용하여 모델화됩니다.

이 클래스는 entPhysicalClass `fan(7)`을 갖습니다.

sunPlat 팬 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatFan* 접두어가 생략되었습니다.

- ***클래스***
  이 읽기 전용 속성은 냉각 장치의 클래스를 표시하는 열거된 유형이며, 다음 값을 취합니다.
  - `other(1)`
  - `fan(2)`
  - `refrigeration(3)`
  - `heatPipe(4)`

sunPlat 센서

sunPlat 센서 수퍼클래스는 다른 하드웨어 자원의 성질을 측정하는 하드웨어 자원의 일반 특성을 표시하는 데 사용됩니다.

이 클래스는 entPhysicalClass `sensor(8)`을 갖습니다.

sunPlat 센서 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatSensor* 접두어가 생략되었습니다.

- ***클래스***
  이 읽기 전용 속성은 센서의 클래스를 표시하는 열거된 유형이며, 다음 값을 취합니다.
  - `binary(1)`
  - `numeric(2)`
  - `discrete(3)`

■ *유형*
이 읽기 전용 속성은 센서가 측정하는 성질을 표시하는 열거된 유형입니다. *유형*의 가능한 값 중 일부가 표 6-9에 표시됩니다.

표 6-9 센서 유형 속성 값

유형	설명
`temperature(3)`	환경 온도 측정용 센서
`voltage(4)`	전압 측정용 센서
`current(5)`	전류 측정용 센서
`tachometer(6)`	장치의 속도/회전률 측정용 센서
`counter(7)`	정의된 이벤트를 세는 범용 센서

■ *대기 시간*
이 읽기 전용 속성은 다음을 표시합니다.
- 센서가 폴링되는 경우, 이 정수는 밀리초 단위로 측정된 갱신 간격을 표시합니다.
- 센서가 이벤트 구동되는 경우, 이 값은 해당 이벤트 처리에서 최대 예상 대기 시간을 표시합니다.

sunPlat 2진 센서

sunPlat 2진 센서 클래스는 2진 출력을 반환하는 센서의 특성을 표시하는 데 사용됩니다. sunPlatSensor 테이블을 증대시켜서 2진 센서에 특정한 속성을 제공합니다.

이 클래스는 entPhysicalClass `sensor(8)` 및 sunPlatSensorClass `binary(1)`를 갖습니다.

sunPlat 2진 센서 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatBinarySensor* 접두어가 생략되었습니다.

■ *최근 값*
이 읽기 전용 속성은 센서의 가장 최근 값을 표시하는 부울 값을 갖습니다.

■ *예상 값*
이 읽기 전용 속성은 센서의 예상 값을 표시하는 부울 값을 갖습니다.

■ ***True 값 설명***
이 읽기 전용 속성은 센서의 `true` 값의 해석을 표시하는 텍스트 문자열입니다.

■ ***False 값 설명***
이 읽기 전용 속성은 센서의 `false` 값의 해석을 표시하는 텍스트 문자열입니다.

sunPlat 숫자 센서

sunPlat 숫자 센서 클래스는 숫자 값을 반환할 수 있는 센서의 특성을 표시하는 데 사용됩니다. 숫자 센서 값은 아래에 정의되는 것처럼 측정 단위에 의해 규정됩니다.

측정 단위 = *기본 단위* * $10^{지수}$

이 규정은 밀리암페어 및 마이크로볼트 같은 측정 단위에 허용됩니다. *비율 단위*가 정의되는 경우, 측정 단위는 아래와 같이 더욱 정제됩니다.

측정 단위 = *비율 단위*당 *기본 단위* * $10^{지수}$

이 규정은 rpm 및 lm/hr 같은 측정 단위에 허용됩니다.

이 클래스는 entPhysicalClass `sensor(8)` 및 sunPlatSensorClass `numeric`(2)를 갖습니다.

sunPlat 숫자 센서 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatNumericSensor* 접두어가 생략되었습니다.

- ***기본 단위***
  이 읽기 전용 속성은 위에서 정의되는 것같은 규정에 앞서서 측정 단위를 표시하는 열거된 유형입니다. 이 유형의 값의 예는 다음과 같습니다.
  - `degC(3)`
  - `volts(6)`
  - `amps(7)`
- ***진수***
  이 읽기 전용 속성은 *Base Unit*를 10의 승수로 크기를 맞추는 데 사용되는 정수입니다. 예를 들어 *sunPlatNumericSensorBaseUnits*가 `volts`로 설정되고 *sunPlatNumericSensorExponent*가 `-6`으로 설정되는 경우, 반환되는 값의 단위는 `microVolts`입니다.
- ***비율 단위***
  이 읽기 전용 속성은 센서가 절대값(이 값이 `none`일 때)을 측정하는지 아니면 비율을 측정하는지 여부를 표시하는 열거된 유형입니다. 후자의 경우에, *sunPlatNumericSensorBaseUnits*에 지정되는 단위가 '단위 시간당'으로 표현됩니다. 예를 들어 *sunPlatNumericSensorBaseUnits*가 `degC`로 설정되고 *sunPlatNumericSensorRateUnits*가 `perSecond`로 설정되면 표시되는 값은 `degC/second` 단위를 갖습니다.

  이 유형의 값의 예는 다음과 같습니다.
  - `perMicrosecond(2)`
  - `perMillisecond(3)`
  - `perSecond(4)`

- perMinute(5)
- perHour(6)
- none(1)

- ***최근 값***
  이 읽기 전용 속성은 센서의 가장 최근 값을 표시하는 정수입니다.
- ***최소 예상 값***
  이 읽기 전용 속성은 센서 측정값이 그 이하로 떨어질 것으로 예상되지 않는 정의된 임계값을 표시하는 정수입니다. 이 값은 위에서 정의한 것과 같은 측정 단위로 표현됩니다. 이 속성은 일부 센서에 적용 불가능할 수 있습니다.
- ***최대 예상 값***
  이 읽기 전용 속성은 센서 측정값이 그 이상으로 올라갈 것으로 예상되지 않는 정의된 임계값을 표시하는 정수입니다. 이 값은 위에서 정의한 것과 같은 측정 단위로 표현됩니다. 이 속성은 일부 센서에 적용 불가능할 수 있습니다.
- ***정확성***
  이 읽기 전용 속성은 측정된 성질에 대한 센서의 오류 정도를 소수점 아래 두 자리까지의 퍼센트로 표시하는 정수입니다. 값은 센서 측정값이 동적 범위 전체에서 선형적인지 여부에 따라 달라질 수 있습니다.
- ***Non Critical 하부 임계값***
  이 읽기 전용 속성은 `nonCritical` 조건이 발생하는 하부 임계값을 표시하는 정수입니다.
- ***Non Critical 상부 임계값***
  이 읽기 전용 속성은 `nonCritical` 조건이 발생하는 상부 임계값을 표시하는 정수입니다.
- ***Lower Critical 하부 임계값***
  이 읽기 전용 속성은 `critical` 조건이 발생하는 하부 임계값을 표시하는 정수입니다.
- ***Critical 상부 임계값***
  이 읽기 전용 속성은 `critical` 조건이 발생하는 상부 임계값을 표시하는 정수입니다.
- ***Fatal 하부 임계값***
  이 읽기 전용 속성은 `fatal` 조건이 발생하는 하부 임계값을 표시하는 정수입니다.
- ***Fatal 하부 임계값***
  이 읽기 전용 속성은 `fatal` 조건이 발생하는 상부 임계값을 표시하는 정수입니다.
- ***이력 현상***
  이 읽기 전용 속성은 임계값 주위의 이력 현상을 설명합니다.
- ***가능 임계값***
  이것은 작성될 때 센서를 기본값으로 재설정하는 읽기 전용 속성입니다.

sunPlat 이산형 센서

sunPlat 이산형 센서 클래스는 sunPlat 숫자 센서나 sunPlat 2진 센서 클래스로 표시할 수 없는 센서에 사용됩니다.

이 클래스는 entPhysicalClass `sensor(8)` 및 sunPlatSensorClass `discrete`(3)를 갖습니다.

이 클래스는 두 테이블로 구성됩니다. sunPlatDiscreteSensor 테이블은 하나의 속성, *sunPlatDiscreteSensorCurrent*를 갖는데, 이것은 sunPlatDiscreteSensorStates 테이블의 색인으로 표현되는 센서의 현재 상태를 표시합니다.

sunPlat 이산형 센서 클래스는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatDiscreteSensorState* 접두어가 생략되었습니다.

- *색인*
  이 읽기 전용 속성은 이 센서 상태를 식별하는 sunPlatDiscreteSensorStates 테이블에 있는 행의 색인을 표시하는 숫자를 갖습니다.
- *설명*
  이 읽기 전용 속성은 sunPlatDiscreteSensorStatesTable의 대응하는 행에 의해 표시되는 상태를 설명하는 문자열입니다.
- *승인 가능*
  이 읽기 전용 속성은 테이블의 이 행에 의해 표시되는 상태가 승인 가능한 것으로 간주되는지 여부를 표시하는 부울 값을 취합니다.

sunPlat 섀시

sunPlat 섀시 클래스는 주 격납장치를 표시하는 데 사용됩니다. sunPlat 장비 클래스의 특성을 확장하지 않습니다. 섀시에는 모든 모델화된 하드웨어 자원이 들어있으며, 다른 어떤 자원 안에도 포함되지 않습니다.

이 클래스는 entPhysicalClass `chassis(3)`을 갖습니다.

7 장

논리적 모델

이 장에서는 sunPlat 논리적 클래스 계층과 sunPlat 모델에서 정의되는 관리 개체 클래스가 SUN-PLATFORM-MIB에 의해 표시되는 방법에 대해 설명합니다.

이 장에는 다음 절이 들어있습니다.

sunPlat 논리적 클래스 계층

그림 7-1은 sunPlat 논리적 클래스의 상속 계층을 보여줍니다.

논리적 엔티티 클래스는 모든 논리적 개체에 공통적인 정보를 제공합니다.

단일화 컴퓨터 시스템 클래스는 모델화된 컴퓨터 시스템(예: Sun Fire B1600 새시의 Sun Fire B100 블레이드)의 전원 상태 보고와 관련된 등록정보를 추가하는데, 이것은 또한 강제 재설정을 초래하는 데 사용할 수도 있습니다.

관리 도메인 클래스는 추가 등록정보를 추가하지 않지만, 모델화된 시스템과의 관리 접속을 표시하는 논리적 개체를 표시하는 데 사용됩니다. Sun Fire B1600 플랫폼의 경우에 이것은 시스템 컨트롤러를 표시하는 데 사용됩니다.

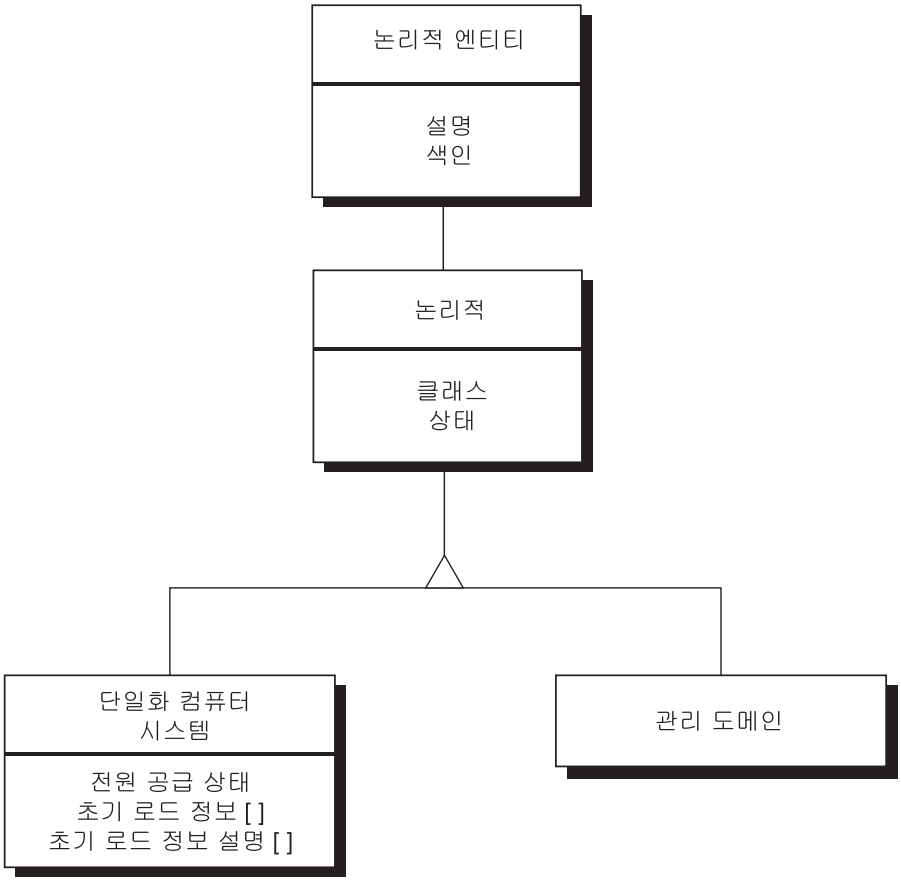

그림 7-1 sunPlat 논리적 자원 상속 클래스 도표

sunPlat 논리적 클래스 정의

논리적 sunPlat 클래스의 속성이 논리적 자원의 특성을 표시하는 데 사용됩니다. 그런 자원은 다중 도메인 시스템의 도메인 같은 *고가* 개체를 대표합니다. 관리자에 대한 자원의 가용성과 조작성은 관리 개체의 상태로 표시됩니다. 여러 sunPlat 클래스가 관리 개체 상태 측면을 표현하는 다양한 속성을 갖습니다.

논리적 엔티티

이 클래스는 식별 정보를 제공하는 논리적 엔티티를 표시합니다. 중요한 개체는 다음과 같습니다.

참고 – 명확성을 위해 다음 개체 이름에서 *entLogical* 접두어가 생략되었습니다.

■ ***설명***

이 개체는 관리되는 개체의 유형을 식별합니다.

■ ***주소***

이것은 그를 통해 엔티티를 직접 관리할 수 있는 IP 주소와 UDP 포트 번호를 제공합니다. Sun Fire B1600 시스템의 Sun Fire B100 블레이드의 경우, 이것은 블레이드의 IP 주소와, 블레이드의 표준 Solaris SNMP 에이전트가 접속되는 포트 161을 제공합니다.

논리적

이 클래스는 이 논리적 엔티티에 의해 표시되는 자원의 상태 유형을 표시합니다. 클래스에 다음 개체가 들어있습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatLogical* 접두어가 생략되었습니다.

■ ***클래스***

이 속성은 논리적 클래스의 유형을 표시하는 열거된 유형이며, 다음 값을 갖습니다.

- `other(1)`
- `computerSystem(2)`
- `adminDomain(3)`

■ ***상태***

이 속성은 논리적 클래스의 상태를 표시하는 열거된 유형입니다. 이 속성은 다음 값을 가질 수 있습니다.

- `ok(1)`
- `error(2)`
- `degraded(3)`
- `unknown(4)`
- `predFail(5)`
- `starting(6)`
- `stopping(7)`

- service(8)
- stressed(9)
- nonRecover(10)
- noContact(11)
- lostComm(12)
- stopped(13)

sunPlat 단일화 컴퓨터 시스템

이 클래스의 특정 등록정보가 sunPlatUnitaryComputerSystemTable을 사용하여 표시됩니다. computerSystem(2)의 sunPlat 논리적 클래스를 갖습니다.

클래스에 다음 개체가 들어있습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatUnitaryComputerSystem* 접두어가 생략되었습니다.

- ***전원 상태***

이 속성은 읽을 때 현재 전원 상태를 표시합니다. 또한 전원 상태의 원격 제어를 가능케 하는데, 예를 들면 블레이드를 켜고 끄거나, 강제 재설정을 초래하도록 전원 주기를 수행할 수 있습니다.

이 속성은 다음 값을 가질 수 있습니다.

- unknown(1)
- fullPower(2)
- psLowPower(3)
- psStandby(4)
- psOther(5)
- powerCycle(6)
- powerOff(7)
- psWarning(8)
- hibernate(9)
- softOff(10)
- reset(11)

- ***적용설정***

이 등록정보를 쓰면 부트 매개변수의 기본값 또는 사용자 정의 세트가 적용될 수 있습니다.

sunPlatUnitaryComputerSystemTable의 각 항목이 현재 부트 매개변수 설정 및 *sunPlatUnitaryComputerSystemApplySettings* 개체에 써서 적용될 수 있는 대체 세트 모두를 정의하는 *sunPlatInitialLoadInfoTable*의 항목 세트와 연관됩니다.

sunPlat 관리 도메인

이 클래스는 논리적 엔티티 클래스에 등록정보를 추가하지 않으며 따라서 연관된 MIB 개체가 없습니다. 클래스는 `adminDomain(3)`의 sunPlat 논리적 클래스를 갖습니다.

8 장

sunPlat 통지

이 장에서는 SUN-PLATFORM-MIB에 정의된 것처럼 SunPlat 통지 클래스 및 속성에 대해 설명합니다.

sunPlat 통지 클래스는 에이전트가 등록된 네트워크 관리자에 전송한 비동기 메시지입니다. 이들은 관리 개체 폴링을 통해 달성할 수 있는 것보다 효율적으로 이벤트 정보를 전달하는 데 사용됩니다.

이 장에는 다음 절이 들어있습니다.

sunPlat 통지 클래스 계층

그림 8-1은 sunPlat 통지 클래스의 상속 계층을 보여줍니다.

통지 클래스 세트는 이들 클래스 사이에 공통 속성을 전개하는 추상 및 실재 클래스 모두의 계층을 사용하여 표시됩니다.

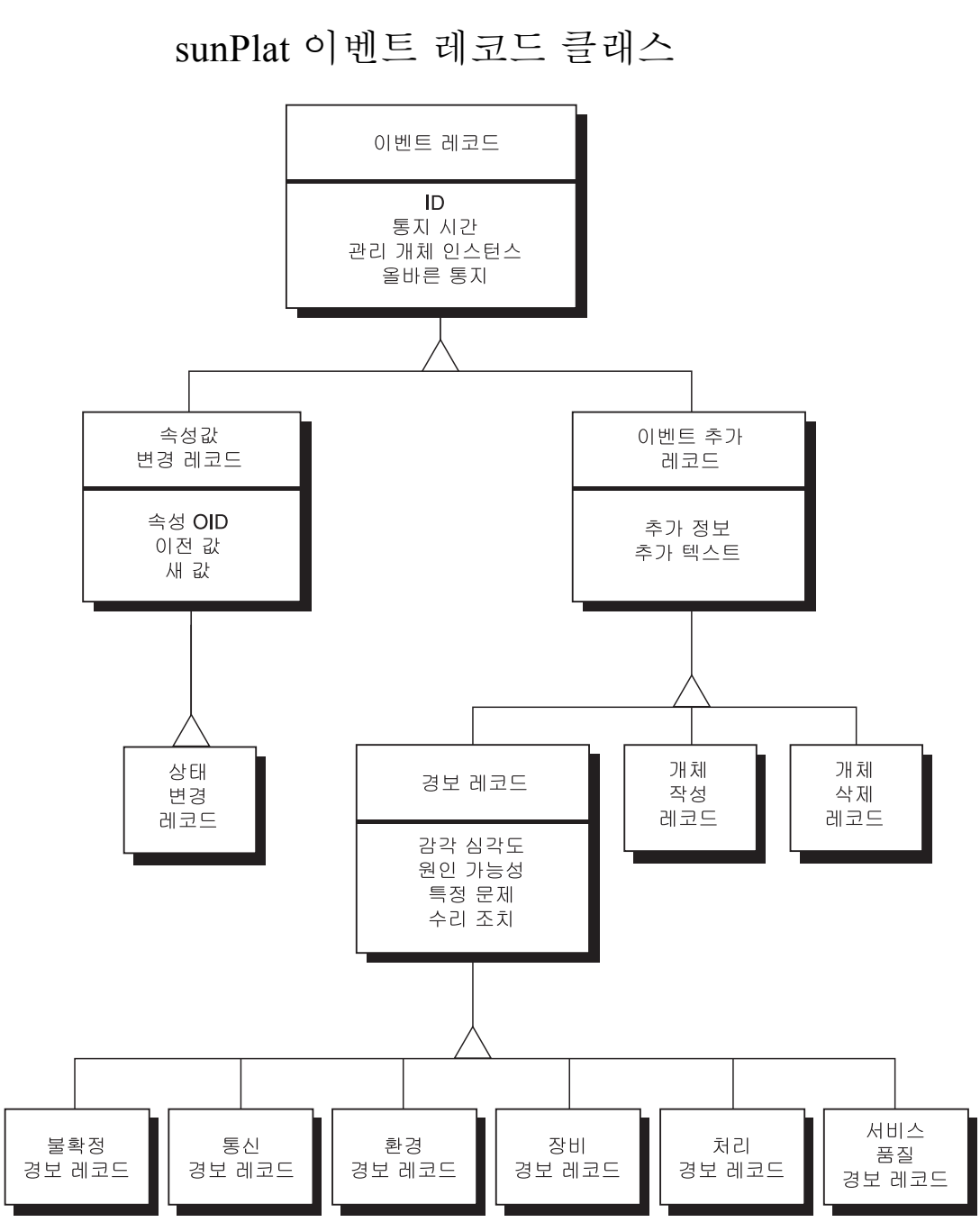

그림 8-1 이벤트 레코드 상속 클래스 도표

sunPlat 클래스 정의

sunPlat 이벤트 레코드

sunPlat 이벤트 레코드 수퍼클래스는 모든 통지에 공통인 속성을 표시합니다. 이 클래스는 클래스가 기록하는 특정 이벤트와 관련된 추가 정보를 제공하도록 추가로 서브클래스화됩니다.

sunPlat 이벤트 레코드는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatLogRecord* 접두어가 생략되었습니다.

- ***ID***
  이것은 통지 및 통지가 에이전트에 의해 생성된 순서의 표시를 고유하게 식별하는 정수입니다. 에이전트가 통지 순서가 통지가 생성된 기초 이벤트의 순서를 반영한다고 보증하지 않음을 주의하십시오.
- ***통지 시간***
  이 읽기 전용 속성은 통지가 생성된 시간을 손쉽게 식별하는 시간소인입니다.
- ***지원 대상 예***
  이 읽기 전용 속성은 이벤트가 연관된 자원을 표시하는 MIB의 항목에 대한 직접 참조를 제공하는 OID입니다.
- ***연관 통지***
  이 읽기 전용 속성은 이 이벤트가 연관되는 다른 이벤트를 식별하는 ID 값의 쉼표로 구분된 목록입니다.

sunPlat 이벤트 추가 레코드

sunPlat 이벤트 추가 레코드 수퍼클래스는 다음 이벤트가 발생할 때 생성되는 통지에 공통적인 추가 속성을 표시합니다.

- 개체 작성
- 개체 삭제
- 경보

이 클래스는 클래스가 기록하는 특정 이벤트에 적용할 수 있는 추가 정보를 제공하도록 추가로 서브클래스화됩니다.

sunPlat 이벤트 추가 레코드는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatLogRecord* 접두어가 생략되었습니다.

- ***추가 정보***
  이 읽기 전용 속성은 이 통지에 관련된 추가 정보를 제공할 수 있는 개체의 선택적 OID입니다.
- ***추가 텍스트***
  레이블과 `entPhysical` 이름으로 영향을 받는 구성요소를 식별하는 이 통지와 관련된 추가 정보를 제공하는 읽기 전용 속성 선택적 텍스트 문자열.

sunPlat 개체 작성 레코드

sunPlat 개체 작성 레코드 클래스는 자원이 연관된 자원 아래의 계층에 추가되었음을 표시하는 데 사용됩니다. *Additional Info* 속성은 추가된 자원을 표시하는 *물리적 엔티티 테이블* 항목의 OID를 포함합니다.

논리적 개체는 `0.0`의 관리자 개체 인스턴스 아래에 작성됩니다.

sunPlat 개체 삭제 레코드

sunPlat 개체 삭제 레코드는 자원이 연관된 자원 아래에서 계층에서 제거되었음을 표시하는 데 사용됩니다. *Additional Info* 속성은 제거된 자원을 표시하는 *물리적 엔티티 테이블* 항목의 OID를 포함합니다.

참고 – 이 OID는 더 이상 유효하지 않지만 여전히 수신 관리자에 사용될 수 있습니다.

---

sunPlat 경보 레코드

sunPlat 경보 레코드 수퍼클래스는 경보를 표시하는 모든 통지에 공통인 추가 속성을 표시합니다.

이 클래스는 발생한 경보의 클래스 식별을 위해 더욱 서브클래스화됩니다.

sunPlat 경보 레코드는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatAlarmRecord* 접두어가 생략되었습니다.

■ *예상 심각도*
이 읽기 전용 속성은 자원의 서비스가 문제점에 의해 영향을 받은 방법을 표시하는 6가지 심각도 레벨을 정의하는 열거된 유형입니다. 이들 값이 표 8-1에 표시됩니다.

표 8-1 sunPlat 경보 레코드 심각도 값

감각 심각도	설명
`indeterminate(1)`	경보에 대한 심각도 레벨을 판별할 수 없습니다.
`critical(2)`	서비스에 영향을 주는 조건이 발생했고 즉시 정정 조치가 필요합니다.
`major(3)`	서비스에 영향을 주는 조건이 발생했고 긴급한 정정 조치가 필요합니다.
`minor(4)`	서비스에 영향을 주지 않는 조건이 발생했고 더 심각한 조건이 발생하는 것을 막기 위해 정정 조치를 취해야 합니다.
`warning(5)`	서비스에 영향을 주는 잠재적인 또는 박두한 결함 조건이 발견되었으며 더 심각한 조건 발생을 막기 위해 조치를 취해야 합니다.
`cleared(6)`	이것은 동일한 *Probable Cause* 및 *Specific Problem*(주어진 경우)을 갖는 동일한 경보 클래스의 이 자원에 대한 모든 경보를 지웁니다.

■ *가능한 원인*
이 읽기 전용 속성은 경보가 생성되게 한 조건의 유형에 따라서 추가 규정을 제공하는 선택적 열거 유형입니다. 이 유형의 값의 예는 다음과 같습니다.

- `coolingSystemFailure(134)`
- `IODeviceError(75)`
- `powerProblem(58)`
- `softwareProgramError(283)`

■ *특정 문제*
이 읽기 전용 속성은 경보의 *가능한 원인*에 대한 추가 정제를 식별하는 선택적 텍스트 문자열입니다.

■ *수리 조치*
이 읽기 전용 속성은 권장 수리 조치를 나열하는 문자열입니다.

sunPlat 불확정 경보 레코드

sunPlat 불확정 경보 레코드는 sunPlat 경보 레코드 클래스가 제공하는 정보를 확장하지 않습니다. 이 클래스는 다음 클래스 중 하나에 들어가지 않는 모든 경보를 기록하는 데 사용됩니다.

sunPlat 통신 경보 레코드

sunPlat 통신 경보 레코드는 sunPlat 경보 레코드 클래스가 제공하는 정보를 확장하지 않습니다. 이 클래스는 연관된 자원이 통신 오류를 발견했음을 기록하는 데 사용됩니다.

sunPlat 환경 경보 레코드

sunPlat 환경 경보 레코드는 sunPlat 경보 레코드 클래스가 제공하는 정보를 확장하지 않습니다. 이 클래스는 연관된 자원이 환경에서 문제점을 발견했음을 기록하는 데 사용됩니다.

sunPlat 장비 경보 레코드

sunPlat 장비 경보 레코드는 sunPlat 경보 레코드 클래스가 제공하는 정보를 확장하지 않습니다. 이 클래스는 연관된 자원이 결함을 발견했음을 기록하는 데 사용됩니다.

sunPlat 처리 경보 레코드

sunPlat 처리 경보 레코드는 sunPlat 경보 레코드 클래스가 제공하는 정보를 확장하지 않습니다. 이 클래스는 연관된 자원이 소프트웨어 또는 처리 실패를 발견했음을 기록하는 데 사용됩니다.

sunPlat 서비스 품질 경보 레코드

sunPlat 서비스 품질 경보 레코드는 sunPlat 경보 레코드 클래스가 제공하는 정보를 확장하지 않습니다. 이 클래스는 연관된 자원이 서비스 품질의 변화를 발견했음을 기록하는 데 사용됩니다.

sunPlat 속성 값 변경 레코드

sunPlat 속성 값 변경 레코드 수퍼클래스는 연관된 자원에서의 속성 변경을 표시하는 통지에 공통된 추가 속성을 표시합니다.

이 클래스는 가능한 각 속성 값에 대해 추가로 서브클래스화됩니다.

sunPlat 속성 값 변경 레코드는 다음 속성을 갖습니다.

참고 – 명확성을 위해 다음 속성 이름에서 *sunPlatLogRecordChange* 접두어가 생략되었습니다.

- ***OID***
  이 읽기 전용 속성은 그의 값이 변경된 관리 개체 속성을 표시하는 물리적 엔티티 테이블 또는 논리적 엔티티 테이블의 개체에 대한 직접 참조를 제공하는 OID입니다.

영향을 받은 속성의 구문에 따라서, 새 값과 이전 값이 다음 개체 쌍 중 하나를 사용하여 표시됩니다.

- ***새 정수***
  이 읽기 전용 속성은 관리 개체의 변경된 속성의 새 `INTEGER` 값을 식별합니다. 부호있음 또는 부호없음 유형은 변경된 속성의 설정에 대응합니다.
- ***이전 정수***
  이 읽기 전용 속성은 관리 개체의 변경된 속성의 이전 `INTEGER` 값을 식별합니다. 부호있음 또는 부호없음 유형은 변경된 속성의 설정에 대응합니다.
- ***새 문자열***
  이 읽기 전용 속성은 속성 변경 통지의 새 `OCTET STRING` 값을 식별합니다.
- ***이전 문자열***
  이 읽기 전용 속성은 속성 변경 통지의 이전 `OCTET STRING` 값을 식별합니다.
- ***새 OID***
  이 읽기 전용 속성은 속성 변경 통지의 새 `OBJECT IDENTIFIER` 값을 식별합니다.
- ***이전 OID***
  이 읽기 전용 속성은 속성 변경 통지의 이전 `OBJECT IDENTIFIER` 값을 식별합니다.

sunPlat 상태 변경 레코드

sunPlat 상태 변경 레코드 클래스는 sunPlat 속성 값 변경 레코드 클래스가 제공하는 정보를 확장하지 않습니다. 이 클래스는 자원의 *상태* 측면을 반영하는 관리 개체 속성의 변경을 표시하는 데 사용됩니다.

2 부 설치 및 구성

9 장

관리 소프트웨어 구성요소

이 장에서는 Sun Fire B1600용을 위한 관리 소프트웨어를 구성하는 구성요소에 대해 설명하고 SNMP 소프트웨어 설치를 위한 요구사항을 나열합니다.

이 장에는 다음 절이 들어있습니다.

시스템 관리 옵션

다음 시스템 관리 옵션이 Sun Fire B1600용을 위해 제공됩니다.

- SNMP를 사용한 시스템 모니터링 및 제어
- 보안 관리를 지원하는 SNMPv3 기능
- Sun Management Center 3.0[1] 을 사용한 시스템 모니터링

1. 설치 및 구성을 포함한 자세한 설명에 대해서는 *Sun Fire B1600용 Sun Management Center 3.0 부록*(**부품 번호** 817-2539-10)을 참조하십시오.

계측

플랫폼 유형에 따라서 다음을 채택할 수 있습니다.

- 도메인 에이전트, Sun Fire B100 블레이드에서 실행(도메인 하드웨어 모니터링)

  소프트웨어는 모니터되는 서버에 로컬로 설치되고 해당 서버만을 모니터할 수 있습니다. Sun Fire B1600의 경우, 각 블레이드가 개별적으로 모니터되고 각 에이전트 인스턴스가 단 하나의 블레이드만을 볼 수 있습니다.

- 플랫폼 에이전트, 시스템 컨트롤러를 통해 프록시됨(플랫폼 하드웨어 모니터링)

  소프트웨어는 시스템 컨트롤러를 통해 플랫폼 지침에 액세스하는 원격 서버에 설치됩니다. 플랫폼 에이전트를 사용하면 시스템 컨트롤러가 관리하는 모든 하드웨어를 모니터할 수 있습니다. Sun Fire B1600의 경우, 모든 유형의 모든 블레이드, 전원 공급장치 및 시스템 컨트롤러를 포함하여 블레이드의 전체 선반을 모니터할 수 있습니다.

그림 9-1에서, 플랫폼 하드웨어 모니터링이 Sun Fire B1600용 선반 A 및 B에 채택되고 도메인 하드웨어 모니터링이 Sun Fire B1600용 선반 C에 채택됩니다.

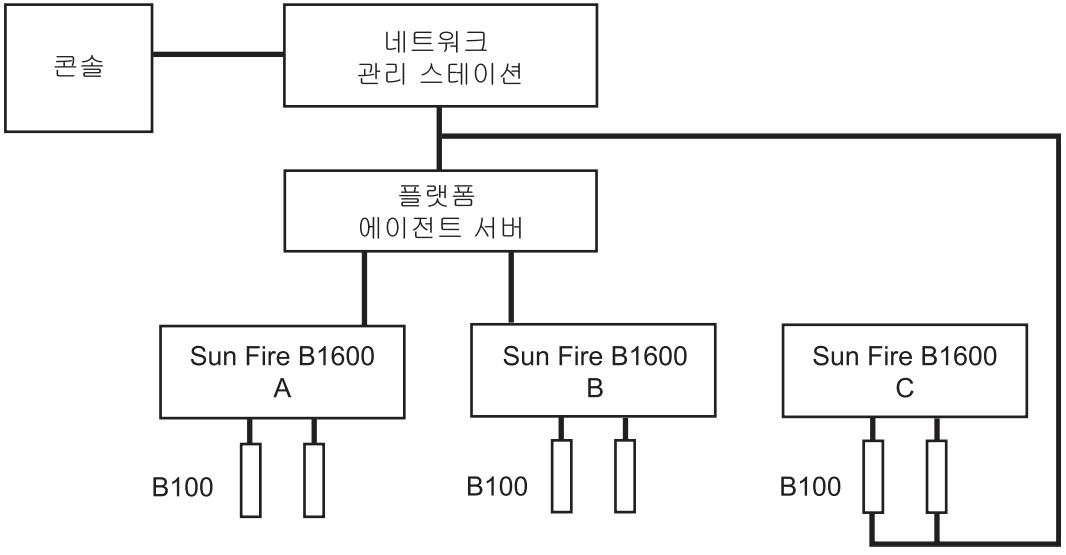

그림 9-1 도메인 및 플랫폼 플랫폼 모니터링의 예

시스템 요구사항

SNMP Management Agent를 설치하기 전에 시스템이 이 절에서 설명하는 전제조건 및 종속성을 준수하는지 확인하십시오.

운영 환경

SNMP Management Agent 소프트웨어는 Solaris 8 업데이트 3 이상의 최신 버전이 필요합니다.

디스크 공간 요구 사항

플랫폼 에이전트 서버에서 최소한 512MB가 사용 가능해야 합니다(1.0GB를 권장합니다).

패치

표준 Solaris 운영 환경 소프트웨어 외에 다음 패치를 설치해야 합니다.

Solaris 8

Sun Fire B100 블레이드의 경우에는 패치가 필요하지 않습니다.

SNMP Management Agent 소프트웨어를 설치하기 전에 플랫폼 에이전트 서버와 블레이드에(플랫폼 및 도메인 하드웨어 모니터링 모두를 위해) Java 1.4를 설치해야 합니다(68 페이지의 “Java 환경”을 참조하십시오).

Solaris 9

패치가 필요없습니다.

Java 환경

Sun Fire B100 블레이드를 완전히 모니터하려면, 모니터되는 각 Sun Fire B100 블레이드와 플랫폼 에이전트 서버에 Java J2SE 1.4 구성요소를 사전 설치해야 합니다.

참고 – 이 설치는 모든 기존 J2SE 소프트웨어를 업그레이드합니다. 소프트웨어를 업그레이드하지 않으려는 경우(예를 들면, 기본 J2SE 버전 1.3.1에 대해 규정된 응용프로그램이 있기 때문에) 기본 시스템 J2SE와 공존하기 위해 J2RE를 설치할 수 있습니다. 이것은 Sun SNMP 관리 에이전트 소프트웨어의 추가 구성이 필요하며, 부록 A에서 설명됩니다.

대상 계측 없이 Sun Fire B1600용 선반만을 모니터하려는 경우, 플랫폼 에이전트 서버에만 Java J2SE 1.4 구성요소를 사전 설치해야 합니다. 이 경우에 하드 디스크 드라이브, CPU 정보 및 이더넷 MAC 주소에 대한 계측을 사용할 수 없습니다.

Java 1.4 파일이 올바른 위치(`/usr/j2se`)에 설치되도록 보장하려면 `j2sdk-1_4_0_03-solaris-sparc.tar.Z` 패키지를 사용하여 Java 1.4를 설치하십시오.

이 파일은 다음 웹 페이지에서 구할 수 있습니다.

`http://java.sun.com/j2se/1.4/download.html`

`Solaris SPARC 32-bit tar.Z`에 대한 SDK 다운로드를 선택하십시오.

위의 위치에서 사용 가능한 이 다운로드에 대한 지침을 따르십시오.

참고 – 이 파일이름은 원고 작성 시점에서는 맞습니다. 이 파일의 최신 버전이 있는지 확인하십시오. 파일 이름은 `j2sdk-1_4_0_<version>-solaris-sparc.tar.Z` 형식을 갖는데, `<version>`은 소프트웨어의 개정판입니다.

이 설치가 시스템 J2SE를 대체하므로, 모든 기존 Java 응용프로그램이 계속 올바르게 실행하도록 보장하기 위해 64 비트 J2SE 1.4 패키지도 설치해야 합니다. 이 패키지는 `j2sdk-1_4_0_<version>-solaris-sparcv9.tar.Z` 파일에 들어있습니다.

주의 – J2SE 1.4는 Solaris 8의 J2SE 1.3.1을 대체하기 위한 것이며 J2SE 1.4를 설치하기 전에 후자를 설치제거해야 합니다. Solaris 8에 대한 후속 분기별 업데이트를 설치하는 경우, 일부 J2SE 1.4 패키지가 J2SE 1.3.1 패키지로 겹쳐써지게 됩니다. J2SE 1.4가 올바른 위치에 설치되도록 하려면 `pkgadd`를 사용하여 설치하십시오.

설치 확인

올바르게 설치했음을 확인하기 위해 다음 명령을 사용하십시오.

```
# /usr/j2se/bin/java -version
java version "1.4.1_03"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_03-
b04)
Java HotSpot(TM) Client VM (build 1.4.1_03-b04, mixed mode)
```

이것은 사용자 시스템에 설치된 버전을 보고합니다.

Java SNMP API

설치 패키지에는 Java SNMP API의 최신 버전인 `SUNWjsnmp`가 포함됩니다. 관리 에이전트 소프트웨어를 설치하기 전에 `pkgrm`을 사용하여 이 패키지의 기존 버전을 제거하십시오.

설치 패키지

관리 소프트웨어를 구성하는 패키지는 다음 그룹으로 나눌 수 있습니다.

- 도메인 하드웨어 모니터링에 필요한 패키지
- 플랫폼 에이전트 서버에 대한 플랫폼 하드웨어 모니터링에 필수인 패키지
- 대상 시스템에 대한 플랫폼 하드웨어 모니터링에 필수인 패키지

이들 패키지가 표 9-1에 표시됩니다.

표 9-1 SNMP Management Agent 소프트웨어 패키지 설명

패키지	패키지 이름	기능
SUNWbgpc	SPA Personality Module Framework	퍼스널리티 모듈을 지원하는 프레임워크
SUNWbgptk	SPA Personality Module Toolkit	구성요소 모델 및 데이터 액세스 라이브러리의 재사용 가능한 라이브러리
SUNWbgpr	SPA Personality Module (루트)	RDP 시작 스크립트
SUNWbgcm	SPA HW Platform Object Manager	플랫폼 개체 관리자
SUNWbgcmr	SPA HW Platform Object Manager (루트)	플랫폼 개체 관리자 시작 스크립트
SUNWbgpm	SPA SNMP Protocol Mediator/Master Agent	SNMP 프로토콜 지원 및 마스터 에이전트
SUNWbgpmr	SPA SNMP Protocol Mediator/Master Agent (루트)	SNMP 구성요소 시작 스크립트
SUNWbgidr	SPA Domain Discovery (루트)	도메인 에이전트 디스커버리 시작 스크립트
SUNWbgod	SPA Platform Discovery	플랫폼 에이전트 디스커버리 디먼
SUNWbgodr	SPA Platform Discovery (루트)	플랫폼 에이전트 디스커버리 시작 스크립트
SUNWbgpji	SPA Sun Fire B100 Domain Personality Module	B100 도메인 계측
SUNWbgpjo	SPA Sun Fire B1600용 Platform Personality Module	B1600 플랫폼 계측

참고 – 패키지 간에 내부 종속성이 존재하며 특정 순서로 설치해야 합니다(77 페이지의 “SNMP 소프트웨어 설치”를 참조하십시오).

소프트웨어 업그레이드

소프트웨어를 업그레이드하려면 새 버전을 다시 설치하기 전에 기존 소프트웨어를 제거해야 합니다(13 장을 참조하십시오).

패키지 인도

패키지는 tar 아카이브 번들로 제공됩니다. 표 9-2는 `SUNWspa.1.0.tar.Z` 아카이브 번들의 내용을 보여주는데, Sun Fire B1600의 플랫폼 관리 및 Sun Fire B100의 도메인 관리를 위한 SNMP 지원을 제공하는 패키지를 포함합니다.

압축을 풀 때, 패키지는 표에 표시된 디렉토리에 위치됩니다. 소프트웨어 설치에 대한 자세한 지침은 10 장을 참조하십시오.

참고 – 이 파일의 최신 버전이 있는지 확인하십시오.

표 9-2 SNMP Management Agent 패키지 번들

번들	설명	내용
SUNspa.1.0.22.tar.Z	플랫폼 에이전트 서버에 설치될 SNMP 플랫폼 에이전트 패키지	platform/proxy/SUNWbgpc platform/proxy/SUNWbgpk platform/proxy/SUNWbgcm platform/proxy/SUNWbgcmr platform/proxy/SUNWbgod platform/proxy/SUNWbgodr platform/proxy/SUNWbgpjo platform/proxy/SUNWbgpm platform/proxy/SUNWbgpmr platform/proxy/SUNWjdrt platform/proxy/SUNWjsnmp
	Sun Fire B100 블레이드에 설치될 SNMP 계측 패키지	platform/target/SUNWbgpc platform/target/SUNWbgptk platform/target/SUNWbgpr platform/target/SUNWbgcm
	Sun Fire B100 블레이드에 설치될 SNMP 도메인 에이전트 패키지	domain/SUNWbgpc domain/SUNWbgptk domain/SUNWbgcm domain/SUNWbgcmr domain/SUNWbgidr domain/SUNWbgpji domain/SUNWbgpm domain/SUNWbgpmr domain/SUNWjdrt domain/SUNWjsnmp

tar 파일의 압축을 풀려면 다음을 입력하십시오.

```
$ zcat SUNWspa.1.0.tar.Z | tar xf -
```

Sun Fire B100에 도메인 또는 대상 패키지 설치

도메인(도메인 하드웨어 모니터링의 경우) 또는 대상(플랫폼 하드웨어 모니터링의 경우) 패키지는 Sun Fire B100 블레이드에 설치되어야 합니다. 다음을 포함하여, 이를 달성할 수 있는 여러 가지 방법이 있습니다.

- 각 개별 Sun Fire B100 블레이드에 도메인 또는 대상 패키지 압축 풀기
- 소프트웨어가 설치될 각 Sun Fire B100 블레이드의 루트 사용자가 볼 수 있는 공유 디렉토리에 도메인 또는 대상 패키지 압축 풀기
- 네트워크 설치를 위한 도메인 또는 대상 패키지 설정

시스템 파일에 대한 효과

표 9-3에 표시된 것처럼, /etc/rc<n>.d에 대한 링크를 갖는 여러 가지 새로운 시작 파일이 /etc/init.d에 작성됩니다.

표 9-3 시작 스크립트

구성요소	시작 스크립트	패키지 이름	패키지 설명
Platform Object Manager	spapom	SUNWbgcmr	플랫폼 개체 관리자(루트)
Domain Hardware Discovery Module	spaibdm*	SUNWbgidr	도메인 하드웨어 디스커버리 모듈(루트)
Remote Data Plug-ins	spardp†	SUNWbgpr	퍼스널리티 모듈(루트)
SNMP Protocol Mediator/Master Agent	spama	SUNWbgpmr	SNMP 프로토콜 조정자 SNMP 마스터 에이전트(루트)

* 도메인 패키지만을 갖는 Sun Fire B100에 있음.

† 플랫폼/대상 패키지만을 갖는 Sun Fire B100에 있음.

디스커버리 모듈은 inetd에 의해 자동으로 시작되며 그것에 의하여 새 항목이 /etc/inetd.conf 파일에 작성됩니다.

POM(Platform Object Manager)은 IP 포트에서 클라이언트의 요청에 대해 활동을 모니터합니다. 이들 포트는 /etc/services 파일에 등록됩니다.

10 장

설치

이 장에서는 Sun Fire B1600에 관리 소프트웨어를 설치하는 방법에 대해 설명합니다.

이 장에는 다음 절이 들어있습니다.

설치 선택

설치할 소프트웨어의 구성을 결정할 때 고려해야 하는 두 가지 주요 고려사항이 있습니다.

1. 계측 구성
2. 관리 인터페이스 구성

계측 구성

플랫폼 유형에 따라서 다음을 채택할 수 있습니다.

- 모니터되는 시스템에서 실행하는 도메인 에이전트

소프트웨어는 모니터될 Sun Fire B100 블레이드(도메인)에 로컬로 설치됩니다. 각 블레이드는 개별적으로 모니터되고 한 번에 한 블레이드만을 볼 수 있습니다. 이것은 *도메인 하드웨어 모니터링*이라고 합니다.

■ 시스템 컨트롤러를 통해 프록시되는 플랫폼 에이전트

소프트웨어는 Sun Fire B1600 시스템 컨트롤러를 사용하여 플랫폼 지침에 액세스하는 원격(플랫폼 에이전트) 서버와, 모니터될 Sun Fire B100(대상) 블레이드에 설치됩니다. 이 소프트웨어를 사용하여 전원 공급장치, 시스템 컨트롤러 및 모든 블레이드를 포함하여 시스템 컨트롤러가 관리하는 모든 하드웨어를 모니터할 수 있습니다. 이것은 *플랫폼 하드웨어 모니터링*이라고 합니다.

플랫폼 하드웨어 모니터링을 사용하려는 경우, 하드 디스크 드라이브, CPU 및 이더넷 Mac 주소에 관한 정보를 얻으려면 모니터되는 각 Sun Fire B100 블레이드에 대상 패키지를 설치해야 합니다.

참고 – Sun Fire B100 블레이드는 도메인 하드웨어 모니터링에서는 *도메인*으로, 플랫폼 하드웨어 모니터링에서는 *대상*으로 참조됩니다.

---

66 페이지의 “계측”도 참조하십시오.

관리 인터페이스 구성

관리 소프트웨어는 많은 방법으로 전개되도록 설계되었으며, 다음이 지원됩니다.

■ 마스터 에이전트 없이 `snmpdx`를 사용한 SNMP(이것이 기본 설치입니다)

■ 마스터 에이전트 및 `snmpdx`를 사용한 SNMP

이 경우에, 11 장 및 12 장에서 설명하는 것처럼 수동으로 설치를 구성해야 합니다.

SNMP 소프트웨어 설치

이 절은 모니터링 소프트웨어 설치를 위한 절차를 요약합니다. 각 유형의 설치에 대한 상세한 프로세스가 이 장의 후속 절에 나옵니다.

소프트웨어를 설치하기 전에 다음을 확인하십시오.

- 도메인 또는 대상(Sun Fire B100) 및 플랫폼 에이전트 서버 모두에 Solaris의 필수 레벨이 설치되어 있습니다(67 페이지의 "운영 환경"을 참조하십시오).
- 모든 필수 패치(67 페이지의 "패치" 참조) 및 SNMP 소프트웨어의 일부로서 제공되지 않는 모든 추가 필수 패키지(68 페이지의 "Java 환경" 참조)를 설치했습니다.
- 기존 J2SE를 업그레이드하거나 68 페이지의 "Java 환경"에 설명하는 대로 별도의 J2SE를 설치하여 Java 1.4를 설치했습니다.

시스템이 이러한 모든 요구사항을 만족한다고 확신할 때, SNMP 소프트웨어 설치를 계속할 수 있습니다.

이제 도메인 하드웨어 모니터링 또는 플랫폼 하드웨어 모니터링을 사용 중인지 여부를 결정해야 합니다.

- 도메인 하드웨어 모니터링을 선택하는 경우, 77 페이지의 "도메인 하드웨어 모니터링을 위한 소프트웨어 설치"에 설명된 절차를 따르십시오.
- 플랫폼 하드웨어 모니터링을 선택하는 경우, 78 페이지의 "플랫폼 하드웨어 모니터링을 위한 소프트웨어 설치"에 설명된 절차를 따르십시오.

도메인 하드웨어 모니터링을 위한 소프트웨어 설치

모니터되는 각 블레이드에 이들 패키지를 설치하십시오(72 페이지의 "Sun Fire B100에 도메인 또는 대상 패키지 설치"을 참조하십시오).

▼ 소프트웨어 설치

1. **Java 1.4를 설치했는지 확인합니다.**
   68 페이지의 "Java 환경"을 참조하십시오.
2. **SUNWjsnmp의 모든 기존 버전이 제거되었는지 확인합니다.**
   69 페이지의 "Java SNMP API"를 참조하십시오.

3. **종속성 문제가 보고되는 것을 피하기 위해 표시된 순서로 Sun Fire B100 블레이드에 도메인 에이전트 패키지를 설치합니다.**

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgcmr SUNWbgidr \
SUNWbgpji SUNWjsnmp SUNWjdrt SUNWbgpm SUNWbgpmr
```

4. **Java 환경을 구성합니다.**

   a. **68 페이지의 "Java 환경"에 설명된 대로 J2SE 1.4를 설치한 경우, 이 단계를 무시하십시오.**

   b. **123 페이지의 "J2RE 1.4 설치"에 설명한 대로 J2RE 1.4를 설치한 경우, 125 페이지의 "도메인 하드웨어 모니터링"에 설명한 대로 도메인 하드웨어 모니터링 시작 스크립트를 편집하십시오.**

5. **소프트웨어를 구성합니다.**

   11 장을 참조하십시오.

6. **Sun Fire B100 블레이드를 재부트합니다.**

   이제 도메인 하드웨어 모니터링을 위한 소프트웨어 설치를 완료했습니다. 84 페이지의 "인터페이스 옵션"에서 계속하십시오.

7. **다음을 입력하여 프로세스가 올바르게 시작했는지 확인합니다.**

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
#
# ps -ef | grep spa.wbem

    root   278     1  0   Feb 24 ?       44:19 /usr/j2se/bin/java
    -Dcom.sun.spa.wbem.pomi.port=3333 -Xms64m -Xmx768m -Dcom.sun
#
```

출력이 위와 비슷한 경우 프로세스가 실행 중입니다.

플랫폼 하드웨어 모니터링을 위한 소프트웨어 설치

- Sun Fire B100 블레이드에 Java 1.4를 설치할 계획인지 여부를 결정하십시오(68 페이지의 "Java 환경"의 설명을 참조하십시오). 대상 계측을 지원하려는 경우 Java 1.4 설치가 필수입니다.
  - 대상 계측을 갖는 소프트웨어를 설치하려면, 79 페이지의 "대상 계측과 함께 소프트웨어 설치"에 있는 설치 프로세스를 시작하십시오.

- 대상 계측 없이 소프트웨어를 설치하려면, 81 페이지의 “대상 계측 없이 소프트웨어 설치”에 있는 설치 프로세스를 시작하십시오.
- 플랫폼 에이전트 서버에 플랫폼 에이전트 패키지를 설치하십시오.
- 필요한 경우 모니터되는 각 블레이드에 대상 플랫폼 에이전트 패키지를 설치하십시오.
- 시스템 컨트롤러 SMS IP 주소를 설정하십시오.

▼ 대상 계측과 함께 소프트웨어 설치

1. **플랫폼 에이전트로 작용하는 서버에 Java 1.4를 설치했는지 확인합니다.**

   68 페이지의 “Java 환경” 및 4 단계를 참조하십시오.

2. **SUNWjsnmp의 모든 기존 버전이 플랫폼 에이전트 서버에서 제거되었는지 확인하십시오.**

   69 페이지의 “Java SNMP API”를 참조하십시오.

3. **종속성 문제점을 피하기 위해 표시된 순서로 플랫폼 에이전트 서버에 플랫폼 에이전트 패키지를 설치합니다.**

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgcmr SUNWbgod \
SUNWbgodr SUNWbgpjo SUNWjsnmp SUNWjdrt SUNWbgpm SUNWbgpmr
```

4. **Java 환경을 구성합니다.**

   a. **68 페이지의 “Java 환경”에 설명된 대로 J2SE 1.4를 설치한 경우, 이 단계를 무시하십시오.**

   b. **123 페이지의 “J2RE 1.4 설치”에 설명한 대로 J2RE 1.4를 설치한 경우, 126 페이지의 “플랫폼 하드웨어 모니터링”에 설명한 대로 플랫폼 하드웨어 모니터링 시작 스크립트를 편집하십시오.**

5. **소프트웨어를 구성합니다.**

   11 장을 참조하십시오.

6. **다음을 입력하여 수동으로 플랫폼 에이전트를 시작합니다.**

```
# /etc/init.d/spapom start
# /etc/init.d/init.snmpdx stop
# /etc/init.d/spama stop
# /etc/init.snmpdx start
# pkill -1 inetd
```

   또는 플랫폼 에이전트 서버를 재부트하여 시작하십시오.

7. **다음을 입력하여 프로세스가 올바르게 시작했는지 확인합니다.**

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
#
# ps -ef | grep spa.wbem

    root   278     1  0   Feb 24 ?       44:19 /usr/j2se/bin/java
    -Dcom.sun.spa.wbem.pomi.port=3333 -Xms64m -Xmx768m -Dcom.sun
#
# netstat -a | grep mismi
     *.mismi            *.*             0     0 24576     0 LISTEN
      *.mismi                            *.*  0
0 24576      0 LISTEN
#
```

출력이 위와 비슷한 경우 프로세스가 실행 중입니다.

8. **모니터될 대상 Sun Fire B100 블레이드에 Java 1.4를 설치했는지 확인합니다.**

68 페이지의 “Java 환경” 및 11 단계를 참조하십시오.

9. **SUNWjsnmp의 모든 기존 버전이 대상 블레이드에서 제거되었는지 확인하십시오.**

69 페이지의 “Java SNMP API”를 참조하십시오.

10. **대상 블레이드에 대상 플랫폼 에이전트 패키지를 설치합니다.**

이들 패키지는 모니터되는 시스템의 Solaris 인터페이스를 사용하여 계측에 대한 액세스를 가능하게 합니다.

종속성 문제점을 피하기 위해 표시된 순서로 이들 패킨지를 설치하십시오.

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgpr
```

11. **Java 환경을 구성합니다.**

    a. **68 페이지의 “Java 환경”에 설명된 대로 J2SE 1.4를 설치한 경우, 이 단계를 무시하십시오.**

    b. **123 페이지의 “J2RE 1.4 설치”에 설명한 대로 J2RE 1.4를 설치한 경우, 126 페이지의 “플랫폼 하드웨어 모니터링”에 설명한 대로 대상 하드웨어 모니터링 시작 스크립트를 편집하십시오.**

12. **다음을 입력하여 수동으로 대상 계측을 시작합니다.**

```
# /etc/init.d/spardp start
```

또는 시스템을 재부트하여 시작하십시오.

13. **다음을 입력하여 프로세스가 올바르게 시작했는지 확인합니다.**

```
# netstat -an | grep 1099
      *.1099         *.*          0      0 24576      0 LISTEN
```

출력이 위와 비슷한 경우 프로세스가 실행 중입니다.

14. **setupsc를 사용하여 SMS IP 주소를 설정합니다.**

82 페이지의 “시스템 컨트롤러 구성”에서 계속하십시오.

▼ 대상 계측 없이 소프트웨어 설치

1. **플랫폼 에이전트로 작용하는 서버에 Java 1.4를 설치했는지 확인합니다.**

68 페이지의 “Java 환경”을 참조하십시오.

2. **SUNWjsnmp의 모든 기존 버전이 플랫폼 에이전트 서버에서 제거되었는지 확인하십시오.**

69 페이지의 “Java SNMP API”를 참조하십시오.

3. **종속성 문제점을 피하기 위해 표시된 순서로 플랫폼 에이전트 서버에 플랫폼 에이전트 패키지를 설치합니다.**

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgcmr SUNWbgod \
SUNWbgodr SUNWbgpjo SUNWjsnmp SUNWjdrt SUNWbgpm SUNWbgpmr
```

4. **Java 환경을 구성합니다.**

   a. **68 페이지의 “Java 환경”에 설명된 대로 J2SE 1.4를 설치한 경우, 이 단계를 무시하십시오.**

   b. **123 페이지의 “J2RE 1.4 설치”에 설명한 대로 J2RE 1.4를 설치한 경우, 126 페이지의 “플랫폼 하드웨어 모니터링”에 설명한 대로 플랫폼 하드웨어 모니터링 시작 스크립트를 편집하십시오.**

5. **소프트웨어를 구성합니다.**

11 장을 참조하십시오.

6. **다음을 입력하여 수동으로 플랫폼 에이전트를 시작합니다.**

```
# /etc/init.d/spapom start
# /etc/init.d/init.snmpdx stop
# /etc/init.d/spama stop
# /etc/init.d/snmpdx start
# pkill -1 inetd
```

또는 플랫폼 에이전트 서버를 재부트하여 시작하십시오.

7. **다음을 입력하여 프로세스가 올바르게 시작했는지 확인합니다.**

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
#
# ps -ef | grep spa.wbem

    root   278     1  0   Feb 24 ?       44:19 /usr/j2se/bin/java
    -Dcom.sun.spa.wbem.pomi.port=3333 -Xms64m -Xmx768m -Dcom.sun
#
# netstat -a | grep mismi
     *.mismi           *.*             0     0 24576     0 LISTEN
      *.mismi                          *.*  0
0 24576      0 LISTEN
#
```

출력이 위와 비슷한 경우 프로세스가 실행 중입니다.

8. **`setupsc`를 사용하여 SMS IP 주소를 설정합니다.**

아래에서 계속하십시오.

시스템 컨트롤러 구성

SNMP 소프트웨어를 설치한 후, 시스템 컨트롤러의 SMS IP 주소를 플랫폼 에이전트 서버의 해당 주소로 설정해야 합니다. 이렇게 하려면, 시스템 컨트롤러의 콘솔에 로그온하고, `setupsc`를 실행하고, 플랫폼 에이전트 서버의 IP 주소를 추가하십시오.

아래 예에서는, IP 주소가 `10.5.1.1`로 설정됩니다.

다음 행이 표시될 때까지 현재 값을 승인하려면 각 질문 뒤에 [ENTER]를 누르십시오.

```
Enter the SMS IP address
```

IP 주소를 입력하고 [ENTER]를 누른 후, 나머지 질문에 대한 응답으로 계속 [ENTER]를 누르십시오.

참고 – `setupsc` 명령은 *Sun Fire B1600 Blade System Chassis Software Setup Guide*에 설명되어 있습니다.

코드 예 10-1 SMS IP 주소 설정

```
hornet-sc>setupsc
Entering Interactive setup mode.
Use Ctrl-z to exit & save. Use Ctrl-c to abort.
Do you want to configure the enabled interfaces [y]?
Should the SC network interface be enabled [y]?
Should the SC telnet interface be enabled for new connections [y]?
Do you want to configure the network interface [y]?
Should the SC use DHCP to obtain its network configuration [n]?
Enter the SC IP address [129.156.174.140]:
Enter the SC IP netmask [255.255.255.0]:
Enter the SC IP gateway [129.156.174.1]:
Do you want to configure the SC private addresses [y]?
Enter the SSC0/SC IP private address [129.156.174.118]:
Enter the SSC1/SC IP private address [129.156.174.128]:
Do you want to enable a VLAN for the SC [n]?
Enter the SMS IP address [0.0.0.0]: 10.5.1.1
<truncated>

hornet-sc>
```

인터페이스 옵션

기본 설치는 snmpdx의 서브에이전트로서 작용하는 SNMP를 통해 관리를 제공합니다. 구성 후에 전개를 사용자 정의할 수 있지만, 설치 중에는 사용자 입력이 필요없습니다.

구성 파일을 편집하여 SNMP 및 snmpdx에 마스터 에이전트 기능을 추가할 수 있습니다.

참고 – 모든 경우에, SNMP 액세스 제어 목록(ACL)은 액세스를 막는 기본 구성을 갖습니다. 이들을 액세스가 사용 가능하도록 구성해야 합니다(11 장을 참조하십시오).

---

snmpdx를 사용하는 SNMP(기본)

이 옵션은 조정자에 대한 요청이 경로 지정되는 자동 할당되는 UDP 포트 번호를 사용하여 SNMP 조정자를 snmpdx의 서브에이전트로 등록합니다. 이들 요청은 snmpdx를 통하거나, 조정자 ACL 파일에서 사용 가능한 경우 그림 10-1에서 점선으로 표시된 것처럼 직접 전달될 수 있습니다(12 장을 참조하십시오).

표 10-1 그림 10-1에 대한 포트 요약

키	기능	spama.conf의 매개변수	기본값
1	snmpdx가 SNMP 조정자로 요청을 전송하는 포트	SPAPM_REQ_PORT	
2	조정자가 SNMP 관리자로 트랩을 전송하는 포트	SPAPM_TRAP_PORT	162

참고 – 기본 구성이 자동이지만, 그래도 관리자 구성을 지원하도록 snmpdx 및/또는 조정자에 대한 ACL 파일을 구성해야 합니다.

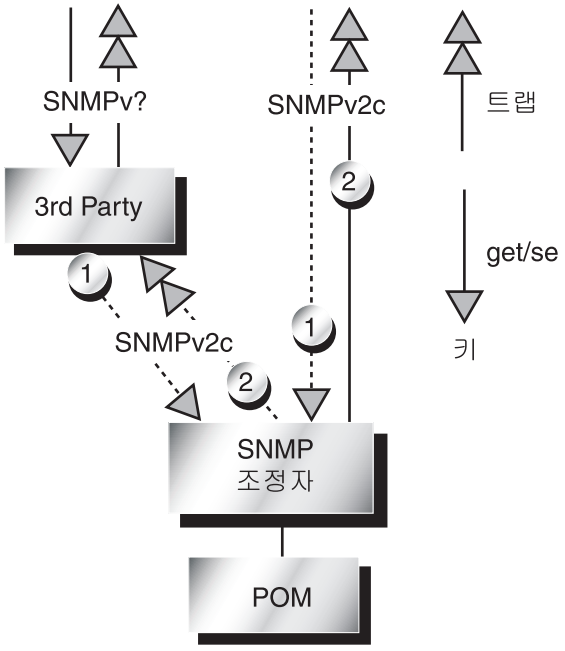

그림 10-1 SNMP가 snmpdx의 서브에이전트일 때의 데이터 흐름

SNMP 및 마스터 에이전트와 snmpdx

이 옵션은 SNMP 및 snmpdx에 마스터 AgentSNMPv3 보안 기능을 추가합니다. snmpdx 자동 시작은 사용 불가능하며 마스터 에이전트가 포트 161에 등록됩니다. Snmpdx에 새 포트 번호가 자동으로 지정됩니다.

조정자의 트랩은 선택적으로 마스터 에이전트에 의해 SNMPv3로 변환되거나 직접 전송됩니다.

snmpdx의 트랩은 단지 SNMP 관리자로 직접 전송되며 마스터 에이전트에 의해 변환될 수 없습니다.

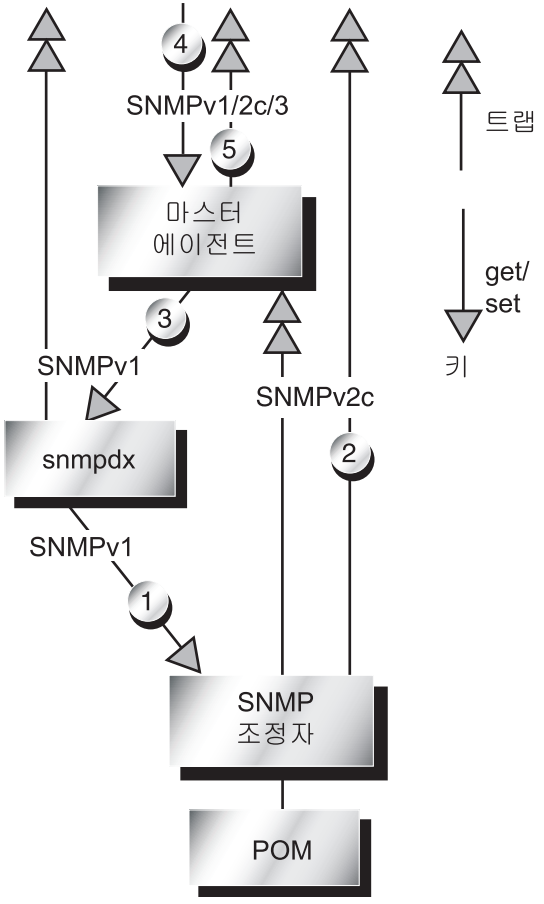

그림 10-2 마스터 에이전트가 채택될 때의 데이터 흐름

표 10-2 그림 10-2에 대한 포트 요약

키	기능	spama.conf의 매개변수	기본값
1	snmpdx가 SNMP 조정자 서브에이전트로 요청을 전송하는 포트	SPAPM_REQ_PORT	
2	조정자가 직접 SNMP 관리자로 트랩을 전송하는 포트	SPAPM_TRAP_PORT	162
3	마스터 에이전트가 요청을 snmpdx로 전달하는 포트	SNMPDX_REQ_FORWARD_PORT	
4	마스터 에이전트가 요청에 대해 모니터하는 포트	MASTER_AGENT_REQ_PORT	161
5	마스터 에이전트가 SNMP 관리자로 트랩을 전송하는 포트	SPAPM_TRAP_PORT	162

타사의 마스터 에이전트 및 SNMP

이 옵션은 수동으로 또는 타사 마스터 에이전트에 의해 할당되는 포트 번호를 사용하는 서브에이전트로 등록합니다. 직접 액세스를 가능하게 하려면, 12 장에서 설명하는 것처럼 수동으로 조정자 ACL 파일을 구성해야 합니다.

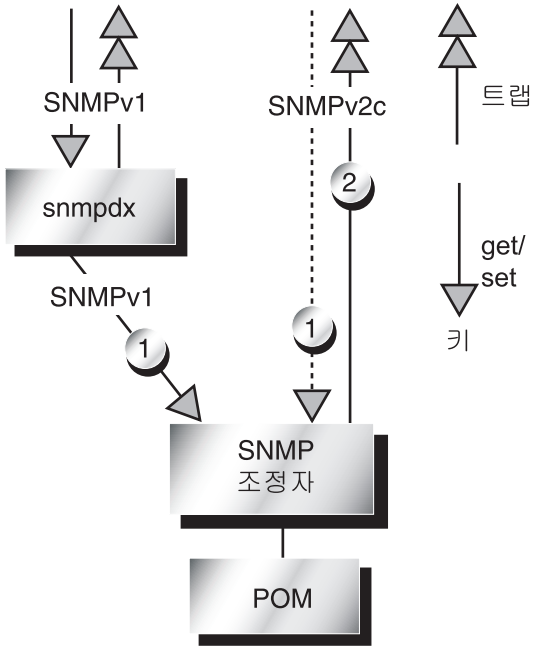

그림 10-3 타사 마스터 에이전트가 채택될 때의 데이터 흐름

표 10-3 그림 10-3에 대한 포트 요약

키	기능	spama.conf의 매개변수	기본값
1	직접 액세스를 사용 가능하게 하는 SNMP 조정자가 사용하는 포트 또는 타사 마스터 에이전트가 요청을 SNMP 조정자로 전달하는 포트	SPAPM_REQ_PORT	
2	SNMPv2c 트랩을 타사 마스터 에이전트로 전송하기 위해 또는 직접 SNMP 관리자에게 전송하기 위해 SNMP 조정자가 사용하는 포트	SPAPM_TRAP_PORT	

11 장

구성 파일

이 장에서는 소프트웨어를 구성하기 위해 편집할 수 있는 파일의 개요를 제공합니다. 구성 가능한 매개변수를 나열하고 액세스 제어의 개념을 소개합니다.

여기에서 설명하는 파일을 사용하여 기본 SNMP 옵션을 구성하는 방법을 설명하는 12 장을 참조하기 전에 이 장을 읽으십시오.

이 장에는 다음 절이 들어있습니다.

참고 – SNMP 패키지 및 이들을 설치하는 방법에 대한 정보는 9 장 및 10 장을 참조하십시오.

구성 파일

/etc/opt/SUNWspa/에 위치한 다음 파일들이 SNMP의 구성을 결정합니다.

- **일반 구성 파일**
  - spama.conf—마스터 에이전트 및 조정자가 구성되는 방법을 정의합니다.
- **조정자 구성 파일**
  - spapm.acl—조정자에 대한 액세스 제어를 정의합니다.
  - spapm_snmpdx.acl—snmpdx의 서브에이전트로서의 조정자에 대한 액세스 제어를 정의합니다.
- **마스터 에이전트 구성 파일**

마스터 에이전트를 사용하지 않는 경우, 이들 파일을 구성할 필요가 없습니다.

- spama.acl—마스터 에이전트에 대한 액세스 제어를 정의합니다.
- spama.uacl—마스터 에이전트에 대한 SNMPv3 사용자 및 문맥 액세스 제어를 정의합니다.
- spama.security—spama.uacl에서 참조되는 SNMPv3 사용자를 정의합니다.

이들 파일은 아래 절에서 자세히 설명됩니다.

일반 구성 파일

spama.conf

spama.conf 파일에는 많은 구성 가능한 매개변수가 들어있는데, 아래 절과 표 11-1에서 설명합니다. spama.conf 파일의 예가 코드 예 11-1에 표시됩니다.

일반 옵션

START_MEDIATOR

조정자를 실행하려면 이 매개변수를 yes로 설정하고, 그렇지 않으면 no로 설정하십시오(그림 10-1도 참조하십시오).

기본값은 다음과 같습니다.

```
START_MEDIATOR=yes
```

START_MASTER_AGENT

SNMPv3 보안을 사용하기 원하고 이 보안을 제공하기 위해 타사 마스터 에이전트를 사용 중이 아닌 경우, 마스터 에이전트를 시작하려면 이 매개변수를 yes로 설정하십시오(그림 10-2 및 동반되는 표를 참조하십시오).

SNMPv3 보안이 필요없고 다른 마스터 에이전트(snmpdx 포함)를 사용 중이거나, 어떤 마스터 에이전트도 사용하지 않으려는 경우, 이 매개변수를 no로 설정하십시오(그림 10-1과 그림 10-3 및 동반되는 표를 참조하십시오).

기본값은 다음과 같습니다.

```
START_MASTER_AGENT=no
```

AGENT_INTERFACE_NAME

마스터 에이전트가 사용 가능한 경우(위 참조), 여기에 설정되는 값은 마스터 에이전트가 바인드할 네트워크 인터페이스, 조정자가 localhost에 바인드될 프로토콜을 지정합니다.

마스터 에이전트가 사용 가능하지 않은 경우, 여기에 설정되는 값은 조정자가 바인드하는 네트워크 인터페이스의 호스트 이름을 정의합니다. 값을 지정하지 않으면 기본값은 액세스가 기본 인터페이스를 사용하여 이루어지는 것입니다.

조정자를 서브에이전트(예: snmpdx)로서 사용하려는 경우, 값을 localhost로 설정하십시오.

기본값은 다음과 같습니다.

```
AGENT_INTERFACE_NAME=localhost
```

마스터 에이전트 옵션

마스터 에이전트가 snmpdx와 함께 작동할 수 있기 위해서, 마스터 에이전트 시작 스크립트가 snmpdx를 중지하고, 그의 SNMP 포트를 인수하고 다른 포트에서 snmpdx를 서브에이전트로서 다시 시작합니다.

SNMPDX_REQ_FORWARD_PORT 매개변수가 null 값을 갖는 경우, 마스터 에이전트 시작 스크립트가 수명이 짧은 익명 범위 32768 - 65535에서 사용 가능한 포트를 검색하고 해당 포트에서 snmpdx를 서브에이전트로서 다시 시작합니다. 시작 스크립트는 또한 /etc/services를 검색하고 그곳에 나열된 어떤 포트도 사용하지 않습니다.

그러나, SNMPDX_REQ_FORWARD_PORT 값을 지정하면 마스터 에이전트는 이 포트를 사용하여 요청을 snmpdx에 전달하며, 이 경우 마스터 에이전트는 해당 포트가 이미 사용 중인지 확인하지 않습니다.

MASTER_AGENT_REQ_PORT

이것은 마스터 에이전트가 관리자로부터 요청을 수신하는 포트입니다. 대부분의 구성에서는 값을 변경할 필요가 없습니다(그림 10-2 및 동반되는 표도 참조하십시오).

지정되지 않는 경우 기본값은 161입니다.

ENABLE_SNMPV2C_SETS

이 매개변수는 마스터 에이전트가 SNMPv1 또는 SNMPv2c를 사용하여 설정 작업이 수행되도록 허용하는지 여부를 제어합니다. 값을 yes로 설정하면 SNMPv1 및 SNMPv2C 프로토콜이 본질적으로 보안되지 않기 때문에 실질적으로 보안이 축소됩니다.

기본값은 다음과 같습니다.

```
ENABLE_SNMPV2C_SETS=no
```

SNMPDX_REQ_FORWARD_PORT

이 매개변수는 마스터 에이전트가 snmpdx로 요청을 전달하는 포트를 제어합니다. 값을 지정하지 않으면 마스터 에이전트가 자동 구성을 수행합니다(이 절에 대한 소개, 그림 10-2 및 동반되는 표를 참조하십시오).

값을 지정하는 경우, 이 포트에서 청취하도록 수동으로 snmpdx도 구성해야 합니다.

기본값은 다음과 같습니다.

```
SNMPDX_REQ_FORWARD_PORT
```

SNMPV3_USER

이 매개변수는 SNMPv3 트랩을 발행하는 SNMPv3 사용자를 판별합니다.

기본값은 다음과 같습니다.

```
SNMPV3_USER=defaultUser
```

참고 – SNMPv3 트랩을 전송하려면, SPAPM_TRAPS_ARE_V3=yes를 설정해야 합니다.

프로토콜 조정자 옵션

SUB_AGENT

이 매개변수는 조정자 또는 마스터 에이전트가 시작 시에 자동으로 시작되는 대신 snmpdx 같은 마스터 에이전트에 의해 시작되는지 여부를 판별합니다.

기본값은 다음과 같습니다.

```
SUB_AGENT=yes
```

yes를 지정하는 경우, 다음과 같이 <port> 매개변수를 전달하여 조정자를 시작해야 합니다.

```
# /etc/init.d/spama start <port>
```

여기서 <port>는 조정자가 SNMP 요청을 청취할 UDP 포트입니다. 예를 들어, snmpdx와 함께 사용될 때 조정자의 호출은 /etc/snmp/conf/spapm.rsrc 파일의 다음 행에 의해 제어됩니다.

```
command = "etc/init.d/spama start $PORT"
```

참고 – SUB-AGENT=YES인 경우, START_MASTER_AGENT의 값이 무시됩니다. 마스터 에이전트 기능을 사용 가능하게 하는 경우, SUB_AGENT를 no로 설정해야 합니다.

---

no를 지정하는 경우, 조정자가 시작 시에 실행되고(시작 스크립트 /etc/rc3.d/S80spama에 의해), 조정자가 SNMP 요청을 청취하는 UDP 포트는 아래에서 설명하는 SPAPM_REQ_PORT 설정에 의해 정의됩니다.

SPAPM_REQ_PORT

이 매개변수는 SUB_AGENT가 no로 설정될 때 조정자가 요청을 수신하는 포트를 판별합니다. 그림 10-1, 그림 10-2 및 그림 10-3과 동반되는 표를 참조하십시오.

기본값은 다음과 같습니다.

```
SPAPM_REQ_PORT=
```

기본 설정과 함께, START_MASTER_AGENT=yes인 경우 포트 번호는 자동으로 할당됩니다. START_MASTER_AGENT=no인 경우, 기본 포트 번호 33000이 사용됩니다.

START_MASTER_AGENT=no인 경우, 조정자가 고정 포트를 사용하는 로컬 마스터 에이전트에 의해 또는 원격 SNMP 관리자에 의해 직접 액세스될 수 있도록 SPAPM_REQ_PORT를 필수 값으로 설정합니다.

SPAPM_TRAPS_ARE_V3

이 매개변수는 조정자 트랩이 SNMPv3 또는 SNMPv2c인지 여부를 판별합니다.

트랩을 SNMPv2c로 설정하는 기본값은 다음과 같습니다.

```
SPAPM_TRAPS_ARE_V3=no
```

참고 – SNMPv3 트랩을 사용 가능하게 하는 경우(SPAPM_TRAPS_ARE_V3=yes), START_MASTER_AGENT=yes 및 START_MEDIATOR=yes도 설정해야 합니다.

SPAPM_TRAP_PORT

이 매개변수는 조정자 트랩이 전송되는 포트 번호를 판별합니다. 그림 10-1, 그림 10-2 및 그림 10-3과 동반되는 표를 참조하십시오.

기본값은 다음과 같습니다.

```
SPAPM_TRAP_PORT=162
```

SPAPM_TRAP_INTERFACE

이 매개변수는 조정자가 SNMP 트랩을 전송하는 인터페이스를 판별합니다. 이것이 정의되지 않는 경우, 트랩은 호스트의 기본 인터페이스에서 발행됩니다.

참고 – SNMPv2c 트랩은 snmpdx를 통하는 대신 직접 전송됩니다.

기본값은 정의되지 않습니다.

```
SPAPM_TRAP_INTERFACE=
```

SPAPM_OPTIONS

이 매개변수를 사용하여 다음 옵션 중 하나 이상을 지정함으로써 조정자의 작동을 수정할 수 있습니다.

- -a 속성 변경 통지 전송
- -s 상태 변경 통지 전송

- -c 개체 작성 통지 전송
- -C 초기화 중에 개체 작성 통지 사용 가능
- -d 개체 삭제 통지 전송
- -l 기본적으로 현재 문제점 목록 사용 가능

이 매개변수에 대한 형식과 기본값은 다음과 같습니다.

```
SPAPM_OPTIONS="-ascCdl"
```

표 11-1 spama.conf의 기본값

매개변수	기본 설치 후의 값
START_MEDIATOR	START_MEDIATOR=yes
START_MASTER_AGENT	START_MASTER_AGENT=no
AGENT_INTERFACE_NAME	AGENT_INTERFACE_NAME=localhost
MASTER_AGENT_REQ_PORT	MASTER_AGENT_REQ_PORT=161 (지정되지 않는 경우에도 이 값으로 기본 설정됩니다)
ENABLE_SNMPV2C_SETS	ENABLE_SNMPV2C_SETS=no
SNMPDX_REQ_FORWARD_PORT	SNMPDX_REQ_FORWARD_PORT=
SNMPV3_USER	SNMPV3_USER=defaultUser
SUB_AGENT	SUB_AGENT=yes
SPAPM_REQ_PORT	SPAPM_REQ_PORT=
SPAPM_TRAPS_ARE_V3	SPAPM_TRAPS_ARE_V3=no
SPAPM_TRAP_PORT	SPAPM_TRAP_PORT=162
SPAPM_TRAP_INTERFACE	SPAPM_TRAP_INTERFACE=
SPAPM_OPTIONS	SPAPM_OPTIONS=î-ascCdlî

코드 예 11-1 spama.conf 파일의 예

```
#!/sbin/sh
#
#ident  "@(#)spama.conf1.17 01/29/03 SMI"
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.

#
# This file is used to control the configuration of the Master Agent and
# Protocol Mediator
#
```

코드 예 11-1 spama.conf 파일의 예 *(계속)*

```
#
# Master Agent / Mediator configuration
#

################################################################################
# General options
################################################################################

#
# Set to "yes" if the mediator component should be started
#
START_MEDIATOR=yes

#
# Set to "yes" to enable the master agent
#
START_MASTER_AGENT=no

#
# Hostname of the network interface for the agent to bind to. If this
# is not specified the agent will be accessible via the default
# interface.
#
# If the mediator is being used as a sub-agent this should be
# set to localhost.
#
# If the master agent is enabled, this setting applies to its interface,
# the protocol mediator being bound to localhost.
AGENT_INTERFACE_NAME=localhost

################################################################################
# Master Agent options
################################################################################

#
# SNMP port for master agent to receive SNMP get/set requests.
#
# This port number will be used to listen for SNMP get/set
# requests.
#
# If this value is blank, default will be 161 if START_MASTER_AGENT=yes
#
MASTER_AGENT_REQ_PORT=

#
# set to "yes" to enable SNMPv1/SNMPv2c SET operations via the master agent.
```

코드 예 11-1 spama.conf 파일의 예 *(계속)*

```
#
ENABLE_SNMPV2C_SETS=no

#
# SNMP sub-agent port to which non-SNMP Protocol Mediator (snmpdx) requests
# will be sent.
#
# If this port setting is blank (default), automatic configuration will be
# performed. The port number for snmpdx will be dynamically determined
# if snmpdx is already using the UDP port where the Master Agent listens
# for SNMP get/set requests (by default UDP port 161) at Master Agent startup.
#
# If this port setting is blank but snmpdx is not using the same port
# as the Master Agent will listen for SNMP get/set requests, the Master
# Agent will use the port number which is being used by snmpdx to forward
# the SNMP set/get requests to snmpdx.
#
# If this port is set, it is expected that the user should perform
# the "listening" port configuration for the sub-agents and the Master
# Agent will use this port number to forward non-SNMP Protocol Mediator
# requests.
#
SNMPDX_REQ_FORWARD_PORT=

#
# SNMPv3 user
#
# The Mediator will use this user to send the V3 traps (if enabled with
# SPAPM_TRAPS_ARE_V3).
#
# If this value is blank, default will be ídefaultUserí.
SNMPV3_USER=

##############################################################################
# Protocol Mediator options
##############################################################################

#
#
# Sub-agent configuration
#
# If the master agent is not being used (i.e. START_MASTER_AGENT=no), then
# setting SUB_AGENT=yes indicates that the mediator should be started with a
# port number argument by snmpdx or a third party master agent. Otherwise, set
# to no if the mediator is to be started with a manually configured port
# number.
#
```

코드 예 11-1 spama.conf 파일의 예 *(계속)*

```
# If START_MASTER_AGENT=yes then this setting is ignored.
#
SUB_AGENT=yes

#
# Mediator request port. If the START_MASTER_AGENT="no" and SUB_AGENT="no", the
# default is 33000, otherwise it is dynamically allocated.
#
SPAPM_REQ_PORT=

#
# set to yes to enable v3 mediator traps (requires START_MASTER_AGENT=yes and
# START_MEDIATOR=yes)
#
SPAPM_TRAPS_ARE_V3=no

#
# Default port for traps
#
SPAPM_TRAP_PORT=162

# This is also used to define the interface to which SNMP traps will be
# sent by the protocol mediator independently of the setting of
# AGENT_INTERFACE_NAME. If not defined, traps will be issued from the
# hostís default interface.
#
SPAPM_TRAP_INTERFACE=
#
# Agent option flags
#
# -a    Send attribute change notifications
# -s    Send state change notifications
# -c    Send object creation notifications
# -C    Enable object creation notifications during initialization
# -d    Send object deletion notifications
# -l    Enable current problem list logs by default
#
SPAPM_OPTIONS="-ascCdl"
```

액세스 제어

액세스 제어는 관리자의 호스트 시스템의 IP 주소와 커뮤니티 및 각 서브에이전트에 지정된 커뮤니티를 바탕으로 합니다. 커뮤니티 및 호스트 시스템에 대한 액세스 권한이 ACL 파일에 정의됩니다.

ACL 파일은 또한 에이전트가 트랩을 전송하는 호스트를 정의합니다. 트랩이 전송될 때, 에이전트는 ACL 파일의 <*trapInterestHostList*>에 나열된 모든 호스트에 트랩을 전송합니다.

다음 세 가지 ACL 파일이 있습니다.

- `spapm.acl`은 관리 응용프로그램에서 직접 또는 보다 일반적으로는 `snmpdx`로부터 조정자에 대한 액세스를 제어합니다. 또한 SNMP 트랩의 필수 수신인을 정의합니다.
- `spapm_snmpdx.acl`은 `snmpdx`의 서브에이전트로서 구성될 때 조정자에 대한 액세스를 제어합니다.
- `spama.acl`은 SNMPv3 마스터 에이전트를 통해 액세스를 제어합니다.

SNMPv3에 대한 액세스 제어, 커뮤니티 트랩 전송 매개변수는 `spama.uacl` 및 `spama.security`에 정의됩니다.

ACL 파일의 형식

ACL 파일에는 커뮤니티 및 관리자 액세스 권리를 정의하는 `acl` 그룹과 트랩 전송을 위한 커뮤니티와 호스트를 정의하는 `trap` 그룹이 들어있습니다. ACL 파일은 또한 행에 있는 첫 번째 문자로서 해시 기호(#)로 표시되는 주석 행을 포함할 수 있습니다.

참고 – `spapm_snmpdx.acl` 파일의 구문에는 확실한 차이가 있습니다. 자세한 내용은 스크립트의 주석을 참조하십시오.

---

acl 그룹

acl 그룹에는 다음 구문을 사용하는 커뮤니티 구성의 하나 이상의 목록이 있습니다.

```
acl = {
    < 목록 1>
    < 목록 2>
    . . .
    < 목록 N>
}
```

이 파일의 acl 그룹은 특정 커뮤니티 및 관리자에 대한 액세스 권한을 지정합니다. 이 그룹은 다음 형식을 갖는 커뮤니티 구성의 목록으로 구성됩니다.

```
{
   communities = < 커뮤니티목록 >
   access = < 액세스권한 >
   managers = < 호스트목록 >
}
```

<*커뮤니티목록*>은 이 액세스 제어가 적용되는 SNMP 커뮤니티 이름의 목록입니다. 이 목록의 커뮤니티 이름은 쉼표로 구분됩니다.

<*액세스권한*>은 managers 항목에 지정되는 시스템에서 실행 중인 모든 관리자에게 부여될 권한을 지정합니다. 두 개의 가능한 값은 다음과 같습니다.

- read-write
- read-only

<*호스트목록*> 항목은 액세스 권한이 부여될 관리자의 호스트 시스템을 지정합니다. <*호스트목록*>은 쉼표로 구분된 호스트 목록으로, 각각을 다음 중 하나로 표현할 수 있습니다.

- 호스트 이름(예: hubble)
- IP 주소(예: 123.456.789.12)
- 서브네트 마스크(예: 123!255!255!255)

참고 – ACL 파일에서 IP 주소와 서브네트 마스크를 구별하기 위해, 서브네트 마스크의 각 정수는 마침표 대신 느낌표(!)로 구분됩니다.

코드 예 11-2 acl 그룹 예

```
acl = {
    {
        communities = public, private
        access = read-only
        managers = rag, tag, bobtail
    }
    {
        communities = tigger
        access = read-write
        managers = brittas
    }
}
```

trap 그룹

trap 그룹은 에이전트가 트랩을 전송할 수 있는 호스트를 지정합니다. 조정자가 SNMP 마스터 에이전트를 사용하여 SNMPv3 트랩을 전송해야 하는 경우가 아니라, 조정자가 SNMPv2 트랩을 전송해야 하는 경우에만 구성이 필요합니다.

이 그룹에는 다음 구문을 사용하는 하나 이상의 트랩 커뮤니티 정의가 있습니다.

```
trap = {
    < 커뮤니티 1>
    < 커뮤니티 2>
   ...
    < 커뮤니티 N>
}
```

각 행은 호스트 세트와 해당 호스트에 전송될 트랩에 있는 SNMP 커뮤니티 문자열 사이의 연관을 정의합니다. 각 trap-community 정의는 다음 형식을 갖습니다.

```
{
   trap-community = < 트랩커뮤니티문자열 >
   hosts = < 트랩관심호스트목록 >
}
```

<트랩커뮤니티문자열> 항목은 SNMP 커뮤니티 문자열을 지정합니다. 이 문자열이 `hosts` 항목에 지정된 호스트에 전송되는 트랩에 포함됩니다.

<트랩관심호스트목록> 항목은 쉼표로 구분된 호스트 목록을 지정합니다. 각 호스트는 다음 예에서 보는 것처럼 이름이나 전체 IP 주소로 식별되어야 합니다.

코드 예 11-3 `trap` 그룹 예

```
trap = {
     {
          trap-community = tigger
          hosts = gandalf, frodo
     }
}
```

조정자 구성 파일

이 절은 다음의 형식에 대해 설명합니다.

102 페이지의 “`spapm.acl` 파일”

103 페이지의 “`spapm_snmpdx.acl` 파일”

`spapm.acl` 파일

`spapm.acl` 파일은 프로토콜 조정자에 대한 액세스 제어를 정의합니다. 파일의 일반 형식이 99 페이지의 “ACL 파일의 형식”에 설명되어 있습니다. 이 절에는 특히 `spapm.acl` 파일과 관련되는 정보가 들어있으며, 파일의 예가 코드 예 11-4에 표시됩니다.

이 파일은 기본적으로 `/etc/opt/SUNWspa`에 위치합니다.

ACL 파일이 존재하는 경우, 해당 파일이 정의하는 액세스 권한이 모든 관리자 또는 그의 SNMP 어댑터를 통해 에이전트에 액세스하는 플랫폼 에이전트 서버에 적용됩니다. 에이전트가 시작될 때 ACL 파일이 존재하지 않는 경우, 모든 관리자에게 SNMP 어댑터를 통해 에이전트에 대한 전체 액세스가 부여되고 트랩은 생성되지 않습니다.

액세스 제어 및 SNMP 어댑터에 대한 트랩을 사용 가능하게 하려면, 임의의 에이전트가 시작될 때 ACL 파일이 존재하는지 확인하십시오. ACL 파일이 보안 관련 정보를 포함하기 때문에, ACL 파일에 `root`만이 읽을 수 있는 제한된 액세스 권한을 지정하십시오.

`acl` 및 `trap` 그룹은 각각 100 페이지의 “`acl` 그룹” 및 101 페이지의 “`trap` 그룹”에 설명된 형식을 따릅니다.

조정자가 마스터 에이전트(예: snmpdx)의 서브에이전트로 등록되는 경우, spapm.acl 파일에서 localhost를 관리자로 지정해야 합니다. 이것이 마스터 에이전트가 전달하는 SNMP 패킷의 출발점이기 때문입니다. snmpdx는 사용될 때 커뮤니티 문자열을 변경하지 않고 전달하므로 이 파일에 나열된 커뮤니티를 spapm_snmpdx.acl 파일에도 지정해야 합니다(코드 예 11-4를 참조하십시오).

코드 예 11-4 spapm.acl 파일 예

```
#
#  @(#)spapm.acl     1.6 03/01/29 SMI
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
#
# Template ACL file for Sun SNMP Management Agent for Sun Fire B1600

acl = {
    {
        communities = public, private
        access = read-only
        managers = rag, tag, bobtail
    }
    {
        communities = tigger
        access = read-write
         managers = brittas
     }
}

trap = {
    {
        trap-community = tigger
        hosts = brittas
    }
}
```

trap 그룹은 SNMPv2c 통지가 전송되는 곳을 정의합니다.

spapm_snmpdx.acl 파일

기본 구성에서, 조정자는 snmpdx의 서브에이전트로서 실행합니다. 이 파일을 수정하여 소스 호스트 이름을 바탕으로 하는 액세스를 사용 가능하게 할 수 있습니다. 커뮤니티와 액세스 권한이 spama.acl 파일에 있는 것과 일치해야 합니다.

이 파일의 acl 그룹은 특정 커뮤니티 및 관리자에 대한 액세스 권한을 지정합니다. 이 그룹은 다음 형식을 갖는 커뮤니티 구성의 목록으로 구성됩니다.

```
#   {
#       communities = <커뮤니티목록>
#      access = <액세스권한>
#       managers = <호스트목록>
#   }
```

- *커뮤니티목록*은 이 액세스 제어가 적용되는 커뮤니티 이름의 쉼표로 구분된 목록입니다.
- *액세스권한*은 *호스트목록*에 이름이 지정되는 관리자에 부여되는 권한을 지정합니다.
- *호스트목록*은 지정된 *액세스권한*이 부여될 호스트 이름의 쉼표로 구분된 목록입니다.

코드 예 11-5의 첫 번째 예에서, 시스템 rag, tag 및 bobtail이 커뮤니티 public 및 private에 대한 read-write 액세스를 위해 구성됩니다. 시스템 brittas는 커뮤니티 tigger에 대한 read-write 액세스를 위해 구성됩니다.

두 번째 예는 snmpdx가 수신하는 SNMP 패킷이 localhost에서 출발한 것으로 표시되는 SNMPv3 마스터 에이전트(spama.conf의 START_MASTER_AGENT=yes)를 사용한 구성에 적용됩니다. 커뮤니티 public 및 private에 대해 읽기 전용 액세스가 구성됩니다. 커뮤니티 tigger에 대해서는 읽기–쓰기 액세스가 구성됩니다. 이들 커뮤니티는 마스터 에이전트에 의해 SNMPv3 문맥으로부터 맵핑됩니다. 그러므로 이 액세스가 필요한 모든 SNMPv3 문맥이 이 파일에 지정된 대응하는 커뮤니티를 가져야 합니다.

코드 예 11-5 spapm_snmpdx.acl 파일 예

```
#  @(#)spapm_snmpdx.acl1.8 03/01/29 SMI
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# Template snmpdx Access Control file for Sun SNMP Management Agent for Sun
# Fire B1600
#
# Example 1:
#
# acl = {
#     {
#         communities = public, private
#         access = read-only
#         managers = rag, tag, bobtail
#     }
#     {
#         communities = tigger
```

코드 예 11-5 spapm_snmpdx.acl 파일 예 *(계속)*

```
#          access = read-write
#          managers = rag, tag, bobtail
#      }
# }
#
# Example 2:
#
 acl = {
     {
         communities = public, private
         access = read-only
         managers = localhost
     }
     {
         communities = tigger
         access = read-write
         managers = localhost
     }
 }
#
# Trap destinations are defined in spapm.acl and spama.acl.
# This entry does not need to be edited.
trap = {
}
```

마스터 에이전트 구성 파일

마스터 에이전트 기능을 사용하려는 경우에만(spama.conf의 START_MASTER_AGENT=yes) 이들 파일을 구성해야 합니다.

이 절은 다음의 형식에 대해 설명합니다.

- 105 페이지의 “spama.acl 파일”
- 106 페이지의 “spama.uacl 파일”
- 107 페이지의 “spama.security 파일”

spama.acl 파일

spama.acl 파일은 마스터 에이전트에 대한 액세스 제어를 정의합니다. 파일의 일반 형식이 99 페이지의 “ACL 파일의 형식”에 설명되어 있습니다. 이 절에는 특히 spama.acl 파일과 관련된 정보가 들어있습니다.

이 파일은 기본적으로 /etc/opt/SUNWspa에 위치합니다.

액세스 제어 및 SNMP 어댑터에 대한 트랩을 사용 가능하게 하려면, 임의의 에이전트가 시작될 때 ACL 파일이 존재하는지 확인하십시오. ACL 파일이 보안 관련 정보를 포함하기 때문에, ACL 파일에 root만이 읽을 수 있는 제한된 액세스 권한을 지정하십시오.

이 파일은 SNMPv1 및 SNMPv2c 액세스 권한, 및 SNMP 통지에 대한 수신인을 정의합니다. spama.conf에 SPAPM_TRAPS_ARE_V3=yes가 있는 경우 트랩은 SNMPv3 트랩으로 전송되며, 그렇지 않으면 SNMPv2c 트랩으로 전송됩니다.

acl 그룹

SNMPv3를 채택하려는 경우, SNMPv1 및 SNMPv2c에 의한 쓰기 액세스를 금지하기 원할 것입니다. 그러므로 이 acl은 보통 읽기 전용 액세스만을 허용합니다.

코드 예 11-6 acl 그룹 예

```
acl = {
     {
          communities = public, private
          access = read-only
          manager = localhost
     }
}
```

trap 그룹

이 파일에 대한 trap 그룹은 101 페이지의 “trap 그룹”에 설명된 형식을 따릅니다.

spama.uacl 파일

이 파일은 spama.security 파일과 결합되어 마스터 에이전트가 활성일 때 SNMPv3 보안을 사용하는 데 사용됩니다. 파일의 일반 형식이 99 페이지의 “ACL 파일의 형식”에 설명되어 있습니다. 이 절에는 추가 구성 매개변수를 설명합니다. 명확성을 위해 대부분의 주석이 제거된 파일의 예가 코드 예 11-7에 표시됩니다.

이 파일은 기본적으로 /etc/opt/SUNWspa에 위치합니다.

acl 그룹

acl 그룹은 다음 매개변수를 포함합니다.

- *context-names*—이것은 쉼표로 구분된 문맥 이름 목록입니다.
- *access*—가능한 값은 다음과 같습니다.
  - `read-only`
  - `read-write`
- *security-level*—가능한 값은 다음과 같습니다.
  - `noAuthNoPrivacy`
  - `authNoPrivacy`
  - `authPrivacy`
- *users*—이것은 쉼표로 구분된 사용자 이름 목록입니다.

다음 예에서, `authNoPrivacy`의 최소 보안을 갖고 `public` 및 `null`의 문맥에서 `defaultUser` 및 `defaultUser`의 모든 요청에 대해 액세스가 부여됩니다. 다른 모든 SNMP 요청은 거부됩니다.

이 파일에 `trap` 그룹은 없습니다.

코드 예 11-7 `spama.uacl` 파일 예

```
#ident ì@(#)spama.uacl 1.4 01/29/03 SMIî
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
# This software is the proprietary information of Sun Microsystems, Inc.
# Use is subject to license terms.
#
# Template ACL file
#
 acl = {
     {
         context-names = public,null
         access = read-write
         security-level=authNoPriv
         users = defaultUser
     }
 }
```

spama.security 파일

spama.security 파일은 마스터 에이전트에 대한 액세스가 허용되는 사용자, SNMPv3 예외 및 인증 키를 지정합니다.

파일은 다음 형식을 갖는 userEntry 행의 세트로 구성됩니다.

```
userEntry=<엔진ID>,<사용자 이름>,<보안 이름>,<인증 알고리즘>,<인증 키>,<프라이버시 알고리즘>,<프라이버시 키>,<기억장치 유형>,<템플리트>
```

주의 – 이 파일에서 userEntry 행 이외의 어떤 매개변수도 편집하지 마십시오.

이들 필드는 표 11-2에서 설명됩니다.

표 11-2 spama.security에서 사용자가 구성 가능한 매개변수

매개변수	설명
엔진 ID	사용할 SNMP 엔진의 ID. 다음 중 하나일 수 있습니다. ■ 16진 문자열 ■ <address>:<port>:<IANA number> 양식으로 engineID를 표시하는 텍스트 문자열 ■ 대부분의 경우에 적합한 문자열 localEngineID
사용자 이름	이 항목이 적용되는 사용자 이름
보안 이름	이 사용자 이름에 맵핑될 보안 이름. 일반적으로 동일해야 합니다.
인증 알고리즘	사용할 인증 알고리즘. 다음 중 하나일 수 있습니다. • usmHMACMD5AuthProtocol • usmHMACSHHAuthProtocol • usmNoAuthProtocol
인증 키	인증 알고리즘과 함께 사용할 키. 다음 중 하나일 수 있습니다. • 텍스트 암호 (최소 8 자) • 국지화된 16 진 키 , 예를 들면 0x0098768905AB67EFAA855A453B665B12
프라이버시 알고리즘	사용할 프라이버시 알고리즘. 다음 중 하나일 수 있습니다. • usmDESPrivProtocol • usmNoPrivProtocol(지정하지 않는 경우의 기본값)
프라이버시 키	프라이버시 알고리즘과 함께 사용할 키. 다음 중 하나일 수 있습니다. • 텍스트 암호 (최소 8 자) • 국지화된 16 진 키 , 예를 들면 0x0098768905AB67EFAA855A453B665B12
기억장치 유형	유일하게 승인 가능한 값이 3으로, 지정하지 않는 경우의 기본값입니다.
템플리트	기본값은 false입니다(이 값을 변경할 필요가 없습니다).

명확성을 위해 대부분의 주석이 제거된 파일의 예가 코드 예 11-8에 표시됩니다.

이 파일은 기본적으로 /etc/opt/SUNWspa에 위치합니다.

기본 spama.security 파일에는 수정하고 주석을 제거하여 고유한 사용자를 정의할 수 있는 두 샘플 사용자가 들어있습니다. 첫 번째 샘플은 다음을 갖는 defaultUser라는 사용자를 지정합니다.

- MD5 알고리즘만을 사용하는 인증
- 프라이버시 없음
- 인증 암호 "mypassword"

두 번째 샘플은 다음을 갖는 defaultUser라는 사용자를 지정합니다.

- MD5 알고리즘과 인증 암호 "mypassword"를 사용한 인증
- DES 알고리즘과 프라이버시 암호 "mypassword"를 사용하는 프라이버시
- DES 알고리즘을 사용하는 프라이버시

코드 예 11-8 spama-security 파일 예

```
#ident “@(#)spama.security 1.7 01/29/03 SMI”
#
# Copyright 2002 Sun Microsystems, Inc. All rights reserved.
# This software is the proprietary information of Sun Microsystems, Inc.
# Use is subject to license terms.
#
# Template security file

# localEngineBoots=0

# defaultUser configuration. Authentication only.
# userEntry=localEngineID,defaultUser,,usmHMACMD5AuthProtocol,mypasswd

# defaultUser configuration. Authentication and encryption.
# userEntry=
localEngineID,defaultUser,null,usmHMACMD5AuthProtocol,mypasswd,usmDESPrivProt
ocol,mypasswd,3,
```

12 장

소프트웨어 구성

이 장에서는 설치 후의 기본 구성에 대해 설명하고 11 장에서 설명한 파일을 수정하는 방법을 설명합니다.

이 장에는 다음 절이 들어있습니다.

기본 구성

소프트웨어는 다음 기본 구성과 함께 설치됩니다.

- 마스터 에이전트는 사용 불가능합니다(START_MASTER_AGENT=no).
- 조정자는 사용 가능합니다(START_MEDIATOR_AGENT=yes).
- 조정자가 snmpdx의 서브에이전트로서 구성됩니다.

참고 – 보안상의 이유 때문에, 에이전트를 모니터링하는 것을 제외한 모든 시스템을 배제하도록 액세스를 제한하도록 snmpdx ACL 파일을 구성해야 합니다.

액세스 제어

조정자에 대한 액세스 제어를 사용 가능하게 하려면, 102 페이지의 “spapm.acl 파일”에 설명한 대로 조정자 ACL 파일을 구성하십시오.

snmpdx를 사용하려는 경우(기본 구성), spapm_snmpdx.acl을 수정하여 액세스 권한을 설정하고 spapm.acl을 수정하여 트랩 수신인을 설정하십시오.

조정자 시작 및 중지

정상 snmpdx 시작 스크립트를 사용하여 조정자를 시작하십시오.

```
# /etc/init.d/init.snmpdx start
```

조정자 스크립트를 사용하여 조정자를 중지하십시오.

```
# /etc/init.d/spama stop
```

직접 액세스를 위한 수동 구성

snmpdx가 SNMPv1만을 지원하므로, SNMPv2c 특정 get-bulk 조작을 사용하고 마스터 에이전트를 사용하지 않으려는 경우 직접 SNMPv2c 액세스가 사용 가능하도록 조정자가 사용하는 포트를 구성할 수 있습니다.

조정자를 수동으로 구성하려면, spama.conf를 다음과 같이 변경하십시오.

1. **SUB_AGENT=no를 설정합니다.**
2. **SPAPM_REQ_PORT를 필수 포트 번호로 설정합니다.**
   조정자에 전송된 SNMPv2c 요청이 이 포트로 전송되어야 합니다.

타사 마스터 에이전트의 서브에이전트로서의 조정자

조정자를 명령행 매개변수를 사용한 포트 번호 지정을 지원하는 타사 마스터 에이전트의 서브에이전트로서 구성하려면 다음을 수행하십시오.

1. **localhost로부터의 액세스를 허용하도록 조정자 ACL 파일을 구성합니다(102 페이지의 "spapm.acl 파일"을 참조하십시오).**
2. **적당한 포트 번호와 함께 다음 호출로 조정자를 시작하도록 마스터 에이전트를 구성합니다.**

```
/etc/init.d/spama start <port>
```

3. **다음 OID 서브 트리로 요청을 전달하도록 마스터 에이전트를 구성합니다.**
   - .iso.org.dod.internet.mgmt.mib-2.entityMIB
   - .iso.org.dod.internet.private.enterprises.sun.products.sunFire.sunPlatMIB

   숫자로는
   - .1.3.6.1.2.1.47
   - .1.3.6.1.4.1.42.2.70.101

조정자 및 SNMPv3 마스터 에이전트

조정자 및 마스터 에이전트를 사용 가능하게 하려면 최소한 다음 변경을 수행해야 합니다.

1. **spama.conf 파일에서,**

   a. START_MASTER_AGENT=yes**를 설정합니다.**

   b. SUB_AGENT=no**를 설정합니다.**

2. **102 페이지의 "`spapm.acl` 파일"에서 설명하는 것처럼 localhost로부터의 액세스가 사용 가능하도록 조정자 ACL 파일을 구성합니다.**

3. **103 페이지의 "`spapm_snmpdx.acl` 파일"에서 설명하는 것처럼 localhost로부터의 액세스가 사용 가능하도록 snmpdx를 구성합니다.**

4. **105 페이지의 "`spama.acl` 파일"에서 설명하는 것처럼 원하는 관리자로부터의 액세스가 사용 가능하도록 마스터 에이전트 ACL 파일을 구성합니다.**

5. **105 페이지의 "`spama.acl` 파일" 및 107 페이지의 "`spama.security` 파일"에서 설명하는 것처럼 SNMPv3 사용자, 문맥 및 인증과 암호화 레벨을 정의하도록 보안 파일을 구성합니다.**

에이전트 시작 및 중지

조정자 스크립트를 사용하여 조정자 및 마스터 에이전트를 시작하십시오.

```
# /etc/init.d/spama start
```

조정자 스크립트를 사용하여 조정자 및 마스터 에이전트를 중지하십시오.

```
# /etc/init.d/spama stop
```

SNMPv3 트랩 전송

조정자로부터 SNMPv3 트랩을 전송하도록 마스터 에이전트를 구성하려면, spama.conf 파일에서(90 페이지의 “spama.conf”를 참조하십시오),

1. **SPAPM_TRAPS_ARE_V3=yes를 설정합니다.**
2. **선택적으로 SNMPV3_USER를 설정합니다.**

참고 – 트랩 사용자가 spama.uacl 및 spama.security에서 SNMPV3 사용자로 구성되어야 합니다.

13장

소프트웨어 설치 제거

이 장에서는 소프트웨어를 설치 제거하는 방법에 대해 설명합니다.

일반적으로, SNMP를 설치 제거하기 위한 필요한 것은 `pkgrm` 명령을 사용하여 설치한 패키지를 제거하는 것이 전부입니다. 이 절차는 모든 관련 파일과 링크를 제거하고, `snmpdx`를 다시 사용 가능하게 합니다.

SNMP 소프트웨어에 의해 자동으로 수행된 구성 변경이 원래 상태로 복원됩니다. 그러나, snmpdx ACL 파일 같은 임의의 외부 파일 설정을 수정한 경우 SNMP 소프트웨어를 제거한 후 수동으로 해당 파일을 복원해야 합니다.

참고 – 아래에 나열되는 절차는 Java SNMP API 패키지인 `SUNWjsnmp`를 설치 제거하지 않습니다. 이 패키지의 Solaris 버전을 다시 설치하려는 경우, 먼저 Java SNMP API를 제거해야 합니다.

플랫폼 에이전트 및 대상 에이전트 패키지

플랫폼 에이전트 서버에서 플랫폼 에이전트 패키지를 제거하려면 다음을 입력하십시오.

```
# pkgrm SUNWbgpmr SUNWbgpm SUNWjdrt SUNWjsnmp SUNWbgpjo \
SUNWbgodr SUNWbgod SUNWbgcmr SUNWbgcm SUNWbgpc SUNWbgptk
```

주의 – `SUNWjdrt`와 `SUNWjsnmp` 패키지는 둘다 시스템 패키지이며 다른 제품이 사용할 수 있으므로 주의해서 제거하십시오.

Sun Fire B100 블레이드에서 대상 플랫폼 패키지를 제거하려면, 다음을 입력하십시오.

```
# pkgrm SUNWbgpr SUNWbgcm SUNWbgpc SUNWbgptk
```

도메인 에이전트 패키지

Sun Fire B100 블레이드에서 도메인 에이전트 패키지를 제거하려면, 다음을 입력하십시오.

```
# pkgrm SUNWbgpmr SUNWbgpm SUNWjdrt SUNWjsnmp SUNWbgpji \
SUNWbgidr SUNWbgcmr SUNWngcm SUNWbgpc SUNWbgptk
```

주의 – SUNWjdrt와 SUNWjsnmp 패키지는 둘다 시스템 패키지이며 다른 제품이 사용할 수 있으므로 주의해서 제거하십시오.

14장

문제 해결

이 장에서는 시스템 문제 해결을 돕기 위한 정보를 제공합니다.

문제점

● **기본 구성(snmpdx)을 사용할 때 SNMP 에이전트에서 응답이 없습니다.**

1. 다음을 입력하여 조정자가 실행 중인지 확인합니다.

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
```

응답이 위와 비슷한 경우 조정자 프로세스가 실행 중입니다.
다음을 입력하여 조정자를 중지한 후 다시 시작하십시오.

```
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
```

2. 다음을 입력하여 올바른 Java 버전이 설치되었는지 확인합니다.

```
# /usr/j2se/bin/java –version
java version "1.4.1_03"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_03–b04)
Java HotSpot(TM) Client VM (build 1.4.1_03–b04, mixed mode)
```

이것은 버전 1.4 이상을 보고해야 합니다. 이 경우에 해당하지 않는 경우 68 페이지의 “Java 환경”에 설명한 대로 Java 1.4 JDK를 설치하십시오.

3. **다음을 입력하여 SUNWjsnmp의 올바른 버전이 설치되었는지 확인합니다.**

```
# pkginfo -l SUNWjsnmp | grep VERSION
   VERSION:  5.0
```

버전 1.0이 표시되는 경우, SUNWjsnmp 패키지를 제거하고 SUNWspa.*.tar.Z 아카이브에 있는 버전을 다시 설치하십시오(69 페이지의 "Java SNMP API"를 참조하십시오).

4. **spama.conf 파일이 다음 항목을 포함하는지 확인합니다.**

START_MASTER_AGENT=no

START_MEDIATOR=yes

SUB_AGENT=yes

5. **다음을 입력하여 조정자가 snmpdx에 올바르게 등록되었는지 확인합니다.**

```
# cat /var/snmp/snmpdx.st
spapm spapm 2516 34050
snmpd snmpd 2567 34053
```

위의 spapm 항목은 조정자가 snmpdx의 서브에이전트로 등록되었음을 보여줍니다.

6. **/etc/snmp/conf/spapm.reg 및 /etc/snmp/conf/spapm.rsrc가 손상되지 않았는지 확인합니다.**

다음을 입력하여 조정자를 중지한 후 다시 시작하십시오.

```
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
```

7. **ACL 파일의 권한이 올바르게 설정되었는지 확인합니다.**

- spapm_snmpdx.acl은 사용되는 SNMP 관리자에 대한 액세스를 정의합니다.
- spapm.acl은 localhost에 대한 액세스를 정의합니다.

자세한 정보에 대해서는 11 장을 참조하십시오.

문제점

- **플랫폼 에이전트를 사용할 때 하드 디스크 드라이브(HDD) 또는 이더넷 MAC 주소에 대한 계측이 없습니다.**

이 정보는 운용 환경이 대상 Sun Fire B100 블레이드에서 실행 중일 때만 사용할 수 있습니다.

1. **Sun Fire B100 블레이드가 부트되는지 확인합니다.**

2. **다음을 입력하여 대상 계측이 Sun Fire B100 블레이드에서 실행 중인지 확인합니다.**

```
# netstat –an | grep 1099
      *.1099        *.*          0      0 24576      0 LISTEN
```

청취 중인 포트가 없는 경우, 대상 계측이 실행하지 않고 있습니다.

다음을 입력하여 Sun Fire B100 블레이드에 대한 계측을 시작하십시오.

```
# /etc/init.d/spardp start
```

3. **다음을 입력하여 올바른 Java 버전이 설치되었는지 확인합니다.**

```
# /usr/j2se/bin/java -version
java version "1.4.1_03"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_03-b04)
Java HotSpot(TM) Client VM (build 1.4.1_03–b04, mixed mode)
```

이것은 버전 1.4 이상을 보고해야 합니다. 잘못된 버전이 계측이 시작하도록 유발할 수 있지만, 짧은 기간 후에 실패합니다.

이 경우에 해당하지 않는 경우 68 페이지의 “Java 환경”에 설명한 대로 Java 1.4 JDK를 설치하십시오.

문제점

● **에이전트가 액세스 가능하지만, 모니터되는 플랫폼에 대한 계측이 없습니다.**

1. **다음을 입력하여 디스커버리 데몬이 실행 중인지 확인합니다.**

```
# netstat -a | grep mismi
      *.mismi              *.*                0      0 24576      0 LISTEN
      *.mismi                           *.*  0
0 24576      0 LISTEN
```

위의 출력은 디스커버리 데몬이 관리되는 플랫폼(들)의 요청을 청취 중임을 보여줍니다.

a. /etc/services에 다음 항목이 있는지 확인합니다.

```
mismi           8265/tcp                # MISMI Discovery
```

b. /etc/inetd.conf에 다음 항목이 있는지 확인합니다.

```
# MISMIDISCOVERY - mismiDiscovery daemon
mismi   stream  tcp6    nowait  root    /opt/SUNWspa/bin/mismiDiscovery
mismiDiscovery
```

c. 다음을 입력하여 /etc/inetd.conf가 /etc/inet/inetd.conf에 대한 기호 링크인지 확인합니다.

```
# ls –l /etc/inetd.conf
lrwxrwxrwx   1 root     root          17 Jan  7 17:08 /etc/inetd.conf ->
./inet/inetd.conf
```

링크가 존재하지 않는 경우, 파일의 업데이트가 SUNWbgodr 패키지의 설치 중에 실패합니다.

다음을 입력하여 inetd 구성을 정정하고 다시 시작하십시오.

```
# pkill –1 inetd
```

2. 다음을 입력하여 플랫폼이 검색되었는지 확인합니다.

```
# netstat –a | grep mismi
      *.mismi              *.*                0      0 24576      0 LISTEN
blade-174-119.36780  hornet-sc.mismi       8192      0 24820      0 ESTABLISHED
      *.mismi                           *.*  0
0 24576      0 LISTEN
```

출력은 디스커버리 데몬이 청취 중이고 플랫폼 시스템 컨트롤러(<hornet-sc>라고 부름)에 대한 연결이 구축되었는지 확인합니다.

연결이 존재하지 않는 경우, 82 페이지의 “시스템 컨트롤러 구성”에 설명한 대로 시스템 컨트롤러 설정을 점검하십시오.

문제점

● **SNMPv3 get 및 set 요청이 시간종료합니다.**

■ **가능한 원인**

spama.security 파일에 있는 localEngineId 또는 localEngineBoots 수가 편집 또는 삭제되었을 수 있습니다.

■ **점검**

파일이 편집되었는지 여부를 판별하는 것은 간단하지 않습니다.

■ **수정**

아래에 표시된 것처럼 에이전트를 다시 시작한 후, 관리 응용프로그램을 다시 시작하여 재동기화하십시오.

```
# /etc/init.d/spama stop
# /etc/init.d/spama start
```

문제점

● **SNMP get 및 set 요청이 시간종료합니다.**

■ **가능한 원인**

부하가 과중한 시스템에서는 snmpdx 마스터 에이전트가 SNMP 조정자에 대한 요청을 시간종료할 수 있습니다. 이 시간종료는 현재 2초(2000000μs)로 설정됩니다.

■ **점검**

관리 응용프로그램이 보고한 시간종료가 snmpdx와 SNMP 조정자 사이에서 또는 관리 응용프로그램과 snmpdx 사이에서 발생했는지를 판별하는 것은 간단하지 않습니다.

■ **수정**

/etc/snmp/conf/spapm.reg 파일의 시간종료 등록정보를 편집하여 시간종료를 증가시킬 수 있습니다. 이 파일을 편집하는 경우, 다음을 입력하여 조정자를 다시 시작하십시오.

```
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
```

부록 **A**

J2SE 1.3.1과 공존하기 위한 J2RE 설치

이 부록은 플랫폼 에이전트 서버와 B100 도메인에 J2RE(Java 2 Runtime Environment) Standard Edition 1.4를 J2SE 1.3.1과 공존하도록 설치하는 방법과, 설치를 찾기 위해 시작 스크립트를 수정하는 방법을 설명합니다.

이 부록에는 다음 절이 들어있습니다.

J2RE 1.4 설치

J2SE 1.3.1과 공존하기 위해 J2RE 1.4를 설치하려면, 68 페이지의 “Java 환경”에서 설명하는 것처럼 아래 절차를 따르십시오.

J2RE 1.4는 다음 웹 페이지에서 자체 추출 이진 파일로 사용할 수 있습니다.

```
http://java.sun.com/j2se/1.4/download.html
```

아래 지침에 따라서 J2RE를 설치하십시오. 파일 다운로드에 대한 추가 정보가 위의 웹 사이트에 있습니다.

참고 – 이 제품은 32 비트 지원만이 필요하므로, J2RE에 대한 64 비트 부록을 설치할 필요는 없습니다.

다음 단계에서, <버전>을 적당한 J2RE 갱신 버전 번호로 대체하십시오.

예를 들어, 업데이트 1.4.0_01을 다운로드하려는 경우, 명령

```
# chmod +x j2re-1_4_<version>-solaris-sparc.sh
```

이 다음과 같이 됩니다.

```
chmod +x j2re-1_4_0_01-solaris-sparc.sh
```

1. **다운로드하고 파일 크기를 점검합니다.**

   필수 파일은 다음과 같습니다.

   j2re-1_4_<version>-solaris-sparc.sh

   파일을 다운로드하기 전에, 다운로드 페이지에 제공되는 파일 크기를 기록하십시오. 다운로드가 완료된 후에는 손상되지 않은 전체 소프트웨어 파일을 다운로드했는지 확인하십시오.

   파일을 루트가 액세스할 수 있는 위치에 다운로드하는지 확인하십시오(예:in /tmp).

2. **su를 실행하고 수퍼유저 암호를 입력하여 루트가 됩니다.**

3. **자체 추출 이진에 실행 권한이 설정되었는지 확인합니다.**

```
# chmod +x j2re-1_4_<version>-solaris-sparc.sh
```

4. **파일이 설치되어야 하는 디렉토리로 변경합니다.**

```
# cd /usr
```

5. **자체 추출 이진을 실행합니다.**

   /usr/j2re1.4.<version>이라고 부르는 디렉토리가 작성되는데, 이 디렉토리에 J2RE가 들어있습니다.

6. **J2RE가 올바르게 설치되었는지 확인합니다.**

```
# /usr/j2re1.4.1_01/bin/java -version
java version "1.4.1_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_01-
b01)
Java HotSpot(TM) Client VM (build 1.4.1_01-b01, mixed mode)
```

이것은 버전 `1.4` 이상을 보고해야 합니다. 이 예는 버전이 `1.4.1_01`임을 표시합니다.

7. **자체 추출 이진을 삭제합니다.**
8. **루트 쉘을 종료합니다.**

시작 스크립트 편집

이 절은 J2RE 1.4를 설치할 때 시작 스크립트를 수정하는 방법을 설명합니다.
수정

10 장과 함께 이 절을 읽으십시오.

도메인 하드웨어 모니터링

이들 단계는 77 페이지의 “도메인 하드웨어 모니터링을 위한 소프트웨어 설치”의 4 단계와 관련됩니다.

1. **모니터되는 각 B100 블레이드에서 다음의 각 시작 스크립트를 편집합니다.**
   - `/etc/init.d/spaibdm`
   - `/etc/init.d/spapom`

   아래와 같은 행

   `JAVA=/usr/j2se/bin/java`

   을 다음 행으로 대체하십시오.

   `JAVA=/usr/j2re1.4.<version>/bin/java`

2. **모니터되는 각 B100 블레이드에서 다음 시작 스크립트를 편집합니다.**

   ■ `/etc/init.d/spama`

   아래와 같은 행

   ```
   JAVA_JAVA=/usr/j2se/bin/java
   ```

   을 다음 행으로 대체하십시오.

   ```
   JAVA_JAVA=/usr/j2re1.4.<version>/bin.java
   ```

플랫폼 하드웨어 모니터링

단계 1과 단계 2는 79 페이지의 "대상 계측과 함께 소프트웨어 설치"의 4 단계, 및 81 페이지의 "대상 계측 없이 소프트웨어 설치"의 4 단계와 관련됩니다.

단계 3은 81 페이지의 "대상 계측 없이 소프트웨어 설치"의 11 단계와만 관련됩니다.

1. **플랫폼 에이전트 서버에서 다음의 각 시작 스크립트를 편집합니다.**

   ■ `/etc/init.d/spapom`

   아래와 같은 행

   ```
   JAVA=/usr/j2se/bin/java
   ```

   을 다음 행으로 대체하십시오.

   ```
   JAVA=/usr/j2re1.4.<version>/bin/java
   ```

2. **플랫폼 에이전트 서버에서 다음 시작 스크립트를 편집합니다.**

   `/etc/init.d/spama`

   아래와 같은 행

   ```
   JAVA_JAVA=/usr/j2se/bin/java
   ```

   을 다음 행으로 대체하십시오.

   ```
   JAVA_JAVA=/usr/j2re1.4.<version>/bin.java
   ```

3. **모니터되는 각 B100 블레이드(대상)에서 다음 시작 스크립트를 편집합니다.**

   ■ `/etc/init.d/spardp`

   아래와 같은 행

   `JAVA=/usr/j2se/bin/java`

   을 다음 행으로 대체하십시오.

   `JAVA=/usr/j2re1.4.<version>/bin/java`

색인

ㄹ

ㅁ

ㅂ

ㅅ

L

M

N

O

S

T

W