Oracle® Solaris 관리: Oracle Solaris Zones, Oracle Solaris 10 Zones 및 리소스 관리

부품 번호: E25830
2012년 3월

ORACLE

120410@25097

목차

머리말

이 책은 Oracle Solaris 운영 체제 관리 정보의 중요 부분을 포함하는 다중 볼륨 세트의 일부입니다. 이 책에서는 이미 운영 체제를 설치하고 사용할 네트워킹 소프트웨어를 설정한 것으로 가정합니다.

Oracle Solaris 영역 정보

Oracle Solaris 영역 제품은 응용 프로그램을 위한 완벽한 런타임 환경입니다. 영역은 응용 프로그램에서 플랫폼 리소스로의 가상 매핑을 제공합니다. 영역을 사용하면 영역에서 Oracle Solaris 운영 체제의 단일 인스턴스를 공유하는 경우라도 응용 프로그램 구성 요소를 서로 격리할 수 있습니다. 일반적으로 리소스 관리 기능이라고 하는 Oracle Solaris 리소스 관리자 제품 구성 요소를 통해 작업 부하에서 받는 리소스의 수량을 할당할 수 있습니다.

영역은 CPU 등의 리소스 소비에 대한 경계를 설정합니다. 이러한 경계는 영역에서 실행 중인 응용 프로그램의 변경되는 처리 요구 사항에 맞도록 확장될 수 있습니다.

추가 격리를 위해 읽기 전용 루트가 있는 영역(변경할 수 없는 영역이라고 함)을 구성할 수 있습니다.

Oracle Solaris 10 영역 정보

`solaris10` 브랜드 비전역 영역이라고도 하는 Oracle Solaris 10 영역은 BrandZ 기술을 사용하여 Oracle Solaris 11 운영 체제에서 Oracle Solaris 10 응용 프로그램을 실행합니다. 즉, 응용 프로그램을 비전역 영역에서 제공한 안전한 환경에서 수정 없이 실행할 수 있습니다. 이로 인해 Oracle Solaris 10 시스템을 사용하여 응용 프로그램을 개발, 테스트 및 배치할 수 있습니다. 이러한 브랜드 영역 내에서 실행되는 작업 부하는 Oracle Solaris 11 릴리스에서만 사용할 수 있는 커널에 대한 향상된 기능을 활용하고 혁신적인 일부 기술을 사용할 수 있습니다.

이 제품을 사용하려면 제3부를 참조하십시오.

Oracle Solaris Trusted Extensions 시스템에서의 Oracle Solaris 영역 사용에 대한 정보

Oracle Solaris Trusted Extensions 시스템에서의 영역 사용에 대한 자세한 내용은 **Trusted Extensions 구성 및 관리**의 13 장, “Trusted Extensions에서 영역 관리(작업)”를 참조하십시오. `labeled` 브랜드만 Oracle Solaris Trusted Extensions 시스템에서 부트될 수 있습니다.

Oracle Solaris Cluster 영역 클러스터

영역 클러스터는 Oracle Solaris Cluster 소프트웨어의 기능입니다. 영역 클러스터의 모든 노드는 `cluster` 속성을 사용하여 비전역 `solaris` 영역으로 구성됩니다. 다른 브랜드 유형은 허용되지 않습니다. 영역에서 제공되는 격리를 사용하여 전역 클러스터에서와 같은 방식으로 영역 클러스터에서 지원되는 서비스를 실행할 수 있습니다. 영역 클러스터 구성에 대한 자세한 내용은 **Oracle Solaris Cluster 소프트웨어 설치 설명서**를 참조하십시오.

Oracle Solaris 리소스 관리자

리소스 관리를 통해 응용 프로그램이 사용 가능한 시스템 리소스를 사용하는 방법을 제어할 수 있습니다. 제1부를 참조하십시오.

이 책의 대상

이 책은 Oracle Solaris 릴리스가 실행되는 한 대 이상의 시스템을 관리하는 사용자를 대상으로 작성되었습니다. 이 책을 사용하려면 적어도 1~2년의 UNIX 시스템 관리 경험이 있어야 합니다.

시스템 관리 설명서의 구성

시스템 관리 설명서에서 설명하는 항목 목록은 다음과 같습니다.

책 제목	내용
SPARC 플랫폼에서 Oracle Solaris 부트 및 종료	SPARC 플랫폼에서 시스템 부트 및 종료, 부트 서비스 관리, 부트 동작 수정, ZFS에서 부트, 부트 아카이브 관리 및 부트 문제 해결
x86 플랫폼에서 Oracle Solaris 부트 및 종료	x86 플랫폼에서 시스템 부트 및 종료, 부트 서비스 관리, 부트 동작 수정, ZFS에서 부트, 부트 아카이브 관리 및 부트 문제 해결

책 제목	내용
Oracle Solaris 관리: 일반 작업	Oracle Solaris 명령 사용, 시스템 부트 및 종료, 사용자 계정 및 그룹 관리, 서비스, 하드웨어 오류, 시스템 정보, 시스템 리소스 및 시스템 성능 관리, 소프트웨어, 인쇄, 콘솔 및 터미널 관리, 시스템 및 소프트웨어 문제 해결
Oracle Solaris 관리: 장치 및 파일 시스템	이동식 매체, 디스크 및 장치, 파일 시스템, 데이터 백업 및 복원
Oracle Solaris 관리: IP 서비스	TCP/IP 네트워크 관리, IPv4 및 IPv6 주소 관리, DHCP, IPsec, IKE, IP 필터 및 IPQoS
Oracle Solaris Administration: Naming and Directory Services	NIS에서 LDAP으로 전환을 비롯한 DNS, NIS 및 LDAP 이름 지정 및 디렉토리 서비스
Oracle Solaris 관리: 네트워크 인터페이스 및 네트워크 가상화	WiFi 무선을 포함하는 자동 및 수동 IP 인터페이스 구성, 브릿지, VLAN, 통합, LLDP 및 IPMP 관리, 가상 NIC 및 리소스 관리
Oracle Solaris 관리: 네트워크 서비스	웹 캐시 서버, 시간 관련 서비스, 네트워크 파일 시스템(NFS 및 Autofs), 메일, SLP, PPP
Oracle Solaris 관리: Oracle Solaris Zones, Oracle Solaris 10 Zones 및 리소스 관리	응용 프로그램이 사용 가능한 시스템 리소스를 사용하는 방식을 제어할 수 있는 리소스 관리 기능, 운영 체제 서비스를 가상화하여 응용 프로그램을 실행하기 위한 격리된 환경을 만드는 Oracle Solaris Zones 소프트웨어 분할 기술, Oracle Solaris 11 커널에서 실행되는 Oracle Solaris 10 환경을 호스트하는 Oracle Solaris 10 영역
Oracle Solaris 관리: 보안 서비스	감사, 장치 관리, 파일 보안, BART, Kerberos 서비스, PAM, 암호화 프레임워크, 키 관리 프레임워크, 권한, RBAC, SASL, 보안 셸 및 바이러스 검사
Oracle Solaris Administration: SMB and Windows Interoperability	SMB 클라이언트에서 SMB 공유를 사용할 수 있도록 Oracle Solaris 시스템을 구성할 수 있는 SMB 서비스, SMB 공유에 액세스할 수 있는 SMB 클라이언트, Oracle Solaris 시스템과 Windows 시스템 간에 사용자 및 그룹 ID를 매핑할 수 있는 고유의 ID 매핑 서비스
Oracle Solaris 관리: ZFS 파일 시스템	ZFS 저장소 풀 및 파일 시스템 만들기/관리, 스냅샷, 복제, 백업, 액세스 제어 목록(ACL)을 통한 ZFS 파일 보호, 영역이 설치된 Oracle Solaris 시스템에서 ZFS 사용, 에뮬레이트된 볼륨, 문제 해결 및 데이터 복구
Trusted Extensions 구성 및 관리	Trusted Extensions와 관련된 시스템 설치, 구성 및 관리
Oracle Solaris 11 보안 지침	영역, ZFS 및 Trusted Extensions와 같은 보안 기능에 대한 사용 시나리오와 Oracle Solaris 시스템의 보안 설정

책 제목	내용
Oracle Solaris 10에서 Oracle Solaris 11로 전환	설치, 장치, 디스크 및 파일 시스템 관리, 소프트웨어 관리, 네트워킹, 시스템 관리, 보안, 가상화, 데스크탑 기능, 사용자 계정 관리, 사용자 환경 에뮬레이트된 볼륨, 문제 해결 및 데이터 복구 영역에서 Oracle Solaris 10에서 Oracle Solaris 11로의 전환을 위한 시스템 관리 정보 및 예 제공

Oracle Support에 액세스

Oracle 고객은 My Oracle Support를 통해 온라인 지원에 액세스할 수 있습니다. 자세한 내용은 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info를 참조하거나, 청각 장애가 있는 경우 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs를 방문하십시오.

활자체 규약

다음 표는 이 책에서 사용되는 활자체 규약에 대해 설명합니다.

표 P–1 활자체 규약

활자체	의미	예
`AaBbCc123`	명령 및 파일, 디렉토리 이름; 컴퓨터 화면에 출력되는 내용입니다.	`.login` 파일을 편집하십시오. 모든 파일 목록을 보려면 `ls -a` 명령을 사용하십시오. `machine_name% you have mail.`
`AaBbCc123`	사용자가 입력하는 내용으로 컴퓨터 화면의 출력 내용과 대조됩니다.	`machine_name%` **`su`** `Password:`
AaBbCc123	새로 나오는 용어, 강조 표시할 용어입니다. 명령줄 변수를 실제 이름이나 값으로 바꾸십시오.	`rm` *filename* 명령을 사용하여 파일을 제거합니다.
AaBbCc123	책 제목, 장, 절	**사용자 설명서**의 6장을 읽으십시오. **캐시**는 로컬로 저장된 복사본입니다. 파일을 저장하면 **안 됩니다**. **주:** 일부 강조된 항목은 온라인에서 굵은체로 나타납니다.

명령 예의 셸 프롬프트

다음 표에는 Oracle Solaris OS에 포함된 셸의 기본 UNIX 시스템 프롬프트 및 수퍼유저 프롬프트가 나와 있습니다. 명령 예제에 표시된 기본 시스템 프롬프트는 Oracle Solaris 릴리스에 따라 다릅니다.

표 P–2 셸 프롬프트

셸	프롬프트
Bash 셸, Korn 셸 및 Bourne 셸	`$`
수퍼유저용 Bash 셸, Korn 셸 및 Bourne 셸	`#`
C 셸	`machine_name%`
수퍼유저용 C 셸	`machine_name#`

권한 및 관리 권한에 대한 정보 가져오기

역할 및 관리 권한에 대한 자세한 내용은 **Oracle Solaris 관리: 보안 서비스**의 제III부, "역할, 권한 프로파일 및 권한"을 참조하십시오.

제 1 부

Oracle Solaris 리소스 관리

이 부분에서는 응용 프로그램이 사용 가능한 시스템 리소스를 사용하는 방법을 제어할 수 있는 Oracle Solaris 리소스 관리에 대해 설명합니다.

◆ ◆ ◆ 1 장

1 리소스 관리 소개

Oracle Solaris 리소스 관리 기능을 사용하면 응용 프로그램의 사용 가능한 시스템 리소스 사용 방식을 제어할 수 있습니다. 다음과 같은 작업을 수행할 수 있습니다.

- 컴퓨팅 리소스(예: 프로세서 시간) 할당
- 할당 사용 방식을 모니터링하고 나서 필요에 따라 할당 조정
- 분석, 청구 및 용량 계획에 대한 확장 계정 정보 생성

이 장에서는 다음 내용을 다룹니다.

리소스 관리 개요

현대식 컴퓨팅 환경은 시스템에서 다양한 응용 프로그램이 생성하는 가변 작업 부하에 대해 유연한 응답을 제공해야 합니다. **작업 부하**는 응용 프로그램 또는 응용 프로그램 그룹의 모든 프로세스를 총칭하는 용어입니다. 리소스 관리 기능이 사용되는 경우 Oracle Solaris 운영 체제는 새 응용 프로그램 요청에 동적으로 맞춤으로써 작업 부하 요구에 응답합니다. 이 기본 응답은 일반적으로 시스템의 모든 작업에 리소스에 대한 동일한 액세스 권한이 제공됨을 의미합니다. 리소스 관리 기능을 사용하면 작업 부하를 개별적으로 처리할 수 있습니다. 다음과 같은 작업을 수행할 수 있습니다.

- 특정 리소스에 대한 액세스 제한
- 우선 순위에 따라 작업 부하에 리소스 제공
- 작업 부하를 서로 간에 분리

교차 작업 부하 성능을 최소화하는 기능과 리소스 사용 및 활용을 모니터링하는 기능을 합쳐서 **리소스 관리**라고 합니다. 리소스 관리는 알고리즘 컬렉션을 통해 구현됩니다. 알고리즘은 응용 프로그램이 실행 과정에서 제공하는 일련의 기능 요청을 처리합니다.

리소스 관리 기능을 사용하면 다양한 작업 부하와 관련하여 운영 체제의 기본 동작을 수정할 수 있습니다. **동작**은 기본적으로 응용 프로그램이 시스템에 하나 이상의 리소스 요청을 제공할 때 운영 체제 알고리즘에 의한 결정 집합을 말합니다. 리소스 관리 기능을 사용하여 다음을 수행할 수 있습니다.

- 별도로 허용되지 않을 경우 큰 할당 집합에 대해 한 응용 프로그램을 다른 응용 프로그램보다 우선하도록 하거나 리소스를 거부합니다.
- 분리된 방식을 사용하는 대신에, 집합적으로 특정 할당을 처리합니다.

리소스 관리 기능을 사용하는 시스템 구성의 구현은 몇 가지 목적으로 사용할 수 있습니다. 다음과 같은 작업을 수행할 수 있습니다.

- 응용 프로그램의 분별 없는 리소스 사용을 방지합니다.
- 외부 이벤트에 따라 응용 프로그램의 우선 순위를 변경합니다.
- 리소스가 응용 프로그램 집합에 대해 보장하는 것과 시스템 활용 최대화 목적을 비교 평가합니다.

리소스 관리 구성을 계획할 때 주요 요구 사항에 다음 항목이 포함됩니다.

- 시스템에서 경합 중인 작업 부하 식별
- 기본 작업 부하를 구성하는 성능 요구 사항을 가진 작업 부하와 충돌하지 않는 작업 부하 구분

조정 및 충돌하는 작업 부하를 식별한 후 시스템의 기능 제한 내에서 비즈니스의 서비스 목표에 최소한의 절충을 제공하는 리소스 구성을 만들 수 있습니다.

효과적인 리소스 관리는 컨트롤 방식, 알림 방식 및 모니터링 방식을 제공하여 Oracle Solaris 시스템에서 사용할 수 있습니다. 이러한 기능 중 대부분은 proc(4) 파일 시스템, 프로세서 세트 및 예약 클래스 등 기존 방식에 비해 향상된 기능을 통해 제공됩니다. 기타 기능은 리소스 관리와 관련된 것입니다. 이러한 기능은 이후 단원에서 설명합니다.

리소스 분류

리소스는 응용 프로그램 동작을 변경할 목적으로 조작할 수 있는 컴퓨팅 시스템의 한 측면입니다. 따라서 리소스는 응용 프로그램이 암시적 또는 명시적으로 요청하는 기능입니다. 기능이 거부되거나 제한된 경우 견고하게 작성된 응용 프로그램의 실행이 보다 느리게 진행됩니다.

리소스 식별과는 달리, 축 수를 통해 리소스를 분류할 수 있습니다. 지정된 CPU 할당 등과 같이 시간에 종속되지 않은 축과 비교할 때, 이 축은 시간에 기반을 둔 CPU 시간 등이 명시적으로 요청되는 것과는 반대로 암시적으로 요청될 수 있습니다.

일반적으로 스케줄러 기반 리소스 관리는 응용 프로그램이 암시적으로 요청할 수 있는 리소스에 적용됩니다. 예를 들면 응용 프로그램은 실행을 계속하기 위해 추가 CPU

시간을 암시적으로 요청합니다. 응용 프로그램은 데이터를 네트워크 소켓에 쓰기 위해 대역폭을 암시적으로 요청합니다. 암시적으로 요청된 리소스의 총 사용량에 대한 제약 조건이 부과될 수 있습니다.

추가 인터페이스가 제공되어 대역폭이나 CPU 서비스 레벨에 대한 협상이 명시적으로 이루어질 수도 있습니다. 추가 스레드 요청 등 명시적으로 요청되는 리소스는 제약 조건에 의해 관리될 수 있습니다.

리소스 관리 제어 방식

Oracle Solaris 운영 체제에서 사용할 수 있는 세 가지 유형의 제어 방식은 제약 조건, 예약 및 분할입니다.

제약 조건 방식

제약 조건을 통해 관리자나 응용 프로그램 개발자는 작업 부하에 대한 특정 리소스 사용량의 한계를 설정할 수 있습니다. 한계를 알 경우 리소스 사용 모델링 시나리오는 보다 간단한 프로세스가 됩니다. 한계는 잘못된 동작을 하는 응용 프로그램을 제어하는 데도 사용할 수 있습니다. 이러한 응용 프로그램은 제어하지 않을 경우 규제되지 않은 리소스 요청을 통해 시스템 성능이나 가용성을 저하시킬 것입니다.

제약 조건은 응용 프로그램에 대해 복잡한 상태를 제시합니다. 응용 프로그램과 시스템 간의 관계는 응용 프로그램이 더 이상 작동할 수 없는 지점까지 수정할 수 있습니다. 이 위험을 완화할 수 있는 한 가지 접근 방법은 리소스 동작을 알지 못하는 응용 프로그램에 대해 제약 조건의 범위를 점차적으로 좁히는 것입니다. 6 장, “리소스 제어(개요)”에 설명된 리소스 제어는 제약 조건 방식을 제공합니다. 리소스 제약 조건을 인식하도록 새 응용 프로그램을 작성할 수 있지만, 일부 응용 프로그램 작성자는 이런 식으로 응용 프로그램을 작성하지 않을 수도 있습니다.

예약 방식

예약이란 특정 간격을 두고 할당 결정 순서를 정하는 것을 말합니다. 결정은 예측 가능한 알고리즘을 기반으로 하여 이루어집니다. 현재 할당이 필요하지 않은 응용 프로그램은 리소스를 다른 응용 프로그램에서 사용할 수 있도록 합니다. 예약 기반 리소스 관리를 통해 완결되지 않은 구성을 최대로 활용하는 반면, 중앙에서 미완결 또는 과다 완결된 시나리오에서 제어된 할당을 제공할 수 있습니다. 기본 알고리즘은 "제어된"이라는 용어가 어떻게 해석되는지를 정의합니다. 경우에 따라, 예약 알고리즘을 통해 모든 응용 프로그램이 리소스에 액세스할 수도 있습니다. 8 장, “FSS(Fair Share Scheduler)(개요)”에 설명된 FSS(Fare Share Scheduler)는 제어된 방식으로 CPU 리소스에 대한 응용 프로그램 액세스를 관리합니다.

분할 방식

분할은 작업 부하를 시스템의 사용 가능한 일부 리소스에 바인딩하는 데 사용됩니다. 이 바인딩을 통해 해당 작업 부하는 항상 알려진 리소스 양을 사용할 수 있게 됩니다. 12 장, "리소스 풀(개요)"에 설명된 리소스 풀 기능을 사용하면 시스템의 특정 부분으로 작업 부하를 제한할 수 있습니다.

분할을 사용하는 구성에서는 시스템 전체에서 과다 완결을 방지할 수 있습니다. 하지만 이러한 과다 완결을 방지하면서 높은 활용률을 실현할 수 있는 능력이 축소될 수 있습니다. 프로세서와 같은 예약된 리소스 그룹은 바인딩된 작업 부하가 유휴 상태일 때 다른 작업 부하에 사용될 수 없습니다.

리소스 관리 구성

리소스 관리 구성의 일부가 네트워크 이름 서비스에 배치될 수 있습니다. 이 기능을 사용하면 관리자가 각각의 시스템보다는 시스템 컬렉션에 대해 리소스 관리 제약 조건을 적용할 수 있습니다. 관련 작업이 공통 식별자를 공유할 수 있으므로 계정 데이터로부터 해당 작업의 총 사용량을 표로 작성할 수 있습니다.

리소스 관리 구성 및 작업 부하 지향 식별자는 2 장, "프로젝트 및 작업(개요)"에서 자세히 설명합니다. 이러한 식별자를 응용 프로그램 리소스 사용과 연결하는 확장 계정 기능은 4 장, "확장 계정(개요)"에서 설명합니다.

비전역 영역과 상호 작용

리소스 관리 기능을 영역과 함께 사용하여 응용 프로그램 환경을 좀더 구체화할 수 있습니다. 이러한 기능과 영역 간의 상호 작용은 이 설명서의 해당 절에서 설명합니다.

리소스 관리를 사용해야 하는 경우

리소스 관리를 사용하면 응용 프로그램에서 필요한 응답 시간을 제공할 수 있습니다.

리소스 관리는 리소스 활용을 향상시킬 수도 있습니다. 사용을 분류하고 우선 순위를 설정함으로써 한가한 시간대에 예약 용량을 효과적으로 사용할 수 있으므로 대체로 추가 처리 능력이 필요하지 않게 됩니다. 또한 부하 가변성으로 인해 리소스가 낭비되지 않습니다.

서버 통합

리소스 관리는 단일 서버에 여러 응용 프로그램을 통합하는 환경에 이상적입니다.

많은 시스템을 관리할 때 드는 비용과 복잡성으로 인해 더 크고 확장성이 더 나은 서버에 여러 응용 프로그램을 통합하는 것이 좋습니다. 개별 시스템에 각 작업 부하를 실행하는 대신에, 리소스 관리를 사용하여 시스템 내에서 작업 부하를 분리할 수 있습니다. 리소스 관리를 통해 단일 Oracle Solaris 시스템에서 유사하지 않은 여러 응용 프로그램을 실행 및 제어하여 전체 총 소유 비용을 낮출 수 있습니다.

인터넷과 응용 프로그램 서비스를 제공하려는 경우 리소스 관리를 사용하여 다음을 수행할 수 있습니다.

- 단일 시스템에서 여러 웹 서버를 호스트합니다. 각 웹 사이트에 대한 리소스 사용을 제어하고 다른 사이트의 잠재적 과다 사용으로부터 각 사이트를 보호할 수 있습니다.
- 결함이 있는 CGI(공통 게이트웨이 인터페이스) 스크립트에 CPU 리소스가 낭비되는 것을 방지할 수 있습니다.
- 잘못 작동하는 응용 프로그램의 모든 사용 가능한 가상 메모리 누출이 중지됩니다.
- 고객의 응용 프로그램이 동일 사이트에서 실행되는 다른 고객의 응용 프로그램에 영향을 받지 않게 됩니다.
- 동일 시스템에서 구분된 서비스 클래스나 레벨을 제공합니다.
- 청구 목적으로 회계 정보를 가져옵니다.

대규모 또는 가변 사용자 집단 지원

교육 기관과 같이 다양한 대규모 사용자 기반을 보유한 모든 시스템에서 리소스 관리 기능을 사용합니다. 여러 가지 작업 부하가 존재하는 경우 특정 프로젝트에 우선 순위를 부여하도록 소프트웨어를 구성할 수 있습니다.

예를 들면, 대형 증권 업체에서 트레이더는 계산을 수행하거나 쿼리를 실행하기 위해 간헐적으로 빠른 액세스를 필요로 합니다. 그러나 다른 시스템 사용자의 작업 부하는 보다 일관됩니다. 비례적으로 더 많은 처리 능력을 트레이더의 프로젝트에 할당하면 트레이더가 자신이 필요로 하는 응답성을 얻게 됩니다.

리소스 관리는 서버의 시스템 기능에 의존하는 클라이언트 시스템을 지원하기에도 이상적입니다. 이러한 플랫폼은 스마트 카드와 같은 입력 장치와 프레임 버퍼를 사용하는 Stateless 콘솔을 제공합니다. 실제 계산은 공유 서버에서 수행되므로 시간 공유 유형의 환경이 됩니다. 리소스 관리 기능을 사용하여 서버에서 사용자를 분리할 수 있습니다. 이렇게 하고 나면 과다한 부하를 생성하는 사용자가 하드웨어 리소스를 독점할 수 없게 되고, 따라서 다른 사용자에게 심각한 영향을 미칠 수 없게 됩니다.

리소스 관리 설정(작업 맵)

다음 작업 맵은 시스템에서 리소스 관리를 설정하기 위한 단계의 고급 레벨 개요를 제공합니다.

작업	설명	수행 방법
시스템에서 작업 부하를 식별하고 프로젝트별로 각 작업을 분류합니다.	/etc/project 파일, NIS 맵 또는 LDAP 디렉토리 서비스에서 프로젝트 항목을 만듭니다.	39 페이지 “project 데이터베이스”
시스템에서 작업 부하의 우선 순위를 지정합니다.	중용한 응용 프로그램을 결정합니다. 이러한 작업 부하에는 리소스에 대한 우선적인 액세스가 필요할 수 있습니다.	비즈니스 서비스 목표를 참조하십시오.
시스템에서 실시간 작업을 모니터링합니다.	성능 도구를 사용하여 시스템에서 실행 중인 작업 부하의 현재 리소스 사용을 확인합니다. 그러면 지정된 리소스에 대한 액세스를 제한하는지 아니면 특정 작업 부하를 다른 작업 부하와 분리해야 하는지 여부를 평가할 수 있습니다.	cpustat(1M), iostat(1M), mpstat(1M), prstat(1M), sar(1) 및 vmstat(1M) 매뉴얼 페이지
시스템에서 실행 중인 작업 부하를 임시로 수정합니다.	변경할 수 있는 값을 결정하려면 Oracle Solaris 시스템에서 사용할 수 있는 리소스 제어를 참조하십시오. 작업이나 프로세스가 실행 중인 동안 명령줄에서 값을 업데이트할 수 있습니다.	78 페이지 “사용 가능한 리소스 제어”, 84 페이지 “리소스 제어 값에 대한 전역 및 로컬 동작”, 89 페이지 “실행 중인 시스템에서 리소스 제어 값 임시 업데이트” 및 rctladm(1M) 및 prctl(1) 매뉴얼 페이지.
project 데이터베이스나 이름 지정 서비스 프로젝트 데이터베이스의 모든 프로젝트 항목에 대해 리소스 제어와 프로젝트 속성을 설정합니다.	/etc/project 파일이나 이름 지정 프로젝트 데이터베이스의 각 프로젝트 항목에는 하나 이상의 리소스 제어나 속성이 들어 있을 수 있습니다. 리소스 제어는 해당 프로젝트에 연결된 작업과 프로세스를 제한합니다. 리소스 제어에 지정되는 각 임계치 값의 경우, 해당 값에 도달할 때 수행해야 할 하나 이상의 작업을 연결할 수 있습니다. 명령줄 인터페이스를 사용하여 리소스 제어를 설정할 수 있습니다.	39 페이지 “project 데이터베이스”, 40 페이지 “로컬 /etc/project 파일 형식”, 78 페이지 “사용 가능한 리소스 제어”, 84 페이지 “리소스 제어 값에 대한 전역 및 로컬 동작” 및 8 장, “FSS(Fair Share Scheduler)(개요)”
프로젝트에 연결된 프로세스 컬렉션을 통해 물리적 메모리의 리소스 사용에 대한 상한값을 설정합니다.	리소스 상한값 적용 데몬은 /etc/project 파일에서 프로젝트의 rcap.max-rss 속성에 대해 정의된 물리적 메모리 리소스 상한값을 적용합니다.	39 페이지 “project 데이터베이스”와 10 장, “리소스 상한값 지원 데몬을 사용한 물리적 메모리 제어(개요)”

작업	설명	수행 방법
리소스 풀 구성을 만듭니다.	리소스 풀은 프로세서와 같은 시스템 리소스를 분할하기 위한 방법을 제공하며 재부트 간에도 이러한 분할 영역을 유지합니다. `project.pool` 속성을 `/etc/project` 파일의 각 항목에 추가할 수 있습니다.	39 페이지 "project 데이터베이스"와 12장, "리소스 풀(개요)"
FSS(Faire Share Scheduler)를 기본 시스템 스케줄러로 설정합니다.	모든 사용자 프로세스가 단일 CPU 시스템 안에 속하거나 프로세서 세트가 동일 예약 클래스에 속하도록 합니다.	113 페이지 "FSS 구성" 및 `dispadmin`(1M) 매뉴얼 페이지
확장 회계 기능을 활성화하여 작업이나 프로세스별로 리소스 사용을 모니터링 및 기록합니다.	확장 회계 기능을 사용하여 현재 리소스 제어를 평가하고 향후 작업 부하의 용량 요구 사항을 계획합니다. 시스템 전체의 총 사용량을 추적할 수 있습니다. 두 개 이상의 시스템에 걸친 관련 작업 부하에 대한 전체 사용 통계를 얻기 위해 몇 대의 시스템에서 프로젝트 이름을 공유할 수 있습니다.	68 페이지 "흐름, 프로세스, 작업 및 네트워크 구성 요소에 대해 확장 계정을 활성화하는 방법" 및 `acctadm`(1M) 매뉴얼 페이지
(옵션) 구성을 추가로 조정해야 하는 경우 계속해서 명령줄에서 값을 변경할 수 있습니다. 작업이나 프로세스가 실행 중인 동안 값을 변경할 수 있습니다.	프로젝트를 다시 시작하지 않고 기존 작업에 대한 수정 사항을 임시로 적용할 수 있습니다. 성능에 만족할 때까지 값을 조정합니다. 그런 다음 `/etc/project` 파일이나 이름 지정 서비스 프로젝트 데이터베이스에서 현재 값을 업데이트합니다.	89 페이지 "실행 중인 시스템에서 리소스 제어 값 임시 업데이트" 및 `rctladm`(1M) 및 `prctl`(1) 매뉴얼 페이지
(옵션) 확장 회계 데이터를 수집합니다.	활성 프로세스 및 활성 작업에 대한 확장 회계 레코드를 작성합니다. 생성된 파일을 계획, 지불 거절 및 청구 목적으로 사용할 수 있습니다. 사용자 정의된 보고 및 추출 스크립트를 개발할 수 있도록 하는 `libexacct`에 대한 Perl(Practical Extraction and Report Language) 인터페이스도 있습니다.	`wracct`(1M) 매뉴얼 페이지와 63 페이지 "`libexacct`에 대한 Perl 인터페이스"

◆ ◆ ◆ 2 장

2 프로젝트 및 작업(개요)

이 장에서는 Oracle Solaris 리소스 관리의 **프로젝트** 및 **작업** 기능에 대해 설명합니다. 프로젝트 및 작업은 작업 부하에 레이블을 지정하고 이들을 서로 구분하기 위해 사용됩니다.

이 장에서는 다음 항목을 다룹니다.

프로젝트 및 작업 기능을 사용하려면 3 장, "프로젝트 및 작업 관리"를 참조하십시오.

프로젝트 및 작업 기능

작업 부하 응답을 최적화하려면 먼저 분석할 시스템에서 실행 중인 작업 부하를 식별할 수 있어야 합니다. 이러한 정보는 순수한 프로세스 중심 방식 또는 사용자 중심 방식 중 하나를 단독으로 사용해서는 얻기 어려울 수 있습니다. Oracle Solaris 시스템에서는 작업 부하를 구분하고 식별하는 데 사용할 수 있는 프로젝트와 작업이라는 두 가지 추가 기능이 제공됩니다. **프로젝트**는 관련 작업에 대한 네트워크 차원의 관리 식별자를 제공합니다. **작업**은 프로세스 그룹을 작업 부하 구성 요소를 나타내는 관리 가능 엔티티로 수집합니다.

`project` 이름 서비스 데이터베이스에서 지정된 제어가 프로세스, 작업 및 프로젝트에서 설정됩니다. `fork` 및 `settaskid` 시스템 호출을 통해 프로세스 및 작업 제어가 상속되므로 프로젝트 내에서 생성되는 모든 프로세스 및 작업은 이러한 제어를 상속합니다. 이러한 시스템 호출에 대한 자세한 내용은 `fork(2)` 및 `settaskid(2)` 매뉴얼 페이지를 참조하십시오.

해당 프로젝트 또는 작업 구성원을 기준으로, 실행 중인 프로세스를 표준 Oracle Solaris 명령을 사용하여 조작할 수 있습니다. 확장 계정 기능을 통해 프로세스 사용도 및 작업 사용도 모두를 보고하고, 지배 프로젝트 식별자를 사용하여 각 레코드에 태그를 지정할

수 있습니다. 이 프로세스를 사용하여 오프라인 작업 부하 분석을 온라인 모니터링과 연관시킬 수 있습니다. project 이름 서비스 데이터베이스를 통해 여러 시스템에서 프로젝트 식별자를 공유할 수 있습니다. 따라서 여러 시스템에서 실행되는(또는 포괄하는) 관련된 작업 부하의 리소스 사용을 시스템 모두에 대해 최종적으로 분석할 수 있습니다.

프로젝트 식별자

프로젝트 식별자는 관련된 작업을 식별하기 위해 사용되는 관리 식별자입니다. 프로젝트 식별자는 사용자 및 그룹 식별자와 동등한 작업 부하 태그로 생각할 수 있습니다. 사용자나 그룹은 하나 이상의 프로젝트에 속할 수 있습니다. 이러한 프로젝트는 사용자(또는 사용자 그룹)의 참여가 허용된 작업 부하를 나타내기 위해 사용할 수 있습니다. 이러한 구성원은 예를 들면 사용 또는 초기 리소스 할당 등의 기반이 되는 차지백의 기준이 될 수 있습니다. 사용자에게 기본 프로젝트가 지정되어야 하지만 사용자가 실행하는 프로세스와 해당 사용자가 구성원으로 속해 있는 모든 프로젝트를 연결할 수 있습니다.

사용자의 기본 프로젝트 결정

시스템에 로그인하려면 사용자에게 기본 프로젝트가 지정되어야 합니다. 사용자가 기본 프로젝트에 지정된 사용자 또는 그룹 목록에 없는 경우에도 사용자는 자동으로 해당 기본 프로젝트의 구성원이 됩니다.

시스템의 각 프로세스가 프로젝트 구성원을 보유하므로 기본 프로젝트를 로그인 또는 다른 초기 프로세스에 지정하기 위한 알고리즘이 필요합니다. 알고리즘은 getprojent(3C) 매뉴얼 페이지에 설명되어 있습니다. 시스템에서는 지정된 단계에 따라 기본 프로젝트를 확인합니다. 기본 프로젝트를 찾을 수 없는 경우 사용자의 로그인 또는 프로세스 시작 요청이 거부됩니다.

시스템은 이러한 단계를 순차적으로 따라 사용자의 기본 프로젝트를 확인합니다.

1. 사용자에게 /etc/user_attr 확장 사용자 속성 데이터베이스에 정의된 project 속성을 가진 항목이 있는 경우 project 속성의 값이 기본 프로젝트입니다. user_attr(4) 매뉴얼 페이지를 참조하십시오.
2. 이름이 user.*user-id*인 프로젝트가 project 데이터베이스에 있는 경우 해당 프로젝트가 기본 프로젝트입니다. 자세한 내용은 project(4) 매뉴얼 페이지를 참조하십시오.
3. 이름이 group.*group-name*인 프로젝트가 project 데이터베이스에 있고, 여기서 *group-name*이 passwd 파일에 지정된 사용자의 기본 그룹 이름이면 해당 프로젝트가 기본 프로젝트입니다. passwd 파일에 대한 자세한 내용은 passwd(4) 매뉴얼 페이지를 참조하십시오.
4. 특수 프로젝트 default가 project 데이터베이스에 있는 경우 해당 프로젝트가 기본 프로젝트입니다.

이 논리는 getdefaultproj() 라이브러리 기능에서 제공됩니다. 자세한 내용은 getprojent(3PROJECT) 매뉴얼 페이지를 참조하십시오.

useradd 및 usermod 명령을 사용하여 사용자 속성 설정

-K 옵션 및 *key=value* 쌍과 함께 다음 명령을 사용하여 로컬 파일에서 사용자 속성을 설정할 수 있습니다.

useradd 사용자에 대한 기본 프로젝트 설정

usermod 사용자 정보 수정

로컬 파일에는 다음이 포함될 수 있습니다.

- /etc/group
- /etc/passwd
- /etc/project
- /etc/shadow
- /etc/user_attr

NIS와 같은 네트워크 이름 지정 서비스가 추가 항목으로 로컬 파일을 보완하기 위해 사용되는 경우 이러한 명령으로 네트워크 이름 서비스에서 제공하는 정보를 변경할 수 없습니다. 그러나 이 명령으로 외부 **이름 지정 서비스 데이터베이스**에 대해 다음을 확인할 수는 있습니다.

- 사용자 이름(또는 역할)의 고유성
- 사용자 ID의 고유성
- 지정된 그룹 이름의 존재

자세한 내용은 useradd(1M), usermod(1M) 및 user_attr(4) 매뉴얼 페이지를 참조하십시오.

project 데이터베이스

프로젝트 데이터를 로컬 파일 DNS(도메인 이름 시스템), NIS(네트워크 정보 서비스) 프로젝트 맵 또는 LDAP(Lightweight Directory Access Protocol) 디렉토리 서비스에 저장할 수 있습니다. /etc/project 파일 또는 이름 지정 서비스는 로그인 시 PAM(플러그 가능한 인증 모듈)의 계정 관리에 대한 모든 요청에 따라 사용자를 기본 프로젝트에 바인드하기 위해 사용됩니다.

주 – 프로젝트 데이터베이스의 항목에 대한 업데이트는 /etc/project 파일에 대한 것이든 네트워크 이름 지정 서비스의 데이터베이스 표현에 대한 것이든 관계없이 현재의 활성 프로젝트에는 적용되지 않습니다. 업데이트는 login 또는 newtask 명령 중 하나가 사용될 때 프로젝트에 연결된 새 작업에만 적용됩니다. 자세한 내용은 login(1) 및 newtask(1) 매뉴얼 페이지를 참조하십시오.

PAM 부속 시스템

ID를 변경하거나 설정하는 작업에는 시스템 로그인, rcp 또는 rsh 명령 호출, ftp 사용 또는 su 사용이 포함됩니다. 작업에 ID 변경 또는 설정이 포함되는 경우 인증, 계정 관리, 자격 증명 관리 및 세션 관리를 제공하기 위해 일련의 구성 가능한 모듈이 사용됩니다.

PAM에 대한 개요는 **Oracle Solaris 관리: 보안 서비스**의 15 장, "PAM 사용"을 참조하십시오.

이름 지정 서비스 구성

리소스 관리는 이름 지정 서비스 project 데이터베이스를 지원합니다. project 데이터베이스가 저장되는 위치는 /etc/nsswitch.conf 파일에서 정의됩니다. 기본적으로 files가 먼저 나열되지만 소스는 순서에 관계없이 나열될 수 있습니다.

```
project: files [nis] [ldap]
```

프로젝트 정보에 대한 소스가 둘 이상 나열되는 경우 nsswitch.conf 파일에서 첫번째로 나열된 소스에서 정보를 검색한 후 다음 소스를 검색하도록 루틴을 지정합니다.

/etc/nsswitch.conf 파일에 대한 자세한 내용은 **Oracle Solaris Administration: Naming and Directory Services**의 2 장, "Name Service Switch (Overview)" 및 nsswitch.conf(4)를 참조하십시오.

로컬 /etc/project 파일 형식

nsswitch.conf 파일에서 files를 project 데이터베이스의 소스로 선택한 경우 로그인 프로세스는 /etc/project 파일에서 프로젝트 정보를 검색합니다. 자세한 내용은 projects(1) 및 project(4) 매뉴얼 페이지를 참조하십시오.

project 파일에는 시스템에서 인식한 각 프로젝트에 대해 다음과 같은 형태의 한 행으로 된 항목이 포함되어 있습니다.

```
projname:projid:comment:user-list:group-list:attributes
```

필드 정의는 다음과 같습니다.

projname 프로젝트의 이름입니다. 이름은 영숫자, 밑줄(_), 하이픈(-) 및 점(.)으로 구성된 문자열이어야 합니다. 운영 체제에 대해 특별한 의미를 갖는 프로젝트를 위해 예약된 점(.)은 사용자의 기본 프로젝트 이름에만 사용할 수 있습니다. *projname*에는 콜론(:) 또는 개행 문자가 포함될 수 없습니다.

projid 시스템 내에서 프로젝트의 고유한 숫자 ID(PROJID)입니다. *projid* 필드의 최대값은 `UID_MAX(2147483647)`입니다.

comment 프로젝트의 설명입니다.

user-list 프로젝트에서 허용된 사용자의 쉼표로 구분된 목록입니다.

이 필드에서는 와일드카드를 사용할 수 있습니다. 별표(*)는 모든 사용자가 프로젝트에 연결할 수 있도록 허용합니다. 느낌표 뒤에 별표(!*)가 오면 프로젝트에서 모든 사용자가 제외됩니다. 느낌표(!) 다음에 사용자 이름이 오면 프로젝트에서 지정된 사용자가 제외됩니다.

group-list 프로젝트에서 허용된 사용자 그룹의 쉼표로 구분된 목록입니다.

이 필드에서는 와일드카드를 사용할 수 있습니다. 별표(*)는 모든 그룹이 프로젝트에 연결할 수 있도록 허용합니다. 느낌표 뒤에 별표(!*)가 오면 프로젝트에서 모든 그룹이 제외됩니다. 느낌표(!) 다음에 그룹 이름이 오면 프로젝트에서 지정된 그룹이 제외됩니다.

attributes 리소스 제어(6 장, “리소스 제어(개요)” 참조)와 같이 이름-값 쌍의 세미콜론으로 구분된 목록입니다. *name*은 객체 관련 속성을 지정하는 임의의 문자열이며, *value*는 해당 속성의 선택적 값입니다.

```
name[=value]
```

이름-값 쌍에서 이름에는 문자, 숫자, 밑줄 및 점만 사용할 수 있습니다. 점은 리소스 제어(rctl)의 범주와 하위 범주 간의 구분자로 사용됩니다. 속성 이름의 첫 글자는 문자여야 합니다. 이름은 대소문자를 구분합니다.

우선 순위를 설정하기 위해 값을 쉼표 및 괄호를 사용하여 구조화할 수 있습니다.

이름-값 쌍을 구분하기 위해 세미콜론을 사용합니다. 값 정의에는 세미콜론을 사용할 수 없습니다. 프로젝트 필드를 구분하기 위해 콜론을 사용합니다. 값 정의에는 콜론을 사용할 수 없습니다.

주 – 이 파일을 읽는 루틴에서 형식이 잘못된 항목을 발견하면 루틴이 정지됩니다. 잘못된 항목 다음에 지정되는 모든 프로젝트는 할당되지 않습니다.

이 예는 기본 `/etc/project` 파일을 보여 줍니다.

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
```

이 예는 끝부분에 추가된 프로젝트 항목이 있는 기본 /etc/project 파일을 보여 줍니다.

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
user.ml:2424:Lyle Personal:::
booksite:4113:Book Auction Project:ml,mp,jtd,kjh::
```

/etc/project 파일에 리소스 제어 및 속성을 추가할 수도 있습니다.

- 프로젝트에 대한 리소스 제어를 추가하려면 92 페이지 "리소스 제어 설정"을 참조하십시오.
- rcapd(1M)에 설명된 리소스 상한값 지원 데몬을 사용하여 프로젝트에 대한 물리적 메모리 리소스 상한값을 정의하려면 118 페이지 "프로젝트의 물리적 메모리 사용 제한을 위한 속성"을 참조하십시오.
- 프로젝트 항목에 project.pool 속성을 추가하려면 176 페이지 "구성 만들기"를 참조하십시오.

NIS에 대한 프로젝트 구성

NIS를 사용하는 경우 /etc/nsswitch.conf 파일에서 프로젝트에 대한 NIS 맵을 검색하도록 지정할 수 있습니다.

```
project: nis files
```

project.byname 또는 project.bynumber 중 하나인 NIS 맵은 /etc/project 파일과 동일한 형태를 갖습니다.

```
projname:projid:comment:user-list:group-list:attributes
```

자세한 내용은 **Oracle Solaris Administration: Naming and Directory Services**의 5 장, "Network Information Service (Overview)"를 참조하십시오.

LDAP에 대한 프로젝트 구성

LDAP를 사용하는 경우 /etc/nsswitch.conf 파일에서 프로젝트에 대한 LDAP project 데이터베이스를 검색하도록 지정할 수 있습니다.

```
project: ldap files
```

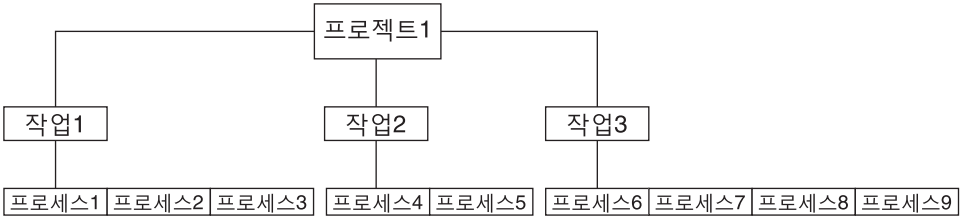
LDAP에 대한 자세한 내용은 **Oracle Solaris Administration: Naming and Directory Services**의 9 장, “Introduction to LDAP Naming Services (Overview)”를 참조하십시오. LDAP 데이터베이스 내 프로젝트 항목의 스키마에 대한 자세한 내용은 **Oracle Solaris Administration: Naming and Directory Services**의 “Oracle Solaris Schemas”를 참조하십시오.

작업 식별자

프로젝트에 성공적으로 로그인할 때마다 로그인 프로세스를 포함하는 새 **작업**이 만들어집니다. 작업은 일정 기간 동안의 일련의 작업을 나타내는 프로세스 컬렉티브입니다. 작업을 **작업 부하 구성 요소**로 볼 수도 있습니다. 각 작업에 작업 ID가 자동으로 지정됩니다.

각 프로세스는 한 작업의 구성원이며 각 작업에는 하나의 프로젝트가 연결됩니다.

그림 2–1 프로젝트 및 작업 트리



신호 전달과 같은 프로세스 그룹에 대한 모든 작업도 작업에서 지원됩니다. 작업을 **프로세서 세트**에 바인드하고, 작업에 대한 예약 우선 순위 및 클래스를 설정할 수도 있습니다. 이렇게 하면 작업의 모든 현재 및 이후 프로세스가 수정됩니다.

프로젝트가 연결될 때마다 작업이 만들어집니다. 다음 작업, 명령 및 기능이 작업을 만듭니다.

- 로그인
- `cron`
- `newtask`
- `setproject`
- `su`

다음 방법 중 하나를 사용하여 최종 작업을 만들 수 있습니다. 새 작업을 만들려는 추가 시도는 실패합니다.

- `newtask` 명령을 `-F` 옵션과 함께 사용할 수 있습니다.

- project 이름 지정 서비스 데이터베이스에서 프로젝트에 대한 task.final 속성을 설정할 수 있습니다. setproject를 사용하여 해당 프로젝트에서 생성된 모든 작업에는 TASK_FINAL 태그가 지정됩니다.

자세한 내용은 login(1), newtask(1), cron(1M), su(1M) 및 setproject(3PROJECT) 매뉴얼 페이지를 참조하십시오.

확장 계정 기능은 프로세스에 대한 계정 데이터를 제공할 수 있습니다. 데이터는 작업 레벨에서 통합됩니다.

프로젝트 및 작업과 함께 사용되는 명령

다음 표에 표시된 명령은 프로젝트 및 작업 기능에 대한 주요 관리 인터페이스를 제공합니다.

매뉴얼 페이지 참조	설명
projects(1)	사용자에 대한 프로젝트 구성원을 표시합니다. project 데이터베이스의 프로젝트를 나열합니다. 지정된 프로젝트에 대한 정보를 인쇄합니다. 프로젝트 이름이 제공되지 않은 경우 모든 프로젝트에 대한 정보가 표시됩니다. projects 명령을 -l 옵션과 함께 사용하여 상세 정보 출력을 인쇄합니다.
newtask(1)	사용자의 기본 셸 또는 지정된 명령을 실행하고 지정된 프로젝트에 속하는 새 작업에 실행 명령을 배치합니다. 작업 및 실행 중인 프로세스에 대한 프로젝트 바인딩을 변경하기 위해 newtask를 사용할 수도 있습니다. -F 옵션과 함께 사용하여 최종 작업을 만듭니다.
projadd(1M)	/etc/project 파일에 새 프로젝트 항목을 추가합니다. projadd 명령은 로컬 시스템에서만 프로젝트 항목을 만듭니다. projadd는 네트워크 이름 지정 서비스에서 제공하는 정보를 변경할 수 없습니다. 기본 파일 /etc/project 이외의 프로젝트 파일을 편집하기 위해 사용할 수 있습니다. project 파일을 검사하는 구문을 제공합니다. 프로젝트 속성을 검증 및 편집합니다. 확장된 값을 지원합니다.

매뉴얼 페이지 참조	설명
projmod(1M)	로컬 시스템에서 프로젝트에 대한 정보를 수정합니다. projmod는 네트워크 이름 지정 서비스에서 제공하는 정보를 변경할 수 없습니다. 그러나 명령을 사용하여 외부 이름 지정 서비스에 대해 프로젝트 이름 및 프로젝트 ID의 고유성을 확인할 수 있습니다. 기본 파일 /etc/project 이외의 프로젝트 파일을 편집하기 위해 사용할 수 있습니다. project 파일을 검사하는 구문을 제공합니다. 프로젝트 속성을 검증 및 편집합니다. 새 속성 추가, 속성에 값 추가 또는 속성 제거를 위해 사용할 수 있습니다. 확장된 값을 지원합니다. -A 옵션과 함께 사용하여 프로젝트 데이터베이스에서 찾은 리소스 제어 값을 활성 프로젝트에 적용할 수 있습니다. project 파일에 정의된 값과 일치하지 않는 기존 값은 제거됩니다.
projdel(1M)	로컬 시스템에서 프로젝트를 삭제합니다. projdel은 네트워크 이름 지정 서비스에서 제공하는 정보를 변경할 수 없습니다.
useradd(1M)	로컬 파일에 기본 프로젝트 정의를 추가합니다. -K *key=value* 옵션과 함께 사용하여 사용자 속성을 추가 또는 대체합니다.
userdel(1M)	로컬 파일에서 사용자의 계정을 삭제합니다.
usermod(1M)	시스템에서 사용자의 로그인 정보를 수정합니다. -K *key=value* 옵션과 함께 사용하여 사용자 속성을 추가 또는 대체합니다.

◆ ◆ ◆ 3 장

3 프로젝트 및 작업 관리

이 장에서는 Oracle Solaris 리소스 관리의 프로젝트 및 작업 기능을 사용하는 방법에 대해 설명합니다.

다음 항목을 다룹니다.

- 48 페이지 “명령 및 명령 옵션 예”
- 50 페이지 “프로젝트 관리”

프로젝트 및 작업 기능에 대한 개요는 2 장, “프로젝트 및 작업(개요)”을 참조하십시오.

주 – 영역이 설치되어 있는 Oracle Solaris 시스템에서 이러한 기능을 사용하는 경우 동일한 영역에 있는 프로세스만 이러한 명령이 비전역 영역에서 실행될 때 프로세스 ID를 가져오는 시스템 호출 인터페이스를 통해 볼 수 있습니다.

프로젝트 및 작업 관리(작업 맵)

작업	설명	수행 방법
프로젝트 및 작업과 함께 사용되는 명령 및 옵션의 예를 봅니다.	시스템에서 현재 실행 중인 프로세스 및 프로젝트에 대한 다양한 통계를 표시하는 작업 및 프로젝트 ID를 표시합니다.	48 페이지 “명령 및 명령 옵션 예”
프로젝트를 정의합니다.	`/etc/project` 파일에 프로젝트 항목을 추가하고 해당 항목의 값을 변경합니다.	50 페이지 “프로젝트를 정의하고 현재 프로젝트를 보는 방법”
프로젝트를 삭제합니다.	`/etc/project` 파일에서 프로젝트 항목을 제거합니다.	53 페이지 “`/etc/project` 파일에서 프로젝트를 삭제하는 방법”

작업	설명	수행 방법
project 파일 또는 프로젝트 데이터베이스를 검증합니다.	Check the syntax of the /etc/project 파일의 구문을 확인하거나 외부 이름 지정 서비스에 대해 프로젝트 이름 및 프로젝트 ID의 고유성을 확인합니다.	54 페이지 "/etc/project 파일의 내용을 검증하는 방법"
프로젝트 구성원 정보를 가져옵니다.	호출 프로세스의 현재 프로젝트 구성원을 표시합니다.	54 페이지 "프로젝트 구성원 정보를 가져오는 방법"
새 작업을 만듭니다.	newtask 명령을 사용하여 특정 프로젝트에서 새 작업을 만듭니다.	54 페이지 "새 작업을 만드는 방법"
실행 중인 프로세스와 서로 다른 작업 및 프로젝트를 연결합니다.	프로세스 번호를 지정된 프로젝트의 새 작업 ID와 연결합니다.	55 페이지 "실행 중인 프로세스를 새 작업으로 이동하는 방법"
프로젝트 속성을 추가하고 사용합니다.	프로젝트 데이터베이스 관리 명령을 사용하여 프로젝트 속성을 추가, 편집, 검증 및 제거합니다.	55 페이지 "프로젝트 속성 편집 및 검증"

명령 및 명령 옵션 예

이 절에서는 프로젝트 및 작업과 함께 사용되는 명령 및 옵션의 예를 제공합니다.

프로젝트 및 작업과 함께 사용되는 명령 옵션

ps 명령

ps 명령을 -o 옵션과 함께 사용하여 작업 및 프로젝트 ID를 표시합니다. 예를 들어, 프로젝트 ID를 보려면 다음을 입력합니다.

```
# ps -o user,pid,uid,projid
USER PID   UID  PROJID
jtd  89430 124  4113
```

id 명령

id 명령을 -p 옵션과 함께 사용하여 사용자 및 그룹 ID와 함께 현재 프로젝트 ID를 인쇄합니다. *user* 피연산자가 제공되는 경우 해당 사용자의 일반적인 로그인과 연결된 프로젝트가 인쇄됩니다.

```
#  id -p
uid=124(jtd) gid=10(staff) projid=4113(booksite)
```

pgrep 및 pkill 명령

특정 목록에 있는 프로젝트 ID를 가진 프로세스만 일치시키려면 pgrep 및 pkill 명령을 -J 옵션과 함께 사용합니다.

```
# pgrep -J projidlist
# pkill -J projidlist
```

특정 목록에 있는 작업 ID를 가진 프로세스만 일치시키려면 pgrep 및 pkill 명령을 -T 옵션과 함께 사용합니다.

```
# pgrep -T taskidlist
# pkill -T taskidlist
```

prstat 명령

해당 시스템에서 현재 실행 중인 프로세스 및 프로젝트에 대한 다양한 통계를 표시하려면 prstat 명령을 -J 옵션과 함께 사용합니다.

```
% prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 12905 root     4472K 3640K cpu0    59    0   0:00:01 0.4% prstat/1
   829 root       43M   33M sleep   59    0   0:36:23 0.1% Xorg/1
   890 gdm        88M   26M sleep   59    0   0:22:22 0.0% gdm-simple-gree/1
   686 root     3584K 2756K sleep   59    0   0:00:34 0.0% automountd/4
     5 root       0K    0K sleep   99  -20   0:02:43 0.0% zpool-rpool/138
  9869 root       44M   17M sleep   59    0   0:02:06 0.0% poold/9
   804 root     7104K 5968K sleep   59    0   0:01:28 0.0% intrd/1
   445 root     7204K 4680K sleep   59    0   0:00:38 0.0% nscd/33
   881 gdm      7140K 5912K sleep   59    0   0:00:06 0.0% gconfd-2/1
   164 root     2572K 1648K sleep   59    0   0:00:00 0.0% pfexecd/3
   886 gdm      7092K 4920K sleep   59    0   0:00:00 0.0% bonobo-activati/2
    45 netcfg   2252K 1308K sleep   59    0   0:00:00 0.0% netcfgd/2
   142 daemon   7736K 5224K sleep   59    0   0:00:00 0.0% kcfd/3
    43 root     3036K 2020K sleep   59    0   0:00:00 0.0% dlmgmtd/5
   405 root     6824K 5400K sleep   59    0   0:00:18 0.0% hald/5
PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
     1        4 4728K   19M   0.9%   0:00:01 0.4% user.root
     0      111  278M  344M    17%   1:15:02 0.1% system
    10        2 1884K 9132K   0.4%   0:00:00 0.0% group.staff
     3        3 1668K 6680K   0.3%   0:00:00 0.0% default

Total: 120 processes, 733 lwps, load averages: 0.01, 0.00, 0.00
```

해당 시스템에서 현재 실행 중인 프로세스 및 작업에 대한 다양한 통계를 표시하려면 prstat 명령을 -T 옵션과 함께 사용합니다.

```
% prstat -T
       PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 12907 root     4488K 3588K cpu0    59    0   0:00:00 0.3% prstat/1
   829 root       43M   33M sleep   59    0   0:36:24 0.1% Xorg/1
   890 gdm        88M   26M sleep   59    0   0:22:22 0.0% gdm-simple-gree/1
  9869 root       44M   17M sleep   59    0   0:02:06 0.0% poold/9
     5 root        0K    0K sleep   99  -20   0:02:43 0.0% zpool-rpool/138
```

```
   445 root     7204K 4680K sleep   59    0   0:00:38 0.0% nscd/33
   881 gdm      7140K 5912K sleep   59    0   0:00:06 0.0% gconfd-2/1
   164 root     2572K 1648K sleep   59    0   0:00:00 0.0% pfexecd/3
   886 gdm      7092K 4920K sleep   59    0   0:00:00 0.0% bonobo-activati/2
    45 netcfg   2252K 1308K sleep   59    0   0:00:00 0.0% netcfgd/2
   142 daemon   7736K 5224K sleep   59    0   0:00:00 0.0% kcfd/3
    43 root     3036K 2020K sleep   59    0   0:00:00 0.0% dlmgmtd/5
   405 root     6824K 5400K sleep   59    0   0:00:18 0.0% hald/5
   311 root     3488K 2512K sleep   59    0   0:00:00 0.0% picld/4
   409 root     4356K 2768K sleep   59    0   0:00:00 0.0% hald-addon-cpuf/1
TASKID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
  1401        2 2540K 8120K   0.4%   0:00:00 0.3% user.root
    94       15   84M  162M   7.9%   0:59:37 0.1% system
   561        1   37M   24M   1.2%   0:02:06 0.0% system
     0        2    0K    0K   0.0%   0:02:47 0.0% system
    46        1 4224K 5524K   0.3%   0:00:38 0.0% system
Total: 120 processes, 733 lwps, load averages: 0.01, 0.00, 0.00
```

주 – -J 및 -T 옵션은 함께 사용할 수 없습니다.

프로젝트 및 작업에서 cron 및 su 사용

cron 명령

cron 명령은 제출하는 사용자에 대한 해당 기본 프로젝트를 사용하여 별도의 작업에서 각 cron, at 및 batch 작업이 실행되도록 하기 위해 settaskid를 발행합니다. at 및 batch 명령은 현재 프로젝트 ID도 캡처하여 at 작업을 실행할 때 프로젝트 ID가 복원되도록 합니다.

su 명령

su 명령은 로그인 시뮬레이션의 일부로 새 작업을 만들어 대상 사용자의 기본 프로젝트를 연결합니다.

su 명령을 사용하여 사용자의 기본 프로젝트를 전환하려면 다음을 입력합니다.

```
# su - user
```

프로젝트 관리

▼ 프로젝트를 정의하고 현재 프로젝트를 보는 방법

이 예에서는 projadd 명령을 사용하여 프로젝트 항목을 추가하고 projmod 명령을 사용하여 해당 항목을 변경하는 방법을 보여 줍니다.

1 관리자로 전환합니다.

2 **projects -l을 사용하여 해당 시스템에서 기본 /etc/project 파일을 확인합니다.**

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

3 **이름이 *booksite*인 프로젝트를 추가합니다. 프로젝트 ID 번호 *4113*으로 이름이 *mark*인 사용자에게 프로젝트를 지정합니다.**

```
# projadd -U mark -p 4113 booksite
```

4 **/etc/project 파일을 다시 확인합니다.**

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
```

```
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
booksite
        projid : 4113
        comment: ""
        users  : mark
        groups : (none)
        attribs:
```

5 주석 필드에 프로젝트를 설명하는 주석을 추가합니다.

```
# projmod -c 'Book Auction Project' booksite
```

6 /etc/project 파일의 변경 사항을 확인합니다.

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
booksite
        projid : 4113
```

```
        comment: "Book Auction Project"
        users  : mark
        groups : (none)
        attribs:
```

참조 프로젝트, 작업 및 프로세스를 풀에 바인드하려면 172 페이지 "풀 속성 설정 및 풀에 바인드"를 참조하십시오.

▼ /etc/project 파일에서 프로젝트를 삭제하는 방법

이 예는 `projdel` 명령을 사용하여 프로젝트를 삭제하는 방법을 보여 줍니다.

1. **관리자로 전환합니다.**

2. **`projdel` 명령을 사용하여 *booksite* 프로젝트를 제거합니다.**

   `#` `projdel` *booksite*

3. **`/etc/project` 파일을 표시합니다.**

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

4 **사용자 *mark*로 로그인하여 `projects`를 입력하여 이 사용자에게 지정된 프로젝트를 확인합니다.**

```
# su - mark
# projects
default
```

/etc/project 파일의 내용을 검증하는 방법

편집 옵션이 지정되지 않은 경우 `projmod` 명령은 `project` 파일의 내용을 검증합니다.

NIS 맵을 검증하려면 다음을 입력합니다.

```
# ypcat project | projmod -f –
```

`/etc/project` 파일의 구문을 확인하려면 다음을 입력합니다.

```
# projmod -n
```

프로젝트 구성원 정보를 가져오는 방법

`-p` 플래그로 `id` 명령을 사용하여 호출 프로세스의 현재 프로젝트 구성원을 표시합니다.

```
$ id -p
uid=100(mark) gid=1(other) projid=3(default)
```

▼ 새 작업을 만드는 방법

1 **이 예에서는 대상 프로젝트인 *booksite*의 구성원으로 로그인합니다.**

2 **시스템 작업 ID를 가져오기 위해 `newtask` 명령을 `-v` (verbose) 옵션과 함께 사용하여 *booksite* 프로젝트에서 새 작업을 만듭니다.**

```
machine% newtask -v -p booksite
16
```

`newtask`를 실행하면 지정된 프로젝트에 새 작업이 생성되고 사용자의 기본 셸이 이 작업에 배치됩니다.

3 **호출 프로세스의 현재 프로젝트 구성원을 확인합니다.**

```
machine% id -p
uid=100(mark) gid=1(other) projid=4113(booksite)
```

이 프로세스가 이제 새 프로젝트의 구성원입니다.

▼ 실행 중인 프로세스를 새 작업으로 이동하는 방법

이 예에서는 실행 중인 프로세스와 다른 작업 및 새 프로젝트를 연결하는 방법을 보여 줍니다. 이 작업을 수행하려면 루트 사용자여야 하며, 필요한 권한 프로파일이 있거나 프로세스의 소유자이고 새 프로젝트의 구성원이어야 합니다.

1 관리자로 전환합니다.

주 – 프로세스의 소유자이거나 새 프로젝트의 구성원인 경우 이 단계를 건너뛸 수 있습니다.

---

2 ***book_catalog*** **프로세스의 프로세스 ID를 가져옵니다.**

```
# pgrep book_catalog
    8100
```

3 ***booksite*** **프로젝트에서 프로세스** ***8100*** **과 새 작업 ID를 연결합니다.**

```
# newtask -v -p booksite -c 8100
    17
```

`-c` 옵션은 `newtask`가 기존의 명명된 프로세스에서 실행된다는 것을 지정합니다.

4 ID 매핑을 처리할 작업을 확인합니다.

```
# pgrep -T 17
    8100
```

프로젝트 속성 편집 및 검증

`projadd` 및 `projmod` 프로젝트 데이터베이스 관리 명령을 사용하여 프로젝트 속성을 편집합니다.

`-K` 옵션은 속성의 대체 목록을 지정합니다. 속성은 세미콜론(`;`)으로 구분됩니다. `-K` 옵션과 `-a` 옵션이 함께 사용되는 경우 속성 또는 속성 값이 추가됩니다. `-K` 옵션과 `-r` 옵션이 함께 사용되는 경우 속성 또는 속성 값이 제거됩니다. `-K` 옵션과 `-s` 옵션이 함께 사용되는 경우 속성 또는 속성 값이 대체됩니다.

▼ 프로젝트에 속성 및 속성 값을 추가하는 방법

`projmod` 명령을 `-a` 및 `-K` 옵션과 함께 사용하여 프로젝트 속성에 값을 추가합니다. 속성이 없으면 생성됩니다.

1 관리자로 전환합니다.

2. **task.max-lwps 리소스 제어 속성을 값 없이 *myproject* 프로젝트에 추가합니다. 프로젝트에 들어가는 작업에는 속성에 대한 시스템 값만 있습니다.**

   ```
   # projmod -a -K task.max-lwps myproject
   ```

3. **그런 다음 *myproject* 프로젝트에서 task.max-lwps에 값을 추가할 수 있습니다. 값은 권한 레벨, 임계값 및 임계값 도달과 연결된 작업으로 구성됩니다.**

   ```
   # projmod -a -K "task.max-lwps=(priv,100,deny)" myproject
   ```

4. **리소스 제어는 여러 값을 가질 수 있으므로 동일한 옵션을 사용하여 기존의 값 목록에 다른 값을 추가할 수 있습니다.**

   ```
   # projmod -a -K "task.max-lwps=(priv,1000,signal=KILL)" myproject
   ```

   여러 값은 쉼표로 구분됩니다. 이제 task.max-lwps 항목이 다음과 같습니다.

   ```
   task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
   ```

▼ 프로젝트에서 속성 값을 제거하는 방법

이 절차는 다음 값을 사용합니다.

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

1. **관리자로 전환합니다.**

2. ***myproject* 프로젝트의 리소스 제어 task.max-lwps에서 속성 값을 제거하려면 projmod 명령을 -r 및 -K 옵션과 함께 사용합니다.**

   ```
   # projmod -r -K "task.max-lwps=(priv,100,deny)" myproject
   ```

   task.max-lwps가 다음과 같이 여러 값을 갖는 경우에는

   ```
   task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
   ```

   첫번째 일치하는 값이 제거됩니다. 그러면 결과가 다음과 같습니다.

   ```
   task.max-lwps=(priv,1000,signal=KILL)
   ```

▼ 프로젝트에서 리소스 제어 속성을 제거하는 방법

myproject 프로젝트에서 리소스 제어 task.max-lwps를 제거하려면 projmod 명령을 -r 및 -K 옵션과 함께 사용합니다.

1. **관리자로 전환합니다.**

2. ***myproject* 프로젝트에서 task.max-lwps 속성과 해당 값을 모두 제거합니다.**

   ```
   # projmod -r -K task.max-lwps myproject
   ```

▼ 프로젝트의 속성 및 속성 값을 대체하는 방법

myproject 프로젝트에서 `task.max-lwps` 속성을 다른 값으로 대체하려면 `projmod` 명령을 `-s` 및 `-K` 옵션과 함께 사용합니다. 속성이 없으면 생성됩니다.

1. **관리자로 전환합니다.**
2. **표시된 새 값으로 현재 `task.max-lwps` 값을 대체합니다.**

   ```
   # projmod -s -K "task.max-lwps=(priv,100,none),(priv,120,deny)" myproject
   ```

   결과가 다음과 같습니다.

   ```
   task.max-lwps=(priv,100,none),(priv,120,deny)
   ```

▼ 리소스 제어 속성의 기존 값을 제거하는 방법

1. **관리자로 전환합니다.**
2. ***myproject* 프로젝트에서 `task.max-lwps`의 현재 값을 제거하려면 다음을 입력합니다.**

   ```
   # projmod -s -K task.max-lwps myproject
   ```

◆ ◆ ◆ 4 장

4 확장 계정(개요)

2 장, "프로젝트 및 작업(개요)"에 설명되어 있는 프로젝트 및 작업 기능을 사용하여 작업 부하에 레이블을 지정하고 분리하여 각 작업 부하별로 리소스 사용을 모니터할 수 있습니다. **확장 계정** 부속 시스템을 사용하여 프로세스와 작업 모두에서 리소스 사용 통계의 세부 집합을 캡처할 수 있습니다.

이 장에서는 다음 내용을 다룹니다.

확장 계정 사용을 시작하려면 68 페이지 "흐름, 프로세스, 작업 및 네트워크 구성 요소에 대해 확장 계정을 활성화하는 방법"으로 건너뜁니다.

확장 계정 소개

확장 계정 부속 시스템은 작업이 수행된 프로젝트의 사용 레코드에 레이블을 지정합니다. 시스템의 네트워크 흐름 정보를 캡처하기 위해 확장 계정을 **Oracle Solaris 관리: IP 서비스**의 31 장, "흐름 계산 및 통계 수집 사용(작업)"에 설명되어 있는 IPQoS(Internet Protocol Quality of Service) 흐름 계정 모듈과 함께 사용할 수도 있습니다.

리소스 관리 방식을 적용하려면 먼저 시스템에 있는 다양한 작업 부하의 리소스 사용 요구의 특징을 파악할 수 있어야 합니다. Oracle Solaris 운영 체제의 확장 계정 기능은 다음에 대한 시스템 및 네트워크 리소스 사용을 기록하는 유연한 방법을 제공합니다.

- 작업
- 프로세스
- IPQoS `flowacct` 모듈에서 제공된 선택기. 자세한 내용은 `ipqos`(7IPP)를 참조하십시오.

- 네트워크 관리. dladm(1M) 및 flowadm(1M)을 참조하십시오.

실시간으로 시스템 사용을 측정할 수 있도록 하는 온라인 모니터링 도구와 달리 확장 계정을 사용하면 사용 기록을 검사할 수 있습니다. 그런 다음 향후 작업 부하에 필요한 용량 요구 사항을 평가할 수 있습니다.

제공된 확장 계정 데이터를 사용하여 리소스 차지백, 작업 부하 모니터링 또는 용량 계획을 위한 소프트웨어를 개발 또는 구매할 수 있습니다.

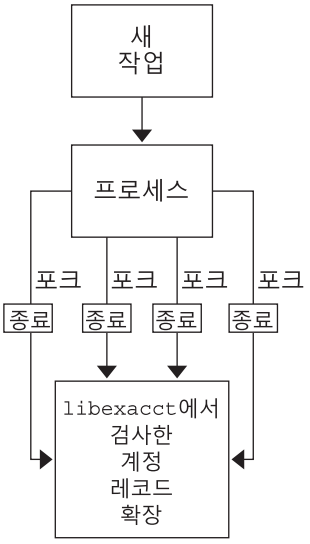
확장 계정이 작동하는 방법

Oracle Solaris 운영 체제의 확장 계정 기능은 계정 데이터를 포함하기 위해 버전이 지정된 확장 가능 파일 형식을 사용합니다. 이 데이터 형식을 사용하는 파일은 포함된 라이브러리, libexacct(libexacct(3LIB) 참조)에서 제공되는 API를 사용하여 액세스하거나 만들 수 있습니다. 그러면 이러한 파일은 확장 계정이 지원되는 모든 플랫폼에서 분석할 수 있으며 해당 데이터를 용량 계획 및 차지백에 사용할 수 있습니다.

확장 계정이 활성화되어 있는 경우, libexacct API에서 검사할 수 있는 통계가 수집됩니다. libexacct는 앞으로 또는 뒤로의 exacct 파일 검사를 허용합니다. API는 libexacct에 의해 생성된 타사 파일뿐 아니라 커널에 의해 생성된 파일도 지원합니다. 사용자 정의된 보고 및 추출 스크립트를 개발할 수 있는 libexacct에 대한 Perl(Practical Extraction and Report Language) 인터페이스가 있습니다. 63 페이지 “libexacct에 대한 Perl 인터페이스”를 참조하십시오.

예를 들어, 확장 계정이 지원되는 경우 작업은 해당 구성원 프로세스의 리소스 사용 집계를 추적합니다. 작업 완료 시 작업 계정 레코드가 기록됩니다. 실행 중인 프로세스 및 작업에 대한 중간 레코드도 기록할 수 있습니다. 작업에 대한 자세한 내용은 2 장, “프로젝트 및 작업(개요)”을 참조하십시오.

그림 4–1 활성화된 확장 계정으로 작업 추적



확장 가능한 형식

확장 계정 형식은 레거시 시스템 계정 소프트웨어 형식보다 대체로 더 많이 확장 가능합니다. 확장 계정은 릴리스 간에, 심지어 시스템 작동 중에도 시스템에서 계정 메트릭을 추가 또는 제거할 수 있습니다.

주 – 확장 계정 및 레거시 시스템 계정 소프트웨어 모두 동시에 시스템에서 활성화할 수 있습니다.

---

exacct 레코드 및 형식

exacct 레코드의 생성을 허용하는 루틴은 두 가지 용도를 지원합니다.

- 타사 exacct 파일을 생성할 수 있도록 합니다.
- putacct 시스템 호출(getacct(2) 참조)을 사용하여 커널 계정 파일에 내장되는 태그 처리 레코드를 작성할 수 있도록 합니다.

주 – Perl 인터페이스에서도 putacct 시스템 호출을 사용할 수 있습니다.

이 형식은 모든 변경에 대해 명시적 버전 변경을 요구하지 않고도 서로 다른 형태의 계정 레코드를 캡처할 수 있도록 허용합니다. 계정 데이터를 사용하는 잘 작성된 응용 프로그램은 이해하지 못하는 데이터를 무시해야 합니다.

libexacct 라이브러리는 exacct 형식으로 파일을 변환 및 생성합니다. 이 라이브러리가 exacct 형식의 파일에 대해 지원되는 **유일한** 인터페이스입니다.

주 – getacct, putacct 및 wracct 시스템 호출은 흐름에 적용되지 않습니다. IPQoS 흐름 계정이 구성되어 있는 경우 커널은 흐름 레코드를 만들고 이를 파일에 기록합니다.

영역이 설치된 Oracle Solaris 시스템에서 확장 계정 사용

확장 계정 부속 시스템은 전역 영역에서 실행될 때 전체 시스템(비전역 영역 포함)에 대한 정보를 수집 및 보고합니다. 전역 관리자 또는 zonecfg 유틸리티를 통해 적합한 인증을 받은 사용자는 각 영역을 기준으로 리소스 사용을 결정할 수도 있습니다. 자세한 내용은 336 페이지 "영역이 설치된 시스템의 확장된 계정"을 참조하십시오.

확장 계정 구성

/var/adm/exacct 디렉토리는 확장 계정 데이터가 저장되는 표준 위치입니다. acctadm 명령을 사용하여 프로세스 및 작업 계정 데이터 파일에 대해 다른 위치를 지정할 수 있습니다. 자세한 내용은 acctadm(1M)을 참조하십시오.

확장 계정 시작 및 지속적 사용

acctadm(1M)에 설명되어 있는 acctadm 명령은 smf(5)에 설명되어 있는 Oracle Solaris SMF(서비스 관리 기능) 서비스를 통해 확장 계정을 시작합니다.

확장 계정 구성은 SMF 저장소에 저장됩니다. 구성은 서비스 인스턴스에 의해 부트 시에 각 계정 유형별로 복원됩니다. 각 확장 계정 유형은 SMF 서비스의 별도 인스턴스로 표현됩니다.

```
svc:/system/extended-accounting:flow
   흐름 계정

svc:/system/extended-accounting:process
   프로세스 계정

svc:/system/extended-accounting:task
   작업 계정

svc:/system/extended-accounting:net
   네트워크 계정
```

acctadm(1M)을 사용하여 확장 계정을 사용으로 설정하면 서비스 인스턴스가 사용으로 설정되어 있지 않았으면 사용으로 설정되어 확장 계정 구성이 다음 부트 시에

복원됩니다. 마찬가지로, 구성에 의해 계정이 서비스에 사용되지 않게 되면 서비스 인스턴스도 사용되지 않습니다. 필요에 따라 acctadm에 의해 인스턴스가 사용되거나 사용되지 않습니다.

리소스에 대한 확장 계정을 영구적으로 활성화하려면 다음을 실행합니다.

```
# acctadm -e resource_list
```

*resource_list*는 리소스 또는 리소스 그룹의 쉼표로 구분된 목록입니다.

레코드

acctadm 명령은 기존 /var/adm/exacct 파일에 새로운 레코드를 추가합니다.

확장 계정과 함께 사용되는 명령

명령 참조	설명
acctadm(1M)	확장 계정 기능의 다양한 속성을 수정하고, 확장 계정을 중지 및 시작하며, 프로세스, 작업, 흐름 및 네트워크에 대해 추적할 계정 속성을 선택하는 데 사용됩니다.
wracct(1M)	활성 프로세스 및 활성 작업에 대한 확장 계정 레코드를 기록합니다.
lastcomm(1)	이전에 호출된 명령을 표시합니다. lastcomm은 표준 계정 프로세스 데이터 또는 확장 계정 프로세스 데이터 중 하나를 사용할 수 있습니다.

작업 및 프로젝트와 연관된 명령에 대한 자세한 내용은 48 페이지 “명령 및 명령 옵션 예”를 참조하십시오. IPQoS 흐름 계정에 대한 자세한 내용은 ipqosconf(1M)을 참조하십시오.

libexacct에 대한 Perl 인터페이스

Perl 인터페이스를 사용하여 exacct 프레임워크에서 생성된 계정 파일을 읽을 수 있는 Perl 스크립트를 만들 수 있습니다. 또한 exacct 파일을 쓸 수 있는 Perl 스크립트를 작성할 수 있습니다.

인터페이스는 기능적으로 기본 C API와 동일합니다. 가능한 경우 기본 C API에서 가져온 데이터가 Perl 데이터 유형으로 표시됩니다. 이 인터페이스를 통해 데이터에 좀 더 쉽게 액세스할 수 있고 버퍼 압축 및 압축 풀기 작업이 필요 없게 되었습니다. 또한 모든 메모리 관리가 Perl 라이브러리에서 수행됩니다.

다양한 프로젝트, 작업 및 exacct 관련 기능이 그룹으로 분리됩니다. 각 기능 그룹은 별도의 Perl 모듈에 위치합니다. 각 모듈은 Oracle Solaris 표준 Sun::Solaris:: Perl 패키지

접두어로 시작됩니다. Perl exacct 라이브러리에서 제공된 모든 클래스는 Sun::Solaris::Exacct 모듈 아래에 있습니다.

기본 libexacct(3LIB) 라이브러리는 exacct 형식 파일, 카탈로그 태그 및 exacct 객체에 대한 작업을 제공합니다. exacct 객체는 다음 두 가지 유형으로 나뉩니다.

- 단일 데이터 값(스칼라)인 Item
- 항목의 목록인 Group

다음 표에 각 모듈이 요약되어 있습니다.

모듈(공백을 포함할 수 없음)	설명	자세한 정보
Sun::Solaris::Project	이 모듈은 프로젝트 조작 기능 getprojid(2), endprojent(3PROJECT), fgetprojent(3PROJECT), getdefaultproj(3PROJECT), getprojbyid(3PROJECT), getprojbyname(3PROJECT), getprojent(3PROJECT), getprojidbyname(3PROJECT), inproj(3PROJECT), project_walk(3PROJECT), setproject(3PROJECT) 및 setprojent(3PROJECT)에 액세스하기 위한 기능을 제공합니다.	Project(3PERL)
Sun::Solaris::Task	이 모듈은 작업 조작 기능 gettaskid(2) 및 settaskid(2)를 액세스하기 위한 기능을 제공합니다.	Task(3PERL)
Sun::Solaris::Exacct	이 모듈은 최상위 레벨 exacct 모듈입니다. 이 모듈은 exacct 관련 시스템 호출 getacct(2), putacct(2) 및 wracct(2)에 액세스하기 위한 기능을 제공합니다. 이 모듈은 또한 libexacct(3LIB) 라이브러리 기능 ea_error(3EXACCT)에 액세스하기 위한 기능도 제공합니다. 모든 exacct EO_*, EW_*, EXR_*, P_* 및 TASK_* 매크로 상수도 이 모듈에서 제공됩니다.	Exacct(3PERL)
Sun::Solaris::Exacct::Catalog	이 모듈은 exacct 카탈로그 태그의 비트 필드에 액세스하기 위한 객체 지향 메소드를 제공합니다. 이 모듈은 또한 EXC_*, EXD_* 및 EXD_* 매크로의 상수에 대한 액세스도 제공합니다.	Exacct::Catalog(3PERL)

모듈(공백을 포함할 수 없음)	설명	자세한 정보
Sun::Solaris::Exacct:: File	이 모듈은 libexacct 계정 파일 기능 ea_open(3EXACCT), ea_close(3EXACCT), ea_get_creator(3EXACCT), ea_get_hostname(3EXACCT), ea_next_object(3EXACCT), ea_previous_object(3EXACCT) 및 ea_write_object(3EXACCT)에 액세스하기 위한 객체 지향 메소드를 제공합니다.	Exacct::File(3PERL)
Sun::Solaris::Exacct:: Object	이 모듈은 개별 exacct 계정 파일 객체에 액세스하기 위한 객체 지향 메소드를 제공합니다. exacct 객체는 해당 Sun::Solaris::Exacct::Object 하위 클래스로 블레스되어 불명확한 참조로 표현됩니다. 이 모듈은 객체 유형 Item 및 Group으로 추가로 나뉩니다. 이 레벨에 ea_match_object_catalog(3EXACCT) 및 ea_attach_to_object(3EXACCT) 기능에 액세스하기 위한 메소드가 있습니다.	Exacct::Object(3PERL)
Sun::Solaris::Exacct:: Object::Item	이 모듈은 개별 exacct 계정 파일 Item에 액세스하기 위한 객체 지향 메소드를 제공합니다. 이러한 유형의 객체는 Sun::Solaris::Exacct::Object에서 상속됩니다.	Exacct::Object::Item(3PERL)
Sun::Solaris::Exacct:: Object::Group	이 모듈은 개별 exacct 계정 파일 Group에 액세스하기 위한 객체 지향 메소드를 제공합니다. 이러한 유형의 객체는 Sun::Solaris::Exacct::Object에서 상속됩니다. 이러한 객체는 ea_attach_to_group(3EXACCT) 기능에 대한 액세스를 제공합니다. Group 내에 포함된 Item은 Perl 배열로 표시됩니다.	Exacct::Object::Group(3PERL)
Sun::Solaris::Kstat	이 모듈은 kstat 기능에 대한 Perl 연결 해시 인터페이스를 제공합니다. 이 모듈의 사용 예는 /bin/kstat에서 찾을 수 있으며 Perl로 작성되어 있습니다.	Kstat(3PERL)

이전 표에서 설명된 모듈의 사용 방법을 보여 주는 예는 71 페이지 “libexacct에 대한 Perl 인터페이스 사용”을 참조하십시오.

◆ ◆ ◆ 5 장

5 확장 계정 관리(작업)

이 장에서는 확장 계정 부속 시스템을 관리하는 방법에 대해 설명합니다.

확장 계정 부속 시스템에 대한 개요는 4 장, “확장 계정(개요)”을 참조하십시오.

확장 계정 기능 관리(작업 맵)

작업	설명	수행 방법
확장 계정 기능을 활성화합니다.	확장 계정을 사용하여 시스템에서 실행 중인 각 프로젝트별 리소스 사용을 모니터할 수 있습니다. **확장 계정** 부속 시스템을 사용하여 작업, 프로세스 및 흐름에 대한 기록 데이터를 캡처할 수 있습니다.	68 페이지 “흐름, 프로세스, 작업 및 네트워크 구성 요소에 대해 확장 계정을 활성화하는 방법”
확장 계정 상태를 표시합니다.	확장 계정 기능의 상태를 결정합니다.	69 페이지 “확장 계정 상태를 표시하는 방법”
사용 가능한 계정 리소스를 확인합니다.	시스템에서 사용 가능한 계정 리소스를 확인합니다.	69 페이지 “사용 가능한 계정 리소스를 보는 방법”
흐름, 프로세스, 작업 및 네트워크 계정 인스턴스를 비활성화합니다.	확장 계정 기능을 해제합니다.	70 페이지 “프로세스, 작업, 흐름 및 네트워크 관리 계정을 비활성화하는 방법”
확장 계정 기능에 Perl 인터페이스를 사용합니다.	Perl 인터페이스를 사용하여 사용자 정의된 보고 및 추출 스크립트를 개발할 수 있습니다.	71 페이지 “libexacct에 대한 Perl 인터페이스 사용”

확장 계정 기능 사용

사용자는 관리되는 계정 유형에 대한 해당 권한 프로파일이 있는 경우 확장 계정을 관리(계정 시작, 계정 중지 및 계정 구성 매개변수 변경)할 수 있습니다.

- 확장 계정 흐름 관리
- 프로세스 관리
- 작업 관리
- 네트워크 관리

▼ 흐름, 프로세스, 작업 및 네트워크 구성 요소에 대해 확장 계정을 활성화하는 방법

작업, 프로세스, 흐름 및 네트워크 구성 요소에 대해 확장 계정을 활성화하려면 `acctadm` 명령을 사용합니다. `acctadm`에 대한 선택적 최종 매개변수는 명령이 확장 계정 기능의 흐름, 프로세스, 시스템 작업 또는 네트워크 계정 구성 요소에서 작동할지 여부를 나타냅니다.

주 – 역할에는 권한 부여 및 권한이 있는 명령이 포함됩니다. Oracle Solaris의 RBAC(역할 기반 액세스 제어) 기능을 통해 역할을 만들고 사용자에게 역할을 지정하는 방법에 대한 자세한 내용은 **시스템 관리 설명서: 보안 서비스**의 RBAC 관리(작업 맵)를 참조하십시오.

---

1. **관리자로 전환합니다.**
2. **프로세스에 대해 확장 계정을 활성화합니다.**

   ```
   # acctadm -e extended -f /var/adm/exacct/proc process
   ```

3. **작업에 대해 확장 계정을 활성화합니다.**

   ```
   # acctadm -e extended,mstate -f /var/adm/exacct/task task
   ```

4. **흐름에 대해 확장 계정을 활성화합니다.**

   ```
   # acctadm -e extended -f /var/adm/exacct/flow flow
   ```

5. **네트워크에 대해 확장 계정을 활성화합니다.**

   ```
   # acctadm -e extended -f /var/adm/exacct/net net
   ```

   `dladm` 및 `flowadm` 명령에 의해 관리되는 링크 및 흐름에서 `acctadm`을 실행합니다.

참조 자세한 내용은 `acctadm(1M)`을 참조하십시오.

확장 계정 상태를 표시하는 방법

acctadm을 인수 없이 입력하여 확장 계정 기능의 현재 상태를 표시할 수 있습니다.

```
machine% acctadm
                Task accounting: active
           Task accounting file: /var/adm/exacct/task
          Tracked task resources: extended
        Untracked task resources: none
              Process accounting: active
         Process accounting file: /var/adm/exacct/proc
       Tracked process resources: extended
     Untracked process resources: host
                Flow accounting: active
           Flow accounting file: /var/adm/exacct/flow
          Tracked flow resources: extended
        Untracked flow resources: none
```

이전 예에서는 시스템 작업 계정이 확장 모드 및 mstate 모드에서 활성화되어 있습니다. 프로세스 및 흐름 계정은 확장 모드에서 활성화되어 있습니다.

주 – 확장 계정의 컨텍스트에서 microstate(mstate)는 microstate 프로세스 전환과 연관된 확장 데이터를 나타냅니다. 이 데이터는 프로세스 사용 파일(proc(4) 참조)에서 제공됩니다. 이 데이터는 대체로 기본 또는 확장 레코드보다 프로세스의 작업에 대한 정보를 자세히 제공합니다.

사용 가능한 계정 리소스를 보는 방법

사용 가능한 리소스는 시스템 및 플랫폼별로 다를 수 있습니다. acctadm 명령을 -r 옵션과 함께 사용하여 해당 시스템에서 사용 가능한 계정 리소스 그룹을 확인할 수 있습니다.

```
machine% acctadm -r
process:
extended pid,uid,gid,cpu,time,command,tty,projid,taskid,ancpid,wait-status,zone,flag,
memory,mstate     displays as one line
basic    pid,uid,gid,cpu,time,command,tty,flag
task:
extended taskid,projid,cpu,time,host,mstate,anctaskid,zone
basic    taskid,projid,cpu,time
flow:
extended
saddr,daddr,sport,dport,proto,dsfield,nbytes,npkts,action,ctime,lseen,projid,uid
basic    saddr,daddr,sport,dport,proto,nbytes,npkts,action
net:
  extended name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
  priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
  oerrpkts,saddr,daddr,sport,dport,protocol,dsfield
  basic    name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
```

```
priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
oerrpkts
```

▼ 프로세스, 작업, 흐름 및 네트워크 관리 계정을 비활성화하는 방법

프로세스, 작업, 흐름 및 네트워크 계정을 비활성화하려면 `acctadm` 명령을 `-x` 옵션과 함께 사용하여 개별적으로 각각을 해제합니다.

1. **관리자로 전환합니다.**

2. **프로세스 계정을 해제합니다.**

   ```
   # acctadm -x process
   ```

3. **작업 계정을 해제합니다.**

   ```
   # acctadm -x task
   ```

4. **흐름 계정을 해제합니다.**

   ```
   # acctadm -x flow
   ```

5. **네트워크 관리 계정을 해제합니다.**

   ```
   # acctadm -x net
   ```

6. **작업 계정, 프로세스 계정, 흐름 및 네트워크 계정이 해제되었는지 확인합니다.**

   ```
      # acctadm
              Task accounting: inactive
         Task accounting file: none
       Tracked task resources: none
     Untracked task resources: extended
            Process accounting: inactive
      Process accounting file: none
    Tracked process resources: none
  Untracked process resources: extended
              Flow accounting: inactive
         Flow accounting file: none
       Tracked flow resources: none
     Untracked flow resources: extended
              Net accounting: inactive
         Net accounting file: none
       Tracked Net resources: none
     Untracked Net resources: extended
   ```

libexacct에 대한 Perl 인터페이스 사용

exacct 객체의 내용을 반복적으로 인쇄하는 방법

다음 코드를 사용하여 exacct 객체의 내용을 반복적으로 인쇄할 수 있습니다. 이 기능은 라이브러리에서 Sun::Solaris::Exacct::Object::dump() 기능으로 제공됩니다. 또한 이 기능은 ea_dump_object() 편의 기능을 통해서도 사용할 수 있습니다.

```
sub dump_object
     {
             my ($obj, $indent) = @_;
             my $istr = '  ' x $indent;

             #
             # Retrieve the catalog tag.  Because we are
             # doing this in an array context, the
             # catalog tag will be returned as a (type, catalog, id)
             # triplet, where each member of the triplet will behave as
             # an integer or a string, depending on context.
             # If instead this next line provided a scalar context, e.g.
             #    my $cat  = $obj->catalog()->value();
             # then $cat would be set to the integer value of the
             # catalog tag.
             #
             my @cat = $obj->catalog()->value();

             #
             # If the object is a plain item
             #
             if ($obj->type() == &EO_ITEM) {
                     #
                     # Note: The '%s' formats provide s string context, so
                     # the components of the catalog tag will be displayed
                     # as the symbolic values. If we changed the '%s'
                     # formats to '%d', the numeric value of the components
                     # would be displayed.
                     #
                     printf("%sITEM\n%s  Catalog = %s|%s|%s\n",
                        $istr, $istr, @cat);
                     $indent++;

                     #
                     # Retrieve the value of the item.  If the item contains
                     # in turn a nested exacct object (i.e., an item or
                     # group),then the value method will return a reference
                     # to the appropriate sort of perl object
                     # (Exacct::Object::Item or Exacct::Object::Group).
                     # We could of course figure out that the item contained
                     # a nested item orgroup by examining the catalog tag in
                     # @cat and looking for a type of EXT_EXACCT_OBJECT or
                     # EXT_GROUP.
                     #
                     my $val = $obj->value();
                     if (ref($val)) {
```

```
                        # If it is a nested object, recurse to dump it.
                        dump_object($val, $indent);
                } else {
                        # Otherwise it is just a 'plain' value, so
                        # display it.
                        printf("%s  Value = %s\n", $istr, $val);
                }

        #
        # Otherwise we know we are dealing with a group.  Groups
        # represent contents as a perl list or array (depending on
        # context), so we can process the contents of the group
        # with a 'foreach' loop, which provides a list context.
        # In a list context the value method returns the content
        # of the group as a perl list, which is the quickest
        # mechanism, but doesn't allow the group to be modified.
        # If we wanted to modify the contents of the group we could
        # do so like this:
        #    my $grp = $obj->value();   # Returns an array reference
        #    $grp->[0] = $newitem;
        # but accessing the group elements this way is much slower.
        #
        } else {
                printf("%sGROUP\n%s  Catalog = %s|%s|%s\n",
                    $istr, $istr, @cat);
                $indent++;
                # 'foreach' provides a list context.
                foreach my $val ($obj->value()) {
                        dump_object($val, $indent);
                }
                printf("%sENDGROUP\n", $istr);
        }
}
```

새 그룹 레코드를 만들고 이를 파일에 쓰는 방법

이 스크립트를 사용하여 새 그룹 레코드를 만들고 이를 /tmp/exacct라는 파일에 씁니다.

```
#!/usr/bin/perl

use strict;
use warnings;
use Sun::Solaris::Exacct qw(:EXACCT_ALL);
# Prototype list of catalog tags and values.
     my @items = (
             [ &EXT_STRING | &EXC_DEFAULT | &EXD_CREATOR      => "me"       ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_PID     => $$         ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_UID     => $<         ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_GID     => $(         ],
             [ &EXT_STRING | &EXC_DEFAULT | &EXD_PROC_COMMAND => "/bin/rec" ],
     );

     # Create a new group catalog object.
     my $cat = ea_new_catalog(&EXT_GROUP | &EXC_DEFAULT | &EXD_NONE)

     # Create a new Group object and retrieve its data array.
```

```
my $group = ea_new_group($cat);
my $ary = $group->value();

# Push the new Items onto the Group array.
foreach my $v (@items) {
        push(@$ary, ea_new_item(ea_new_catalog($v->[0]), $v->[1]));
}

# Open the exacct file, write the record & close.
my $f = ea_new_file('/tmp/exacct', &O_RDWR | &O_CREAT | &O_TRUNC)
   || die("create /tmp/exacct failed: ", ea_error_str(), "\n");
$f->write($group);
$f = undef;
```

exacct 파일의 내용을 인쇄하는 방법

다음 Perl 스크립트를 사용하여 exacct 파일의 내용을 인쇄할 수 있습니다.

```
#!/usr/bin/perl

     use strict;
     use warnings;
     use Sun::Solaris::Exacct qw(:EXACCT_ALL);

     die("Usage is dumpexacct <exacct file>\n") unless (@ARGV == 1);

     # Open the exact file and display the header information.
     my $ef = ea_new_file($ARGV[0], &O_RDONLY) || die(error_str());
     printf("Creator:  %s\n", $ef->creator());
     printf("Hostname: %s\n\n", $ef->hostname());

     # Dump the file contents
     while (my $obj = $ef->get()) {
             ea_dump_object($obj);
     }

     # Report any errors
     if (ea_error() != EXR_OK && ea_error() != EXR_EOF)  {
             printf("\nERROR: %s\n", ea_error_str());
             exit(1);
     }
     exit(0);
```

Sun::Solaris::Exacct::Object->dump()의 출력 예

이 출력 예는 72 페이지 "새 그룹 레코드를 만들고 이를 파일에 쓰는 방법"에서 만든 파일에 대해 Sun::Solaris::Exacct::Object->dump()를 실행하여 생성된 것입니다.

```
Creator:  root
Hostname: localhost
GROUP
       Catalog = EXT_GROUP|EXC_DEFAULT|EXD_NONE
```

```
        ITEM
          Catalog = EXT_STRING|EXC_DEFAULT|EXD_CREATOR
          Value = me
        ITEM
          Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_PID
          Value = 845523
        ITEM
          Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_UID
          Value = 37845
        ITEM
          Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_GID
          Value = 10
        ITEM
          Catalog = EXT_STRING|EXC_DEFAULT|EXD_PROC_COMMAND
          Value = /bin/rec
ENDGROUP
```

◆ ◆ ◆ 6 장

6 리소스 제어(개요)

4 장, "확장 계정(개요)"에 설명된 시스템에서 작업 부하의 리소스 사용을 확인한 후 리소스 사용에 대한 한계를 설정할 수 있습니다. 한계는 작업 부하로 인해 리소스가 과다 사용되는 것을 방지합니다. **리소스 제어** 기능은 이러한 용도로 사용되는 제약 조건 방식입니다.

이 장에서는 다음 내용을 다룹니다.

리소스 제어를 관리하는 방법에 대한 자세한 내용은 7 장, "리소스 제어 관리(작업)"를 참조하십시오.

리소스 제어 개념

Oracle Solaris 운영 체제에서는 프로세스별 리소스 제한의 개념이 2 장, "프로젝트 및 작업(개요)"에 설명된 작업 및 프로젝트까지 확장되었습니다. 이러한 향상된 기능은 리소스 제어(rctls) 기능을 통해 제공됩니다. 또한 `/etc/system` 조정 기능을 통해 설정된 할당이 리소스 제어 방식을 통해 구성되거나 자동으로 구성됩니다.

리소스 제어는 `zone`, `project`, `task` 또는 `process` 접두어를 통해 식별됩니다. 시스템 전체에서 리소스 제어를 살펴볼 수 있습니다. 실행 중인 시스템에서 리소스 제어 값을 업데이트할 수 있습니다.

이 릴리스에서 사용할 수 있는 표준 리소스 제어 목록은 78 페이지 "사용 가능한 리소스 제어"를 참조하십시오. 사용할 수 있는 영역 전체 리소스 제어에 대한 자세한 내용은 227 페이지 "리소스 유형 등록 정보"를 참조하십시오.

리소스 제한 및 리소스 제어

UNIX 시스템에서 예전부터 리소스 제한 기능(*rlimit*)을 제공해 왔습니다. 관리자는 rlimit 기능을 사용하여 프로세스에 사용할 수 있는 리소스 양에 대해 하나 이상의 수치 제한을 설정할 수 있습니다. 이러한 제한에는 사용된 프로세스별 CPU 시간, 프로세스별 코어 파일 크기 및 프로세스별 최대 힙 크기가 포함됩니다. **힙 크기**는 프로세스 데이터 세그먼트에 대해 할당된 스크래치 메모리의 양입니다.

리소스 제어 기능은 리소스 제한 기능에 대한 호환성 인터페이스를 제공합니다. 리소스 제한을 사용하는 기존 응용 프로그램은 변경되지 않은 상태로 계속 실행됩니다. 리소스 제어 기능을 활용하도록 수정된 응용 프로그램과 동일한 방식으로 이러한 응용 프로그램을 관찰할 수 있습니다.

프로세스간 통신 및 리소스 제어

프로세스는 IPC(프로세스간 통신)의 몇 가지 유형 중 하나를 사용하여 서로 간에 통신할 수 있습니다. IPC를 통해 정보 전송 또는 동기화가 프로세스 사이에서 발생할 수 있습니다. 리소스 제어 기능은 커널의 IPC 기능의 동작을 정의하는 리소스 제어를 제공합니다. 이러한 리소스 제어는 `/etc/system` 튜너블을 대체합니다.

기본 리소스 제어 값을 초기화하는 데 사용되는 오래된 매개변수가 이 Oracle Solaris 시스템의 `/etc/system` 파일에 포함될 수도 있습니다. 그러나 폐기된 매개변수를 사용하는 것은 좋지 않습니다.

프로젝트의 사용에 참여하는 IPC 객체를 살펴보려면 `ipcs` 명령과 `-J` 옵션을 함께 사용합니다. 예제 화면을 보려면 99 페이지 “ipcs를 사용하는 방법”을 참조하십시오. `ipcs` 명령에 대한 자세한 내용은 `ipcs(1)`를 참조하십시오.

Oracle Solaris 시스템 조정에 대한 자세한 내용은 **Oracle Solaris 조정 가능 매개변수 참조 설명서**를 참조하십시오.

리소스 제어 제약 조건 방식

리소스 제어는 시스템 리소스에 대한 제약 조건 방식을 제공합니다. 프로세스, 작업, 프로젝트 및 영역이 지정된 시스템 리소스 양을 사용할 수 없게 됩니다. 이 방식은 리소스 과다 소비를 방지하여 더 효율적으로 시스템을 관리할 수 있게 합니다.

제약 조건 방식을 사용하여 용량 계획 프로세스를 지원할 수 있습니다. 제약 조건은 응용 프로그램에 대한 리소스를 거부하지 않고 응용 프로그램 리소스 요구에 대한 정보를 제공할 수 있습니다.

프로젝트 속성 방식

리소스 제어는 리소스 관리 기능을 위한 단순 속성 방식으로도 사용할 수 있습니다. 예를 들어 FSS(Fare Share Scheduler) 예약 클래스에서 프로젝트에 사용할 수 있는 CPU 수는 `project.cpu-shares` 리소스 제어를 통해 정의됩니다. 이 컨트롤을 통해 프로젝트에 고정된 몫이 지정되므로 컨트롤 초과와 관련된 다양한 동작에 신경을 쓰지 않아도 됩니다. 이 컨텍스트에서 `project.cpu-shares` 컨트롤에 대한 현재 값은 지정된 프로젝트에 대한 속성으로 간주됩니다.

프로젝트 속성의 또 다른 유형은 프로젝트에 연결된 프로세스 컬렉션에서 물리적 메모리의 리소스 사용을 규제하는 데 사용됩니다. 이러한 속성에는 `rcap` 접두어가 붙습니다. 예를 들면, `rcap.max-rss`가 있습니다. 이 유형의 속성은 리소스 제어와 마찬가지로 `project` 데이터베이스에서 구성됩니다. 하지만 커널에서 리소스 제어를 동기적으로 적용할 경우 리소스 상한값은 `rcapd` ((Resource Capping Daemon)에 의해 사용자 레벨에서 비동기적으로 적용됩니다. `rcapd`에 대한 자세한 내용은 10 장, "리소스 상한값 지원 데몬을 사용한 물리적 메모리 제어(개요)"와 `rcapd` (1M)을 참조하십시오.

`project.pool` 속성은 프로젝트에 대해 풀 바인딩을 지정하는 데 사용됩니다. 리소스 풀에 대한 자세한 내용은 12 장, "리소스 풀(개요)"을 참조하십시오.

리소스 제어 및 속성 구성

리소스 제어 기능은 `project` 데이터베이스를 통해 구성됩니다. 2 장, "프로젝트 및 작업(개요)"을 참조하십시오. 리소스 제어 및 기타 속성은 `project` 데이터베이스 항목의 최종 필드에 설정됩니다. 각 리소스 제어와 연결된 값은 괄호 안에 지정되며 쉼표로 구분된 일반 텍스트로 나타납니다. 괄호 안의 값은 "동작 절(action clause)"을 구성합니다. 각 동작 절은 권한 레벨, 임계치 값 및 특정 임계치와 연결된 동작으로 구성됩니다. 각 리소스 제어에는 여러 개의 동작 절이 쉼표로 구분되어 있을 수 있습니다. 다음 항목은 프로젝트 엔티티에 대한 작업별 경량 프로세스 제한 및 프로세스별 최대 CPU 시간 제한을 정의합니다. `process.max-cpu-time`은 1시간 동안 프로세스가 실행된 후 SIGTERM 프로세스를 전송하며, 프로세스가 총 1시간 1분 동안 계속 실행된 경우 SIGKILL을 전송합니다. 표 6–3을 참조하십시오.

```
development:101:Developers:::task.max-lwps=(privileged,10,deny);
  process.max-cpu-time=(basic,3600,signal=TERM),(priv,3660,signal=KILL)
    typed as one line
```

주 – 영역을 사용할 수 있는 시스템에서는 영역 전체 리소스 제어가 다소 다른 형식을 사용하여 영역 구성에 지정됩니다. 자세한 내용은 222 페이지 "영역 구성 데이터"를 참조하십시오.

`rctladm` 명령을 사용하면 **전역 범위**에서 리소스 제어 기능에 대해 런타임으로 질의하고 수정할 수 있습니다. `prctl` 명령을 사용하면 **로컬 범위**에서 리소스 제어 기능에 대해 런타임으로 질문 및 수정을 수행할 수 있습니다.

자세한 내용은 84 페이지 “리소스 제어 값에 대한 전역 및 로컬 동작”, rctladm(1M) 및 prctl(1)을 참조하십시오.

주 – 영역이 설치된 시스템의 비전역 영역에서는 rctladm을 사용하여 설정을 수정할 수 없습니다. 비전역 영역에서는 rctladm을 사용하여 각 리소스 제어의 전역 로깅 상태를 볼 수 있습니다.

사용 가능한 리소스 제어

이 릴리스에서 사용할 수 있는 표준 리소스 제어의 목록이 다음 표에 표시됩니다.

이 표는 각 컨트롤에 의해 제한된 리소스를 설명합니다. 이 표는 해당 리소스에 대한 project 데이터베이스에서 사용되는 기본 단위를 식별합니다. 기본 단위에는 두 가지 유형이 있습니다.

- 수량은 제한된 양을 나타냅니다.
- 인덱스는 최대 유효 식별자를 나타냅니다.

따라서 project.cpu-shares는 프로젝트에 부여된 할당 수를 지정합니다. process.max-file-descriptor는 open(2) 시스템 호출을 통해 프로세스에 지정할 수 있는 최고 파일 번호를 지정합니다.

표 6–1 표준 프로젝트, 작업 및 프로세스 리소스 제어

컨트롤 이름	설명	기본 단위
project.cpu-cap	프로젝트별로 소모될 수 있는 CPU 리소스 용량의 절대 한계입니다. 값 100은 project.cpu-cap 설정을 기준으로 CPU 하나의 100%를 의미합니다. 100%는 CPU 상한값을 사용할 때 시스템에서 하나의 전체 CPU를 말하는 것이므로 값 125는 125%입니다.	수량(CPU 수)
project.cpu-shares	FSS(Fair Share Scheduler)에서 사용하기 위해 이 프로젝트에 부여된 CPU 할당 수입니다(FSS(7) 참조).	수량(할당)
project.max-crypto-memory	하드웨어 암호화 가속화를 위해 libpkcs11에 사용할 수 있는 총 커널 메모리 양입니다. 이 리소스 제어에 대해 커널 버퍼 및 세션 관련 구조 할당이 부과됩니다.	크기(바이트)

표 6–1 표준 프로젝트, 작업 및 프로세스 리소스 제어 *(계속)*

컨트롤 이름	설명	기본 단위
project.max-locked-memory	허용되는 물리적 잠긴 메모리의 총량입니다. priv_proc_lock_memory가 사용자에게 지정된 경우 해당 사용자가 모든 메모리를 잠그지 않도록 이 리소스 제어를 설정해 보십시오. 이 리소스 제어는 제거된 project.max-device-locked-memory를 대체합니다.	크기(바이트)
project.max-msg-ids	이 프로젝트에 허용되는 최대 메시지 대기열 ID 수입니다.	수량(메시지 대기열 ID)
project.max-port-ids	허용되는 최대 이벤트 포트 수입니다.	수량(이벤트 포트 수)
project.max-processes	이 프로젝트에 동시에 사용할 수 있는 최대 프로세스 테이블 슬롯 수입니다. 일반 프로세스와 좀비 프로세스는 모두 프로세스 테이블 슬롯을 사용하기 때문에 max-processes 컨트롤은 좀비 프로세스가 프로세스 테이블을 소모하지 못하도록 방지합니다. 좀비 프로세스에는 정의상 LWP가 없으므로 max-lwps 컨트롤은 이러한 방지를 수행할 수 없습니다.	수량(프로세스 테이블 슬롯)
project.max-sem-ids	이 프로젝트에 대해 허용된 최대 세마포 ID 수입니다.	수량(세마포 ID)
project.max-shm-ids	이 프로젝트에 대해 허용된 공유 메모리 ID 수입니다.	수량(공유 메모리 ID)
project.max-shm-memory	이 프로젝트에 대해 허용되는 전체 시스템 V 공유 메모리 양입니다.	크기(바이트)
project.max-lwps	이 프로젝트에 동시에 사용할 수 있는 최대 LWP 수입니다.	수량(LWP)
project.max-tasks	이 프로젝트에서 허용하는 최대 작업 수입니다.	수량(작업)
project.max-contracts	이 프로젝트에 허용되는 최대 계약 수입니다.	수량(계약)
task.max-cpu-time	이 작업의 프로세스에 사용할 수 있는 최대 CPU 시간입니다.	시간(초)

표 6–1 표준 프로젝트, 작업 및 프로세스 리소스 제어 *(계속)*

컨트롤 이름	설명	기본 단위
task.max-lwps	이 작업의 프로세스에 동시에 사용할 수 있는 최대 LWP 수입니다.	수량(LWP)
task.max-processes	이 작업의 프로세스에 동시에 사용할 수 있는 최대 프로세스 테이블 슬롯 수입니다.	수량(프로세스 테이블 슬롯)
process.max-cpu-time	이 프로세스에 사용할 수 있는 최대 CPU 시간입니다.	시간(초)
process.max-file-descriptor	이 프로세스에 사용할 수 있는 최대 파일 설명자 인덱스입니다.	인덱스(최대 파일 설명자)
process.max-file-size	이 프로세스에서 작성할 수 있는 최대 파일 오프셋입니다.	크기(바이트)
process.max-core-size	이 프로세스에서 생성하는 코어 파일의 최대 크기입니다.	크기(바이트)
process.max-data-size	이 프로세스에 사용할 수 있는 최대 힙 메모리입니다.	크기(바이트)
process.max-stack-size	이 프로세스에 사용할 수 있는 최대 스택 메모리 세그먼트입니다.	크기(바이트)
process.max-address-space	이 프로세스에 사용할 수 있는 최대 주소 공간 크기로, 세그먼트 크기로 합산됩니다.	크기(바이트)
process.max-port-events	이벤트 포트당 최대 허용 이벤트 수입니다.	수량(이벤트)
process.max-sem-nsems	세마포 집합당 허용되는 최대 세마포 수입니다.	수량(집합당 세마포)
process.max-sem-ops	semop 호출당 허용되는 최대 세마포 작업 수(semget() 시간에 리소스 제어로부터 복사된 값)입니다.	수량(작업 수)
process.max-msg-qbytes	메시지 대기열의 최대 메시지 바이트 수(msgget() 시간에 리소스 제어에서 복사된 값)입니다.	크기(바이트)
process.max-msg-messages	메시지 대기열의 최대 메시지 수(msgget() 시간에 리소스 제어에서 복사된 값)입니다.	수량(메시지 수)

리소스 제어가 설정되지 않았거나 변경되지 않은 시스템에서 리소스 제어의 기본값을 표시할 수 있습니다. 이러한 시스템의 /etc/system 또는 project 데이터베이스에는 기본값이 아닌 항목이 들어 있지 않습니다. 값을 표시하려면 prctl 명령을 사용합니다.

영역 전체 리소스 제어

영역 전체 리소스 제어는 영역 내에서 모든 프로세스 항목의 총 리소스 사용을 제한합니다. 영역 전체 리소스 제어는 215 페이지 "영역 전체 리소스 제어 설정" 및 243 페이지 "영역 구성 방법"에 설명된 대로 전역 등록 정보 이름을 사용하여 설정할 수도 있습니다.

표 6–2 영역 리소스 제어

컨트롤 이름	설명	기본 단위
`zone.cpu-cap`	비전역 영역별로 소모될 수 있는 CPU 리소스 용량의 절대 한계입니다. 값 `100`은 `project.cpu-cap` 설정을 기준으로 CPU 하나의 100%를 의미합니다. 100%는 CPU 상한값을 사용할 때 시스템에서 하나의 전체 CPU를 말하는 것이므로 값 125는 125%입니다.	수량(CPU 수)
`zone.cpu-shares`	이 영역에 대한 FSS(Fair Share Scheduler) CPU 할당 수	수량(할당)
`zone.max-lofi`	영역에서 만들 수 있는 최대 `lofi` 장치 수입니다. 이 값은 각 영역의 부 노드 이름 공간 사용을 제한합니다.	수량(`lofi` 장치 수)
`zone.max-locked-memory`	영역에 사용할 수 있는 전체 물리적 잠긴 메모리의 양입니다. `priv_proc_lock_memory`가 영역에 지정된 경우 해당 영역이 모든 메모리를 잠그지 않도록 이 리소스 제어를 설정하는 것도 고려해 보십시오.	크기(바이트)
`zone.max-lwps`	이 영역에 동시에 사용할 수 있는 최대 LWP 수입니다	수량(LWP)
`zone.max-msg-ids`	이 영역에 허용되는 최대 메시지 대기열 ID 수입니다	수량(메시지 대기열 ID)

표 6–2 영역 리소스 제어 (계속)

컨트롤 이름	설명	기본 단위
zone.max-processes	이 영역에 동시에 사용할 수 있는 최대 프로세스 테이블 슬롯 수입니다. 일반 프로세스와 좀비 프로세스는 모두 프로세스 테이블 슬롯을 사용하기 때문에 max-processes 컨트롤은 좀비 프로세스가 프로세스 테이블을 소모하지 못하도록 방지합니다. 좀비 프로세스에는 정의상 LWP가 없으므로 max-lwps 컨트롤은 이러한 방지를 수행할 수 없습니다.	수량(프로세스 테이블 슬롯)
zone.max-sem-ids	이 영역에 허용되는 최대 세마포 ID 수입니다	수량(세마포 ID)
zone.max-shm-ids	이 영역에 허용되는 공유 메모리 ID 수입니다	수량(공유 메모리 ID)
zone.max-shm-memory	이 영역에 허용되는 시스템 V 공유 메모리의 총량입니다.	크기(바이트)
zone.max-swap	이 영역에 대한 사용자 프로세스 주소 공간 매핑 및 tmpfs 마운트에 사용할 수 있는 총 스왑 크기입니다.	크기(바이트)

영역 전체 리소스 제어 구성에 대한 자세한 내용은 227 페이지 “리소스 유형 등록 정보” 및 243 페이지 “영역 구성 방법”을 참조하십시오.

영역 전체 리소스 제어를 전역 영역에 적용할 수 있습니다. 자세한 내용은 366 페이지 “영역이 설치된 Oracle Solaris 시스템에서 Fair Share Scheduler 사용”을 참조하십시오.

단위 지원

모든 리소스 제어마다 리소스 제어 형식을 식별하는 전역 플래그가 정의되어 있습니다. 이러한 플래그는 시스템에서 prctl 명령과 같은 응용 프로그램에 기본 형식 정보를 전달하는 데 사용됩니다. 응용 프로그램에서는 이 정보를 사용하여 다음을 결정합니다.

- 각 리소스 제어에 적합한 단위 문자열
- 배율 값을 해석할 때 사용할 올바른 배율

다음 전역 플래그를 사용할 수 있습니다.

전역 플래그	리소스 제어 형식 문자열	수정자	배율
RCTL_GLOBAL_BYTES	바이트	B	1
		KB	2^{10}
		MB	2^{20}
		GB	2^{30}
		TB	2^{40}
		PB	2^{50}
		EB	2^{60}
RCTL_GLOBAL_SECONDS	초	s	1
		Ks	10^{3}
		Ms	10^{6}
		Gs	10^{9}
		Ts	10^{12}
		Ps	10^{15}
		Es	10^{18}
RCTL_GLOBAL_COUNT	개수	none	1
		K	10^{3}
		M	10^{6}
		G	10^{9}
		T	10^{12}
		P	10^{15}
		E	10^{18}

배율 값은 리소스 제어와 함께 사용할 수 있습니다. 다음 예에서는 배율 임계치 값을 보여 줍니다.

```
task.max-lwps=(priv,1K,deny)
```

주 – 단위 수정자는 `prctl`, `projadd` 및 `projmod` 명령에 의해 허용됩니다. `project` 데이터베이스 자체에서는 단위 수정자를 사용할 수 없습니다.

리소스 제어 값과 권한 레벨

리소스 제어에 대한 임계치 값은 로컬 동작을 트리거하거나 로깅과 같은 전역 동작이 발생할 수 있는 적용 지점을 구성합니다.

리소스 제어에 대한 각 임계치 값은 권한 레벨과 연결되어야 합니다. 권한 레벨은 다음 세 가지 유형 중 하나여야 합니다.

- 기본 유형 - 호출 프로세스의 소유자가 수정할 수 있습니다.
- 권한 부여 - 권한 부여된(루트) 호출자만 수정할 수 있습니다.
- 시스템 - 운영 체제 인스턴스 기간 동안 고정됩니다.

리소스 제어에는 시스템이나 리소스 공급자가 정의하는 하나의 시스템 값이 지정됩니다. 시스템 값은 운영 체제의 현재 구현에서 제공할 수 있는 리소스의 양을 나타냅니다.

권한 부여된 값의 수를 정의할 수 있으며 하나의 기본 값만 허용됩니다. 권한 값을 지정하지 않고 수행되는 작업에는 기본적으로 기본 권한이 할당됩니다.

리소스 제어 값에 대한 권한 레벨은 RCTL_BASIC, RCTL_PRIVILEGED 또는 RCTL_SYSTEM으로 리소스 제어 블록의 권한 필드에 정의되어 있습니다. 자세한 내용은 `setrctl(2)`를 참조하십시오. `prctl` 명령을 사용하여 기본 및 권한 부여된 레벨과 연결된 값을 수정할 수 있습니다.

리소스 제어 값에 대한 전역 및 로컬 동작

리소스 제어 값에 대한 동작 범주에는 전역과 로컬의 두 가지가 있습니다.

리소스 제어 값에 대한 전역 동작

전역 동작은 시스템의 모든 리소스 제어에 대한 리소스 제어 값에 적용됩니다. `rctladm(1M)` 매뉴얼 페이지에 설명된 `rctladm` 명령을 사용하여 다음 동작을 수행할 수 있습니다.

- 활성 시스템 리소스 제어의 전역 상태 표시
- 전역 로깅 동작 설정

리소스 제어에 대한 전역 로깅 동작을 사용 또는 사용 안함으로 설정할 수 있습니다. 심각도 레벨을 지정하여(`syslog=`*level*) `syslog` 동작을 특정 레벨로 설정할 수 있습니다. *level*에 사용할 수 있는 설정은 다음과 같습니다.

- `debug`
- `info`
- `notice`
- `warning`
- `err`

- crit
- alert
- emerg

기본적으로 리소스 제어 위반에 대한 전역 로깅은 없습니다. n/a 레벨은 전역 동작을 구성할 수 없는 리소스 제어를 나타냅니다.

리소스 제어 값에 대한 로컬 동작

제어 값을 초과하려는 프로세스에 대해 로컬 동작이 수행됩니다. 리소스 제어에 설정된 각 임계값에 대해 하나 이상의 동작을 연결할 수 있습니다. 로컬 동작에는 none, deny 및 signal=의 세 가지 유형이 있습니다. 이러한 세 가지 동작은 다음과 같이 사용됩니다.

none 임계치보다 많은 양의 리소스 요청에 대해 동작이 수행되지 않습니다. 이 동작은 응용 프로그램 진행에 영향을 주지 않고 리소스 사용을 모니터링하는 데 유용합니다. 또한 임계치를 초과하는 프로세스에 영향을 주지 않지만 리소스 제어가 초과될 때 표시되는 전역 메시지를 사용으로 설정할 수도 있습니다.

deny 임계치보다 많은 양의 리소스 요청을 거부할 수 있습니다. 예를 들어 task.max-lwps 리소스 제어와 deny 동작을 사용하면 새 프로세스가 제어 값을 초과할 경우 fork 시스템 호출이 실패합니다. fork(2) 매뉴얼 페이지를 참조하십시오.

signal= 리소스 제어를 초과할 때 전역 단일 메시지 동작을 사용으로 설정할 수 있습니다. 임계치 값이 초과될 때 신호가 해당 프로세스로 전송됩니다. 해당 프로세스가 추가 리소스를 사용할 경우 추가 신호가 전송되지 않습니다. 사용할 수 있는 신호는 표 6–3에 나열되어 있습니다.

일부 동작은 리소스 제어에 적용되지 않을 수 있습니다. 예를 들어 프로세스는 자신이 구성원으로 있는 프로젝트에 지정된 CPU 할당 수를 초과할 수 없습니다. 따라서 project.cpu-shares 리소스 제어에는 거부(deny) 동작이 허용되지 않습니다.

구현 제한으로 인해 각 컨트롤의 전역 등록 정보는 임계치 값에 설정할 수 있는 사용 가능한 동작의 범위를 제한할 수 있습니다. rctladm(1M) 매뉴얼 페이지를 참조하십시오. 사용 가능한 신호 동작 목록은 다음 표에 나와 있습니다. 신호에 대한 자세한 내용은 signal(3HEAD) 매뉴얼 페이지를 참조하십시오.

표 6–3 리소스 제어 값에 사용할 수 있는 신호

신호	설명	주
SIGABRT	프로세스를 종료합니다.	
SIGHUP	행업 신호를 전송합니다. 사용 가능한 회선에서 반송파가 중단될 때 발생합니다. 터미널을 제어하는 프로세스 그룹으로 전송되는 신호입니다.	

표 6–3 리소스 제어 값에 사용할 수 있는 신호 (계속)

신호	설명	주
SIGTERM	프로세스를 종료합니다. 소프트웨어에서 전송한 종료 신호입니다.	
SIGKILL	프로세스를 종료하고 프로그램을 강제 종료합니다.	
SIGSTOP	프로세스를 중지합니다. 작업 제어 신호입니다.	
SIGXRES	리소스 제어 제한을 초과했습니다. 리소스 제어 기능을 통해 생성되었습니다.	
SIGXFSZ	프로세스를 종료합니다. 파일 크기 제한을 초과했습니다.	RCTL_GLOBAL_FILE_SIZE 등록 정보를 가진 리소스 제어(process.max-file-size)에만 사용할 수 있습니다. 자세한 내용은 rctlblk_set_value(3C)를 참조하십시오.
SIGXCPU	프로세스를 종료합니다. CPU 시간 제한을 초과했습니다.	RCTL_GLOBAL_CPUTIME 등록 정보를 가진 리소스 제어(process.max-cpu-time)에만 사용할 수 있습니다. 자세한 내용은 rctlblk_set_value(3C)를 참조하십시오.

리소스 제어 플래그 및 등록 정보

시스템의 각 리소스 제어에는 연결된 등록 정보 집합이 있습니다. 이 등록 정보 집합은 플래그 집합으로 정의되며, 플래그 집합은 해당 리소스의 제어된 모든 인스턴스와 연결되어 있습니다. 전역 플래그를 수정할 수 없지만, rctladm 또는 getrctl 시스템 호출을 사용하여 플래그를 검색할 수 있습니다.

로컬 플래그는 특정 프로세스나 프로세스 컬렉션에 대해 해당 리소스 제어의 특정 임계치 값에 대한 기본 동작 및 구성을 정의합니다. 한 임계치 값에 대한 로컬 플래그는 동일한 리소스 제어에 대한 다른 정의된 임계치 값의 동작에 영향을 주지 않습니다. 하지만 전역 플래그는 특정 컨트롤과 연결된 모든 값에 대한 동작에 영향을 줍니다. 로컬 플래그는 prctl 명령이나 setrctl 시스템 호출을 통해 해당 전역 플래그가 제공하는 제약 조건 내에서 수정할 수 있습니다. setrctl(2)를 참조하십시오.

전체 로컬 플래그/전역 플래그 목록과 그에 대한 정의는 rctlblk_set_value(3C)를 참조하십시오.

특정 리소스 제어에 대한 임계치 값에 도달할 때의 시스템 동작을 알아보려면 rctladm을 사용하여 해당 리소스 제어에 대한 전역 플래그를 표시합니다. 예를 들어, process.max-cpu-time의 값을 표시하려면 다음을 입력합니다.

```
$ rctladm process.max-cpu-time
    process.max-cpu-time  syslog=off  [ lowerable no-deny cpu-time inf seconds ]
```

전역 플래그는 다음을 나타냅니다.

lowerable 이 컨트롤에 대한 권한 부여된 값을 낮추는 데 수퍼 유저 권한은 필요하지 않습니다.

no-deny 임계치 값을 초과하더라도 리소스에 대한 액세스가 거부되지 않습니다.

cpu-time 이 리소스의 임계치 값에 도달할 때 SIGXCPU가 전송될 수 있습니다.

seconds 리소스 제어에 대한 시간 값입니다.

no-basic 권한 유형이 basic인 리소스 제어 값은 설정할 수 없습니다. 권한 부여된 리소스 제어 값만 허용됩니다.

no-signal 리소스 제어 값에는 로컬 신호 작업을 설정할 수 없습니다.

no-syslog 이 리소스 제어에 대해 전역 syslog 메시지 작업은 설정할 수 없습니다.

deny 임계치 값을 초과하는 경우 리소스에 대한 요청을 항상 거부합니다.

count 리소스 제어에 대한 개수(정수) 값입니다.

bytes 리소스 제어의 크기 단위입니다.

prctl 명령을 사용하여 리소스 제어에 대한 로컬 값과 작업을 표시합니다.

```
$ prctl -n process.max-cpu-time $$
    process 353939: -ksh
    NAME    PRIVILEGE    VALUE    FLAG   ACTION              RECIPIENT
 process.max-cpu-time
         privileged   18.4Es    inf   signal=XCPU                -
         system       18.4Es    inf   none
```

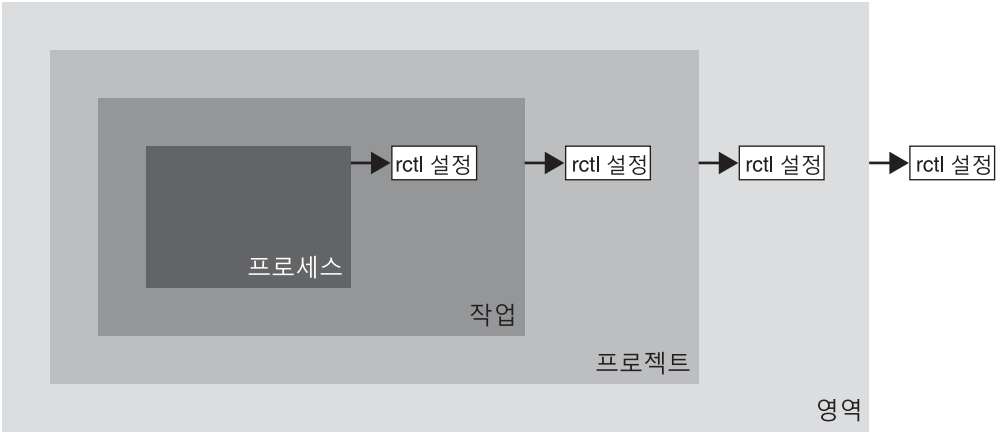
두 임계치 값에 대해 max (RCTL_LOCAL_MAXIMAL) 플래그가 설정되며, 이 리소스 제어에 대해 inf (RCTL_GLOBAL_INFINITE) 플래그가 정의됩니다. inf 값의 수량에는 제한이 없습니다. 값이 적용되지 않습니다. 따라서 구성된 대로 두 임계치 수량은 결코 초과되지 않는 무한 값을 나타냅니다.

리소스 제어 적용

하나의 리소스에 두 개 이상의 리소스 제어가 있을 수 있습니다. 프로세스 모델의 각 포함 레벨에 리소스 제어가 있을 수 있습니다. 리소스 제어가 서로 다른 컨테이너 레벨에서 동일 리소스에 대해 활성 상태이면 최소 컨테이너 컨트롤이 먼저 적용됩니다. 따라서 두

컨트롤(task.max-cpu-time 및 process.max-cpu-time)이 동시에 적용될 경우 process.max-cpu-time보다 task.max-cpu-time에 대해 작업이 먼저 수행됩니다.

그림 6–1 프로세스 컬렉션, 컨테이너 관계 및 관련 리소스 제어 집합



리소스 제어 이벤트에 대한 전역 모니터링

대체로 프로세스의 리소스 사용은 알 수 없습니다. 자세한 정보를 보려면 rctladm 명령과 함께 사용할 수 있는 전역 리소스 제어 동작을 사용해 보십시오. rctladm을 사용하여 리소스 제어에 대해 syslog 작업을 설정합니다. 그런 다음 해당 리소스 제어에서 관리하는 엔티티가 임계치 값을 만나면 구성된 로깅 레벨에서 시스템 메시지가 기록됩니다. 자세한 내용은 7 장, “리소스 제어 관리(작업)” 및 rctladm(1M) 매뉴얼 페이지를 참조하십시오.

리소스 제어 적용

로그인할 때 또는 newtask, su나 기타 프로젝트 인식 실행 프로그램인 at, batch 또는 cron이 호출될 때 표 6–1에 나열된 각 리소스 제어가 프로젝트에 지정될 수 있습니다. 시작되는 각 명령은 호출하는 사용자의 기본 프로젝트와 함께 별도 작업으로 실행됩니다. 자세한 내용은 login(1), newtask(1), at(1), cron(1M) 및 su(1M) 매뉴얼 페이지를 참조하십시오.

/etc/project 파일이든 네트워크 이름 서비스의 데이터베이스 표현이든 상관없이 project 데이터베이스의 항목에 대한 업데이트는 현재 활성 프로젝트에 적용되지 않습니다. 이러한 업데이트는 로그인 또는 newtask를 통해 새 작업이 프로젝트를 참가시킬 때 적용됩니다.

실행 중인 시스템에서 리소스 제어 값 임시 업데이트

project 데이터베이스에서 변경된 값은 프로젝트에서 시작되는 새 작업에만 유효합니다. 하지만 rctladm 및 prctl 명령을 사용하여 실행 중인 시스템에서 리소스 제어를 업데이트할 수 있습니다.

로깅 상태 업데이트

rctladm 명령은 시스템 전체의 각 리소스 제어에 대한 전역 로깅 상태에 영향을 줍니다. 이 명령을 사용하여 전역 상태를 확인하고 컨트롤이 초과될 때 syslog 로깅 레벨을 설정할 수 있습니다.

리소스 제어 업데이트

prctl 명령을 사용하여 프로세스별로, 작업별로 또는 프로젝트별로 리소스 제어 값과 작업을 확인하고 임시로 변경할 수 있습니다. 프로젝트, 작업 또는 프로세스 ID가 입력으로 제공되면 컨트롤이 정의된 레벨의 리소스 제어에 대해 명령이 실행됩니다.

값과 동작에 대한 수정 사항은 즉시 적용됩니다. 하지만 이러한 수정 사항은 현재 프로세스, 작업 또는 프로젝트에만 적용됩니다. project 데이터베이스에는 변경 사항이 기록되지 않습니다. 시스템을 다시 시작하면 수정 사항이 손실됩니다. 리소스 제어를 영구적으로 변경하려면 project 데이터베이스에서 변경해야 합니다.

project 데이터베이스에서 수정할 수 있는 리소스 제어 설정은 모두 prctl 명령을 사용하여 수정할 수 있습니다. 기본 및 권한 부여된 값 둘 다 추가하거나 삭제할 수 있습니다. 이러한 유형의 작업도 수정할 수 있습니다. 기본적으로 설정된 모든 작업은 기본 유형으로 가정되지만, 루트 권한을 가진 사용자와 프로세스도 권한 부여된 리소스 제어를 수정할 수 있습니다. 시스템 리소스 제어는 변경할 수 없습니다.

리소스 제어와 함께 사용되는 명령

다음 표에는 리소스 제어와 함께 사용되는 명령이 나와 있습니다.

명령 참조	설명
ipcs(1)	프로젝트 사용에 참여하는 IPC 객체를 관찰할 수 있습니다.
prctl(1)	로컬 범위를 사용하여 리소스 제어 기능에 대해 런타임으로 질의하고 수정할 수 있습니다.

명령 참조	설명
rctladm(1M)	전역 범위를 사용하여 리소스 제어 기능에 대해 런타임으로 질의하고 수정할 수 있습니다.

resource_controls(5) 매뉴얼 페이지에서는 단위 및 배율을 비롯하여 프로젝트 데이터베이스를 통해 사용할 수 있는 리소스 제어에 대해 설명합니다.

◆ ◆ ◆ 7 장

7 리소스 제어 관리(작업)

이 장에서는 리소스 제어 기능을 관리하는 방법에 대해 설명합니다.

리소스 제어 기능에 대한 개요는 6 장, “리소스 제어(개요)”를 참조하십시오.

리소스 제어 관리(작업 맵)

작업	설명	수행 방법
리소스 제어를 설정합니다.	`/etc/project` 파일에서 프로젝트에 대한 리소스 제어를 설정합니다.	92 페이지 “리소스 제어 설정”
로컬 범위에서 활성 프로세스, 작업 또는 프로젝트에 대한 리소스 제어를 가져오거나 수정합니다.	시스템의 활성 프로세스, 작업 또는 프로젝트와 연관된 리소스 제어에 대해 런타임 질의 및 수정을 수행합니다.	94 페이지 “`prctl` 명령 사용”
실행 중인 시스템에서 리소스 제어의 전역 상태를 확인 또는 업데이트합니다.	시스템 전체를 기반으로 각 리소스 제어의 전역 로깅 상태를 확인합니다. 또한 제어가 초과될 때의 `syslog` 로깅 레벨을 설정합니다.	98 페이지 “`rctladm` 사용”
활성 IPC(프로세스간 통신) 기능의 상태를 보고합니다.	활성 IPC(프로세스간 통신) 기능에 대한 정보를 표시합니다. 프로젝트 사용에 영향을 주는 IPC 객체가 무엇인지 관찰합니다.	99 페이지 “`ipcs` 사용”

작업	설명	수행 방법
웹 서버에 충분한 CPU 용량이 할당되어 있는지 여부를 확인합니다.	리소스 제어에서 전역 작업을 설정합니다. 이 작업을 사용하여 리소스 제어 값이 너무 낮게 설정된 엔티티가 있는 경우 알림을 받을 수 있습니다.	100 페이지 “웹 서버에 충분한 CPU 용량이 할당되어 있는지 여부를 확인하는 방법”

리소스 제어 설정

▼ 프로젝트의 각 작업에 대한 LWP 최대값을 설정하는 방법

이 절차는 x-files라는 프로젝트를 /etc/project 파일에 추가하고 해당 프로젝트에서 만들어진 작업에 대한 LWP 최대값을 설정합니다.

1 관리자로 전환합니다.

2 projadd 명령을 -K 옵션과 함께 사용하여 x-files라는 프로젝트를 만듭니다. 프로젝트에서 만들어진 각 작업의 LWP 최대값을 3으로 설정합니다.

```
# projadd -K 'task.max-lwps=(privileged,3,deny)' x-files
```

3 다음 중 한 가지 방법을 사용하여 /etc/project 파일의 항목을 확인합니다.

- 다음을 입력합니다.

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
.
.
.
x-files
        projid : 100
        comment: ""
        users  : (none)
        groups : (none)
        attribs: task.max-lwps=(privileged,3,deny)
```

- 다음을 입력합니다.

```
# cat /etc/project
system:0:System:::
.
```

```
.
.
x-files:100::::task.max-lwps=(privileged,3,deny)
```

예 7–1 예제 세션

이 절차의 단계를 구현한 후 루트 사용자가 x-files 프로젝트에서 프로젝트를 newtask와 연결하여 새 작업을 만들면 이 작업을 실행하는 동안에는 사용자가 세 개를 초과하는 LWP는 만들 수 없게 됩니다. 이것이 아래의 예제 세션에 주석으로 표시되어 있습니다.

```
# newtask -p x-files csh

# prctl -n task.max-lwps $$
process: 111107: csh
NAME    PRIVILEGE    VALUE    FLAG   ACTION            RECIPIENT
task.max-lwps
        usage            3
        privileged       3       -   deny                    -
        system       2.15G     max   deny                    -
# id -p
uid=0(root) gid=1(other) projid=100(x-files)

# ps -o project,taskid -p $$
 PROJECT TASKID
 x-files    73

# csh        /* creates second LWP */

# csh        /* creates third LWP */

# csh        /* cannot create more LWPs */
Vfork failed
#
```

▼ 프로젝트에서 여러 제어를 설정하는 방법

/etc/project 파일에 각 프로젝트에 대한 여러 리소스 제어 및 각 제어에 대한 여러 임계값에 대한 설정이 포함될 수 있습니다. 임계값은 작업 절에서 정의되며, 여러 값의 경우 쉼표로 구분됩니다.

1 관리자로 전환합니다.

2 projmod 명령을 -s 및 -K 옵션과 함께 사용하여 x-files 프로젝트에 대한 리소스 제어를 설정합니다.

```
# projmod -s -K 'task.max-lwps=(basic,10,none),(privileged,500,deny);
process.max-file-descriptor=(basic,128,deny)' x-files      one line in file
```

다음 제어가 설정됩니다.

- 작업별 최대 LWP에 대한 작업이 없는 basic 제어.

- 작업별 최대 LWP에 대한 권한이 있는 deny 제어. 이러한 제어로 인해 이전 예 92 페이지 "프로젝트의 각 작업에 대한 LWP 최대값을 설정하는 방법"에 표시된 대로 최대값을 초과하는 LWP 생성은 실패하게 됩니다.
- basic 레벨에서의 프로세스별 최대 파일 설명자에 대한 제한. 이것은 최대값을 초과하는 open 호출을 실패로 처리합니다.

3 다음 중 한 가지 방법을 사용하여 파일의 항목을 확인합니다.

- 다음을 입력합니다.

```
# projects -l
.
.
.
x-files
        projid : 100
        comment: ""
        users  : (none)
        groups : (none)
        attribs: process.max-file-descriptor=(basic,128,deny)
                 task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in file
```

- 다음을 입력합니다.

```
# cat etc/project
.
.
.
x-files:100::::process.max-file-descriptor=(basic,128,deny);
task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in file
```

prctl 명령 사용

prctl 명령을 사용하여 시스템의 활성 프로세스, 작업 또는 프로젝트와 연관된 리소스 제어에 대해 런타임 질의 및 수정을 수행합니다. 자세한 내용은 prctl(1) 매뉴얼 페이지를 참조하십시오.

▼ 기본 리소스 제어 값을 표시하기 위해 prctl 명령을 사용하는 방법

이 절차는 리소스 제어가 설정되거나 변경된 적이 없는 시스템에서 사용해야 합니다. /etc/system 파일 또는 project 데이터베이스에 기본값이 아닌 항목만 존재할 수 있습니다.

- **실행 중인 현재 셸 등의 프로세스에서 prctl 명령을 사용합니다.**

```
# prctl $$
process: 3320: bash
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
```

```
process.max-port-events
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-msg-messages
        privileged      8.19K       -   deny                                 -
        system          4.29G     max   deny                                 -
process.max-msg-qbytes
        privileged      64.0KB      -   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-sem-ops
        privileged        512       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-sem-nsems
        privileged        512       -   deny                                 -
        system          32.8K     max   deny                                 -
process.max-address-space
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-descriptor
        basic             256       -   deny                              3320
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-core-size
        privileged      8.00EB    max   deny                                 -
        system          8.00EB    max   deny                                 -
process.max-stack-size
        basic           10.0MB      -   deny                              3320
        privileged      32.0TB      -   deny                                 -
        system          32.0TB    max   deny                                 -
process.max-data-size
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-size
        privileged      8.00EB    max   deny,signal=XFSZ                     -
        system          8.00EB    max   deny                                 -
process.max-cpu-time
        privileged      18.4Es    inf   signal=XCPU                          -
        system          18.4Es    inf   none                                 -
task.max-cpu-time
        usage               0s
        system          18.4Es    inf   none                                 -
task.max-processes
        usage               2
        system          2.15G     max   deny                                 -
task.max-lwps
        usage               3
        system          2.15G     max   deny                                 -
project.max-contracts
        privileged      10.0K       -   deny                                 -
        system          2.15G     max   deny                                 -
project.max-locked-memory
        usage               0B
        system          16.0EB    max   deny                                 -
project.max-port-ids
        privileged      8.19K       -   deny                                 -
        system          65.5K     max   deny                                 -
project.max-shm-memory
        privileged       510MB      -   deny                                 -
        system          16.0EB    max   deny                                 -
```

```
project.max-shm-ids
        privileged        128       -   deny                                 -
        system           16.8M     max   deny                                 -
project.max-msg-ids
        privileged        128       -   deny                                 -
        system           16.8M     max   deny                                 -
project.max-sem-ids
        privileged        128       -   deny                                 -
        system           16.8M     max   deny                                 -
project.max-crypto-memory
        usage               0B
        privileged       510MB      -   deny                                 -
        system           16.0EB    max   deny                                 -
project.max-tasks
        usage               2
        system           2.15G     max   deny                                 -
project.max-processes
        usage               4
        system           2.15G     max   deny                                 -
project.max-lwps
        usage              11
        system           2.15G     max   deny                                 -
project.cpu-cap
        usage               0
        system           4.29G     inf   deny                                 -
project.cpu-shares
        usage               1
        privileged          1       -   none                                 -
        system           65.5K     max   none                                 -
zone.max-lofi
        usage               0
        system           18.4E     max   deny                                 -
zone.max-swap
        usage            180MB
        system           16.0EB    max   deny                                 -
zone.max-locked-memory
        usage               0B
        system           16.0EB    max   deny                                 -
zone.max-shm-memory
        system           16.0EB    max   deny                                 -
zone.max-shm-ids
        system           16.8M     max   deny                                 -
zone.max-sem-ids
        system           16.8M     max   deny                                 -
zone.max-msg-ids
        system           16.8M     max   deny                                 -
zone.max-processes
        usage              73
        system           2.15G     max   deny                                 -
zone.max-lwps
        usage             384
        system           2.15G     max   deny                                 -
zone.cpu-cap
        usage               0
        system           4.29G     inf   deny                                 -
zone.cpu-shares
        usage               1
        privileged          1       -   none                                 -
        system           65.5K     max   none
```

▼ 지정된 리소스 제어에 대한 정보를 표시하기 위해 prctl 명령을 사용하는 방법

- **실행 중인 현재 셸에 대한 최대 파일 설명자를 표시합니다.**

```
# prctl -n process.max-file-descriptor $$
process: 110453: -sh
NAME    PRIVILEGE       VALUE    FLAG   ACTION       RECIPIENT
process.max-file-descriptor
        basic             256       -   deny            11731
        privileged      65.5K       -   deny                -
        system          2.15G     max   deny
```

▼ 값을 임시로 변경하기 위해 prctl을 사용하는 방법

이 절차 에에서는 `prctl` 명령을 사용하여 권한이 있는 새 값을 임시로 추가하여 `x-files` 프로젝트에서 프로젝트별로 세 개를 초과하는 LWP를 사용하지 못하도록 합니다. 결과는 92 페이지 "프로젝트의 각 작업에 대한 LWP 최대값을 설정하는 방법"의 결과와 비슷합니다.

1. **관리자로 전환합니다.**

2. **`newtask`를 사용하여 `x-files` 프로젝트를 연결합니다.**

```
# newtask -p x-files
```

3. **`id` 명령을 `-p` 옵션과 함께 사용하여 올바른 프로젝트가 연결되어 있는지 확인합니다.**

```
# id -p
uid=0(root) gid=1(other) projid=101(x-files)
```

4. **`project.max-lwps`에 대해 LWP 수를 세 개로 제한하는 권한이 있는 새 값을 추가합니다.**

```
# prctl -n project.max-lwps -t privileged -v 3 -e deny -i project x-files
```

5. **결과를 확인합니다.**

```
# prctl -n project.max-lwps -i project x-files
process: 111108: csh
NAME    PRIVILEGE    VALUE    FLAG   ACTION           RECIPIENT
project.max-lwps
        usage          203
        privileged    1000       -   deny                   -
        system       2.15G     max   deny                   -
```

▼ 리소스 제어 값을 낮추기 위해 prctl을 사용하는 방법

1. **관리자로 전환합니다.**

2 **prctl 명령을 -r 옵션과 함께 사용하여 process.max-file-descriptor 리소스 제어의 가장 낮은 값을 변경합니다.**

```
# prctl -n process.max-file-descriptor -r -v 128 $$
```

▼ 프로젝트에 대한 제어 값을 표시, 대체 및 확인하기 위해 prctl을 사용하는 방법

1 **관리자로 전환합니다.**

2 *group.staff* **프로젝트에서 project.cpu-shares 값을 표시합니다.**

```
# prctl -n project.cpu-shares -i project group.staff
project: 2: group.staff
NAME    PRIVILEGE       VALUE    FLAG   ACTION     RECIPIENT
project.cpu-shares
        usage              1
        privileged         1       -   none               -
        system          65.5K     max   none
```

3 **현재 project.cpu-shares 값 1을 값 10으로 대체합니다.**

```
# prctl -n project.cpu-shares -v 10 -r -i project group.staff
```

4 **group.staff 프로젝트에서 project.cpu-shares 값을 표시합니다.**

```
# prctl -n project.cpu-shares -i project group.staff
project: 2: group.staff
NAME    PRIVILEGE       VALUE    FLAG   ACTION     RECIPIENT
project.cpu-shares
        usage              1
        privileged         1       -   none                                 -
        system          65.5K     max   none
```

rctladm 사용

rctladm을 사용하는 방법

rctladm 명령을 사용하여 리소스 제어 기능의 전역 상태에 대해 런타임 질의 및 수정을 수행합니다. 자세한 내용은 rctladm(1M) 매뉴얼 페이지를 참조하십시오.

예를 들어, rctladm을 -e 옵션과 함께 사용하여 리소스 제어의 전역 syslog 속성을 설정할 수 있습니다. 제어가 초과되는 경우 지정된 syslog 레벨에 알림이 기록됩니다. process.max-file-descriptor의 전역 syslog 속성을 사용으로 설정하려면 다음을 입력합니다.

```
# rctladm -e syslog process.max-file-descriptor
```

인수와 함께 사용하는 경우 rctladm 명령은 각 리소스 제어에 대한 전역 유형 플래그를 포함하여 전역 플래그를 표시합니다.

```
# rctladm
process.max-port-events     syslog=off  [ deny count ]
process.max-msg-messages    syslog=off  [ deny count ]
process.max-msg-qbytes      syslog=off  [ deny bytes ]
process.max-sem-ops         syslog=off  [ deny count ]
process.max-sem-nsems       syslog=off  [ deny count ]
process.max-address-space   syslog=off  [ lowerable deny no-signal bytes ]
process.max-file-descriptor syslog=off  [ lowerable deny count ]
process.max-core-size       syslog=off  [ lowerable deny no-signal bytes ]
process.max-stack-size      syslog=off  [ lowerable deny no-signal bytes ]
.
.
.
```

ipcs 사용

ipcs를 사용하는 방법

ipcs 유틸리티를 사용하여 활성 IPC(프로세스간 통신) 기능에 대한 정보를 표시합니다. 자세한 내용은 ipcs(1) 매뉴얼 페이지를 참조하십시오.

ipcs를 -J 옵션과 함께 사용하여 할당된 IPC 객체를 제한하는 프로젝트를 확인할 수 있습니다.

```
# ipcs -J
    IPC status from <running system> as of Wed Mar 26 18:53:15 PDT 2003
T         ID      KEY        MODE       OWNER    GROUP    PROJECT
Message Queues:
Shared Memory:
m       3600      0       --rw-rw-rw-   uname    staff    x-files
m        201      0       --rw-rw-rw-   uname    staff    x-files
m       1802      0       --rw-rw-rw-   uname    staff    x-files
m        503      0       --rw-rw-rw-   uname    staff    x-files
m        304      0       --rw-rw-rw-   uname    staff    x-files
m        605      0       --rw-rw-rw-   uname    staff    x-files
m          6      0       --rw-rw-rw-   uname    staff    x-files
m        107      0       --rw-rw-rw-   uname    staff    x-files
Semaphores:
s          0      0       --rw-rw-rw-   uname    staff    x-files
```

용량 경고

리소스 제어에 대한 전역 작업을 사용하여 너무 낮게 설정된 리소스 제어 값을 초과하는 항목에 대한 알림을 받을 수 있습니다.

예를 들어, 웹 서버에 일반적인 작업 부하를 위한 충분한 CPU가 있는지 여부를 확인하려고 할 수 있습니다. 유휴 CPU 시간 및 로드 평균에 대한 `sar` 데이터를 분석할 수 있습니다. 또한 확장 계정 데이터를 검사하여 웹 서버 프로세스에 대해 동시에 실행 중인 프로세스의 수를 확인할 수도 있습니다.

그러나 좀 더 쉬운 방법은 웹 서버를 작업 내에 배치하는 것입니다. 그런 다음 `syslog`를 사용하여 전역 작업을 설정하면 작업에서 시스템 용량에 적합한 예약된 LWP 수를 초과할 때마다 알림을 받을 수 있습니다.

자세한 내용은 `sar(1)` 매뉴얼 페이지를 참조하십시오.

▼ 웹 서버에 충분한 CPU 용량이 할당되어 있는지 여부를 확인하는 방법

1 `prctl` 명령을 사용하여 `httpd` 프로세스를 포함하는 작업에 권한이 있는(루트 소유) 리소스 제어를 배치합니다. 각 작업의 LWP 총 수를 40으로 제한하고 모든 로컬 작업을 사용 안함으로 설정합니다.

```
# prctl -n task.max-lwps -v 40 -t privileged -d all 'pgrep httpd'
```

2 `task.max-lwps` 리소스 제어에서 시스템 로그 전역 작업을 사용으로 설정합니다.

```
# rctladm -e syslog task.max-lwps
```

3 작업 부하가 리소스 제어를 초과하는지 여부를 관찰합니다.

초과하는 경우 다음과 같은 `/var/adm/messages`가 표시됩니다.

```
Jan  8 10:15:15 testmachine unix: [ID 859581 kern.notice]
NOTICE: privileged rctl task.max-lwps exceeded by task 19
```

◆ ◆ ◆ 8 장

8 FSS(Fair Share Scheduler)(개요)

작업 부하 데이터의 분석은 특정 작업 부하 또는 작업 부하 그룹이 CPU 리소스를 독점 사용하고 있음을 나타냅니다. 이러한 작업 부하가 CPU 사용에 대한 리소스 제약 조건에 위배되지 않을 경우 시스템에 대한 CPU 시간의 할당 정책을 수정할 수 있습니다. 이 장에서 설명된 Fair Share Scheduling 클래스를 사용하면 TS(시간 공유) 예약 클래스의 우선 순위 체계 대신에 공유 기반 CPU 시간을 할당할 수 있습니다.

이 장에서는 다음 내용을 다룹니다.

FSS(Fair Share Scheduler)를 사용하려면 9 장, "FSS(Fair Share Scheduler) 관리(작업)"를 참조하십시오.

스케줄러 소개

운영 체제의 기본 작업은 시스템의 리소스에 대한 액세스 권한을 얻는 프로세스를 중재하는 것입니다. 전달자라고 하는 프로세스 스케줄러는 프로세스에 대한 CPU 할당을 제어하는 커널 부분입니다. 스케줄러는 예약 클래스 개념을 제공합니다. 각 클래스는 해당 클래스 내에서 프로세스를 예약하는 데 사용되는 예약 정책을 정의합니다. Oracle Solaris 운영 체제의 기본 스케줄러인 TS 스케줄러는 사용할 수 있는

CPU에 대한 상대적으로 동일한 액세스 권한을 모든 프로세스에 제공하려 합니다. 하지만 다른 프로세스보다 더 많은 리소스가 특정 프로세스에 제공되도록 지정하고 싶을 수도 있습니다.

FSS(*Fair Share Scheduler*)를 사용하면 중요성에 따라 사용할 수 있는 CPU 리소스를 작업 부하에 할당하는 일을 제어할 수 있습니다. 이 중요성은 각 작업 부하에 지정하는 CPU 리소스의 **할당** 수로 표현됩니다.

각 프로젝트에 CPU 할당 수를 제공하여 CPU 리소스에 대한 프로젝트의 권리를 제어할 수 있습니다. FSS는 프로젝트에 연결된 프로세스 수와는 상관없이 할당된 수에 따라 프로젝트에 CPU 리소스를 공정하게 분산할 수 있습니다. FSS는 다른 프로젝트에 따라 사용량이 적은 경우 권리를 높이고 CPU 사용량이 많은 경우 프로젝트의 권리를 낮춤으로써 공정성을 실현합니다.

FSS는 `dispadmin(1M)` 및 `priocntl(1)` 명령의 클래스별 버전과 커널 예약 클래스 모듈로 구성됩니다. FSS에서 사용하는 프로젝트 할당은 `project(4)` 데이터베이스의 `project.cpu-shares` 등록 정보를 통해 지정됩니다.

주 – 영역이 설치된 Oracle Solaris 시스템에서 `project.cpu-shares` 리소스 제어를 사용하는 경우 222 페이지 "영역 구성 데이터", 335 페이지 "비전역 영역에서 사용되는 리소스 제어" 및 366 페이지 "영역이 설치된 Oracle Solaris 시스템에서 Fair Share Scheduler 사용"을 참조하십시오.

CPU 할당 정의

"할당"이라는 용어는 프로젝트에 지정되는 시스템의 CPU 리소스 양을 정의하는 데 사용됩니다. 프로젝트에 다른 프로젝트보다 더 많은 CPU 수를 지정할 경우 해당 프로젝트는 FSS(Fair Share Scheduler)에서 더 많은 CPU 리소스를 받습니다.

CPU 할당은 CPU 리소스 백분율과는 다릅니다. 할당은 다른 작업 부하에 대한 해당 작업 부하의 상대적 중요성을 정의하는 데 사용됩니다. CPU 할당을 프로젝트에 지정할 경우 프로젝트에 지정된 할당 수는 기본적으로 문제가 되지 않습니다. 다른 프로젝트와 비교하여 해당 프로젝트에 얼마나 많은 수가 할당되는지를 아는 것은 매우 중요합니다. 또한 이러한 다른 프로젝트 중에서 몇 개와 CPU 리소스 경합을 벌여야 하는지도 고려해야 합니다.

주 – 할당 수가 0개인 프로젝트의 프로세스는 항상 최저 우선 순위(0)로 실행됩니다. 이러한 프로젝트는 할당 수가 0이 아닌 프로젝트가 CPU 리소스를 사용하지 않는 경우에만 실행됩니다.

CPU 할당 및 프로세스 상태

Oracle Solaris 시스템에서는 프로젝트 작업 부하가 일반적으로 두 개 이상의 프로세스로 구성됩니다. FSS(Fair Share Scheduler) 관점에서는 각 프로젝트 작업 부하가 **유휴** 상태 또는 **활성** 상태일 수 있습니다. 프로젝트가 CPU 리소스를 사용하지 않는 경우 해당 프로젝트를 유휴 상태로 간주합니다. 이는 일반적으로 해당 프로세스가 **일시 정지**(I/O 완료 대기) 또는 중지됨 상태임을 의미합니다. 프로세스 중 하나 이상이 CPU 리소스를 사용 중인 경우 프로젝트를 활성 상태로 간주합니다. 모든 활성 프로젝트의 할당 수 합계는 프로젝트에 지정될 CPU 리소스 양을 계산하는 데 사용됩니다.

더 많은 프로젝트가 활성화되는 경우 각 프로젝트의 CPU 할당이 감소되지만 다양한 프로젝트의 할당 간 비례가 변경되지 않습니다.

CPU 할당과 사용률

배분 할당은 사용률과는 다릅니다. CPU 리소스의 50%가 할당된 프로젝트는 평균 20%의 CPU를 사용할 수 있습니다. 또한, 할당 수는 다른 프로젝트와 경합이 있는 경우에만 CPU 사용을 제한하는 역할을 합니다. 프로젝트의 할당이 얼마나 적든 상관없이 시스템에서 단독으로 실행 중인 경우 항상 처리 능력의 100%를 받습니다. 사용 가능한 CPU 주기가 낭비되지 않습니다. 사용 가능한 CPU 주기가 프로젝트 간에 분산됩니다.

사용 중인 작업 부하에 할당 수가 적게 지정되면 성능이 저하될 수 있습니다. 하지만 시스템이 과부하 상태가 아닌 경우 작업 부하가 작업 완료를 방해하지 않습니다.

CPU 배분 할당의 예

각각 *A*와 *B*라는 두 가지 병렬 CPU 바인딩 작업 부하를 실행하는 CPU 두 개가 장착된 시스템이 있다고 가정해 보겠습니다. 각 작업 부하는 개별 프로젝트로 실행 중입니다. 이러한 각각의 프로젝트는 프로젝트 *A*에 S_A 할당이 지정되고 *B*에 S_B 할당이 지정되도록 구성되었습니다.

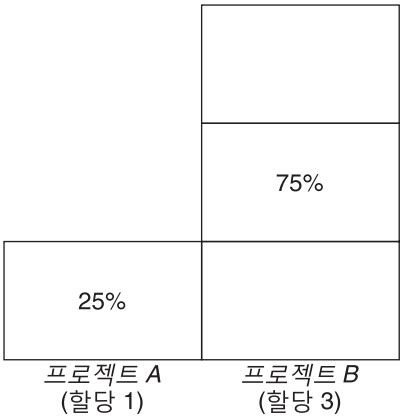
평균적으로 기존 TS 스케줄러에서는 시스템에서 실행 중인 각 작업 부하에 동일한 양의 CPU 리소스가 제공됩니다. 각 작업 부하는 시스템 용량의 50%를 얻게 됩니다.

S_A=S_B를 사용하여 FSS 스케줄러의 제어 하에서 실행할 경우 이러한 프로젝트에는 거의 동일한 CPU 리소스가 제공됩니다. 하지만 프로젝트에 서로 다른 할당 수가 지정되는 경우 CPU 리소스 할당이 다릅니다.

다음 세 가지 예는 서로 다른 구성에서의 할당 수 작동 방식을 보여 줍니다. 이러한 예는 요구량이 사용 가능한 리소스와 일치하거나 초과할 경우에만 수치상 할당에 따라 사용량을 정확히 표시됨을 보여 줍니다.

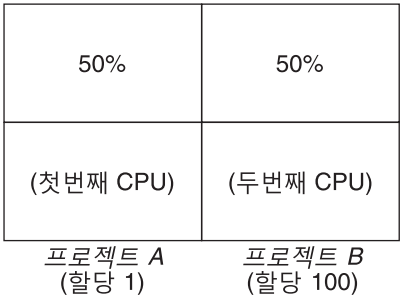
예 1: 각 프로젝트의 CPU에 바인딩된 두 개 프로세스

*A*와 *B* 각각에 CPU에 바인딩된 두 개 프로세스가 있으며, $S_A = 1$과 $S_B = 3$인 경우 총 할당 수는 1 + 3 = 4입니다. 이 구성에서는 CPU 수요가 충분히 제공된 경우 프로젝트 *A* 및 *B*에 각각 25%와 75%의 리소스가 할당됩니다.



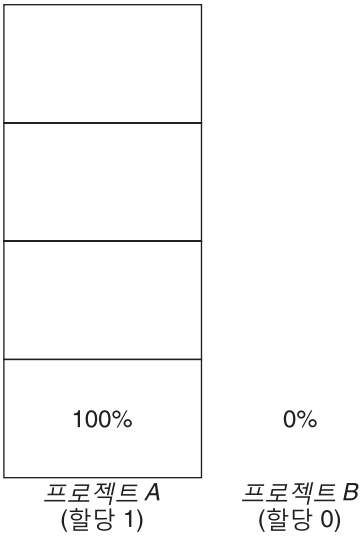
예 2: 프로젝트 간 경합 없음

*A*와 *B*에 각각 CPU에 바인딩된 프로세스 **하나**가 있고 $S_A = 1$ 및 $S_B = 100$인 경우 총 할당 수는 101입니다. 각 프로젝트에는 실행 중인 프로세스가 하나뿐이므로 각 프로젝트에서 두 개 이상의 CPU를 사용할 수 없습니다. 이 구성에서는 프로젝트 간에 CPU 리소스 경합이 없으므로 프로젝트 *A* 및 *B*에 각각 모든 CPU 리소스의 50%가 할당됩니다. 이 구성에서는 CPU 할당 값이 무관합니다. 두 프로젝트에 할당 0이 지정되었더라도 프로젝트의 할당은 동일합니다(50/50).



예 3: 프로젝트 한 개를 실행할 수 없음

*A*와 *B*에는 각각 두 개의 CPU에 바인딩된 프로세스가 있고 프로젝트 *A*에 할당 1이 제공되며 프로젝트 *B*에 할당 0이 제공된 경우 프로젝트 *B*에는 CPU 리소스가 할당되지 않으며 프로젝트 *A*에 모든 CPU 리소스가 할당됩니다. *B*의 프로세스는 항상 시스템 우선 순위 0에서 실행되므로 프로젝트 *A*의 프로세스에 더 높은 우선 순위가 지정되었기 때문에 B의 프로세스를 실행할 수 없습니다.



FSS 구성

프로젝트 및 사용자

프로젝트는 FSS 스케줄러의 작업 부하 컨테이너입니다. 프로젝트에 지정된 사용자 그룹이 단일 제어 가능 블록으로 처리됩니다. 개별 사용자마다 고유한 할당 수로 프로젝트를 만들 수 있습니다.

사용자는 할당이 서로 다르게 지정된 여러 프로젝트의 구성원일 수 있습니다. 한 프로젝트에서 다른 프로젝트로 프로세스를 옮김으로써 프로세스에 가변 양의 CPU 리소스가 지정될 수 있습니다.

`project(4)` 데이터베이스 및 이름 서비스에 대한 자세한 내용은 39 페이지 “project 데이터베이스”를 참조하십시오.

CPU 할당 구성

CPU 할당의 구성은 이름 서비스를 통해 project 데이터베이스의 등록 정보로 관리됩니다.

setproject(3PROJECT) 라이브러리 함수를 통해 프로젝트와 연결된 첫번째 작업(또는 프로세스)을 생성하는 경우 project 데이터베이스에 project.cpu-shares 리소스 제어로 정의된 CPU 할당 수가 커널에 전달됩니다. project.cpu-shares 리소스 제어가 정의되지 않은 프로젝트에는 할당 1이 지정됩니다.

다음 예에서 /etc/project 파일의 이 항목은 *x-files* 프로젝트에 대한 할당 수를 5로 설정합니다.

```
x-files:100::::project.cpu-shares=(privileged,5,none)
```

프로젝트가 이미 실행 중일 때 데이터베이스에서 프로젝트에 할당된 CPU 할당 수를 변경할 경우에는 해당 프로젝트에 대한 할당 수가 수정되지 않습니다. 변경 사항을 적용하려면 프로젝트를 다시 시작해야 합니다.

project 데이터베이스에서 프로젝트의 속성을 변경하지 않고 프로젝트에 지정된 할당 수를 임시로 변경하려면 prctl 명령을 사용합니다. 예를 들어 해당 프로젝트와 연결된 프로세스가 실행 중인 동안 *x-files* 프로젝트의 project.cpu-shares 리소스 제어 값을 *3*으로 변경하려면 다음을 입력합니다.

```
# prctl -r -n project.cpu-shares -v 3 -i project x-files
```

자세한 내용은 prctl(1) 매뉴얼 페이지를 참조하십시오.

-r	이름이 지정된 리소스 제어의 현재 값을 바꿉니다.
-n *name*	리소스 제어의 이름을 지정합니다.
-v *val*	리소스 제어의 값을 지정합니다.
-i *idtype*	다음 인수의 ID 형식을 지정합니다.
x-files	변경 객체를 지정합니다. 이 예서는 *x-files* 프로젝트가 객체입니다.

프로젝트 ID가 0인 system 프로젝트에는 부트 시 초기화 스크립트를 통해 시작되는 모든 시스템 데몬이 포함됩니다. system은 할당 수에 제한이 없는 프로젝트로 표시될 수 있습니다. 다시 말해서 다른 프로젝트에 지정된 할당 수와 상관없이 system이 항상 첫번째로 예약된다는 뜻입니다. system 프로젝트에 지정되는 할당 수를 제한하려면 project 데이터베이스에서 이 프로젝트에 대한 할당 수를 지정합니다.

앞에서 설명했듯이, 할당 수가 0인 프로젝트에 속하는 프로세스에는 항상 시스템 우선 순위 0이 지정됩니다. 할당 수가 1 이상인 프로젝트는 1 이상의 우선 순위로 실행됩니다. 따라서, 할당 수가 0인 프로젝트는 할당 수가 0이 아닌 프로젝트로부터의 요청이 없어서 사용 가능한 CPU 리소스가 남아 있을 때만 예약됩니다.

한 프로젝트에 지정할 수 있는 최대 할당 수는 65535입니다.

FSS와 프로세서 세트

FSS를 프로세서 세트와 함께 사용하면 프로세서 세트만 사용할 때보다 각 프로세서 세트에서 실행하는 프로젝트 간 CPU 리소스 할당을 보다 세부적으로 제어할 수 있습니다. FSS 스케줄러는 프로세서 세트를 완전히 독립된 분할 영역으로 간주하므로, 각 프로세서 세트가 CPU 할당과 관련하여 독립적으로 제어됩니다.

한 프로세서 세트에서 실행되는 프로젝트의 CPU 할당은 프로젝트가 동일한 리소스를 놓고 경합하지 않으므로 다른 프로세서 세트에서 실행되는 프로젝트의 작업이나 CPU 할당 수에 영향을 주지 않습니다. 프로젝트는 동일한 프로세서 세트 내에서 실행 중인 경우에만 서로 간에 경합을 벌입니다.

프로젝트에 지정된 할당 수는 시스템 전체에 적용됩니다. 어느 프로세서 세트에서 실행 중인지와 상관없이 프로젝트의 각 부분에는 동일한 할당이 제공됩니다.

프로세서 세트를 사용하는 경우 각 프로세서 세트 내에서 실행되는 활성 프로젝트에 대해 프로젝트 CPU 할당이 계산됩니다.

다른 프로세서 세트에서 실행되는 프로젝트 분할 영역의 경우 CPU 할당이 다를 수도 있습니다. 프로세서 세트 내 각 프로젝트 분할 영역의 CPU 할당은 동일 프로세서 세트에서 실행되는 다른 프로젝트의 할당에만 종속됩니다.

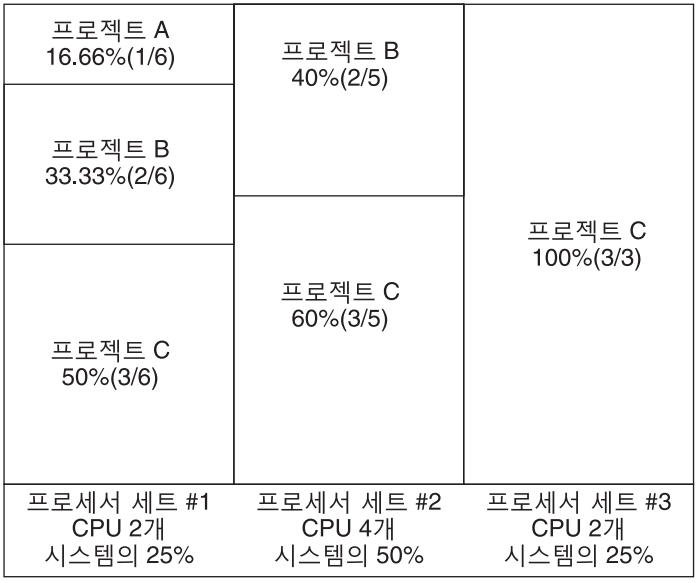
프로세서 세트의 경계 내에서 실행되는 응용 프로그램의 성능 및 가용성은 새 프로세서 세트가 추가되어도 영향을 받지 않습니다. 또한 응용 프로그램은 다른 프로세서 세트에서 실행되는 프로젝트의 할당이 변경되어도 영향을 받지 않습니다.

빈 프로세서 세트(세트 내 프로세서가 없음)나 해당 세트에 바인딩된 프로세스가 없는 프로세서 세트는 FSS 스케줄러 동작에 아무런 영향을 주지 않습니다.

FSS 및 프로세서 세트 예

CPU가 8개인 서버가 프로젝트 *A*, *B* 및 *C*에서 CPU에 바인딩된 몇 개 응용 프로그램을 실행 중이라고 가정해 보겠습니다. 프로젝트 *A*에 한 개가 할당되고, 프로젝트 *B*에 두 개가 할당되며, 프로젝트 *C*에 세 개가 할당됩니다.

프로젝트 *A*는 프로세서 세트 1에서만 실행 중입니다. 프로젝트 *B*는 프로세서 세트 1과 2에서 실행 중입니다. 프로젝트 *C*는 프로세서 세트 1, 2 및 3에서 실행 중입니다. 각 프로젝트에 사용 가능한 모든 CPU 처리 능력을 활용할 수 있는 충분한 프로세스가 있다고 가정해 보겠습니다. 따라서 각 프로세서 세트에서는 항상 CPU 리소스 경쟁이 벌어집니다.



다음 표는 이러한 시스템 하나에서 시스템 전체의 프로젝트 CPU 할당을 보여 줍니다.

프로젝트	할당
프로젝트 A	$4\% = (1/6 \text{ X } 2/8)_{pset1}$
프로젝트 B	$28\% = (2/6 \text{ X } 2/8)_{pset1} + (2/5 * 4/8)_{pset2}$
프로젝트 C	$67\% = (3/6 \text{ X } 2/8)_{pset1} + (3/5 \text{ X } 4/8)_{pset2} + (3/3 \text{ X } 2/8)_{pset3}$

이러한 백분율은 프로젝트에 제공되는 해당 CPU 할당량과 일치하지 않습니다. 하지만 각 프로세서 세트 내에서 프로젝트당 CPU 할당 비율은 각각의 할당에 비례합니다.

프로세서 세트가 **없는** 동일 시스템에서는 CPU 리소스의 할당이 다음 표에서처럼 다르게 나타납니다.

프로젝트	할당
프로젝트 A	16.66% = (1/6)
프로젝트 B	33.33% = (2/6)
프로젝트 C	50% = (3/6)

FSS를 다른 예약 클래스와 결합

기본적으로 FSS 예약 클래스는 TS(시간 공유), IA(대화형) 및 FX(고정 우선 순위) 예약 클래스와 동일한 우선 순위 범위(0 ~ 59)를 사용합니다. 따라서 가급적이면 이러한 예약 클래스의 프로세스가 **동일한** 프로세서 세트를 공유하지 않도록 해야 합니다. FSS, TS, IA 및 FX 클래스의 프로세스 혼합으로 예기치 않은 예약 동작이 발생할 수 있습니다.

프로세서 세트를 사용하는 경우 한 시스템에서 TS, IA 및 FX와 FSS를 함께 사용할 수 있습니다. 하지만 각 프로세서 세트에서 실행되는 모든 프로세서가 **하나의** 예약 클래스 안에 있어야 하므로 동일 CPU를 놓고 경합이 벌어지지 않습니다. 특히 FX 스케줄러는 프로세서 세트가 사용되는 경우가 아니면 FSS와 함께 사용해서는 안 됩니다. 이렇게 하면 FX 클래스의 응용 프로그램이 너무 높은 우선 순위를 사용하게 되어 FSS 클래스의 응용 프로그램에 사용될 리소스가 부족해지는 경우를 막을 수 있습니다.

동일한 프로세서에서 또는 프로세서 세트가 없는 동일 시스템에서 TS 및 IA 클래스의 프로세스를 함께 사용할 수 있습니다.

Oracle Solaris 시스템은 루트 권한을 가진 사용자에게 RT(실시간) 스케줄러도 제공합니다. 기본적으로 RT 예약 클래스는 FSS와는 다른 범위(일반적으로 100 ~ 159)에서 시스템 우선 순위를 사용합니다. RT 및 FSS가 *disjoint* 또는 겹치지 않는 우선 순위 범위를 사용 중이므로 FSS가 동일한 프로세서 세트 내에서 RT 예약 클래스와 공존할 수 있습니다. 하지만 FSS 예약 클래스는 RT 클래스에서 실행되는 프로세스를 제어할 수 없습니다.

예를 들면, 4-프로세서 시스템에서 단일 스레드 RT 프로세스는 프로세스가 CPU에 바인딩된 경우 하나의 전체 프로세서를 사용할 수 있습니다. 시스템에서 FSS도 실행하는 경우 일반 사용자 프로세스는 RT 프로세스에 사용되고 있지 않은 남은 CPU 3개를 놓고 경합을 벌이게 됩니다. RT 프로세스는 CPU를 계속적으로 사용할 수 없습니다. RT 프로세스가 유휴 상태가 되면 FSS에서 4개 프로세서 모두를 활용합니다.

다음 명령을 입력하여 프로세서 세트가 실행 중인 예약 클래스를 찾고 각 프로세서가 TS, IA, FX 또는 FSS 프로세스를 실행하도록 구성되어 있는지 확인할 수 있습니다.

```
$ ps -ef -o pset,class | grep -v CLS | sort | uniq
1 FSS
1 SYS
2 TS
2 RT
3 FX
```

시스템용 예약 클래스 설정

시스템용 기본 예약 클래스를 설정하려면 113 페이지 “FSS를 기본 스케줄러 클래스로 설정하는 방법”, 209 페이지 “예약 클래스” 및 dispadmin(1M)을 참조하십시오. 실행 중인 프로세스를 다른 예약 클래스로 이동하려면 113 페이지 “FSS 구성” 및 priocntl(1)을 참조하십시오.

영역이 설치된 시스템의 예약 클래스

비전역 영역은 시스템용 기본 예약 클래스를 사용합니다. 시스템을 새로운 기본 예약 클래스 설정으로 업데이트하는 경우 비전역 영역은 부트 또는 재부트될 때 새 설정을 얻게 됩니다.

여기서의 기본적인 FSS 사용 방법은 dispadmin 명령을 사용하여 FSS를 시스템 기본 예약 클래스로 설정하는 것입니다. 그러면 모든 영역에 시스템 CPU 리소스가 공정하게 할당됩니다. 영역이 사용 중인 경우 예약 클래스에 대한 자세한 내용은 209 페이지 “예약 클래스”를 참조하십시오.

기본 예약 클래스 변경 및 재부트 없이 실행 중인 프로세스를 다른 예약 클래스로 이동하는 작업에 대한 자세한 내용은 표 25–5 and the priocntl(1) 매뉴얼 페이지를 참조하십시오.

FSS에 사용되는 명령

다음 표에 표시된 명령은 FSS(Fair Share Scheduler)에 대한 기본 관리 인터페이스를 제공합니다.

명령 참조	설명
priocntl(1)	지정된 프로세스의 예약 매개 변수를 표시하거나 설정하고, 실행 중인 프로세스를 다른 예약 클래스로 이동합니다.
ps(1)	실행 중인 프로세스에 대한 정보를 나열하고 프로세서 세트가 실행 중인 예약 클래스를 식별합니다.
dispadmin(1M)	시스템에서 사용 가능한 스케줄러를 나열합니다. 시스템용 기본 스케줄러를 설정합니다. FSS 스케줄러의 시간 양자 값을 검사 및 조정하는 데 사용됩니다.
FSS(7)	FSS(Fare Share Scheduler)를 설명합니다.

◆ ◆ ◆ 9 장

9 FSS(Fair Share Scheduler) 관리(작업)

이 장에서는 FSS(Fare Share Scheduler) 사용 방법에 대해 설명합니다.

FSS 개요는 8 장, “FSS(Fair Share Scheduler)(개요)”를 참조하십시오. 영역을 사용 중인 경우 예약 클래스에 대한 자세한 내용은 209 페이지 “예약 클래스”를 참조하십시오.

FSS(Fair Share Scheduler) 관리(작업 맵)

작업	설명	정보
CPU 사용을 모니터링합니다.	프로젝트의 CPU 사용과 프로세서 세트의 프로젝트를 모니터링합니다.	112 페이지 “FSS 모니터링”
기본 스케줄러 클래스를 설정합니다.	FSS와 같은 스케줄러를 시스템에 대한 기본 스케줄러로 설정합니다.	113 페이지 “FSS를 기본 스케줄러 클래스로 설정하는 방법”
실행 중인 프로세스를 하나의 스케줄러 클래스에서 FSS 클래스와 같은 다른 예약 클래스로 이동합니다.	기본 예약 클래스를 변경 및 재부트하지 않고 프로세스를 하나의 예약 클래스에서 다른 예약 클래스로 수동으로 이동합니다.	114 페이지 “TS 클래스에서 FSS 클래스로 프로세스를 수동으로 이동하는 방법”
모든 예약 클래스에서 실행 중인 모든 프로세스를 FSS 클래스와 같은 다른 예약 클래스로 이동합니다.	기본 예약 클래스를 변경 및 재부트하지 않고 모든 예약 클래스의 프로세스를 다른 예약 클래스로 수동으로 이동합니다.	114 페이지 “모든 사용자 클래스에서 FSS 클래스로 프로세스를 수동으로 이동하는 방법”
프로젝트의 프로세스를 FSS 클래스와 같은 다른 예약 클래스로 이동합니다.	현재 예약 클래스에서 다른 예약 클래스로 프로젝트의 프로세스를 수동으로 이동합니다.	115 페이지 “프로젝트의 프로세스를 FSS 클래스로 수동으로 이동하는 방법”

작업	설명	정보
FSS 매개 변수를 검사하고 조정합니다.	스케줄러의 시간 양자 값을 조정합니다. **시간 양자**는 프로세서를 포기하기 전에 스레드를 실행하도록 허용되는 시간을 말합니다.	115 페이지 “스케줄러 매개 변수를 조정하는 방법”

FSS 모니터링

prstat(1M) 매뉴얼 페이지에 설명된 prstat 명령을 사용하여 활성 프로젝트별로 CPU 사용을 모니터링할 수 있습니다.

작업에 대한 확장 회계 데이터를 사용하여 장기간에 걸쳐 사용되는 CPU 리소스 양에 대한 프로젝트별 통계를 구할 수 있습니다. 자세한 내용은 4 장, “확장 계정(개요)”을 참조하십시오.

▼ 프로젝트별로 시스템 CPU 사용을 모니터링하는 방법

- **시스템에서 실행되는 프로젝트의 CPU 사용을 모니터링하려면 prstat 명령을 -J 옵션과 함께 사용합니다.**

```
# prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  5107 root     4556K 3268K cpu0    59    0   0:00:00 0.0% prstat/1
  4570 root       83M   47M sleep   59    0   0:00:25 0.0% java/13
  5105 bobbyc   3280K 2364K sleep   59    0   0:00:00 0.0% su/1
  5106 root     3328K 2580K sleep   59    0   0:00:00 0.0% bash/1
     5 root       0K    0K sleep   99  -20   0:00:14 0.0% zpool-rpool/138
   333 daemon   7196K 2896K sleep   59    0   0:00:07 0.0% rcapd/1
    51 netcfg   4436K 3460K sleep   59    0   0:00:01 0.0% netcfgd/5
  2685 root     3328K 2664K sleep   59    0   0:00:00 0.0% bash/1
   101 netadm   4164K 2824K sleep   59    0   0:00:01 0.0% ipmgmtd/6
   139 root     6940K 3016K sleep   59    0   0:00:00 0.0% syseventd/18
  5082 bobbyc   2236K 1700K sleep   59    0   0:00:00 0.0% csh/1
    45 root       15M 7360K sleep   59    0   0:00:01 0.0% dlmgmtd/7
    12 root       23M   22M sleep   59    0   0:00:45 0.0% svc.configd/22
    10 root       15M   13M sleep   59    0   0:00:05 0.0% svc.startd/19
   337 netadm   6768K 5620K sleep   59    0   0:00:01 0.0% nwamd/9
PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
     1        6   25M   18M   0.9%   0:00:00 0.0% user.root
     0       73  479M  284M    14%   0:02:31 0.0% system
     3        4   28M   24M   1.1%   0:00:26 0.0% default
    10        2   14M 7288K   0.3%   0:00:00 0.0% group.staff

Total: 85 processes, 553 lwps, load averages: 0.00, 0.00, 0.00
```

▼ 프로세서 세트에서 프로젝트별로 CPU 사용을 모니터링하는 방법

- **프로세서 세트 목록에서 프로젝트의 CPU 사용을 모니터링하려면 다음을 입력합니다.**

  % **prstat -J -C** *pset-list*

  여기서 *pset-list*는 쉼표로 구분된 프로세서 세트 목록입니다.

FSS 구성

Oracle Solaris 시스템의 다른 예약 클래스에 사용하는 동일한 명령을 FSS에도 사용할 수 있습니다. 스케줄러 클래스를 설정하고 스케줄러의 조정 가능 매개 변수를 구성하며 개별 프로세스의 등록 정보를 구성할 수 있습니다.

`svcadm restart`를 사용하여 스케줄러 서비스를 다시 시작할 수 있습니다. 자세한 내용은 `svcadm`(1M)을 참조하십시오.

시스템의 스케줄러 클래스 나열

시스템의 스케줄러 클래스를 표시하려면 `dispadmin` 명령을 `-l` 옵션과 함께 사용합니다.

```
$ dispadmin -l
CONFIGURED CLASSES
==================

SYS      (System Class)
TS       (Time Sharing)
SDC      (System Duty-Cycle Class)
FSS      (Fair Share)
FX       (Fixed Priority)
IA       (Interactive)
```

▼ FSS를 기본 스케줄러 클래스로 설정하는 방법

CPU 할당 지정을 적용하려면 FSS가 시스템의 기본 스케줄러여야 합니다.

`priocntl` 및 `dispadmin` 명령의 조합을 사용하면 즉시 또는 재부트 후에 FSS를 기본 스케줄러로 설정할 수 있습니다.

1. **관리자로 전환합니다.**

2. **FSS가 시스템의 기본 스케줄러가 되도록 설정합니다.**

   # dispadmin -d FSS

이 변경 사항은 다음 재부트 시 적용됩니다. 재부트 후 시스템의 모든 프로세스가 FSS 예약 클래스에서 실행됩니다.

3 **재부트하지 않고 이 구성을 즉시 적용합니다.**

```
# priocntl -s -c FSS -i all
```

▼ TS 클래스에서 FSS 클래스로 프로세스를 수동으로 이동하는 방법

기본 예약 클래스를 변경하고 재부트하지 않아도 하나의 예약 클래스에서 다른 예약 클래스로 프로세스를 수동으로 이동할 수 있습니다. 이 절차에서는 TS 예약 클래스에서 FSS 예약 클래스로 프로세스를 수동으로 이동하는 방법을 보여 줍니다.

1 **관리자로 전환합니다.**

2 **init 프로세스(pid 1)를 FSS 예약 클래스로 이동합니다.**

```
# priocntl -s -c FSS -i pid 1
```

3 **TS 예약 클래스에서 FSS 예약 클래스로 모든 프로세스를 이동합니다.**

```
# priocntl -s -c FSS -i class TS
```

주 – 재부트 후에는 모든 프로세스가 다시 TS 예약 클래스에서 실행됩니다.

▼ 모든 사용자 클래스에서 FSS 클래스로 프로세스를 수동으로 이동하는 방법

TS 이외의 기본 클래스를 사용 중일 수도 있습니다. 예를 들면 시스템이 기본적으로 IA 클래스를 사용하는 창 환경에서 실행 중일 수 있습니다. 기본 예약 클래스를 변경하고 재부트하지 않아도 모든 프로세스를 FSS 예약 클래스로 수동으로 이동할 수 있습니다.

1 **관리자로 전환합니다.**

2 **init 프로세스(pid 1)를 FSS 예약 클래스로 이동합니다.**

```
# priocntl -s -c FSS -i pid 1
```

3 **현재 예약 클래스에서 FSS 예약 클래스로 모든 프로세스를 이동합니다.**

```
# priocntl -s -c FSS -i all
```

주 – 재부트 후에는 모든 프로세스가 다시 기본 예약 클래스에서 실행됩니다.

▼ 프로젝트의 프로세스를 FSS 클래스로 수동으로 이동하는 방법

현재 예약 클래스에서 FSS 예약 클래스로 프로젝트의 프로세스를 수동으로 이동할 수 있습니다.

1 **관리자로 전환합니다.**

2 **프로젝트 ID *10*에서 실행되는 프로세스를 FSS 예약 클래스로 이동합니다.**

```
# priocntl -s -c FSS -i projid 10
```

재부트 후에는 프로젝트의 프로세스가 다시 기본 예약 클래스에서 실행됩니다.

스케줄러 매개 변수를 조정하는 방법

`dispadmin` 명령을 사용하여 시스템이 실행 중인 동안 프로세스 스케줄러 매개 변수를 표시하거나 변경할 수 있습니다. 예를 들어 `dispadmin`을 사용하여 FSS 스케줄러의 시간 양자 값을 검사 및 조정할 수 있습니다. **시간 양자**는 프로세서를 포기하기 전에 스레드를 실행하도록 허용되는 시간을 말합니다.

시스템이 실행 중인 동안 FSS 스케줄러의 현재 시간 양자를 표시하려면 다음을 입력합니다.

```
$ dispadmin -c FSS -g
#
# Fair Share Scheduler Configuration
#
RES=1000
#
# Time Quantum
#
QUANTUM=110
```

`-g` 옵션을 사용하는 경우 `-r` 옵션을 사용하여 시간 양자 값을 인쇄하는 데 사용되는 해상도를 지정할 수도 있습니다. 해상도가 지정되지 않은 경우 기본적으로 시간 양자 값이 밀리초 단위로 표시됩니다.

```
$ dispadmin -c FSS -g -r 100
#
# Fair Share Scheduler Configuration
#
RES=100
#
```

```
# Time Quantum
#
QUANTUM=11
```

FSS 예약 클래스에 대한 예약 매개 변수를 설정하려면 `dispadmin -s`를 사용하십시오. *file*의 값은 `-g` 옵션에 의해 출력된 형식이어야 합니다. 이러한 값은 커널에서 현재 값을 덮어씁니다. 다음과 같이 입력하십시오.

```
$ dispadmin -c FSS -s file
```

◆ ◆ ◆ 10 장

10 리소스 상한값 지원 데몬을 사용한 물리적 메모리 제어(개요)

리소스 상한값 지원 데몬 rcapd를 사용하여 리소스 상한값이 정의되어 있는 프로젝트에서 실행 중인 프로세스의 물리적 메모리 사용을 규제할 수 있습니다. 시스템에서 영역을 실행 중인 경우, 전역 영역에서 rcapd를 사용하여 비전역 영역에서의 물리적 메모리 사용을 규제할 수 있습니다. 17 장, “비전역 영역 계획 및 구성(작업)”을 참조하십시오.

이 장에서는 다음 내용을 다룹니다.

rcapd 유틸리티 사용 절차에 대해서는 11 장, “리소스 상한값 지원 데몬 관리(작업)”를 참조하십시오.

리소스 상한값 지원 데몬 소개

리소스 **상한값**은 물리적 메모리 등과 같이 리소스 사용에 설정된 상한입니다. 프로젝트당 물리적 메모리 상한값이 지원됩니다.

리소스 상한값 지원 데몬 및 이와 연관된 유틸리티는 물리적 메모리 상한값 적용 및 관리를 위한 방식을 제공합니다.

리소스 제어와 마찬가지로 리소스 상한값을 project 데이터베이스의 프로젝트 항목 속성을 사용하여 정의할 수 있습니다. 그러나, 리소스 제어는 커널에 의해 동기적으로 적용되지만 리소스 상한값은 리소스 상한값 지원 데몬에 의해 사용자 레벨에서 비동기적으로 적용됩니다. 비동기 적용에서는 데몬에서 사용하는 샘플링 간격으로 인해 약간의 지연이 발생합니다.

rcapd에 대한 자세한 내용은 rcapd(1M) 매뉴얼 페이지를 참조하십시오. 프로젝트 및 project 데이터베이스에 대한 자세한 내용은 2 장, “프로젝트 및 작업(개요)” 및 project(4) 매뉴얼 페이지를 참조하십시오. 리소스 제어에 대한 자세한 내용은 6 장, “리소스 제어(개요)”를 참조하십시오.

리소스 상한값이 작동하는 방법

데몬은 물리적 메모리 상한값을 가진 프로젝트의 리소스 사용률을 반복적으로 샘플링합니다. 데몬에서 사용하는 샘플링 간격은 관리자가 지정합니다. 자세한 내용은 122 페이지 “샘플 간격 결정”을 참조하십시오. 시스템의 물리적 메모리 사용률이 상한값 적용 임계치를 초과하고 다른 조건이 충족되는 경우, 메모리 상한값 레벨 또는 이보다 낮게 프로젝트의 리소스 사용을 줄이기 위해 데몬이 조치를 취합니다.

가상 메모리 시스템은 물리적 메모리를 페이지라고 하는 세그먼트로 분할합니다. 페이지는 Oracle Solaris 메모리 관리 부속 시스템에서 물리적 메모리의 기본적인 단위입니다. 파일에서 메모리로 데이터를 읽어들이기 위해 가상 메모리 시스템은 파일에서 한 번에 한 페이지 또는 **여러 페이지**를 읽어들입니다. 리소스 사용을 줄이기 위해 데몬은 물리적 메모리 외부 영역에 있는 스왑 장치로 자주 사용되지 않는 페이지를 **페이징아웃**하거나 재배치할 수 있습니다.

데몬은 프로젝트 작업 부하의 상주 집합 크기를 작업 집합의 크기와 상대적으로 규제하여 물리적 메모리를 관리합니다. 상주 메모리 페이지 세트란 물리적 메모리에 상주하는 페이지 세트를 말합니다. 작업 집합은 작업 부하에서 해당 처리 주기 중에 활발히 사용하는 페이지의 집합입니다. 작업 집합은 프로세스의 작업 모드 및 처리되는 데이터 유형에 따라 시간이 경과되면 변경됩니다. 이상적으로는 작업 집합이 상주 상태를 유지할 수 있도록 모든 작업 부하가 충분한 물리적 메모리에 액세스할 수 있습니다. 그러나 작업 집합은 물리적 메모리에 들어가지 않는 메모리를 보관하기 위한 보조 디스크 저장소의 사용을 포함할 수도 있습니다.

지정된 시간에 rcapd의 한 인스턴스만 실행할 수 있습니다.

프로젝트의 물리적 메모리 사용 제한을 위한 속성

프로젝트의 물리적 메모리 리소스 상한값을 정의하려면 이 속성을 project 데이터베이스 항목에 추가하여 RSS(Resident Set Size)를 설정합니다.

rcap.max-rss 프로젝트에서 처리에 사용할 수 있는 물리적 메모리의 전체 용량(바이트)입니다.

예를 들어, /etc/project 파일의 다음 행은 db라는 프로젝트에 대해 10기가바이트의 RSS 상한값을 설정합니다.

```
db:100::db,root::rcap.max-rss=10737418240
```

주 – 시스템에서 지정된 상한값을 페이지 크기로 반올림할 수 있습니다.

`projmod` 명령을 사용하여 `/etc/project` 파일에서 `rcap.max-rss` 속성을 설정할 수도 있습니다.

자세한 내용은 RSS(Resident Set Size) 상한값 설정을 참조하십시오.

rcapd 구성

리소스 상한값 데몬을 구성하려면 `rcapadm` 명령을 사용합니다. 다음 작업을 수행할 수 있습니다.

- 상한값 적용을 위한 임계값 설정
- `rcapd`에서 수행하는 작업의 간격 설정
- 리소스 상한값 사용 또는 사용 안함
- 구성된 리소스 상한값 데몬의 현재 상태 표시

데몬을 구성하려면 루트 사용자이거나 필요한 관리 권한이 있어야 합니다.

구성 간격에 따라(121 페이지 “rcapd 작업 간격” 참조) 또는 필요에 따라 `SIGHUP`(`kill`(1) 매뉴얼 페이지 참조)을 보내 구성 변경 사항을 `rcapd`에 반영할 수 있습니다.

인수 없이 사용되는 경우 `rcapadm` 명령은 리소스 상한값 데몬이 구성된 경우 현재 상태를 표시합니다.

다음 세부절에서는 상한값 적용, 상한값 및 `rcapd` 작업 간격에 대해 설명합니다.

영역이 설치된 시스템에서 리소스 상한값 데몬 사용

영역을 구성할 때 `capped-memory` 리소스를 설정하여 영역의 RSS(Resident Set Size) 사용을 제어할 수 있습니다. 자세한 내용은 209 페이지 “물리적 메모리 제어 및 capped-memory 리소스”를 참조하십시오. `capped-memory` 리소스를 사용하려면 `resource-cap` 패키지가 전역 영역에 설치되어 있어야 합니다. 전역 영역을 포함하여 영역 **내에서** `rcapd` 명령을 실행하여 해당 영역의 프로젝트에 메모리 상한값을 적용할 수 있습니다.

다음 재부트 전까지 지정된 영역에서 사용할 수 있는 최대 메모리 용량에 대해 임시 상한값을 설정할 수 있습니다. 129 페이지 “영역에 대한 임시 리소스 상한값을 지정하는 방법”을 참조하십시오.

리소스 상한값이 정의된 프로젝트에서 실행 중인 프로세스의 물리적 메모리 사용을 규제하기 위해 영역에서 rcapd 명령을 사용하는 경우, 해당 영역에서 데몬을 구성해야 합니다.

다른 영역에 있는 응용 프로그램에 대한 메모리 상한값을 선택할 때는 일반적으로 다른 영역에 상주하는 응용 프로그램은 고려할 필요가 없습니다. 영역별 서비스는 예외입니다. 영역별 서비스는 메모리를 사용합니다. 시스템의 물리적 메모리 용량 및 메모리 상한값을 결정할 때 메모리 사용을 고려해야 합니다.

메모리 상한값 적용 임계치

메모리 상한값 적용 임계치는 시스템에서 상한값 적용을 트리거하는 물리적 메모리 사용률의 백분율입니다. 시스템에서 이 사용률을 초과하면 상한값이 적용됩니다. 응용 프로그램 및 커널에서 사용되는 물리적 메모리가 이 백분율에 포함됩니다. 사용률의 백분율에 따라 메모리 상한값이 적용되는 방법이 결정됩니다.

상한값을 적용하기 위해 프로젝트 작업 부하에서 메모리를 페이징아웃할 수 있습니다.

- 지정된 작업 부하에서 상한값을 초과하는 메모리 부분의 크기를 줄이기 위해 메모리를 페이징아웃할 수 있습니다.
- 시스템에서 메모리 상한값 적용 임계치를 초과하여 사용되는 물리적 메모리 부분을 줄이기 위해 메모리를 페이징아웃할 수 있습니다.

작업 부하는 물리적 메모리를 상한값까지 사용할 수 있습니다. 작업 부하는 시스템의 메모리 사용률이 메모리 상한값 적용 임계치 아래에 있는 한 추가 메모리를 사용할 수 있습니다.

상한값 적용을 위한 값을 설정하려면 127 페이지 “메모리 상한값 적용 임계치를 설정하는 방법”을 참조하십시오.

상한값 결정

프로젝트 상한값이 너무 낮게 설정되는 경우 정상적인 조건에서 작업 부하를 효율적으로 진행하기 위해 필요한 충분한 메모리가 없을 수 있습니다. 작업 부하에 추가 메모리가 필요하기 때문에 발생하는 페이징은 시스템 성능에 부정적인 영향을 줍니다.

상한값이 너무 높게 설정된 프로젝트는 해당 상한값을 초과하기 전에 사용 가능한 물리적 메모리를 사용할 수 있습니다. 이 경우 물리적 메모리는 rcapd가 아닌 커널에 의해 효율적으로 관리됩니다.

프로젝트 상한값 결정 시 이러한 요인을 고려합니다.

I/O 시스템에 대한 영향

데몬은 샘플링된 사용량이 프로젝트 상한값을 초과할 때마다 프로젝트 작업 부하의 물리적 메모리 사용량을 줄이려고 시도할 수 있습니다. 상한값 적용 중에 작업 부하에서 매핑한 파일을 포함하는 스왑 장치 및 기타 장치가 사용됩니다. 스왑 장치의 성능은 일상적으로 상한값을 초과하는 작업 부하의 성능을 결정하는 데 중요한 요인입니다. 작업 부하의 실행은 작업 부하의 상한값과 동일한 물리적 메모리 용량을 가진 시스템에서 이를 실행하는 것과 유사합니다.

CPU 사용량에 대한 영향

데몬의 CPU 사용량은 상한값을 제한하는 프로젝트 작업 부하의 프로세스 수 및 작업 부하의 주소 공간 크기에 따라 달라집니다.

데몬의 CPU 시간 중 작은 부분이 각 작업 부하의 사용량을 샘플링하는 데 소비됩니다. 작업 부하에 프로세스를 추가하면 사용량 샘플링에 소비되는 시간이 길어집니다.

데몬의 CPU 시간 중 또 다른 일부는 상한값이 초과될 때 이를 적용하는 데 소비됩니다. 소비되는 시간은 관련된 가상 메모리 용량에 비례합니다. 작업 부하 주소 공간의 전체 크기에서 해당 변경 사항에 따라 소비되는 CPU 시간이 늘어나거나 줄어듭니다. 이 정보는 `rcapstat` 출력의 `vm` 열에 보고됩니다. 자세한 내용은 123 페이지 “rcapstat를 사용하여 리소스 사용률 모니터링” 및 `rcapstat(1)` 매뉴얼 페이지를 참조하십시오.

공유 메모리에 대한 보고

`rcapd` 데몬은 다른 프로세스와 공유하거나 동일한 프로세스 내에서 여러 번 매핑되는 메모리 페이지의 RSS를 비교적 정확한 추정값으로 보고합니다. 다른 프로젝트의 프로세스에서 동일한 메모리를 공유하면 메모리를 공유하는 모든 프로젝트에 대한 RSS에 해당 메모리가 포함됩니다.

이 추정값은 공유 메모리를 광범위하게 사용하는 데이터베이스 등의 작업 부하에서 유용합니다. 데이터베이스 작업 부하의 경우 `prstat` 명령의 `-J` 또는 `-Z` 옵션의 출력을 사용하여 프로젝트의 정기적인 사용량을 샘플링하여 적합한 초기 상한값을 결정할 수도 있습니다. 자세한 내용은 `prstat(1M)` 매뉴얼 페이지를 참조하십시오.

rcapd 작업 간격

`rcapd`에서 수행하는 주기적 작업에 대한 간격을 조정할 수 있습니다.

모든 간격은 초 단위로 지정됩니다. rcapd 작업 및 기본 간격 값이 다음 표에 설명되어 있습니다.

작업	기본 간격 값(초)	설명
scan	15	프로젝트 작업 부하에 연결된 또는 작업 부하에서 나간 프로세스를 검색하는 간격의 초 수입니다. 최소값은 1초입니다.
sample	5	RSS(Resident Set Size)와 이후 상한값 적용을 샘플링하는 간격의 초 수입니다. 최소값은 1초입니다.
report	5	페이징 통계를 업데이트하는 간격의 초 수입니다. 0으로 설정되면 통계가 업데이트되지 않고 rcapstat의 출력이 최신이 아닙니다.
config	60	재구성하는 간격의 초 수. 재구성 이벤트에서 rcapadm은 업데이트를 위해 구성 파일을 읽고 project 데이터베이스에서 새로 설정되거나 수정된 프로젝트 상한값을 검색합니다. SIGHUP을 rcapd로 보내면 즉시 재구성됩니다.

간격을 조정하려면 127 페이지 “작업 간격을 설정하는 방법”을 참조하십시오.

rcapd 검색 간격 결정

스캔 간격은 rcapd에서 새 프로세스를 검색하는 빈도를 제어합니다. 많은 프로세스가 실행 중인 시스템의 경우 목록 전체를 검색하는 데 더 많은 시간이 걸리므로 전체 CPU 시간 소비를 줄이기 위해 간격을 늘리는 것이 좋을 수 있습니다. 그러나 검색 간격은 프로세스가 상한값이 지정된 작업 부하에 포함되기 위해 존재해야 하는 최소 시간을 나타낼 수도 있습니다. 단기간 진행되는 많은 프로세스를 실행하는 작업 부하의 경우, 검색 간격이 길어지면 rcapd에서 이러한 프로세스를 작업 부하에 포함시키지 못할 수도 있습니다.

샘플 간격 결정

rcapadm을 사용하여 구성하는 샘플 간격은 rcapd에서 작업 부하 사용량 샘플링과 상한값이 초과되는 경우 이를 적용하는 사이에 대기하는 최소 시간입니다. 이 간격을 줄이는 경우 대부분의 조건에서 rcapd가 더 자주 상한값을 적용하여 페이징으로 인한 I/O가 증가할 수 있습니다. 그러나 짧은 샘플 간격이 특정 작업 부하의 갑작스러운

물리적 메모리 증가로 인해 다른 작업 부하에 미치는 영향을 줄여 줄 수도 있습니다. 작업 부하에서 메모리를 제한 없이 사용하고 상한값이 설정된 다른 작업 부하의 메모리를 사용할 수 있는 샘플링 간의 시간 간격이 좁아집니다.

rcapstat로 지정된 샘플 간격이 rcapadm을 사용하여 rcapd로 지정된 간격보다 짧은 경우, 일부 간격의 출력이 0이 될 수 있습니다. 이러한 상황은 rcapd가 rcapadm을 사용하여 지정한 간격보다 더 자주 통계를 업데이트하지 않기 때문에 발생합니다. rcapadm을 사용하여 지정한 간격은 rcapstat에서 사용하는 샘플링 간격과 별개입니다.

rcapstat를 사용하여 리소스 사용률 모니터링

rcapstat를 사용하여 상한값이 설정된 프로젝트의 리소스 사용률을 모니터합니다. rcapstat 보고서의 예를 보려면 129 페이지 “rcapstat를 사용하여 보고서 생성”을 참조하십시오.

보고서의 샘플링 간격을 설정하고 통계가 반복되는 횟수를 지정할 수 있습니다.

interval 샘플링 간격을 초 단위로 지정합니다. 기본 간격은 5초입니다.

count 통계가 반복되는 횟수를 지정합니다. 기본적으로 rcapstat는 종료 신호가 수신되거나 rcapd 프로세스가 끝날 때까지 통계를 보고합니다.

rcapstat에서 발행한 첫 보고서의 페이징 통계에는 데몬이 시작된 이후의 작업이 표시됩니다. 이후 보고서에는 마지막 보고서가 발행된 후의 작업이 반영됩니다.

다음 표에는 rcapstat 보고서의 열 제목이 정의되어 있습니다.

rcapstat 열 제목	설명
id	상한값이 설정된 프로젝트의 프로젝트 ID입니다.
프로젝트	프로젝트 이름입니다.
nproc	프로젝트의 프로세스 수입니다.
vm	매핑된 모든 파일 및 장치를 포함하여 프로젝트의 프로세스에서 사용하는 가상 메모리 크기의 전체 용량으로 킬로바이트(K), 메가바이트(M) 또는 기가바이트(G) 단위로 표시됩니다.
rss	프로젝트 내 프로세스의 전체 RSS(Resident Set Size)의 추정 용량으로 킬로바이트(K), 메가바이트(M) 또는 기가바이트(G) 단위로 표시되며, 공유되는 페이지는 반영되지 않습니다.

rcapstat 열 제목	설명
상한값	프로젝트에 대해 정의된 RSS 상한값입니다. 메모리 상한값 지정 방법에 대한 자세한 내용은 118 페이지 "프로젝트의 물리적 메모리 사용 제한을 위한 속성" 또는 rcapd(1M) 매뉴얼 페이지를 참조하십시오.
at	마지막 rcapstat 샘플 이후에 rcapd에서 페이징 아웃하려고 시도한 메모리의 전체 용량입니다.
avgat	마지막 rcapstat 샘플 이후에 발생한 각 샘플 주기 동안 rcapd에서 페이징 아웃하려고 시도한 메모리의 평균 용량입니다. rcapadm을 사용하여 rcapd의 모음 RSS 샘플링 비율을 설정할 수 있습니다. 121 페이지 "rcapd 작업 간격"을 참조하십시오.
pg	마지막 rcapstat 샘플 이후에 rcapd에서 성공적으로 페이징 아웃한 메모리의 전체 용량입니다.
avgpg	마지막 rcapstat 샘플 이후에 발생한 각 샘플 주기 동안 rcapd에서 성공적으로 페이징 아웃한 메모리의 추정 평균 용량입니다. rcapadm을 사용하여 rcapd의 프로세스 RSS 크기 샘플링 비율을 설정할 수 있습니다. 121 페이지 "rcapd 작업 간격"을 참조하십시오.

rcapd와 함께 사용되는 명령

명령 참조	설명
rcapstat(1)	상한값이 설정된 프로젝트의 리소스 사용률을 모니터합니다.
rcapadm(1M)	리소스 상한값 지원 데몬을 구성하고, 구성된 경우 리소스 상한값 지원 데몬의 현재 상태를 표시하며, 리소스 상한값을 사용 또는 사용 안함으로 설정합니다. 임시 메모리 상한값을 설정하기 위해서도 사용됩니다.
rcapd(1M)	리소스 상한값 지원 데몬입니다.

◆ ◆ ◆ 11장

11 리소스 상한값 지원 데몬 관리(작업)

이 장은 리소스 상한값 지원 데몬 rcapd를 구성 및 사용하는 절차로 구성되어 있습니다.

rcapd에 대한 개요는 10장, "리소스 상한값 지원 데몬을 사용한 물리적 메모리 제어(개요)"를 참조하십시오.

RSS(Resident Set Size) 상한값 설정

project 데이터베이스 항목에 rcap.max-rss 속성을 추가하여 프로젝트에 대한 물리적 메모리 RSS(Resident Set Size) 상한값을 정의합니다.

▼ 프로젝트에 대한 rcap.max-rss 속성을 추가하는 방법

1. **관리자로 전환합니다.**
2. **/etc/project 파일에 이 속성을 추가합니다.**

   rcap.max-rss=*value*

예 11–1 RSS 프로젝트 상한값

/etc/project 파일의 다음 행은 db라는 프로젝트에 대해 10기가바이트의 RSS 상한값을 설정합니다.

```
db:100::db,root::rcap.max-rss=10737418240
```

시스템에서 지정된 상한값을 페이지 크기로 반올림할 수 있습니다.

▼ 프로젝트에 대한 rcap.max-rss 속성을 추가하기 위해 projmod 명령을 사용하는 방법

1 **관리자로 전환합니다.**

2 **이 경우 /etc/project 파일에서 db라는 프로젝트에 대해 10기가바이트의 rcap.max-rss 속성을 설정합니다.**

```
# projmod -a -K rcap.max-rss=10GB db
```

그러면 /etc/project 파일에 다음 행이 포함됩니다.

```
db:100::db,root::rcap.max-rss=10737418240
```

리소스 상한값 지원 데몬 구성 및 사용(작업 맵)

작업	설명	수행 방법
메모리 상한값 적용 임계치를 설정합니다.	프로세스에 사용할 수 있는 물리적 메모리가 적을 때 적용할 상한값을 구성합니다.	127 페이지 "메모리 상한값 적용 임계치를 설정하는 방법"
작업 간격을 설정합니다.	간격은 리소스 상한값 지원 데몬에 의해 수행되는 주기적 작업에 적용됩니다.	127 페이지 "작업 간격을 설정하는 방법"
리소스 상한값 사용으로 설정합니다.	해당 시스템에서 리소스 상한값을 활성화합니다.	128 페이지 "리소스 상한값 사용으로 설정하는 방법"
리소스 상한값 사용 안함으로 설정합니다.	해당 시스템에서 리소스 상한값을 비활성화합니다.	128 페이지 "리소스 상한값을 사용 안함으로 설정하는 방법"
상한값 및 프로젝트 정보를 보고합니다.	보고서 생성을 위한 명령의 예를 보여 줍니다.	129 페이지 "상한값 및 프로젝트 정보 보고"
프로젝트의 RSS(Resident Set Size)를 모니터합니다.	프로젝트의 RSS(Resident Set Size)에 대한 보고서를 생성합니다.	130 페이지 "프로젝트의 RSS 모니터링"
프로젝트의 작업 집합 크기를 결정합니다.	프로젝트의 작업 집합 크기에 대한 보고서를 생성합니다.	131 페이지 "프로젝트의 작업 집합 크기 결정"
메모리 사용률 및 메모리 상한값에 대해 보고합니다.	보고서 끝에 각 간격에 대한 메모리 사용률 및 상한값 적용 행을 인쇄합니다.	132 페이지 "메모리 사용률 및 메모리 상한값 적용 임계치 보고"

rcapadm을 사용하여 리소스 상한값 지원 데몬 관리

이 절은 rcapadm을 사용하여 리소스 상한값 지원 데몬을 구성하는 절차로 구성되어 있습니다. 자세한 내용은 119 페이지 “rcapd 구성” 및 rcapadm(1M) 매뉴얼 페이지를 참조하십시오. rcapadm을 사용하여 영역에 대한 임시 리소스 상한값을 지정하는 내용도 다룹니다.

인수 없이 사용되는 경우 rcapadm 명령은 리소스 상한값 데몬이 구성된 경우 현재 상태를 표시합니다.

▼ 메모리 상한값 적용 임계치를 설정하는 방법

프로세스에 사용할 수 있는 물리적 메모리가 적어지기 전에는 상한값이 적용되지 않도록 구성할 수 있습니다. 자세한 내용은 120 페이지 “메모리 상한값 적용 임계치”를 참조하십시오.

최소값(및 기본값)은 0이며, 이것은 메모리 상한값이 항상 적용된다는 것을 의미합니다. 다른 최소값을 설정하려면 다음 절차를 따릅니다.

1. **관리자로 전환합니다.**
2. **rcapadm의 -c 옵션을 사용하여 메모리 상한값 적용을 위한 다른 물리적 메모리 사용률 값을 설정합니다.**

   # rcapadm -c *percent*

   *percent*는 0부터 100사이의 범위에 있습니다. 값이 높을수록 덜 제한적입니다. 높은 값은 시스템의 메모리 사용률이 이 임계치를 초과하기 전까지는 상한값이 설정된 프로젝트 작업 부하를 상한값 적용 없이 실행할 수 있다는 것을 의미합니다.

참조 현재의 물리적 메모리 사용률 및 상한값 적용 임계치를 표시하려면 132 페이지 “메모리 사용률 및 메모리 상한값 적용 임계치 보고”를 참조하십시오.

▼ 작업 간격을 설정하는 방법

121 페이지 “rcapd 작업 간격”에는 rcapd에서 수행하는 주기적 작업의 간격에 대한 정보가 포함됩니다. rcapadm을 사용하여 작업 간격을 설정하려면 다음 절차를 따릅니다.

1. **관리자로 전환합니다.**
2. **-i 옵션을 사용하여 간격 값을 설정합니다.**

   # rcapadm -i *interval=value,...,interval=value*

주 – 모든 간격 값은 초 단위로 지정됩니다.

▼ 리소스 상한값 사용으로 설정하는 방법

시스템에서 리소스 상한값을 사용하도록 설정하는 데는 세 가지 방법이 있습니다. 리소스 상한값 사용으로 설정하면 기본값으로 `/etc/rcap.conf` 파일도 설정됩니다.

1. **관리자로 전환합니다.**
2. **다음 중 한 가지 방법으로 리소스 상한값 지원 데몬 사용으로 설정합니다.**
   - `svcadm` 명령을 사용하여 리소스 상한값을 설정합니다.

     ```
     # svcadm enable rcap
     ```
   - 리소스 상한값 지원 데몬을 사용으로 설정하여 지금 그리고 시스템이 부트될 때마다 시작되도록 합니다.

     ```
     # rcapadm -E
     ```
   - `-n` 옵션을 지정하여 지금 시작하지 않고 부트 시 리소스 상한값 지원 데몬 사용으로 설정할 수도 있습니다.

     ```
     # rcapadm -n -E
     ```

▼ 리소스 상한값을 사용 안함으로 설정하는 방법

시스템에서 리소스 상한값을 사용 안함으로 설정하는 데는 세 가지 방법이 있습니다.

1. **관리자로 전환합니다.**
2. **다음 중 한 가지 방법으로 리소스 상한값 지원 데몬 사용 안함으로 설정합니다.**
   - **`svcadm` 명령을 사용하여 리소스 상한값을 해제합니다.**

     ```
     # svcadm disable rcap
     ```
   - **지금 중지하고 시스템이 부트될 때 시작되지 않도록 리소스 상한값 지원 데몬을 사용 안함으로 설정하려면 다음을 입력합니다.**

     ```
     # rcapadm -D
     ```
   - **리소스 상한값 지원 데몬을 중지하지 않고 사용 안함으로 설정하려면 `-n` 옵션도 지정합니다.**

     ```
     # rcapadm -n -D
     ```

참고 – 리소스 상한값 지원 데몬 안전하게 사용 안함으로 설정

rcapd를 안전하게 사용 안함으로 설정하려면 rcapadm -D를 사용합니다. 데몬이 강제 종료되는 경우(kill(1) 매뉴얼 페이지 참조), 프로세스가 중지된 상태로 남게 되어 수동으로 다시 시작해야 할 수 있습니다. 프로세스를 계속하려면 prun 명령을 사용합니다. 자세한 내용은 prun(1) 매뉴얼 페이지를 참조하십시오.

▼ 영역에 대한 임시 리소스 상한값을 지정하는 방법

이 절차는 지정된 영역에서 사용할 수 있는 메모리의 최대 용량을 할당하기 위해 사용됩니다. 이 값은 다음 재부트 전까지만 사용됩니다. 지속 상한값을 설정하려면 zonecfg 명령을 사용합니다.

1. **관리자로 전환합니다.**

2. **my-zone 영역에 대해 512메가바이트의 최대 메모리 값을 설정합니다.**

```
# rcapadm -z testzone -m 512M
```

rcapstat를 사용하여 보고서 생성

rcapstat를 사용하여 리소스 상한값 통계를 보고합니다. 123 페이지 “rcapstat를 사용하여 리소스 사용률 모니터링”에서 rcapstat 명령을 사용하여 보고서를 생성하는 방법에 대해 설명합니다. 이 절에서는 보고서의 열 제목에 대해서도 설명합니다. rcapstat(1) 매뉴얼 페이지에도 이 정보가 포함되어 있습니다.

다음 세부절에서는 예제를 사용하여 특정 목적의 보고서를 생성하는 방법에 대해 설명합니다.

상한값 및 프로젝트 정보 보고

이 예에서는 두 사용자와 연관된 두 프로젝트에 대해 상한값이 정의되어 있습니다. user1은 상한값이 50메가바이트이고, user2는 상한값이 10메가바이트입니다.

다음 명령은 5초 샘플링 간격으로 5개의 보고서를 생성합니다.

```
user1machine% rcapstat 5 5
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M   50M    0K 3312K    0K
 78194   user2      1  2368K  1856K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1856K   10M    0K    0K    0K    0K
```

```
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
```

출력의 처음 세 개 행이 첫번째 보고서를 구성하며, 여기에는 두 프로젝트에 대한 상한값과 프로젝트 정보 및 rcapd가 시작된 이후의 페이징 통계가 포함됩니다. at 및 pg 열은 user1의 경우 0보다 큰 수이고 user2의 경우 0입니다. 이것은 데몬의 내역 중 일정 시점에 user1은 해당 상한값은 초과했지만 user2는 초과하지 않았음을 나타냅니다.

이후 보고서에는 중요한 작업이 표시되지 않습니다.

프로젝트의 RSS 모니터링

다음 예는 해당 RSS 상한값을 초과하는 RSS가 있는 user1 프로젝트 를 사용합니다.

다음 명령은 5초 샘플링 간격으로 5개의 보고서를 생성합니다.

```
user1machine% rcapstat 5 5

    id project  nproc    vm   rss   cap    at avgat     pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M  220M  5528K  2764K
376565   user1      3 6249M 6144M 6144M    0M  131M  4912K  1637K
376565   user1      3 6249M 6171M 6144M   27M  147M  6048K  2016K
376565   user1      3 6249M 6146M 6144M 4872M  174M  4368K  1456K
376565   user1      3 6249M 6156M 6144M   12M  161M  3376K  1125K
```

user1 프로젝트에는 물리적 메모리를 활발히 사용하는 세 개의 프로세스가 있습니다. pg 열의 양수 값은 rcapd가 프로젝트 프로세스의 물리적 메모리 사용률을 낮춰 상한값을 맞추기 위해 지속적으로 메모리를 페이징 아웃하고 있다는 것을 나타냅니다. 그러나 rcapd는 RSS를 상한값 아래로 유지하는 데 성공하지 못합니다. 이것은 이러한 감소를 보이지 않는 다양한 rss 값으로 표시됩니다. 메모리가 페이징 아웃되는 즉시 작업 부하는 이를 다시 사용하고 RSS 카운트가 백업됩니다. 이것은 프로젝트의 모든 상주 메모리가 활발히 사용되고 있으며 작업 집합 크기(*WSS*)가 상한값보다 크다는 것을 의미합니다. 따라서 rcapd가 상한값을 맞추기 위해 작업 집합의 일부를 페이징 아웃해야 합니다. 이러한 상황에서는 다음 중 하나가 발생하기 전까지는 시스템에서 페이지 폴트 비율이 계속 높고 연관된 I/O가 많습니다.

- WSS가 작아집니다.
- 상한값이 증가됩니다.
- 응용 프로그램의 메모리 액세스 패턴이 변경됩니다.

이러한 상황에서 샘플 간격을 줄이면 rcapd에서 좀 더 자주 작업 부하를 샘플링하고 상한값을 적용하게 되어 RSS 값과 상한값 간의 차이가 줄어들 수 있습니다.

주 – 새 페이지를 만들어야 하거나 시스템이 스왑 장치에서 페이지를 복사해 와야 하는 경우 페이지 폴트가 발생합니다.

프로젝트의 작업 집합 크기 결정

다음 예는 이전 예의 연속으로 샘플 프로젝트를 사용합니다.

이전 예에서는 user1 프로젝트가 해당 상한값에서 허용하는 것보다 많은 물리적 메모리를 사용하고 있다는 것을 보여 주었습니다. 이 예에서는 프로젝트 작업 부하에 필요한 메모리 용량을 보여 줍니다.

```
user1machine% rcapstat 5 5
    id project  nproc    vm   rss   cap    at avgat    pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M    0K   689M     0K
376565   user1      3 6249M 6144M 6144M    0K    0K     0K     0K
376565   user1      3 6249M 6171M 6144M   27M    0K    27M     0K
376565   user1      3 6249M 6146M 6144M 4872K    0K  4816K     0K
376565   user1      3 6249M 6156M 6144M   12M    0K    12M     0K
376565   user1      3 6249M 6150M 6144M 5848K    0K  5816K     0K
376565   user1      3 6249M 6155M 6144M   11M    0K    11M     0K
376565   user1      3 6249M 6150M   10G   32K    0K    32K     0K
376565   user1      3 6249M 6214M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
```

주기의 중간에 user1 프로젝트의 상한값이 6기가바이트에서 10기가바이트로 증가되었습니다. 이러한 증가로 상한값 적용이 중지되고 RSS(Resident Set Size)가 커질 수 있어 크기가 다른 프로세스나 시스템의 메모리 용량에 따라서만 제한됩니다. rss 열은 프로젝트 WSS(작업 세트 크기)(이 예에는 6247M)를 반영하여 안정될 수 있습니다. 이것은 프로젝트의 프로세스에서 지속적으로 페이지 폴트를 발생시키지 않고 작업할 수 있는 최소 상한값입니다.

user1의 상한값이 6기가바이트이지만 5초 샘플 간격마다 rcapd가 작업 부하 메모리의 일부를 페이징 아웃하므로 RSS가 감소하고 I/O가 증가합니다. 페이징 아웃이 완료된 직후 해당 페이지가 필요한 작업 부하에서 다시 페이지를 들여와 실행을 계속합니다. 이러한 주기는 상한값이 10기가바이트로 증가될 때까지, 대략 이 예의 약 중간까지 반복됩니다. 그러면 RSS가 6.1기가바이트에서 안정됩니다. 이제 작업 부하의 RSS가 상한값 아래이므로 더 이상 페이징이 발생하지 않습니다. 페이징과 연관된 I/O도 역시 중지됩니다. 따라서 관찰되는 시점에 프로젝트에서 진행 중인 작업을 수행하려면 6.1기가바이트가 필요합니다.

vmstat(1M) 및 iostat(1M) 매뉴얼 페이지도 참조하십시오.

메모리 사용률 및 메모리 상한값 적용 임계치 보고

rcapstat의 -g 옵션을 사용하여 다음을 보고할 수 있습니다.

- 시스템에 설치된 물리적 메모리에 대한 백분율로 표시한 현재의 물리적 메모리 사용률
- rcapadm에 의해 설정된 시스템 메모리 상한값 적용 임계치

-g 옵션을 사용하면 보고서 끝에 각 간격에 대한 메모리 사용률 및 상한값 적용 행이 인쇄됩니다.

```
# rcapstat -g
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565    rcap       0    0K    0K   10G    0K    0K   0K     0K
physical memory utilization: 55%   cap enforcement threshold: 0%
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565    rcap       0    0K    0K   10G    0K    0K   0K     0K
physical memory utilization: 55%   cap enforcement threshold: 0%
```

◆ ◆ ◆ 12 장

12 리소스 풀(개요)

이 장에서는 다음 기술에 대해 설명합니다.

- 시스템 리소스를 분할하는 데 사용하는 리소스 풀
- 설정된 시스템 목표를 충족하도록 각 리소스 풀의 리소스 할당을 동적으로 조정하는 DRP(동적 리소스 풀)

리소스 풀 및 동적 리소스 풀은 Oracle Solaris SMF(서비스 관리 기능)의 서비스입니다. 이러한 서비스는 각각 별도로 사용됩니다.

이 장에서는 다음 항목을 다룹니다.

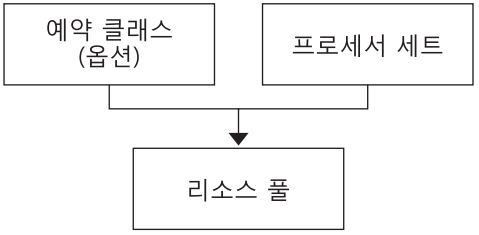
이 기능을 사용하는 절차에 대해서는 13 장, "리소스 풀 만들기 및 관리(작업)"를 참조하십시오.

리소스 풀 소개

리소스 풀을 사용하면 특정 리소스의 작업 부하 사용이 겹치지 않도록 작업 부하를 분리할 수 있습니다. 이 리소스 예약은 작업 부하가 혼합된 시스템의 성능을 예측할 수 있게 도와줍니다.

리소스 풀은 프로세서 세트(pset) 구성 및 선택적으로 예약 클래스 할당에 대한 일관된 구성 방식을 제공합니다.

그림 12–1 리소스 풀 프레임워크



풀은 시스템에서 사용할 수 있는 여러 리소스 집합의 특정 바인딩으로 생각할 수 있습니다. 다른 종류의 사용 가능한 리소스 조합을 나타내는 풀을 만들 수 있습니다.

```
pool1: pset_default
pool2: pset1
pool3: pset1, pool.scheduler="FSS"
```

풀은 여러 개의 분할 영역을 그룹화하여 레이블이 지정된 작업 부하와 연결할 핸들을 제공합니다. `/etc/project` 파일의 각 프로젝트 항목은 해당 항목과 연관된 단일 풀을 가질 수 있습니다. 이러한 풀은 `project.pool` 속성을 사용하여 지정합니다.

풀이 사용되면 **기본 풀**과 **기본 프로세서 세트**가 기본 구성을 형성합니다. 추가로 사용자 정의 풀과 프로세서 세트를 만들어 구성에 추가할 수 있습니다. CPU는 프로세서 세트 하나에만 속할 수 있습니다. 사용자 정의 풀과 프로세서 세트를 삭제할 수 있습니다. 기본 풀과 기본 프로세서 세트는 삭제할 수 없습니다.

기본 풀에는 `pool.default` 등록 정보가 `true`로 설정되어 있습니다. 기본 프로세서 세트에는 `pset.default` 등록 정보가 `true`로 설정되어 있습니다. 따라서 기본 풀과 기본 프로세서 세트 모두 이름을 변경해도 식별할 수 있습니다.

사용자 정의 풀 방식은 주로 CPU가 4개가 넘는 대형 시스템에서 사용하기 위한 것입니다. 하지만 소형 시스템에서도 이 기능을 이용할 수 있습니다. 소형 시스템에서 중요하지 않은 리소스 분할 영역을 공유하는 풀을 만들 수 있습니다. 풀은 중요한 리소스를 기준으로 해서만 구분됩니다.

동적 리소스 풀 소개

동적 리소스 풀은 시스템 이벤트 및 응용 프로그램 로드 변경에 대한 응답으로 각 풀의 리소스 할당을 동적으로 조정하기 위한 방식을 제공합니다. DRP는 관리자가 해야 하는 여러 결정을 단순화하고 그 수를 줄여 줍니다. 조정은 관리자가 지정한 시스템 성능 목표를 유지하도록 자동으로 수행됩니다. 구성 변경 사항은 기록됩니다. 이러한 기능은 주로 리소스 제어기 `poold`를 통해 실행되며, 이는 동적 리소스 할당이 필요할 때 항상 활성화되어야 하는 시스템 데몬입니다. `poold`는 주기적으로 시스템의 부하를 검사하고 시스템이 리소스 사용과 관련하여 최적의 성능을 유지하도록 하기 위해 간섭이 필요한지를 확인합니다. `poold` 구성은 `libpool` 구성에 보관됩니다. `poold`에 대한 자세한 내용은 `poold`(1M) 매뉴얼 페이지를 참조하십시오.

리소스 풀 및 동적 리소스 풀 사용 및 사용 안함 정보

리소스 풀과 동적 리소스 풀을 사용하거나 사용하지 않으려면 157 페이지 “풀 기능을 사용 또는 사용 안함으로 설정”을 참조하십시오.

영역에서 사용되는 리소스 풀

영역을 시스템에 구성된 리소스 풀과 연결하지 않고 `zonecfg` 명령을 사용하여 영역이 실행되는 동안 유효한 임시 풀을 만들 수 있습니다. 자세한 내용은 208 페이지 “`dedicated-cpu` 리소스”를 참조하십시오.

영역이 사용되는 시스템에서는 비전역 영역을 리소스 풀 하나와 연결할 수 있지만 이 풀을 특정 영역에 지정할 필요는 없습니다. 또한, 전역 영역에서 `poolbind` 명령을 사용하여 비전역 영역의 개별 프로세스를 다른 풀에 바인드할 수 없습니다. 비전역 영역을 풀과 연결하려면 242 페이지 “영역 구성, 확인 및 커밋”을 참조하십시오.

풀에 대한 예약 클래스를 설정하고 비전역 영역을 이 풀과 연결하는 경우 이 영역에서는 기본적으로 이 예약 클래스를 사용합니다.

동적 리소스 풀을 사용하는 경우 `poold`의 실행 인스턴스 범위가 전역 영역으로 제한됩니다.

비전역 영역에서 `poolstat` 유틸리티를 실행하면 이 영역과 연결된 풀에 대한 정보만 표시됩니다. 비전역 영역에서 인수 없이 `pooladm` 명령을 실행하면 이 영역과 연결된 풀에 대한 정보만 표시됩니다.

리소스 풀 명령에 대한 자세한 내용은 153 페이지 “리소스 풀 기능에 사용되는 명령”을 참조하십시오.

풀 사용 시기

리소스 풀은 여러 관리 시나리오에 적용할 수 있는 다양한 방식을 제공합니다.

일괄 처리 연산 서버	서버를 두 개 풀로 분리할 때 풀 기능을 사용합니다. 풀 하나는 시간 공유 사용자의 로그인 세션과 대화식 작업에 사용합니다. 다른 풀은 일괄 처리 시스템을 통해 제출되는 작업에 사용합니다.
응용 프로그램 또는 데이터베이스 서버	응용 프로그램의 요구 사항에 따라 대화식 응용 프로그램에 대한 리소스를 분할합니다.
단계별 응용 프로그램 설정	사용자의 기대 수준을 설정합니다. 처음에는 시스템에서 전달할 서비스의 일부만 실행하는 시스템을 배치할 수 있습니다. 예약 기반 리소스 관리 방식이 설정되지 않은 상태에서 시스템이 온라인 상태가 되면 문제가 발생할 수 있습니다. 예를 들어, FSS(Fair Share Scheduler)는 CPU 사용률을 최적화합니다. 하나의 응용 프로그램만 실행 중인 시스템의 응답 시간이 지나치게 빠를 수 있습니다. 여러 개의 응용 프로그램이 로드된 경우 이러한 응답 시간을 기대할 수 없습니다. 각 응용 프로그램에 별도의 풀을 사용하여 모든 응용 프로그램을 배치하기 전에 각 응용 프로그램에 사용할 수 있는 CPU 수에 최대값을 지정할 수 있습니다.
복잡한 시간 공유 서버	많은 사용자를 지원하는 서버를 분할합니다. 서버 분할은 사용자별 응답을 예측하기 쉽게 하는 분리 방식을 제공합니다. 사용자를 별도 풀에 바인드되는 그룹으로 나누고 FSS(Fair Share Scheduler) 기능을 사용하여 우선 순위가 있는 사용자 집합에 유리하게 CPU 할당을 조정할 수 있습니다. 이 할당은 사용자 역할, 계정 차지백 등을 기준으로 할 수 있습니다.
계절적으로 바뀌는 작업 부하	변화하는 수요에 맞게 조정하려면 리소스 풀을 사용합니다.

	사이트에 대해 월별, 분기별 또는 연간 주기 등 장기간 동안의 작업 부하 수요의 변화를 예측할 수 있습니다. 사이트에 대한 이러한 변화를 예측하면 cron 작업에서 pooladm을 호출하여 여러 풀 구성 간에 대체할 수 있습니다. 137 페이지 "리소스 풀 프레임워크"를 참조하십시오.
실시간 응용 프로그램	RT 스케줄러와 지정된 프로세서 리소스를 사용하여 실시간 풀을 만듭니다.
시스템 사용률	설정한 시스템 목표를 적용합니다. 자동 풀 데몬 기능을 사용하여 사용 가능한 리소스를 식별한 다음 작업 부하를 모니터하여 지정된 목표가 더 이상 충족되지 않는 시점을 감지합니다. 가능한 경우 데몬이 수정 작업을 수행하지만 그렇지 않은 경우 조건이 기록될 수 있습니다.

리소스 풀 프레임워크

/etc/pooladm.conf 구성 파일에서 정적 풀 구성이 설명됩니다. 정적 구성은 관리자가 리소스 풀 기능과 관련하여 시스템을 구성하려는 방식을 나타냅니다. 다른 파일 이름을 지정할 수 있습니다.

SMF(서비스 관리 기능) 또는 pooladm - e 명령을 사용하여 리소스 풀 프레임워크를 사용하려는 경우 /etc/pooladm.conf 파일이 있으면 이 파일에 포함된 구성이 시스템에 적용됩니다.

커널은 리소스 풀 프레임워크 내에 있는 리소스의 배치에 대한 정보를 보유합니다. 이것을 동적 구성이라고 하며, 한 시점에서 특정 시스템에 대한 리소스 풀 기능을 나타냅니다. 동적 구성은 pooladm 명령을 사용하여 볼 수 있습니다. 풀과 리소스 집합에 대한 등록 정보가 표시되는 순서는 다를 수 있습니다. 동적 구성을 수정하는 방법은 다음과 같습니다.

- 간접적 - 정적 구성 파일 적용
- 직접적 - poolcfg 명령을 -d 옵션과 함께 사용

다른 시간에 활성화할 정적 풀 구성 파일이 여러 개 있을 수 있습니다. cron 작업에서 pooladm을 호출하여 여러 풀 구성 간에 대체할 수 있습니다. cron 유틸리티에 대한 자세한 내용은 cron(1M) 매뉴얼 페이지를 참조하십시오.

기본적으로 리소스 풀 프레임워크는 활성화되어 있지 않습니다. 동적 구성을 만들거나 수정하려면 리소스 풀을 사용해야 합니다. 정적 구성 파일은 리소스 풀 프레임워크가

사용 안함으로 설정되어 있어도 poolcfg 또는 libpool 명령을 사용하여 조작할 수 있습니다. 풀 기능이 활성화되어 있지 않으면 정적 구성 파일을 만들 수 없습니다. 구성 파일에 대한 자세한 내용은 140 페이지 "풀 구성 만들기"를 참조하십시오.

리소스 풀과 poold 시스템 데몬에 사용되는 명령은 다음 매뉴얼 페이지에 설명되어 있습니다.

- pooladm(1M)
- poolbind(1M)
- poolcfg(1M)
- poold(1M)
- poolstat(1M)
- libpool(3LIB)

/etc/pooladm.conf 내용

동적 구성을 포함하여 모든 리소스 풀 구성에 포함될 수 있는 요소는 다음과 같습니다.

system 시스템의 모든 동작에 영향을 주는 등록 정보

풀 리소스 풀 정의

pset 프로세서 세트 정의

cpu 프로세서 정의

이러한 요소 모두에는 조작하여 리소스 풀 프레임워크의 상태와 동작을 변경할 수 있는 등록 정보가 있습니다. 예를 들어, 풀 등록 정보 pool.importance는 제공된 풀의 상대적 중요도를 나타냅니다. 이 등록 정보는 가능한 리소스 분쟁 해결에 사용됩니다. 자세한 내용은 libpool(3LIB)를 참조하십시오.

풀 등록 정보

풀 기능은 풀, 리소스 또는 구성 요소에 배치할 수 있는 이름이 지정되고 형식이 지정된 등록 정보를 지원합니다. 관리자는 여러 풀 요소에 추가 등록 정보를 저장할 수 있습니다. 프로젝트 속성과 비슷한 등록 정보 이름 공간이 사용됩니다.

예를 들어, 다음 주석은 제공된 pset가 특정 Datatree 데이터베이스와 연결되어 있음을 나타냅니다.

```
Datatree,pset.dbname=warehouse
```

등록 정보 유형에 대한 자세한 내용은 146 페이지 "poold 등록 정보"를 참조하십시오.

주 – 여러 특수 등록 정보는 내부용으로 예약되어 있어 설정하거나 제거할 수 없습니다. 자세한 내용은 `libpool(3LIB)` 매뉴얼 페이지를 참조하십시오.

시스템에 풀 구현

사용자 정의 풀은 다음 방법 중 하나를 사용하여 시스템에 구현할 수 있습니다.

- Oracle Solaris 소프트웨어가 부트되면 `init` 스크립트는 `/etc/pooladm.conf` 파일이 있는지 확인합니다. 이 파일이 있고 풀이 사용하도록 설정되어 있으면 `pooladm`을 호출하여 이 구성을 활성 풀 구성으로 지정합니다. 시스템은 `/etc/pooladm.conf`에서 요청된 구성을 반영하기 위해 동적 구성을 만들고 이에 따라 시스템 리소스가 분할됩니다.
- Oracle Solaris 시스템이 실행 중일 때 풀 구성이 없으면 풀 구성을 활성화하거나 `pooladm` 명령을 사용하여 수정할 수 있습니다. 기본적으로 `pooladm` 명령은 `/etc/pooladm.conf`에서 수행됩니다. 하지만 선택적으로 대체 위치와 파일 이름을 지정하고 이 파일을 사용하여 풀 구성을 업데이트할 수 있습니다.

리소스 풀 사용 및 사용 안함에 대한 자세한 내용은 157 페이지 “풀 기능을 사용 또는 사용 안함으로 설정”을 참조하십시오. 사용자 정의 풀이나 리소스를 사용 중이면 풀 기능을 사용 안함으로 설정할 수 없습니다.

리소스 풀을 구성하려면 루트 권한이나 필수 권한 프로파일이 있어야 합니다.

`poold` 리소스 제어기는 동적 리소스 풀 기능과 함께 시작됩니다.

project.pool 속성

`project.pool` 속성을 `/etc/project` 파일의 프로젝트 항목에 추가하여 단일 풀과 해당 항목을 연결할 수 있습니다. 프로젝트에서 시작된 새 작업은 해당 풀에 바인드됩니다. 자세한 내용은 2 장, “프로젝트 및 작업(개요)”을 참조하십시오.

예를 들어, `projmod` 명령을 사용하여 `/etc/project` 파일의 *sales* 프로젝트에 대한 `project.pool` 속성을 설정할 수 있습니다.

```
# projmod -a -K project.pool=mypool sales
```

SPARC: 동적 재구성 작업 및 리소스 풀

DR(동적 재구성)을 사용하면 시스템이 실행되는 동안 하드웨어를 재구성할 수 있습니다. DR 작업은 지정된 유형의 리소스를 늘리거나 줄이거나 영향을 주지 않을 수 있습니다. DR은 사용 가능한 리소스의 양에 영향을 줄 수 있으므로 이러한 작업에 풀 기능이 포함되어야 합니다. DR 작업을 시작하면 풀 프레임워크가 작동하여 구성을 검증합니다.

현재 풀 구성의 무효화를 유발하지 않고 DR 작업을 진행할 수 있을 경우 전용 구성 파일이 업데이트됩니다. 무효화된 구성은 사용 가능한 리소스가 지원할 수 없는 구성입니다.

DR 작업으로 인해 풀 구성이 무효화될 경우 작업이 실패하고 메시지 로그에 대한 메시지가 표시됩니다. 강제로 구성을 완료하려면 DR 강제 옵션을 사용해야 합니다. 그런 다음 새 리소스 구성을 준수하도록 풀 구성을 수정합니다. DR 프로세스 및 강제 옵션에 대한 자세한 내용은 Sun 하드웨어에 대한 동적 재구성 사용자 설명서를 참조하십시오.

동적 리소스 풀을 사용하는 경우 데몬이 활성 상태인 동안 분할 영역이 `poold` 제어를 벗어날 수 있습니다. 자세한 내용은 150 페이지 “리소스 부족 식별”을 참조하십시오.

풀 구성 만들기

구성 파일에는 시스템에서 생성될 풀에 대한 설명이 들어 있습니다. 이 파일은 조작할 수 있는 요소를 설명합니다.

- 시스템
- pool
- pset
- cpu

조작할 수 있는 요소에 대한 자세한 내용은 `poolcfg(1M)`을 참조하십시오.

풀을 사용으로 설정하면 구조화된 `/etc/pooladm.conf` 파일을 두 가지 방법으로 만들 수 있습니다.

- `pooladm` 명령을 `-s` 옵션과 함께 사용하여 현재 시스템에서 리소스를 찾고 결과를 구성 파일에 배치할 수 있습니다.

  이 방법을 사용하는 것이 좋습니다. 시스템에서 풀 기능으로 조작할 수 있는 모든 활성 리소스와 구성 요소가 기록됩니다. 이 리소스에 기존 프로세서 세트 구성이 포함됩니다. 그러면 필요한 경우 이 구성을 수정하여 프로세서 세트 이름을 바꾸거나 추가 풀을 만들 수 있습니다.
- `-c` 옵션 및 `discover` 또는 `create system` *name* 하위 명령과 함께 `poolcfg` 명령을 사용하여 새 풀 구성을 만들 수 있습니다.

이러한 옵션은 이전 릴리스와의 호환성을 위해 유지됩니다.

/etc/pooladm.conf 파일을 수정하려면 poolcfg 또는 libpool을 사용합니다. 이 파일을 직접 편집하지 마십시오.

직접 동적 구성 조작

poolcfg 명령을 -d 옵션과 함께 사용하여 동적 구성에서 CPU 리소스 유형을 직접 조작할 수 있습니다. 리소스 전송에 두 가지 방법이 사용됩니다.

- 세트 간에 확인된 사용 가능한 리소스를 전송하도록 일반적인 요청을 할 수 있습니다.
- 특정 ID가 있는 리소스를 대상 세트로 전송할 수 있습니다. 리소스 구성이 변경되거나 시스템을 재부트하면 리소스와 연결된 시스템 ID가 변경될 수 있습니다.

예에 대해서는 169 페이지 “리소스 전송”을 참조하십시오.

DRP를 사용 중이면 리소스 전송이 poold에서 작업을 트리거할 수 있습니다. 자세한 내용은 141 페이지 “poold 개요”를 참조하십시오.

poold 개요

풀 리소스 제어기 poold는 시스템 대상 및 측정 가능한 통계를 사용하여 사용자가 지정한 시스템 성능 목표를 유지합니다. 동적 리소스 할당이 필요한 경우 이 시스템 데몬은 항상 활성 상태여야 합니다.

poold 리소스 제어기는 사용 가능한 리소스를 식별한 다음 작업 부하를 모니터하여 시스템 사용 목표가 더 이상 충족되지 않는 시점을 확인합니다. 그러면 poold가 목표 측면에서 대체 구성을 고려하여 치료 작업을 합니다. 가능한 경우 목표를 충족할 수 있도록 리소스가 재구성됩니다. 이 작업이 불가능한 경우 데몬은 사용자가 지정한 목표를 더 이상 달성할 수 없다고 기록합니다. 재구성이 성공하면 데몬은 작업 부하 목표를 계속 모니터합니다.

poold는 검사할 수 있는 결정 내역을 유지합니다. 결정 내역은 지금까지 개선을 보이지 않은 재구성을 제거하기 위해 사용됩니다.

작업 부하 목표가 변경된 경우 또는 시스템에서 사용할 수 있는 리소스가 수정된 경우 재구성이 비동기적으로 트리거될 수도 있습니다.

동적 리소스 풀 관리

DRP 서비스는 서비스 식별자 svc:/system/pools/dynamic에 있는 SMF(서비스 관리 기능)에서 관리합니다.

사용으로 설정, 사용 안함으로 설정 또는 다시 시작 요청과 같은 이 서비스 관리 작업은 svcadm 명령을 사용하여 수행할 수 있습니다. 서비스 상태는 svcs 명령을 사용하여 쿼리할 수 있습니다. 자세한 내용은 svcs(1) 및 svcadm(1M) 매뉴얼 페이지를 참조하십시오.

SMF 인터페이스는 DRP를 제어하는 권장 방법이지만 역호환성을 위해 다음 방법을 사용할 수도 있습니다.

- 동적 리소스 할당이 필요하지 않은 경우에는 SIGQUIT 또는 SIGTERM 신호로 인해 poold가 중지될 수 있습니다. 이 중 한 신호로 인해 poold가 정상적으로 종료될 수 있습니다.
- poold가 리소스가나 풀 구성에서 자동으로 변경을 감지하지만 SIGHUP 신호를 사용하여 재구성이 발생하도록 강제할 수도 있습니다.

구성 제약 조건 및 목표

구성을 변경할 때 poold는 사용자가 제공한 지시에 따라 작업을 합니다. 이러한 지시를 일련의 제약 조건 및 목표로 지정합니다. poold는 사용자의 지시 사항을 사용하여 기존 구성을 기준으로 다른 구성 가능성에 대한 관련 값을 결정합니다. 그런 다음 poold는 현재 구성의 리소스 할당을 변경하여 새 후보 구성을 생성합니다.

구성 제약 조건

제약 조건은 구성에 적용될 수 있는 잠재적 변경 사항의 일부를 제거하여 가능한 구성 범위에 영향을 줍니다. libpool 구성에 지정된 다음 제약 조건을 사용할 수 있습니다.

- 최소 및 최대 CPU 할당
- 세트에서 이동할 수 없는 고정된 구성 요소
- 풀의 중요도 인자

풀 등록 정보에 대한 자세한 내용은 libpool(3LIB) 매뉴얼 페이지 및 138 페이지 “풀 등록 정보”를 참조하십시오.

사용 지침에 대해서는 166 페이지 “구성 제약 조건을 설정하는 방법”을 참조하십시오.

pset.min 등록 정보 및 pset.max 등록 정보 제약 조건

이 두 가지 등록 정보는 프로세서 세트에 할당할 수 있는 프로세서 수에 대한 제한, 즉 최소값과 최대값을 지정합니다. 이러한 등록 정보에 대한 자세한 내용은 표 12–1을 참조하십시오.

이러한 제약 조건 내에서 리소스 분할 영역의 리소스를 같은 Oracle Solaris 인스턴스에 있는 다른 리소스 분할 영역에 할당할 수 있습니다. 리소스에 대한 액세스 권한을 얻으려면 리소스 집합와 연결된 풀에 바인드합니다. 바인딩은 로그인 시 수행되거나 `PRIV_SYS_RES_CONFIG` 권한이 있는 관리자가 수동으로 수행합니다.

cpu.pinned 등록 정보 제약 조건

`cpu-pinned` 등록 정보는 DRP가 있는 프로세서 세트에서 DRP를 사용하여 특정 CPU를 이동하지 않아야 한다는 것을 나타냅니다. 프로세서 세트에서 실행 중인 특정 응용 프로그램에 대한 캐시 사용률을 최대화하도록 이 `libpool` 등록 정보를 설정할 수 있습니다.

이 등록 정보에 대한 자세한 내용은 표 12–1을 참조하십시오.

pool.importance 등록 정보 제약 조건

`pool.importance` 등록 정보는 관리자가 정의한 대로 풀의 상대적 중요도를 설명합니다.

구성 목표

목표는 제약 조건과 비슷하게 지정됩니다. 전체 목표 세트는 표 12–1에 설명되어 있습니다.

목표에는 두 가지 범주가 있습니다.

작업 부하 종속
: 작업 부하 종속 목표는 시스템에서 실행되는 작업 부하의 특성에 따라 달라지는 목표입니다. 예로는 `utilization` 목표가 있습니다. 리소스 집합에 대한 사용률 수치는 세트에서 활성 상태인 작업 부하의 특징에 따라 달라집니다.

작업 부하 비종속
: 작업 부하 비종속 목표는 시스템에서 실행되는 작업 부하의 특성에 따라 달라지지 않는 목표입니다. 예로는 CPU `locality` 목표가 있습니다. 리소스 집합 소재지에 대해 평가한 측정값은 세트에서 활성 상태인 작업 부하의 특성에 따라 달라지지 않습니다.

세 가지 유형의 목표를 정의할 수 있습니다.

이름	유효한 요소	연산자	값
wt-load	system	해당 없음	해당 없음
locality	pset	해당 없음	loose | tight | none
utilization	pset	< > ~	0–100%

목표는 libpool 구성에 등록 정보 문자열로 저장됩니다. 등록 정보 이름은 다음과 같습니다.

- system.poold.objectives
- pset.poold.objectives

목표 구문은 다음과 같습니다.

- objectives = objective [; objective]*
- objective = [n:] keyword [op] [value]

모든 목표에는 선택적 중요도 접두어가 있습니다. 중요도는 목표의 승수 역할을 하므로 목표 기능 평가에 영향을 주는 중요도를 높입니다. 범위는 0 - INT64_MAX(9223372036854775807)입니다. 범위를 지정하지 않으면 기본 중요도 값이 1입니다.

일부 요소 유형은 두 가지 이상의 목표 유형을 지원합니다. 예로는 pset가 있습니다. 이러한 요소에 대한 목표 유형을 여러 개 지정할 수 있습니다. 또한 단일 pset 요소에 대한 사용률 목표를 여러 개 지정할 수 있습니다.

사용 예에 대해서는 166 페이지 “구성 목표를 정의하는 방법”을 참조하십시오.

wt-load 목표

wt-load 목표는 리소스 할당을 리소스 사용률에 일치시키는 구성을 사용합니다. 이 목표가 활성 상태이면 리소스를 더 많이 사용하는 리소스 세트에 리소스가 더 많이 제공됩니다. wt-load는 *weighted* **로드**를 의미합니다.

최소값 및 최대값 등록 정보를 사용하여 설정한 제약 조건에 만족하며 그러한 제약 조건 내에서 데몬이 리소스를 자유롭게 조작할 수 있도록 하려면 이 목표를 사용합니다.

locality 목표

locality 목표는 소재지 그룹(lgroup) 데이터로 측정했을 때 소재지가 선택한 구성에 미치는 영향을 제어합니다. 소재지에 대한 다른 정의는 대기 시간입니다. lgroup은 CPU 및 메모리 리소스를 설명합니다. lgroup은 Oracle Solaris 시스템이 시간을 측정 단위로 사용하여 리소스 간의 거리를 결정하는 데 사용합니다. 소재지 그룹 추상화에 대한 자세한 내용은 **Programming Interfaces Guide**의 “Locality Groups Overview”를 참조하십시오.

이 목표는 다음 세 가지 값 중 하나를 사용할 수 있습니다.

`tight` 설정하면 리소스 소재지를 최대화하는 구성을 사용합니다.

`loose` 설정하면 리소스 소재지를 최소화하는 구성을 사용합니다.

`none` 설정하면 리소스 소재지에 영향을 받지 않는 구성을 사용합니다. `locality` 목표의 기본값입니다.

일반적으로 `locality` 목표는 `tight`로 설정해야 합니다. 하지만 메모리 대역폭을 최대화하거나 리소스 집합에 대한 DR 작업의 영향을 최소화하기 위해 이 목표를 `loose`로 설정하거나 기본 설정 `none`으로 유지할 수 있습니다.

utilization 목표

`utilization` 목표는 지정된 사용률 목표를 충족하지 않는 분할 영역에 리소스를 할당하는 구성을 사용합니다.

이 목표는 연산자와 값을 사용하여 지정합니다. 연산자는 다음과 같습니다.

`<` "미만(less than)" 연산자는 지정한 값이 최대 대상 값임을 나타냅니다.

`>` "초과(greater than)" 연산자는 지정한 값이 최소 대상 값임을 나타냅니다.

`~` "about" 연산자는 지정한 값이 일부 변동이 허용되는 대상 값임을 나타냅니다.

pset는 각 연산자 종류에 대해 하나의 사용률 목표 세트만 사용할 수 있습니다.

- `~` 연산자를 설정하면 `<` 및 `>` 연산자를 설정할 수 없습니다.
- `<` 및 `>` 연산자를 설정하면 `~` 연산자를 설정할 수 없습니다. `<` 연산자와 `>` 연산자 설정은 서로 모순될 수 없습니다.

범위를 만들기 위해 `<` 및 `>` 연산자를 모두 설정할 수 있습니다. 값이 겹치지 않도록 유효성을 검사합니다.

구성 목표 예

다음 예에서 `poold`는 pset에 대한 다음 목표를 평가합니다.

- `utilization`은 30% ~ 80%를 유지해야 합니다.
- `locality`는 프로세서 세트에 대해 최대화되어야 합니다.
- 이 목표는 기본 중요도 `1`을 사용해야 합니다.

예 12–1 poold 목표 예

```
pset.poold.objectives "utilization > 30; utilization < 80; locality tight"
```

추가 사용 예에 대해서는 166 페이지 “구성 목표를 정의하는 방법”을 참조하십시오.

pool 등록 정보

등록 정보에는 네 가지 범주가 있습니다.

- 구성
- 제약 조건
- 목표
- 목표 매개변수

표 12–1 정의된 등록 정보 이름

등록 정보 이름	유형	범주	설명
system.poold.log-level	문자열	구성	로깅 레벨
system.poold.log-location	문자열	구성	로깅 위치
system.poold.monitor-interval	uint64	구성	모니터링 샘플 간격
system.poold.history-file	문자열	구성	결정 내역 위치
pset.max	uint64	제약 조건	이 프로세서 세트의 최대 CPU 수
pset.min	uint64	제약 조건	이 프로세서 세트의 최소 CPU 수
cpu.pinned	부울	제약 조건	이 프로세서 세트에 고정된 CPU
system.poold.objectives	문자열	목표	poold의 목표 표현식 구문 다음에 오는 서식이 지정된 문자열
pset.poold.objectives	문자열	목표	poold의 표현식 구문 다음에 오는 서식이 지정된 문자열
pool.importance	int64	목표 매개변수	사용자가 지정한 중요도

구성할 수 있는 poold 기능

다음과 같은 데몬 동작을 구성할 수 있습니다.

- 모니터링 간격
- 로깅 레벨
- 로깅 위치

이러한 옵션은 풀 구성에서 지정됩니다. poold를 호출하여 명령줄에서 로깅 레벨을 제어할 수도 있습니다.

poold 모니터링 간격

값을 밀리초로 지정하려면 등록 정보 이름 system.poold.monitor-interval을 사용합니다.

poold 로깅 정보

로깅을 통해 세 가지 범주의 정보가 제공됩니다. 이러한 범주는 로그에서 확인할 수 있습니다.

- 구성
- 모니터링
- 최적화

등록 정보 이름 system.poold.log-level을 사용하여 로깅 매개변수를 지정합니다. 이 등록 정보를 지정하지 않은 경우 기본 로깅 레벨은 NOTICE입니다. 이 매개변수 레벨은 계층적입니다. DEBUG의 로그 레벨 설정으로 인해 poold가 정의된 모든 메시지를 기록합니다. INFO 레벨은 대부분의 관리자에게 유용한 정보를 제공합니다.

명령줄에서 -l 옵션 및 매개변수와 함께 poold 명령을 사용하여 생성되는 로깅 정보의 레벨을 지정할 수 있습니다.

사용할 수 있는 매개변수는 다음과 같습니다.

- ALERT
- CRIT
- ERR
- WARNING
- NOTICE
- INFO
- DEBUG

매개변수 레벨은 동일한 기능의 해당 syslog에 직접 매핑됩니다. syslog 사용에 대한 자세한 내용은 149 페이지 “로깅 위치”를 참조하십시오.

poold 로깅 구성 방법에 대한 자세한 내용은 168 페이지 “poold 로깅 레벨을 설정하는 방법”을 참조하십시오.

구성 정보 로깅

생성할 수 있는 메시지 유형은 다음과 같습니다.

ALERT — libpool 구성에 액세스할 때 발생하는 문제이거나, 예기치 못한 기본적인 libpool 기능 오류입니다. 데몬이 종료되며 관리자의 즉각적인 주의가 필요합니다.

CRIT — 예기치 못한 오류로 인한 문제입니다. 데몬이 종료되며 관리자의 즉각적인 주의가 필요합니다.

ERR 리소스 집합의 해결할 수 없는 사용률 목표 충돌 등의 작업을 제어하는 사용자 지정 매개변수의 문제입니다. 목표를 수정하려면 관리자의 개입이 필요합니다. poold가 충돌하는 목표를 무시하여 치료 작업을 하려고 하지만, 몇 가지 오류로 인해 데몬이 종료됩니다.

WARNING 기술적으로 맞지만 주어진 실행 환경에 적합하지 않을 수 있는 구성 매개변수 설정과 관련된 경고입니다. 예로는 모든 CPU 리소스를 고정된 것으로 지정하는 것입니다. 즉, poold에서 프로세서 세트 간에 CPU 리소스를 이동할 수 없습니다.

DEBUG 구성 처리를 디버깅할 때 필요한 세부 정보가 들어 있는 메시지입니다. 이 정보는 일반적으로 관리자가 사용하지 않습니다.

모니터링 정보 로깅

생성할 수 있는 메시지 유형은 다음과 같습니다.

CRIT 예기치 못한 모니터링 오류로 인한 문제입니다. 데몬이 종료되며 관리자의 즉각적인 주의가 필요합니다.

ERR 예기치 못한 모니터링 오류로 인한 문제입니다. 문제를 해결하기 위해 관리자의 개입이 필요할 수 있습니다.

NOTICE 리소스 제어 영역 변환에 대한 메시지입니다.

INFO 리소스 사용률 통계에 대한 메시지입니다.

DEBUG 모니터링 처리를 디버깅할 때 필요한 세부 정보가 들어 있는 메시지입니다. 이 정보는 일반적으로 관리자가 사용하지 않습니다.

최적화 정보 로깅

생성할 수 있는 메시지 유형은 다음과 같습니다.

WARNING 최적의 결정을 내리는 문제와 관련된 메시지가 표시될 수 있습니다. 예를 들면 최소값과 최대값 또는 고정된 구성 요소 수로 너무 좁게 제한된 리소스 집합이 있습니다.

예기치 않은 제한 사항으로 인해 최적의 재할당을 수행하는 중에 발생하는 문제에 대한 메시지가 표시될 수 있습니다. 예를 들면 바운드 리소스 사용자가 포함된 프로세서 세트에서 마지막 프로세서를 제거할 때입니다.

NOTICE 사용 가능한 구성 또는 결정 내역 대체로 인해 구현되지 않는 구성에 대한 메시지가 표시될 수 있습니다.

INFO 고려한 대체 구성에 대한 메시지가 표시될 수 있습니다.

DEBUG 최적화 처리를 디버깅할 때 필요한 세부 정보가 들어 있는 메시지입니다. 이 정보는 일반적으로 관리자가 사용하지 않습니다.

로깅 위치

system.poold.log-location 등록 정보는 poold 기록 출력 위치를 지정하는 데 사용됩니다. poold 출력에 대한 SYSLOG 위치를 지정할 수 있습니다(syslog(3C) 참조).

이 등록 정보를 지정하지 않은 경우 poold 기록 출력의 기본 위치는 /var/log/pool/poold입니다.

poold가 명령줄에서 호출되면 이 등록 정보가 사용되지 않습니다. 로그 항목이 호출 터미널에서 stderr에 기록됩니다.

logadm으로 로그 관리

poold가 활성 상태이면 logadm.conf 파일에 기본 파일 /var/log/pool/poold를 관리하는 항목이 포함되어 있습니다. 이 항목은 다음과 같습니다.

```
/var/log/pool/poold -N -s 512k
```

logadm(1M) 및 logadm.conf(4) 매뉴얼 페이지를 참조하십시오.

동적 리소스 할당이 작동하는 방식

이 절에서는 poold가 리소스를 동적으로 할당하기 위해 사용하는 프로세스 및 인자를 설명합니다.

사용 가능한 리소스 정보

사용 가능한 리소스는 poold 프로세스 범위 내에서 사용할 수 있는 모든 리소스로 간주됩니다. 제어 범위는 최대 단일 Oracle Solaris 인스턴스입니다.

영역을 사용하는 시스템에서 poold의 실행 인스턴스 범위는 전역 영역으로 제한됩니다.

사용 가능한 리소스 결정

리소스 풀에는 응용 프로그램에서 사용할 수 있는 모든 시스템 리소스가 포함됩니다.

단일 실행 Oracle Solaris 인스턴스의 경우 CPU와 같은 단일 유형의 리소스는 단일 분할 영역에 할당되어야 합니다. 각 유형의 리소스에는 한 개 이상의 분할 영역이 있을 수 있습니다. 각 분할 영역에는 고유한 리소스 집합이 포함됩니다.

예를 들어, CPU가 네 개 있고 프로세서 세트가 두 개 있는 시스템은 다음과 같은 설정을 사용할 수 있습니다.

pset 0: 0 1

pset 1: 2 3

여기서, 콜론 뒤에 있는 0, 1, 2 및 3은 CPU ID를 나타냅니다. 두 개의 프로세서 세트에 모두 네 개의 CPU가 포함됩니다.

같은 시스템은 다음과 같은 설정을 사용할 수 없습니다.

pset 0: 0 1

pset 1: 1 2 3

CPU 1은 한 번에 하나의 pset에만 표시될 수 있으므로 이 설정을 사용할 수 없습니다.

리소스가 속해 있는 분할 영역 이외의 분할 영역에서 리소스에 액세스할 수 없습니다.

사용 가능한 리소스를 찾기 위해 `poold`는 활성 풀 구성에 대한 정보를 얻어 분할 영역을 찾습니다. 모든 분할 영역 내에 있는 리소스를 모두 합하여 제어되는 각 리소스 유형에 사용할 수 있는 총 리소스의 양을 결정합니다.

이 리소스 양은 `poold`가 해당 작업에 사용하는 기본 수치입니다. 하지만 `poold`가 할당해야 하는 유연성을 제한하는 제약 조건이 있습니다. 사용 가능한 제약 조건에 대한 자세한 내용은 142 페이지 “구성 제약 조건”을 참조하십시오.

리소스 부족 식별

`poold`의 제어 범위는 `poold`가 유효한 분할 및 관리에 대한 주요 책임이 있는 사용 가능한 리소스 집합으로 정의됩니다. 하지만 이 제어 범위 내에서 리소스를 조작할 수 있는 다른 방식이 여전히 구성에 영향을 줄 수 있습니다. `poold`가 활성 상태일 때 분할 영역이 제어를 벗어나는 경우 `poold`는 사용 가능한 리소스를 적절히 조작하여 제어를 복원하려고 합니다. `poold`가 해당 범위 내에서 추가 리소스를 찾을 수 없는 경우 데몬은 리소스 부족에 대한 정보를 기록합니다.

리소스 사용률 결정

`poold`는 일반적으로 해당 제어 범위 내에서 리소스의 사용을 관찰하는 데 많은 시간을 할애합니다. 이러한 모니터링은 작업 부하 종속 목표를 충족하는지 확인하기 위해 수행됩니다.

예를 들어, 프로세서 세트에서 한 세트에 있는 모든 프로세서를 측정합니다. 리소스 사용률은 샘플 간격 동안 리소스가 사용 중인 시간의 비율을 나타냅니다. 리소스 사용률은 0부터 100까지의 백분율로 표시됩니다.

제어 위반 식별

142 페이지 “구성 제약 조건 및 목표”에서 설명된 지시어는 해당 목표를 충족하기 위해 시스템의 접근 오류를 감지하는 데 사용됩니다. 이러한 목표는 작업 부하와 직접 관련되어 있습니다.

사용자가 구성한 목표를 충족하지 않는 분할 영역은 제어 위반입니다. 제어 위반 유형에는 동기와 비동기의 두 가지가 있습니다.

- 목표의 동기적 위반은 작업 부하 모니터링 과정에서 데몬에 의해 감지됩니다.
- 목표의 비동기적 위반은 데몬의 모니터링 작업과 별도로 발생합니다.

다음과 같은 이벤트로 인해 비동기적 목표 위반이 발생합니다.

- 리소스가 제어 범위에 추가되거나 제어 범위에서 제거됩니다.
- 제어 범위가 재구성됩니다.
- `poold` 리소스 제어기가 다시 시작됩니다.

작업 부하와 관련이 없는 목표의 영향은 목표 기능의 평가 간에 일정하게 유지된다고 가정합니다. 작업 부하와 관련이 없는 목표는 비동기적 위반 중 하나를 통해 재평가가 트리거되는 경우에만 재평가됩니다.

적합한 치료 작업 결정

리소스 제어기가 리소스 사용자에게 리소스가 부족하다고 판단하면 초기 대응은 리소스를 늘려 성능을 향상시키기 것입니다.

구성에서 제어 범위에 대해 지정된 목표를 충족하는 대체 구성을 검사하고 평가합니다.

이 프로세스는 리소스 이동 결과가 모니터되고 각 리소스 분할 영역에 대한 응답성이 평가되므로 시간이 지남에 따라 개선됩니다. 결정 내역을 참조하여 이전에 목표 기능 달성에서 개선을 보이지 않은 재구성을 제거합니다. 프로세스 이름 및 수량과 같은 기타 정보는 과거 데이터의 관련성을 평가하는 데 사용됩니다.

데몬이 수정 작업을 할 수 없는 경우 상태가 기록됩니다. 자세한 내용은 147 페이지 “`poold` 로깅 정보”를 참조하십시오.

poolstat를 사용하여 풀 기능 및 리소스 사용률 모니터

poolstat 유틸리티는 시스템에서 풀이 사용되는 경우 리소스 사용률을 모니터하는 데 사용됩니다. 이 유틸리티는 시스템에 있는 활성 풀을 모두 반복해서 검사하고 선택한 출력 모드를 기준으로 통계를 보고합니다. poolstat 통계를 사용하여 사용률이 높은 리소스 분할 영역을 확인할 수 있습니다. 이러한 통계를 분석하여 시스템에 리소스가 부족할 때 리소스 재할당에 대한 결정을 내릴 수 있습니다.

poolstat 유틸리티에는 특정 풀을 검사하고 리소스 집합별 통계를 보고하는 데 사용할 수 있는 옵션이 포함되어 있습니다.

시스템에 영역이 구현되고 poolstat를 비전역 영역에서 사용하면 이 영역의 풀과 연결된 리소스에 대한 정보가 표시됩니다.

poolstat 유틸리티에 대한 자세한 내용은 poolstat(1M) 매뉴얼 페이지를 참조하십시오. poolstat 작업 및 사용 정보에 대해서는 173 페이지 “poolstat를 사용하여 풀 관련 리소스에 대한 통계 보고”를 참조하십시오.

poolstat 출력

기본 출력 형식에서 poolstat는 제목 행을 출력한 다음 각 풀에 대한 행을 표시합니다. 풀 행은 풀 ID과 풀 이름으로 시작하며, 그 뒤에 풀에 연결된 프로세서 세트에 대한 통계 데이터 열이 표시됩니다. 두 개 이상의 풀에 연결된 리소스 집합은 각 풀에 대해 한 번씩 여러 번 나열됩니다.

열 제목은 다음과 같습니다.

`id` 풀 ID

`pool` 풀 이름

`rid` 리소스 집합 ID

`rset` 리소스 집합 이름

`type` 리소스 집합 유형

`min` 최소 리소스 집합 크기

`max` 최대 리소스 집합 크기

`size` 현재 리소스 집합 크기

`used` 현재 사용 중인 리소스 집합의 측정량.

이 사용량은 리소스 세트의 사용률과 리소스 세트의 크기를 곱한 값의 백분율로 계산됩니다. 마지막 샘플링 간격 중에 리소스 세트가 재구성된 경우 이 값이 보고되지 않을 수 있습니다. 보고되지 않은 값은 하이픈(-)으로 표시됩니다.

`load` 리소스 세트에 지정된 로드의 절대 표시.

이 등록 정보에 대한 자세한 내용은 `libpool(3LIB)` 매뉴얼 페이지를 참조하십시오.

`poolstat` 출력에 다음을 지정할 수 있습니다.

- 열 순서
- 표시되는 제목

poolstat 작업 간격 조정

`poolstat`에 의해 수행되는 작업을 사용자 정의할 수 있습니다. 보고서에 대한 샘플링 간격을 설정하고 통계가 반복되는 횟수를 지정할 수 있습니다.

interval `poolstat`에 의해 수행되는 주기적인 작업에 대한 간격을 조정합니다. 모든 간격은 초 단위로 지정됩니다.

count 통계가 반복되는 횟수를 지정합니다. 기본적으로 `poolstat`는 통계를 한 번만 보고합니다.

interval 및 *count*를 지정하지 않은 경우에는 통계가 한 번만 보고됩니다. *interval*을 지정하고 *count*를 지정하지 않은 경우에는 통계가 무한대로 보고됩니다.

리소스 풀 기능에 사용되는 명령

다음 표에 설명된 명령은 풀 기능에 대한 기본 관리 인터페이스를 제공합니다. 영역을 사용하는 시스템에서 이러한 명령을 사용하는 방법에 대한 자세한 내용은 135 페이지 “영역에서 사용되는 리소스 풀”을 참조하십시오.

매뉴얼 페이지 참조	설명
`pooladm(1M)`	시스템에서 풀 기능을 사용 또는 사용 안함으로 설정합니다. 특정 구성을 활성화하거나 현재 구성을 제거하고 관련된 리소스를 기본 상태로 되돌립니다. 옵션을 지정하지 않고 실행하면 `pooladm`은 현재의 동적 풀 구성을 인쇄합니다.
`poolbind(1M)`	리소스 풀에 대한 프로젝트, 작업 및 프로세스의 수동 바인딩을 사용합니다.

매뉴얼 페이지 참조	설명
poolcfg(1M)	풀과 세트에 대한 구성 작업을 제공합니다. 이 도구를 사용하여 만든 구성은 pooladm을 사용하여 대상 호스트에서 인스턴스화됩니다. -c 옵션에 info 하위 명령 인수와 함께 실행하면 poolcfg는 /etc/pooladm.conf에 정적 구성에 대한 정보를 표시합니다. 파일 이름 인수가 추가되면 이 명령은 명명된 파일에 있는 정적 구성에 대한 정보를 표시합니다. 예를 들어, poolcfg - c info /tmp/newconfig는 /tmp/newconfig 파일에 포함된 정적 구성에 대한 정보를 표시합니다.
poold(1M)	풀 시스템 데몬입니다. 이 데몬은 시스템 대상 및 관찰 가능한 통계를 사용하여 관리자가 지정한 시스템 성능 목표를 유지합니다. 목표를 충족하지 못했을 때 수정 작업을 할 수 없는 경우에는 poold가 상태를 기록합니다.
poolstat(1M)	풀 관련 리소스에 대한 통계를 표시합니다. 성능 분석을 단순화하고 리소스 분할 및 재분할 작업에서 시스템 관리자를 지원하는 정보를 제공합니다. 지정된 풀 검사 및 리소스 세트별 통계 보고에 대한 옵션이 제공됩니다.

libpool에 의해 라이브러리 API가 제공됩니다(libpool(3LIB) 매뉴얼 페이지 참조). 이 라이브러리는 프로그램이 풀 구성을 조작할 때 사용할 수 있습니다.

◆ ◆ ◆ 13 장

13 리소스 풀 만들기 및 관리(작업)

이 장에서는 시스템에서 리소스 풀을 설정하고 관리하는 방법에 대해 설명합니다.

리소스 풀에 대한 배경 정보에 대해서는 12 장, “리소스 풀(개요)”을 참조하십시오.

리소스 풀 관리(작업 맵)

작업	설명	수행 방법
리소스 풀 사용 또는 사용 안함으로 설정	시스템에서 리소스 풀을 활성화하거나 사용 안함으로 설정합니다.	157 페이지 “풀 기능을 사용 또는 사용 안함으로 설정”
동적 리소스 풀 사용 또는 사용 안함으로 설정	시스템에서 동적 리소스 풀 기능을 활성화하거나 사용 안함으로 설정합니다.	157 페이지 “풀 기능을 사용 또는 사용 안함으로 설정”
정적 리소스 풀 구성 만들기	현재 동적 구성과 일치하는 정적 구성 파일을 만듭니다. 자세한 내용은 137 페이지 “리소스 풀 프레임워크”를 참조하십시오.	161 페이지 “정적 구성을 만드는 방법”
리소스 풀 구성 수정	예를 들면 추가 풀을 만들어 시스템에서 풀 구성을 수정합니다.	162 페이지 “구성을 수정하는 방법”
리소스 풀과 예약 클래스 연결	풀에 바인드된 모든 프로세스가 지정된 스케줄러를 사용하도록 풀과 예약 클래스를 연결합니다.	164 페이지 “풀과 예약 클래스를 연결하는 방법”

작업	설명	수행 방법
구성 제약 조건 설정 및 구성 목표 정의	수정 작업을 할 때 고려할 poold에 대한 목표를 지정합니다. 구성 목표에 대한 자세한 내용은 141 페이지 “poold 개요”를 참조하십시오.	166 페이지 “구성 제약 조건을 설정하는 방법” 및 166 페이지 “구성 목표를 정의하는 방법”
로깅 레벨 설정	poold가 생성하는 로깅 정보 레벨을 지정합니다.	168 페이지 “poold 로깅 레벨을 설정하는 방법”
poolcfg 명령에 텍스트 파일 사용	poolcfg 명령은 텍스트 파일에서 입력을 받을 수 있습니다.	169 페이지 “poolcfg에서 명령 파일을 사용하는 방법”
커널에서 리소스를 전송합니다.	커널에서 리소스를 전송합니다. 예를 들어, 특정 ID가 있는 리소스를 대상 집합으로 전송할 수 있습니다.	169 페이지 “리소스 전송”
풀 구성 활성화	기본 구성 파일에서 구성을 활성화합니다.	170 페이지 “풀 구성을 활성화하는 방법”
구성을 커밋하기 전에 풀 구성의 유효성 검사	풀 구성의 유효성을 검사하여 유효성 검사가 수행될 때 어떻게 되는지 테스트합니다.	171 페이지 “구성을 커밋하기 전에 구성 유효성을 검사하는 방법”
시스템에서 풀 구성 제거	프로세서 세트와 같은 연결된 모든 리소스가 해당 기본 상태로 반환됩니다.	171 페이지 “풀 구성을 제거하는 방법”
프로세스를 풀에 바인드	시스템에서 실행 중인 프로세스와 리소스 풀을 수동으로 연결합니다.	172 페이지 “프로세스를 풀에 바인드하는 방법”
작업 또는 프로젝트를 풀에 바인드	작업 또는 프로젝트와 리소스 풀을 연결합니다.	172 페이지 “작업 또는 프로젝트를 풀에 바인드하는 방법”
새 프로세스를 리소스 풀에 바인드	프로젝트에서 새 프로세스를 제공된 풀에 자동으로 바인드하려면 속성을 project 데이터베이스의 각 항목에 추가합니다.	172 페이지 “프로젝트에 대한 project.pool 속성을 설정하는 방법”
project 속성을 사용하여 프로세스를 다른 풀에 바인드	시작된 새 프로세스에 대한 풀 바인딩을 수정합니다.	173 페이지 “project 속성을 사용하여 프로세스를 다른 풀에 바인드하는 방법”
poolstat 유틸리티를 사용하여 보고서 생성	여러 개의 보고서를 지정된 간격으로 생성합니다.	174 페이지 “특정 간격으로 여러 개의 보고서 생성”

작업	설명	수행 방법
리소스 집합 통계 보고	poolstat 유틸리티를 사용하여 pset 리소스 집합에 대한 통계를 보고합니다.	174 페이지 “리소스 집합 통계 보고”

풀 기능을 사용 또는 사용 안함으로 설정

svcadm(1M) 매뉴얼 페이지에 설명된 svcadm 명령을 사용하여 시스템에서 리소스 풀 및 동적 리소스 풀 서비스를 사용 및 사용 안함으로 설정할 수 있습니다.

pooladm(1M) 매뉴얼 페이지에 설명된 pooladm 명령을 사용하여 다음 작업을 수행할 수도 있습니다.

- 풀을 조작할 수 있도록 풀 기능을 사용으로 설정
- 풀을 조작할 수 없도록 풀 기능을 사용 안함으로 설정

주 – 시스템을 업그레이드할 때 리소스 풀 프레임워크가 사용되고 /etc/pooladm.conf 파일이 있으면 풀 서비스가 사용되며 파일에 포함된 구성이 시스템에 적용됩니다.

▼ svcadm을 사용하여 리소스 풀 서비스를 사용으로 설정하는 방법

1. **관리자로 전환합니다.**
2. **리소스 풀 서비스를 사용으로 설정합니다.**

```
# svcadm enable system/pools:default
```

▼ svcadm을 사용하여 리소스 풀 서비스를 사용 안함으로 설정하는 방법

1. **관리자로 전환합니다.**
2. **리소스 풀 서비스를 사용 안함으로 설정합니다.**

```
# svcadm disable system/pools:default
```

▼ svcadm을 사용하여 동적 리소스 풀 서비스를 사용으로 설정하는 방법

1. **관리자로 전환합니다.**

2. **동적 리소스 풀 서비스를 사용으로 설정합니다.**

```
# svcadm enable system/pools/dynamic:default
```

예 13–1 리소스 풀 서비스에서 동적 리소스 풀 서비스의 종속성

이 예에서는 DRP를 실행하려면 먼저 리소스 풀을 사용으로 설정해야 한다는 것을 보여줍니다.

리소스 풀과 동적 리소스 풀 사이에 종속성이 있습니다. DRP는 이제 리소스 풀에 종속된 서비스입니다. DRP는 리소스 풀과 별도로 사용 및 사용 안함으로 설정할 수 있습니다.

다음은 리소스 풀과 동적 리소스 풀이 현재 모두 사용 안함으로 설정되어 있음을 보여 줍니다.

```
# svcs *pool*
STATE          STIME    FMRI
disabled       10:32:26 svc:/system/pools/dynamic:default
disabled       10:32:26 svc:/system/pools:default
```

동적 리소스 풀을 사용으로 설정:

```
# svcadm enable svc:/system/pools/dynamic:default
# svcs -a | grep pool
disabled       10:39:00 svc:/system/pools:default
offline        10:39:12 svc:/system/pools/dynamic:default
```

DRP 서비스가 아직 오프라인 상태입니다.

svcs 명령의 -x 옵션을 사용하여 DRP 서비스가 오프라인 상태인 이유를 확인합니다.

```
# svcs -x *pool*
svc:/system/pools:default (resource pools framework)
 State: disabled since Wed 25 Jan 2006 10:39:00 AM GMT
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: libpool(3LIB)
   See: pooladm(1M)
   See: poolbind(1M)
   See: poolcfg(1M)
   See: poolstat(1M)
   See: /var/svc/log/system-pools:default.log
Impact: 1 dependent service is not running.  (Use -v for list.)

svc:/system/pools/dynamic:default (dynamic resource pools)
 State: offline since Wed 25 Jan 2006 10:39:12 AM GMT
```

```
Reason: Service svc:/system/pools:default is disabled.
   See: http://sun.com/msg/SMF-8000-GE
   See: poold(1M)
   See: /var/svc/log/system-pools-dynamic:default.log
Impact: This service is not running.
```

DRP 서비스가 실행될 수 있도록 리소스 풀 서비스를 사용으로 설정:

```
# svcadm enable svc:/system/pools:default
```

svcs *pool* 명령을 사용하면 시스템에 다음과 같이 표시됩니다.

```
# svcs *pool*
STATE          STIME    FMRI
online         10:40:27 svc:/system/pools:default
online         10:40:27 svc:/system/pools/dynamic:default
```

예 13–2 리소스 풀 서비스를 사용 안함으로 설정했을 때 동적 리소스 풀에 대한 영향

두 서비스가 온라인 상태이고 리소스 풀 서비스를 사용 안함으로 설정한 경우:

```
# svcadm disable svc:/system/pools:default
```

svcs *pool* 명령을 사용하면 시스템에 다음과 같이 표시됩니다.

```
# svcs *pool*
STATE          STIME    FMRI
disabled       10:41:05 svc:/system/pools:default
online         10:40:27 svc:/system/pools/dynamic:default
# svcs *pool*
STATE          STIME    FMRI
disabled       10:41:05 svc:/system/pools:default
online         10:40:27 svc:/system/pools/dynamic:default
```

하지만 리소스 풀 서비스가 사용 안함으로 설정되었기 때문에 결과적으로 DRP 서비스는 offline 상태가 됩니다.

```
# svcs *pool*
STATE          STIME    FMRI
disabled       10:41:05 svc:/system/pools:default
offline        10:41:12 svc:/system/pools/dynamic:default
```

DRP 서비스가 오프라인 상태인 이유 확인:

```
# svcs -x *pool*
svc:/system/pools:default (resource pools framework)
 State: disabled since Wed 25 Jan 2006 10:41:05 AM GMT
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: libpool(3LIB)
   See: pooladm(1M)
   See: poolbind(1M)
```

```
   See: poolcfg(1M)
   See: poolstat(1M)
   See: /var/svc/log/system-pools:default.log
Impact: 1 dependent service is not running.  (Use -v for list.)

svc:/system/pools/dynamic:default (dynamic resource pools)
 State: offline since Wed 25 Jan 2006 10:41:12 AM GMT
Reason: Service svc:/system/pools:default is disabled.
   See: http://sun.com/msg/SMF-8000-GE
   See: poold(1M)
   See: /var/svc/log/system-pools-dynamic:default.log
Impact: This service is not running.
```

DRP가 작동하려면 리소스 풀을 시작해야 합니다. 예를 들어, `pooladm` 명령을 `-e` 옵션과 함께 사용하여 리소스 풀을 시작할 수 있습니다.

```
# pooladm -e
```

그러면 `svcs *pool*` 명령을 실행하면 다음과 같이 표시됩니다.

```
# svcs *pool*
STATE          STIME    FMRI
online         10:42:23 svc:/system/pools:default
online         10:42:24 svc:/system/pools/dynamic:default
```

▼ svcadm을 사용하여 동적 리소스 풀을 사용 안함으로 설정하는 방법

1. **관리자로 전환합니다.**
2. **동적 리소스 풀 서비스를 사용 안함으로 설정합니다.**

   ```
   # svcadm disable system/pools/dynamic:default
   ```

▼ pooladm을 사용하여 리소스 풀을 사용으로 설정하는 방법

1. **관리자로 전환합니다.**
2. **풀 기능을 사용으로 설정합니다.**

   ```
   # pooladm -e
   ```

▼ pooladm을 사용하여 리소스 풀을 사용 안함으로 설정하는 방법

1 **관리자로 전환합니다.**

2 **풀 기능을 사용 안함으로 설정합니다.**

```
# pooladm -d
```

풀 구성

▼ 정적 구성을 만드는 방법

/usr/sbin/pooladm에 -s 옵션을 사용하여 현재 동적 구성과 일치하는 정적 구성 파일을 만들면 재부트 시 변경 사항이 유지됩니다. 다른 파일 이름을 지정하지 않는 한 기본 위치 /etc/pooladm.conf가 사용됩니다.

pooladm 명령을 -c 옵션과 함께 사용하여 구성을 커밋합니다. 그런 다음 pooladm 명령을 -s 옵션과 함께 사용하여 동적 구성 상태에 일치하도록 정적 구성을 업데이트합니다.

주 – 동적 구성에 일치하는 새 구성을 만들 때 새 기능 pooladm -s가 이전 기능 poolcfg -c discover보다 우선합니다.

시작하기 전에 시스템에서 풀을 사용으로 설정합니다.

1 **관리자로 전환합니다.**

2 **현재 동적 구성에 일치하도록 정적 구성 파일을 업데이트합니다.**

```
# pooladm -s
```

3 **읽을 수 있는 형태로 된 구성 파일의 내용을 확인합니다.**

시스템에서 생성된 기본 요소가 구성에 포함됩니다.

```
# poolcfg -c info
system tester
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
```

```
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line
```

4 **/etc/pooladm.conf에서 구성을 커밋합니다.**

```
# pooladm -c
```

5 **(옵션) 동적 구성을 /tmp/backup이라고 하는 정적 구성 파일에 복사하려면 다음을 입력합니다.**

```
# pooladm -s /tmp/backup
```

▼ 구성을 수정하는 방법

구성을 향상시키려면 프로세서 세트 pset_batch와 풀 pool_batch를 만듭니다. 그런 다음 연결을 사용하여 풀과 프로세서 세트를 결합합니다.

공백이 포함된 하위 명령 인수에 따옴표를 사용해야 합니다.

1 **관리자로 전환합니다.**

2 프로세서 세트 pset_batch를 만듭니다.

```
# poolcfg -c 'create pset pset_batch (uint pset.min = 2; uint pset.max = 10)'
```

3 풀 pool_batch를 만듭니다.

```
# poolcfg -c 'create pool pool_batch'
```

4 연결을 사용하여 풀과 프로세서 세트를 결합합니다.

```
# poolcfg -c 'associate pool pool_batch (pset pset_batch)'
```

5 편집한 구성을 표시합니다.

```
# poolcfg -c info
system tester
        string  system.comment kernel state
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line
```

```
pool pool_batch
        boolean pool.default false
        boolean pool.active true
        int pool.importance 1
        string pool.comment
        pset pset_batch

pset pset_batch
        int pset.sys_id -2
        string pset.units population
        boolean pset.default true
        uint pset.max 10
        uint pset.min 2
        string pset.comment
        boolean pset.escapable false
        uint pset.load 0
        uint pset.size 0

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line
```

6 **/etc/pooladm.conf에서 구성을 커밋합니다.**

```
# pooladm -c
```

7 **(옵션) 동적 구성을 /tmp/backup이라고 하는 정적 구성 파일에 복사하려면 다음을 입력합니다.**

```
# pooladm -s /tmp/backup
```

▼ 풀과 예약 클래스를 연결하는 방법

풀에 바인드된 모든 프로세스가 이 스케줄러를 사용하도록 풀과 예약 클래스를 연결할 수 있습니다. 이렇게 하려면 `pool.scheduler` 등록 정보를 스케줄러 이름으로 설정합니다. 이 예에서는 풀 `pool_batch`와 FSS(Fair Share Scheduler)를 연결합니다.

1 **관리자로 전환합니다.**

2 **FSS와 연결할 풀 pool_batch를 수정합니다.**

```
# poolcfg -c 'modify pool pool_batch (string pool.scheduler="FSS")'
```

3 **편집한 구성을 표시합니다.**

```
# poolcfg -c info
system tester
        string  system.comment
```

```
int     system.version 1
boolean system.bind-default true
int     system.poold.pid 177916

pool pool_default
        int     pool.sys_id 0
        boolean pool.active true
        boolean pool.default true
        int     pool.importance 1
        string  pool.comment
        pset    pset_default

pset pset_default
        int     pset.sys_id -1
        boolean pset.default true
        uint    pset.min 1
        uint    pset.max 65536
        string  pset.units population
        uint    pset.load 10
        uint    pset.size 4
        string  pset.comment
        boolean testnullchanged true

        cpu
                int     cpu.sys_id 3
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 1
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 0
                string  cpu.comment
                string  cpu.status on-line

pool pool_batch
        boolean pool.default false
        boolean pool.active true
        int pool.importance 1
        string pool.comment
        string pool.scheduler FSS
        pset batch

pset pset_batch
        int pset.sys_id -2
        string pset.units population
        boolean pset.default true
        uint pset.max 10
        uint pset.min 2
        string pset.comment
        boolean pset.escapable false
```

```
uint pset.load 0
uint pset.size 0

cpu
        int     cpu.sys_id 5
        string  cpu.comment
        string  cpu.status on-line

cpu
        int     cpu.sys_id 4
        string  cpu.comment
        string  cpu.status on-line
```

4 **/etc/pooladm.conf 에서 구성을 커밋합니다.**

```
# pooladm -c
```

5 **(옵션) 동적 구성을 /tmp/backup이라고 하는 정적 구성 파일에 복사하려면 다음을 입력합니다.**

```
# pooladm -s /tmp/backup
```

▼ 구성 제약 조건을 설정하는 방법

제약 조건은 구성에 적용할 수 있는 잠재적 변경 사항의 일부를 제거하여 가능한 구성 범위에 영향을 줍니다. 이 절차는 cpu.pinned 등록 정보를 설정하는 방법을 보여 줍니다.

다음 예에서 cpuid는 정수입니다.

1 **관리자로 전환합니다.**

2 **정적 또는 동적 구성에서 cpu.pinned 등록 정보를 수정합니다.**

- **부트 시간(정적) 구성 수정:**

  ```
  # poolcfg -c 'modify cpu <cpuid> (boolean cpu.pinned = true)'
  ```

- **부트 시간 구성을 수정하지 않고 실행 중인(동적) 구성 수정:**

  ```
  # poolcfg -dc 'modify cpu <cpuid> (boolean cpu.pinned = true)'
  ```

▼ 구성 목표를 정의하는 방법

수정 작업을 할 때 고려할 poold에 대한 목표를 지정할 수 있습니다.

다음 절차에서는 poold가 리소스 할당을 리소스 사용률에 일치시키도록 하기 위해 wt-load 목표를 설정합니다. 이 구성 목표를 달성할 수 있도록 하기 위해 locality 목표가 사용 안함으로 설정됩니다.

1 **관리자로 전환합니다.**

2 **wt-load 목표를 사용하도록 시스템 tester를 수정합니다.**

```
# poolcfg -c 'modify system tester (string system.poold.objectives="wt-load")'
```

3 **기본 프로세서 세트에 대해 locality 목표를 사용 안함으로 설정합니다.**

```
# poolcfg -c 'modify pset pset_default (string pset.poold.objectives="locality none")'
```
one line

4 **pset_batch 프로세서 세트에 대해 locality 목표를 사용 안함으로 설정합니다.**

```
# poolcfg -c 'modify pset pset_batch (string pset.poold.objectives="locality none")'
```
one line

5 **편집한 구성을 표시합니다.**

```
# poolcfg -c info
system tester
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916
        string  system.poold.objectives wt-load

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true
                string  pset.poold.objectives locality none

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
```

```
                string  cpu.comment
                string  cpu.status on-line

        pool pool_batch
                boolean pool.default false
                boolean pool.active true
                int pool.importance 1
                string pool.comment
                string pool.scheduler FSS
                pset batch

        pset pset_batch
                int pset.sys_id -2
                string pset.units population
                boolean pset.default true
                uint pset.max 10
                uint pset.min 2
                string pset.comment
                boolean pset.escapable false
                uint pset.load 0
                uint pset.size 0
                string  pset.poold.objectives locality none

                cpu
                        int     cpu.sys_id 5
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 4
                        string  cpu.comment
                        string  cpu.status on-line
```

6 **/etc/pooladm.conf에서 구성을 커밋합니다.**

```
# pooladm -c
```

7 **(옵션) 동적 구성을 /tmp/backup이라고 하는 정적 구성 파일에 복사하려면 다음을 입력합니다.**

```
# pooladm -s /tmp/backup
```

▼ poold 로깅 레벨을 설정하는 방법

poold에서 생성하는 로깅 정보 레벨을 지정하려면 poold 구성에서 system.poold.log-level 등록 정보를 설정합니다. poold 구성은 libpool 구성에 보관됩니다. 자세한 내용은 147 페이지 “poold 로깅 정보”와 poolcfg(1M) 및 libpool(3LIB) 매뉴얼 페이지를 참조하십시오.

명령줄에서 poold 명령을 사용하여 poold가 생성하는 로깅 정보 레벨을 지정할 수도 있습니다.

1 **관리자로 전환합니다.**

2 **-l 옵션 및 매개변수(예: INFO)와 함께 poold 명령을 사용하여 로깅 레벨을 설정합니다.**

```
# /usr/lib/pool/poold -l INFO
```

사용 가능한 매개변수에 대한 자세한 내용은 147 페이지 “poold 로깅 정보”를 참조하십시오. 기본 로깅 레벨은 NOTICE입니다.

▼ poolcfg에서 명령 파일을 사용하는 방법

poolcfg 명령을 -f 옵션과 함께 사용하면 -c 옵션에 대한 poolcfg 하위 명령 인수가 포함된 텍스트 파일에서 입력을 받을 수 있습니다. 이 방법은 작업 세트를 수행하고자 할 때 적합합니다. 여러 개의 명령을 처리할 때 명령이 모두 성공한 경우에만 구성이 업데이트됩니다. 구성이 크거나 복잡한 경우 이 방법이 하위 명령별 호출보다 유용할 수 있습니다.

명령 파일에서 # 문자는 나머지 행에 대한 주석 표시 역할을 합니다.

1 **입력 파일 poolcmds.txt 를 만듭니다.**

```
$ cat > poolcmds.txt
create system tester
create pset pset_batch (uint pset.min = 2; uint pset.max = 10)
create pool pool_batch
associate pool pool_batch (pset pset_batch)
```

2 **관리자로 전환합니다.**

3 **명령을 실행합니다.**

```
# /usr/sbin/poolcfg -f poolcmds.txt
```

리소스 전송

-d 옵션과 함께 poolcfg의 -c 옵션에 transfer 하위 명령 인수를 사용하여 커널에서 리소스를 전송합니다. -d 옵션은 명령이 커널에서 바로 실행되고 파일에서 입력을 받지 않도록 지정합니다.

다음 절차는 커널에서, 프로세서 세트 pset1에서 프로세서 세트 pset2로 CPU 두 개를 이동합니다.

▼ 프로세서 세트 간에 CPU를 이동하는 방법

1 **관리자로 전환합니다.**

2 **pset1에서 pset2로 CPU 두 개를 이동합니다.**

`from` 및 `to` 하위 절은 순서에 관계없이 사용할 수 있습니다. 명령별로 하나의 `to` 및 `from` 하위 절만 지원됩니다.

```
# poolcfg -dc 'transfer 2 from pset pset1 to pset2'
```

예 13–3 프로세서 세트 간에 CPU를 이동하는 다른 방법

리소스 유형의 알려진 특정 ID를 전송하려면 대체 구문을 제공합니다. 예를 들어, 다음 명령은 ID가 `0`과 `2`인 두 개의 CPU를 `pset_large` 프로세서 세트에 지정합니다.

```
# poolcfg -dc 'transfer to pset pset_large (cpu 0; cpu 2)'
```

자세한 정보 문제 해결

요청에 일치하는 리소스가 부족하거나 지정된 ID를 찾을 수 없기 때문에 전송이 실패하면 시스템에 오류 메시지가 표시됩니다.

풀 구성 활성화 및 제거

특정 풀 구성을 활성화하거나 현재 활성 상태인 풀 구성을 제거하려면 `pooladm` 명령을 사용합니다. 이 명령에 대한 자세한 내용은 `pooladm(1M)` 매뉴얼 페이지를 참조하십시오.

▼ 풀 구성을 활성화하는 방법

기본 구성 파일 `/etc/pooladm.conf`에서 구성을 활성화하려면 `-c` 옵션, "구성 커밋"과 함께 `pooladm`을 호출합니다.

1 **관리자로 전환합니다.**

2 **`/etc/pooladm.conf`에서 구성을 커밋합니다.**

```
# pooladm -c
```

3 **(옵션) 동적 구성을 정적 구성 파일(예: `/tmp/backup`)에 복사합니다.**

```
# pooladm -s /tmp/backup
```

▼ 구성을 커밋하기 전에 구성 유효성을 검사하는 방법

유효성 검사가 수행될 때 어떻게 되는지 테스트하려면 -c 옵션과 함께 -n 옵션을 사용합니다. 구성이 실제로 커밋되지는 않습니다.

다음 명령은 /home/admin/newconfig에 포함된 구성의 유효성을 검사합니다. 발생한 오류 상태가 표시되지만 구성 자체는 수정되지 않습니다.

1. **관리자로 전환합니다.**
2. **구성을 커밋하기 전에 유효성을 테스트합니다.**

```
# pooladm -n -c /home/admin/newconfig
```

▼ 풀 구성을 제거하는 방법

현재 활성 구성을 제거하고 프로세서 세트와 같은 연결된 모든 리소스를 기본 상태로 되돌리려면 "구성 제거"를 위한 -x 옵션을 사용합니다.

1. **관리자로 전환합니다.**
2. **현재 활성 구성을 제거합니다.**

```
# pooladm -x
```

pooladm에 대한 -x 옵션은 동적 구성에서 사용자 정의 요소를 모두 제거합니다. 모든 리소스는 기본 상태로 되돌아가고 모든 풀 바인딩은 기본 풀에 대한 바인딩으로 대체됩니다.

자세한 정보

프로세서 세트 내에서 예약 클래스 혼합

같은 프로세서 세트에 있는 TS와 IA 클래스의 프로세스를 안전하게 혼합할 수 있습니다. 한 프로세서 세트 내에서 다른 예약 클래스를 혼합하면 예측할 수 없는 결과가 나타날 수 있습니다. pooladm -x를 사용하여 한 프로세서 세트 내에서 예약 클래스가 혼합된 경우에는 priocntl 명령을 사용하여 실행 중인 프로세스를 다른 예약 클래스로 이동합니다. 114 페이지 “TS 클래스에서 FSS 클래스로 프로세스를 수동으로 이동하는 방법”을 참조하십시오. priocntl(1) 매뉴얼 페이지도 참조하십시오.

풀 속성 설정 및 풀에 바인드

리소스 풀과 프로젝트를 연결하도록 project.pool 속성을 설정할 수 있습니다.

실행 중인 프로세스를 다음 두 가지 방법으로 풀에 바인드할 수 있습니다.

- poolbind(1M) 명령에서 설명한 poolbind 명령을 사용하여 특정 프로세스를 명명된 리소스 풀에 바인드할 수 있습니다.
- project 데이터베이스에서 project.pool 속성을 사용하여 newtask 명령을 통해 시작된 작업이나 새 로그인 세션에 대한 풀 바인딩을 식별할 수 있습니다. newtask(1), projmod(1M) 및 project(4) 매뉴얼 페이지를 참조하십시오.

▼ 프로세스를 풀에 바인드하는 방법

다음 절차에서는 -p 옵션과 함께 poolbind를 사용하여 프로세스(이 경우 현재 셸)를 풀 ohare에 수동으로 바인드합니다.

1. **관리자로 전환합니다.**

2. **프로세스를 풀에 수동으로 바인드:**

```
# poolbind -p ohare $$
```

3. **-q 옵션과 함께 poolbind를 사용하여 프로세스에 대한 풀 바인딩을 확인합니다.**

```
$ poolbind -q $$
155509 ohare
```

시스템에 프로세스 ID 및 풀 바인딩이 표시됩니다.

▼ 작업 또는 프로젝트를 풀에 바인드하는 방법

작업 또는 프로젝트를 풀에 바인드하려면 poolbind 명령을 -i 옵션과 함께 사용합니다. 다음 예에서는 airmiles 프로젝트에서 모든 프로세스를 laguardia 풀에 바인드합니다.

1. **관리자로 전환합니다.**

2. **airmiles 프로젝트에서 모든 프로세스를 laguardia 풀에 바인드합니다.**

```
# poolbind -i project -p laguardia airmiles
```

▼ 프로젝트에 대한 project.pool 속성을 설정하는 방법

프로젝트의 프로세스를 리소스 풀에 바인드하도록 project.pool 속성을 설정할 수 있습니다.

1 **관리자로 전환합니다.**

2 **project.pool 속성을 project 데이터베이스의 각 항목에 추가합니다.**

```
# projmod -a -K project.pool=poolname  project
```

▼ project 속성을 사용하여 프로세스를 다른 풀에 바인드하는 방법

studio와 backstage라는 두 개의 풀이 있는 구성이 있다고 가정합니다. /etc/project 파일의 내용은 다음과 같습니다.

```
user.paul:1024::::project.pool=studio
user.george:1024::::project.pool=studio
user.ringo:1024::::project.pool=backstage
passes:1027::paul::project.pool=backstage
```

이 구성을 사용하면 사용자 paul이 시작한 프로세스가 기본적으로 studio 풀에 바인드됩니다.

사용자 paul은 자신이 시작한 프로세스에 대한 풀 바인딩을 수정할 수 있습니다. paul은 newtask를 사용하여 passes 프로젝트에서 시작하여 backstage 풀에 작업을 바인드할 수 있습니다.

1 **passes 프로젝트에서 프로세스를 시작합니다.**

```
$ newtask -l -p passes
```

2 **poolbind 명령을 -q 옵션과 함께 사용하여 프로세스에 대한 풀 바인딩을 확인합니다. 또한 달러 기호 두 개($$)를 사용하여 상위 셸의 프로세스 수를 명령에 전달합니다.**

```
$ poolbind -q $$
6384  pool backstage
```

시스템에 프로세스 ID 및 풀 바인딩이 표시됩니다.

poolstat를 사용하여 풀 관련 리소스에 대한 통계 보고

poolstat 명령은 풀 관련 리소스에 대한 통계를 표시하는 데 사용됩니다. 자세한 내용은 152 페이지 “poolstat를 사용하여 풀 기능 및 리소스 사용률 모니터” 및 poolstat(1M) 매뉴얼 페이지를 참조하십시오.

다음 세부절에서는 예제를 사용하여 특정 목적의 보고서를 생성하는 방법에 대해 설명합니다.

기본 poolstat 출력 표시

인수 없이 poolstat를 입력하면 각 풀에 대한 헤더 행과 정보 행이 출력됩니다. 정보 행에는 풀에 연결된 프로세서 세트에 대한 풀 ID, 풀 이름 및 리소스 통계가 표시됩니다.

```
machine% poolstat
                           pset
       id pool           size used load
        0 pool_default      4  3.6  6.2
        1 pool_sales        4  3.3  8.4
```

특정 간격으로 여러 개의 보고서 생성

다음 명령은 5초 샘플링 간격으로 세 개의 보고서를 생성합니다.

```
machine% poolstat 5 3
                            pset
 id pool                  size used load
 46 pool_sales               2  1.2  8.3
  0 pool_default             2  0.4  5.2
                            pset
 id pool                  size used load
 46 pool_sales               2  1.4  8.4
  0 pool_default             2  1.9  2.0
                            pset
 id pool                  size used load
 46 pool_sales               2  1.1  8.0
  0 pool_default             2  0.3  5.0
```

리소스 집합 통계 보고

다음 예에서는 poolstat 명령을 -r 옵션과 함께 사용하여 프로세서 세트 리소스 집합에 대한 통계를 보고합니다. 리소스 집합 pset_default가 둘 이상의 풀에 연결되므로 이 프로세서 세트는 각 풀 구성원에 대해 한 번씩 표시됩니다.

```
machine% poolstat -r pset
      id pool          type rid rset          min  max size used load
       0 pool_default  pset  -1 pset_default    1  65K    2  1.2  8.3
       6 pool_sales    pset   1 pset_sales      1  65K    2  1.2  8.3
       2 pool_other    pset  -1 pset_default    1  10K    2  0.4  5.2
```

◆ ◆ ◆ 14장

14 리소스 관리 구성 예

이 장에서는 리소스 관리 프레임워크를 검토하고 가상의 서버 통합 프로젝트에 대해 설명합니다.

이 장에서는 다음 항목을 다룹니다.

통합되는 구성

이 예에서는 다섯 개의 응용 프로그램이 단일 시스템에 통합됩니다. 대상 응용 프로그램의 리소스 요구 사항은 각기 다르며 사용자 수 및 구조도 다릅니다. 현재 각 응용 프로그램은 응용 프로그램 요구 사항에 맞게 설계된 전용 서버에 있습니다. 응용 프로그램과 각 특성은 아래 표를 참조하십시오.

응용 프로그램 설명	특성
응용 프로그램 서버	CPU 2개 이상으로 확장할 수 없음
응용 프로그램 서버의 데이터베이스 인스턴스	트랜잭션 처리량이 많음
테스트 및 개발 환경의 응용 프로그램 서버	GUI 기반으로 테스트되지 않은 코드 실행 포함
트랜잭션 처리 서버	주요 문제는 응답 시간임
독립형 데이터베이스 인스턴스	많은 수의 트랜잭션을 처리하고 여러 시간대를 제공함

통합 구성

다음 구성을 사용하여 리소스 풀이 있고 동적 리소스 풀 기능을 지원하는 단일 시스템에 응용 프로그램을 통합합니다.

- 응용 프로그램 서버에 두 개의 CPU 프로세서 세트가 있습니다.
- 응용 프로그램 서버의 데이터베이스 인스턴스와 독립형 데이터베이스 인스턴스가 네 개 이상의 CPU로 구성된 단일 프로세서 세트에 통합됩니다. 독립형 데이터베이스 인스턴스는 해당 리소스의 75%가 보장됩니다.
- 테스트 및 개발 응용 프로그램 서버에는 UI 응답성을 보장하기 위해 IA 예약 클래스가 필요합니다. 잘못된 코드 빌드로 인한 영향을 줄이기 위해 메모리 제한이 설정됩니다.
- 트랜잭션 처리 서버에는 응답 지연을 최소화하기 위해 CPU 2개 이상의 전용 프로세서 세트가 지정됩니다.

이러한 구성은 각 리소스 집합에서 프로세서 주기를 실행 및 사용하는 알려진 응용 프로그램에 적용됩니다. 따라서 리소스가 필요한 세트로 프로세서 리소스를 전송할 수 있도록 하는 제약 조건을 설정할 수 있습니다.

- 사용률이 높은 리소스 집합이 사용률이 낮은 집합보다 더 많은 리소스를 할당받을 수 있도록 `wt-load` 목표가 설정됩니다.
- `locality` 목표는 프로세서 지역성을 최대화하기 위해 사용되는 `tight`로 설정됩니다.

사용률이 리소스 집합의 80%를 초과하는 것을 방지하기 위해 추가 제약 조건이 적용됩니다. 이러한 제약 조건은 응용 프로그램이 필요로 하는 리소스에 액세스할 수 있도록 보장합니다. 또한 트랜잭션 프로세서 세트의 경우 사용률을 80% 아래로 유지하는 목표가 지정된 다른 목표보다 두 배로 중요합니다. 이러한 중요도는 구성에서 정의됩니다.

구성 만들기

`/etc/project` 데이터베이스 파일을 편집합니다. 필요한 리소스 제어를 구현하고 사용자를 리소스 풀에 매핑하기 위한 항목을 추가한 다음 파일을 확인합니다.

```
# cat /etc/project
.
.
.
user.app_server:2001:Production Application Server:::project.pool=appserver_pool
user.app_db:2002:App Server DB:::project.pool=db_pool;project.cpu-shares=(privileged,1,deny)
development:2003:Test and development::staff:project.pool=dev_pool;
process.max-address-space=(privileged,536870912,deny)      keep with previous line
user.tp_engine:2004:Transaction Engine:::project.pool=tp_pool
user.geo_db:2005:EDI DB:::project.pool=db_pool;project.cpu-shares=(privileged,3,deny)
.
.
.
```

주 – 이 프로젝트에 대한 액세스는 사용자의 GID(그룹 ID)를 기반으로 하므로 개발 팀은 개발 프로젝트 내에서 작업을 실행해야 합니다.

필요한 리소스 풀을 구성하기 위해 사용할 `pool.host`라는 입력 파일을 만듭니다. 파일을 확인합니다.

```
# cat pool.host
create system host
create pset dev_pset (uint pset.min = 0; uint pset.max = 2)
create pset tp_pset (uint pset.min = 2; uint pset.max=8)
create pset db_pset (uint pset.min = 4; uint pset.max = 6)
create pset app_pset (uint pset.min = 1; uint pset.max = 2)
create pool dev_pool (string pool.scheduler="IA")
create pool appserver_pool (string pool.scheduler="TS")
create pool db_pool (string pool.scheduler="FSS")
create pool tp_pool (string pool.scheduler="TS")
associate pool dev_pool (pset dev_pset)
associate pool appserver_pool (pset app_pset)
associate pool db_pool (pset db_pset)
associate pool tp_pool (pset tp_pset)
modify system tester (string system.poold.objectives="wt-load")
modify pset dev_pset (string pset.poold.objectives="locality tight; utilization < 80")
modify pset tp_pset (string pset.poold.objectives="locality tight; 2: utilization < 80")
modify pset db_pset (string pset.poold.objectives="locality tight;utilization < 80")
modify pset app_pset (string pset.poold.objectives="locality tight; utilization < 80")
```

`pool.host` 입력 파일을 사용하여 구성을 업데이트합니다.

```
# poolcfg -f pool.host
```

구성을 활성화합니다.

```
# pooladm -c
```

이제 시스템에서 프레임워크가 작동합니다.

DRP를 사용으로 설정합니다.

```
# svcadm enable pools/dynamic:default
```

구성 보기

시스템에서 생성된 기본 요소가 포함되어 있는 프레임워크 구성을 보려면 다음을 입력합니다.

```
# pooladm
system host
        string  system.comment
```

```
int     system.version 1
boolean system.bind-default true
int     system.poold.pid 177916
string  system.poold.objectives wt-load

pool dev_pool
        int     pool.sys_id 125
        boolean pool.default false
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler IA
        pset    dev_pset

pool appserver_pool
        int     pool.sys_id 124
        boolean pool.default false
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler TS
        pset    app_pset

pool db_pool
        int     pool.sys_id 123
        boolean pool.default false
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler FSS
        pset    db_pset

pool tp_pool
        int     pool.sys_id 122
        boolean pool.default false
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler TS
        pset    tp_pset

pool pool_default
        int     pool.sys_id 0
        boolean pool.default true
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler TS
        pset    pset_default

pset dev_pset
        int     pset.sys_id 4
        string  pset.units population
        boolean pset.default false
        uint    pset.min 0
        uint    pset.max 2
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
```

```
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80

pset tp_pset
        int     pset.sys_id 3
        string  pset.units population
        boolean pset.default false
        uint    pset.min 2
        uint    pset.max 8
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; 2: utilization < 80

        cpu
                int     cpu.sys_id 1
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line

pset db_pset
        int     pset.sys_id 2
        string  pset.units population
        boolean pset.default false
        uint    pset.min 4
        uint    pset.max 6
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80

        cpu
                int     cpu.sys_id 3
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 6
                string  cpu.comment
                string  cpu.status on-line
pset app_pset
        int     pset.sys_id 1
        string  pset.units population
```

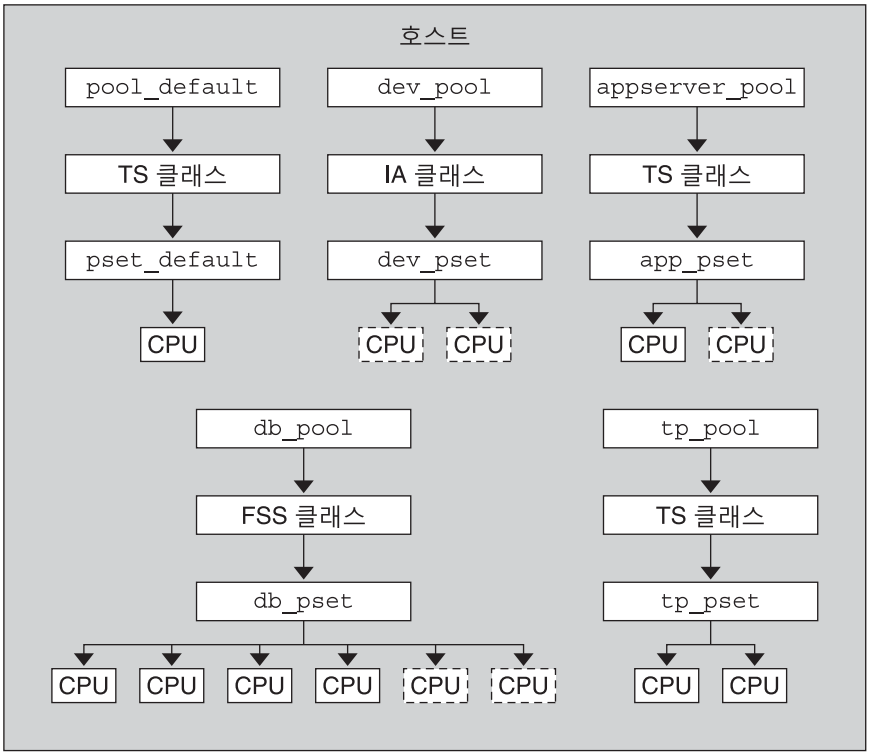
```
                boolean pset.default false
                uint    pset.min 1
                uint    pset.max 2
                string  pset.comment
                boolean pset.escapable false
                uint    pset.load 0
                uint    pset.size 0
                string  pset.poold.objectives locality tight; utilization < 80
                cpu
                        int     cpu.sys_id 7
                        string  cpu.comment
                        string  cpu.status on-line

        pset pset_default
                int     pset.sys_id -1
                string  pset.units population
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 4294967295
                string  pset.comment
                boolean pset.escapable false
                uint    pset.load 0
                uint    pset.size 0

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line
```

다음은 이 프레임워크를 그림으로 보여줍니다.

그림 14–1 서버 통합 구성



주 – db_pool 풀에서 독립형 데이터베이스 인스턴스에 CPU 리소스의 75%가 보장됩니다.

제 2 부

Oracle Solaris Zones

이 부분에서는 운영 체제 서비스를 가상화하여 응용 프로그램 실행을 위한 격리된 환경을 만드는 방법을 제공하는 Oracle Solaris Zones 소프트웨어 분할 기술에 대해 설명합니다. 이렇게 격리하면 하나의 영역에서 실행되는 프로세스가 다른 영역에서 실행되는 프로세스를 모니터하거나 영향을 미치지 않도록 방지할 수 있습니다.

◆ ◆ ◆ 15 장

15 Oracle Solaris Zones 소개

Oracle Solaris 운영 체제의 Oracle Solaris Zones 기능은 시스템에서 응용 프로그램을 실행할 분리된 환경을 제공합니다.

이 장에서는 영역에 대해 개괄적으로 설명합니다.

이 장에서는 다음과 같은 일반적인 영역의 내용을 다룹니다.

시스템에서 영역을 만들 준비가 되었으면 16 장, "비전역 영역 구성(개요)"를 참조하십시오.

주 – Oracle Solaris 10 Zones에 대한 자세한 내용은 제3부를 참조하십시오.

Oracle Solaris Trusted Extensions 시스템에서의 영역 사용에 대한 자세한 내용은 **Trusted Extensions 구성 및 관리**의 13 장, "Trusted Extensions에서 영역 관리(작업)"를 참조하십시오.

영역 개요

Oracle Solaris 영역 분할 기술은 운영 체제 서비스를 가상화하고 실행 중인 응용 프로그램을 위해 안전한 격리 환경을 제공하는 데 사용됩니다. **영역**은 Oracle Solaris 운영 체제의 단일 인스턴스 내에서 만들어진 가상화된 운영 체제 환경입니다.

가상화의 목표는 개별 데이터 센터 구성 요소 관리에서 리소스 풀 관리로 이동하는 것입니다. 서버 가상화를 성공적으로 수행하면 서버 활용을 개선하고 서버 자산을 더욱 효율적으로 사용할 수 있습니다. 서버 가상화는 개별 시스템의 격리를 유지 관리하는 성공적인 서버 통합 프로젝트에도 중요합니다.

가상화는 단일 시스템에서 여러 호스트와 서비스를 통합해야 하는 필요에 따라 수행됩니다. 가상화는 하드웨어, 기반구조 및 관리를 공유하여 비용을 축소합니다. 혜택은 다음과 같습니다.

- 하드웨어 활용 증가
- 리소스 할당의 유연성 향상
- 전원 요구 사항 축소
- 관리 비용 감소
- 소유 비용 감소
- 시스템에서 응용 프로그램 간에 관리 및 리소스 경계 설정

영역을 만들면 프로세스를 나머지 시스템으로부터 분리시키는 응용 프로그램 실행 환경이 만들어집니다. 이러한 분리는 하나의 영역에서 실행되는 프로세스가 다른 영역에서 실행되는 프로세스를 모니터링하거나 영향을 미치는 것을 방지합니다. 루트 자격 증명으로 실행되는 프로세스라도 다른 영역의 작업을 보거나 영향을 미칠 수 없습니다. Oracle Solaris Zones에서는 하드웨어 리소스를 동시에 공유하면서 서버 배포 모델당 하나의 응용 프로그램을 유지할 수 있습니다.

또한, 영역은 응용 프로그램과 응용 프로그램이 배포된 시스템의 물리적 속성을 분리하는 추상 계층을 제공합니다. 이러한 속성의 예로는 물리적 장치 경로가 있습니다.

영역은 Oracle Solaris 10 이상의 Oracle Solaris 릴리스를 실행 중인 모든 시스템에서 영역을 사용할 수 있습니다. 시스템에서 만들 수 있는 최대 영역 수는 8192입니다. 단일 시스템에 효과적으로 호스팅할 수 있는 영역 수는 시스템의 모든 영역과 크기에서 실행 중인 응용 프로그램 소프트웨어의 전체 리소스 요구 사항에 따라 결정됩니다.

이 개념은 17 장, “비전역 영역 계획 및 구성(작업)”에서 다룹니다.

이 Oracle Solaris Zones 릴리스 정보

이 절에서는 Oracle Solaris 10 릴리스 이후 새 기능과 영역 변경 사항에 대해 간략히 설명합니다.

Oracle Solaris 11 릴리스에서 기본 비전역 영역은 `solaris`이며, 이 설명서와 `solaris` (5) 매뉴얼 페이지에 설명되어 있습니다.

`solaris` 비전역 영역은 Oracle Solaris 11 릴리스에서 지원 플랫폼으로 정의한 모든 `sun4u`, `sun4v` 및 x86 구조 시스템에서 지원됩니다.

Oracle Solaris 릴리스와 시스템 구조를 확인하려면 다음을 입력합니다.

```
#uname -r -m
```

`solaris` 영역에서는 `brands(5)` 매뉴얼 페이지에 설명된 브랜드 영역 프레임워크를 사용하여 전역 영역과 동일한 소프트웨어로 설치된 영역을 실행합니다. 이 시스템 소프트웨어는 `solaris` 비전역 영역 사용 시 전역 영역과 동기화되어야 합니다. 영역 내 시스템 소프트웨어 패키지는 IPS(이미지 패키징 시스템)를 사용하여 관리됩니다. IPS는 Oracle Solaris 11 릴리스의 패키징 시스템이므로, `solaris` 영역은 이 모델을 사용합니다.

Oracle Solaris 11 Express에서 만들어진 기본 `ipkg` 영역은 `solaris` 영역으로 매핑됩니다. 189 페이지 “ipkg 영역을 solaris 영역으로 변환에 대한 정보”를 참조하십시오.

AI(자동 설치) 매니페스트에 지정된 각각의 비전역 영역이 클라이언트 설치의 일부로 설치 및 구성됩니다. 비전역 영역은 전역 영역이 설치된 후에 처음 재부트 시 설치 및 구성됩니다. 시스템이 처음 부트할 때 영역 자체 어셈블리 SMF 서비스인 `svc:/system/zones-install:default`가 전역 영역 AI 매니페스트에 정의된 각각의 비전역 영역을 구성 및 설치합니다. 자세한 내용은 **Oracle Solaris 11 시스템**를 참조하십시오. 또한 설치된 Oracle Solaris 시스템에서 영역을 수동으로 구성 및 설치할 수도 있습니다.

기본적으로 영역은 배타적 IP 유형으로 만들어집니다. 네트워크 구성을 지정하지 않은 경우 `anet` 리소스를 통해 영역 구성에 VNIC이 자동으로 포함됩니다. 자세한 내용은 210 페이지 “영역 네트워크 인터페이스”를 참조하십시오.

이번 릴리스에서는 322 페이지 “영역에서 NFS 서버 실행”에 설명된 것처럼 `solaris` 영역이 NFS 서버일 수 있습니다.

dry-run이라고도 하는 테스트 실행, `zoneadm attach -n`은 `zonecfg` 검증을 제공하지만 패키지 내용 검증은 수행하지 않습니다.

파일을 인수로 가져오는 모든 `zoneadm` 옵션에는 절대 경로가 필요합니다.

Oracle Solaris 10 Zones는 Oracle Solaris 11에서 Oracle Solaris 10 환경을 제공합니다. Oracle Solaris 10 시스템 또는 영역을 Oracle Solaris 11 시스템의 `solaris10` 영역으로 마이그레이션할 수 있습니다.

zonep2vchk 도구는 Oracle Solaris 11 시스템이나 Oracle Solaris 10 시스템을 Oracle Solaris 11 릴리스의 영역으로 마이그레이션하는 데 영향을 줄 수 있는 네트워킹 문제를 비롯한 다양한 문제를 식별합니다. zonep2vchk 도구는 마이그레이션이 시작되기 전에 소스 시스템에서 실행됩니다. 또한 이 도구는 대상 시스템에서 사용할 zonecfg 스크립트를 출력합니다. 스크립트는 소스 시스템 구성과 일치하는 영역을 만듭니다. 자세한 내용은 22 장, “영역 마이그레이션 및 zonep2vchk 도구 정보”를 참조하십시오.

Oracle Solaris 10 릴리스의 native 영역과 solaris 영역은 다음과 같은 차이점이 있습니다.

- Oracle Solaris 10 시스템의 기본값인 native 브랜드 대신에 solaris 브랜드가 기본값입니다.
- solaris 영역은 전체 루트 형식뿐입니다.

  Oracle Solaris 10에서 사용할 수 있는 native 영역의 희소 루트 형식은 SVR4 패키지 관리 시스템을 사용하며, IPS는 이 프레임워크를 사용하지 않습니다. 희소 루트 형식과 비슷한 읽기 전용 루트 영역 구성을 사용할 수 있습니다.
- 이 번 릴리스의 영역은 다음 부분에서 Oracle Solaris 10 릴리스와는 다른 소프트웨어 관리 관련 기능을 제공합니다.
  - IPS와 SVR4 패키징
  - 설치, 분리, 연결 및 물리-가상 기능
  - 비전역 루트 영역은 ZFS 데이터 세트입니다.

    전역 영역에 설치된 패키지는 더 이상 현재와 미래의 모든 영역에 설치되지 않습니다. 일반적으로 IPS 및 SVR4 패키징에 대해 전역 영역의 패키지 컨텐츠가 더 이상 각 영역의 패키지 컨텐츠를 좌우하지 않습니다.
- 비전역 영역은 부트 환경을 사용합니다. 영역은 ZFS BE(부트 환경)를 관리하기 위한 사용자 인터페이스 명령인 beadm과 통합됩니다. 시스템에서 영역 BE를 보려면 다음을 입력합니다.

  ```
  # zoneadm list zbe
  global
  test2
  ```

  전역 영역에서처럼 pkg 업데이트를 위해 영역 내부에서 beadm 명령이 지원됩니다. beadm 명령은 영역과 연결된 비활성 영역 BE를 삭제할 수 있습니다. beadm(1M) 매뉴얼 페이지를 참조하십시오.
- 영역을 설치하는 동안 모든 사용 가능 IPS 패키지 저장소에 액세스할 수 있어야 합니다. 자세한 내용은 269 페이지 “구성된 영역 설치 방법”을 참조하십시오.
- 영역 소프트웨어가 최소화되어 시작됩니다. 영역에 필요한 추가 패키지를 추가해야 합니다. 자세한 내용은 solaris publisher (http://pkg.oracle.com/solaris/release/)를 참조하십시오.

영역은 다음과 같은 Oracle Solaris 11 제품과 기능을 사용할 수 있습니다.

- Oracle Solaris ZFS 암호화
- 네트워크 가상화 및 QoS

- CIFS 및 NFS

다음 기능은 비전역 영역에서 구성할 수 없습니다.

- 공유 IP 영역에서 DHCP 주소 지정
- `ndmpd`
- SMB 서버
- SSL 프록시 서버
- `zpool` 명령을 통한 ZFS 풀 관리

읽기 전용 solaris 비전역 영역

변경할 수 없는 영역은 읽기 전용 루트가 있는 영역입니다. 읽기 전용 영역은 `file-mac-profile` 등록 정보를 설정하여 구성할 수 있습니다. 몇 가지 구성을 사용할 수 있습니다. 읽기 전용 영역 루트는 보안 런타임 경계를 확장합니다.

`zonecfg add dataset`을 사용하여 추가 데이터 집합이 지정되는 영역에는 여전히 이러한 데이터 집합에 대한 모든 권한이 부여됩니다. `zonecfg add fs`를 사용하여 추가 파일 시스템이 지정되는 영역에는 파일 시스템이 읽기 전용으로 설정된 경우가 아니면 이러한 파일 시스템에 대한 모든 권한이 부여됩니다.

자세한 내용은 27 장, “변경할 수 없는 영역 구성 및 관리”를 참조하십시오.

ipkg 영역을 solaris 영역으로 변환에 대한 정보

Oracle Solaris 11 Express 고객을 지원하기 위해 `ipkg` 영역으로 구성된 영역이 `solaris` 영역으로 변환되고 `pkg` 업데이트 시 `solaris`로 보고되거나 Oracle Solaris 11에 `zoneadm attach`로 보고됩니다. `ipkg` 이름은 영역을 구성할 때 사용된 경우 `solaris` 이름으로 매핑됩니다. Oracle Solaris 11 Express 호스트에서 내보낸 `zonecfg` 파일의 가져오기가 지원됩니다.

`zonecfg info` 또는 `zoneadm list -v` 등의 명령 출력은 Oracle Solaris 11 시스템에서 기본 영역의 `solaris` 브랜드를 표시합니다.

브랜드 영역 정보

기본적으로 시스템의 비전역 영역은 전역 영역과 동일한 운영 체제 소프트웨어를 실행합니다. Oracle Solaris 운영 체제의 브랜드 영역(BrandZ) 기능은 Oracle Solaris Zones를 단순히 확장한 것입니다. BrandZ 프레임워크는 전역 영역과는 다른 운영 환경을 포함하는 비전역 영역 브랜드 영역을 만드는 데 사용됩니다. 브랜드 영역은 Oracle Solaris 운영 체제에서 응용 프로그램을 실행하는 데 사용됩니다. BrandZ 프레임워크는 다양한 방식으로 Oracle Solaris Zones 기반구조를 확장합니다. 이러한 확장은 복잡할 수도

있고(예: 영역 내 다양한 운영 체제 환경을 실행할 수 있는 기능 제공) 간단할 수도 있습니다(예: 기본 영역 명령을 향상시켜 새 기능 제공). 예를 들어 Oracle Solaris 10 Zones는 Oracle Solaris 10 운영 체제를 에뮬레이트할 수 있는 브랜드 비전역 영역입니다. 전역 영역과 동일한 운영 체제를 공유하는 기본 영역조차 **브랜드**로 구성됩니다.

브랜드는 영역 안에 설치할 수 있는 운영 환경을 정의하며, 영역에 설치된 소프트웨어가 올바르게 작동하도록 영역 내에서 시스템의 작동 방식을 결정합니다. 또한, 영역의 브랜드는 응용 프로그램 시작 시 올바른 응용 프로그램 유형을 식별하는 데 사용됩니다. 모든 브랜드 영역 관리는 표준 영역 구조에 대한 확장을 통해 수행됩니다. 대부분의 관리 절차는 모든 영역에 대해 동일합니다.

정의된 파일 시스템과 권한 등 기본적으로 구성에 포함된 리소스는 브랜드 관련 문서에 설명되어 있습니다.

BrandZ는 다음과 같은 방식으로 영역 도구를 확장합니다.

- `zonecfg` 명령은 영역이 구성될 때 영역의 브랜드 유형을 설정하는 데 사용됩니다.
- `zoneadm` 명령은 영역의 브랜드 유형을 보고하고 영역을 관리하는 데 사용됩니다.

라벨이 사용으로 설정된 Oracle Solaris Trusted Extensions 시스템에서 브랜드 영역을 구성 및 설치할 수 있다고 해도, 부트되는 브랜드가 인증된 시스템 구성의 `labeled` 브랜드가 **아니면** 이 시스템 구성에서 브랜드 영역을 부트할 수 없습니다.

영역의 브랜드를 **구성됨** 상태로 변경할 수 있습니다. 브랜드 영역이 **설치**되고 나면 브랜드를 변경하거나 제거할 수 없습니다.

주의 – 기존 Oracle Solaris 10 시스템을 Oracle Solaris 11 릴리스를 실행하는 `solaris10` 브랜드 영역으로 마이그레이션하려는 경우 먼저 기존 영역을 대상 시스템으로 마이그레이션해야 합니다. 영역이 중첩되지 않으므로 시스템 마이그레이션 프로세스에서 기존 영역이 사용할 수 없는 상태로 표시됩니다. 자세한 내용은 제3부를 참조하십시오.

브랜드 영역에서 실행 중인 프로세스

브랜드 영역은 커널에 브랜드 영역에서 실행 중인 프로세스에만 적용되는 삽입점을 제공합니다.

- 이러한 지점은 `syscall` 경로, 프로세스 로딩 경로 및 스레드 생성 경로 등에서 발견됩니다.
- 이러한 각각의 지점에서 브랜드는 표준 Oracle Solaris 동작을 보완하거나 대체하도록 선택할 수 있습니다.

또한 브랜드는 `librtld_db`에 대한 플러그인 라이브러리를 제공할 수도 있습니다. 이 플러그인 라이브러리를 사용하면 `mdb(1)`에 설명된 디버거나 `dtrace(1M)`에 설명된 DTrace와 같은 Oracle Solaris 도구를 통해 브랜드 영역 내부에서 실행 중인 프로세스의 기호 정보에 액세스할 수 있습니다.

영역은 정적으로 링크된 바이너리를 지원하지 않습니다.

이 릴리스에 사용할 수 있는 비전역 영역

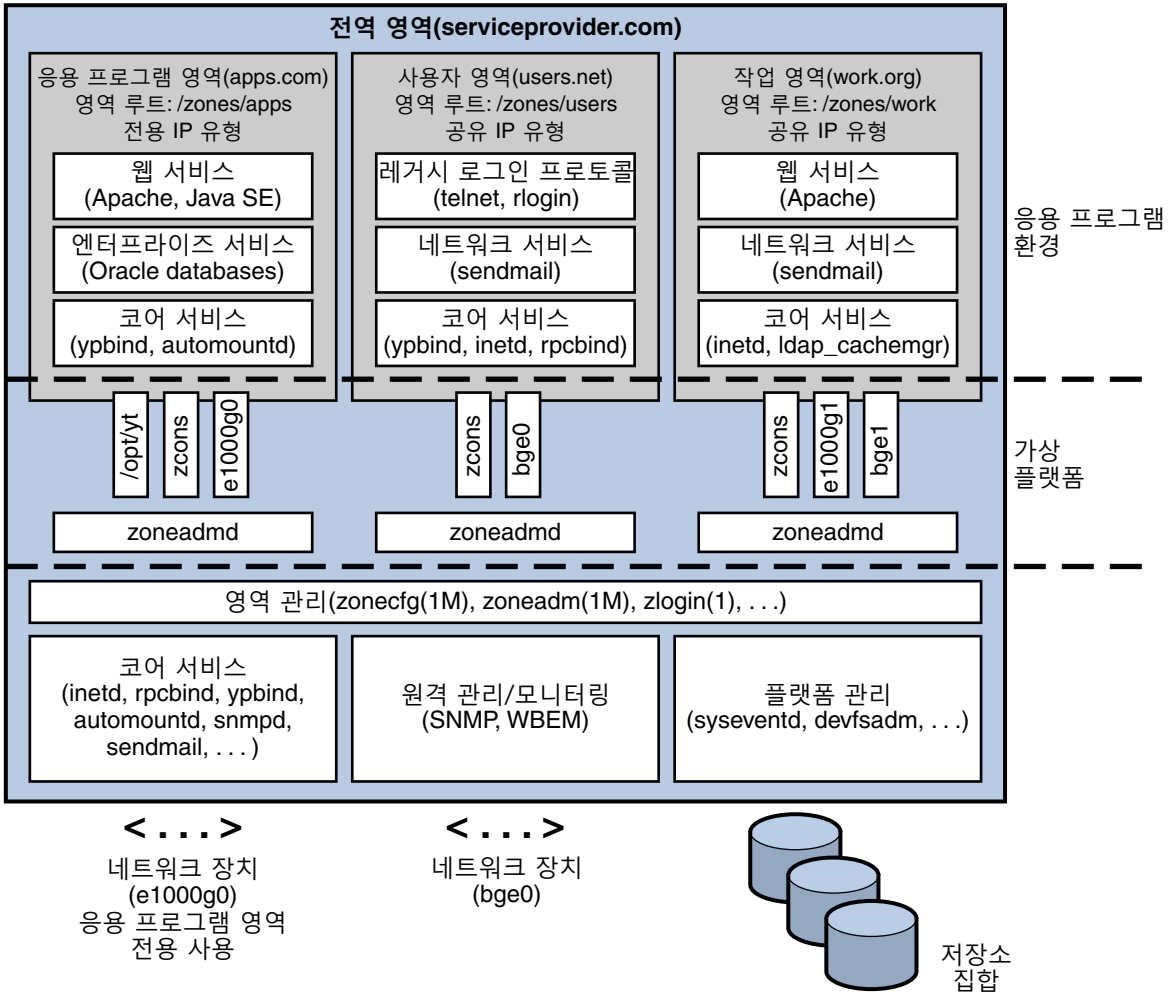
기본 Oracle Solaris Zone 외에도, Oracle Solaris 10 Zones(`solaris10` 브랜드 영역) 제품이 이 릴리스에 포함되어 있습니다. 자세한 내용은 제3부를 참조하십시오.

영역을 사용해야 하는 경우

영역은 단일 서버에 여러 응용 프로그램을 통합하는 환경에 이상적입니다. 다양한 기기를 관리하는 데 드는 비용과 복잡성 문제로 인해 더 크고 확장성이 더 뛰어난 서버에 여러 응용 프로그램을 통합하는 것이 유리합니다.

다음 그림은 세 개 영역으로 구성된 시스템을 보여 줍니다. `apps`, `users` 및 `work`라는 각각의 영역이 샘플 통합 환경에서 다른 영역의 작업 부하와 무관하게 작업 부하를 실행 중입니다. 이 예에서는 서로 다른 버전의 동일한 응용 프로그램을 다른 영역에 부정적인 영향을 미치지 않으면서 실행하여 통합 요구 사항을 충족할 수 있음을 보여 줍니다. 각 영역에서는 일련의 사용자 정의된 서비스가 제공될 수 있습니다.

그림 15–1 영역 서버 통합 예



영역을 사용하면 시스템에서 리소스를 보다 효율적으로 활용할 수 있습니다. 동적 리소스 재할당을 통해 필요에 따라 사용되지 않는 리소스를 이동할 수 있습니다. 결함 및 보안 분리란 잘못 작동하는 응용 프로그램에 전용 시스템 및 활용률이 낮은 시스템이 필요하지 않음을 의미합니다. 영역을 사용하면 이러한 응용 프로그램을 다른 응용 프로그램과 통합할 수 있습니다.

영역을 사용하면 전체 시스템 보안을 유지하면서 특정 관리 기능을 위임할 수 있습니다.

영역 작동 방법

비전역 영역은 하나의 상자로 생각할 수 있습니다. 하나 이상의 응용 프로그램이 시스템의 나머지와 상호 작용 없이 이 상자에서 실행될 수 있습니다. 영역은 소프트웨어에 의해 정의된 유연한 경계를 사용하여 소프트웨어 응용 프로그램이나 서비스를 분리합니다. 그러면 Oracle Solaris 운영 체제의 동일 인스턴스에서 실행 중인 응용 프로그램을 다른 응용 프로그램과 독립적으로 관리할 수 있습니다. 따라서 서로 다른 버전의 동일 응용 프로그램을 서로 다른 영역에서 실행하며 구성 요구 사항을 충족할 수 있습니다.

영역에 지정된 프로세스는 동일 영역에 지정된 다른 프로세스와 직접 통신하고 이러한 프로세스를 조작하며 모니터링할 수 있습니다. 이 프로세스는 시스템의 다른 영역에 지정된 프로세스와 함께 또는 영역에 지정되지 않은 프로세스와 함께 이러한 기능을 수행할 수 있습니다. 다양한 영역에 지정된 프로세스는 네트워크 API를 통해서만 통신할 수 있습니다.

영역에 고유의 배타적 IP 인스턴스가 지정되었는지 아니면 전역 영역과 IP 계층 구성 및 상태를 공유하는지에 따라 두 가지 방법으로 IP 네트워킹을 구성할 수 있습니다. 기본 유형은 배타적 IP입니다. 영역의 IP 유형에 대한 자세한 내용은 210 페이지 “영역 네트워크 인터페이스”를 참조하십시오. 구성 정보는 243 페이지 “영역 구성 방법”을 참조하십시오.

모든 Oracle Solaris 시스템에는 **전역 영역**이 있습니다. 전역 영역에는 두 가지 기능이 있습니다. 전역 영역은 시스템의 기본 영역이며 또한 시스템 전체의 관리 제어용으로 사용되는 영역입니다. **비전역** 영역(간단히 영역이라고 함)이 없는 경우 전역 영역에서 실행되는 모든 프로세스는 **전역 관리자** 또는 영역 보안 프로파일을 사용하는 사용자가 만듭니다.

전역 영역은 비전역 영역을 구성, 설치, 관리 또는 제거할 수 있는 유일한 영역입니다. 오직 전역 영역만 시스템 하드웨어에서 부트할 수 있습니다. 물리적 장치, 공유 IP 영역의 경로 지정 또는 DR(동적 재구성)과 같은 시스템 기반구조의 관리는 전역 영역에서만 가능합니다. 전역 영역에서 적절한 권한으로 실행되는 프로세스는 다른 영역에 연결된 객체에 액세스할 수 있습니다.

전역 영역의 권한이 없는 프로세스는 비전역 영역의 권한이 부여된 프로세스에 허용되지 않는 작업을 수행할 수 있습니다. 예를 들면 전역 영역의 사용자는 시스템의 모든 프로세스에 대한 정보를 볼 수 있습니다. 이 기능이 사이트에 문제를 야기시키는 경우에는 전역 영역에 대한 액세스를 제한할 수 있습니다.

전역 영역을 비롯한 각각의 영역에는 영역 이름이 지정됩니다. 전역 영역에는 항상 `global` 이름이 할당됩니다. 각 영역에도 고유한 숫자 식별자가 지정되는데, 이 식별자는 영역이 부트될 때 시스템에서 지정됩니다. 전역 영역은 항상 ID `0`으로 매핑됩니다. 영역 이름과 숫자 ID는 219 페이지 “`zonecfg` 명령 사용”에서 설명합니다.

각 영역에는 영역 이름과 완전히 독립된 노드 이름도 제공됩니다. 노드 이름은 영역의 관리자가 지정합니다. 자세한 내용은 321 페이지 “비전역 영역 노드 이름”을 참조하십시오.

각 영역에는 전역 영역의 루트 디렉토리를 기준으로, 해당 루트 디렉토리에 대한 경로가 있습니다. 자세한 내용은 219 페이지 “zonecfg 명령 사용”을 참조하십시오.

비전역 영역의 예약 클래스는 기본적으로 시스템의 예약 클래스로 설정됩니다. 영역의 예약 클래스를 설정하는 데 사용되는 방법에 대한 설명은 209 페이지 “예약 클래스”를 참조하십시오.

기능별 영역 요약

다음 표에는 전역 및 비전역 영역의 특성이 요약되어 있습니다.

영역 유형	특성
전역	■ 시스템에서 ID `0`을 지정합니다. ■ 부트 가능하며 시스템에서 실행 중인 Oracle Solaris의 단일 인스턴스를 제공합니다. ■ Oracle Solaris 시스템 소프트웨어 패키지의 전체 설치를 포함합니다. ■ 추가 소프트웨어 패키지 또는 패키지를 통해 설치되지 않은 추가 소프트웨어, 디렉토리, 파일 및 기타 데이터를 포함할 수 있습니다. ■ 전역 영역에 설치된 모든 소프트웨어 구성 요소에 대한 정보를 포함하는 완전하고 일관된 전체 제품 데이터베이스를 제공합니다. ■ 전역 영역 호스트 이름 및 파일 시스템 테이블 등 전역 영역에만 국한된 구성 정보를 보유합니다. ■ 모든 장치 및 모든 파일 시스템을 인식하는 유일한 영역입니다. ■ 비전역 영역 존재 여부와 구성을 알고 있는 유일한 영역입니다. ■ 비전역 영역을 구성, 설치, 관리 또는 제거할 수 있는 유일한 영역입니다.

영역 유형	특성
비전역	■ 영역이 부트될 때 시스템에서 영역 ID를 지정합니다. ■ 전역 영역에서 부트된 Oracle Solaris 커널 아래에서 작업을 공유합니다. ■ 전체 Oracle Solaris 운영 체제 소프트웨어 패키지 중 설치된 일부 소프트웨어를 포함합니다. ■ 추가로 설치된 소프트웨어 패키지를 포함할 수 있습니다. ■ 패키지를 통해 설치되지 않는 비전역 영역에 생성된 추가 소프트웨어, 디렉토리, 파일 및 기타 데이터를 포함할 수 있습니다. ■ 영역에 설치된 모든 소프트웨어 구성 요소에 대한 정보를 포함하는 완전하고 일관된 전체 제품 데이터베이스가 있습니다. ■ 다른 영역의 존재 여부를 인식하지 못합니다. ■ 자체 영역을 비롯하여 다른 영역을 설치, 관리 또는 제거할 수 없습니다. ■ 비전역 영역 호스트 이름 및 파일 시스템 테이블 등 비전역 영역에만 국한된 구성 정보를 보유합니다. ■ 고유의 표준 시간대 설정이 있을 수 있습니다.

비전역 영역이 관리되는 방식

영역 관리자에게는 수퍼 유저 권한 또는 이에 상응하는 관리 권한이 부여됩니다. 전역 영역에 로그인한 경우 전역 관리자는 시스템 전체를 모니터링하고 제어할 수 있습니다.

비전역 영역은 **영역 관리자**가 관리할 수 있습니다. 전역 관리자는 207 페이지 “admin 리소스”에 설명된 대로 영역 관리자에게 필수 권한을 지정합니다. 영역 관리자의 권한은 특정 비전역 영역으로 제한됩니다.

비전역 영역이 생성되는 방식

AI(자동 설치) 클라이언트 설치의 일부로서 비전역 영역의 구성 및 설치를 지정할 수 있습니다. 자세한 내용은 **Oracle Solaris 11 시스템**를 참조하십시오.

Oracle Solaris 11 시스템에서 영역을 만들려면 영역 관리자는 `zonecfg` 명령을 사용하여 영역의 가상 플랫폼과 응용 프로그램 환경에 대한 다양한 매개 변수를 지정함으로써 영역을 구성할 수 있습니다. 그러고 나면 전역 관리자가 영역을 설치합니다. 전역 관리자는 영역 관리 명령인 `zoneadm`을 사용하여 패키지 레벨의 소프트웨어를 영역에 대해 설정된 파일 시스템 계층에 설치합니다. `zoneadm` 명령은 영역을 부트하는 데 사용됩니다. 그러고 나면 전역 관리자나 권한이 부여된 사용자가 `zlogin` 명령을 사용하여 설치된 영역에 로그인할 수 있습니다. RBAC(역할 기반 액세스 제어)를 사용 중인 경우 영역 관리자에게 `solaris.zone.manage/`*zonename* 권한이 부여되어야 합니다.

영역 구성에 대한 자세한 내용은 16 장, “비전역 영역 구성(개요)”을 참조하십시오. 영역 설치에 대한 자세한 내용은 18 장, “비전역 영역, 설치, 종료, 정지 및 복제 정보(개요)”를 참조하십시오. 영역 로그인에 대한 자세한 내용은 20 장, “비전역 영역 로그인(개요)”을 참조하십시오.

비전역 영역 상태 모델

비전역 영역은 다음 여섯 가지 상태 중 하나일 수 있습니다.

구성됨	영역의 구성이 완료되고 안정된 저장소로 완결됩니다. 하지만 영역의 응용 프로그램 환경에서 초기 부트 후에 지정해야 하는 요소는 아직 존재하지 않습니다.
불완전	설치 또는 제거 작업 중에 `zoneadm`은 대상 영역의 상태를 불완전으로 설정합니다. 작업이 완료되면 상태가 올바름 상태로 설정됩니다. `zoneadm`의 하위 명령인 `mark`를 사용하여 손상된 설치 영역을 불완전 상태로 표시할 수 있습니다. 불완전 상태의 영역이 `zoneadm list -iv`의 출력에 표시됩니다.
설치됨	영역의 구성이 시스템에서 인스턴스화됩니다. `zoneadm` 명령은 지정된 Oracle Solaris 시스템에서 구성을 정상적으로 사용할 수 있음을 확인하는 데 사용됩니다. 패키지는 영역의 루트 경로 아래에 설치됩니다. 이 상태에서는 영역에 연결된 가상 플랫폼이 없습니다.
준비	영역의 가상 플랫폼이 설정됩니다. 커널이 `zsched` 프로세스를 만들고, 네트워크 인터페이스가 설정되어 영역에서 사용할 수 있게 되며, 파일 시스템이 마운트되고 장치가 구성됩니다. 고유의 영역 ID는 시스템에서 지정합니다. 이 단계에서는 영역과 연결된 프로세스가 시작되지 않았습니다.
실행 중	영역 응용 프로그램 환경과 연결된 사용자 프로세스가 실행 중입니다. 응용 프로그램 환경과 연결된 첫번째 사용자 프로세스(`init`)가 만들어지자 마자 영역이 실행 중 상태로 들어갑니다.
종료 및 작동 중지	이러한 상태는 영역이 정지되고 있는 동안 표시되는 중간 상태입니다. 하지만 어떤 이유로든 종료할 수 없는 영역은 이러한 상태 중 하나로 중지됩니다.

19 장, “비전역 영역 설치, 부트, 종료, 정지, 제거 및 복제(작업)” 및 `zoneadm`(1M) 매뉴얼 페이지에서는 `zoneadm` 명령을 사용하여 이러한 상태 간 전환을 시작하는 방법에 대해 설명합니다.

표 15–1 영역 상태에 영향을 주는 명령

현재 영역 상태	해당되는 명령

구성됨	`zonecfg -z` *zonename* `verify`
	`zonecfg -z` *zonename* `commit`
	`zonecfg -z` *zonename* `delete`
	`zoneadm -z` *zonename* `attach`
	`zoneadm -z` *zonename* `verify`
	`zoneadm -z` *zonename* `install`
	`zoneadm -z` *zonename* `clone`
	`zonecfg`를 사용하여 구성됨 또는 설치됨 상태에 있는 영역의 이름을 바꿀 수도 있습니다.

불완전	`zoneadm -z` *zonename* `uninstall`

설치됨	`zoneadm -z` *zonename* `ready`(옵션)
	`zoneadm -z` *zonename* `boot`
	`zoneadm -z` *zonename* `uninstall`은 시스템에서 지정된 영역의 구성을 제거합니다.
	`zoneadm -z` *zonename* `move` *path*
	`zoneadm -z` *zonename* `detach`
	`zonecfg -z` *zonename*을 사용하여 `attr`, `bootargs`, `capped-memory`, `dataset`, `capped-cpu`, `dedicated-cpu`, `device`, `fs`, `ip-type`, `limitpriv`, `net`, `rctl` 또는 `scheduling-class` 등록 정보를 추가하거나 제거할 수 있습니다. 또한 설치됨 상태에 있는 영역의 이름을 바꿀 수도 있습니다.

준비	`zoneadm -z` *zonename* `boot` `zoneadm halt`를 실행하고 시스템을 재부트하면 준비 상태의 영역이 설치됨 상태로 복원됩니다. `zonecfg -z` *zonename*을 사용하면 `attr`, `bootargs`, `capped-memory`, `dataset`, `capped-cpu`, `dedicated-cpu`, `device`, `fs`, `ip-type`, `limitpriv`, `net`, `rctl` 또는 `scheduling-class` 등록 정보를 추가하거나 제거할 수 있습니다.

표 15–1 영역 상태에 영향을 주는 명령 (**계속**)

현재 영역 상태	해당되는 명령

실행 중	`zlogin` *options* `zonename`
	
	`zoneadm -z` *zonename* `reboot`
	
	`zoneadm -z` *zonename* `halt`는 준비 상태의 영역을 설치됨 상태로 되돌립니다.
	
	`zoneadm halt`를 실행하고 시스템을 재부트하면 실행 중 상태의 영역이 설치됨 상태로 복원됩니다.
	
	`zoneadm - z shutdown`은 영역을 완전히 종료합니다.
	
	`zonecfg -z` *zonename*을 사용하면 `attr`, `bootargs`, `capped-memory`, `dataset`, `capped-cpu`, `dedicated-cpu`, `device`, `fs`, `ip-type`, `limitpriv`, `anet`, `net`, `rctl` 또는 `scheduling-class` 등록 정보를 추가하거나 제거할 수 있습니다. `zonepath` 리소스는 변경할 수 없습니다.

주 – `zonecfg`를 통해 변경된 매개 변수는 실행 중 영역에 영향을 주지 않습니다. 변경 내용을 적용하려면 영역을 재부트해야 합니다.

비전역 영역 특성

영역은 필요한 거의 모든 레벨의 세분성에서 분리를 제공합니다. 영역에는 전용 CPU, 실제 장치 또는 실제 메모리의 부분이 필요하지 않습니다. 이들 리소스는 단일 도메인 또는 시스템에서 실행 중인 여러 영역에 걸쳐 다중화되거나 운영 체제에서 사용 가능한 리소스 관리 기능을 사용하여 각 영역마다 할당할 수 있습니다.

각 영역에서는 일련의 사용자 정의된 서비스가 제공될 수 있습니다. 기본 프로세스 분리를 강제로 적용하기 위해 동일한 영역에 존재하는 프로세스만 확인하거나 신호를 보낼 수 있습니다. 영역 간의 기본 통신은 각 영역에 IP 네트워크 연결을 제공함으로써 이루어집니다. 하나의 영역에서 실행되는 응용 프로그램은 다른 영역의 네트워크 트래픽을 볼 수 없습니다. 이러한 분리는 각각의 패킷 스트림이 동일한 실제 인터페이스를 통과하는 경우에도 유지됩니다.

각 영역에는 파일 시스템 계층의 일부분이 부여됩니다. 각 영역은 해당 파일 시스템 계층의 하위 트리로 제한되므로 특정 영역에서 실행되는 작업 부하가 다른 영역에서 실행되는 다른 작업 부하의 디스크 데이터에 액세스할 수 없습니다.

이름 지정 서비스에서 사용되는 파일은 영역의 자체 루프 파일 시스템 보기에 상주합니다. 따라서 서로 다른 영역의 이름 지정 서비스는 서로 분리되며 각각 다르게 구성될 수 있습니다.

리소스 관리 기능을 비전역 영역과 함께 사용

리소스 관리 기능을 사용하는 경우 리소스 관리 컨트롤의 경계를 영역의 경계와 맞춰야 합니다. 이 정렬을 통해 더욱 완벽한 가상 시스템 모델이 구축되며, 이러한 가상 시스템에서는 이름 공간, 보안 분리 및 리소스 사용이 모두 제어됩니다.

영역과 함께 다양한 리소스 관리 기능의 사용에 대한 특별 요구 사항은 이 설명서에서 이 기능을 설명하는 개별 단원에서 설명합니다.

비전역 영역 모니터링

현재 실행 중인 영역의 CPU, 메모리 및 리소스 제어 활용률에 대한 정보는 353 페이지 "비전역 영역에서 zonestat 유틸리티 사용"을 참조하십시오. zonestat 유틸리티는 배타적 IP 영역의 네트워크 대역폭 활용률도 보고합니다. 하나의 배타적 IP 영역에는 하나 이상의 전용 데이터 링크와 고유한 IP 관련 상태가 지정됩니다.

비전역 영역에서 제공하는 기능

비전역 영역은 다음 기능을 제공합니다.

보안
: 프로세스가 전역 영역이 아닌 다른 영역에 배치되고 나면 해당 프로세스와 이후 하위 프로세스에서 영역을 변경할 수 없습니다.

  네트워크 서비스는 영역에서 실행할 수 있습니다. 영역에서 네트워크 서비스를 실행하여 보안 위반 시 발생할 수 있는 손상을 제한할 수 있습니다. 영역 내에서 실행 중인 소프트웨어의 보안 결점을 악용하는 침입자는 해당 영역 내에서 가능한 특정 동작만 제한적으로 수행할 수 있습니다. 영역 내에서 사용할 수 있는 권한은 시스템 전체에서 사용할 수 있는 권한 중 일부입니다.

분리
: 여러 응용 프로그램이 서로 다른 트러스트 도메인에서 작동하거나, 전역 리소스에 대한 배타적 액세스를 필요로 하거나, 또는 전역 구성과 관련된 어려움이 있는 경우라도, 영역에서는 동일 시스템에 여러 응용 프로그램을 배포할 수 있습니다. 예를 들면 각 영역에 연결된 고유한 IP 주소를 사용하거나 와일드카드 주소를 사용하여 동일 시스템에 있는 서로 다른 공유 IP 영역에서 실행 중인 여러 응용 프로그램을 동일 네트워크 포트로 바인딩할 수 있습니다. 또한 응용 프로그램이 서로의 네트워크 트래픽, 파일 시스템 데이터 또는 프로세스 작업을 모니터링하거나 가로챌 수 없게 됩니다.

네트워크 분리
: 영역은 기본적으로 배타적 IP 유형으로 구성됩니다. 이 영역은 전역 영역으로부터 격리되고 IP 계층에서 서로 격리됩니다. 이러한 격리는 운영 및 보안상 유용합니다. 영역을 사용하면 고유한 LAN 또는

VLAN을 사용하는 다른 서브넷에서 통신해야 하는 응용 프로그램을 통합할 수 있습니다. 영역마다 고유한 IP 계층 보안 규칙을 정의할 수도 있습니다.

가상화
: 영역은 물리적 장치와 시스템의 기본 IP 주소와 호스트 이름 등 세부 정보를 응용 프로그램으로부터 숨길 수 있는 가상화 환경을 제공합니다. 동일한 응용 프로그램 환경이 서로 다른 물리적 시스템에서 유지될 수 있습니다. 가상화된 환경에서는 각 영역을 개별적으로 관리할 수 있습니다. 비전역 영역에서 영역 관리자가 수행하는 동작은 시스템의 나머지 부분에 영향을 미치지 않습니다.

세분성
: 영역은 거의 모든 세분성 레벨에서 분리를 제공합니다. 자세한 내용은 198 페이지 “비전역 영역 특성”을 참조하십시오.

환경
: 영역은 보안 및 분리의 목적을 실현하는 데 필요한 경우를 제외하고는 응용 프로그램이 실행되는 환경을 변경하지 않습니다. 영역은 응용 프로그램을 이식해야 하는 새로운 API 또는 ABI를 제공하지 않습니다. 대신에, 영역은 약간의 제한과 함께 표준 Oracle Solaris 인터페이스 및 응용 프로그램 환경을 제공합니다. 이 제한은 기본적으로 권한이 부여된 작업을 수행하려 하는 응용 프로그램에 영향을 줍니다.

  전역 영역의 응용 프로그램은 추가 영역의 구성 여부와 상관없이 수정되지 않은 상태로 실행됩니다.

시스템에서 영역 설정(작업 맵)

다음 표에서는 처음으로 시스템에 영역을 설치하는 데 필요한 작업에 대해 간략하게 설명합니다.

작업	설명	수행 방법
영역에서 실행하려는 응용 프로그램을 식별합니다.	시스템에서 실행 중인 응용 프로그램을 검토합니다. ■ 비즈니스 목표에 중요한 응용 프로그램을 결정합니다. ■ 실행 중인 응용 프로그램의 시스템 요구를 평가합니다.	필요할 경우 비즈니스 목표와 시스템 설명서를 참조하십시오.

작업	설명	수행 방법
구성할 영역 수를 결정합니다.	다음 사항을 평가합니다. ■ 영역에서 실행하려는 응용 프로그램의 성능 요구사항 ■ 설치할 영역당 사용 가능한 1GB 디스크 공간 필요한 양은 영역 내부에 설치할 소프트웨어에 따라 달라지므로 그에 맞게 조정해야 합니다. ZFS 압축을 사용하면 필요한 디스크 공간의 양이 감소됩니다. 비전역 영역 설치와 이후 패키지 설치 및 업데이트 중에는 약간의 임시 공간이 필요합니다. 1GB의 디스크 공간 요구 사항은 이를 고려한 것입니다.	238 페이지 “현재 시스템 설정 평가”를 참조하십시오.
리소스 풀이나 지정된 CPU를 사용하여 시스템 리소스를 분할할지 여부를 결정합니다.	또한 시스템에서 리소스 관리 기능을 사용 중인 경우 영역과 리소스 관리 경계를 맞춥니다. 영역을 구성하기 전에 리소스 풀을 구성합니다. `zonecfg` 등록 정보를 사용하여 영역 전체 리소스 제어와 풀 기능을 영역에 신속하게 추가할 수 있습니다.	243 페이지 “영역 구성 방법”과 13 장, “리소스 풀 만들기 및 관리(작업)”를 참조하십시오.

작업	설명	수행 방법
사전 구성 작업을 수행합니다.	영역 이름과 영역 경로를 결정합니다. 영역에 대한 추가 요구 사항을 결정합니다. 기본적으로 비전역 영역은 anet 리소스를 사용하여 배타적 IP 유형으로 만들어집니다. anet 리소스는 비전역 영역을 위한 자동 VNIC(가상 NIC)을 만듭니다. 또는 net 리소스를 사용하여 영역을 배타적 IP 영역 또는 공유 IP 영역으로 구성할 수 있습니다. 각 영역에 대한 필요한 파일 시스템과 장치를 결정합니다. 영역에 대한 예약 클래스를 결정합니다. 표준 기본 세트가 충분하지 않은 경우 영역 내부 프로세스가 제한되어야 하는 권한 세트를 결정합니다. 일부 zonecfg 설정은 권한을 자동으로 추가합니다. 예를 들면 ip-type=exclusive는 네트워크 스택을 구성 및 관리하는 데 필요한 여러 권한을 자동으로 추가합니다.	영역 이름 및 경로, IP 유형, IP 주소, 파일 시스템, 장치, 예약 클래스 및 권한에 대한 자세한 내용은 16 장, “비전역 영역 구성(개요)” 및 238 페이지 “현재 시스템 설정 평가”를 참조하십시오. 기본 권한과 비전역 영역에서 구성할 수 있는 권한의 목록은 337 페이지 “비전역 영역의 권한”을 참조하십시오. IP 기능에 대한 자세한 내용은 329 페이지 “공유 IP 비전역 영역의 네트워킹” 및 331 페이지 “배타적 IP 비전역 영역의 네트워킹”을 참조하십시오.
구성을 작성합니다.	비전역 영역을 구성합니다.	242 페이지 “영역 구성, 확인 및 커밋” 및 zonecfg(1M) 매뉴얼 페이지를 참조하십시오.
전역 관리자나 적절한 권한을 가진 사용자가 구성된 영역을 확인하고 설치합니다.	로그인하기 전에 영역을 확인하고 설치해야 합니다. 영역의 초기 내부 구성은 설치 시 만들어지고 구성됩니다.	18 장, “비전역 영역, 설치, 종료, 정지 및 복제 정보(개요)”, 19 장, “비전역 영역 설치, 부트, 종료, 정지, 제거 및 복제(작업)”, sysconfig(1M) 및 **Oracle Solaris 11 시스템**의 6 장, “Oracle Solaris 인스턴스 구성 해제 또는 재구성”을 참조하십시오.
전역 관리자나 적절한 권한이 부여된 사용자가 비전역 영역을 부트합니다.	각 영역을 부트하여 영역을 실행 중 상태로 설정합니다.	18 장, “비전역 영역, 설치, 종료, 정지 및 복제 정보(개요)” 및 19 장, “비전역 영역 설치, 부트, 종료, 정지, 제거 및 복제(작업)”를 참조하십시오.

작업	설명	수행 방법
새 영역에서 생산 프로세스를 수행할 수 있도록 준비합니다.	사용자 계정을 만들고 추가 소프트웨어를 추가하며 영역의 구성을 사용자 정의합니다.	새로 설치된 시스템을 설정하는 데 사용하는 설명서를 참조하십시오. 본 설명서에서는 영역이 설치된 시스템에 적용되는 특별 고려 사항을 다룹니다.

◆ ◆ ◆ 16 장

16 비전역 영역 구성(개요)

이 장에서는 비전역 영역 구성에 대해 소개합니다.

이 장에서는 다음 항목을 다룹니다.

영역 구성에 대해 알아본 후에는 17 장, “비전역 영역 계획 및 구성(작업)”으로 이동하여 시스템에 설치할 비전역 영역을 구성하십시오.

영역 내 리소스 정보

영역에서 제어할 수 있는 리소스에는 다음 항목이 포함됩니다.

- 시스템 리소스 분할에 사용되는 리소스 풀 또는 지정된 CPU
- 시스템 리소스에 대한 제약 조건 방식을 제공하는 리소스 제어
- 중요성에 따라 영역 간의 사용 가능한 CPU 리소스 할당을 제어할 수 있는 예약 클래스. 이 중요성은 각 영역에 지정하는 CPU 리소스의 할당 수로 표현됩니다.

영역 관리 시 권한 프로파일과 역할 사용

Oracle Solaris 11 보안 지침의 “Oracle Solaris 11 보안 기술”.

설치 전 구성 프로세스

비전역 영역을 설치하고 시스템에서 사용하기 전에 영역을 구성해야 합니다.

`zonecfg` 명령은 구성을 생성하고 지정된 리소스와 등록 정보가 가설 시스템에서 유효한지 여부를 결정하는 데 사용됩니다. 지정된 구성에 대해 `zonecfg`를 통해 수행되는 검사에서는 다음 사항을 확인합니다.

- 영역 경로가 지정되었는지 여부를 확인합니다.
- 각 리소스에 대한 필요한 모든 등록 정보가 지정되었는지 확인합니다.
- 구성에 충돌이 없는지 확인합니다. 예를 들어 `anet` 리소스가 있을 경우 영역은 배타적 IP 유형이므로 공유 IP 영역이 될 수 없습니다. 또한 `zonecfg` 명령은 가칭된 데이터 집합이 장치와 충돌할 가능성이 있는 경우 경고를 표시합니다.

`zonecfg` 명령에 대한 자세한 내용은 `zonecfg(1M)` 매뉴얼 페이지를 참조하십시오.

영역 구성 요소

이 절에서는 구성할 수 있는 필수 및 선택적 영역 구성 요소에 대해 설명합니다. 영역 이름과 영역 경로만 필요합니다. 추가 정보는 222 페이지 “영역 구성 데이터”를 참조하십시오.

영역 이름 및 경로

영역의 이름 및 경로를 선택해야 합니다. 영역이 ZFS 데이터 집합에 있어야 합니다. ZFS 데이터 집합은 영역을 설치하거나 연결할 때 자동으로 생성됩니다. ZFS 데이터 집합을 만들 수 없을 경우 영역이 설치되거나 연결되지 않습니다. 영역 경로의 부모 디렉토리도 데이터 집합이어야 합니다.

영역 자동 부트

`autoboot` 등록 정보 설정은 전역 영역이 부트될 때 영역이 자동으로 부트되는지 여부를 결정합니다. 영역 서비스인 `svc:/system/zones:default`도 사용으로 설정해야 합니다.

읽기 전용 루트 영역의 file-mac-profile 등록 정보

solaris 영역에서는 file-mac-profile이 읽기 전용 루트를 사용하여 영역을 구성하는 데 사용됩니다.

자세한 내용은 27 장, “변경할 수 없는 영역 구성 및 관리”를 참조하십시오.

admin 리소스

admin 설정을 사용하면 영역 관리 권한 부여를 설정할 수 있습니다. 권한 부여를 정의하는 기본 방법은 zonecfg 명령을 사용하는 것입니다.

user 사용자 이름을 지정합니다.

auths 사용자 이름에 대한 권한을 지정합니다.

solaris.zone.login	RBAC(역할 기반 액세스 제어)를 사용 중인 경우 solaris.zone.login/*zonename* 권한이 대화식 로그인에 필요합니다. 영역에서는 암호 인증이 수행됩니다.
solaris.zone.manage	RBAC가 사용 중인 경우 비대화식 로그인 시 또는 암호 인증을 생략하려면 solaris.zone.manage/*zonename* 권한이 필요합니다.
solaris.zone.clonefrom	RBAC가 사용 중인 경우 다른 영역을 복사하는 하위 명령에 solaris.zone.clonefrom/*source_zone* 권한이 필요합니다.

리소스 풀 연결

13 장, “리소스 풀 만들기 및 관리(작업)”의 설명에 따라 시스템에서 리소스 풀을 구성한 경우 pool 등록 정보를 사용하여 영역을 구성할 때 리소스 풀 중 하나를 영역에 연결할 수 있습니다.

리소스 풀을 구성하지 않은 경우 dedicated-cpu 리소스를 사용하여 비전역 영역을 실행하는 동안 시스템의 일부 프로세서를 해당 영역 전용으로 지정할 수 있습니다. 영역이 실행 중인 동안 시스템에서 사용할 임시 풀을 동적으로 만듭니다. zonecfg를 통해 지정한 상태에서는 풀 설정이 마이그레이션 중에 전파됩니다.

주 – pool 등록 정보를 통해 설정된 지속성 풀을 사용하는 영역 구성은 dedicated-cpu 리소스를 통해 구성된 임시 풀과 호환되지 않습니다. 이러한 두 가지 등록 정보 중 하나만 설정할 수 있습니다.

dedicated-cpu 리소스

dedicated-cpu 리소스는 비전역 영역이 실행되는 동안 시스템의 일부 프로세서를 해당 영역 전용으로 지정합니다. 영역이 부트될 때 시스템은 영역이 실행 중인 동안 사용할 임시 풀을 동적으로 만듭니다.

zonecfg을 통해 지정한 상태에서는 풀 설정이 마이그레이션 중에 전파됩니다.

dedicated-cpu 리소스는 ncpus 및 importance(옵션)의 제한을 설정합니다.

ncpus
: CPU 수를 지정하거나 2-4개 CPU 등 범위를 지정합니다. 동적 리소스 풀 동작을 원해서 범위를 지정하는 경우 다음 작업도 수행합니다.
  - importance 등록 정보를 설정합니다.
  - poold 서비스를 사용으로 설정합니다. 자세한 내용은 158 페이지 "svcadm을 사용하여 동적 리소스 풀 서비스를 사용으로 설정하는 방법"을 참조하십시오.

importance
: CPU 범위를 사용하여 동작 동작을 수행하려면 importance 등록 정보도 설정합니다. importance 등록 정보는 **선택 사항**으로, 풀의 상대적 중요성을 정의합니다. 이 등록 정보는 ncpus에 대한 범위를 지정하고 poold를 통해 관리되는 동적 리소스 풀을 사용할 때만 필요합니다. poold를 실행하고 있지 않은 경우 importance가 무시됩니다. poold가 실행 중이고 importance가 설정되지 않은 경우 importance의 기본값은 1입니다. 자세한 내용은 143 페이지 "pool.importance 등록 정보 제약 조건"을 참조하십시오.

주 – capped-cpu 리소스와 dedicated-cpu 리소스는 호환되지 않습니다. cpu-shares rctl과 dedicated-cpu 리소스는 호환되지 않습니다.

capped-cpu 리소스

capped-cpu 리소스는 프로젝트나 영역에서 사용할 수 있는 CPU 리소스 양에 대한 세분화된 절대적 제한을 제공합니다. CPU 상한값은 프로세서 세트와 함께 사용되는 경우 집합 내에서 CPU 사용을 제한합니다. capped-cpu 리소스에는 소수점 이하 자리 수가 두 개인 양수를 나타내는 단일 ncpus 등록 정보가 있습니다. 이 등록 정보는 CPU 단위에 해당합니다. 리소스는 범위를 허용하지 않습니다. 리소스는 십진수를

허용합니다. ncpus를 지정하는 경우 값 1은 CPU 100%를 의미합니다. 100%가 시스템에서 한 개의 CPU 전체를 의미하므로 값 1.25는 125%를 의미합니다.

주 – capped-cpu 리소스와 dedicated-cpu 리소스는 호환되지 않습니다.

예약 클래스

FSS(Fair Share Scheduler))를 사용하여 중요성에 따라 사용 가능한 CPU 리소스 할당을 제어할 수 있습니다. 이 중요성은 각 영역에 지정하는 CPU 리소스의 **할당** 수로 표현됩니다. FSS를 사용하여 영역 간 CPU 리소스 할당을 관리하기를 원치 않더라도 영역 내에서 프로젝트에 대한 할당을 설정할 수 있도록 영역의 예약 클래스를 설정하여 FSS를 사용할 수 있습니다.

cpu-shares 등록 정보를 명시적으로 설정하는 경우 FSS(Fair Share Scheduler)가 해당 영역에 대한 예약 클래스로 사용됩니다. 하지만 여기서의 기본적인 FSS 사용 방법은 dispadmin 명령을 사용하여 FSS를 시스템 기본 예약 클래스로 설정하는 것입니다. 즉, 모든 영역이 시스템 CPU 리소스를 공정하게 할당받을 수 있습니다. cpu-shares가 영역에 대해 설정되지 않을 경우 영역에서는 시스템의 기본 예약 클래스를 사용합니다. 다음 작업은 영역에 대한 예약 클래스를 설정합니다.

- zonecfg에서 scheduling-class 등록 정보를 사용하여 해당 영역에 대한 예약 클래스를 설정할 수 있습니다.
- 리소스 풀 기능을 통해 영역에 대한 예약 클래스를 설정할 수 있습니다. pool.scheduler 등록 정보가 유효한 예약 클래스로 설정된 풀과 영역이 연결된 경우 영역에서 실행 중인 클래스가 기본적으로 해당 예약 클래스에서 실행됩니다. 134 페이지 “리소스 풀 소개” 및 164 페이지 “풀과 예약 클래스를 연결하는 방법”을 참조하십시오.
- cpu-shares rctl이 설정되었으며 FSS가 다른 작업을 통해 영역의 예약 클래스로 설정되지 않은 경우 zoneadmd는 영역이 부트될 때 예약 클래스를 FSS로 설정합니다.
- 다른 작업을 통해 예약 클래스가 설정되지 않은 경우 영역은 시스템 기본 예약 클래스를 상속합니다.

priocntl(1) 매뉴얼 페이지에 설명된 priocntl을 사용하여 기본 예약 클래스와 재부트 없이 실행 중인 프로세스를 다른 예약 클래스로 이동할 수 있습니다.

물리적 메모리 제어 및 capped-memory 리소스

capped-memory 리소스는 physical, swap 및 locked 메모리에 대한 제한을 설정합니다. 각 제한은 선택 사항이지만 하나 이상의 제한을 설정해야 합니다. capped-memory 리소스를 사용하려면 resource-cap 패키지를 전역 영역에 설치해야 합니다.

- 전역 영역에서 rcapd를 사용하여 해당 영역에 대한 메모리를 제한하려는 경우 이 리소스의 값을 결정합니다. capped-memory 리소스의 physical 등록 정보는 해당 영역의 max-rss 값으로 사용되는 rcapd에 사용됩니다.
- capped-memory 리소스의 swap 등록 정보는 zone.max-swap 리소스 제어를 설정하는 기본 방법입니다.
- capped-memory 리소스의 locked 등록 정보는 zone.max-locked-memory 리소스 제어를 설정하는 기본 방법입니다.

주 – 응용 프로그램은 일반적으로 많은 양의 메모리를 잠그지 않지만, 영역의 응용 프로그램이 메모리를 잠그는 것으로 알려진 경우에는 잠김 메모리를 설정할 수 있습니다. 영역 신뢰가 문제가 되는 경우 잠긴 메모리 상한값을 시스템의 물리적 메모리 중 10% 또는 영역 물리적 메모리 상한값의 10%로 설정하는 것을 고려해 볼 수도 있습니다.

자세한 내용은 10 장, “리소스 상한값 지원 데몬을 사용한 물리적 메모리 제어(개요)”, 11 장, “리소스 상한값 지원 데몬 관리(작업)” 및 243 페이지 “영역 구성 방법”을 참조하십시오. 영역에 대한 리소스 상한값을 임시로 설정하려면 129 페이지 “영역에 대한 임시 리소스 상한값을 지정하는 방법”을 참조하십시오.

영역 네트워크 인터페이스

zonecfg 유틸리티에서 네트워크 연결을 제공하기 위해 구성된 영역 네트워크 인터페이스는 영역이 부트될 때 자동으로 설정되어 영역에 배치됩니다.

IP(인터넷 프로토콜) 계층은 네트워크의 패킷을 받고 전달합니다. 이 계층에는 IP 경로 지정, ARP(Address Resolution Protocol), IPsec(IP 보안 구조) 및 IP 필터가 포함됩니다.

비전역 영역에 사용할 수 있는 IP 유형에는 공유 IP와 배타적 IP라는 두 가지 유형이 있습니다. 배타적 IP는 기본 IP 유형입니다. 공유 IP 영역은 전역 영역과 네트워크 인터페이스를 공유합니다. 공유 IP 영역을 사용하려면 ipadm 유틸리티를 통해 전역 영역의 구성을 수행해야 합니다. 배타적 IP 영역에는 전용 네트워크 인터페이스가 있어야 합니다. anet 리소스를 사용하여 배타적 IP 영역이 구성된 경우 전용 VNIC가 자동으로 만들어지고 해당 영역에 지정됩니다. 자동화된 anet 리소스를 사용하면 전역 영역에서 데이터 링크를 생성 및 구성하고 데이터 링크를 비전역 영역에 지정할 필요가 없습니다. anet 리소스를 사용하여 다음을 수행합니다.

- 전역 영역 관리자가 비전역 영역에 지정된 데이터 링크에 대한 특정 이름을 선택할 수 있습니다.
- 여러 영역에서 동일한 이름의 데이터 링크를 사용할 수 있습니다.

이전 버전과의 호환성을 위해 미리 구성된 데이터 링크를 비전역 영역에 지정할 수 있습니다.

각 유형의 IP 기능에 대한 자세한 내용은 329 페이지 “공유 IP 비전역 영역의 네트워킹” 및 331 페이지 “배타적 IP 비전역 영역의 네트워킹”을 참조하십시오.

주 – 영역을 실행하는 시스템에서 **Oracle Solaris 관리: 네트워크 인터페이스 및 네트워크 가상화**의 20 장, “가상화된 환경에서 링크 보호 사용”에 설명된 링크 보호를 사용할 수 있습니다. 이 기능은 전역 영역에서 구성됩니다.

데이터 링크 정보

데이터 링크는 OSI 프로토콜 스택의 계층 2에 있는 인터페이스로, 시스템에서 STREAMS DLPI(v2) 인터페이스로 표현됩니다. 이러한 인터페이스는 TCP/IP 등 프로토콜 스택에서 연결될 수 있습니다. 데이터 링크는 NIC(네트워크 인터페이스 카드)과 같은 물리적 인터페이스라고도 합니다. 데이터링크는 `zonecfg` (1M)을 사용하여 구성된 `physical` 등록 정보입니다. `physical` 등록 정보는 **Oracle Solaris 관리: 네트워크 인터페이스 및 네트워크 가상화**의 제III부, “네트워크 가상화 및 리소스 관리”에서 설명한 대로 VNIC일 수 있습니다.

예를 들어 `e1000g0` 및 `bge1`과 같은 물리적 인터페이스, `bge3`과 같은 NIC, `aggr1` 및 `aggr2`와 같은 합계 또는 `e1000g123000` 및 `bge234003`과 같은 VLAN 태그된 인터페이스(각각 `e1000g0`의 VLAN123과 `bge3`의 VLAN234)는 모두 데이터 링크입니다.

공유 IP 비전역 영역

공유 IP 영역은 전역 영역의 기존 IP 인터페이스를 사용합니다. 영역에는 하나 이상의 배타적 IP 주소가 있어야 합니다. 공유 IP 영역은 IP 계층 구성 및 상태를 전역 영역과 공유합니다. 다음 두 가지 조건이 모두 참이면 영역은 공유 IP 인스턴스를 사용해야 합니다.

- 비전역 영역은 전역 및 비전역 영역이 동일 서브넷에 있는지 여부와 상관없이 전역 영역에서 사용되는 동일한 데이터 링크를 사용합니다.
- 배타적 IP 영역이 제공하는 다른 기능은 필요하지 않습니다.

공유 IP 영역에는 `zonecfg` 영역의 `net`리소스를 사용하여 하나 이상의 주소가 지정됩니다. 전역 영역에서 데이터 링크 이름도 구성해야 합니다.

`zonecfg net` 리소스에서 `address` 및 `physical` 등록 정보를 설정해야 합니다. `defrouter` 등록 정보는 선택 사항입니다.

전역 영역의 공유 IP 네트워킹 구성을 사용하려면 자동 네트워크 구성이 아닌 `ipadm`를 사용해야 합니다. `ipadm`을 통해 네트워킹 구성이 수행되는지 여부를 확인하려면 다음 명령을 실행합니다. 표시되는 응답이 `DefaultFixed`여야 합니다.

```
# svcprop -p netcfg/active_ncp svc:/network/physical:default
DefaultFixed
```

공유 IP 영역에 지정된 IP 주소는 논리 네트워크 인터페이스와 연결됩니다.

전역 영역에서 `ipadm` 명령을 사용하여 실행 중인 영역에서 논리 인터페이스를 지정하거나 제거할 수 있습니다.

인터페이스를 추가하려면 다음 명령을 사용합니다.

```
global# ipadm set-addrprop -p zone=my-zone net0/addr1
```

인터페이스를 제거하려면 다음 명령 중 하나를 사용합니다.

```
global# ipadm set-addrprop -p zone=global net0/addr
```

또는

```
global# ipadm reset-addrprop -p zone net0/addr1
```

자세한 내용은 330 페이지 “공유 IP 네트워크 인터페이스”를 참조하십시오.

배타적 IP 비전역 영역

배타적 IP는 비전역 영역을 위한 기본 네트워킹 구성입니다.

하나의 배타적 IP 영역에는 하나 이상의 전용 데이터 링크와 고유한 IP 관련 상태가 지정됩니다.

배타적 IP 영역에서 다음 기능을 사용할 수 있습니다.

- DHCPv4 및 IPv6 Stateless 주소 자동 구성
- IP 필터, NAT(네트워크 주소 변환) 기능 포함
- IPMP(IP Network Multipathing)
- IP 경로 지정
- TCP/UDP/SCTP 및 IP/ARP 레벨 노브 설정용 `ipadm`
- IPsec 보안 연결을 위한 인증된 키 관련 자료 프로비저닝을 자동화하는 IKE(Internet Key Exchange)와 IPSec(IP 보안)

배타적 IP 영역을 구성하는 방법에는 두 가지가 있습니다.

- `zonecfg` 유틸리티의 `anet` 사용을 사용하여 영역이 부트될 때 영역의 임시 VNIC를 자동으로 생성하고 영역이 정지할 때 삭제합니다.
- `zonecfg` 유틸리티의 `net` 리소스를 사용하여 전역 영역에서 데이터 링크를 미리 구성하고 배타적 IP 영역에 지정합니다. `net` 리소스의 `physical` 등록 정보를 사용하여 데이터 링크가 지정됩니다. `physical` 등록 정보는 **Oracle Solaris 관리: 네트워크**

인터페이스 및 네트워크 가상화의 제III부, "네트워크 가상화 및 리소스 관리"에서 설명한 대로 VNIC일 수 있습니다. net 리소스의 address 등록 정보는 설정되지 않습니다.

기본적으로 배타적 IP 영역은 연결된 인터페이스의 IP 주소를 구성하고 사용할 수 있습니다. 또는 allowed-address 등록 정보를 사용하여 쉼표로 구분된 IP 주소 목록을 지정할 수 있습니다. 배타적 IP 영역은 allowed-address 목록에 없는 IP 주소를 사용할 수 없습니다. 또한, allowed-address 목록의 모든 주소는 영역이 부트될 때 배타적 IP 영역에 대해 자동으로 영구 구성됩니다. 이 인터페이스 구성을 원치 않는 경우에는 configure-allowed-address 등록 정보를 false로 설정해야 합니다. 기본값은 true입니다.

지정된 데이터 링크는 snoop 명령을 사용으로 설정합니다.

dladm 명령을 show-linkprop 하위 명령과 함께 사용하여 실행 중인 배타적 IP 영역에 데이터 링크의 할당을 표시할 수 있습니다. dladm 명령을 set-linkprop 하위 명령과 함께 사용하여 실행 중인 영역에 추가 데이터 링크를 지정할 수 있습니다. 사용 예는 364 페이지 "배타적 IP 비전역 영역에서 데이터 링크 관리"를 참조하십시오.

고유한 데이터 링크 집합이 지정된 실행 중인 배타적 IP 영역 내부에서는 ipadm 명령을 사용하여 IP를 구성합니다. 여기에는 논리적 인터페이스를 추가하거나 제거할 수 있는 기능이 포함됩니다. sysconfig(1M) 매뉴얼 페이지에 설명된 sysconfig 인터페이스를 사용하여 전역 영역에서와 똑같이 영역 내 IP 구성을 설정할 수 있습니다.

배타적 IP 영역의 IP 구성은 zlogin 명령을 사용하여 전역 영역에서만 볼 수 있습니다.

```
global# zlogin zone1 ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
nge0/_b           dhcp     ok           10.134.62.47/24
lo0/v6            static   ok           ::1/128
nge0/_a           addrconf ok           fe80::2e0:81ff:fe5d:c630/10
```

공유 IP 비전역 영역과 배타적 IP 비전역 영역의 보안 차이점

공유 IP 영역에서는 슈퍼 유저를 비롯하여 영역의 응용 프로그램이 zonecfg 유틸리티를 통해 영역에 지정된 주소와는 다른 소스 IP 주소와 함께 패킷을 전송할 수 없습니다. 이 영역 유형에는 임의 데이터 링크(계층 2) 패킷을 보내고 받을 수 있는 권한이 없습니다.

배타적 IP 영역의 경우 대신에 zonecfg는 지정된 전체 데이터 링크를 영역에 부여합니다. 결과적으로 배타적 IP 영역에서는 슈퍼 유저나 필요한 권한 프로파일을 가진 사용자가 전역 영역에서처럼 이러한 데이터 링크를 통해 스푸핑된 패킷을 보낼 수 있습니다. allowed-address 등록 정보를 설정하여 IP 주소 스푸핑을 사용 안함으로 설정할 수 있습니다. anet 리소스의 경우 mac-nospoof 및 dhcp-nospoof 등 추가 보호는 link-protection 등록 정보를 설정하여 사용으로 설정할 수 있습니다.

공유 IP 및 배타적 IP 비전역 영역 동시 사용

공유 IP 영역은 항상 IP 계층을 전역 영역과 공유하며 배타적 IP 영역은 항상 IP 계층의 고유 인스턴스를 가지고 있습니다. 공유 IP 영역과 배타적 IP 영역 둘 다 동일한 시스템에서 사용할 수 있습니다.

영역에서 마운트된 파일 시스템

각 영역에는 기본적으로 ZFS 데이터 집합이 위임됩니다. 이 기본 위임 데이터 집합은 기본 전역 영역 데이터 집합 레이아웃의 데이터 집합 레이아웃을 대신합니다. .../rpool/ROOT라고 하는 데이터 집합에는 부트 환경이 포함되어 있습니다. 이 데이터 집합은 직접 조작하면 안 됩니다. 존재해야 하는 rpool 데이터 집합은 기본적으로 .../rpool에서 마운트됩니다. .../rpool/export 및 .../rpool/export/home 데이터 집합은 /export 및 /export/home에서 마운트됩니다. 이러한 비전역 영역은 해당 전역 영역 데이터 집합과 동일하게 사용되며 동일한 방식으로 관리할 수 있습니다. 영역 관리자는 .../rpool, .../rpool/export 및 ... /rpool/export/home 데이터 집합 내에서 추가 데이터 집합을 만들 수 있습니다.

일반적으로 영역에 마운트된 파일 시스템에는 다음 항목이 포함됩니다.

- 가상 플랫폼이 초기화될 때 마운트된 파일 시스템 집합
- 응용 프로그램 환경 자체 내에서 마운트된 파일 시스템 집합

예를 들어 이러한 집합에는 다음과 같은 파일 시스템이 포함될 수 있습니다.

- canmount 등록 정보의 값이 yes인 legacy 또는 none 이외의 mountpoint를 사용하는 ZFS 파일 시스템
- 영역의 /etc/vfstab 파일에 지정된 파일 시스템
- AutoFS 및 AutoFS 트리거 마운트 autofs 등록 정보는 sharectl(1M)에 설명된 sharectl을 사용하여 설정됩니다.
- 영역 관리자가 명시적으로 수행하는 마운트

  실행 중인 영역 내 파일 시스템 마운트 권한도 zonecfg fs-allowed 등록 정보를 통해 정의됩니다. 이 등록 정보는 zonecfg add fs 또는 add dataset 리소스를 사용하여 영역에 마운트된 파일 시스템에는 적용되지 않습니다. 기본적으로 영역의 기본 위임 데이터 집합 내 파일 시스템, hsfs 파일 시스템 및 네트워크 파일 시스템(예: NFS)의 마운트만 영역 내에서 허용됩니다.

주의 – 특정 제한은 응용 프로그램 환경 내에서 수행되는 기본값과 다른 마운트에만 적용됩니다. 이러한 제한으로 인해 영역 관리자는 시스템의 나머지 부분에 대한 서비스를 거부하지 못합니다. 그렇지 않으면 다른 영역에 부정적인 영향을 미치게 됩니다.

영역 내 특정 파일 시스템 마운트와 관련하여 보안 제한이 있습니다. 영역에서 다른 파일 시스템이 마운트될 경우 특정 동작을 나타냅니다. 자세한 내용은 322 페이지 “파일 시스템 및 비전역 영역”을 참조하십시오.

영역의 호스트 ID

전역의 hostid와는 다른 비전역 영역에 대한 hostid 등록 정보를 설정할 수 있습니다. 예를 들면 다른 시스템의 영역으로 마이그레이션된 시스템의 경우에 이와 같이 설정할 수 있습니다. 이제 영역 내 응용 프로그램은 원래의 hostid에 따라 다를 수 있습니다. 자세한 내용은 222 페이지 “리소스 유형과 등록 정보”를 참조하십시오.

영역에 구성된 장치

zonecfg 명령은 규칙 일치 시스템을 사용하여 특정 영역에 표시되어야 할 장치를 지정합니다. 규칙 중 하나와 일치하는 장치가 영역의 /dev 파일 시스템에 포함됩니다. 자세한 내용은 243 페이지 “영역 구성 방법”을 참조하십시오.

비전역 영역의 디스크 포맷 지원

uscsi 명령의 디스크 분할 및 사용은 zonecfg 도구를 통해 사용으로 설정됩니다. 관련 예는 227 페이지 “리소스 유형 등록 정보”의 device를 참조하십시오. uscsi 명령에 대한 자세한 내용은 uscsi(7I)를 참조하십시오.

- 위임은 solaris 영역에 대해서만 지원됩니다.
- 디스크는 표시된 것처럼 -D 옵션과 prtconf 명령을 함께 사용하여 sd 대상을 사용해야 합니다. prtconf(1M)을 참조하십시오.

영역 전체 리소스 제어 설정

영역 관리자 또는 해당 권한을 가진 사용자는 영역에 대해 권한 부여된 영역 전체 리소스 제어를 설정할 수 있습니다. 영역 전체 리소스 제어는 영역 내에서 모든 프로세스 항목의 총 리소스 사용을 제한합니다.

이러한 제한은 zonecfg 명령을 사용하여 전역 및 비전역 영역 모두에 대해 지정됩니다. 243 페이지 “영역 구성 방법”을 참조하십시오.

영역 전체 리소스 제어를 설정하기 위한 보다 간단한 기본적인 방법은 rctl 리소스(예: cpu-cap) 대신에 capped-cpu와 같은 등록 정보 이름 또는 리소스를 사용하는 것입니다.

zone.cpu-cap 리소스 제어는 영역에 사용될 수 있는 CPU 리소스의 양에 대한 절대적 제한을 설정합니다. 값 100은 한 CPU의 100%를 설정함을 의미합니다. 100%는 CPU 상한값을 사용할 때 시스템의 CPU 하나 전체에 해당하므로 125 값은 125%입니다.

주 – capped-cpu 리소스를 설정할 때 단위로 십진수를 사용할 수 있습니다. 이 값은 zone.cpu-cap 리소스 제어와 상관되지만 설정이 100분의 1로 낮춰집니다. 설정 1은 리소스 제어의 설정 100에 해당합니다.

---

zone.cpu-shares 리소스 제어는 영역에 대해 FSS(Fair Share Scheduler) CPU 할당에 대한 제한을 설정합니다. 먼저 CPU 할당이 영역에 할당되고 나서 project.cpu-shares 항목에 지정된 대로 영역 내 여러 프로젝트에 세분됩니다. 자세한 내용은 366 페이지 “영역이 설치된 Oracle Solaris 시스템에서 Fair Share Scheduler 사용”을 참조하십시오. 이 컨트롤의 전역 등록 정보 이름은 cpu-shares입니다.

zone.max-locked-memory 리소스 제어는 영역에 사용할 수 있는 잠긴 물리적 메모리의 양을 제한합니다. 영역 내 프로젝트 전체에 대해 잠긴 메모리 리소스 할당은 project.max-locked-memory 리소스 제어를 사용하여 제어할 수 있습니다. 자세한 내용은 표 6–1을 참조하십시오.

zone.max-lofi 리소스 제어는 영역에서 생성할 수 있는 잠재적 lofi 장치 수를 제한합니다.

zone.max-lwps 리소스 제어는 한 영역의 너무 많은 LWP가 다른 영역에 영향을 주지 않도록 함으로써 리소스 격리를 향상시킵니다. 영역 내 프로젝트 간의 LWP 리소스 할당은 project.max-lwps 리소스 제어를 사용하여 제어할 수 있습니다. 자세한 내용은 표 6–1을 참조하십시오. 이 컨트롤의 전역 등록 정보 이름은 max-lwps입니다.

zone.max-processes 리소스 제어는 한 영역에서 너무 많은 프로세스 테이블 슬롯을 사용하여 다른 영역에 영향을 주지 못하게 함으로써 리소스 격리를 향상시킵니다. 영역 내 프로젝트 간의 프로세스 테이블 슬롯 리소스 할당은 78 페이지 “사용 가능한 리소스 제어”에 설명된 project.max-processes 리소스 제어를 사용하여 설정할 수 있습니다. 이 컨트롤의 전역 등록 정보 이름은 max-processes입니다. zone.max-processes 리소스 제어는 zone.max-lwps 리소스 제어를 포함할 수도 있습니다. zone.max-processes가 설정되고 zone.max-lwps가 설정되지 않은 경우 영역이 부트될 때 zone.max-lwps가 암시적으로 zone.max-processes 값의 10배로 설정됩니다. 일반 프로세스와 좀비 프로세스는 모두 프로세스 테이블 슬롯을 사용하기 때문에 max-processes 컨트롤은 좀비 프로세스가 프로세스 테이블을 소모하지 못하도록 방지합니다. 좀비 프로세스에는 정의상 LWP가 없으므로 max-lwps가 이 가능성으로부터 보호할 수 없습니다.

zone.max-msg-ids, zone.max-sem-ids , zone.max-shm-ids 및 zone.max-shm-memory 리소스 제어는 영역 내 모든 프로세스에 사용되는 시스템 V 리소스를 제한하는 데 사용됩니다. 영역 내 프로젝트 간의 시스템 V 리소스 할당은 이러한 리소스 제어의

프로젝트 버전을 사용하여 제어할 수 있습니다. 이러한 컨트롤의 전역 등록 정보 이름은 max-msg-ids, max-sem-ids, max-shm-ids 및 max-shm-memory입니다.

zone.max-swap 리소스 제어는 영역 내 tmpfs 마운트와 사용자 프로세스 주소 공간 매핑에 사용되는 스왑을 제한합니다. prstat -Z의 출력에 SWAP 열이 표시됩니다. 보고된 스왑은 영역의 프로세스 및 tmpfs 마운트에서 사용된 총 스왑입니다. 이 값은 zone.max-swap 설정을 선택하는 데 사용할 수 있는 각 영역에 예약된 스왑을 모니터링하는 데 도움이 됩니다.

표 16–1 영역 전체 리소스 제어

컨트롤 이름	전역 등록 정보 이름	설명	기본 단위	사용된 값
zone.cpu-cap		이 영역의 CPU 리소스 양에 대한 절대 제한	수량(CPU 수), 백분율로 표현됨 **주 –** capped-cpu 리소스로 설정하는 경우 단위로 십진수를 사용할 수 있습니다.	
zone.cpu-shares	cpu-shares	이 영역에 대한 FSS(Fair Share Scheduler) CPU 할당 수	수량(할당)	
zone.max-locked-memory		영역에 사용할 수 있는 전체 물리적 잠긴 메모리의 양입니다. priv_proc_lock_memory가 영역에 지정된 경우 이 리소스 제어를 설정하여 해당 영역이 모든 메모리를 잠그는 것을 방지할 수 있습니다.	크기(바이트)	capped-memory의 locked 등록 정보
zone.max-lofi	max-lofi	영역에서 생성할 수 있는 잠재적 lofi 장치 수에 대한 제한	수량(lofi 장치 수)	

표 16–1 영역 전체 리소스 제어 *(계속)*

컨트롤 이름	전역 등록 정보 이름	설명	기본 단위	사용된 값
zone.max-lwps	max-lwps	이 영역에 동시에 사용할 수 있는 최대 LWP 수입니다	수량(LWP)	
zone.max-msg-ids	max-msg-ids	이 영역에 허용되는 최대 메시지 대기열 ID 수입니다	수량(메시지 대기열 ID)	
zone.max-processes	max-processes	이 영역에 동시에 사용할 수 있는 최대 프로세스 테이블 슬롯 수	수량(프로세스 테이블 슬롯)	
zone.max-sem-ids	max-sem-ids	이 영역에 허용되는 최대 세마포 ID 수입니다	수량(세마포 ID)	
zone.max-shm-ids	max-shm-ids	이 영역에 허용되는 공유 메모리 ID 수입니다	수량(공유 메모리 ID)	
zone.max-shm-memory	max-shm-memory	이 영역에 허용되는 시스템 V 공유 메모리의 총량입니다.	크기(바이트)	
zone.max-swap		이 영역에 대한 사용자 프로세스 주소 공간 매핑 및 tmpfs 마운트에 사용할 수 있는 총 스왑 크기입니다.	크기(바이트)	capped-memory의 swap 등록 정보

prctl 명령을 사용하여 프로세스 실행을 위해 이러한 제한을 지정할 수 있습니다. 366 페이지 “prctl 명령을 사용하여 전역 영역에서 FSS 할당을 설정하는 방법”에는 이와 관련된 예가 제공됩니다. prctl 명령을 통해 지정된 제한은 지속적이지 않습니다. 시스템을 재부트할 때까지만 제한이 적용됩니다.

구성 가능한 권한

영역이 부트될 때 구성에 기본 **안전** 권한 집합이 포함됩니다. 영역의 권한 부여된 프로세스는 시스템의 다른 비전역 영역 또는 전역 영역의 프로세스에 영향을 주지 않으므로 이러한 권한은 안전한 것으로 간주됩니다. zonecfg 명령을 사용하여 다음을 수행할 수 있습니다.

- 전역 리소스를 제어할 수 있게 됨에 따라 이러한 변경으로 인해 한 영역의 프로세스가 다른 영역의 프로세스에 영향을 줄 수 있음을 고려하여 기본 권한 집합에 추가합니다.
- 이러한 권한을 실행해야 하는 경우 이러한 변경으로 인해 일부 프로세스가 올바르게 작동하지 못할 수도 있다는 점을 고려하여 기본 권한 집합에서 제거합니다.

주 – 영역의 기본 권한 집합에서 제거할 수 없는 권한이 있는 반면, 이러한 집합에 추가할 수 없는 권한도 있습니다.

자세한 내용은 337 페이지 "비전역 영역의 권한", 243 페이지 "영역 구성 방법" 및 `privileges(5)`를 참조하십시오.

영역에 대한 설명 포함

`attr` 리소스 유형을 사용하여 영역에 대한 설명을 추가할 수 있습니다. 자세한 내용은 243 페이지 "영역 구성 방법"을 참조하십시오.

zonecfg 명령 사용

`zonecfg`(1M) 매뉴얼 페이지에 설명되어 있는 `zonecfg` 명령은 비전역 영역을 구성하는 데 사용됩니다.

`zonecfg` 명령을 사용하여 전역 영역에 대한 리소스 관리 설정을 영구적으로 지정할 수도 있습니다. 예를 들면, `dedicated-cpu` 리소스를 사용하여 전용 CPU를 사용하도록 전역 영역을 구성하는 데 이 명령을 사용할 수 있습니다.

`zonecfg` 명령은 대화식 모드, 명령줄 모드 또는 명령 파일 모드로 사용할 수 있습니다. 이 명령을 사용하여 다음 작업을 수행할 수 있습니다.

- 영역 구성 만들기 또는 삭제(완전 삭제)
- 특정 구성에 리소스 추가
- 구성에 추가된 리소스에 대한 등록 정보 설정
- 특정 구성에서 리소스 제거
- 구성 쿼리 또는 확인
- 구성으로 완결
- 이전 구성으로 되돌리기
- 영역 이름 바꾸기
- `zonecfg` 세션 종료

`zonecfg` 프롬프트는 다음과 같은 형태입니다.

`zonecfg:`*zonename*`>`

파일 시스템과 같이 특정 리소스 유형을 구성할 때 해당 리소스 유형도 프롬프트에 포함됩니다.

```
zonecfg:zonename:fs>
```

이 장에서 설명한 다양한 zonecfg 구성 요소를 사용하는 방법을 보여 주는 절차를 비롯하여 자세한 내용은 17 장, “비전역 영역 계획 및 구성(작업)”을 참조하십시오.

zonecfg 모드

범위의 개념은 사용자 인터페이스용으로 사용됩니다. 범위는 **전역** 범위이거나 **리소스 특정** 범위입니다. 기본 범위는 전역입니다.

전역 범위에서 add 하위 명령과 select 하위 명령은 특정 리소스를 선택하는 데 사용됩니다. 그러면 범위가 해당 리소스 유형으로 변경됩니다.

- add 하위 명령의 경우 리소스 지정을 완료하기 위해 end 또는 cancel 하위 명령이 사용됩니다.
- select 하위 명령의 경우 리소스 수정을 완료하는 데 end 또는 cancel 하위 명령이 사용됩니다.

그러면 범위가 전역으로 되돌려집니다.

add, remove 및 set과 같은 특정 하위 명령은 각 범위마다 의미가 다릅니다.

zonecfg 대화식 모드

대화식 모드에서는 다음 하위 명령이 지원됩니다. 하위 명령과 함께 사용되는 의미와 옵션에 대한 자세한 내용은 zonecfg(1M) 매뉴얼 페이지를 참조하십시오. 하위 명령으로 인해 완전히 삭제되거나 작업이 손실될 수 있는 경우 진행하기 전에 시스템에서 사용자 확인을 요청합니다. -F(강제) 옵션을 사용하여 이 확인을 생략할 수 있습니다.

help
: 일반 도움말을 인쇄하거나 지정된 리소스에 대한 도움말을 표시합니다.

```
zonecfg:my-zone:capped-cpu> help
```

create
: 다음 목적 중 하나로 지정된 새 영역에 대한 메모리 내 구성을 구성하기 시작합니다.
  - Oracle Solaris 기본 설정을 새 구성에 적용하기 위해. 이 방법은 기본값입니다.
  - -t *template* 옵션을 사용하여 지정된 템플리트와 동일한 구성을 생성하기 위해. 영역 이름은 템플리트 이름을 새 영역 이름으로 변경됩니다.
  - -F 옵션을 사용하여 기존 구성을 덮어쓰기 위해.
  - -b 옵션을 사용하여 아무것도 설정되는 빈 구성을 생성하기 위해.

export
: 구성을 표준 출력이나 명령 파일에 사용할 수 있는 형태의 지정된 출력 파일로 인쇄합니다.

`add` 전역 범위에서 지정된 리소스 유형을 구성에 추가합니다.

리소스 범위에서 지정된 값과 지정된 이름의 등록 정보를 추가합니다.

자세한 내용은 243 페이지 “영역 구성 방법”과 zonecfg(1M) 매뉴얼 페이지를 참조하십시오.

`set` 지정된 등록 정보 이름을 지정된 등록 정보 값으로 설정합니다. `zonepath`와 같은 일부 등록 정보는 전역이지만, 그 밖의 등록 정보는 리소스에 따라 달라집니다. 따라서 이 명령은 전역 및 리소스 범위 모두에서 사용할 수 있습니다.

`select` 전역 범위에서만 사용할 수 있습니다. 수정에 대한 지정된 등록 정보 이름-등록 정보 값 쌍 기준과 일치하는 지정된 유형의 리소스를 선택합니다. 범위가 해당 리소스 유형으로 변경됩니다. 리소스를 고유하게 식별할 충분한 수의 등록 정보 이름-값 쌍을 지정해야 합니다.

`clear` 선택적 설정의 값을 지웁니다. 필수 설정은 지울 수 없습니다. 하지만, 일부 설정은 새 값을 지정하여 변경할 수 있습니다.

`remove` 전역 범위에서 지정된 리소스 유형을 제거합니다. 리소스를 고유하게 식별할 충분한 개수의 등록 정보 이름-값 쌍을 지정해야 합니다. 등록 정보 이름-값 쌍을 지정하지 않은 경우 모든 인스턴스가 제거됩니다. 두 개 이상 존재할 경우 `-F` 옵션을 사용하지 않는 한 확인이 필요합니다.

리소스 범위에서 현재 리소스로부터 지정된 등록 정보 이름-등록 정보 값을 제거합니다.

`end` 리소스 범위에서만 사용할 수 있습니다. 리소스 지정을 종료합니다.

그러면 `zonecfg` 명령이 현재 리소스가 완전히 지정되었음을 확인합니다.

- 리소스가 완전히 지정된 경우 메모리 내 구성에 추가되고 범위가 전역으로 되돌려집니다.
- 지정이 완전하지 않으면 시스템에서 수행해야 할 작업을 설명하는 오류 메시지가 표시됩니다.

`cancel` 리소스 범위에서만 사용할 수 있습니다. 리소스 지정을 종료하고 범위를 전역으로 재설정합니다. 부분적으로 지정된 리소스는 유지되지 않습니다.

`delete` 지정된 구성을 완전히 삭제합니다. 구성을 메모리와 안정된 저장소에서 모두 삭제합니다. `-F`(강제) 옵션을 `delete`와 함께 사용해야 합니다.

주의 – 이 작업은 즉시 수행됩니다. 완결이 필요하지 않으며 삭제된 영역을 되돌릴 수 없습니다.

`info` 전역 리소스 등록 정보 `zonepath`, `autoboot` 및 `pool` 또는 현재 구성에 대한 정보를 표시합니다. 리소스 유형이 지정된 경우 해당 유형의 리소스에 대해서만 정보를 표시합니다. 리소스 범위에서 이 하위 명령은 추가되거나 수정되는 리소스에만 적용됩니다.

`verify` 현재 구성의 정확성을 확인합니다. 모든 리소스에 모든 필수 등록 정보가 지정되었는지 확인합니다.

`commit` 메모리의 현재 구성을 안정된 저장소로 완결합니다. 메모리 내 구성이 완료될 때까지 `revert` 하위 명령을 사용하여 변경을 제거할 수 있습니다. `zoneadm`에 사용될 구성을 완결해야 합니다. `zonecfg` 세션을 완료할 때 이 작업이 자동으로 시도됩니다. 올바른 구성만 완결할 수 있으므로 완결 작업이 자동으로 확인을 수행합니다.

`revert` 구성을 마지막 완결 상태로 되돌립니다.

`exit` `zonecfg` 세션을 종료합니다. `exit`를 `-F`(강제) 옵션과 함께 사용할 수 있습니다.

`commit`은 필요할 경우 자동으로 시도됩니다. EOF 문자를 사용하여 세션을 종료할 수도 있습니다.

zonecfg 명령 파일 모드

명령 파일 모드에서는 파일에서 입력을 받습니다. 이 파일을 생성하는 데 220 페이지 “zonecfg 대화식 모드”에 설명된 `export` 하위 명령이 사용됩니다. 구성을 표준 출력으로 인쇄하거나 `-f` 옵션을 사용하여 출력 파일을 지정할 수 있습니다.

영역 구성 데이터

영역 구성 데이터는 리소스와 등록 정보라는 두 종류의 엔티티로 구성됩니다. 각 리소스마다 유형이 지정되어 있으며 각 리소스에는 하나 이상의 등록 정보 집합이 있을 수도 있습니다. 등록 정보마다 이름과 값이 있습니다. 등록 정보 집합은 리소스 유형에 따라 달라집니다.

필수 등록 정보는 `zonename`과 `zonepath`뿐입니다.

리소스 유형과 등록 정보

다음은 각 리소스와 등록 정보 유형에 대한 설명입니다.

`zonename` 영역 이름입니다. 영역 이름에는 다음 규칙이 적용됩니다.

- 각 영역마다 고유한 이름이 지정되어야 합니다.

- 영역 이름은 대/소문자를 구분합니다.
- 영역 이름은 영숫자 문자로 시작해야 합니다.

  이름에는 영숫자 문자, 밑줄(_), 하이픈(-) 및 마침표(.)를 사용할 수 있습니다.
- 이름은 63자를 초과할 수 없습니다.
- `global`이라는 이름과 `SYS`로 시작하는 모든 이름은 예약되어 있으므로 사용할 수 없습니다.

`zonepath`

`zonepath` 등록 정보는 영역이 설치될 경로를 지정합니다. 각 영역에는 전역 영역의 루트 디렉토리를 기준으로, 해당 루트 디렉토리에 대한 경로가 있습니다. 설치 시 전역 영역 디렉토리의 표시 여부를 제한해야 합니다. 영역 경로는 `root`와 `700` 모드를 함께 사용하여 소유해야 합니다. 영역 경로는 존재하지 않을 경우 설치 시 자동으로 만들어집니다. 권한이 올바르지 않으면 자동으로 수정됩니다.

비전역 영역의 루트 경로는 한 레벨 낮습니다. 영역의 루트 디렉토리에는 전역 영역의 루트 디렉토리(`/`)와 동일한 소유권과 권한이 부여됩니다. 영역 디렉토리는 `root`와 `755` 모드를 함께 사용하여 소유해야 합니다. 이 계층을 사용하면 전역 영역의 권한이 없는 사용자가 비전역 영역의 파일 시스템을 탐색할 수 없습니다.

영역이 ZFS 데이터 집합에 있어야 합니다. ZFS 데이터 집합은 영역이 설치되거나 연결될 때 자동으로 만들어집니다. ZFS 데이터 집합을 만들 수 없을 경우 영역이 설치되거나 연결되지 않습니다.

경로	설명
`/zones/my-zone`	`zonecfg zonepath`
`/zones/my-zone/root`	영역 루트

자세한 내용은 327 페이지 “파일 시스템 탐색”을 참조하십시오.

주 – `zoneadm` 명령의 `move` 하위 명령을 사용하여 새로운 전체 `zonepath`를 지정하면 영역을 동일한 시스템의 다른 위치로 이동할 수 있습니다. 자세한 내용은 279 페이지 “비전역 영역 이동”을 참조하십시오.

autoboot

이 등록 정보가 true로 설정되면 전역 영역이 부트될 때 해당 영역이 자동으로 부트됩니다. 기본적으로는 false로 설정됩니다. 영역 서비스 svc:/system/zones:default를 사용 안함으로 설정하면 이 등록 정보의 설정과 상관없이 영역이 자동으로 부트됩니다. svcadm(1M) 매뉴얼 페이지에 설명된 svcadm 명령을 사용하여 영역 서비스를 사용으로 설정할 수 있습니다.

```
global# svcadm enable zones
```

pkg 업데이트 중에 이 설정에 대한 자세한 내용은 313 페이지 "영역 패키징 개요"를 참조하십시오.

bootargs

이 등록 정보는 영역의 부트 인수를 설정하는 데 사용됩니다. reboot, zoneadm boot 또는 zoneadm reboot 명령에 의해 대체되지 않을 경우 부트 인수가 적용됩니다. 264 페이지 "영역 부트 인수"를 참조하십시오.

pool

이 등록 정보는 시스템의 리소스 풀과 영역을 연결하는 데 사용됩니다. 여러 영역이 풀 하나의 리소스를 공유할 수 있습니다. 208 페이지 "dedicated-cpu 리소스"를 참조하십시오.

limitpriv

이 등록 정보는 기본값 이외의 권한 마스크를 지정하는 데 사용됩니다. 337 페이지 "비전역 영역의 권한"을 참조하십시오.

선행 priv_를 지정하든 지정하지 않든 권한 이름을 지정하면 권한이 추가됩니다. 이름 앞에 대시(-) 또는 느낌표(!)를 붙이면 권한이 제외됩니다. 권한 값은 쉼표로 구분되며 따옴표(") 안에 지정됩니다.

priv_str_to_set(3C)에 설명된 바와 같이 none, all 및 basic의 특수 권한 집합이 일반적인 정의로 확장됩니다. 전역 영역에서 영역 구성을 수행하므로 특수 권한 집합 zone을 사용할 수 없습니다. 일반적인 사용은 특정 권한을 추가하거나 제거하여 기본 권한 집합을 변경하는 것이므로 특수 집합 default가 기본 권한 집합으로 매핑됩니다. default가 limitpriv 등록 정보의 시작 부분에 나타나면 기본 집합으로 확장됩니다.

다음 항목은 영역에서 dtrace_proc 및 dtrace_user 권한만 필요로 하는 DTrace 프로그램을 사용할 수 있는 기능을 추가합니다.

```
global# zonecfg -z userzone
zonecfg:userzone> set limitpriv="default,dtrace_proc,dtrace_user"
```

영역의 권한 집합에 허용되지 않은 권한만 들어 있거나 필요한 권한이 없거나, 알 수 없는 권한이 포함된 경우 영역을 확인, 준비 또는 부트하려는 시도가 실패하며 오류 메시지가 표시됩니다.

scheduling-class	이 등록 정보는 영역의 예약 클래스를 설정합니다. 자세한 내용과 팁은 209 페이지 “예약 클래스”를 참조하십시오.
ip-type	이 등록 정보는 모든 비전역 영역에 대해 설정해야 합니다. 212 페이지 “배타적 IP 비전역 영역”, 211 페이지 “공유 IP 비전역 영역” 및 243 페이지 “영역 구성 방법”을 참조하십시오.
dedicated-cpu	이 리소스는 영역이 실행 중인 동안 시스템의 일부 프로세서를 해당 영역 전용으로 사용합니다. dedicated-cpu 리소스는 ncpus 및 importance(옵션)에 대한 제한을 제공합니다. 자세한 내용은 208 페이지 “dedicated-cpu 리소스”를 참조하십시오.
capped-cpu	이 리소스는 영역이 실행 중인 동안 사용될 수 있는 CPU 리소스 양에 대한 제한을 설정합니다. capped-cpu 리소스는 ncpus에 대한 제한을 제공합니다. 자세한 내용은 208 페이지 “capped-cpu 리소스”를 참조하십시오.
capped-memory	이 리소스는 영역에 대한 메모리를 제한할 때 사용되는 등록 정보를 그룹화합니다. capped-memory 리소스는 physical, swap 및 locked 메모리에 대한 제한을 제공합니다. 이러한 등록 정보 중 하나 이상을 지정해야 합니다. 전역 영역에 capped-memory 리소스를 사용하려면 service/resource-cap 패키지를 설치해야 합니다.
anet	anet 리소스는 영역이 부트될 때 배타적 IP 영역의 임시 VNIC 인터페이스를 자동으로 만들고 영역이 정지할 때 해당 VNIC를 삭제합니다.
net	net 리소스는 전역 영역의 기존 네트워크 인터페이스를 비전역 영역에 지정합니다. 네트워크 인터페이스 리소스는 인터페이스 이름입니다. 각 영역에는 영역이 설치됨 상태에서 준비 상태로 전환될 때 설정되는 네트워크 인터페이스가 있을 수 있습니다.
dataset	ZFS 데이터 세트 리소스를 추가하면 저장소 관리를 비전역 영역으로 위임할 수 있습니다. 위임된 데이터 세트가 파일 시스템인 경우 영역 관리자가 데이터 세트 내 파일 시스템을 생성 및 완전 삭제할 수 있으며 데이터 세트의 등록 정보를 수정할 수 있습니다. 영역 관리자는 스냅샷, 자식 파일 시스템 및 볼륨, 그리고 그 후손의 복제본을 만들 수 있습니다. 위임된 데이터 세트가 볼륨인 경우 영역 관리자가 등록 정보를 설정하고 스냅샷을 만들 수 있습니다. 영역 관리자는 영역에 지정된 데이터 세트에 설정되어 있는 최상위 할당량을 초과하거나 영역에 추가되지 않은 데이터 세트에 영향을 줄 수 없습니다. 데이터 세트가 비전역 영역으로 위임된 후에는 zoned 등록 정보가 자동으로 설정됩니다. zoned 파일 시스템은 영역 관리자가 마운트 지점을 허용할 수 없는 값으로 설정해야 할 수 있으므로 전역 영역에서 마운트할 수 없습니다.

다음과 같은 방식으로 ZFS 데이터 세트를 영역에 추가할 수 있습니다.

- lofs 마운트된 파일 시스템으로(전역 영역과의 공간 공유가 유일한 목표인 경우)
- 위임된 데이터 세트로

Oracle Solaris 관리: ZFS 파일 시스템의 10 장, "Oracle Solaris ZFS 고급 주제" 및 322 페이지 "파일 시스템 및 비전역 영역"을 참조하십시오.

데이터 세트 문제에 대한 자세한 내용은 28 장, "그 밖의 기타 Oracle Solaris Zones 문제 해결"을 참조하십시오.

`fs`

각 영역에는 영역이 설치된 상태에서 준비 상태로 전환될 때 마운트되는 다양한 파일 시스템이 있을 수 있습니다. 파일 시스템 리소스는 파일 시스템 마운트 지점의 경로를 지정합니다. 영역 내 파일 시스템 사용에 대한 자세한 내용은 322 페이지 "파일 시스템 및 비전역 영역"을 참조하십시오.

주 – `fs` 리소스를 통해 비전역 영역에서 UFS 파일 시스템을 사용하려면 설치 후에 또는 AI 매니페스트 스크립트를 통해 `system/file-system/ufs` 패키지를 영역 안에 설치해야 합니다.

`quota(1M)`에 설명된 `quota` 명령을 사용하여 `fs` 리소스를 통해 추가된 UFS 파일 시스템의 할당량 정보를 검색할 수 없습니다.

`fs-allowed`

이 등록 정보를 설정하면 영역 관리자는 자신이 생성했거나 NFS를 사용하여 가져온 해당 유형의 파일 시스템을 마운트하고 해당 파일 시스템을 관리할 수 있습니다. `fs-allowed` 등록 정보를 통해 실행 중인 영역 내 파일 시스템 마운트 권한도 제한됩니다. 기본적으로 영역 내에서는 `hsfs` 파일 시스템 및 네트워크 파일 시스템(예: NFS)의 마운트만 허용됩니다.

이 등록 정보는 영역 안에 위임된 ZVOL 장치나 블록 장치와 함께 사용할 수도 있습니다.

`fs-allowed` 등록 정보는 `ufs,pcfs`와 같은 영역 내에서 마운트할 수 있는 쉼표로 구분된 추가 파일 시스템 목록을 받습니다.

```
zonecfg:my-zone> set fs-allowed=ufs,pcfs
```

이 등록 정보는 `add fs` 또는 `add dataset` 등록 정보를 통해 전역 영역에서 관리되는 영역 마운트에만 적용됩니다.

보안 고려 사항은 322 페이지 "파일 시스템 및 비전역 영역" 및 333 페이지 "비전역 영역에서 장치 사용"을 참조하십시오.

device

장치 리소스는 장치와 일치하는 지정자입니다. 각 영역에는 영역이 설치됨 상태에서 준비 상태로 전환될 때 구성해야 하는 장치가 있을 수 있습니다.

주 – device 리소스를 통해 비전역 영역에서 UFS 파일 시스템을 사용하려면 system/file-system/ufs 패키지를 설치 후에 또는 AI 매니페스트 스크립트를 통해 영역 안에 설치해야 합니다.

---

rctl

rctl 리소스는 영역 전체 리소스 제어용으로 사용됩니다. 컨트롤은 영역이 설치됨 상태에서 준비 상태로 전환될 때 사용으로 설정됩니다.

자세한 내용은 215 페이지 "영역 전체 리소스 제어 설정"을 참조하십시오.

주 – rctl 리소스 대신에 zonefig의 set *global_property_name* 하위 명령을 사용하여 영역 전체 컨트롤을 구성하려면 243 페이지 "영역 구성 방법"을 참조하십시오.

---

hostid

전역 영역의 호스트 ID와 다른 호스트 ID는 hostid 등록 정보를 사용하여 설정할 수 있습니다.

attr

이 일반 속성은 사용자 설명이나 다른 부속 시스템용으로 사용할 수 있습니다. attr의 name 등록 정보는 영숫자 문자로 시작되어야 합니다. name 등록 정보에는 영숫자 문자, 하이픈(-) 및 마침표(.)가 포함될 수 있습니다. zone.로 시작하는 속성 이름은 시스템용으로 예약되어 있습니다.

리소스 유형 등록 정보

리소스에도 구성할 등록 정보가 있습니다. 다음 등록 정보는 표시된 리소스 유형과 연결됩니다.

admin

지정된 영역에 대한 해당 사용자의 권한과 사용자 이름을 정의합니다.

```
zonecfg:my-zone> add admin
zonecfg:my-zone:admin> set user=zadmin
zonecfg:my-zone:admin> set auths=login,manage
zonecfg:my-zone:admin> end
```

다음 값은 auths 등록 정보에 사용할 수 있습니다.

- login(solaris.zone.login)
- manage(solaris.zone.manage)
- clone(solaris.zone.clonefrom)

이러한 auths 등록 정보를 사용하면 영역을 생성할 수 없습니다. 이 기능은 영역 보안 프로파일에 포함되어 있습니다.

dedicated-cpu — ncpus, importance

CPU 수와 풀의 상대적 중요성(옵션)을 지정합니다. 다음 예는 my-zone 영역에서 사용할 CPU 범위를 지정합니다. importance도 설정되어 있습니다.

```
zonecfg:my-zone> add dedicated-cpu
zonecfg:my-zone:dedicated-cpu> set ncpus=1-3
zonecfg:my-zone:dedicated-cpu> set importance=2
zonecfg:my-zone:dedicated-cpu> end
```

capped-cpu — ncpus

CPU 수를 지정합니다. 다음 예에서는 my-zone 영역에 대해 CPU 상한값을 3.5개 CPU로 지정합니다.

```
zonecfg:my-zone> add capped-cpu
zonecfg:my-zone:capped-cpu> set ncpus=3.5
zonecfg:my-zone:capped-cpu> end
```

capped-memory — physical, swap, locked

my-zone 영역에 대한 메모리 제한을 지정합니다. 각 제한은 선택 사항이지만 하나 이상의 제한을 설정해야 합니다.

```
zonecfg:my-zone> add capped-memory
zonecfg:my-zone:capped-memory> set physical=50m
zonecfg:my-zone:capped-memory> set swap=100m
zonecfg:my-zone:capped-memory> set locked=30m
zonecfg:my-zone:capped-memory> end
```

capped-memory 리소스를 사용하려면 전역 영역에 resource-cap 패키지를 설치해야 합니다.

fs — dir, special, raw, type, options

fs 리소스 매개 변수는 파일 시스템을 마운트하는 방법과 위치를 결정하는 값을 제공합니다. fs 매개 변수는 다음과 같이 정의됩니다.

dir — 파일 시스템의 마운트 지점을 지정합니다.

special — 전역 영역에서 마운트할 블록 특수 장치 이름 또는 디렉토리를 지정합니다.

raw — 파일 시스템(ZFS에는 해당되지 않음)을 마운트하기 전에 fsck을 실행할 원시 장치를 지정합니다.

type — 파일 시스템 유형을 지정합니다.

options — mount 명령을 사용하여 찾은 것과 유사한 마운트 옵션을 지정합니다.

다음 예의 코드는 전역 영역 내 pool1/fs1 이름의 데이터 세트가, 구성되는 영역에서 /shared/fs1로 마운트되도록 지정합니다. 사용할 파일 시스템 유형은 ZFS입니다.

```
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/shared/fs1
zonecfg:my-zone:fs> set special=pool1/fs1
zonecfg:my-zone:fs> set type=zfs
zonecfg:my-zone:fs> end
```

매개 변수에 대한 자세한 내용은 322 페이지 “-o nosuid 옵션”, 324 페이지 “보안 제한 및 파일 시스템 동작”과 fsck(1M) 및 mount(1M) 매뉴얼 페이지를 참조하십시오. 또한 섹션 1M 매뉴얼 페이지는 특정 파일 시스템에 고유한 마운트 옵션에도 사용할 수 있습니다. 이러한 매뉴얼 페이지의 이름은 mount_ **파일 시스템** 형태로 되어 있습니다.

주 – quota(1M)에 설명된 quota 명령을 사용하여 이 리소스를 통해 추가된 UFS 파일 시스템의 할당량 정보를 검색할 수 없습니다.

---

dataset name, alias — name

다음 예의 코드는 데이터 집합 *sales* 가 비전역 영역에서 표시되고 마운트되며 전역 영역에서는 더 이상 표시되지 않도록 지정합니다.

```
zonecfg:my-zone> add dataset
zonecfg:my-zone> set name=tank/sales
zonecfg:my-zone> end
```

위임된 데이터 집합에는 다음 예에 표시된 것처럼 기본값이 아닌 별칭이 있을 수 있습니다. 데이터 집합 별칭에는 슬래시(/)를 사용할 수 없습니다.

```
zonecfg:my-zone> add dataset
zonecfg:my-zone:dataset> set name=tank/sales
zonecfg:my-zone:dataset> set alias=data
zonecfg:my-zone:dataset> end
```

기본 별칭으로 되돌리려면 clear alias를 사용합니다.

```
zonecfg:my-zone> clear alias
```

anet

linkname, lower-link, allowed-address, configure-allowed-address, defrouter, mac-address, mac-slot, mac-prefix, mtu, maxbw, priority, vlan-id, rxfanout, rxrings, txrings, link-protection, allowed-dhcp-cids, bandwidth-limit

anet 리소스는 영역이 부트될 때 자동 VNIC 인터페이스를 생성하고 영역이 정지할 때 VNIC를 삭제합니다. 리소스 등록 정보는 zonecfg 명령을 통해 관리됩니다. 사용할 수 있는 등록 정보에 대한 전체 텍스트는 zonecfg(1M) 매뉴얼 페이지를 참조하십시오.

lower-link
: 생성해야 할 기본 링크를 지정합니다. auto로 설정한 경우 영역이 부트될 때마다 zoneadmd 데몬이 VNIC이 생성될 링크를 자동으로 선택합니다.

linkname
: 자동 생성된 VNIC의 이름을 지정합니다.

mac-address
: 지정된 값이나 키워드에 따라 VNIC MAC 주소를 설정합니다. 키워드가 아닌 값은 단일 유형 변환 MAC 주소로 해석됩니다. 지원되는 키워드는 zonecfg(1M) 매뉴얼 페이지를 참조하십시오. 임의 MAC 주소를 선택한 경우 영역 부트, 영역 분리 및 연결 작업 간에 생성된 주소가 유지됩니다.

allowed-address
: 배타적 IP 영역의 IP 주소를 구성하고 배타적 IP 영역에 사용할 수 있는 구성 가능한 IP 주소 집합을 제한합니다. 여러 주소를 지정하려면 쉼표로 구분된 IP 주소를 사용합니다.

defrouter
: 비전역 영역과 전역 영역이 개별 네트워크에 상주하는 경우 defrouter 등록 정보를 사용하여 기본 경로를 설정할 수 있습니다.

  defrouter 등록 정보가 설정된 모든 영역은 전역 영역에 대해 구성되지 않은 서브넷에 있어야 합니다.

zonecfg 명령이 SYSdefault 템플리트를 사용하여 영역을 생성하는 경우 다음 등록 정보를 가진 anet 리소스가 영역 구성에 자동으로 포함됩니다. linkname은 물리적 이더넷 링크를

통해 자동으로 만들어지고 net *N*, net*0* 형태의 첫번째 사용 가능한 이름으로 설정합니다. 기본값을 변경하려면 zonecfg 명령을 사용합니다.

기본값은 물리적 이더넷 링크(예: nxge0)를 통해 자동 VNIC를 만들고 팩토리 MAC 주소를 VNIC에 지정합니다. 선택적 lower-link 등록 정보는 기본 링크인 nxge0으로 설정됩니다. 자동 VNIC은 이 링크를 통해 생성되어야 합니다. 링크 이름, 기본 물리적 링크, MAC 주소, 대역폭 제한과 같은 VNIC 등록 정보와 그 밖의 VNIC 등록 정보는 zonecfg 명령을 사용하여 지정할 수 있습니다. ip-type=exclusive도 지정해야 합니다.

```
zonecfg:my-zone> set ip-type=exclusive
zonecfg:my-zone:anet> add anet
zonecfg:my-zone:anet> set linkname=net0
zonecfg:my-zone:anet> set lower-link=auto
zonecfg:my-zone:anet> set mac-address=random
zonecfg:my-zone:anet> set link-protection=mac-nospoof
zonecfg:my-zone:anet> end
```

등록 정보에 대한 자세한 내용은 zonecfg(1M) 매뉴얼 페이지를 참조하십시오. 링크 등록 정보에 대한 자세한 내용은 dladm(1M) 매뉴얼 페이지를 참조하십시오.

net — address, allowed-addressphysical , defrouter

주 – 공유 IP 영역의 경우 IP 주소와 물리적 장치를 둘 다 지정해야 합니다. 또는 기본 라우터를 설정할 수도 있습니다.

배타적 IP 영역의 경우에는 물리적 인터페이스만 지정하면 됩니다.

- allowed-address 등록 정보는 배타적 IP 영역에 사용될 수 있는 구성 가능한 IP 주소 집합을 제한합니다.
- 비전역 영역과 전역 영역이 개별 네트워크에 상주하는 경우 defrouter 등록 정보를 사용하여 기본 경로를 설정할 수 있습니다.
- defrouter 등록 정보가 설정된 모든 영역은 전역 영역에 대해 구성되지 않은 서브넷에 있어야 합니다.
- 기본 라우터를 사용하는 영역의 트래픽은 대상 영역으로 들어가기 전에 라우터를 거칩니다.

다양한 서브넷에 공유 IP 영역이 존재하는 경우 전역 영역에서 데이터 링크를 구성하지 마십시오.

공유 IP 영역에 대한 다음 예에서는 IP 주소가 192.168.0.1인 영역에 물리적 인터페이스인 nge0이 추가됩니다. 시스템에서 네트워크 인터페이스를 나열하려면 다음을 입력합니다.

```
global# ipadm show-if -po ifname,class,active,persistent
lo0:loopback:yes:46--
nge0:ip:yes:----
```

루프백 행이 아닌 각각의 출력 행에는 네트워크 인터페이스의 이름이 있습니다. 설명에 loopback을 포함하는 행은 카드에 적용되지 않습니다. 46 지속성 플래그는 인터페이스가 전역 영역에서 지속적으로 구성됨을 나타냅니다. yes 활성 값은 인터페이스가 현재 구성되어 있음을 나타내며, ip의 클래스 값은 nge0이 비루프백 인터페이스임을 나타냅니다. 해당 영역에 대한 기본 경로는 10.0.0.1로 설정됩니다. 필요에 따라 defrouter 등록 정보를 설정할 수 있습니다. ip-type=shared는 필수입니다.

```
zonecfg:my-zone> set ip-type=shared
zonecfg:my-zone> add net
zonecfg:my-zone:net> set physical=nge0
zonecfg:my-zone:net> set address=192.168.0.1
zonecfg:my-zone:net> set defrouter=10.0.0.1
zonecfg:my-zone:net> end
```

배타적 IP 영역에 대한 다음 예에서는 bge32001 링크가 bge1에 있는 VLAN인 물리적 인터페이스에 사용됩니다. 사용할 수 있는 데이터 링크를 확인하려면 dladm show-link를 사용합니다. allowed-address 등록 정보는 영역에서 사용할 수 있는 IP 주소를 제한합니다. defrouter 등록 정보는 기본 경로를 설정하는 데 사용됩니다. ip-type=exclusive도 지정해야 합니다.

```
zonecfg:my-zone> set ip-type=exclusive
zonecfg:my-zone> add net
zonecfg:myzone:net> set allowed-address=11.1.1.32/24
zonecfg:my-zone:net> set physical=bge32001
zonecfg:myzone:net> set defrouter=11.1.1.1
zonecfg:my-zone:net> end
```

물리적 장치 유형만 add net 단계에서 지정됩니다. physical 등록 정보는 **Oracle Solaris 관리: 네트워크 인터페이스 및 네트워크 가상화**의 제III부, "네트워크 가상화 및 리소스 관리"에서 설명한 대로 VNIC일 수 있습니다.

주 – Oracle Solaris 운영 체제에서는 모든 이더넷 유형의 인터페이스를 지원하므로 이러한 인터페이스의 데이터 링크를 dladm 명령으로 관리할 수 있습니다.

---

device

match, allow-partition, allow-raw-io

일치시킬 장치 이름은 일치시켜야 할 패턴이거나 절대 경로일 수 있습니다. allow-partition 및 allow-raw-io 둘 다 true 또는 false로 설정할 수 있습니다. 기본값은 false입니다. allow-partition은 분할을 사용으로 설정합니다. allow-raw-io는 uscsi를 사용으로 설정합니다. 이러한 리소스에 대한 자세한 내용은 zonecfg(1M)을 참조하십시오.

다음 예에서는 디스크 장치에 대한 uscsi 작업이 영역 구성 안에 포함됩니다.

```
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/*dsk/cXtYdZ*
zonecfg:my-zone:device> set set allow-raw-io=true
zonecfg:my-zone:device> end
```

주의 – 장치를 추가하기 전에 333 페이지 "비전역 영역에서 장치 사용", 335 페이지 "비전역 영역에서 실행 중인 응용 프로그램" 및 337 페이지 "비전역 영역의 권한"에서 제한 사항과 보안 고려 사항을 참조하십시오.

rctl

name, value

다음과 같은 영역 전체 리소스 제어를 사용할 수 있습니다.

- zone.cpu-cap
- zone.cpu-shares(기본: cpu-shares)
- zone.max-locked-memory
- zone.max-lofi
- zone.max-lwps(기본: max-lwps)
- zone.max-msg-ids(기본: max-msg-ids)
- zone.max-processes(기본: max-processes
- zone.max-sem-ids(기본: max-sem-ids)
- zone.max-shm-ids(기본: max-shm-ids)
- zone.max-shm-memory(기본: max-shm-memory)
- zone.max-swap

영역 전체 리소스 제어를 설정하기 위한 보다 간단한 기본적인 방법은 243 페이지 "영역 구성 방법"에서처럼 rctl 리소스 대신에 등록 정보 이름을 사용하는 것입니다. add rctl을 사용하여 영역 내 영역 전체 리소스 제어 항목을 구성한 경우 형식이 project 데이터베이스의 리소스 제어 항목과는

다릅니다. 영역 구성에서 rctl 리소스 유형은 세 개의 이름/값 쌍으로 구성됩니다. 이름은 priv, limit 및 action입니다. 각 이름은 단순 값을 받습니다.

```
zonecfg:my-zone> add rctl
zonecfg:my-zone:rctl> set name=zone.cpu-shares
zonecfg:my-zone:rctl> add value (priv=privileged,limit=10,action=none)
zonecfg:my-zone:rctl> end

zonecfg:my-zone> add rctl
zonecfg:my-zone:rctl> set name=zone.max-lwps
zonecfg:my-zone:rctl> add value (priv=privileged,limit=100,action=deny)
zonecfg:my-zone:rctl> end
```

리소스 제어 및 속성에 대한 일반적인 정보는 6 장, “리소스 제어(개요)”과 335 페이지 “비전역 영역에서 사용되는 리소스 제어”를 참조하십시오.

attr — name, type, value

다음 예에서는 영역에 대한 설명이 추가됩니다.

```
zonecfg:my-zone> add attr
zonecfg:my-zone:attr> set name=comment
zonecfg:my-zone:attr> set type=string
zonecfg:my-zone:attr> set value="Production zone"
zonecfg:my-zone:attr> end
```

export 하위 명령을 사용하여 영역 구성을 표준 출력으로 인쇄할 수 있습니다. 구성은 명령 파일에서 사용할 수 있는 형태로 저장됩니다.

명령줄 편집 라이브러리

Tecla 명령줄 편집 라이브러리는 zonecfg 명령에 사용하기 위한 것입니다. 이 라이브러리는 명령줄 내역 및 편집 지원 방식을 제공합니다.

자세한 내용은 tecla(5) 매뉴얼 페이지를 참조하십시오.

◆ ◆ ◆ 17 장

17 비전역 영역 계획 및 구성(작업)

이 장에서는 시스템의 영역을 구성하기 위해 먼저 수행해야 하는 작업에 대해 설명합니다. 또한 영역을 구성하고, 영역 구성을 수정하며, 시스템에서 영역 구성을 삭제하는 방법에 대해서도 설명합니다.

영역 구성 프로세스에 대한 개요는 16 장, “비전역 영역 구성(개요)”을 참조하십시오.

`solaris10` 브랜드 영역 구성에 대한 자세한 내용은 제3부를 참조하십시오.

비전역 영역 계획 및 구성(작업 맵)

시스템에서 영역을 사용하도록 설정하려면 먼저 정보를 수집하고 영역을 어떻게 구성할지를 결정해야 합니다. 다음 작업 맵은 영역을 계획하고 구성하는 방법을 요약하여 보여 줍니다.

작업	설명	수행 방법
영역 전략을 계획합니다.	■ 시스템에서 실행 중인 응용 프로그램을 평가하여 영역에서 실행하려는 응용 프로그램을 결정합니다. ■ 영역에 고유한 파일을 저장하기 위해 디스크 공간의 가용성을 평가합니다. ■ 리소스 관리 기능도 사용할 경우 리소스 관리 경계에 맞춰 영역을 조정할 방법을 결정합니다. ■ 리소스 풀을 사용할 경우 필요하면 풀을 구성합니다.	과거의 용례를 참조하십시오. 또한 238 페이지 “디스크 공간 요구 사항” 및 135 페이지 “영역에서 사용되는 리소스 풀”을 참조하십시오.
영역의 이름을 결정합니다.	이름 지정 규약에 따라 영역을 어떻게 지칭할지 결정합니다.	222 페이지 “영역 구성 데이터” 및 239 페이지 “영역 호스트 이름”을 참조하십시오.
영역 경로를 정의합니다(필수).	각 영역에는 전역 영역의 루트 디렉토리를 기준으로, 해당 루트 디렉토리에 대한 경로가 있습니다.	222 페이지 “영역 구성 데이터”를 참조하십시오.
리소스 풀을 구성하지 않을 경우 CPU 제한에 대한 필요성을 평가합니다. `zonecfg` 명령에 사양을 지정할 경우 마이그레이션 중에 풀 설정이 전파됩니다.	해당 응용 프로그램 요구 사항을 검토합니다.	208 페이지 “`dedicated-cpu` 리소스”를 참조하십시오.
전역 영역에서 `rcapd`를 사용하여 영역의 메모리 상한값을 설정할 계획일 경우 메모리 할당의 필요성을 평가합니다.	해당 응용 프로그램 요구 사항을 검토합니다.	10 장, “리소스 상한값 지원 데몬을 사용한 물리적 메모리 제어(개요)”, 11 장, “리소스 상한값 지원 데몬 관리(작업)” 및 209 페이지 “물리적 메모리 제어 및 `capped-memory` 리소스”를 참조하십시오.
시스템의 기본 스케줄러를 FSS로 설정합니다.	각 영역에 CPU를 분배하여 CPU 리소스에 대한 영역의 권한을 제어합니다. FSS로 설정하면 할당된 분배를 기준으로 영역 간에 CPU 리소스가 공정하게 분산됩니다.	8 장, “FSS(Fair Share Scheduler)(개요)”, 209 페이지 “예약 클래스”.

작업	설명	수행 방법
영역의 기본 유형이 배타적 IP인지 확인합니다.	anet 리소스로 구성된 배타적 IP 영역의 경우 영역이 부트될 때마다 자동으로 VNIC가 생성됩니다. net 리소스로 구성된 배타적 IP 영역의 경우 영역에 지정될 데이터 링크를 결정합니다. 영역은 하나 이상의 네트워크 인터페이스에 배타적으로 액세스할 수 있어야 합니다. 이 인터페이스는 VNIC가 될 수도 있고, bge1과 같은 별도의 LAN 또는 bge2000과 같은 별도의 VLAN이 될 수도 있습니다. **Oracle Solaris 관리: 네트워크 인터페이스 및 네트워크 가상화**의 제III부, "네트워크 가상화 및 리소스 관리"를 참조하십시오. 공유 IP 영역을 구성할 경우 영역의 IP 주소를 받거나 구성합니다. 구성에 따라 네트워크로 액세스하려는 각각의 비전역 영역에 대해 적어도 한 개의 IP 주소를 받아야 합니다.	239 페이지 "영역 호스트 이름과 네트워크 요구 사항 정의", 243 페이지 "영역 구성 방법" 및 **Oracle Solaris 관리: IP 서비스**를 참조하십시오.
영역에 마운트하려는 파일 시스템을 결정합니다.	해당 응용 프로그램 요구 사항을 검토합니다.	자세한 내용은 214 페이지 "영역에서 마운트된 파일 시스템"을 참조하십시오.
영역에서 사용 가능하도록 설정할 네트워크 인터페이스를 결정합니다	해당 응용 프로그램 요구 사항을 검토합니다.	자세한 내용은 330 페이지 "공유 IP 네트워크 인터페이스"를 참조하십시오.
비전역 영역의 기본 권한 집합을 변경해야 할지 여부를 결정합니다.	기본 권한 집합, 추가 및 제거할 수 있는 권한 집합, 이번에는 사용할 수 없는 권한 집합을 확인합니다.	337 페이지 "비전역 영역의 권한"을 참조하십시오.
각 영역에서 구성해야 할 장치를 결정합니다.	해당 응용 프로그램 요구 사항을 검토합니다.	해당 응용 프로그램의 설명서를 참조하십시오.
영역을 구성합니다.	zonecfg를 사용하여 영역에 대한 구성을 만듭니다.	242 페이지 "영역 구성, 확인 및 커밋"을 참조하십시오.
구성된 영역을 확인하고 커밋합니다.	지정된 리소스와 등록 정보가 가정한 시스템에서 유효한지 확인합니다.	242 페이지 "영역 구성, 확인 및 커밋"을 참조하십시오.

현재 시스템 설정 평가

영역은 Oracle Solaris 10 이상 릴리스를 실행하는 시스템에서 사용할 수 있습니다. 다음은 영역 사용을 위해 고려해야 할 기본 시스템 사항입니다.

- 각 영역 내에서 실행하는 응용 프로그램의 성능 요구 사항
- 각 영역 내에 고유한 파일을 저장하기 위한 디스크 공간의 가용성

디스크 공간 요구 사항

영역에서 사용 가능한 디스크 공간에 대한 제한은 없습니다. 공간 제한은 전역 관리자나 적합한 인증을 받은 사용자가 책임집니다. 전역 관리자는 로컬 저장소가 비전역 영역의 루트 파일 시스템을 저장하기에 충분하도록 해야 합니다. 소규모 단일 프로세서 시스템에서도 동시에 여러 개의 영역을 실행할 수 있습니다.

비전역 영역에 설치된 패키지의 특성은 해당 영역의 공간 요구 사항에 영향을 줍니다. 또한 패키지 수도 한 요인이 됩니다.

디스크 요구 사항은 전역 영역에 현재 설치된 패키지 및 설치된 소프트웨어에서 사용하는 디스크 공간에 따라 결정됩니다.

한 영역에는 영역당 최소 150MB의 사용 가능한 디스크 공간이 있어야 합니다. 하지만 전역 영역이 모든 표준 Oracle Solaris 패키지를 포함하여 설치되었을 경우 사용 가능한 디스크 공간은 일반적으로 500MB~1GB입니다. 소프트웨어가 추가될 경우 이 값이 증가할 수 있습니다.

영역당 40MB의 RAM이 권장되지만 충분한 스왑 공간이 있는 시스템에서는 필수 요구 사항이 아닙니다.

영역 크기 제한

ZFS 데이터 집합에서 지원하는 `zonepath`가 있는 영역에 ZFS 데이터 집합 할당량을 사용하여 영역을 제한할 수 있습니다. `zonepath` 데이터 집합에 액세스할 수 있는 관리자는 데이터 집합의 `quota`, `userquota`, `groupquota` 및 `refquota` 등록 정보를 수정하여 각 영역에서 사용할 수 있는 디스크 공간의 최대 크기를 제어할 수 있습니다. 이러한 등록 정보에 대한 설명은 `zfs(1M)` 매뉴얼 페이지에 나와 있습니다.

또한 관리자는 고정 크기의 ZFS 볼륨을 만들고 볼륨의 데이터 집합에 영역을 설치할 수도 있습니다. 볼륨은 해당 볼륨 내에 설치된 영역의 크기를 제한합니다.

영역 호스트 이름과 네트워크 요구 사항 정의

영역의 호스트 이름을 정의해야 합니다. 그런 다음 네트워크에 연결되는 공유 IP 영역에 대해 다음 중 하나를 수행해야 합니다.

- 영역에 IPv4 주소 지정
- 영역에 대해 IPv6 주소 수동 구성 및 지정

배타적 IP 영역 내에서 전역 영역과 마찬가지로 주소를 구성합니다.

배타적 IP 및 공유 IP 유형에 대한 자세한 내용은 210 페이지 "영역 네트워크 인터페이스"를 참조하십시오.

영역 호스트 이름

영역 호스트 이름은 시스템 호스트 이름에 해당하는 영역입니다. 영역 호스트 이름은 전역 영역에서 설정됩니다. 영역 호스트 이름은 일반적으로 영역 이름으로 설정되며 대개 영역의 `/etc/inet/hosts` 파일에 공식 호스트 이름 또는 영역의 IP 주소 중 하나의 별명으로 설정됩니다. 자세한 내용은 nodename(4) 및 hosts(4)를 참조하십시오.

이름 지정 서비스를 위해 로컬 파일을 사용할 경우 `hosts` 데이터베이스가 `/etc/inet/hosts` 파일에 유지됩니다. 영역 네트워크 인터페이스의 호스트 이름은 `/etc/inet/hosts`에 있는 `hosts` 데이터베이스에서 결정됩니다. 또는 공유 IP 영역의 경우 영역을 구성할 때 IP 주소를 직접 지정하여 호스트 이름 풀기(name resolution)를 사용하지 않을 수 있습니다.

자세한 내용은 **Oracle Solaris 관리: IP 서비스**의 "네트워크 구성 파일" 및 **Oracle Solaris 관리: IP 서비스**의 "name-service/switch SMF 서비스"를 참조하십시오.

공유 IP 영역 네트워크 주소

네트워크 연결이 필요한 각각의 공유 IP 영역에는 하나 이상의 고유한 IP 주소가 있습니다. IPv4 및 IPv6 주소가 모두 지원됩니다.

IPv4 영역 네트워크 주소

IPv4를 사용할 경우 주소를 받아 영역에 주소를 지정합니다.

IP 주소에 접두어 길이를 지정할 수도 있습니다. 이 접두어 길이의 형식은 *address*/*prefix-length*(예: `192.168.1.1/24`)입니다. 따라서 사용할 주소는 `192.168.1.1`이며 사용할 넷마스크는 `255.255.255.0` 또는 처음 24비트가 1비트인 마스크입니다.

이름 지정 서비스를 위해 로컬 파일을 사용할 경우 `hosts` 데이터베이스가 `/etc/inet/hosts` 파일에 유지됩니다. 영역 네트워크 인터페이스의 호스트 이름은

/etc/inet/hosts에 있는 hosts 데이터베이스에서 결정됩니다. 또는 공유 IP 영역의 경우 영역을 구성할 때 IP 주소를 직접 지정하여 호스트 이름 풀기(name resolution)를 사용하지 않을 수 있습니다.

자세한 내용은 hosts(4), nodename(4) 및 **Oracle Solaris Administration: IP Services**를 참조하십시오.

IPv6 영역 네트워크 주소

IPv6를 사용할 경우 주소를 수동으로 구성해야 합니다. 일반적으로 적어도 다음 두 가지 유형의 주소를 구성해야 합니다.

링크 로컬 주소
: 링크 로컬 주소는 fe80:: *64-bit interface ID*/10 형식입니다. /10은 10비트의 접두어 길이를 나타냅니다.

서브넷에 구성된 전역 접두어로 구성된 주소
: 전역 유니캐스트 주소는 관리자가 각 서브넷에 구성한 64비트 접두어와 64비트 인터페이스 ID로 구성됩니다. 접두어는 IPv6를 사용하도록 구성된 동일한 서브넷의 아무 시스템에서나 ipadm show-addr 명령을 실행하여 확인할 수 있습니다.

  64비트 인터페이스 ID는 일반적으로 시스템의 MAC 주소에서 파생됩니다. IPv4를 사용하는 영역의 경우 다음과 같이 고유한 대체 주소가 전역 영역의 IPv4 주소에서 파생될 수 있습니다.

  ```
  16 bits of zero:upper 16 bits of IPv4 address:lower 16 bits of IPv4 address:a
  zone-unique number
  ```

  예를 들어 전역 영역의 IPv4 주소가 192.168.200.10일 경우 영역 고유 번호 1을 사용하는 비전역 영역의 적절한 링크 로컬 주소는 fe80::c0a8:c80a:1/10입니다. 해당 서브넷에서 사용되는 전역 접두어가 2001:0db8:aabb:ccdd/64일 경우 동일한 비전역 영역의 고유한 전역 유니캐스트 주소는 2001:0db8:aabb:ccdd::c0a8:c80a:1/64입니다. IPv6 주소를 구성할 경우 접두어 길이를 지정해야 합니다.

링크 로컬 주소 및 전역 유니캐스트 주소에 대한 자세한 내용은 ipadm(1M) 및 inet6(7P) 매뉴얼 페이지를 참조하십시오.

배타적 IP 영역 네트워크 주소

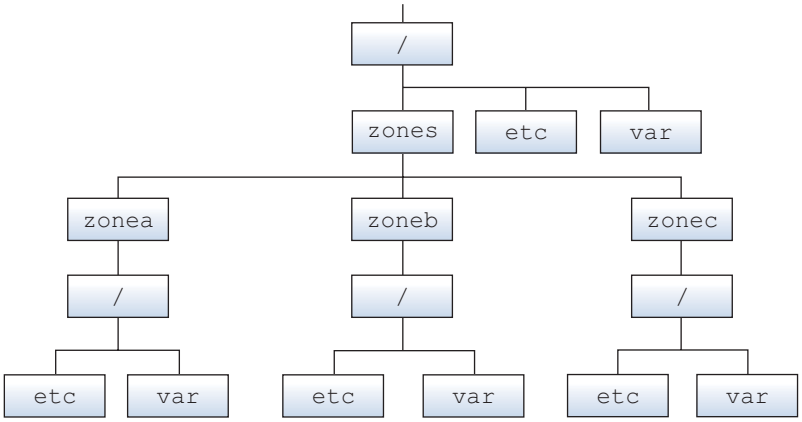
배타적 IP 영역 내에서 전역 영역과 마찬가지로 주소를 구성합니다. DHCP 및 IPv6 Stateless 주소 자동 구성을 사용하여 주소를 구성할 수 있습니다.

파일 시스템 구성

가상 플랫폼을 설정할 때 수행할 많은 마운트를 지정할 수 있습니다. 루프백 가상 파일 시스템(LOFS)을 사용하여 영역에 루프백 마운트된 파일 시스템은 `nodevices` 옵션으로 마운트해야 합니다. `nodevices` 옵션에 대한 자세한 내용은 322 페이지 “파일 시스템 및 비전역 영역”을 참조하십시오.

LOFS를 사용하여 새로운 가상 파일 시스템을 만들면 대체 경로 이름을 사용하여 파일을 액세스할 수 있습니다. 비전역 영역에서 루프백 마운트를 수행하면 파일 시스템 계층이 영역의 루트 아래에 중복된 것처럼 나타납니다. 영역에서 모든 파일은 영역의 루트로 시작하는 경로 이름을 사용하여 액세스할 수 있습니다. LOFS 마운트는 파일 시스템 이름 공간을 유지합니다.

그림 17–1 루프백 마운트된 파일 시스템



자세한 내용은 `lofs`(7S) 매뉴얼 페이지를 참조하십시오.

비전역 영역 구성 만들기, 수정 및 삭제(작업 맵)

작업	설명	수행 방법
비전역 영역을 구성합니다.	zonecfg 명령을 사용하여 영역을 만들고, 구성을 확인하고, 구성을 커밋합니다. 스크립트를 사용하여 시스템에 여러 영역을 구성하고 부트할 수도 있습니다. zonecfg 명령을 사용하여 비전역 영역의 구성을 표시할 수 있습니다.	242 페이지 "영역 구성, 확인 및 커밋", 248 페이지 "여러 영역 구성 스크립트"
영역 구성을 수정합니다.	영역 구성의 리소스 유형을 수정하거나, 영역 이름 등의 등록 정보 유형을 수정하거나, 영역에 전용 장치를 추가하려면 다음 절차를 사용합니다.	253 페이지 "zonecfg 명령을 사용하여 영역 구성 수정"
영역 구성을 되돌리거나 영역 구성을 삭제합니다.	zonecfg 명령을 사용하여 영역 구성에 대해 수행한 리소스 설정을 취소하거나 영역 구성을 삭제합니다.	256 페이지 "zonecfg 명령을 사용하여 영역 구성 되돌리기 또는 제거"
영역 구성을 삭제합니다.	zonecfg 명령과 delete 하위 명령을 함께 사용하여 시스템에서 영역 구성을 삭제합니다.	257 페이지 "영역 구성 삭제 방법"

영역 구성, 확인 및 커밋

zonecfg(1M) 매뉴얼 페이지에서 설명되어 있는 zonecfg 명령은 다음 작업을 수행하는 데 사용됩니다.

- 영역 구성 만들기
- 필요한 모든 정보가 있는지 확인
- 비전역 영역 구성 커밋

또한 zonecfg 명령을 사용하여 전역 영역의 리소스 관리 설정을 영구적으로 지정할 수도 있습니다.

zonecfg 유틸리티를 사용하여 영역을 구성할 때 revert 하위 명령을 사용하여 리소스에 대한 설정을 실행 취소할 수 있습니다. 256 페이지 "영역 구성을 되돌리는 방법"을 참조하십시오.

시스템에 여러 영역을 구성하는 스크립트는 248 페이지 "여러 영역 구성 스크립트"에 나와 있습니다.

비전역 영역의 구성을 표시하려면 253 페이지 “비전역 영역의 구성 표시 방법”을 참조하십시오.

▼ 영역 구성 방법

비전역 영역을 만드는 데 필요한 필수 요소는 zonename 등록 정보와 zonepath 등록 정보입니다. 기타 리소스 및 등록 정보는 선택 사항입니다. 또한 일부 선택적 리소스는 옵션을 선택해야 합니다. 예를 들어 dedicated-cpu 리소스를 사용할 것인지 또는 capped-cpu 리소스를 사용할 것인지를 결정해야 합니다. 사용 가능한 zonecfg 등록 정보 및 리소스에 대한 자세한 내용은 222 페이지 “영역 구성 데이터”를 참조하십시오.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 관리자로 전환합니다.

2 선택한 영역 이름을 사용하여 영역 구성을 설정합니다.

이 예제 절차에서는 my-zone이라는 이름을 사용합니다.

```
global# zonecfg -z my-zone
```

이 영역을 처음으로 구성한 경우 다음 시스템 메시지가 표시됩니다.

```
my-zone: No such zone configured
Use ’create’ to begin configuring a new zone.
```

3 새 영역 구성을 만듭니다.

이 절차에서는 기본 설정을 사용합니다.

```
zonecfg:my-zone> create
create: Using system default template ’SYSdefault’
```

4 이 절차에서는 영역 경로 /zones/my-zone을 설정합니다.

```
zonecfg:my-zone> set zonepath=/zones/my-zone
```

영역이 ZFS 데이터 집합에 있어야 합니다. ZFS 데이터 집합은 영역을 설치하거나 연결할 때 자동으로 생성됩니다. ZFS 데이터 집합을 만들 수 없을 경우 영역이 설치되거나 연결되지 않습니다. 영역 경로의 상위 디렉토리가 존재할 경우 해당 상위 디렉토리가 마운트된 데이터 집합의 마운트 지점이어야 합니다.

5 자동 부트 값을 설정합니다.

true로 설정하는 경우, 전역 영역이 부트될 때 영역이 자동으로 부트됩니다. 기본값은 false입니다. 자동 부트 영역의 경우 영역 서비스 svc:/system/zones:default도 사용으로 설정해야 합니다. 이 서비스는 기본적으로 사용으로 설정됩니다.

```
zonecfg:my-zone> set autoboot=true
```

6 **영역에 대한 지속 부트 인수를 설정합니다.**

```
zonecfg:my-zone> set bootargs="-m verbose"
```

7 **이 영역에 하나의 CPU를 전용으로 지정합니다.**

```
zonecfg:my-zone> add dedicated-cpu
```

a. **CPU 수를 설정합니다.**

```
zonecfg:my-zone:dedicated-cpu> set ncpus=1-2
```

b. **(옵션) 중요도를 설정합니다.**

```
zonecfg:my-zone:dedicated-cpu> set importance=10
```

기본값은 1입니다.

c. **지정을 종료합니다.**

```
zonecfg:my-zone:dedicated-cpu> end
```

8 **기본 권한 집합을 수정합니다.**

```
zonecfg:my-zone> set limitpriv="default,sys_time"
```

이 행은 기본 권한 집합에 시스템 시계를 설정할 수 있는 권한을 추가합니다.

9 **예약 클래스를 FSS로 설정합니다.**

```
zonecfg:my-zone> set scheduling-class=FSS
```

10 **메모리 상한값을 추가합니다.**

```
zonecfg:my-zone> add capped-memory
```

a. **메모리 상한값을 설정합니다.**

```
zonecfg:my-zone:capped-memory> set physical=1g
```

b. **스왑 메모리 상한값을 설정합니다.**

```
zonecfg:my-zone:capped-memory> set swap=2g
```

c. **고정 메모리 상한값을 설정합니다.**

```
zonecfg:my-zone:capped-memory> set locked=500m
```

d. **메모리 상한값 지정을 완료합니다.**

```
zonecfg:my-zone:capped-memory> end
```

주 – `capped-memory` 리소스를 사용하려면 전역 영역에 `resource-cap` 패키지가 설치되어 있어야 합니다.

---

11 파일 시스템을 추가합니다.

```
zonecfg:my-zone> add fs
```

a. 이 절차에서는 파일 시스템의 마운트 지점을 /usr/local로 설정합니다.

```
zonecfg:my-zone:fs> set dir=/usr/local
```

b. 전역 영역의 /opt/local이 구성 중인 영역의 /usr/local로 마운트되도록 지정합니다.

```
zonecfg:my-zone:fs> set special=/opt/local
```

비전역 영역에서 /usr/local 파일 시스템은 읽기 및 쓰기가 가능하게 설정됩니다.

c. 이 절차에서는 파일 시스템 유형을 lofs로 지정합니다.

```
zonecfg:my-zone:fs> set type=lofs
```

이 유형은 커널이 파일 시스템과 상호 작용하는 방식을 지정합니다.

d. 파일 시스템 지정을 완료합니다.

```
zonecfg:my-zone:fs> end
```

이 단계를 두 번 이상 수행하여 둘 이상의 파일 시스템을 추가할 수 있습니다.

12 필요할 경우 hostid를 설정합니다.

```
zonecfg:my-zone> set hostid=80f0c086
```

13 *sales*라는 ZFS 데이터 집합을 *tank* 저장소 풀에 추가합니다.

```
zonecfg:my-zone> add dataset
```

a. ZFS 데이터 집합 *sales*에 대한 경로를 지정합니다.

```
zonecfg:my-zone> set name=tank/sales
```

b. 데이터 집합 지정을 종료합니다.

```
zonecfg:my-zone> end
```

영역 관리자는 데이터 집합 내에서 파일 시스템을 만들고 삭제할 수 있으며 데이터 집합 등록 정보를 수정할 수 있습니다.

14 자동 VNIC를 사용하여 배타적 IP 영역을 만듭니다.

```
zonecfg:my-zone> set ip-type=exclusive

zonecfg:my-zone> add anet
```

a. 생성될 링크의 기본 링크로 auto를 지정합니다.

```
zonecfg:my-zone:anet> set lower-link=auto
```

zoneadmd 데몬에서 영역이 부트될 때마다 VNIC가 생성될 링크를 자동으로 선택합니다.

b. 지정을 종료합니다.

```
zonecfg:my-zone:anet> end
```

15 장치를 추가합니다.

```
zonecfg:my-zone> add device
```

a. 이 절차에서는 장치 일치를 /dev/sound/*로 설정합니다.

```
zonecfg:my-zone:device> set match=/dev/sound/*
```

b. 장치 지정을 완료합니다.

```
zonecfg:my-zone:device> end
```

둘 이상의 장치를 추가하기 위해 이 단계를 여러 번 수행할 수 있습니다.

16 format 명령을 사용한 디스크 레이블 지정을 허용하려면 전체 disk/LUN을 영역에 위임하고 allow-partition 등록 정보를 설정해야 합니다.

```
zonecfg:my-zone> add device
```

a. 이 절차에서는 장치 일치를 /dev/*dsk/c2t40d3*로 설정합니다.

```
zonecfg:my-zone:device> set match=/dev/*dsk/c2t40d3*
```

b. allow-partition을 true로 설정합니다.

```
zonecfg:my-zone:device> set allow-partition=true
```

c. 장치 지정을 완료합니다.

```
zonecfg:my-zone:device> end
```

둘 이상의 장치를 추가하기 위해 이 단계를 여러 번 수행할 수 있습니다.

17 디스크에서 uscsi 작업을 허용하려면 allow-raw-io 등록 정보를 설정해야 합니다.

```
zonecfg:my-zone> add device
```

a. 이 절차에서는 장치 일치를 /dev/*dsk/c2t40d3*로 설정합니다.

```
zonecfg:my-zone:device> set match=/dev/*dsk/c2t40d3*
```

b. allow-raw-io를 true로 설정합니다.

```
zonecfg:my-zone:device> set allow-raw-io=true
```

c. 장치 지정을 완료합니다.

```
zonecfg:my-zone:device> end
```

주의 – 디스크의 영역에서 `uscsi` 작업을 수행할 수 있도록 허용하면 영역에서 디스크와 동일한 버스에 연결된 다른 장치에도 액세스할 수 있게 됩니다. 따라서 이 기능을 사용으로 설정하면 보안 위험이 발생할 수 있고 전역 영역 및 동일한 버스의 리소스를 사용하는 다른 영역이 외부 공격에 노출될 수 있습니다. `uscsi(7I)`를 참조하십시오.

둘 이상의 장치를 추가하기 위해 이 단계를 여러 번 수행할 수 있습니다.

18 등록 정보 이름을 사용하여 영역 전체의 리소스 제어를 추가합니다.

```
zonecfg:my-zone> set max-sem-ids=10485200
```

둘 이상의 리소스 제어를 추가하기 위해 이 단계를 여러 번 수행할 수 있습니다.

19 `attr` 리소스 유형을 사용하여 설명을 추가합니다.

```
zonecfg:my-zone> add attr
```

a. 이름을 `comment`로 설정합니다.

```
zonecfg:my-zone:attr> set name=comment
```

b. 유형을 `string`으로 설정합니다.

```
zonecfg:my-zone:attr> set type=string
```

c. 값을 해당 영역을 나타내는 설명으로 설정합니다.

```
zonecfg:my-zone:attr> set value="This is my work zone."
```

d. `attr` 리소스 유형 지정을 완료합니다.

```
zonecfg:my-zone:attr> end
```

20 영역에 대한 영역 구성을 확인합니다.

```
zonecfg:my-zone> verify
```

21 영역에 대한 영역 구성을 커밋합니다.

```
zonecfg:my-zone> commit
```

22 `zonecfg` 명령을 종료합니다.

```
zonecfg:my-zone> exit
```

프롬프트에서 명시적으로 `commit`를 입력하지 않은 경우에도 `exit`를 입력하거나 EOF가 발생할 때 `commit`가 자동으로 시도됩니다.

자세한 정보 **명령줄에서 여러 하위 명령 사용**

참고 – zonecfg 명령은 여러 하위 명령을 지원합니다. 동일한 셸 호출에서 각 명령을 따옴표로 묶고 세미콜론으로 구분하여 입력하면 됩니다.

```
global# zonecfg -z my-zone "create ; set zonepath=/zones/my-zone"
```

공유 IP 영역의 경우 zonecfg net 리소스에 고정 주소만 지정할 수 있습니다. 명령줄에서 제공할 수 없습니다.

다음 단계

커밋한 영역 구성을 설치하려면 268 페이지 “영역 설치 및 부트”를 참조하십시오.

여러 영역 구성 스크립트

이 스크립트를 사용하여 시스템에서 여러 영역을 구성하고 부트할 수 있습니다. 생성된 영역은 anet 리소스를 포함하는 기본 배타적 IP 영역입니다.

스크립트를 실행하기 전에 SCI 도구를 실행하여 구성 프로파일을 만들어야 합니다.

```
global# sysconfig create-profile -o sc_config.xml
```

이 스크립트는 다음 매개변수를 사용합니다.

- 만들 영역의 수
- *zonename* 접두어
- 기본 디렉토리로 사용할 디렉토리
- 새로 만든 구성 프로파일의 전체 경로 이름

스크립트를 실행하려면 전역 영역에서 루트 권한을 가진 전역 관리자이거나, 적합한 권한 프로파일을 가진 사용자여야 합니다.

```
#!/bin/ksh
#
# Copyright 2006-2011 Oracle Corporation. All rights reserved.
# Use is subject to license terms.
#

#
# This script serves as an example of how to instantiate several zones
# with no administrative interaction.  Run the script with no arguments to
# get a usage message.  The general flow of the script is:
#
#   1) Parse and check command line arguments
#   2) Configure all zones that are not yet configured
```

```
#   3) Install the first zone, if needed
#   4) Create the remaining zones as clones of the first zone
#
# Upon successful completion, the requested number of zones will be
# been installed and booted.
#

export PATH=/usr/bin:/usr/sbin

me=$(basename $0)
function fail_usage {
    print -u2 "Usage:
    $me <#-of-zones> <zonename-prefix> <basedir> <sysconfig.xml>

Generate sysconfig.xml with:
    sysconfig create-profile -o sysconfig.xml

When running sysconfig, choose \"Automatically\" or \"None\" for network
configuration. The value entered for \"Computer Name\" will ignored:
each zone's nodename will be set to match the zone name."

    exit 2
}

function log {
    print "$(date +%T) $@"
}

function error {
    print -u2 "$me: ERROR: $@"
}

function get_zone_state {
    zoneadm -z "$1" list -p 2>/dev/null | cut -d: -f3
}

#
# Parse and check arguments
#
(( $# != 4 )) && fail_usage

# If $1 is not a number nzones will be set to 0.
integer nzones=$1
if (( nzones < 1 )); then
    error "Invalid number of zones \"$1\""
    fail_usage
fi
# Be sure that zonename prefix is an allowable zone name and not too long.
prefix=$2
if [[ $prefix != @([a-zA-Z0-9])*([-_.a-zA-Z0-9]) || ${#prefix} > 62 ]]; then
    error "Invalid zonename prefix"
    fail_usage
fi
# Be sure that basedir is an absolute path.  zoneadm will create the directory
# if needed.
dir=$3
if [[ $dir != /* ]]; then
    error "Invalid basedir"
    fail_usage
```

```
fi
# Be sure the sysconfig profile is readable and ends in .xml
sysconfig=$4
if [[ ! -f $sysconfig || ! -r $sysconfig || $sysconfig != *.xml ]]; then
    error "sysconfig profile missing, unreadable, or not *.xml"
    fail_usage
fi

#
# Create a temporary directory for all temp files
#
export TMPDIR=$(mktemp -d /tmp/$me.XXXXXX)
if [[ -z $TMPDIR ]]; then
    error "Could not create temporary directory"
    exit 1
fi
trap 'rm -rf $TMPDIR' EXIT

#
# Configure all of the zones
#
for (( i=1; i <= nzones; i++ )); do
    zone=$prefix$i
    state=$(get_zone_state $zone)
    if [[ -n $state ]]; then
        log "Skipping configuration of $zone: already $state"
        continue
    fi

    log "Configuring $zone"
    zonecfg -z "$zone" "create; set zonepath=$dir/$zone"
    if (( $? != 0 )); then
        error "Configuration of $zone failed"
        exit 1
    fi
done

#
# Install the first zone, then boot it for long enough for SMF to be
# initialized.  This will make it so that the first boot of all the clones
# goes much more quickly.
#
zone=${prefix}1
state=$(get_zone_state $zone)
if [[ $state == configured ]]; then
    log "Installing $zone"

    # Customize the nodename in the sysconfig profile
    z_sysconfig=$TMPDIR/$zone.xml
    search="<propval type=\"astring\" name=\"nodename\" value=\".*\"/>"
    replace="<propval type=\"astring\" name=\"nodename\" value=\"$zone\"/>"
    sed "s|$search|$replace|" $sysconfig > $z_sysconfig

    zoneadm -z $zone install -c $z_sysconfig
    if (( $? != 0 )); then
        error "Installation of $zone failed."
        rm -f $z_sysconfig
        exit 1
    fi
```

```
    rm -f $z_sysconfig
elif [[ $state != installed ]]; then
    error "Zone $zone is currently in the $state state."
    error "It must be in the installed state to be cloned."
    exit 1
fi
# Boot the zone no further than single-user.  All we really want is for
# svc:/system/manifest-import:default to complete.
log "Booting $zone for SMF manifest import"
zoneadm -z $zone boot -s
if (( $? != 0 )); then
    error "Failed to boot zone $zone"
    exit 1
fi
# This zlogin will return when manifest-import completes
log "Waiting for SMF manifest import in $zone to complete"
state=
while [[ $state != online ]]; do
    printf "."
    sleep 1
    state=$(zlogin $zone svcs -Ho state \
        svc:/system/manifest-import:default 2>/dev/null)
done
printf "\n"
log "Halting $zone"
zoneadm -z $zone halt
if (( $? != 0 )); then
    error "failed to halt $zone"
    exit 1
fi
firstzone=$zone

#
# Clone and boot the remaining zones
#
for (( i=2; i <= $nzones; i++ )); do
    zone=$prefix$i

    # Be sure that it needs to be installed
    state=$(get_zone_state $zone)
    if [[ $state != configured ]]; then
        log "Skipping installation of $zone: current state is $state."
        continue
    fi

    log "Cloning $zone from $firstzone"

    # Customize the nodename in the sysconfig profile
    z_sysconfig=$TMPDIR/$zone.xml
    search='<propval type="astring" name="nodename" value=".*"/>'
    replace='<propval type="astring" name="nodename" value="'$zone'"/>'
    sed "s|$search|$replace|" $sysconfig > $z_sysconfig

    # Clone the zone
    zoneadm -z $zone clone -c $z_sysconfig $firstzone
    if (( $? != 0 )); then
        error "Clone of $firstzone to $zone failed"
        rm -f $z_sysconfig
        exit 1
```

```
    fi
    rm -f $z_sysconfig

    # Boot the zone
    log "Booting $zone"
    zoneadm -z $zone boot
    if (( $? != 0 )); then
        error "Boot of $zone failed"
        exit 1
    fi
done

#
# Boot the first zone now that clones are done
#
log "Booting $firstzone"
zoneadm -z $firstzone boot
if (( $? != 0 )); then
    error "Boot of $firstzone failed"
    exit 1
fi

log "Completed in $SECONDS seconds"
exit 0
```

스크립트 출력:

```
$ ./buildzones
Usage:
        buildzones <#-of-zones> <zonename-prefix> <basedir> <sysconfig.xml>

Generate sysconfig.xml with:
        sysconfig create-profile -o sysconfig.xml

When running sysconfig, choose "Automatically" or "None" for network
configuration. The value entered for "Computer Name" will be ignored:
each zone's nodename will be set to match the zone name.

# ~user/scripts/buildzones 3 bz /tank/bz /var/tmp/sysconfig.xml
12:54:04 Configuring bz1
12:54:05 Configuring bz2
12:54:05 Configuring bz3
12:54:05 Installing bz1
A ZFS file system has been created for this zone.
Progress being logged to /var/log/zones/zoneadm.20110816T195407Z.bz1.install
       Image: Preparing at /tank/bz/bz1/root.

 Install Log: /system/volatile/install.24416/install_log
AI Manifest: /usr/share/auto_install/manifest/zone_default.xml
  SC Profile: /tmp/buildzones.F4ay4T/bz1.xml
    Zonename: bz1
Installation: Starting ....
```

▼ 비전역 영역의 구성 표시 방법

이 절차를 수행하려면 전역 영역의 전역 관리자이거나, 적합한 권한 프로파일을 가진 사용자여야 합니다.

1 관리자로 전환합니다.

2 영역 구성을 표시합니다.

```
global# zonecfg -z zonename info
```

zonecfg 명령을 사용하여 영역 구성 수정

zonecfg 명령을 사용하여 다음을 수행할 수도 있습니다.

- 영역 구성의 리소스 유형 수정
- 영역 구성의 등록 정보 값 지우기
- 영역에 전용 장치 추가

▼ 영역 구성의 리소스 유형 수정 방법

리소스 유형을 선택하고 해당 리소스의 사양을 수정할 수 있습니다.

이 절차를 수행하려면 전역 영역의 전역 관리자이거나, 적합한 권한 프로파일을 가진 사용자여야 합니다.

1 관리자로 전환합니다.

2 수정할 영역을 선택합니다. 이 절차에서는 my-zone입니다.

```
global# zonecfg -z my-zone
```

3 변경할 리소스 유형을 선택합니다. 예를 들어 리소스 제어를 선택합니다.

```
zonecfg:my-zone> select rctl name=zone.cpu-shares
```

4 현재 값을 제거합니다.

```
zonecfg:my-zone:rctl> remove value (priv=privileged,limit=20,action=none)
```

5 새 값을 추가합니다.

```
zonecfg:my-zone:rctl> add value (priv=privileged,limit=10,action=none)
```

6 수정된 rctl 지정을 완료합니다.

```
zonecfg:my-zone:rctl> end
```

7 **영역에 대한 영역 구성을 커밋합니다.**

```
zonecfg:my-zone> commit
```

8 **zonecfg 명령을 종료합니다.**

```
zonecfg:my-zone> exit
```

프롬프트에서 명시적으로 commit를 입력하지 않은 경우에도 exit를 입력하거나 EOF가 발생할 때 commit가 자동으로 시도됩니다.

zonecfg를 통해 커밋한 변경 사항은 다음 번에 영역을 부트할 때 적용됩니다.

▼ 영역 구성의 등록 정보를 지우는 방법

이 절차를 사용하여 독립적 등록 정보를 재설정합니다.

1 **관리자로 전환합니다.**

2 **수정할 영역을 선택합니다. 이 절차에서는 my-zone입니다.**

```
global# zonecfg -z my-zone
```

3 **변경할 등록 정보를 지웁니다. 이 절차에서는 기존 풀 연결입니다.**

```
zonecfg:my-zone> clear pool
```

4 **영역에 대한 영역 구성을 커밋합니다.**

```
zonecfg:my-zone> commit
```

5 **zonecfg 명령을 종료합니다.**

```
zonecfg:my-zone> exit
```

프롬프트에서 명시적으로 commit를 입력하지 않은 경우에도 exit를 입력하거나 EOF가 발생할 때 commit가 자동으로 시도됩니다.

zonecfg를 통해 커밋한 변경 사항은 다음 번에 영역을 부트할 때 적용됩니다.

▼ 영역의 이름을 바꾸는 방법

이 절차를 사용하여 구성된 상태 또는 설치된 상태의 영역의 이름을 바꿀 수 있습니다.

이 절차를 수행하려면 전역 영역의 전역 관리자이거나, 적합한 권한 프로파일을 가진 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **이름을 바꿀 영역을 선택합니다. 이 절차에서는 my-zone입니다.**

```
global# zonecfg -z my-zone
```

3 **영역의 이름을 변경합니다. 예를 들면 newzone으로 변경합니다.**

```
zonecfg:my-zone> set zonename=newzone
```

4 **변경 사항을 커밋합니다.**

```
zonecfg:newzone> commit
```

5 **zonecfg 명령을 종료합니다.**

```
zonecfg:newzone> exit
```

zonecfg를 통해 커밋한 변경 사항은 다음 번에 영역을 부트할 때 적용됩니다.

▼ 영역에 전용 장치 추가 방법

다음 지정은 비전역 영역 구성에 스캐닝 장치를 추가합니다.

이 절차를 수행하려면 전역 영역의 전역 관리자이거나, 적합한 인증을 받은 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **장치를 추가합니다.**

```
zonecfg:my-zone> add device
```

3 **장치 일치를 설정합니다. 이 예에서는 /dev/scsi/scanner/c3t4*입니다.**

```
zonecfg:my-zone:device> set match=/dev/scsi/scanner/c3t4*
```

4 **장치 지정을 완료합니다.**

```
zonecfg:my-zone:device> end
```

5 **zonecfg 명령을 종료합니다.**

```
zonecfg:my-zone> exit
```

▼ 전역 영역의 zone.cpu-shares를 설정하는 방법

이 절차는 전역 영역의 할당 분배를 영구적으로 설정하는 데 사용됩니다.

이 절차를 수행하려면 전역 영역의 전역 관리자이거나, 전역 영역에서 적합한 권한 프로파일을 가진 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **zonecfg 명령을 사용합니다.**

```
# zonecfg -z global
```

3 **전역 영역에 지정할 할당 수를 5로 설정합니다.**

```
zonecfg:global> set cpu-shares=5
```

4 **zonecfg를 종료합니다.**

```
zonecfg:global> exit
```

zonecfg 명령을 사용하여 영역 구성 되돌리기 또는 제거

영역의 구성을 되돌리거나 영역 구성을 삭제하려면 zonecfg(1M)에 설명된 zonecfg 명령을 사용합니다.

▼ 영역 구성을 되돌리는 방법

zonecfg 유틸리티를 사용하여 영역을 구성할 때 revert 하위 명령을 사용하여 영역 구성에 대한 리소스 설정을 실행 취소할 수 있습니다.

이 절차를 수행하려면 전역 영역의 전역 관리자이거나, 전역 영역에서 영역 보안 권한 프로파일을 가진 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **tmp-zone이라는 영역을 구성할 때 구성을 보려면 info라고 입력합니다.**

```
zonecfg:tmp-zone> info
```

구성의 net 리소스 세그먼트가 다음과 같이 표시됩니다.

```
.
.
.
fs:
        dir: /tmp
        special: swap
        type: tmpfs
net:
        address: 192.168.0.1
        physical: eri0
device
        match: /dev/pts/*
.
.
.
```

3 **넷 주소를 제거합니다.**

```
zonecfg:tmp-zone> remove net address=192.168.0.1
```

4 **net 항목이 제거되었는지 확인합니다.**

```
zonecfg:tmp-zone> info

.
.
.
fs:
        dir: /tmp
        special: swap
        type: tmpfs
device
        match: /dev/pts/*
.
.
.
```

5 **revert를 입력합니다.**

```
zonecfg:tmp-zone> revert
```

6 **다음 질문에 예로 답합니다.**

```
Are you sure you want to revert (y/[n])? y
```

7 **넷 주소가 다시 한 번 표시되는지 확인합니다.**

```
zonecfg:tmp-zone> info

.
.
.
fs:
        dir: /tmp
        special: swap
        type: tmpfs
net:
        address: 192.168.0.1
        physical: eri0
device
        match: /dev/pts/*
.
.
.
```

▼ 영역 구성 삭제 방법

zonecfg와 delete 하위 명령을 함께 사용하여 시스템에서 영역 구성을 삭제합니다.

이 절차를 수행하려면 전역 영역에서 전역 관리자이거나 보안 권한 프로파일을 가진 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **다음 두 가지 방법 중 하나를 사용하여 a-zone 영역의 영역 구성을 삭제합니다.**

- -F 옵션을 사용하여 작업을 강제 실행합니다.

```
global# zonecfg -z a-zone delete -F
```

- 표시되는 메시지에 예로 답하여 대화식으로 영역을 삭제합니다.

```
global# zonecfg -z a-zone delete
Are you sure you want to delete zone a-zone (y/[n])? y
```

◆ ◆ ◆ 18 장

18 비전역 영역, 설치, 종료, 정지 및 복제 정보(개요)

이 장에서는 Oracle Solaris 시스템에 영역을 설치하는 내용에 대해 설명합니다. 또한 가상 플랫폼과 응용 프로그램을 관리하는 두 가지 프로세스인 `zoneadmd`와 `zsched`에 대해 설명합니다 영역 정지, 재부트, 복제 및 제거에 대한 정보도 제공합니다

이 장에서는 다음과 같은 내용을 다룹니다.

비전역 영역을 복제하거나 설치 및 부트하거나 정지 또는 제거하려면 19 장, "비전역 영역 설치, 부트, 종료, 정지, 제거 및 복제(작업)"를 참조하십시오.

`solaris10` 브랜드 영역 설치에 대한 자세한 내용은 33 장, "`solaris10` 브랜드 영역 설치"를 참조하십시오.

영역 설치 및 관리 개념

`zoneadm`(1M) 매뉴얼 페이지에 설명된 `zoneadm` 명령은 비전역 영역을 설치하고 관리하는 데 사용하는 기본 도구입니다. `zoneadm` 명령을 사용하는 작업은 전역 영역에서 실행해야 합니다. RBAC가 사용 중일 경우 다른 영역을 복사하는 하위 명령을 실행하려면 `solaris.zone.clonefrom/`*source_zone* 인증이 필요합니다.

`zoneadm` 명령을 수행하여 다음 작업을 수행할 수 있습니다.

- 영역 확인
- 영역 설치
- 설치된 영역의 상태를 불완전으로 변경

- 일반 Oracle Solaris 시스템 부트와 비슷한 영역 부트
- 실행 중인 영역에 대한 정보 표시
- 영역 종료
- 영역 정지
- 영역 재부트
- 설치된 영역 제거
- 시스템의 한 지점에서 동일한 시스템의 다른 지점으로 영역 재배치
- 동일한 시스템에 있는 기존 영역의 구성에 기초하여 새 영역 제공
- `zonecfg` 명령을 사용하여 영역 마이그레이션

영역 설치 및 확인 절차는 19 장, “비전역 영역 설치, 부트, 종료, 정지, 제거 및 복제(작업)” 및 `zoneadm`(1M) 매뉴얼 페이지를 참조하십시오. 또한 `zoneadm list` 명령에 지원되는 옵션은 `zoneadm`(1M) 매뉴얼 페이지를 참조하십시오. 영역 구성 절차는 17 장, “비전역 영역 계획 및 구성(작업)” and the `zonecfg`(1M) 매뉴얼 페이지를 참조하십시오. 영역 상태에 대해서는 196 페이지 “비전역 영역 상태 모델”에서 설명합니다.

영역에 대한 Oracle Solaris Auditing 레코드 생성을 계획하려면 비전역 영역을 설치하기 전에 341 페이지 “영역에서 Oracle Solaris Auditing 사용”을 참조하십시오.

영역 구성

이 단원은 기존 영역 복제가 아니라 비전역 영역 구성을 처음 수행할 때 적용됩니다.

영역은 `zoneadm install -m` 명령에 전달된 매니페스트에서 지정한 패키지를 사용하여 설치됩니다. 매니페스트가 제공되지 않을 경우 기본 매니페스트는 `pkg:/group/system/solaris-small-server`를 사용합니다. 새 영역에는 기본 `solaris` 구성과 로그(SMF 저장소, `/etc`, `/var`)가 있으며 이 구성과 로그는 `zoneadm install -s`에 전달된 프로파일 및 `zonecfg add net` 항목에 지정된 네트워킹 정보를 통해서만 수정됩니다.

시스템 저장소, 영역에 구성된 게시자 및 전역 영역과 동기화를 유지하는 패키지는 24 장, “영역이 설치된 Oracle Solaris 11 시스템의 자동 설치 및 패키지 정보”에서 설명합니다.

영역의 루트 파일 시스템에 필요한 파일은 영역의 루트 경로에 설치됩니다.

영역이 성공적으로 설치되면 이제 부트하고 초기 로그인할 수 있습니다.

다음 출처의 데이터는 영역이 설치될 때 참조되지 않으며 복사되지 않습니다.

- 설치되지 않은 패키지
- CD 및 DVD의 데이터
- 네트워크 설치 이미지

또한 전역 영역에 존재할 수 있는 다음 유형의 정보는 설치하는 영역에 복사되지 않습니다.

- `/etc/passwd` 파일의 새 사용자 또는 변경된 사용자

- /etc/group 파일의 새 그룹 또는 변경된 그룹
- DHCP 주소 지정과 같은 네트워킹 서비스에 대한 구성
- sendmail과 같은 네트워크 서비스에 대한 사용자 정의
- 이름 지정 서비스와 같은 네트워크 서비스에 대한 구성
- 새로 추가되거나 변경된 crontab, 프린터 및 메일 파일
- 시스템 로그, 메시지 및 계정 파일

Oracle Solaris Auditing을 사용할 경우 파일을 수정해야 할 수 있습니다. 자세한 내용은 341 페이지 "영역에서 Oracle Solaris Auditing 사용"을 참조하십시오.

구성 파일에 지정된 리소스는 영역이 설치됨 상태에서 준비 상태로 전환할 때 추가됩니다. 고유의 영역 ID는 시스템에서 지정합니다. 파일 시스템이 마운트되면 네트워크 인터페이스가 설정되고 장치가 구성됩니다. 준비 상태로 전환되면 사용자 프로세스 실행을 시작하기 위해 가장 플랫폼이 준비됩니다. 준비 상태에서는 가상 플랫폼을 관리하기 위해 zsched 및 zoneadmd 프로세스가 시작됩니다.

- sched와 비슷한 시스템 일정 잡기 프로세스인 zsched는 영역과 연관된 커널 리소스를 추적하는 데 사용됩니다.
- zoneadmd는 영역 관리 데몬입니다.

준비 상태의 영역에는 실행 중인 사용자 프로세스가 없습니다. 준비 영역과 실행 중 영역의 기본적인 차이는 실행 중 영역에는 적어도 하나 이상의 프로세스가 실행되고 있다는 것입니다. 자세한 내용은 init(1M) 매뉴얼 페이지를 참조하십시오.

영역 설치 방법

solaris 브랜드 설치 프로그램은 다음 방법 중 하나를 사용하여 영역을 설치하도록 지원합니다.

- 기본 저장소 solaris publisher (http://pkg.oracle.com/solaris/release/).
- Oracle Solaris 릴리스를 실행 중인, 설치된 시스템의 이미지.

  시스템 이미지는 ZFS 전송 스트림이 될 수 있습니다. 다른 이미지에는 cpio(1) 아카이브 또는 pax(1) xustar 아카이브가 포함되어 있습니다. cpio 아카이브는 gzip 또는 bzip2 유틸리티를 사용하여 압축할 수 있습니다. 이미지는 시스템 루트 트리의 최상위 레벨에 대한 경로 또는 기존 영역 경로일 수도 있습니다.

  시스템 이미지에서 영역을 설치하려면 -a or - d 옵션이 필요합니다. -a 또는 -d 옵션을 사용하지 않을 경우 소프트웨어 저장소에서 영역이 설치됩니다.

설치 프로그램 옵션은 다음 표에 나와 있습니다. 명령줄의 예는 269 페이지 "구성된 영역 설치 방법"을 참조하십시오.

옵션	설명
-m *manifest*	AI 매니페스트는 영역 설치 방법을 정의하는 XML 파일입니다. 파일 인수는 절대 경로로 지정해야 합니다.
-c *profile*| *dir*	구성 중 적용할 프로파일 또는 프로파일 디렉토리를 제공합니다. 파일 인수는 절대 경로로 지정해야 합니다. 프로파일이 적용될 경우 구성 절차가 비대화식으로 수행됩니다. 프로파일이 제공되지 않을 경우 시스템 구성에 대화식 시스템 구성 도구가 사용됩니다. 모든 프로파일 파일의 확장자는 .xml이어야 합니다. -c에 디렉토리 옵션을 제공할 경우 해당 디렉토리의 모든 프로파일이 유효하고 올바른 형식의 구성 파일이어야 합니다.
-a *archive*	아카이브에 대한 경로입니다. 아카이브는 gzip 또는 bzip을 사용하여 압축할 수 있습니다. -d 옵션과 -a 옵션은 함께 사용할 수 없습니다.
-d *path*	설치된 시스템의 루트 디렉토리에 대한 경로입니다. *path*가 하이픈(-)일 경우 zonepath가 시스템 이미지로 이미 채워진 것으로 간주됩니다. -d 옵션과 -a 옵션은 함께 사용할 수 없습니다.
-p	영역 설치 후 시스템 ID를 유지합니다. -p 옵션과 -u 옵션은 함께 사용할 수 없습니다.
-s	확인 없이 설치합니다. -s 옵션과 -v 옵션은 함께 사용할 수 없습니다.
-u	설치 후 영역의 구성을 해제하고, 영역 부트 시 새로 구성할지 묻습니다. -p 옵션과 -u 옵션은 함께 사용할 수 없습니다.
-v	설치 프로세스의 출력을 상세 정보로 표시합니다. -s 옵션과 -v 옵션은 함께 사용할 수 없습니다.

zoneadmd 데몬

영역 관리 데몬인 zoneadmd는 영역의 가상 플랫폼을 관리하기 위한 기본 프로세스입니다. 이 데몬은 또한 영역 부트 및 종료도 담당합니다. 시스템의 활성(준비, 실행 중 또는 종료) 영역 각각에는 하나의 zoneadmd 프로세스가 실행되고 있습니다.

zoneadmd 데몬은 영역 구성에 지정된 대로 영역을 설정합니다. 이 프로세스에는 다음 작업이 포함됩니다.

- 영역 ID 할당 및 zsched 시스템 프로세스 시작
- 영역 전체 리소스 제어 설정

- 영역 구성에 지정된 대로 영역의 장치 준비
- 네트워크 인터페이스 설정
- 루프백 및 일반 파일 시스템 마운트
- 영역 콘솔 장치 인스턴스와 및 초기화

zoneadmd 데몬이 이미 실행 중이 아닐 경우 zoneadm을 통해 자동으로 시작됩니다. 따라서 어떤 이유로든 데몬이 실행되고 있지 않을 경우 영역을 관리하기 위해 zoneadm의 호출이 zoneadmd를 다시 시작합니다.

zoneadmd 데몬의 매뉴얼 페이지는 zoneadmd(1M)입니다.

zsched 영역 스케줄러

활성 영역이란 준비 상태, 실행 중 상태 또는 종료 상태의 영역을 말합니다. 모든 활성 영역에는 해당 커널 프로세스 zsched가 있습니다. 영역을 대신하여 작업을 수행하는 커널 스레드는 zsched에서 소유합니다. zsched 프로세스는 영역 부속 시스템에서 영역당 커널 스레드를 추적할 수 있도록 합니다.

영역 응용 프로그램 환경

zoneadm 명령은 영역 응용 프로그램 환경을 만드는 데 사용됩니다.

영역의 내부 구성은 sysconfig 인터페이스를 사용하여 지정됩니다. 내부 구성은 사용할 이름 지정 서비스, 기본 로케일 및 표준 시간대, 영역의 루트 암호 및 기타 응용 프로그램 환경의 특성을 지정합니다. sysconfig 인터페이스에 대한 설명은 **Oracle Solaris 11 시스템**의 6 장, “Oracle Solaris 인스턴스 구성 해제 또는 재구성” 및 sysconfig(1M) 매뉴얼 페이지에 나와 있습니다. 영역의 기본 로케일과 표준 시간대는 전역 설정과 독립적으로 구성할 수 있습니다.

영역 종료, 정지, 재부트 및 제거 정보

이 단원에서는 영역 정지, 재부트, 제거 및 복제 절차에 대한 개요를 제공합니다.

영역 종료

zoneadm shutdown c 명령은 영역을 완전히 종료하는 데 사용됩니다. 이 작업은 영역에서 /usr/sbin/init 0을 실행한 것과 동일합니다. -r 옵션도 지정할 경우 영역이 재부트됩니다. 지원되는 부트 옵션은 264 페이지 “영역 부트 인수”를 참조하십시오.

svc:/system/zones 서비스는 zoneadm shutdown을 사용하여 전역 영역이 종료될 때 영역을 완전히 종료합니다.

shutdown 하위 명령은 영역이 성공적으로 종료될 때까지 기다립니다. 적절한 시간 내에 작업이 완료되지 않을 경우 zoneadm halt를 사용하여 영역을 강제로 정지할 수 있습니다. 275 페이지 "영역 정지 방법"을 참조하십시오.

영역 정지

zoneadm halt 명령은 영역에서 실행 중인 모든 프로세스를 종료하고 가상 플랫폼을 제거하는 데 사용됩니다. 그러면 영역은 다시 설치됨 상태로 됩니다. 모든 프로세스가 강제 종료되고, 장치가 구성 해제되고, 네트워크 인터페이스가 삭제되고, 파일 시스템이 마운트 해제되며, 커널 데이터가 삭제됩니다.

halt 명령은 영역 내의 모든 종료 스크립트를 실행하지 **않습니다**. 영역을 종료하려면 263 페이지 "영역 종료"를 참조하십시오. 또는 영역에 로그인하여 종료를 실행할 수도 있습니다. 297 페이지 "zlogin을 사용하여 영역을 종료하는 방법"을 참조하십시오.

정지 작업이 실패할 경우 377 페이지 "영역이 정지되지 않음"을 참조하십시오.

영역 재부트

zoneadm reboot 명령은 영역을 재부트하는 데 사용됩니다. 영역은 정지된 다음 다시 부트됩니다. 영역을 재부트하면 영역 ID가 변경됩니다.

영역 부트 인수

영역은 zoneadm boot 및 reboot 명령과 함께 사용되는 다음 부트 인수를 지원합니다.

- -i *altinit*
- -m *smf_options*
- -s

다음 정의가 적용됩니다.

-i *altinit*
: 첫번째 프로세스가 될 대체 실행 파일을 선택합니다. *altinit*는 실행 파일에 대한 유효한 경로여야 합니다. 기본적으로 첫번째로 실행되는 프로세스는 init(1M)에서 설명합니다.

-m *smf_options*
: SMF의 부트 동작을 제어합니다. 복구 옵션과 메시지 옵션의 두 가지 옵션 범주가 있습니다. 메시지 옵션은 부트 중 표시되는 메시지의 수와 유형을 결정합니다. 서비스 옵션은 시스템을 부트하는 데 사용되는 서비스를 결정합니다.

  복구 옵션은 다음과 같습니다.

  debug
  : 서비스에 대한 표준 출력과 모든 svc.startd 메시지를 로그에 인쇄합니다.

`milestone=`*milestone* 해당 마일스톤에서 정의한 섹션으로 부트합니다. 사용 가능한 마일스톤에는 `none`, `single-user`, `multi-user`, `multi-user-server` 및 `all`이 있습니다.

메시지 옵션은 다음과 같습니다.

`quiet` 서비스에 대한 표준 출력과 관리자 개입이 필요한 오류 메시지를 인쇄합니다.

`verbose` 서비스에 대한 표준 출력과 자세한 정보를 제공하는 메시지를 인쇄합니다.

`-s` `svc:/milestone/single-user:default` 마일스톤으로만 부트합니다. 이 마일스톤은 `init` 레벨 `s`와 동일합니다.

용례를 보려면 272 페이지 "영역 부트 방법" 및 273 페이지 "단일 사용자 모드로 영역을 부트하는 방법"을 참조하십시오.

Oracle Solaris SMF(서비스 관리 기능) 및 `init`에 대한 자세한 내용은 **Oracle Solaris 관리: 일반 작업**의 6 장, "서비스 관리(개요)", `svc.startd`(1M) 및 `init`(1M)을 참조하십시오.

영역 autoboot 설정

전역 영역이 부트될 때 영역을 자동으로 부트하려면 영역의 구성에서 `autoboot` 리소스 등록 정보를 `true`로 설정합니다. 기본 설정은 `false`입니다.

자동 부트 영역의 경우 영역 서비스 `svc:/system/zones:default`도 사용으로 설정해야 합니다. 이 서비스는 기본적으로 사용으로 설정됩니다.

`pkg update` 중 `autoboot` 설정에 대한 자세한 내용은 313 페이지 "영역 패키징 개요"를 참조하십시오.

설치된 영역 제거

`zoneadm uninstall` 명령은 영역의 루트 파일 시스템에 설치된 모든 파일을 제거하는 데 사용되는 명령입니다. `-F`(강제 실행) 옵션을 사용하지 않는 한 계속하기 전에 작업 수행을 확인하는 메시지가 표시됩니다. `uninstall` 명령은 작업을 되돌릴 수 없기 때문에 신중하게 사용해야 합니다.

비전역 영역 복제

복제를 사용하면 시스템에 구성되어 설치된 기존 영역을 복사하여 동일한 시스템에 새로운 영역을 빨리 만들 수 있습니다. 영역 간에 동일할 수 없는 구성 요소에 대해 등록 정보 및 리소스를 재설정하는 작업은 해야 합니다. 따라서 zonepath는 항상 변경해야 합니다. 또한 공유 IP 영역의 경우 넷 리소스의 IP 주소가 서로 달라야 합니다. 배타적 IP 영역의 경우 넷 리소스의 물리적 등록 정보가 서로 달라야 합니다.

- 영역 복제는 영역을 설치할 수 있는 가장 빠른 방법입니다.
- 새 영역에는 추가된 패키지나 파일 수정 등과 같이 원본 영역을 사용자 정의하기 위해 수행된 모든 변경 사항이 포함됩니다.

원본 zonepath와 대상 zonepath가 모두 ZFS에 있고 동일한 풀에 있을 경우 zoneadm clone 명령은 자동으로 ZFS를 사용하여 영역을 복제합니다. ZFS 복제를 사용할 경우 데이터가 수정되기 전에는 실제로 데이터가 복사되지 않습니다. 따라서 첫번째 복제 작업은 시간이 거의 걸리지 않습니다. zoneadm 명령은 소스 zonepath의 ZFS 스냅샷을 만들고 대상 zonepath를 설정합니다. 대상 영역의 zonepath는 ZFS 복제본의 이름을 지정하는 데 사용됩니다.

주 – ZFS가 복제되는 대신 복사되도록 zonepath를 지정할 수 있습니다. 이 경우 원본이 복제되지 않습니다.

자세한 내용은 278 페이지 “동일한 시스템에서 비전역 영역 복제”를 참조하십시오.

◆ ◆ ◆ 19장

19 비전역 영역 설치, 부트, 종료, 정지, 제거 및 복제(작업)

이 장에서는 비전역 영역을 설치하고 부트하는 방법을 설명합니다. 또한 복제를 사용하여 동일한 시스템에 영역을 설치하는 방법에 대해 설명합니다. 이외에도 영역 정지, 재부트 및 제거 등과 같은 설치와 관련된 작업에 대해서도 다룹니다. 기존 비전역 영역을 동일한 시스템의 새 위치로 이동합니다. 기존 비전역 영역을 동일한 시스템의 새 위치로 이동하고 시스템에서 영역을 완전히 삭제하는 절차도 포함됩니다.

영역 설치 및 관련 작업에 대한 일반적인 정보는 18 장, “비전역 영역, 설치, 종료, 정지 및 복제 정보(개요)”를 참조하십시오.

`solaris10` 브랜드 영역 설치 및 복제에 대한 자세한 내용은 33 장, “`solaris10` 브랜드 영역 설치”를 참조하십시오.

영역 설치(작업 맵)

작업	설명	수행 방법
(옵션) 영역을 설치하기 전에 구성된 영역을 확인합니다.	영역이 설치 요구 사항을 만족하는지 확인합니다. 이 절차를 생략할 경우 영역을 설치할 때 확인이 자동으로 수행됩니다.	268 페이지 “(옵션) 구성된 영역을 설치하기 전에 확인하는 방법”
구성된 영역을 설치합니다.	구성됨 상태의 영역을 설치합니다.	269 페이지 “구성된 영역 설치 방법”
영역의 UUID(Universally Unique Identifier)를 확인합니다.	이 구분 식별자는 영역을 설치할 때 지정되며 영역을 식별하기 위한 대체 방법입니다.	270 페이지 “설치된 비전역 영역의 UUID 확인 방법”

작업	설명	수행 방법
(옵션) 설치된 영역에서 준비 영역으로 전환합니다.	영역을 부트하고 즉시 사용하려는 경우 이 절차를 생략해도 됩니다.	272 페이지 "(옵션) 설치된 영역을 준비 상태로 전환하는 방법"
영역을 부트합니다.	영역을 부트하면 영역이 실행 중 상태로 됩니다. 영역은 준비 상태 또는 설치된 상태에서 부트할 수 있습니다.	272 페이지 "영역 부트 방법"
단일 사용자 모드로 영역을 부트합니다.	`svc:/milestone/single-user:default` 마일스톤으로만 부트합니다. 이 마일스톤은 `init` 레벨 `s`와 동일합니다. `init`(1M) 및 `svc.startd`(1M) 매뉴얼 페이지를 참조하십시오.	273 페이지 "단일 사용자 모드로 영역을 부트하는 방법"

영역 설치 및 부트

`zoneadm`(1M) 매뉴얼 페이지에 설명되어 있는 `zoneadm` 명령을 사용하여 비전역 영역에 대한 설치 작업을 수행합니다. 영역 설치를 수행하려면 전역 관리자이거나 적합한 인증을 받은 사용자여야 합니다. 이 장의 예에서는 242 페이지 "영역 구성, 확인 및 커밋"에서 설정한 영역 이름과 영역 경로를 사용합니다.

▼ (옵션) 구성된 영역을 설치하기 전에 확인하는 방법

영역을 설치하기 전에 확인할 수 있습니다. 확인 작업 중 하나는 디스크 공간이 충분한지 확인하는 것입니다. 이 절차를 생략할 경우 영역을 설치할 때 확인이 자동으로 수행됩니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 관리자로 전환합니다.

2 `verify` 하위 명령에 `-z` 옵션과 영역 이름을 사용하여 `my-zone`이라는 이름의 구성된 영역을 확인합니다.

```
global# zoneadm -z my-zone verify
```

영역 경로 확인에 대한 다음 메시지가 표시됩니다.

```
WARNING: /zones/my-zone does not exist, so it could not be verified.
    When 'zoneadm install' is run, 'install' will try to create
    /zones/my-zone, and 'verify' will be tried again,
    but the 'verify' may fail if:
    the parent directory of /zones/my-zone is group- or other-writable
    or
```

```
/zones/my-zone overlaps with any other installed zones
or
/zones/my-zone is not a mountpoint for a zfs file system.
```

오류 메시지가 표시되고 영역을 확인하는 데 실패할 경우 메시지에 명시된 사항을 수정하고 명령을 다시 실행해 봅니다.

오류 메시지가 표시되지 않을 경우 영역을 설치할 수 있습니다.

▼ 구성된 영역 설치 방법

이 절차는 구성된 비전역 영역을 설치하는 데 사용됩니다. 설치 옵션에 대한 자세한 내용은 261 페이지 "영역 설치 방법"을 참조하십시오.

영역이 해당 ZFS 데이터 집합에 있어야 합니다. ZFS만 지원됩니다. `zoneadm install` 명령은 영역이 설치될 때 `zonepath`에 대해 자동으로 ZFS 파일 시스템(데이터 집합)을 만듭니다. ZFS 데이터 집합이 생성되지 않을 경우 영역이 설치되지 않습니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 관리자로 전환합니다.

2 `zoneadm` 명령과 `install` 하위 명령을 함께 사용하여 구성된 영역 `my-zone`을 설치합니다. 그러면 `zonepath` ZFS에 대해 ZFS 데이터 집합이 자동으로 생성됩니다. 영역 경로의 상위 디렉토리도 데이터 집합이어야 하며 그렇지 않으면 파일 시스템이 생성되지 않습니다.

- **영역 설치:**

  ```
  global# zoneadm -z my-zone install
  ```

- **저장소에서 영역 설치:**

  ```
  global# zoneadm -z my-zone install -m manifest -c [ profile | dir ]
  ```

- **이미지에서 영역 설치:**

  ```
  global# zoneadm -z my-zone install -a archive -s -u
  ```

- **디렉토리에서 영역 설치:**

  ```
  global# zoneadm -z my-zone install -d path -p -v
  ```

이 영역에 대해 ZFS 파일 시스템이 생성되었다고 표시됩니다.

영역의 루트 파일 시스템에 필요한 파일과 디렉토리가 영역의 루트 경로 아래에 설치될 때 다양한 메시지가 표시됩니다.

3 **(옵션) 오류 메시지가 표시되고 영역이 설치되지 않을 경우 다음을 입력하여 영역 상태를 확인합니다.**

```
global# zoneadm -z my-zone list -cdv
# zoneadm list -cvd
  ID NAME             STATUS      PATH                       BRAND    IP
   0 global           running     /                          solaris  shared
   - test             configured  /zones/test                solaris  excl
```

- 상태가 구성된 것으로 나열되어 있으면 메시지에 지정된 대로 수정하고 `zoneadm install` 명령을 다시 시도합니다.
- 상태가 완료되지 않은 것으로 나열되어 있으며 먼저 다음 명령을 실행합니다.

  ```
  global# zoneadm -z my-zone uninstall
  ```

  그런 다음 메시지에 명시된 사항을 수정하고 `zoneadm install` 명령을 다시 실행해 봅니다.

4 **설치가 완료되면 `list` 하위 명령을 `-i` 및 `-v` 옵션과 함께 사용하여 설치된 영역을 나열하고 상태를 확인합니다.**

```
global# zoneadm list -iv
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
```

일반 오류 영역 설치가 인터럽트되거나 실패하는 경우, 영역이 완료되지 않은 상태로 남게 됩니다. `uninstall -F`를 사용하여 영역을 구성됨 상태로 재설정합니다.

다음 순서 이 영역은 기본적으로 **Oracle Solaris 관리: 일반 작업**의 7 장, "서비스 관리(작업)"에서 설명한 최소 네트워크 구성으로 설치되었습니다. 개방형 네트워크 구성으로 전환하거나 영역에 로그인할 때 개별 서비스를 사용 또는 사용 안함으로 설정할 수 있습니다. 자세한 내용은 298 페이지 "서비스를 사용으로 설정"을 참조하십시오.

▼ 설치된 비전역 영역의 UUID 확인 방법

UUID(Universally Unique Identifier)는 영역을 설치할 때 지정됩니다. UUID는 `zoneadm` 명령에 `list` 하위 명령과 `-c -p` 옵션을 사용하여 확인할 수 있습니다. UUID는 다섯번째 필드에 표시됩니다.

- **설치된 영역의 UUID를 확인합니다.**

```
global# zoneadm list -cp
```

다음과 유사하게 표시됩니다.

```
0:global:running:/::solaris:shared:-:none

6:my-zone:running:/zones/my-zone:61901255-35cf-40d6-d501-f37dc84eb504:solaris:excl:-:
```

예 19–1 **명령에 영역 UUID를 사용하는 방법**

```
global# zoneadm -z my-zone -u 61901255-35cf-40d6-d501-f37dc84eb504:solaris:excl list -v
```

-u *uuid-match*와 - z *zonename*이 모두 있을 경우 먼저 UUID를 기반으로 검색됩니다. 지정한 UUID를 가진 영역을 찾을 경우 해당 영역이 사용되고 -z 매개변수가 무시됩니다. 지정한 UUID를 가진 영역이 없을 경우 영역 이름으로 검색이 수행됩니다.

자세한 정보 **UUID 정보**

설치된 영역을 제거하고 다른 내용에 동일한 이름으로 다시 설치할 수 있습니다. 또한 내용을 변경하지 않고 이름을 바꿀 수도 있습니다. 따라서 영역 이름보다 UUID가 더 신뢰할 수 있는 방법입니다.

참조 자세한 내용은 zoneadm(1M) 및 libuuid(3LIB)를 참조하십시오.

▼ 설치된 비전역 영역을 불완전으로 표시하는 방법

시스템에 대한 관리상의 변경으로 인해 영역을 사용할 수 없거나 불일치가 발생할 경우 설치된 영역의 상태를 불완전으로 변경할 수 있습니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **testzone 영역을 불완전으로 표시합니다.**

```
global# zoneadm -z testzone mark incomplete
```

3 **list 하위 명령에 -i 옵션 및 -v 옵션을 사용하여 상태를 확인합니다.**

```
global# zoneadm list -iv
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
 -  testzone incomplete   /zones/testzone                solaris    excl
```

자세한 정보

영역을 불완전으로 표시

zoneadm의 mark 및 list 하위 명령에 -R *root* 옵션을 사용하여 대체 부트 환경을 지정합니다. 자세한 내용은 zoneadm(1M)을 참조하십시오.

주 – 영역을 불완전으로 표시하는 작업은 되돌릴 수 없습니다. 불완전으로 표시된 영역에서 수행할 수 있는 유일한 작업은 설치된 영역을 제거하고 구성됨 상태로 되돌리는 것입니다. 277 페이지 "설치된 영역 제거 방법"을 참조하십시오.

▼ (옵션) 설치된 영역을 준비 상태로 전환하는 방법

준비 상태로 전환하면 사용자 프로세스 실행을 시작하기 위한 가상 플랫폼이 준비됩니다. 준비 상태의 영역에는 실행 중인 사용자 프로세스가 없습니다.

영역을 부트하고 즉시 사용하려는 경우 이 절차를 생략해도 됩니다. 준비 상태로의 전환은 영역을 부트하면 자동으로 수행됩니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 관리자로 전환합니다.

2 zoneadm 명령에 -z 옵션, 영역의 이름(my-zone) 및 ready 하위 명령을 함께 사용하여 영역을 준비 상태로 전환합니다.

```
global# zoneadm -z my-zone ready
```

3 프롬프트에서 zoneadm list 명령에 -v 옵션을 사용하여 상태를 확인합니다.

```
global# zoneadm list -v
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  ready        /zones/my-zone                 solaris    excl
```

고유의 영역 ID 1이 지정되었습니다.

▼ 영역 부트 방법

영역을 부트하면 영역이 실행 중 상태로 됩니다. 영역은 준비 상태 또는 설치된 상태에서 부트할 수 있습니다. 부트되는 설치된 상태의 영역은 준비 상태에서 실행 중 상태로 투명하게 전환됩니다. 실행 중 상태에 있는 영역에 대해 영역 로그인이 허용됩니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1. **관리자로 전환합니다.**

2. **zoneadm 명령에 -z 옵션, 영역의 이름(my-zone) 및 boot 하위 명령을 함께 사용하여 영역을 부트합니다.**

```
global# zoneadm -z my-zone boot
```

3. **부트가 완료되면 list 하위 명령에 -v 옵션을 사용하여 상태를 확인합니다.**

```
global# zoneadm list -v
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  running      /zones/my-zone                 solaris    excl
```

예 19–2 **영역에 대한 부트 인수 지정**

-m verbose 옵션을 사용하여 영역을 부트합니다.

```
global# zoneadm -z my-zone boot -- -m verbose
```

-m verbose 부트 옵션을 사용하여 영역을 재부트합니다.

```
global# zoneadm -z my-zone reboot -- -m verbose
```

영역 관리자가 -m verbose 옵션을 사용하여 *my-zone* 영역을 재부트합니다.

```
my-zone# reboot -- -m verbose
```

일반 오류 영역의 구성에 지정된 IP 주소에 사용될 넷마스크를 찾을 수 없다는 메시지가 표시될 경우 376 페이지 “영역을 부트할 때 표시되는 netmasks 경고”를 참조하십시오. 이 메시지는 경고일 뿐이며 명령은 성공적으로 실행된 것입니다.

▼ 단일 사용자 모드로 영역을 부트하는 방법

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1. **관리자로 전환합니다.**

2. **단일 사용자 모드로 영역을 부트합니다.**

```
global# zoneadm -z my-zone boot -- -s
```

다음 단계

영역에 로그인하여 초기 내부 구성을 수행하려면 20 장, "비전역 영역 로그인(개요)" 및 21 장, "비전역 영역에 로그인(작업)"을 참조하십시오.

비전역 영역 종료, 정지, 재부트, 제거, 복제 및 삭제(작업 맵)

작업	설명	수행 방법
영역을 종료합니다.	`shutdown` 절차는 종료 스크립트를 실행하여 영역을 완전히 종료하기 위해 사용됩니다. `zlogin` 방법도 지원됩니다. 자세한 내용은 297 페이지 "`zlogin`을 사용하여 영역을 종료하는 방법"을 참조하십시오.	275 페이지 "영역 정지 방법"
영역을 정지합니다.	정지 절차는 영역의 응용 프로그램 환경과 가상 플랫폼을 모두 제거하기 위해 사용됩니다. 이 절차는 영역의 상태를 준비 상태에서 설치됨 상태로 되돌립니다. 영역을 완전히 종료하려면 297 페이지 "`zlogin`을 사용하여 영역을 종료하는 방법"을 참조하십시오.	275 페이지 "영역 정지 방법"
영역을 재부트합니다.	재부트 절차는 영역을 정지한 다음 다시 부트합니다.	276 페이지 "영역 재부트 방법"
설치된 영역을 제거합니다.	이 절차는 영역의 루트 파일 시스템에 있는 모든 파일을 제거합니다. **이 절차는 주의해서 사용해야 합니다**. 이 작업은 되돌릴 수 없습니다.	277 페이지 "설치된 영역 제거 방법"
동일한 시스템에 있는 기존 영역의 구성을 바탕으로 새로운 비전역 영역을 제공합니다.	영역 복제는 영역을 빠르게 설치할 수 있는 대체 방법입니다. 새 영역을 설치하려면 먼저 해당 영역을 구성해야 합니다.	278 페이지 "동일한 시스템에서 비전역 영역 복제"
시스템에서 비전역 영역을 삭제합니다.	이 절차는 시스템에서 영역을 완전히 제거합니다.	280 페이지 "시스템에서 비전역 영역 삭제"

영역 종료, 정지, 재부트 및 제거

▼ 영역 종료 방법

이 종료 절차는 영역을 완전히 종료합니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 관리자로 전환합니다.

2 시스템에서 실행 중인 영역 목록을 표시합니다.

```
global# zoneadm list -v
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                          BRAND      IP
 0  global   running      /                             solaris    shared
 1  my-zone  running      /zones/my-zone                solaris    excl
```

3 `zoneadm` 명령에 `-z` 옵션, 영역 이름(예: `my-zone`) 및 `shutdown` 하위 명령을 함께 사용하여 해당 영역을 종료합니다.

```
global# zoneadm -z my-zone shutdown
```

4 또한 `-r` 옵션을 지정하여 영역을 재부트합니다.

```
global# zoneadm -z my-zone shutdown -r boot_options
```

예 19–2를 참조하십시오.

5 시스템에서 실행 중인 영역 목록을 표시하여 영역이 종료되었는지 확인합니다.

```
global# zoneadm list -v
```

▼ 영역 정지 방법

정지 절차는 영역의 응용 프로그램 환경과 가상 플랫폼을 모두 제거하기 위해 사용됩니다. 영역을 완전히 종료하려면 297 페이지 “zlogin을 사용하여 영역을 종료하는 방법”을 참조하십시오.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 관리자로 전환합니다.

2 시스템에서 실행 중인 영역 목록을 표시합니다.

```
global# zoneadm list -v
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  running      /zones/my-zone                 solaris    excl
```

3 zoneadm 명령에 -z 옵션, 영역의 이름(예: my-zone) 및 halt 하위 명령을 함께 사용하여 해당 영역을 정지합니다.

```
global# zoneadm -z my-zone halt
```

4 시스템의 영역 목록을 다시 표시하여 my-zone이 정지되었는지 확인합니다.

```
global# zoneadm list -iv
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
```

5 영역을 다시 시작하려면 영역을 부트합니다.

```
global# zoneadm -z my-zone boot
```

일반 오류 영역이 제대로 정지되지 않을 경우 377 페이지 "영역이 정지되지 않음"에서 문제 해결 팁을 참조하십시오.

▼ 영역 재부트 방법

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다. 275 페이지 "영역 종료 방법"을 참조하십시오.

1 관리자로 전환합니다.

2 시스템에서 실행 중인 영역 목록을 표시합니다.

```
global# zoneadm list -v
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 1  my-zone  running      /zones/my-zone                 solaris    excl
```

3 zoneadm 명령에 -z reboot 옵션을 사용하여 my-zone 영역을 재부트합니다.

```
global# zoneadm -z my-zone reboot
```

4 시스템의 영역 목록을 다시 표시하여 my-zone이 재부트되었는지 확인합니다.

```
global# zoneadm list -v
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 2  my-zone  running      /zones/my-zone                 solaris    excl
```

참고 – my-zone의 영역 ID가 변경되었는지 확인합니다. 영역 ID는 일반적으로 재부트 후에 변경됩니다.

▼ 설치된 영역 제거 방법

주의 – 이 절차는 주의하여 사용하십시오. 영역의 루트 파일 시스템에 있는 모든 파일을 제거하는 작업은 되돌릴 수 없습니다.

영역이 실행 중 상태가 아니어야 합니다. 실행 중인 영역에는 uninstall 작업을 수행할 수 없습니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 관리자로 전환합니다.

2 시스템의 영역 목록을 표시합니다.

```
global# zoneadm list -v
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
```

3 zoneadm 명령에 -z uninstall 옵션을 사용하여 my-zone 영역을 제거합니다.

-F 옵션을 사용하여 작업을 강제 수행할 수도 있습니다. 이 옵션을 지정하지 않을 경우 확인 메시지가 표시됩니다.

```
global# zoneadm -z my-zone uninstall -F
```

zonepath에 대한 자체 ZFS 파일 시스템이 있는 영역을 제거하면 해당 ZFS 파일 시스템이 삭제됩니다.

4 시스템의 영역 목록을 다시 표시하여 my-zone이 더 이상 나열되지 않는지 확인합니다.

```
global# zoneadm list -iv
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
```

일반 오류 설치된 영역을 제거하는 작업이 인터럽트될 경우 영역은 불완전 상태로 남게 됩니다. `zoneadm uninstall` 명령을 사용하여 영역을 구성됨 상태로 재설정합니다.

`zonepath`가 제거되지 않을 경우 해당 영역이 다른 부트 환경에 설치되어 있을 수 있습니다. 지정한 `zonepath`를 포함하는 영역이 설치되어 있는 부트 환경이 존재할 경우 `zonepath` 및 `zonepath` 데이터 집합에 있는 다양한 데이터 집합이 제거되지 않습니다. 부트 환경에 대한 자세한 내용은 `beadm`(1M)을 참조하십시오.

`uninstall` 명령은 작업을 되돌릴 수 없기 때문에 신중히 사용해야 합니다.

동일한 시스템에서 비전역 영역 복제

복제는 원본 `zonepath`의 데이터를 대상 `zonepath`로 복사하는 방법으로 시스템에 새 영역을 제공할 때 사용됩니다.

원본 `zonepath`와 대상 `zonepath`가 모두 ZFS에 있고 동일한 풀에 있을 경우 `zoneadm clone` 명령은 자동으로 ZFS를 사용하여 영역을 복제합니다. 하지만 ZFS 복제가 아니라 ZFS `zonepath`가 복사되도록 지정할 수 있습니다.

▼ 영역 복제 방법

새 영역을 설치하려면 먼저 해당 영역을 구성해야 합니다. `zoneadm create` 하위 명령에 전달되는 매개변수는 복제할 영역의 이름입니다. 이 원본 영역을 정지해야 합니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 관리자로 전환합니다.

2 복제할 원본 영역을 정지합니다. 이 절차에서는 `my-zone`입니다.

```
global# zoneadm -z my-zone halt
```

3 원본 영역 `my-zone`의 구성을 파일(예: `master`)로 내보내서 새 영역 구성을 시작합니다.

```
global# zonecfg -z my-zone export -f /zones/master
```

주 – 기존 구성을 수정하는 대신 243 페이지 "영역 구성 방법" 절차를 사용하여 새 영역 구성을 만들 수도 있습니다. 이 방법을 사용할 경우 영역을 만든 후 6단계로 건너뛰십시오.

4 **master 파일을 편집합니다. 영역 간에 동일할 수 없는 구성 요소에 대해 서로 다른 등록 정보와 리소스를 설정합니다. 예를 들어 새로운 zonepath를 설정해야 합니다. 공유 IP 영역의 경우 넷 리소스의 IP 주소를 변경해야 합니다. 배타적 IP 영역의 경우 넷 리소스의 물리적 등록 정보를 변경해야 합니다.**

5 ***master* 파일의 명령을 사용하여 새 영역 zone1을 만듭니다.**

```
global# zonecfg -z zone1 -f /zones/master
```

6 **my-zone을 복제하여 새 영역 zone1을 설치합니다.**

```
global# zoneadm -z zone1 clone my-zone
```

다음 화면이 표시됩니다.

```
Cloning zonepath /zones/my-zone...
```

7 **시스템의 영역 목록을 표시합니다.**

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
 -  zone1    installed    /zones/zone1                   solaris    excl
```

예 19–3 복제된 영역에 시스템 구성 프로파일 적용

구성 프로파일을 포함하려면 다음과 같이 합니다.

```
# zoneadm -z zone1 clone -c /path/config.xml my-zone
```

구성 파일에 절대 경로를 제공하십시오.

비전역 영역 이동

이 절차는 zonepath를 변경하여 영역을 동일한 시스템의 새 위치로 이동하는 데 사용됩니다. 영역을 정지해야 합니다. 새 zonepath는 로컬 파일 시스템에 있어야 합니다. 222 페이지 "리소스 유형과 등록 정보"에서 설명한 일반적인 zonepath 조건이 적용됩니다.

이 내용은 solaris10 브랜드 영역을 이동하는 데에도 적용됩니다. solaris10 브랜드 영역에 대한 자세한 내용은 제3부를 참조하십시오.

주 – 다른 BE에 있는 영역을 이동할 수 없습니다. 이러한 BE를 먼저 삭제하거나 영역을 복제하여 새 경로에 새 영역을 만들어야 합니다.

▼ 영역 이동 방법

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 수퍼 유저이거나 해당 권한을 부여받아야 합니다.

2 이동할 영역을 정지합니다. 이 절차에서는 db-zone 영역을 정지합니다.

```
global# zoneadm -z db-zone halt
```

3 zoneadm 명령과 move 하위 명령을 함께 사용하여 영역을 새 zonepath인 /zones/db-zone으로 이동합니다.

```
global# zoneadm -z db-zone move /zones/db-zone
```

4 경로를 확인합니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  my-zone  installed    /zones/my-zone                 solaris    excl
 -  db-zone  installed    /zones/db-zone                 solaris    excl
```

시스템에서 비전역 영역 삭제

이 단원에서 설명하는 절차는 시스템에서 영역을 완전히 삭제합니다.

▼ 비전역 영역 제거 방법

1 다음 방법 중 하나를 사용하여 my-zone 영역을 종료합니다. zoneadm shutdown 방법을 사용하는 것이 좋습니다.

- **zoneadm 사용:**

  ```
  global# zoneadm -z my-zone shutdown
  my-zone
  ```

- **zlogin 사용:**

  ```
  global# zlogin my-zone shutdown
  my-zone
  ```

2 my-zone의 루트 파일 시스템을 제거합니다.

```
global# zoneadm -z my-zone uninstall -F
```

작업을 강제 수행하는 -F 옵션은 일반적으로 필요하지 않습니다.

3 my-zone의 구성을 삭제합니다.

```
global# zonecfg -z my-zone delete -F
```

작업을 강제 수행하는 -F 옵션은 일반적으로 필요하지 않습니다.

4 시스템의 영역 목록을 표시하여 my-zone이 더 이상 나열되지 않는지 확인합니다.

```
global# zoneadm list -iv
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
```

◆ ◆ ◆ 20 장

20 비전역 영역 로그인(개요)

이 장에서는 전역 영역에서 영역에 대한 로그인을 설명합니다.

이 장에서는 다음 항목을 다룹니다.

절차 및 사용 정보는 21 장, “비전역 영역에 로그인(작업)”을 참조하십시오. 사용 가능한 전체 옵션 목록은 zlogin(1) 매뉴얼 페이지를 참조하십시오.

zlogin 명령

RBAC를 사용 중인 경우 영역 콘솔에 액세스하려면 solaris.zone.manage/*zonename* 권한이 필요합니다. 슬래시 문자(/) 뒤에 오는 특정 *zonename*은 선택 사항입니다. 생략할 경우 권한이 모든 영역과 일치합니다.

-C 옵션을 사용하여 영역 콘솔에 연결하는 경우가 아니면 zlogin을 사용하여 영역에 로그인할 경우 새 작업이 시작됩니다. 하나의 작업이 두 영역에 걸쳐 수행될 수 없습니다.

zlogin 명령은 전역 영역에서 실행 중 상태 또는 준비 상태에 있는 영역에 로그인하는 데 사용됩니다.

주 – zlogin 명령을 -C 옵션과 함께 사용해야만 실행 중 상태가 아닌 영역에 로그인할 수 있습니다.

---

296 페이지 “비대화식 모드를 사용하여 영역에 액세스하는 방법”에 설명된 바와 같이 영역 내에서 실행할 명령을 제공하여 `zlogin` 명령을 비대화식으로 사용할 수 있습니다. 하지만 이 명령이나 이 명령이 작용하는 파일은 NFS에 상주할 수 없습니다. 열린 파일이나 주소 공간의 한 부분이 NFS에 상주하는 경우에는 이 명령이 실패합니다. 주소 공간에는 실행 가능한 명령 자체와 명령의 연결된 라이브러리가 포함됩니다.

`zlogin` 명령은 전역 영역에서 작동하는 해당 권한을 가진 사용자나 전역 관리자만 사용할 수 있습니다. 자세한 내용은 `zlogin(1)` 매뉴얼 페이지를 참조하십시오.

내부 영역 구성

시스템 구성은 단일 프로파일(`sc_profile.xml`) 또는 SMF 프로파일의 디렉토리(`profiles`)로 존재할 수 있습니다. 단일 파일이나 디렉토리 모두 영역 설치 시 자동화된 설치 프로그램으로 전달될 영역 시스템 구성 데이터를 설명합니다. 영역 설치 중에 `sc_profile.xml` 파일이나 `profiles` 디렉토리를 지정하지 않은 경우 `sysconfig` 대화식 도구가 콘솔 `zlogin` 명령을 처음 사용할 때 관리자에게 이 데이터를 쿼리합니다.

Oracle Solaris 11에서는 SMF를 사용하여 구성 정보를 중앙 집중화합니다.

Oracle Solaris 인스턴스는 설치 중에 생성 및 구성됩니다. Oracle Solaris 인스턴스는 전역 또는 비전역 영역에서 부트 환경으로 정의됩니다. `sysconfig` 유틸리티를 사용하여 Oracle Solaris 인스턴스에 대한 구성 작업을 수행하거나 Oracle Solaris 인스턴스의 구성을 취소했다가 해당 인스턴스를 다시 구성할 수 있습니다. `sysconfig` 명령을 사용하여 SMF 프로파일을 만들 수 있습니다.

시스템 구성이 필요한 전역 또는 비전역 영역에서 Solaris 인스턴스를 설치하거나 만든 후에는 시스템이 자동으로 구성됩니다. 시스템 구성은 시스템 ID를 유지하기 위해 `-p` 옵션을 지정한 `zoneadm clone` 작업의 경우나 `-c`*profile.xml*`sysconfig` 파일 옵션을 지정하지 않은 `attach` 작업의 경우에는 필요하지 않습니다.

다음과 같은 작업을 수행할 수 있습니다.

- `sysconfig configure` 명령을 사용하여 해당 Oracle Solaris 인스턴스를 다기 구성합니다(구성을 취소했다가 구성).
  - `sysconfig configure` 명령을 사용하여 해당 Oracle Solaris 인스턴스를 구성하고 SCI 도구를 콘솔에서 시작되게 합니다.

    ```
    # sysconfig configure
    ```
  - `sysconfig configure` 명령을 사용하여 전역 영역 또는 비전역 영역에서 구성이 취소된 Solaris 인스턴스를 구성합니다.

    ```
    # sysconfig configure -c sc_profile.xml
    ```

    기존 구성 프로파일을 명령에 지정하면 비대화식 구성이 수행됩니다. 이 명령과 함께 기존 구성 프로파일을 지정하지 않으면 SCI(시스템 구성 대화식) 도구가 실행됩니다. SCI 도구를 사용하면 해당 Oracle Solaris 인스턴스에 대한 특정 구성 정보를 제공할 수 있습니다.

- sysconfig create-profile 명령을 사용하여 새로운 시스템 구성 프로파일을 만들 수 있습니다.

sysconfig 인터페이스는 **Oracle Solaris 11 시스템**의 6 장, "Oracle Solaris 인스턴스 구성 해제 또는 재구성" 및 sysconfig(1M) 매뉴얼 페이지에 설명되어 있습니다.

시스템 구성 대화식 도구

SCI(시스템 구성 대화식) 도구를 사용하면 새로 설치된 Oracle Solaris 11 인스턴스에 대한 구성 매개 변수를 지정할 수 있습니다.

-c *profile.xml* 옵션 없이 sysconfig configure만 사용하면 시스템의 구성이 취소되며 SCI 도구에서 관리자에게 쿼리하고 구성을 /etc/svc/profile/site/scit_profile.xml에 기록합니다. 그러고 나서 이 도구는 이 정보로 시스템을 구성합니다.

sysconfig create-profile은 관리자에게 쿼리하고 /system/volatile/scit_profile.xml에 SMF 프로파일 파일을 만듭니다. 매개 변수에는 시스템 호스트 이름, 표준 시간대, 사용자 및 루트 계정, 이름 서비스가 포함됩니다.

도구를 탐색하려면:

- 각 화면 하단에 나열된 기능 키를 사용하여 화면 간에 이동하고 다른 작업을 수행합니다. 키보드에 기능 키가 없거나 키가 응답하지 않으면 Esc 키를 누릅니다. 화면 아래쪽에 있는 범례가 변경되어 탐색과 그 밖의 기능에 사용할 수 있는 ESC 키가 표시됩니다.
- 위쪽 및 아래쪽 화살표 키를 사용하여 선택을 변경하거나 입력 필드 간에 이동합니다.

자세한 내용은 **Oracle Solaris 11 시스템**의 6 장, "Oracle Solaris 인스턴스 구성 해제 또는 재구성" 및 sysconfig(1M) 매뉴얼 페이지를 참조하십시오.

영역 구성 프로파일 예

자동 구성된 배타적 IP 영역:

```
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="sysconfig">
  <service version="1" type="service" name="system/config-user">
    <instance enabled="true" name="default">
      <property_group type="application" name="root_account">
        <propval type="astring" name="login" value="root"/>
        <propval type="astring" name="password" value="$5$KeNRy1zU$lqzy9rIsNloUhfvJFIWmVewE75aB5/EBA77kY7EP6F0"/>
        <propval type="astring" name="type" value="role"/>
      </property_group>
      <property_group type="application" name="user_account">
        <propval type="astring" name="login" value="admin1"/>
        <propval type="astring" name="password" value="$5$/g353K5q$V8Koe/XuAeR/zpBvpLsgVIqPrvc.9z0hYFYoyoBkE37"/>
        <propval type="astring" name="type" value="normal"/>
```

```
        <propval type="astring" name="description" value="admin1"/>
        <propval type="count" name="gid" value="10"/>
        <propval type="astring" name="shell" value="/usr/bin/bash"/>
        <propval type="astring" name="roles" value="root"/>
        <propval type="astring" name="profiles" value="System Administrator"/>
        <propval type="astring" name="sudoers" value="ALL=(ALL) ALL"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/timezone">
    <instance enabled="true" name="default">
      <property_group type="application" name="timezone">
        <propval type="astring" name="localtime" value="UTC"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/environment">
    <instance enabled="true" name="init">
      <property_group type="application" name="environment">
        <propval type="astring" name="LC_ALL" value="C"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/identity">
    <instance enabled="true" name="node">
      <property_group type="application" name="config">
        <propval type="astring" name="nodename" value="my-zone"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/keymap">
    <instance enabled="true" name="default">
      <property_group type="system" name="keymap">
        <propval type="astring" name="layout" value="US-English"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/console-login">
    <instance enabled="true" name="default">
      <property_group type="application" name="ttymon">
        <propval type="astring" name="terminal_type" value="vt100"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="network/physical">
    <instance enabled="true" name="default">
      <property_group type="application" name="netcfg">
        <propval type="astring" name="active_ncp" value="Automatic"/>
      </property_group>
    </instance>
  </service>
</service_bundle>
```

DNS 없이 NIS를 사용하여 정적 구성된 배타적 IP:

```
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="sysconfig">
  <service version="1" type="service" name="system/config-user">
    <instance enabled="true" name="default">
```

```
    <property_group type="application" name="root_account">
      <propval type="astring" name="login" value="root"/>
      <propval type="astring" name="password" value="$5$m8OR3zqK$Ox5XGubRJdi4zj0JzNSmVJ3Ni4opDOGpxi2nK/GGzmC"/>
      <propval type="astring" name="type" value="normal"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="system/timezone">
  <instance enabled="true" name="default">
    <property_group type="application" name="timezone">
      <propval type="astring" name="localtime" value="UTC"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="system/environment">
  <instance enabled="true" name="init">
    <property_group type="application" name="environment">
      <propval type="astring" name="LC_ALL" value="C"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="system/identity">
  <instance enabled="true" name="node">
    <property_group type="application" name="config">
      <propval type="astring" name="nodename" value="my-zone"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="system/keymap">
  <instance enabled="true" name="default">
    <property_group type="system" name="keymap">
      <propval type="astring" name="layout" value="US-English"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="system/console-login">
  <instance enabled="true" name="default">
    <property_group type="application" name="ttymon">
      <propval type="astring" name="terminal_type" value="vt100"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="network/physical">
  <instance enabled="true" name="default">
    <property_group type="application" name="netcfg">
      <propval type="astring" name="active_ncp" value="DefaultFixed"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="network/install">
  <instance enabled="true" name="default">
    <property_group type="application" name="install_ipv4_interface">
      <propval type="astring" name="address_type" value="static"/>
      <propval type="net_address_v4" name="static_address" value="10.10.10.13/24"/>
      <propval type="astring" name="name" value="net0/v4"/>
      <propval type="net_address_v4" name="default_route" value="10.10.10.1"/>
    </property_group>
    <property_group type="application" name="install_ipv6_interface">
      <propval type="astring" name="stateful" value="yes"/>
```

```
        <propval type="astring" name="stateless" value="yes"/>
        <propval type="astring" name="address_type" value="addrconf"/>
        <propval type="astring" name="name" value="net0/v6"/>
      </property_group>
    </instance>
  </service>
  <service version="1" type="service" name="system/name-service/switch">
    <property_group type="application" name="config">
      <propval type="astring" name="default" value="files nis"/>
      <propval type="astring" name="printer" value="user files nis"/>
      <propval type="astring" name="netgroup" value="nis"/>
    </property_group>
    <instance enabled="true" name="default"/>
  </service>
  <service version="1" type="service" name="system/name-service/cache">
    <instance enabled="true" name="default"/>
  </service>
  <service version="1" type="service" name="network/dns/client">
    <instance enabled="false" name="default"/>
  </service>
  <service version="1" type="service" name="network/nis/domain">
    <property_group type="application" name="config">
      <propval type="hostname" name="domainname" value="example.net"/>
      <property type="host" name="ypservers">
        <host_list>
          <value_node value="192.168.224.11"/>
        </host_list>
      </property>
    </property_group>
    <instance enabled="true" name="default"/>
  </service>
  <service version="1" type="service" name="network/nis/client">
    <instance enabled="true" name="default"/>
  </service>
</service_bundle>
```

NIS를 사용하여 동적 구성된 배타적 IP 영역:

```
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="sysconfig">
 <service version="1" type="service" name="system/config-user">
   <instance enabled="true" name="default">
     <property_group type="application" name="root_account">
       <propval type="astring" name="login" value="root"/>
       <propval type="astring" name="password" value="$5$Iq/.A.K9$RQyt6RqsAY8TgnuxL9i0/84QwgIQ/nqcK8QsTQdvMy/"/>
       <propval type="astring" name="type" value="normal"/>
     </property_group>
   </instance>
 </service>
 <service version="1" type="service" name="system/timezone">
   <instance enabled="true" name="default">
     <property_group type="application" name="timezone">
       <propval type="astring" name="localtime" value="UTC"/>
     </property_group>
   </instance>
 </service>
 <service version="1" type="service" name="system/environment">
   <instance enabled="true" name="init">
     <property_group type="application" name="environment">
```

```
      <propval type="astring" name="LC_ALL" value="C"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="system/identity">
  <instance enabled="true" name="node">
    <property_group type="application" name="config">
      <propval type="astring" name="nodename" value="my-zone"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="system/keymap">
  <instance enabled="true" name="default">
    <property_group type="system" name="keymap">
      <propval type="astring" name="layout" value="US-English"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="system/console-login">
  <instance enabled="true" name="default">
    <property_group type="application" name="ttymon">
      <propval type="astring" name="terminal_type" value="sun-color"/>
    </property_group>
  </instance>
</service>
<service version="1" type="service" name="system/name-service/switch">
  <property_group type="application" name="config">
    <propval type="astring" name="default" value="files nis"/>
    <propval type="astring" name="printer" value="user files nis"/>
    <propval type="astring" name="netgroup" value="nis"/>
  </property_group>
  <instance enabled="true" name="default"/>
</service>
<service version="1" type="service" name="system/name-service/cache">
  <instance enabled="true" name="default"/>
</service>
<service version="1" type="service" name="network/dns/client">
  <instance enabled="false" name="default"/>
</service>
<service version="1" type="service" name="network/nis/domain">
  <property_group type="application" name="config">
    <propval type="hostname" name="domainname" value="special.example.com"/>
    <property type="host" name="ypservers">
      <host_list>
        <value_node value="192.168.112.3"/>
      </host_list>
    </property>
  </property_group>
  <instance enabled="true" name="default"/>
</service>
<service version="1" type="service" name="network/nis/client">
  <instance enabled="true" name="default"/>
</service>
</service_bundle>
```

비전역 영역 로그인 방법

이 절에서는 영역에 로그인하는 데 사용할 수 있는 방법에 대해 설명합니다.

영역 콘솔 로그인

각 영역에서는 가상 콘솔인 `/dev/console` 이 유지됩니다. 콘솔에서 작업을 수행하는 것을 콘솔 모드라고 합니다. 영역이 설치된 상태에 있는 경우 영역에 대한 콘솔 로그인을 수행할 수 있습니다. 영역 콘솔은 시스템의 직렬 콘솔과 매우 비슷합니다. 콘솔과의 연결이 영역 재부트 시에도 지속됩니다. 콘솔 모드가 `telnet`과 같은 로그인 세션과 어떻게 다른지 알아보려면 291 페이지 "원격 로그인"을 참조하십시오.

영역 콘솔은 `zlogin` 명령과 `-C` 옵션 및 *zonename*을 사용하여 액세스합니다. 영역이 실행 중 상태로 전환되지 않아도 됩니다.

`-d` 옵션도 사용할 수 있습니다. 이 옵션은 영역이 정지되는 경우 영역이 콘솔과의 연결을 끊도록 지정합니다. 이 옵션은 반드시 `-C` 옵션과 함께 지정해야 합니다.

영역 내 프로세스가 메시지를 열어서 콘솔에 쓸 수 있습니다. `zlogin -C` 프로세스가 종료되면 다른 프로세스가 콘솔에 액세스할 수 있습니다.

RBAC(역할 기반 액세스 제어)를 사용 중인 경우 영역 콘솔에 액세스하려면 `solaris.zone.manage/`*zonename* 권한이 필요합니다. 슬래시 문자(`/`) 뒤에 오는 특정 *zonename*은 선택 사항입니다. 생략할 경우 권한이 모든 영역과 일치합니다.

부트 시 SCI(시스템 구성 대화식) 도구를 불러오려면 다음을 입력합니다.

```
root@test2:~# sysconfig configure -s
```

사용자 로그인 방법

사용자 이름을 사용하여 영역에 로그인하려면 `zlogin` 명령과 함께 `-l` 옵션, 사용자 이름 및 *zonename*을 사용합니다. 예를 들면 전역 영역의 관리자는 `zlogin` 에 대해 `-l` 옵션을 지정하여 비전역 영역에서 일반 사용자로 로그인할 수 있습니다.

```
global# zlogin -l user zonename
```

`root` 사용자로 로그인하려면 옵션 없이 `zlogin` 명령을 사용합니다.

비상 안전 모드

로그인 문제가 발생했으며 `zlogin` 명령이나 `zlogin` 명령을 `-C` 옵션과 함께 사용하여 영역에 액세스할 수 없는 경우에는 대안이 제공됩니다. `zlogin` 명령을 `-S`(안전) 옵션과 함께 사용하여 영역으로 들어갈 수 있습니다. 다른 로그인 방법이 실패하는 경우 손상된 영역을 복구하려면 이 모드만 사용하십시오. 이 최소 환경에서는 영역 로그인이 실패하는 이유를 진단할 수 있습니다.

원격 로그인

영역에 원격으로 로그인하는 기능은 설정하는 네트워크 서비스 선택에 따라 달라집니다. `pkg:/service/network/legacy-remote-utilities` 서비스를 사용으로 설정하여 필요할 경우 `rlogin` 및 `telnet`를 통한 로그인을 추가할 수 있습니다.

로그인 명령에 대한 자세한 내용은 `rlogin(1)`, `ssh(1)` 및 `telnet(1)`을 참조하십시오.

대화식 및 비대화식 모드

`zlogin` 명령을 통해 영역에 액세스하고 영역 내에서 명령을 실행하기 위한 두 가지 방법도 제공됩니다. 이 두 가지 방법이란 대화식 모드와 비대화식 모드입니다.

대화식 모드

대화식 모드에서는 해당 영역 내에서 사용할 새 의사 단말기가 할당됩니다. 콘솔 장치에 대한 배타적 액세스 권한이 부여되는 콘솔 모드와는 달리, 대화식 모드에서는 언제든지 원하는 만큼의 `zlogin` 세션을 열 수 있습니다. 비대화식 모드는 실행할 명령을 포함하지 않는 경우에 활성화됩니다. 편집기와 같은 단말기 장치가 필요한 프로그램은 이 모드에서 올바르게 작동합니다.

RBAC(역할 기반 액세스 제어)를 사용 중인 경우 대화식 로그인을 수행하려면 해당 영역에 대한 `solaris.zone.login/`*zonename* 권한이 필요합니다. 영역에서는 암호 인증이 수행됩니다.

비대화식 모드

비대화식 모드는 영역을 관리하는 셸 스크립트를 실행하는 데 사용됩니다. 비대화식 모드는 새 의사 단말기를 할당하지 않습니다. 비대화식 모드는 사용자가 영역 내에서 실행할 명령을 제공할 때 사용됩니다.

비대화식 로그인을 수행하거나 암호 인증을 생략하려면 `solaris.zone.manage/`*zonename* 권한이 필요합니다.

◆ ◆ ◆ 21 장

21 비전역 영역에 로그인(작업)

이 장에서는 설치된 영역의 구성을 완료하고, 전역 영역에서 영역에 로그인하며, 영역을 종료하는 절차를 설명합니다. 또한 zonename 명령을 사용하여 현재 영역의 이름을 인쇄하는 방법을 보여 줍니다.

영역 로그인 프로세스에 대한 개요는 20 장, “비전역 영역 로그인(개요)”을 참조하십시오.

초기 영역 부트 및 영역 로그인 절차(작업 맵)

작업	설명	수행 방법
내부 구성을 수행하거나 영역 구성을 해제합니다.	시스템 구성은 텍스트 형식의 사용자 인터페이스를 사용하여 대화식으로 수행하거나 프로파일을 사용하여 비대화식으로 수행할 수 있습니다. sysconfig 유틸리티는 Solaris 인스턴스를 구성 해제하는 데에도 사용됩니다.	**Oracle Solaris 11 시스템**의 6 장, “Oracle Solaris 인스턴스 구성 해제 또는 재구성” 및 sysconfig(1M) 매뉴얼 페이지를 참조하십시오.
영역에 로그인합니다.	콘솔을 통해 영역에 로그인하거나, 대화식 모드를 사용하여 의사 터미널을 할당하거나, 영역에서 실행할 명령을 입력하여 영역에 로그인할 수 있습니다. 실행할 명령을 입력하는 방법은 의사 터미널을 할당하지 않습니다. 영역에 대한 연결이 거부될 경우 비상 안전 모드를 사용하여 로그인할 수도 있습니다.	294 페이지 “영역에 로그인”

작업	설명	수행 방법
비전역 영역을 종료합니다.	비전역 영역에 대한 연결을 끊습니다.	297 페이지 "비전역 영역 종료 방법"
영역을 종료합니다.	shutdown 유틸리티나 스크립트를 사용하여 영역을 종료합니다.	297 페이지 "zlogin을 사용하여 영역을 종료하는 방법"
영역 이름을 인쇄합니다.	현재 영역의 영역 이름을 인쇄합니다.	298 페이지 "현재 영역 이름 인쇄"

영역에 로그인

zlogin 명령을 사용하여 전역 영역에서 실행 중이거나 준비 상태의 영역에 로그인합니다. 자세한 내용은 zlogin(1) 매뉴얼 페이지를 참조하십시오.

다음 절차에서 설명하는 다양한 방법을 사용하여 영역에 로그인할 수 있습니다. 291 페이지 "원격 로그인"에서 설명한 대로 원격으로 로그인할 수도 있습니다.

▼ 구성 프로파일을 만드는 방법

주의 – 필요한 모든 데이터를 입력해야 합니다. 데이터가 누락된 프로파일을 제공하면 데이터가 누락된 영역이 구성됩니다. 구성에 데이터가 누락되면 사용자가 로그인하지 못하거나 실행 중인 네트워크에 연결하지 못할 수 있습니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 관리자로 전환합니다.

2 sysconfig 도구를 사용하여 프로파일을 만듭니다.

- **배타적 IP 영역**

  ```
  # sysconfig create-profile -o /path/sysconf.xml
  ```

- **공유 IP 영역:**

```
# sysconfig create-profile -o /path/sysconf.xml -g location,identity,naming_services,users
```

3 영역 설치, 복제 또는 연결 작업 중 생성된 프로파일을 사용합니다.

```
# zoneadm -z my-zone install -c /path/sysconf.xml
```

구성 파일을 사용할 경우 최초로 zlogin을 실행할 때 콘솔에서 SCI(System Configuration Interactive) 도구가 시작되지 **않습니다**. 파일 인수는 절대 경로로 지정해야 합니다.

▼ 영역 콘솔에 로그인하여 내부 영역 구성을 수행하는 방법

config.xml 파일이 zoneadm clone, attach 또는 install 명령으로 전달된 경우 시스템을 구성하는 데 이 구성 파일이 사용됩니다. clone, attach 또는 install 작업 중 config.xml 파일이 제공되지 않은 경우 영역을 처음 부트할 때 콘솔에서 SCI 도구가 시작됩니다.

구성 정보에 대한 초기 프롬프트가 누락되지 않도록 하려면 영역이 두번째 세션에서 부트되기 전에 zlogin이 실행되도록 두 개의 터미널 창을 사용하는 것이 좋습니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **zlogin 명령에 -C 옵션과 영역 이름(my-zone)을 지정하여 사용합니다.**

```
global# zlogin -C my-zone
```

3 **다른 단말기 창에서 영역을 부트합니다.**

```
global# zoneadm -z my-zone boot
```

zlogin 터미널 창에 다음과 유사한 내용이 표시됩니다.

```
[NOTICE: Zone booting up]
```

4 **새로 설치한 영역의 구성 매개변수에 대한 일련의 질문에 답합니다. 매개변수에는 시스템 호스트 이름, 표준 시간대, 사용자 및 루트 계정, 이름 서비스가 포함됩니다. 기본적으로 SCI 도구는 SMF 프로파일 파일을 /system/volatile/scit_profile.xml에 생성합니다.**

일반 오류 초기 SCI 화면이 나타나지 않으면 Ctrl L을 입력하여 SCI 화면을 새로 고칩니다.

▼ 영역 콘솔에 로그인하는 방법

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **zlogin 명령에 -C 옵션, -d 옵션 및 영역 이름(예: my-zone)을 지정하여 사용합니다.**

```
global# zlogin -C -d my-zone
```

구성이 수행되지 않은 경우 zlogin 명령에 -C 옵션을 사용하면 SCI 도구가 시작됩니다.

3 **영역 콘솔이 표시되면 root로 로그인하고 Return을 누른 후 프롬프트가 표시되면 루트 암호를 입력합니다.**

```
my-zone console login: root
Password:
```

▼ 대화식 모드를 사용하여 영역에 액세스하는 방법

대화식 모드에서는 영역 내부에서 사용하도록 새 의사 터미널이 할당됩니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **전역 영역에서 영역(예: my-zone)에 로그인합니다.**

```
global# zlogin my-zone
```

다음과 유사한 내용이 표시됩니다.

```
[Connected to zone 'my-zone' pts/2]
Last login: Wed Jul  3 16:25:00 on console
```

3 **exit를 입력하여 연결을 종료합니다.**

다음과 유사한 메시지가 표시됩니다.

```
[Connection to zone 'my-zone' pts/2 closed]
```

▼ 비대화식 모드를 사용하여 영역에 액세스하는 방법

사용자가 영역 내부에서 실행할 명령을 입력할 경우 비대화식 모드가 사용됩니다. 비대화식 모드는 새 의사 단말기를 할당하지 않습니다.

명령이나 명령이 사용하는 파일은 NFS에 상주할 수 없습니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **전역 영역에서 my-zone 영역에 로그인한 후 명령 이름을 입력합니다.**

여기서는 zonename이 사용됩니다.

```
global# zlogin my-zone zonename
```

다음과 같은 출력이 표시됩니다.

```
my-zone
```

▼ 비전역 영역 종료 방법

- **비전역 영역에 대한 연결을 끊으려면 다음 방법 중 하나를 사용합니다.**
  - **영역 비가상 콘솔을 종료하려면 다음과 같이 합니다.**

    ```
    zonename# exit
    ```

  - **영역 가상 콘솔에 대한 연결을 끊으려면 틸드(~) 문자와 마침표를 사용합니다.**

    ```
    zonename# ~.
    ```

    화면에 다음과 유사한 내용이 표시됩니다.

    ```
    [Connection to zone 'my-zone' pts/6 closed]
    ```

주 – ssh의 기본 제어 시퀀스 또한 ~이며 이는 ssh 세션을 종료합니다. ssh를 사용하여 서버에 원격으로 로그인할 경우 ~~. 를 사용하여 영역을 종료합니다.

참조 zlogin 명령 옵션에 대한 자세한 내용은 zlogin(1) 설명서 페이지를 참조하십시오.

▼ 비상 안전 모드를 사용하여 영역에 액세스하는 방법

영역에 대한 연결이 거부될 경우 zlogin 명령에 -S 옵션을 사용하여 영역에 최소 환경으로 액세스할 수 있습니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1. **관리자로 전환합니다.**
2. **전역 영역에서 zlogin 명령에 -S 옵션을 사용하여 영역(예: my-zone)에 액세스합니다.**

   ```
   global# zlogin -S my-zone
   ```

▼ zlogin을 사용하여 영역을 종료하는 방법

주 – 전역 영역에서 init 0을 실행하여 Oracle Solaris 시스템을 완전히 종료하면 시스템의 각 비전역 영역에서도 init 0이 실행됩니다. init 0은 로컬 사용자와 원격 사용자에게 시스템이 종료되기 전에 로그오프하라는 경고를 표시하지 않습니다.

영역을 완전히 종료하려면 이 절차를 사용합니다. 종료 스크립트를 실행하지 않고 영역을 정지하려면 275 페이지 “영역 정지 방법”을 참조하십시오.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **종료할 영역(예: `my-zone`)에 로그인한 후 유틸리티 이름으로 `shutdown`을, 상태로 `init 0`을 지정합니다.**

```
global# zlogin my-zone shutdown -i 0
```

사이트에 해당 환경에 맞는 자체 종료 스크립트가 있을 수 있습니다.

서비스를 사용으로 설정

영역에서 개별 서비스를 사용 또는 사용 안함으로 설정할 수 있습니다.

현재 영역 이름 인쇄

zonename(1) 매뉴얼 페이지에 설명되어 있는 zonename 명령은 현재 영역의 이름을 인쇄합니다. 다음 예는 전역 영역에서 zonename을 사용할 때의 출력을 보여 줍니다.

```
# zonename
global
```

◆ ◆ ◆ 22장

22 영역 마이그레이션 및 zonep2vchk 도구 정보

이 장에서는 다음에 대해 개괄적으로 설명합니다.

- 물리적-가상 마이그레이션 - 시스템을 비전역 영역으로 마이그레이션
- 가상-가상 마이그레이션 - 기존 영역을 새 시스템으로 마이그레이션

이 장에서는 시스템에서 영역 마이그레이션에 사용되는 `zonep2vchk` 도구에 대해서도 설명합니다.

물리적-가상 개념과 가상-가상 개념

P2V 및 V2V는 다음 작업에 사용할 수 있습니다.

- 단일 서버에서 여러 응용 프로그램 통합
- 작업 로드 균형 조정
- 서버 교체
- 서버 교체
- 재해 복구

마이그레이션 전략 선택

`zonepath`를 새 호스트에서 볼 수 있도록 SAN 기반 저장소를 재구성할 수 있습니다.

한 시스템의 모든 영역을 다른 시스템으로 이동해야 하는 경우 복제 스트림에는 스냅샷과 복제본을 보존하는 장점이 있습니다. 스냅샷과 복제본은 `pkg`, `beadm create` 및 `zoneadm clone` 명령에 광범위하게 사용됩니다.

다섯 단계를 통해 P2V 또는 V2V 마이그레이션을 수행할 수 있습니다.

1. P2V의 경우 모든 Oracle Solaris 구성에 대해 소스 호스트를 분석합니다.
   - 네트워킹 요구 사항에 따라 비전역 영역의 IP 유형(배타적 IP 또는 공유 IP)을 결정합니다.

- 대상 호스트의 전역 영역에서 추가 구성이 필요한지 여부를 결정합니다.
- 응용 프로그램 데이터와 파일 시스템이 마이그레이션되는 방법을 결정합니다.

`-b` 옵션을 사용하여 수행된 `zonep2vchk` 기본 분석은 소스 전역 영역에서 사용되는 Oracle Solaris 구성 또는 기능과 관련된 기본 문제를 식별합니다. `-s` 옵션을 사용하는 `zonep2vchk` 정적 분석은 소스 전역 영역에서 특정 응용 프로그램과 관련된 문제를 식별하는 데 도움이 됩니다. `-r`을 사용하여 수행한 `zonep2vchk` 런타임 분석은 현재 실행 중인 응용 프로그램을 검사하여 영역에서 작동하지 않을 수 있는 작업을 확인합니다.

2. 소스 시스템 또는 영역을 아카이브합니다. 이 Oracle Solaris 인스턴스 아카이브 단계에서는 따로 마이그레이션해야 하는 데이터를 잠재적으로 제외합니다.
   - Oracle Solaris 10 전역 영역을 아카이브하려면 `flarcreate`를 사용할 수 있습니다.

     391 페이지 “`flarcreate`를 사용하여 이미지를 만드는 방법”을 참조하십시오.
   - Oracle Solaris 10 시스템과 비전역 영역을 아카이브하려면 `flarcreate` 명령을 `-R` 또는 `-L` **아카이버**와 함께 사용하여 특정 파일을 아카이브에서 제외할 수 있습니다. 먼저 영역을 정지해야 합니다.

     391 페이지 “`flarcreate`를 사용하여 특정 데이터를 제외하는 방법”을 참조하십시오.
   - Oracle Solaris 11 전역 영역의 경우 `zfs send`를 사용하여 루트 풀을 아카이브할 수 있습니다.
   - Oracle Solaris 11 비전역 영역의 경우 `zfs send`를 사용하여 영역의 `zonepath` 데이터 집합을 아카이브할 수 있습니다.
   - SAN과 같이 공유 저장소에서 `zpool`에 상주하는 `solaris10` 또는 `solaris` 영역의 경우 V2V 마이그레이션 전략에서는 아카이브를 만들 필요가 없습니다. 다음을 수행하여 `zonepath`가 새 호스트에 표시되도록 SAN 기반 저장소를 재구성할 수 있습니다.
     - 대상 전역 영역에서 `zpool`을 내보낸 다음 가져옵니다.
     - 대상 시스템에서 `zoneadm attach`를 사용합니다. 이 단원의 5단계를 참조하십시오.
3. 다음과 같이 추가 데이터 및 파일 시스템에 대한 마이그레이션 전략을 선택합니다.
   - 아카이브에 데이터를 포함시킵니다. 이 단원의 2단계를 참조하십시오.
   - `zfs send`와 같은 기본 아카이브 형식을 사용하여 데이터를 따로 아카이브하고 마이그레이션 후 영역에서 데이터를 복원합니다.
   - 대상 전역 영역에서 SAN 저장소에 액세스하고 `zonecfg add fs`를 사용하여 영역에서 데이터를 사용할 수 있도록 만드는 방법으로 SAN 데이터를 마이그레이션합니다.

- 소스 호스트에서 zpool을 내보내고 저장소를 이동한 다음 대상 전역 영역에서 zpool을 가져와서 ZFS zpools의 저장소를 마이그레이션할 수 있습니다. 그런 다음 zonecfg add dataset 또는 zonecfg add fs를 사용하여 ZFS 파일 시스템을 대상 영역에 추가할 수 있습니다. SAN 저장소의 zpools도 이 방법으로 마이그레이션할 수 있습니다.

4. 대상 호스트에서 대상 영역에 대해 영역 구성(zonecfg)을 만듭니다.
   - P2V에서는 zonep2vchk 명령과 함께 -c 옵션을 사용하여 구성 만들기를 지원합니다.
   - V2V에서는 소스 호스트에서 zonecfg -z source_zone export 명령을 사용합니다. Oracle Solaris 10 Containers를 Oracle Solaris 10 Zones로 마이그레이션할 때는 브랜드를 solaris10으로 설정해야 합니다.

   내보낸 zonecfg를 필요에 따라 검토하고 수정합니다. 네트워킹 리소스를 업데이트할 경우를 예로 들 수 있습니다.
5. 아카이브를 사용하여 대상 호스트에 영역을 설치(P2V)하거나 연결(V2V)합니다. 새 sysconfig 프로파일을 제공하거나 처음 부트할 때 sysconfig 유틸리티를 실행할 수 있습니다.

zonep2vchk 도구를 사용하여 시스템 마이그레이션 준비

이 절에서는 zonep2vchk 도구를 설명합니다. 이 도구에 대한 기본 설명서는 zonep2vchk(1M) 매뉴얼 페이지입니다.

zonep2vchk 도구 정보

P2V 프로세스는 전역 영역(소스)을 아카이브한 다음 해당 아카이브를 사용하여 비전역 영역(대상)을 설치하는 작업으로 이루어집니다. 0의 유효한 사용자 ID를 사용하여 zonep2vchk 유틸리티를 실행해야 합니다.

유틸리티의 이점은 다음과 같습니다.

- 소스 시스템 구성에서 문제 영역 식별
- 필요한 수동 재구성 작업 최소화
- Oracle Solaris 11의 영역으로 Oracle Solaris 10 및 Oracle Solaris 11 시스템 이미지의 마이그레이션 지원
- 여러 IP 인터페이스, IP 다중 경로, VLAN을 포함하여 원래 시스템 이미지에서 복잡한 네트워크 구성 지원

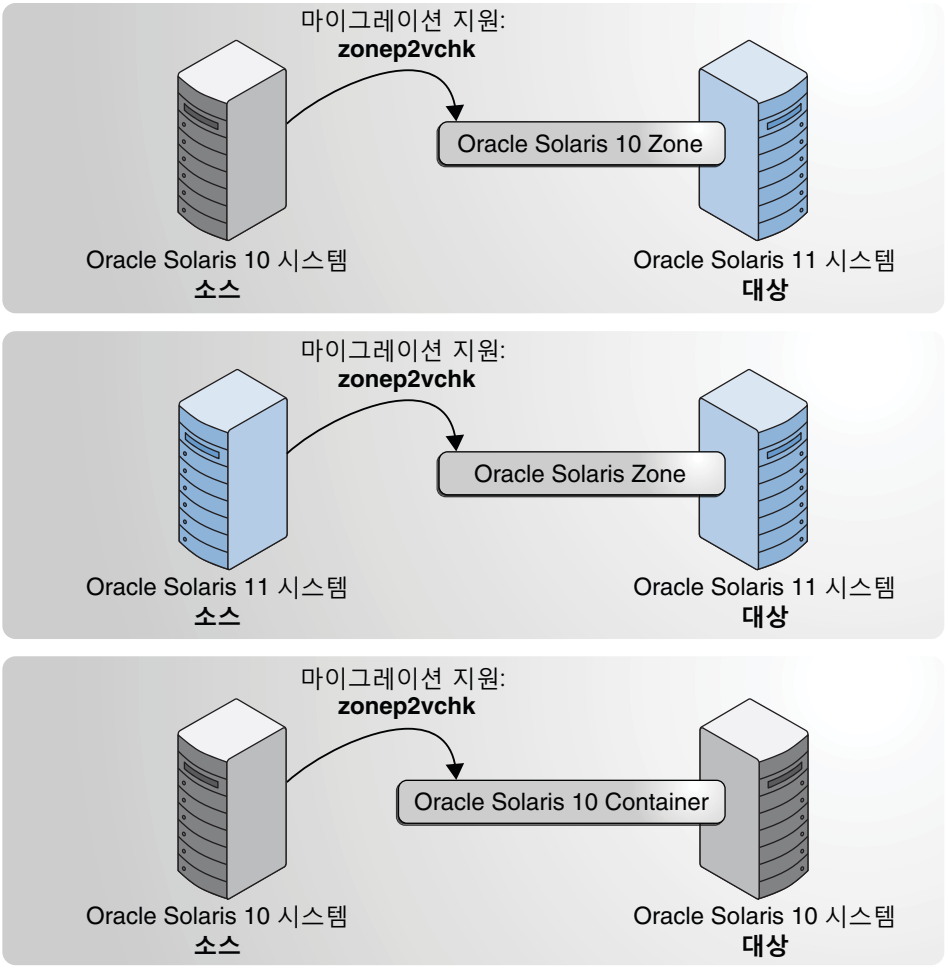
이 도구를 사용하여 Oracle Solaris 11 물리적 시스템 또는 Oracle Solaris 10 물리적 시스템을 이 릴리스의 비전역 영역으로 마이그레이션할 수 있습니다.

- Oracle Solaris 11 시스템을 solaris 브랜드 영역으로 마이그레이션

- Oracle Solaris 10 시스템을 `solaris10` 브랜드 영역으로 마이그레이션

Oracle Solaris 11 대상 시스템의 경우 `add anet` 리소스(VNIC)는 소스 시스템에서 각 네트워크 리소스에 대한 `zonecfg` 출력에 포함됩니다. 기본적으로 배타적 IP는 Oracle Solaris 11 시스템 또는 Oracle Solaris 10 시스템을 Oracle Solaris 11 시스템의 비전역 영역으로 마이그레이션하는 경우 네트워크 유형입니다.

그림 22–1 zonep2vchk 유틸리티



분석 유형

기본 분석(-b 옵션)은 P2V 마이그레이션의 영향을 받을 수 있는 사용 중인 Oracle Solaris 기능을 확인합니다.

정적 분석(-s, 옵션)은 바이너리를 검사하여 영역에서 작동하지 않을 수 있는 시스템 및 라이브러리 호출을 확인합니다.

런타임 분석(-r 옵션)은 현재 실행 중인 응용 프로그램을 검사하여 영역에서 작동하지 않을 수 있는 작업을 확인합니다.

생성된 정보

다음 두 가지 주요 범주의 정보가 분석에서 제공됩니다.

- 특정 영역 구성 또는 전역 영역의 구성 변경을 통해 처리할 수 있는 문제
- 영역 내부에서 작동할 수 없는 기능 식별

예를 들어, 응용 프로그램이 시스템 시계를 설정할 경우 적절한 권한을 영역에 추가하여 이 설정을 사용할 수 있지만, 응용 프로그램이 커널 메모리에 액세스할 경우에는 영역 내에서 이 설정이 허용되지 않습니다. 출력에서는 이러한 두 가지 등급의 문제가 구별됩니다.

기본적으로 유틸리티는 육안으로 읽을 수 있는 형식으로 메시지를 인쇄합니다. 시스템에서 구문 분석할 수 있는 형식으로 메시지를 인쇄하려면 -P 옵션을 사용합니다. 사용 가능한 옵션과 명령 호출 및 출력에 대한 자세한 내용은 zonep2vchk(1M) 매뉴얼 페이지를 참조하십시오.

◆ ◆ ◆ 23장

23 Oracle Solaris 시스템 마이그레이션 및 비전역 영역(작업) 마이그레이션

이 장에서는 Oracle Solaris 11 시스템을 대상 Oracle Solaris 11 시스템의 비전역 영역으로 마이그레이션하는 방법을 설명합니다. 또한, 소스 Oracle Solaris 11 시스템을 마이그레이션하기 전에 소스 시스템의 기존 `solaris` 영역을 새 대상 시스템으로 마이그레이션하는 방법을 설명합니다.

이 정보는 `solaris10` 브랜드 영역의 마이그레이션에도 적용됩니다. `solaris10` 브랜드 영역에 대한 자세한 내용은 제3부를 참조하십시오.

비전역 영역을 다른 시스템으로 마이그레이션

영역 마이그레이션 정보

`zonecfg` 및 `zoneadm` 명령을 사용하여 한 시스템에서 다른 시스템으로 기존 비전역 영역을 마이그레이션할 수 있습니다. 영역이 중지되고 현재 호스트에서 분리됩니다. `zonepath`는 연결된 대상 호스트로 이동합니다.

다음 요구 사항이 영역 마이그레이션에 적용됩니다.

- 마이그레이션하기 전에 소스 시스템에서 비활성 BE를 모두 제거해야 합니다.
- 대상 시스템의 전역 영역에서 원래 소스 호스트와 동일한 Oracle Solaris 11 릴리스가 실행되고 있어야 합니다.
- 영역이 제대로 실행하려면 대상 시스템에 필요한 운영 체제 패키지가 설치되어 있어야 하며 패키지의 버전이 원래 소스 호스트에 설치된 것과 같거나 그 이상의 버전이어야 합니다.

  타사 제품용 패키지 등의 기타 패키지는 다를 수 있습니다.
- 새 호스트에 영역 종속 항목 패키지의 최신 버전이 설치된 경우 `zoneadm attach`를 `-u` 또는 `-U` 옵션과 함께 사용하면 새 호스트와 일치하도록 영역 내 해당 패키지를 업데이트할 수 있습니다. 연결 시 업데이트 소프트웨어는 마이그레이션되는 영역을

살펴보고 새 호스트와 일치하도록 업데이트해야 할 패키지를 결정합니다. 해당 패키지만 업데이트됩니다. 패키지의 나머지 부분은 영역마다 다를 수 있습니다. 영역 내부에 설치되었지만 전역 영역에 설치되지 않은 패키지는 무시되고 있는 그대로 유지됩니다.

`zoneadm detach` 프로세스는 다른 시스템에서 영역을 연결하는 데 필요한 정보를 만듭니다. `zoneadm attach` 프로세스는 대상 시스템에 영역을 호스팅하기 위한 올바른 구성이 있는지 확인합니다.

여러 가지 방법으로 `zonepath`를 새 호스트에서 사용 가능하게 만들 수 있으므로 한 시스템에서 다른 시스템으로 `zonepath`를 실제로 이동하는 작업은 전역 관리자가 수행하는 수동 프로세스입니다.

새 시스템에 연결된 경우 영역은 설치됨 상태가 됩니다.

▼ ZFS 아카이브를 사용하여 비전역 영역을 마이그레이션하는 방법

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 적합한 인증을 받은 사용자여야 합니다.

이 예에서는 영역의 아카이브를 만들고 나서 해당 아카이브를 다른 시스템에 연결하는 방법을 설명합니다. 여기서는 소스 및 대상 호스트의 관리자가 임시 파일 저장소로 공유 NFS 서버에 액세스할 수 있다고 가정합니다. 공유 임시 저장소를 사용할 수 없는 경우에는 `scp`(secure copy: 원격 파일 복사 프로그램) 등의 다른 도구를 사용하여 소스와 대상 시스템 간에 파일을 복사할 수 있습니다. `scp` 프로그램이 인증이 필요한 경우 암호나 암호문을 요청합니다.

1 관리자로 전환합니다.

2 마이그레이션할 영역(이 절차에서는 `my-zone`)을 종료합니다.

```
host1# zoneadm -z my-zone shutdown
```

3 (옵션) 영역을 분리합니다.

```
host1# zoneadm -z my-zone detach
```

분리된 영역이 이제 구성됨 상태에 있습니다. 전역 영역이 다음에 부트할 때 해당 영역이 자동으로 부트되지 않습니다.

4 영역 구성을 내보냅니다.

```
host1# mkdir /net/server/zonearchives/my-zone
host1# zonecfg -z my-zone export > /net/nserver/zonearchives/my-zone/my-zone.zonecfg
```

5 **gzip ZFS 아카이브를 만듭니다.**

```
host1# zfs list -H -o name /zones/my-zone
rpool/zones/my-zone
host1# zfs snapshot -r rpool/zones/my-zone@v2v
host1# zfs send -rc rpool/zones/my-zone@v2v | gzip > /net/server/zonearchives/my-zone/my-zone.zfs.gz
```

압축 사용은 선택 사항이지만 일반적으로 압축을 사용할 경우 아카이브에 대해 쓰고 읽는 동안 I/O가 더 적게 수행되므로 속도가 더 빠릅니다. 자세한 내용은 **Oracle Solaris 관리: ZFS 파일 시스템**을 참조하십시오.

6 **새 호스트에서 영역을 구성합니다.**

```
host2# zonecfg -z my-zone -f /net/server/zonearchives/my-zone/my-zone.zonecfg
```

다음 시스템 메시지가 표시됩니다.

```
my-zone: No such zone configured
Use 'create' to begin configuring a new zone.
```

7 **새 호스트에 my-zone 영역을 만들려면 새 호스트에서 zonecfg 명령을 -a 옵션 및 zonepath와 함께 사용합니다.**

```
zonecfg:my-zone> create -a /zones/my-zone
```

8 **(옵션) 구성을 확인합니다.**

```
zonecfg:my-zone> info
zonename: my-zone
zonepath: /zones/my-zone
autoboot: false
pool:
net:
        address: 192.168.0.90
        physical: bge0
```

9 **구성을 필요에 맞게 조정합니다.**

예를 들어 새 호스트에서는 네트워크 물리적 장치가 다르거나 구성에 포함된 장치가 다른 이름일 수 있습니다.

```
zonecfg:my-zone> select net physical=bge0
zonecfg:my-zone:net> set physical=e1000g0
zonecfg:my-zone:net> end
```

10 **구성을 완결하고 종료합니다.**

```
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

11 **다음 방법 중 하나를 사용하여 영역을 새 호스트에 연결합니다.**

- **소프트웨어를 업데이트하지 않고 영역을 연결합니다.**

  ```
  host2# zoneadm -z my-zone attach -a /net/server/zonearchives/my-zone/my-zone.zfs.gz
  ```

- **연결하는 데 필요한 최소 업데이트를 수행하고 영역을 연결합니다.**

```
host2# zoneadm -z my-zone attach -u -a /net/server/zonearchives/my-zone/my-zone.zfs.gz
```

- **영역의 모든 소프트웨어를 전역 영역과 호환되는 최신 버전으로 업데이트하고 영역을 연결합니다.**

```
host2# zoneadm -z my-zone attach -U -a /net/server/zonearchives/my-zone/my-zone.zfs.gz
```

▼ zonepath를 새 호스트로 옮기는 방법

zonepath의 아카이브를 생성하는 방법에는 여러 가지가 있습니다. 예를 들어 cpio(1), pax(1) 및 zfs(1M) 매뉴얼 페이지에 설명된 zfs send, cpio 또는 pax 명령을 사용할 수 있습니다.

아카이브를 새 호스트로 전송하는 방법에는 몇 가지가 있습니다. 소스 호스트에서 대상 호스트로 zonepath를 전송하는 데 사용되는 방식은 로컬 구성에 따라 다릅니다. SAN과 같이, 경우에 따라서는 zonepath 데이터를 실제로 이동하지 못할 수도 있습니다. zonepath가 새 호스트에서 보이도록 SAN을 간단히 재구성할 수 있습니다. 그 밖의 경우에는 zonepath를 테이프에 기록한 후 테이프를 새 위치로 우편 발송할 수 있습니다.

이러한 이유로 이 단계는 자동화되지 않습니다. 시스템 관리자는 zonepath를 새 호스트로 이동하기 위한 가장 적절한 기술을 선택해야 합니다.

1. **관리자로 전환합니다.**
2. **zonepath를 새 호스트로 이동합니다. 이 절차에 설명된 방법을 사용하거나 선택한 다른 방법을 사용할 수 있습니다.**

예 23–1 tar 명령을 사용한 zonepath 아카이브 및 이동과 영역 연결

1. host1에서 zonepath의 tar 파일을 만들고 sftp 명령을 사용하여 host2로 전송합니다.

```
host1# cd /zones
host1# tar cf my-zone.tar my-zone
host1# sftp host2
Connecting to host2...
Password:
sftp> cd /zones
sftp> put my-zone.tar
Uploading my-zone.tar to /zones/my-zone.tar
sftp> quit
```

2. host2에서 영역을 연결합니다.

```
host2# zoneadm -z my-zone attach -a /zones/my-zone.tar -u
```

자세한 내용은 sftp(1) 및 tar(1)을 참조하십시오.

예 23–2 cpio을 사용한 zonepath 아카이브와 gzip을 사용한 아카이브 압축

이 방법은 예 23-1에서처럼 tar 명령 사용의 대체 방법입니다.

```
host1# zoneadm -z my-zone halt
host1# find my-zone -print | cpio -oP@/ | gzip > my-zone.cpio.gz
```

다음 순서 SAN 재구성 대신에 -a 옵션을 사용한 경우에는 영역이 현재 구성됨 상태에 있더라도 zonepath 데이터가 여전히 소스 호스트에 표시됩니다. 데이터를 새 호스트로 이동한 후 소스 호스트에서 zonepath 호스트를 수동으로 제거하거나, 영역을 소스 호스트에 다시 연결하고 zoneadm uninstall 명령을 사용하여 zonepath를 제거할 수 있습니다.

사용할 수 없는 시스템에서 영역 마이그레이션

비전역 영역을 호스팅하는 시스템을 사용하지 못하게 될 수도 있습니다. 하지만 SAN과 같은 영역이 있는 저장소를 여전히 사용할 수 있는 경우에는 영역을 새 호스트로 마이그레이션할 수 있습니다. 영역의 zonepath를 새 호스트로 이동할 수 있습니다. SAN과 같이, 경우에 따라서는 zonepath 데이터를 실제로 이동하지 못할 수도 있습니다. zonepath가 새 호스트에 보이도록 SAN을 간단히 재구성할 수 있습니다. 영역이 올바르게 분리되지 않았으므로 먼저 zonecfg 명령을 사용하여 새 호스트에서 영역을 만들어야 합니다. 이 작업을 수행했으면 새 호스트에서 연결을 연결합니다.

이 작업의 절차는 306 페이지 “ZFS 아카이브를 사용하여 비전역 영역을 마이그레이션하는 방법”의 4~8단계에 설명되어 있습니다. 308 페이지 “zonepath를 새 호스트로 옮기는 방법”도 참조하십시오.

Oracle Solaris 시스템을 비전역 영역으로 마이그레이션

영역은 중첩되지 않으므로 P2V 프로세스를 수행하면 마이그레이션된 시스템 이미지 내의 기존 영역을 대상 영역에서 사용할 수 없게 됩니다. 전역 영역의 시스템 이미지를 마이그레이션하려면 소스 시스템에 있는 기존의 비전역 영역을 마이그레이션해야 합니다.

Oracle Solaris 11 시스템을 solaris 비전역 영역으로 마이그레이션

기존 Oracle Solaris 11 시스템을 Oracle Solaris 11 시스템의 solaris 브랜드 영역으로 직접 마이그레이션할 수 있습니다. 소스 시스템에서 zonep2vchk 및 zfs 명령을 사용하여 마이그레이션을 준비하고 시스템 이미지를 아카이브합니다. zonecfg 및 zoneadm 명령을 사용하여 대상 시스템의 대상 영역에 아카이브를 구성하고 설치합니다.

전역 영역을 비전역 영역으로 마이그레이션할 때 다음과 같은 제한이 적용됩니다.

- 대상 시스템의 전역 영역에서 원래 소스 호스트와 동일한 Oracle Solaris 11 릴리스가 실행되고 있어야 합니다.
- 영역이 제대로 실행되도록 하려면 대상 시스템에 필수 운영 체제 패키지와 동일하거나 그보다 최신인 버전이 있어야 합니다. 타사 제품용 패키지 등의 기타 패키지는 다를 수 있습니다.

자세한 내용은 zonep2vchk(1M), zfs(1M), zonecfg(1M) 및 zoneadm(1M) 및 solaris(5) 매뉴얼 페이지를 참조하십시오.

▼ zonep2vchk를 사용하여 소스 시스템 검색

1 **관리자로 로그인합니다.**

2 **-b 옵션으로 zonep2vchk 도구를 실행하여 P2V 마이그레이션으로 인해 영향을 받을 수 있는, 사용 중인 Oracle Solaris 기능이 있는지 확인하는 기본 분석을 수행합니다.**

```
source# zonep2vchk -b 11
```

3 **-s 옵션으로 zonep2vchk 도구를 실행하여 응용 프로그램 파일의 정적 분석을 수행합니다. 이 분석은 영역 내의 작업에 영향을 줄 수 있는 시스템 및 라이브러리 호출에 대해 ELF 이진을 검사합니다.**

```
source# zonep2vchk -s /opt/myapp/bin,/opt/myapp/lib
```

4 **-r 옵션으로 zonep2vchk 도구를 실행하여 영역 내부에서 성공적으로 실행할 수 없는 프로세스를 찾는 런타임 검사를 수행합니다.**

```
source# zonep2vchk -r 2h
```

5 **소스 시스템에서 -c 옵션으로 zonep2vchk 도구를 실행하여 이 절차에서 s11-zone.config라고 하는 템플리트 zonecfg 스크립트를 생성합니다.**

```
source# zonep2vchk -c > /net/somehost/p2v/s11-zone.config
```

이 구성에는 소스 호스트의 물리적 리소스 및 네트워킹 구성을 기반으로 한 리소스 제한과 네트워크 구성이 포함됩니다.

▼ 네트워크 장치에서 시스템 이미지의 아카이브를 만드는 방법

전역 영역에서 파일 시스템을 아카이브합니다. 비전역 영역이 소스 시스템에 설치되었는지 확인합니다. cpio, -x xustar(XUSTAR) 형식으로 만든 pax 아카이브 및 zfs를 비롯한 여러 아카이브 형식이 지원됩니다. 이 절의 예에서는 zfs send 명령을 사용하여 아카이브를 만듭니다. 이 예에서는 루트 풀 이름이 rpool이라고 가정합니다.

1 **관리자로 로그인합니다.**

2. **이 절차에서 *rpool@p2v*라는 전체 루트 풀의 스냅샷을 만듭니다.**

   ```
   source# zfs snapshot -r rpool@p2v
   ```

3. **스왑 및 덤프 장치와 연관된 스냅샷은 대상 시스템에서 필요하지 않으므로 이러한 스냅샷을 완전 삭제합니다.**

   ```
   source# zfs destroy rpool/swap@p2v

   source# zfs destroy rpool/dump@p2v
   ```

4. **시스템을 아카이브합니다.**

   - **gzip으로 압축되고 원격 NFS 서버에 저장된 ZFS 복제 스트림 아카이브를 생성합니다.**

     ```
     source# zfs send -R rpool@p2v | gzip > /net/somehost/p2v/s11-zfs.gz
     ```

   - **다음 대체 명령을 사용하면 중간 스냅샷이 저장되지 않으므로 아카이브 크기를 줄일 수 있습니다.**

     ```
     source# zfs send -rc rpool@p2v
     ```

참조 자세한 내용은 cpio(1), pax(1) 및 zfs(1M) 매뉴얼 페이지를 참조하십시오.

▼ 대상 시스템에서 영역을 구성하는 방법

zonep2vchk 도구로 생성된 템플리트 zonecfg 스크립트는 대상 영역 구성에서 지원해야 하는 소스 시스템 구성의 특성을 정의합니다. 추가 대상 시스템 종속 정보는 영역을 완전히 정의할 수 있게 수동으로 제공해야 합니다.

이 절차에서 구성 파일 이름은 s11-zone.config입니다.

1. **관리자로 로그인합니다.**

2. **zonecfg 스크립트의 내용을 검토하여 소스 시스템의 구성 매개 변수를 숙지하십시오.**

   ```
   target# less /net/somehost/p2v/s11-zone.config
   ```

   이 스크립트의 zonepath 초기 값은 소스 시스템의 호스트 이름을 기반으로 합니다. 대상 영역의 이름이 소스 시스템의 호스트 이름과 다른 경우 zonepath 디렉토리를 변경할 수 있습니다.

   주석으로 처리된 명령은 메모리 용량, CPU 수 및 네트워크 카드 MAC 주소를 포함하여 원래 물리적 시스템 환경의 매개 변수에 적용됩니다. 대상 영역에서 리소스 추가 제어를 위해 이러한 행의 주석을 해제할 수 있습니다.

3. **대상 시스템의 전역 영역에서 다음 명령을 사용하여 현재 링크 구성을 확인합니다.**

   ```
   target# dladm show-link
   target# dladm show-physical
   target# ipadm show-addr
   ```

기본적으로 zonecfg 스크립트는 소스 시스템에 구성된 모든 물리적 네트워크 인터페이스에 대한 anet 리소스로 배타적 IP 네트워크 구성을 정의합니다. 대상 시스템은 영역이 부트되면 각 anet 리소스에 대한 VNIC를 자동으로 만듭니다. VNIC를 사용하면 여러 영역에서 같은 물리적 네트워크 인터페이스를 공유할 수 있습니다. anet 리소스의 lower-link 이름은 처음에 zonecfg 명령에 의해 *change-me*로 설정됩니다. 이 필드를 대상 시스템에서 데이터 링크 중 하나의 이름으로 직접 설정해야 합니다. VNIC의 lower-link에 유효한 링크를 지정할 수 있습니다.

4 zonecfg 스크립트를 대상 시스템에 복사합니다.

```
target# cp /net/somehost/p2v/s11-zone.config .
```

5 vi와 같은 텍스트 편집기를 사용하여 구성 파일을 변경합니다.

```
target# vi s11-zone.config
```

6 zonecfg 명령을 사용하여 *s11-zone* 영역을 구성합니다.

```
target# zonecfg -z s11-zone -f s11-zone.config
```

▼ 대상 시스템에서 영역 설치

이 예에서는 설치 중에 원래 시스템 구성을 변경하지 않습니다.

1 관리자로 로그인합니다.

2 소스 시스템에 만들어진 아카이브를 사용하여 영역을 설치합니다.

```
target# zoneadm -z s11-zone install -a /net/somehost/p2v/s11-zfs.gz -p
```

◆ ◆ ◆ 24장

24 영역이 설치된 Oracle Solaris 11 시스템의 자동 설치 및 패키지 정보

AI 클라이언트 설치의 일부로서 비전역 영역의 설치 및 구성을 지정할 수 있습니다. IPS(이미지 패키징 시스템)는 Oracle Solaris 11 릴리스에 적용됩니다. 이 장에서는 영역이 설치된 경우 IPS 패키징을 사용한 운영 체제 설치 및 유지 관리에 대해 설명합니다.

`solaris10` 및 `native` 영역에 사용된 SVR4 패키징 및 패치에 대한 자세한 내용은 **System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones**의 "25장, About Packages on an Solaris System With Zones Installed (Overview)"와 "26장, Adding and Removing Packages and Patches on a Solaris System With Zones Installed (Tasks)"를 참조하십시오. 이 설명서는 Oracle Solaris 10 버전의 설명서입니다.

Oracle Solaris 11 릴리스를 실행하는 시스템의 이미지 패키징 시스템 소프트웨어

그래픽 및 명령줄 도구를 사용하면 저장소에서 패키지를 다운로드하고 설치할 수 있습니다. 이 장에서는 설치된 비전역 영역에 패키지 추가에 대한 정보를 제공합니다. 패키지 제거에 대한 정보도 수록되어 있습니다. 이 장의 내용은 기존의 Oracle Solaris 설치 및 패키징 설명서를 보충합니다. 자세한 내용은 **Oracle Solaris 관리: 일반 작업**을 참조하십시오.

영역 패키징 개요

`solaris` 패키징 저장소는 영역 환경에서 사용됩니다.

영역은 `pkg` 명령을 사용하여 시스템을 새로운 버전의 Oracle Solaris로 업그레이드할 때 자동으로 업데이트됩니다.

pkg(5)에 설명된 IPS(이미지 패키징 시스템)는 패키지 설치, 업그레이드 및 제거 등 소프트웨어 수명 주기 관리용으로 제공되는 프레임워크입니다. IPS는 소프트웨어 패키지를 생성하고, 패키징 저장소를 생성 및 관리하며, 기존 패키징 저장소를 미러링하는 데 사용할 수 있습니다.

Oracle Solaris 운영 체제의 초기 설치 후 이미지 패키징 시스템 CLI 및 GUI(패키지 관리자) 클라이언트를 통해 패키징 저장소에서 추가 소프트웨어 응용 프로그램을 설치할 수 있습니다.

시스템에 패키지를 설치한 후에는 IPS 클라이언트를 사용하여 패키지를 검색, 업그레이드 및 관리할 수 있습니다. IPS 클라이언트를 사용하여 전체 시스템을 Oracle Solaris의 최신 릴리스로 업그레이드하고 저장소를 생성 및 관리하며 기존 저장소를 미러링할 수도 있습니다.

IPS가 설치된 시스템에서 인터넷에 액세스할 수 있는 경우 클라이언트는 Oracle Solaris 11 패키지 저장소(기본 solaris 게시자), http://pkg.oracle.com/solaris/release/에서 소프트웨어에 액세스하고 설치할 수 있습니다.

영역 관리자는 이 설명서에 명시된 제한 내에서 패키징 도구를 사용하여 비전역 영역에 설치된 모든 소프트웨어를 관리할 수 있습니다.

영역이 설치된 경우 다음과 같은 일반 원칙이 적용됩니다.

- 패키지가 전역 영역에 설치된 경우 비전역 영역은 전역 영역의 시스템 저장소 서비스에서 패키지를 설치할 수 있으며 네트워크를 사용하여 해당 패키지를 설치할 필요가 없습니다. 패키지가 전역 영역에 설치되지 않은 경우 전역 영역을 사용하여 네트워크에서 패키지를 설치하려면 영역이 영역 프록시 서비스를 사용하여 게시자에 액세스해야 합니다.
- 전역 관리자나 적절한 권한을 가진 사용자는 시스템의 모든 영역에 있는 소프트웨어를 관리할 수 있습니다.
- 비전역 영역의 루트 파일 시스템은 Oracle Solaris 패키징 도구를 사용하여 전역 영역에서 관리할 수 있습니다. Oracle Solaris 패키징 도구는 비전역 영역 내에서 동시 패키징(번들), 독립형(비번들) 또는 타사 제품을 관리하는 용도로 지원됩니다.
- 패키징 도구는 영역 실행형 환경에서 작동합니다. 이 도구를 사용하면 패키지를 비전역 영역에도 설치할 수 있습니다.

주 – 특정 패키지 작업을 수행하는 동안 이 유형의 다른 작업에 대해 영역이 임시로 잠깁니다. 또한 시스템에서 작업을 진행하기 전에 요청된 작업을 관리자에게 확인할 수도 있습니다.

패키지 및 영역 정보

brands(5)에 설명된 바와 같이 solaris 브랜드 영역에 설치된 소프트웨어는 전역 영역에 설치된 소프트웨어와 호환되어야 합니다. pkg 명령은 호환성을 자동으로 강제 적용합니다. pkg update 명령을 전역 영역에서 실행하여 소프트웨어를 업데이트하는 경우 영역과 전역 영역의 동기화 상태를 유지하기 위해 영역도 업데이트됩니다. 비전역 영역과 전역 영역에 서로 다른 소프트웨어가 설치될 수 있습니다. pkg 명령은 영역에서 해당 영역 내 소프트웨어를 관리하는 데 사용될 수도 있습니다.

전역 영역에서 pkg update 명령(지정된 FMRI 없음)을 실행하는 경우 pkg가 시스템의 전역 영역 및 비전역 영역 모두에서 모든 소프트웨어를 업데이트합니다.

Oracle Solaris Zones에 pkg install 의 설치 기능을 테스트 실행(드라이 실행이라고도 함)할 수 있습니다.

영역 패키지 변경을 사용하면 패키지 내 다양한 구성 요소가 전역 영역(global)이나 비전역 영역(nonglobal)에만 설치되도록 특별히 태그됩니다. 제공된 패키지에 비전역 영역에 설치되지 않도록 태그된 파일이 들어 있을 수 있습니다.

비전역 영역이 설치된 경우 전역 영역에 설치된 Oracle Solaris 패키지의 일부만 완벽하게 복제됩니다. 예를 들어 Oracle Solaris 커널을 포함하는 패키지 중 상당수는 비전역 영역에서 필요하지 않습니다. 모든 비전역 영역은 암시적으로 전역 영역과 동일한 커널을 공유합니다.

자세한 내용은 **Oracle Solaris 11 시스템**와 **Oracle Solaris 11 소프트웨어 패키지 추가 및 업데이트**를 참조하십시오.

주 – 시스템의 전역 영역을 비전역 영역으로 업데이트할 때 영역에 대한 패키지 다운로드 정보가 두 번 표시되는 것처럼 보일 수 있습니다 그러나 패키지는 한 번만 다운로드됩니다.

영역을 설치한 시스템에서 https_proxy 및 http_proxy 사용

이 절에서는 IPS 게시자 저장소에 직접 연결되지 않은 내부 서브넷의 시스템에 대한 프록시를 설정하는 방법을 설명합니다. 시스템은 http_proxy 구성을 사용하여 연결합니다.

예를 들어, solaris 비전역 영역을 실행 중인 시스템의 소프트웨어가 프록시 구성 https_proxy=http:// *129.156.243.243:3128*을 사용하여 IPS로 관리되고 있다고 가정합니다. 129.156.243.243:3128 https 프록시는 IP 주소 *129.156.243.243*을 사용하여 시스템의 포트 3128에 구성됩니다. 분석 가능한 호스트 이름을 사용할 수도 있습니다.

1. 이 예에서 다음 행을 입력하여 전역 영역에 대해 ksh에 프록시를 설정합니다. 그러면 pkg 명령이 https 프록시를 통해 게시자와 통신할 수 있습니다.

```
# export https_proxy=http://129.156.243.243:3128
```

2. 시스템의 solaris 영역이 구성된 시스템 게시자를 사용하도록 허용하려면 다음 명령을 실행합니다.

```
# svccfg -s svc:/application/pkg/system-repository:default setprop config/http_proxy=astring: "https://129.156.243.243
```

3. 실시간 SMF 저장소에 변경 사항을 적용하려면 다음을 실행합니다.

```
# svcadm refresh svc:/application/pkg/system-repository:default setprop
```

4. 설정이 작동하는지 확인하려면 다음을 실행합니다.

```
# svcprop svc:/application/pkg/system-repository:default|grep https_proxy
```

이 프로세스로 solaris 영역이 전역 영역에 설정된 게시자에도 연결할 수 있습니다. solaris 브랜드 영역으로 pkg 작업을 반복해도 성공합니다.

영역이 패키지 작업에 미치는 영향

다음 표는 다양한 상태의 비전역 영역과 함께 패키징 명령이 시스템에서 사용될 때 어떻게 되는지를 설명합니다.

영역 상태	패키지 작업에 대한 영향
구성됨	패키지 도구를 실행할 수 있습니다. 소프트웨어가 아직 설치되지 않았습니다.
설치됨	패키지 도구를 실행할 수 있습니다. zoneadm -z *zonename* install을 완료하자 마자 영역이 설치됨 상태로 전환됩니다.
준비	패키지 도구를 실행할 수 있습니다.
실행 중	패키지 도구를 실행할 수 있습니다.
불완전	zoneadm에 의해 설치되거나 제거되는 영역입니다. 패키지 도구를 사용할 수 없습니다. 영역이 도구를 사용하기에 적당한 상태가 아닙니다.

영역이 설치된 시스템에서 패키지 추가 정보

Oracle Solaris 11 릴리스에서는 pkg install 명령을 사용합니다.

```
# pkg install package_name
```

전역 영역에서 pkg 사용

전역 영역에서 pkg install 명령은 패키지를 전역 영역에만 추가하는 데 사용됩니다. 이 패키지는 다른 영역으로 전파되지 않습니다.

비전역 영역에서 pkg install 명령 사용

pkg install 명령은 비전역 영역의 영역 관리자가 패키지를 비전역 영역에만 추가하는 데 사용됩니다. 지정된 비전역 영역에 패키지를 추가하려면 pkg install 명령을 영역 관리자로 실행합니다.

패키지 종속성은 IPS에서 자동으로 처리됩니다.

사용자 정의 AI 매니페스트를 사용하여 영역에 추가 패키지 추가

설치 시 추가 소프트웨어 추가 프로세스는 AI 매니페스트를 수정하여 자동화할 수 있습니다. 지정한 패키지 및 종속 패키지가 설치됩니다. 기본 패키지 목록은 AI 매니페스트에서 가져옵니다. 기본 AI 매니페스트는 /usr/share/auto_install/manifest/zone_default.xml입니다. 패키지 찾기 및 작업에 대한 자세한 내용은 **Oracle Solaris 11 소프트웨어 패키지 추가 및 업데이트**를 참조하십시오.

예 24–1 매니페스트 수정

다음 절차는 mercurial 및 vim 편집기의 전체 설치를 *my-zone* 이름의 구성된 영역에 추가합니다. (solaris-small-server의 일부인 최소 vim-core만 기본적으로 설치됩니다.)

1. 파일을 편집할 위치에 기본 AI 매니페스트를 복사하고 파일을 쓰기 가능으로 설정합니다.

```
# cp /usr/share/auto_install/manifest/zone_default.xml ~/my-zone-ai.xml
# chmod 644 ~/my-zone-ai.xml
```

2. 다음과 같이 mercurial 및 vim 패키지를 software_data 섹션에 추가하여 파일을 편집합니다.

```
      <software_data action="install">
              <name>pkg:/group/system/solaris-small-server</name>
              <name>pkg:/developer/versioning/mercurial</name>
              <name>pkg:/editor/vim</name>
          </software_data>
```

3. 영역을 설치합니다.

```
# zoneadm -z my-zone install -m ~/my-zone-ai.xml
```

다음 화면이 표시됩니다.

예 24–1 매니페스트 수정 *(계속)*

```
A ZFS file system has been created for this zone.
Progress being logged to /var/log/zones/zoneadm.20111113T004303Z.my-zone.install
       Image: Preparing at /zones/my-zone/root.

 Install Log: /system/volatile/install.15496/install_log
 AI Manifest: /tmp/manifest.xml.XfaWpE
  SC Profile: /usr/share/auto_install/sc_profiles/enable_sci.xml
    Zonename: my-zone
Installation: Starting ...

              Creating IPS image
              Installing packages from:
                  solaris
                      origin:  http://localhost:1008/solaris/54453f3545de891d4daa841ddb3c844fe8804f55/

DOWNLOAD                                  PKGS       FILES    XFER (MB)
Completed                              169/169 34047/34047  185.6/185.6

PHASE                                        ACTIONS
Install Phase                            46498/46498

PHASE                                          ITEMS
Package State Update Phase                   169/169
Image State Update Phase                         2/2
Installation: Succeeded
...
```

영역에서 패키지 제거 정보

Oracle Solaris 11 릴리스에서는 `pkg uninstall` 명령을 사용하여 영역이 설치된 시스템에서 패키지를 제거합니다.

```
# pkg uninstall package_name
```

패키지 정보 쿼리

Oracle Solaris 11 릴리스에서는 `pkg info` 명령을 사용하여 영역이 설치된 시스템에서 소프트웨어 패키지 데이터베이스를 쿼리합니다.

전역 영역에서는 이 명령을 사용하여 전역 영역에서만 소프트웨어 패키지 데이터베이스를 쿼리할 수 있습니다. 비전역 영역에서는 이 명령을 사용하여 비전역 영역에서만 소프트웨어 패키지 데이터베이스를 쿼리할 수 있습니다.

◆ ◆ ◆ 25장

25 Oracle Solaris 영역 관리(개요)

이 장에서는 다음 일반적인 영역의 내용을 다룹니다.

`solaris10` 브랜드 영역에 대한 자세한 내용은 제3부를 참조하십시오.

전역 영역 표시 및 액세스

전역 영역은 시스템의 기본 영역 역할을 하며 또한 시스템 전체의 관리 제어 영역 역할을 합니다. 이 이중 역할과 연관된 관리 문제가 있습니다. 영역 내의 응용 프로그램에서 다른 영역에 속한 프로세스 및 다른 시스템 객체에 액세스할 수 있으므로 관리 작업의 효과가 예상보다 높습니다. 예를 들어 서비스 종료 스크립트는 `pkill`을 사용하여 지정된 이름을 가진 프로세스를 종료하도록 신호를 보냅니다. 그런 스크립트를 전역 영역에서 실행하면 영역에 상관없이 시스템 내의 모든 그런 프로세스에 신호가 전송됩니다.

시스템 전체 범위가 필요한 경우가 자주 있습니다. 예를 들어 시스템 전체 리소스 사용을 모니터링하려면 전체 시스템에 대한 프로세스 통계를 확인해야 합니다. 전역 영역 작업 보기에는 시스템 리소스의 일부 또는 전부를 공유할 수 있는 시스템 내 다른 영역의 관련 정보가 없습니다. 이러한 보기는 리소스 관리 기능을 사용하여 시스템 리소스(예: CPU)을 엄격하게 분할하지 않는 경우에 특히 중요합니다.

따라서, 전역 영역의 프로세스에서 비전역 영역의 프로세스와 다른 객체를 관찰할 수 있습니다. 그러면 그런 프로세스에서 시스템을 전체적으로 확인할 수 있습니다. 다른 영역의 프로세스에 신호를 보내거나 제어하는 기능은 `PRIV_PROC_ZONE` 권한에 의해 제한됩니다. 이 권한을 사용하면 프로세스에서 권한이 없는 프로세스에 적용된 제한을 대체할 수 있으므로 이 권한은 `PRIV_PROC_OWNER`와 비슷합니다. 이 경우, 전역 영역의 권한이 없는 프로세스에서 다른 영역의 프로세스에 신호를 보내거나 제어할 수 없도록 제한합니다. 이는 프로세스의 사용자 ID가 일치하거나 작동 프로세스에 `PRIV_PROC_OWNER` 권한이 있는 경우에도 마찬가지입니다. 별도로 권한이 부여된 프로세스에서 `PRIV_PROC_ZONE` 권한을 제거하여 작업을 전역 영역으로 제한할 수 있습니다.

`zoneidlist`를 사용하여 프로세스를 일치시키는 방법에 대한 자세한 내용은 `pgrep(1)` `pkill(1)` 매뉴얼 페이지를 참조하십시오.

영역의 프로세스 ID 표시 여부

동일한 영역에 있는 프로세스만 프로세스 ID를 사용하는 시스템 호출 인터페이스(예: `kill` 및 `priocntl` 명령)를 통해 표시할 수 있습니다. 자세한 내용은 `kill(1)` 및 `priocntl(1)` 매뉴얼 페이지를 참조하십시오.

영역 내의 시스템 관찰성

`ps` 명령은 다음과 같은 수정 사항이 있습니다.

- `-o` 옵션은 출력 형식을 지정하는 데 사용됩니다. 이 옵션을 사용하면 프로세서의 영역 ID 또는 프로세스가 실행 중인 영역의 이름을 인쇄할 수 있습니다.
- `-z` *zonelist* 옵션은 지정된 영역의 프로세스만 나열하는 데 사용됩니다. 영역 이름 또는 영역 ID를 사용하여 영역을 지정할 수 있습니다. 이 옵션은 전역 영역에서 명령을 실행하는 경우에만 유용합니다.

- `-Z` 옵션은 프로세스와 연결된 영역의 이름을 인쇄하는 데 사용됩니다. 이름은 열 제목 `ZONE` 아래에 인쇄됩니다.

자세한 내용은 `ps(1)` 매뉴얼 페이지를 참조하십시오.

`-z` *zonename* 옵션이 다음 Oracle Solaris 유틸리티에 추가되었습니다. 이 옵션을 사용하여 지정된 영역만 포함하도록 정보를 필터링할 수 있습니다.

- `ipcs`(`ipcs(1)` 매뉴얼 페이지 참조)
- `pgrep`(`pgrep(1)` 매뉴얼 페이지 참조)
- `ptree`(`proc(1)` 매뉴얼 페이지 참조)
- `prstat`(`prstat(1M)` 매뉴얼 페이지 참조)

명령에 대한 전체 변경 목록은 표 25–5를 참조하십시오.

zonestat 유틸리티를 사용하여 활성 영역 통계 보고

`zonestat` 유틸리티를 사용하려면 `zonestat(1)` 매뉴얼 페이지 및 353 페이지 “비전역 영역에서 zonestat 유틸리티 사용”을 참조하십시오.

`zonestat` 유틸리티는 현재 실행 중인 영역의 CPU, 메모리 및 리소스 제어 사용률에 대해 보고합니다. 배타적 IP 영역에 대해서는 네트워킹 대역폭 사용률도 보고됩니다. 각 영역의 사용률은 영역에 대해 구성된 제한 및 시스템 리소스의 백분율로 표시됩니다. `zonestat` 유틸리티는 일련의 보고서를 지정된 간격으로 인쇄합니다. 선택적으로 이 유틸리티는 하나 이상의 요약 보고서를 인쇄할 수 있습니다.

비전역 영역에서 실행하면 해당 영역에 표시되는 프로세서 세트만 보고됩니다. 비전역 영역 출력에는 모든 메모리 리소스와 제한 리소스가 포함됩니다.

온라인 상태에서만 전역 영역의 `zonestat` 서비스에서 비전역 영역의 `zonestat` 서비스를 사용할 수 있습니다. 각 비전역 영역의 `zonestat` 서비스에서는 전역 영역의 `zonestat` 서비스에서 시스템 구성 및 사용률 데이터를 읽습니다.

`zonestatd` 시스템 데몬은 시스템 부트 중에 시작됩니다. 이 데몬은 영역별 시스템 리소스 사용률과 영역 및 시스템 구성 정보(예: `psrset` 프로세서 세트, 풀 프로세서 세트 및 리소스 제어 설정)를 모니터링합니다. 구성 가능한 구성 요소가 없습니다.

비전역 영역 노드 이름

노드 이름은 영역 관리자가 설정할 수 있습니다. 노드 이름은 고유해야 합니다(예: 영역 이름).

```
# svccfg -s svc:/system/identity:node setprop config/nodename = astring:"hostname"
```

영역에서 NFS 서버 실행

영역에서 NFS 공유를 만들려면 NFS 서버 패키지 `svc:/network/nfs/server:default`를 영역에 설치해야 합니다. 영역을 만드는 중에는 NFS 서버 패키지를 설치할 수 없습니다.

영역 구성에서 `sys_share` 권한을 금지하여 영역에서 NFS 공유를 금지할 수 있습니다. 표 25–1을 참조하십시오.

제한 사항은 다음과 같습니다.

- 교차 영역 LOFS 마운트를 영역에서 공유할 수 없습니다.
- 영역에 마운트된 파일 시스템을 전역 영역에서 공유할 수 없습니다.
- NFS over RDMA(Remote Direct Memory Access)는 영역에서 지원되지 않습니다.
- Oracle Sun Cluster HANFS(HA for NFS) 페일오버는 영역에서 지원되지 않습니다.

Oracle Solaris 관리: 네트워크 서비스를 참조하십시오.

파일 시스템 및 비전역 영역

이 절에서는 영역이 설치된 Oracle Solaris 시스템의 파일 시스템 문제에 대해 설명합니다. 각 영역에는 영역 `root`라고 하는 디렉토리에 루트 지정된 자체 파일 시스템 계층 섹션이 있습니다. 영역의 프로세스에서는 영역 루트 아래에 있는 계층에 포함된 파일에만 액세스할 수 있습니다. `chroot` 유틸리티는 영역에서 프로세스를 영역 내의 루트 경로로 제한하는 데에만 사용할 수 있습니다. `chroot`에 대한 자세한 내용은 `chroot(1M)`을 참조하십시오.

-o nosuid 옵션

`mount` 유틸리티에 대한 `-o nosuid` 옵션은 다음과 같은 기능이 있습니다.

- `nosetuid` 옵션을 사용하여 마운트되는 파일 시스템에 있는 `setuid` 이진의 프로세스는 `setuid` 이진 권한으로 실행되지 않습니다. 프로세스는 이진을 실행하는 사용자의 권한으로 실행됩니다.

  예를 들어 사용자가 `root`에서 소유한 `setuid` 이진을 실행하는 경우 프로세스는 사용자 권한으로 실행됩니다.
- 파일 시스템에서 장치 특정 항목을 열 수 없습니다. 이 동작은 `nodevices` 옵션을 지정하는 것과 같습니다.

`mount(1M)` 매뉴얼 페이지에서 설명한 대로 이 파일 시스템 특정 옵션은 `mount` 유틸리티를 사용하여 마운트할 수 있는 모든 Oracle Solaris 파일 시스템에서 사용할 수 있습니다. 이 설명서에서 이 파일 시스템은 323 페이지 “영역에 파일 시스템 마운트”에 나열되어 있습니다. 또한 마운트 기능이 설명되어 있습니다. `-o nosuid` 옵션에 대한 자세한 내용은 **Oracle Solaris 관리: 네트워크 서비스**의 “네트워크 파일 시스템 액세스(참조)”를 참조하십시오.

영역에 파일 시스템 마운트

파일 시스템을 영역에서 마운트할 때 nodevices 옵션이 적용됩니다. 예를 들어 영역에 UFS 파일 시스템에 해당하는 블록 장치(/dev/dsk/c0t0d0s7) 및 원시 장치(/dev/rdsk/c0t0d0s7)에 대한 액세스 권한이 부여되는 경우 파일 시스템은 영역에서 마운트할 때 nodevices에 자동으로 마운트됩니다. 이 규칙은 zonecfg 구성을 통해 지정되는 마운트에는 적용되지 않습니다.

비전역 영역에서 파일 시스템을 마운트하는 데 사용되는 옵션에 대해서는 다음 표에 설명되어 있습니다. 이러한 대체 마운트 절차는 242 페이지 “영역 구성, 확인 및 커밋” 및 358 페이지 “실행 중인 비전역 영역에서 파일 시스템 마운트”를 참조하십시오.

표에 나열되지 않은 파일 시스템 유형은 /usr/lib/*fstype*/mount 에 마운트 이진이 있는 경우 구성에서 지정할 수 있습니다.

영역 관리자가 기본값이 아닌 다른 파일 시스템 마운트를 허용하면 파일 시스템이 손상될 수 있습니다.

파일 시스템	비전역 영역의 마운트 옵션
AutoFS	zonecfg를 사용하여 마운트할 수 없습니다. 영역 내에서 마운트할 수 있습니다.
CacheFS	비전역 영역에서 사용할 수 없습니다.
FDFS	zonecfg를 사용하여 마운트할 수 있으며 영역 내에서 마운트할 수 있습니다.
HSFS	zonecfg를 사용하여 마운트할 수 있으며 영역 내에서 마운트할 수 있습니다.
LOFS	zonecfg를 사용하여 마운트할 수 있으며 영역 내에서 마운트할 수 있습니다.
MNTFS	zonecfg를 사용하여 마운트할 수 없습니다. 영역 내에서 마운트할 수 있습니다.
NFS	zonecfg를 사용하여 마운트할 수 없습니다. 영역에서 현재 지원되는 버전인 V2, V3 및 V4는 영역 내에서 마운트할 수 있습니다.
PCFS	zonecfg를 사용하여 마운트할 수 있으며 영역 내에서 마운트할 수 있습니다.
PROCFS	zonecfg를 사용하여 마운트할 수 없습니다. 영역 내에서 마운트할 수 있습니다.
TMPFS	zonecfg를 사용하여 마운트할 수 있으며 영역 내에서 마운트할 수 있습니다.

파일 시스템	비전역 영역의 마운트 옵션
UDFS	zonecfg를 사용하여 마운트할 수 있으며 영역 내에서 마운트할 수 있습니다.

UFS	zonecfg를 사용하여 마운트할 수 있으며 영역 내에서 마운트할 수 있습니다.
	
	주 – quota(1M)에 설명된 quota 명령은 zonecfg add fs 리소스를 통해 추가되는 UFS 파일 시스템에 대한 쿼터 정보를 검색하는 데 사용할 수 없습니다.
	
	add fs를 사용할 경우 system/file-system/ufs 패키지를 전역 영역에 설치해야 합니다. 비전역 영역에서 zonecfg 명령을 통해 UFS 파일 시스템을 사용하려면 설치 이후에 또는 AI 매니페스트 스크립트를 통해 영역에 패키지를 설치해야 합니다.
	
	다음을 한 행에 입력합니다.
	
	`global# pkg -R /tank/zones/my-zone/root \`
	`install system/file-system/ufs`

ZFS	zonecfg dataset 및 fs 리소스 유형을 사용하여 마운트할 수 있습니다.

자세한 내용은 243 페이지 “영역 구성 방법”, 358 페이지 “실행 중인 비전역 영역에서 파일 시스템 마운트” 및 mount(1M) 매뉴얼 페이지를 참조하십시오.

영역에서 파일 시스템 마운트 해제

파일 시스템을 마운트 해제하는 기능은 초기 마운트를 수행한 사람에 따라 다릅니다. zonecfg 명령을 사용하여 파일 시스템을 영역 구성의 일부로 지정한 경우 전역 영역에서 이 마운트를 소유하고 비전역 영역 관리자는 파일 시스템을 마운트 해제할 수 없습니다. 예를 들어 영역의 /etc/vfstab 파일에서 마운트를 지정하여 비전역 영역에서 파일 시스템을 마운트한 경우 비전역 영역 관리자는 파일 시스템을 마운트 해제할 수 없습니다.

보안 제한 및 파일 시스템 동작

영역에서 특정 파일 시스템을 마운트하는 경우 보안 제한이 있습니다. 영역에서 다른 파일 시스템이 마운트될 경우 특정 동작을 나타냅니다. 마운트된 파일 시스템의 목록은 다음과 같습니다.

AutoFS
: Autofs는 적절한 파일 시스템을 자동으로 마운트하는 클라이언트측 서비스입니다. 클라이언트가 현재 마운트되지 않은 파일 시스템에 액세스하려고 하면 AutoFS 파일 시스템이 요청을 인터셉트하고 `automountd`를 호출하여 요청된 디렉토리를 마운트합니다. 영역 내에서 설정된 AutoFS 마운트는 해당 영역에 로컬입니다. 전역 영역을 포함하여 다른 영역에서 마운트를 액세스할 수 없습니다. 영역이 정지되거나 재부트되면 마운트는 제거됩니다. AutoFS에 대한 자세한 내용은 **Oracle Solaris 관리: 네트워크 서비스**의 "autofs의 작동 방식"을 참조하십시오.

  각 영역에서는 `automountd`의 자체 복사본을 실행합니다. 자동 맵 및 시간 초과는 영역 관리자에 의해 제어됩니다. 전역 영역에서 비전역 영역에 대한 AutoFS 마운트 지점을 교차하여 다른 영역에서 마운트를 트리거할 수 없습니다.

  다른 마운트가 트리거되면 특정 AutoFS 마운트가 커널에 만들어집니다. 그런 마운트는 그룹으로 마운트되거나 마운트 해제해야 하므로 일반 `umount` 인터페이스를 사용하여 제거할 수 없습니다. 이 기능은 영역 종료를 위해 제공됩니다.

MNTFS
: MNTFS는 로컬 시스템의 마운트된 파일 시스템 테이블에 대한 읽기 전용 액세스를 제공하는 가상 파일 시스템입니다. 비전역 영역에서 `mnttab`를 사용하여 표시하는 파일 시스템 집합은 영역에 마운트된 파일 시스템 집합과 루트(`/`)에 대한 항목입니다. 영역에서 액세스할 수 없는 특수 장치(예: `/dev/rdsk/c0t0d0s0`)가 있는 마운트 지점에는 마운트 지점과 동일한 특수 장치 세트가 있습니다. 시스템의 모든 마운트는 전역 영역의 `/etc/mnttab` 테이블에서 확인할 수 있습니다. MNTFS에 대한 자세한 내용은 **Oracle Solaris 관리: 장치 및 파일 시스템**의 "Oracle Solaris 파일 시스템 마운트 및 마운트 해제"를 참조하십시오.

NFS
: 영역 내에서 설정된 NFS 마운트는 해당 영역에 로컬입니다. 전역 영역을 포함하여 다른 영역에서 마운트를 액세스할 수 없습니다. 영역이 정지되거나 재부트되면 마운트는 제거됩니다.

  영역 내에서 NFS 마운트는 `nodevices` 옵션을 사용하여 마운트된 것처럼 동작합니다.

  `nfsstat` 명령 출력은 명령이 실행되는 영역하고만 관련이 있습니다. 예를 들어 명령이 전역 영역에서 실행되는 경우 전역 영역에 대한 정보만 보고됩니다. `nfsstat` 명령에 대한 자세한 내용은 `nfsstat(1M)`을 참조하십시오.

PROCFS
: `/proc` 파일 시스템(PROCFS)은 프로세스 표시 여부 및 액세스 제한을 제공하고 프로세스의 영역 연관에 대한 정보를 제공합니다. 동일한 영역에 있는 프로세스만 `/proc`를 통해 표시할 수 있습니다.

  전역 영역의 프로세스에서 비전역 영역의 프로세스와 다른 객체를 관찰할 수 있습니다. 그러면 그런 프로세스에서 시스템을 전체적으로 확인할 수 있습니다.

  영역 내에서 `procfs` 마운트는 `nodevices` 옵션을 사용하여 마운트된 것처럼 동작합니다. `procfs`에 대한 자세한 내용은 `proc(4)` 매뉴얼 페이지를 참조하십시오.

LOFS
: LOFS를 통해 마운트될 수 있는 범위는 영역에 표시할 수 있는 파일 시스템 부분으로 제한됩니다. 따라서, 영역 내에서 LOFS 마운트에 대한 제한 사항은 없습니다.

UFS, UDFS, PCFS 및 기타 저장소 기반 파일 시스템
: `zonecfg` 명령을 사용하여 `fsck` 이진이 있는 저장소 기반 파일 시스템(예: UFS)을 구성하는 경우 영역 관리자가 `raw` 매개 변수를 지정해야 합니다. 이 매개 변수는 원시(문자) 장치(예: `/dev/rdsk/c0t0d0s7`)를 나타냅니다. `zoneadmd` 데몬은 preen 모드(`fsck -p`)에서 `fsck` 명령을 자동으로 실행하여 파일 시스템을 마운트하기 전에 파일 시스템을 비대화식으로 확인하여 수정합니다. `fsck`가 실패하면 `zoneadmd`가 영역을 준비 상태로 전환할 수 없습니다. `raw`에 의해 지정되는 경로는 상대 경로가 아닙니다.

  `/usr/lib/fs/`*fstype*`/fsck`에 `fsck` 이진을 제공하지 않는 파일 시스템에 대해 `fsck`에 장치를 지정하면 오류가 발생합니다. 또한 해당 파일 시스템에 대해 `fsck` 이진이 존재하지만 `fsck`에 장치를 지정하지 않은 경우에도 오류가 발생합니다.

  자세한 내용은 262 페이지 “zoneadmd 데몬” 및 fsck(1M) 명령을 참조하십시오.

ZFS
: 214 페이지 “영역에서 마운트된 파일 시스템”에서 설명한 기본 데이터 집합 이외에 `zonecfg` 명령을 `add dataset` 리소스와 함께 사용하여 비전역 영역에 ZFS 데이터 집합을 추가할 수 있습니다. 데이터 세트는 비전역 영역에 표시 및 마운트되고 전역 영역에도 표시됩니다. 영역 관리자는 해당 데이터 집합 내에서 파일 시스템을 만들거나 삭제하고 데이터 집합의 등록 정보를 수정할 수 있습니다.

  `zfs`의 `zoned` 속성은 데이터 집합이 비전역 영역에 추가되었는지 여부를 나타냅니다.

```
# zfs get zoned tank/sales
NAME          PROPERTY    VALUE      SOURCE
tank/sales    zoned       on         local
```

  데이터 집합 리소스를 통해 비전역 영역에 위임되는 각 데이터 집합에 별칭이 지정됩니다. 데이터 집합 레이아웃은 영역에 표시되지 않습니다. 각 가칭된 데이터 집합은 풀처럼 영역에 표시됩니다. 데이터 집합의 기본 별칭은 데이터 집합 이름의 마지막 구성 요소입니다. 예를 들어 위임된 데이터 집합 `tank/sales`에 대해 기본 별칭이 사용된 경우 영역에 가상 ZFS 풀이 `sales`로 표시됩니다. 데이터 집합 리소스에서 별칭 등록 정보를 설정하여 별칭을 다른 값으로 사용자 정의할 수 있습니다.

  `rpool` 데이터 세트는 각 비전역 영역의 `zonepath` 데이터 세트에 있습니다. 모든 비전역 영역에 대해 이 영역 `rpool` 데이터 세트는 `rpool`로 가칭됩니다.

```
my-zone# zfs list -o name,zoned,mounted,mountpoint
NAME                 ZONED  MOUNTED  MOUNTPOINT
rpool                   on       no  /rpool
rpool/ROOT              on       no  legacy
rpool/ROOT/solaris      on      yes  /
rpool/export            on       no  /export
rpool/export/home       on       no  /export/home
```

데이터 세트 별칭은 ZFS 풀과 동일한 이름 제한이 적용됩니다. 이러한 제한에 대한 자세한 내용은 zpool(1M) 매뉴얼 페이지를 참조하십시오.

전역 영역에서 데이터 세트를 공유하려면 zonecfg 명령을 add fs 하위 명령과 함께 사용하여 LOFS 마운트 ZFS 파일 시스템을 추가할 수 있습니다. 전역 관리자 또는 해당 권한이 부여된 사용자는 데이터 세트의 등록 정보를 설정 및 제어할 수 있습니다.

ZFS에 대한 자세한 내용은 **Oracle Solaris 관리: ZFS 파일 시스템**의 10 장, "Oracle Solaris ZFS 고급 주제"을 참조하십시오.

NFS 클라이언트인 비전역 영역

영역은 NFS 클라이언트일 수 있습니다. 버전 2, 버전 3 및 버전 4 프로토콜이 지원됩니다. 이 NFS 버전에 대한 자세한 내용은 **Oracle Solaris 관리: 네트워크 서비스**의 "NFS 서비스의 기능"을 참조하십시오.

기본 버전은 NFS 버전 4입니다. 다음 방법 중 하나를 사용하여 클라이언트에서 다른 NFS 버전을 사용하도록 설정할 수 있습니다.

- sharectl(1M)을 사용하여 등록 정보를 설정할 수 있습니다. 영역에서 기본적으로 지정된 버전을 사용하도록 NFS_CLIENT_VERSMAX=*number*를 설정합니다. **Oracle Solaris 관리: 네트워크 서비스**의 "NFS 서비스 설정"을 참조하십시오. **Oracle Solaris 관리: 네트워크 서비스**의 "클라이언트에서 다른 NFS 버전을 선택하는 방법" 절차를 사용합니다.
- 버전 마운트를 수동으로 만들 수 있습니다. 이 방법은 sharectl 설정을 대체합니다. **Oracle Solaris 관리: 네트워크 서비스**의 "NFS 서비스 설정"을 참조하십시오. **Oracle Solaris 관리: 네트워크 서비스**의 "클라이언트에서 다른 NFS 버전을 선택하는 방법" 절차를 사용합니다.

영역에서 금지된 mknod 사용

mknod(1M) 매뉴얼 페이지에 설명된 mknod 명령을 사용하여 비전역 영역에서 특수 파일을 만들 수 없습니다.

파일 시스템 탐색

영역의 파일 시스템 이름 공간은 전역 영역에서 액세스할 수 있는 이름 공간의 일부입니다. 전역 영역에서 권한이 없는 프로세스에서는 다음과 같은 방법으로 비전역 영역의 파일 시스템 계층을 탐색할 수 없습니다.

- 영역 루트의 상위 디렉토리를 루트에서만 소유하고, 읽기, 쓰기 및 실행할 수 있도록 지정
- /proc를 사용하여 내보내는 디렉토리에 대한 액세스 제한

다른 영역에 대해 마운트된 AutoFS 노드에 액세스되지 않습니다. 전역 관리자는 자동 맵이 다른 영역에 종속되지 않도록 해야 합니다.

전역 영역에서 비전역 영역 액세스 제한

비전역 영역이 설치된 후에 시스템 백업 유틸리티 이외의 명령을 사용하여 전역 영역에서 비전역 영역을 직접 액세스해서는 안 됩니다. 또한, 비전역 영역이 알 수 없는 환경에 노출된 후에는 더 이상 안전한 영역으로 간주할 수 없습니다. 예를 들어 공개적으로 액세스할 수 있는 네트워크에 배치된 영역이 있습니다. 이 영역은 손상되거나 파일 시스템의 내용이 변경될 수 있습니다. 손상이 발생했을 가능성이 있는 경우 전역 관리자는 해당 영역을 신뢰할 수 없는 영역으로 간주해야 합니다.

다음과 같은 경우에는 `-R` 또는 `-b` 옵션이나 이와 유사한 옵션을 사용하여 대체 루트를 수락하는 모든 명령을 사용해서는 **안 됩니다**.

- 명령이 전역 영역에서 실행되는 경우
- 현재 실행 중인 시스템의 전역 영역에 대한 상대 경로인지 대체 루트의 전역 영역에 대한 상대 경로인지 여부에 상관없이 대체 루트가 비전역 영역 내의 경로를 참조하는 경우.

예를 들어 비전역 영역 루트 경로를 사용하여 전역 영역에서 실행되는 `pkgadd` 유틸리티에 `-R` *root_path* 옵션을 사용할 수 있습니다.

대체 루트 경로에서 `-R`을 사용하는 명령, 프로그램 및 유틸리티 목록은 다음과 같습니다.

- `auditreduce`
- `bart`
- `installf`
- `localeadm`
- `makeuuid`
- `metaroot`
- `pkg`
- `prodreg`
- `removef`
- `routeadm`
- `showrev`
- `syseventadm`

대체 루트 경로에서 `-b`를 사용하는 명령 및 프로그램 목록은 다음과 같습니다.

- `add_drv`
- `pprosetup`
- `rem_drv`
- `roleadd`
- `update_drv`
- `useradd`

공유 IP 비전역 영역의 네트워킹

영역이 설치된 Oracle Solaris 시스템에서는 네트워크를 통해 영역 간에 통신할 수 있습니다. 모든 영역은 별도의 바인딩 또는 연결을 사용하며, 모두 자체 서버 데몬을 실행할 수 있습니다. 이러한 데몬은 충돌 없이 동일한 포트 번호를 수신할 수 있습니다. IP 스택은 수신 연결의 IP 주소를 고려하여 충돌을 해결합니다. IP 주소는 영역을 식별합니다.

공유 IP 유형을 사용하려면 자동 네트워크 구성이 아니라 `ipadm`을 통해 전역 영역의 네트워킹 구성을 수행해야 합니다. `ipadm`이 사용 중인 경우 다음 명령은 `DefaultFixed`를 반환합니다.

```
# svcprop -p netcfg/active_ncp svc:/network/physical:default
DefaultFixed
```

공유 IP 영역 분할

공유 IP는 기본 유형이 아니지만 지원됩니다.

영역을 지원하는 시스템의 IP 스택에서 영역 간의 네트워크 트래픽 분리를 구현합니다. IP 트래픽을 수신하는 응용 프로그램은 동일한 영역에서 보낸 트래픽만 수신할 수 있습니다.

시스템의 각 논리적 인터페이스는 기본적으로 특정 전역 영역에 속합니다. `zonecfg` 유틸리티를 통해 영역에 지정되는 논리적 네트워크 인터페이스는 네트워크를 통해 통신하는 데 사용됩니다. 각 스트림과 연결은 해당 스트림 및 연결을 연 프로세스의 영역에 속합니다.

상위 계층 스트림과 논리적 인터페이스 사이의 바인딩은 제한됩니다. 스트림은 동일한 영역의 논리적 인터페이스에 대한 바인딩만 설정할 수 있습니다. 마찬가지로, 논리적 인터페이스의 패킷은 논리적 인터페이스와 동일한 영역의 상위 계층 스트림에만 전달될 수 있습니다.

각 영역에는 자체 바인드 집합이 있습니다. 주소가 이미 사용 중이므로 각 영역은 바인드 오류를 발생하지 않고 동일한 포트 번호를 수신하는 동일한 응용 프로그램을 실행할 수 있습니다. 각 영역은 다음과 같은 다양한 네트워킹 서비스의 자체 버전을 실행할 수 있습니다.

- 전체 구성 파일이 있는 인터넷 서비스 데몬(`inetd(1M)` 매뉴얼 페이지 참조)
- `sendmail`(`sendmail(1M)` 매뉴얼 페이지 참조)
- `apache`

전역 영역이 아닌 영역은 네트워크에 대한 액세스가 제한됩니다. 표준 TCP 및 UDP 소켓 인터페이스를 사용할 수 있지만 `SOCK_RAW` 소켓 인터페이스는 ICMP(Internet Control Message Protocol)에 제한됩니다. ICMP는 네트워크 오류 조건을 삭제 및 보고하거나 `ping` 명령을 사용하는 데 필요합니다.

공유 IP 네트워크 인터페이스

네트워크 연결이 필요한 각 비전역 영역은 하나 이상의 배타적 IP 주소가 있습니다. 이러한 주소는 영역에 배치될 수 있는 논리적 네트워크 인터페이스에 연결됩니다. `zonecfg`에 의해 구성되는 영역 네트워크 인터페이스는 부트될 때 자동으로 설정되어 영역에 배치됩니다. `ipadm` 명령을 사용하면 영역이 실행 중인 상태에서 논리적 인터페이스를 추가 또는 제거할 수 있습니다. 전역 관리자나 적절한 권한이 부여된 사용자만 인터페이스 구성과 네트워크 경로를 수정할 수 있습니다.

비전역 영역에서는 해당 영역의 인터페이스만 `ipadm` 명령에 표시됩니다.

자세한 내용은 `ipadm`(1M) 및 `if_tcp`(7P) 매뉴얼 페이지를 참조하십시오.

동일한 시스템의 공유 IP 영역 간 IP 트래픽

공유 IP 영역에서는 모든 지정된 IP 대상에 연결할 수 있습니다(전송 테이블에 해당 대상에 대해 사용 가능한 경로가 있는 경우). 전송 테이블을 보려면 영역 내에서 `netstat` 명령에 `-r` 옵션을 사용합니다. IP 전송 규칙은 다른 영역 또는 다른 시스템의 IP 대상과 동일합니다.

공유 IP 영역의 Oracle Solaris IP 필터

Oracle Solaris IP 필터는 Stateful 패킷 필터링 및 NAT(Network Address Translation)를 제공합니다. Stateful 패킷 필터는 활성 연결의 상태를 모니터링하고 수집된 정보를 사용하여 방화벽을 통과하도록 허용할 네트워크 패킷을 결정할 수 있습니다. 또한 Oracle Solaris IP 필터는 Stateless 패킷 필터링과 주소 풀 생성 및 관리 기능을 제공합니다. 자세한 내용은 **Oracle Solaris 관리: IP 서비스**의 20 장, "Oracle Solaris의 IP 필터(개요)"를 참조하십시오.

Oracle Solaris IP 필터를 비전역 영역에서 사용하려면 **Oracle Solaris 관리: IP 서비스**의 21 장, "IP 필터(작업)"에 설명된 대로 루프백 필터링을 설정합니다.

Oracle Solaris IP 필터는 오픈 소스 IP 필터 소프트웨어에서 파생됩니다.

공유 IP 영역의 IP Network Multipathing

IPMP(IP Network Multipathing)는 동일한 IP 링크에 여러 인터페이스가 있는 시스템에 대해 물리적 인터페이스 장애를 감지하고 투명한 네트워크 액세스 페일오버를 제공합니다. 또한 IPMP는 여러 인터페이스를 가진 시스템에 대해 패킷 로드를 분산시킵니다.

모든 네트워크 구성은 전역 영역에서 수행됩니다. 전역 영역에서 IPMP를 구성한 다음 기능을 비전역 영역으로 확장할 수 있습니다. 영역을 구성할 때 IPMP 그룹에 영역의 주소를 배치하여 기능을 확장합니다. 그러면 전역 영역의 인터페이스 중 하나가 실패할 경우 비전역 영역 주소가 다른 네트워크 인터페이스 카드로 마이그레이션됩니다.

지정된 비전역 영역에서 영역에 연결된 인터페이스만 `ipadm` 명령을 통해 표시할 수 있습니다.

자세한 내용은 363 페이지 "IP 네트워크 다중 경로 기능을 공유 IP 비전역 영역으로 확장하는 방법"을 참조하십시오. 영역 구성 절차는 243 페이지 "영역 구성 방법"을 참조하십시오. IPMP 기능, 구성 요소 및 사용법에 대한 자세한 내용은 **Oracle Solaris 관리: 네트워크 인터페이스 및 네트워크 가상화**의 14 장, "IPMP 소개"를 참조하십시오.

배타적 IP 비전역 영역의 네트워킹

배타적 IP 영역에는 자체 IP 관련 상태가 있습니다. 영역을 구성할 때 영역에 자체 데이터 링크 집합이 지정됩니다.

패킷이 물리적 링크에서 전송됩니다. 그러면 이더넷 스위치 또는 IP 라우터와 같은 장치에서 패킷을 대상으로 전송할 수 있습니다. 대상은 보낸 사람과 동일한 시스템에 있는 다른 영역일 수 있습니다.

가상 링크의 경우 패킷이 먼저 가상 스위치에 전송됩니다. 대상 링크가 동일한 장치(동일한 링크 또는 etherstub의 VNIC)를 통과하는 경우 패킷은 대상 VNIC로 직접 이동합니다. 그렇지 않으면 패킷이 VNIC를 기반으로 물리적 링크를 벗어납니다.

배타적 IP 비전역 영역에서 사용할 수 있는 기능에 대한 자세한 내용은 212 페이지 "배타적 IP 비전역 영역"을 참조하십시오.

배타적 IP 영역 분할

배타적 IP 영역은 별도의 TCP/IP 스택이 있으므로 구분이 데이터 링크 계층까지 연결됩니다. 하나 이상의 데이터 링크 이름(NIC 또는 NIC의 VLAN)이 전역 관리자에 의해 배타적 IP 영역에 지정됩니다. 영역 관리자는 전역 영역에서와 동일한 유연성과 옵션으로 데이터 링크에 대한 IP를 구성할 수 있습니다.

배타적 IP 데이터 링크 인터페이스

데이터 링크 이름을 단일 영역에 배타적으로 지정해야 합니다.

`dladm show-link` 명령을 사용하여 실행 중인 영역에 지정된 데이터 링크를 표시할 수 있습니다.

```
sol-t2000-10{pennyc}1: dladm show-link
LINK                CLASS     MTU    STATE    OVER
vsw0                phys      1500   up       --
e1000g0             phys      1500   up       --
e1000g2             phys      1500   up       --
e1000g1             phys      1500   up       --
e1000g3             phys      1500   up       --
zoneA/net0          vnic      1500   up       e1000g0
zoneB/net0          vnic      1500   up       e1000g0
aggr1               aggr      1500   up       e1000g2 e1000g3
vnic0               vnic      1500   up       e1000g1
zoneA/vnic0         vnic      1500   up       e1000g1
vnic1               vnic      1500   up       e1000g1
zoneB/vnic1         vnic      1500   up       e1000g1
vnic3               vnic      1500   up       aggr1
vnic4               vnic      1500   up       aggr1
zoneB/vnic4         vnic      1500   up       aggr1
```

자세한 내용은 dladm(1M)을 참조하십시오.

동일한 시스템의 배타적 IP 영역 간 IP 트래픽

배타적 IP 영역 간에 내부 IP 패킷 루프백이 없습니다. 모든 패킷이 데이터 링크로 전송됩니다. 일반적으로 패킷이 네트워크 인터페이스에서 전송되는 것을 의미합니다. 그러면 이더넷 스위치 또는 IP 라우터와 같은 장치에서 패킷을 대상으로 전송할 수 있습니다. 대상은 보낸 사람과 동일한 시스템에 있는 다른 영역일 수 있습니다.

배타적 IP 영역의 Oracle Solaris IP 필터

배타적 IP 영역의 전역 영역에서와 동일한 IP 필터 기능이 있습니다. 또한 IP 필터는 배타적 IP 영역과 전역 영역에서 동일한 방법으로 구성됩니다.

배타적 IP 영역의 IP Network Multipathing

IPMP(IP Network Multipathing)는 동일한 IP 링크에 여러 인터페이스가 있는 시스템에 대해 물리적 인터페이스 장애를 감지하고 투명한 네트워크 액세스 페일오버를 제공합니다. 또한 IPMP는 여러 인터페이스를 가진 시스템에 대해 패킷 로드를 분산시킵니다.

데이터 링크 구성은 전역 영역에서 수행됩니다. 먼저 zonecfg를 사용하여 하나의 영역에 여러 데이터 링크 인터페이스가 지정됩니다. 여러 데이터 링크 인터페이스를 동일한 IP 서브넷에 연결해야 합니다. 그런 다음 영역 관리자가 배타적 IP 영역에서 IPMP를 구성할 수 있습니다.

비전역 영역에서 장치 사용

영역에서 사용 가능한 장치 집합은 한 영역의 프로세스가 다른 영역에서 실행 중인 프로세스에 연결하지 못하도록 제한됩니다. 예를 들어 영역의 프로세스에서 커널 메모리를 수정하거나 루트 디스크의 내용을 수정할 수 없습니다. 따라서, 기본적으로 영역에서 사용해도 안전한 것으로 간주되는 특정 의사 장치만 사용할 수 있습니다. `zonecfg` 유틸리티를 사용하여 특정 영역에서 추가 장치를 사용할 수 있도록 지정할 수 있습니다.

/dev 및 /devices 이름 공간

`devfs`(7FS) 매뉴얼 페이지에 설명된 `devfs` 파일 시스템은 Oracle Solaris 시스템에서 `/devices`를 관리하는 데 사용됩니다. 이 이름 공간의 각 요소는 하드웨어 장치, 의사 장치 또는 넥서스 장치의 물리적 경로를 나타냅니다. 이름 공간은 장치 트리를 반영합니다. 따라서, 파일 시스템은 디렉토리 및 장치 특수 파일의 계층으로 채워집니다.

장치는 관련 `/dev` 계층에 따라 그룹화됩니다. 예를 들어 전역 영역의 `/dev` 아래에 있는 모든 장치는 전역 영역 장치로 그룹화됩니다. 비전역 영역의 경우 장치가 영역 루트 경로 아래의 `/dev` 디렉토리에 그룹화됩니다. 각 그룹은 `/dev` 디렉토리 아래에 마운트되는 하나의 마운트된 `/dev` 파일 시스템 인스턴스입니다. 따라서 전역 영역 장치는 `/dev` 아래에 마운트되고, `my-zone`이라는 비전역 영역에 대한 장치는 `/my-zone/root/dev` 아래에 마운트됩니다.

`/dev` 파일 계층은 `dev`(7FS) 매뉴얼 페이지에 설명한 `dev` 파일 시스템에 의해 관리됩니다.

주의 – `/devices` 경로 이름에 의존하는 부속 시스템은 비전역 영역에서 실행할 수 없습니다. `/dev` 경로 이름을 사용하려면 부속 시스템을 업데이트해야 합니다.

주의 – 비전역 영역에 `/dev/zvol` 내의 장치를 포함하는 일치하는 장치 리소스가 있는 경우 비전역 영역에서 이름 공간이 충돌할 수 있습니다. 자세한 내용은 `dev`(7FS) 매뉴얼 페이지를 참조하십시오.

전용 장치

장치를 특정 영역에 지정해야 할 수 있습니다. 권한이 없는 사용자가 블록 장치에 액세스하도록 허용하면 해당 장치를 사용하도록 허용하는 것이므로 시스템 장애, 버스 재설정 또는 기타 부작용이 발생할 수 있습니다. 그런 할당을 하기 이전에 다음 문제점을 고려하십시오.

- SCSI 테이프 장치를 특정 영역에 지정하기 전에 `sgen`(7D) 매뉴얼 페이지를 참조하십시오.

- 물리적 장치를 여러 영역에 배치하면 영역 사이의 위장 채널이 만들어질 수 있습니다. 그런 장치를 사용하는 전역 영역 장치는 비전역 영역에 의해 데이터 손상이 발생할 수 있습니다.

장치 드라이버 관리

비전역 영역에서는 modinfo(1M) 매뉴얼 페이지에 설명된 modinfo 명령을 사용하여 로드된 커널 모듈 목록을 확인할 수 있습니다.

플랫폼 하드웨어 구성을 수정하면 영역 보안 모델을 위반하므로 커널, 장치 및 플랫폼 관리와 관련한 대부분의 작업은 비전역 영역에서는 작동하지 않습니다. 이러한 작업은 다음과 같습니다.

- 드라이버 추가 및 제거
- 커널 모드 명시적 로드 및 언로드
- 동적 재구성(DR) 작업 시작
- 물리적 플랫폼의 상태에 영향을 주는 기능 사용

비전역 영역에서 작동하지 않거나 수정되는 유틸리티

비전역 영역에서 작동하지 않는 유틸리티

다음 유틸리티는 정상적으로 사용할 수 없는 장치를 필요로 하므로 영역에서 작동하지 않습니다.

- add_drv(add_drv(1M) 매뉴얼 페이지 참조)
- disks(disks(1M) 매뉴얼 페이지 참조)
- prtconf(prtconf(1M) 매뉴얼 페이지 참조)
- prtdiag(prtdiag(1M) 매뉴얼 페이지 참조)
- rem_drv(rem_drv(1M) 매뉴얼 페이지 참조)

SPARC: 비전역 영역에서 사용하기 위해 수정되는 유틸리티

eeprom 유틸리티를 영역에서 사용하여 설정을 볼 수 있습니다. 이 유틸리티를 사용하여 설정을 변경할 수는 없습니다. 자세한 내용은 eeprom(1M) 및 openprom(7D) 매뉴얼 페이지를 참조하십시오.

허용되는 보안 관련 유틸리티

allowed-raw-io를 사용하도록 설정한 경우 다음과 같은 유틸리티를 영역에서 사용할 수 있습니다. 보안 고려 사항을 평가해야 합니다. 장치를 추가하기 전에 333 페이지 “비전역 영역에서 장치 사용”, 335 페이지 “비전역 영역에서 실행 중인 응용 프로그램” 및 337 페이지 “비전역 영역의 권한”에서 제한 사항과 보안 고려 사항을 참조하십시오.

- cdrecord(cderecord(1) 매뉴얼 페이지 참조)
- cdrw(cdrw(1) 매뉴얼 페이지 참조)
- rmformat(rmformat(1) 매뉴얼 페이지 참조)

비전역 영역에서 실행 중인 응용 프로그램

일반적으로 모든 응용 프로그램은 비전역 영역에서 실행할 수 있습니다. 그러나 다음과 같은 응용 프로그램은 이 환경에 적합하지 않을 수 있습니다.

- 시스템에 전반적인 영향을 주는, 권한 부여된 작업을 사용하는 응용 프로그램. 예를 들어 전역 시스템 클럭을 설정하거나 물리적 메모리를 잠그는 작업이 있습니다.
- 비전역 영역에 존재하지 않는 특정 장치에 종속되는 일부 응용 프로그램(예: /dev/kmem).
- 공유 IP 영역에서 /dev/ip의 장치에 종속되는 응용 프로그램.

비전역 영역에서 사용되는 리소스 제어

영역에서 리소스 관리 기능을 사용하는 방법에 대한 자세한 내용은 제1부의 기능 설명 장을 참조하십시오.

리소스 관리 장에서 설명하는 모든 리소스 제어와 속성은 전역 및 비전역 영역 /etc/project 파일, NIS 맵 또는 LDAP 디렉토리 서비스에서 설정할 수 있습니다. 지정된 영역에 대한 설정은 해당 영역에만 적용됩니다. 다른 영역에서 자체적으로 실행 중인 프로젝트에서는 제어를 각 영역에서 개별적으로 설정할 수 있습니다. 예를 들어 전역 영역의 프로젝트 A를 project.cpu-shares=10으로 설정하고 비전역 영역의 프로젝트 A를 project.cpu-shares=5로 설정할 수 있습니다. 자체 영역에서만 작동하는 rcapd의 여러 인스턴스가 시스템에서 실행 중일 수 있습니다.

영역에서 해당 영역의 프로젝트, 작업 및 프로세스를 제어하는 데 사용되는 리소스 제어 및 속성은 풀 및 영역 전체 리소스 제어에 관한 추가 요구 사항이 적용됩니다.

풀을 특정 영역에 배타적으로 지정할 필요는 없지만 비전역 영역 하나를 리소스 풀 하나에 연결할 수 있습니다. 여러 비전역 영역에서 한 개 풀의 리소스를 공유할 수 있습니다. 그러나, 권한 있는 프로세스에서 전역 영역의 프로세스를 아무 풀에나 바운드할 수 있습니다. 리소스 제어기 poold는 작동하는 데 필요한 여러 개의 풀이 있는 전역 영역에서만 실행됩니다. 비전역 영역에서 poolstat 유틸리티를 실행하면 이 영역과 연결된 풀에 대한 정보만 표시됩니다. 비전역 영역에서 인수 없이 pooladm 명령을 실행하면 이 영역과 연결된 풀에 대한 정보만 표시됩니다.

영역 전체 리소스 제어는 project 파일에서 설정할 경우 적용되지 않습니다. 영역 전체 리소스 제어는 zonecfg 유틸리티를 통해 설정됩니다.

영역이 설치된 시스템의 FSS(Fair Share Scheduler)

이 절에서는 영역이 있는 FSS(Fair Share Scheduler)를 사용하는 방법에 대해 설명합니다.

전역 영역 또는 비전역 영역의 FSS 공유 구획

영역에 대한 FSS CPU 할당은 계층적입니다. 전역 영역 및 비전역 영역에 대한 할당은 전역 관리자에 의해 영역 전체 리소스 제어 `zone.cpu-shares`를 통해 설정됩니다. 그러면 해당 영역에서 각 프로젝트에 대해 `project.cpu-shares` 리소스 제어를 정의하여 영역 전체 제어를 통해 설정된 공유를 세분화할 수 있습니다.

`zonecfg` 명령을 사용하여 영역 할당을 지정하려면 255 페이지 "전역 영역의 zone.cpu-shares를 설정하는 방법"을 참조하십시오. `project.cpu-shares`에 대한 자세한 내용은 78 페이지 "사용 가능한 리소스 제어"를 참조하십시오. 임시로 할당을 설정하는 방법을 보여 주는 절차의 예는 366 페이지 "영역이 설치된 Oracle Solaris 시스템에서 Fair Share Scheduler 사용"을 참조하십시오.

영역 간의 할당 균형

`zone.cpu-shares`를 사용하여 전역 영역 및 비전역 영역에서 FSS 할당을 지정할 수 있습니다. FSS가 시스템의 기본 스케줄러이고 공유가 할당되지 않은 경우 기본적으로 각 영역에 하나의 공유가 지정됩니다. 시스템에 비전역 영역이 하나 있고 `zone.cpu-shares`를 통해 이 영역에 두 개의 공유를 제공할 경우 비전역 영역에서 전역 영역과 관련하여 수신할 CPU 부분을 정의합니다. 두 영역 사이의 CPU 비율은 2:1입니다.

영역이 설치된 시스템의 확장된 계정

확장 계정 부속 시스템은 전역 영역에서 실행될 때 전체 시스템(비전역 영역 포함)에 대한 정보를 수집 및 보고합니다. 또한 전역 관리자는 영역별로 리소스 사용을 결정할 수 있습니다.

확장된 계정 부속 시스템은 프로세스 기반 계정 및 작업 기반 계정에 대해 영역별로 다른 계정 설정 및 파일을 허용합니다. 프로세스에 대해 `EXD PROC ZONENAME` 영역 이름을 사용하고 작업에 대해 `EXD TASK ZONENAME` 영역 이름을 사용하여 `exacct` 레코드를 태그 지정할 수 있습니다. 계정 레코드는 전역 영역의 계정 파일과 영역별 계정 파일에 기록됩니다. `EXD TASK HOSTNAME`, `EXD PROC HOSTNAME` 및 `EXD HOSTNAME` 레코드는 전역 영역의 노드 이름 대신 프로세스 또는 작업이 실행되는 영역에 대한 `uname -n` 값을 포함합니다.

IPQoS 흐름 계정에 대한 자세한 내용은 **Oracle Solaris 관리: IP 서비스**의 31 장, "흐름 계산 및 통계 수집 사용(작업)"을 참조하십시오.

비전역 영역의 권한

프로세스는 권한의 일부로 제한됩니다. 권한 제한은 영역에서 다른 영역에 영향을 줄 수 있는 작업을 수행하지 못하도록 방지합니다. 권한 설정은 영역 내에서 권한 있는 사용자의 기능을 제한합니다. 지정된 영역에서 사용할 수 있는 권한 목록을 표시하려면 `ppriv` 유틸리티를 사용합니다.

다음 표는 영역에 관한 모든 Oracle Solaris 권한과 각 권한의 상태를 보여 줍니다. 선택적 권한은 기본 권한 집합에 포함되지 않지만 `limitpriv` 등록 정보를 통해 지정할 수 있습니다. 필수 권한은 결과 권한 집합에 포함되어야 합니다. 금지된 권한은 결과 권한 집합에 포함될 수 없습니다.

표 25–1 영역의 권한 상태

권한	상태	주
`cpc_cpu`	선택 사항	특정 `cpc`(3CPC) 카운터에 액세스
`dtrace_proc`	선택 사항	`fasttrap` 및 `pid` 공급자, `plockstat`(1M)
`dtrace_user`	선택 사항	`profile` 및 `syscall` 공급자
`graphics_access`	선택 사항	`ioctl`(2)에서 `agpgart_io`(7I)에 액세스
`graphics_map`	선택 사항	`mmap`(2)에서 `agpgart_io`(7I)에 액세스
`net_rawaccess`	공유 IP 영역에서는 선택 사항입니다. 배타적 IP 영역에서는 기본값입니다.	원시 `PF_INET/PF_INET6` 패킷 액세스
`proc_clock_highres`	선택 사항	고해상도 타이머 사용
`proc_priocntl`	선택 사항	예약 제어: `priocntl`(1)
`sys_ipc_config`	선택 사항	IPC 메시지 대기열 버퍼 크기 증가
`sys_time`	선택 사항	시스템 시간 조작: `xntp`(1M)
`dtrace_kernel`	금지됨	현재 지원되지 않음
`proc_zone`	금지됨	현재 지원되지 않음
`sys_config`	금지됨	현재 지원되지 않음
`sys_devices`	금지됨	현재 지원되지 않음
`sys_dl_config`	금지됨	현재 지원되지 않음
`sys_linkdir`	금지됨	현재 지원되지 않음
`sys_net_config`	금지됨	현재 지원되지 않음

표 25–1 영역의 권한 상태 *(계속)*

권한	상태	주
sys_res_config	금지됨	현재 지원되지 않음
sys_smb	금지됨	현재 지원되지 않음
sys_suser_compat	금지됨	현재 지원되지 않음
proc_exec	필수, 기본값	init(1M)을 시작하는 데 사용됨
proc_fork	필수, 기본값	init(1M)을 시작하는 데 사용됨
sys_mount	필수, 기본값	필수 파일 시스템을 마운트하는 데 필요함
sys_flow_config	필수, 배타적 IP 영역에서는 기본값 공유 IP 영역에서는 금지됨	흐름을 구성하는 데 필요함
sys_ip_config	필수, 배타적 IP 영역에서는 기본값 공유 IP 영역에서는 금지됨	영역을 부트하고 배타적 IP 영역에서 IP 네트워킹을 초기화하는 데 필요함
sys_iptun_config	필수, 배타적 IP 영역에서는 기본값 공유 IP 영역에서는 금지됨	IP 터널 링크 구성
contract_event	기본값	계약 파일 시스템에서 사용됨
contract_identity	기본값	프로세스 계약 템플리트의 서비스 FMRI 값 설정
contract_observer	기본값	UID에 상관없이 계약 관찰
file_chown	기본값	파일 소유권 변경
file_chown_self	기본값	소유한 파일의 소유자/그룹 변경
file_dac_execute	기본값	모드/ACL에 상관없이 실행 액세스
file_dac_read	기본값	모드/ACL에 상관없이 읽기 액세스
file_dac_search	기본값	모드/ACL에 상관없이 검색 액세스
file_dac_write	기본값	모드/ACL에 상관없이 쓰기 액세스
file_link_any	기본값	소유자에 상관없이 링크 액세스
file_owner	기본값	소유자에 상관없이 기타 액세스
file_setid	기본값	setid, setgid, setuid 파일에 대한 권한 변경
ipc_dac_read	기본값	모드에 상관없이 IPC 읽기 액세스

표 25–1 영역의 권한 상태 (계속)

권한	상태	주
ipc_dac_owner	기본값	모드에 상관없이 IPC 쓰기 액세스
ipc_owner	기본값	모드에 상관없이 IPC 기타 액세스
net_icmpaccess	기본값	ICMP 패킷 액세스: ping(1M)
net_privaddr	기본값	권한 부여된 포트에 바인딩
proc_audit	기본값	감사 레코드 생성
proc_chroot	기본값	root 디렉토리 변경
proc_info	기본값	프로세스 검사
proc_lock_memory	기본값	메모리 잠금: shmctl(2)및 mlock(3C) 시스템 관리자가 이 권한을 비전역 영역에 지정한 경우 영역에서 모든 메모리를 잠그지 못하도록 zone.max-locked-memory 리소스 제어를 설정하는 것이 좋습니다.
proc_owner	기본값	소유자에 상관없이 프로세스 제어
proc_session	기본값	세션에 상관없이 프로세스 제어
proc_setid	기본값	사용자/그룹 ID를 마음대로 설정
proc_taskid	기본값	호출자에게 작업 ID 지정
sys_acct	기본값	계정 관리
sys_admin	기본값	간단한 시스템 관리 작업
sys_audit	기본값	감사 관리
sys_nfs	기본값	NFS 클라이언트 지원
sys_ppp_config	배타적 IP 영역의 기본값 공유 IP 영역에서는 금지됨	PPP(sppp) 인터페이스 만들기 및 삭제, PPP 터널(sppptun) 구성
sys_resource	기본값	리소스 제한 조작
sys_share	기본값	파일 시스템을 공유하는 데 필요한 sharefs 시스템 호출을 허용합니다. 영역 구성에서 권한을 금지하여 영역에서 NFS 공유를 금지할 수 있습니다.

다음 표는 영역에 관한 모든 Oracle Solaris Trusted Extensions 권한과 각 권한의 상태를 보여 줍니다. 선택적 권한은 기본 권한 집합에 포함되지 않지만 limitpriv 등록 정보를 통해 지정할 수 있습니다.

주 – Oracle Trusted Solaris 권한은 Oracle Trusted Extensions를 사용하여 시스템을 구성한 경우에만 해석됩니다.

표 25–2 영역의 Oracle Solaris Trusted Extensions 권한 상태

Oracle Solaris Trusted Extensions 권한	상태	주
file_downgrade_sl	선택 사항	파일 또는 디렉토리의 민감도 레이블을 기존 민감도 레이블을 지배하지 않는 민감도 레이블로 설정
file_upgrade_sl	선택 사항	파일 또는 디렉토리의 민감도 레이블을 기존 민감도 레이블을 지배하는 민감도 레이블로 설정
sys_trans_label	선택 사항	민감도 레이블에 의해 지배되지 않는 레이블 변환
win_colormap	선택 사항	색상맵 제한 대체
win_config	선택 사항	X 서버에서 영구히 보존되는 리소스 구성 또는 삭제
win_dac_read	선택 사항	클라이언트의 사용자 ID가 소유하지 않은 창 리소스에서 읽기
win_dac_write	선택 사항	클라이언트의 사용자 ID가 소유하지 않은 창 리소스에 쓰기 또는 창 리소스 만들기
win_devices	선택 사항	입력 장치에서 작업을 수행합니다.
win_dga	선택 사항	직접 그래픽 액세스 X 프로토콜 확장 사용, 프레임 버퍼 권한 필요
win_downgrade_sl	선택 사항	창 리소스의 민감도 레이블을 기존 레이블에 지배되는 새 레이블로 변경
win_fontpath	선택 사항	다른 글꼴 경로 추가
win_mac_read	선택 사항	클라이언트의 레이블을 지배하는 레이블을 가진 창 리소스에서 읽기
win_mac_write	선택 사항	클라이언트의 레이블과 다른 레이블을 가진 창 리소스에 쓰기
win_selection	선택 사항	확인자의 개입 없이 데이터 이동 요청
win_upgrade_sl	선택 사항	창 리소스의 민감도 레이블을 기존 레이블에 지배되지 않는 새 레이블로 변경
net_bindmlp	기본값	MLP(다중 레벨 포트)에 바인딩 허용

표 25–2 영역의 Oracle Solaris Trusted Extensions 권한 상태 *(계속)*

Oracle Solaris Trusted Extensions 권한	상태	주
net_mac_aware	기본값	NFS를 통해 아래로 읽기 허용

비전역 영역 구성에서 권한을 변경하려면 242 페이지 “영역 구성, 확인 및 커밋”을 참조하십시오.

권한 집합을 검사하려면 351 페이지 “ppriv 유틸리티 사용”을 참조하십시오. 권한에 대한 자세한 내용은 ppriv(1) 매뉴얼 페이지 및 **시스템 관리 설명서: 보안 서비스**를 참조하십시오.

영역에서 IP 보안 구조 사용

IP 데이터그램 보호를 제공하는 IPsec(Internet Protocol Security) 구조에 대한 자세한 내용은 **Oracle Solaris 관리: IP 서비스**의 14 장, “IP 보안 아키텍처(개요)”를 참조하십시오. IKE(Internet Key Exchange) 프로토콜은 인증 및 암호화를 위한 필수 키 입력 자료를 자동으로 관리하는 데 사용됩니다.

자세한 내용은 ipsecconf(1M) 및 ipseckey(1M) 매뉴얼 페이지를 참조하십시오.

공유 IP 영역의 IP 보안 구조

IPsec를 전역 영역에서 사용할 수 있습니다. 그러나, 비전역 영역의 IPsec는 IKE를 사용할 수 없습니다. 따라서, 전역 영역의 IKE(Internet Key Exchange) 프로토콜을 사용하여 비전역 영역에 대한 IPsec 키 및 정책을 관리해야 합니다. 구성 중인 비전역 영역에 해당하는 소스 주소를 사용합니다.

배타적 IP 영역의 IP 보안 구조

IPsec를 배타적 IP 영역에서 사용할 수 있습니다.

영역에서 Oracle Solaris Auditing 사용

감사 레코드는 이벤트(예: 시스템 로그인, 파일에 쓰기, 등)에 대해 설명합니다. Oracle Solaris Auditing은 영역을 실행 중인 시스템에서 다음과 같은 두 가지 감사 모델을 제공합니다.

- 전역 영역에서 모든 영역은 동일하게 감사됩니다. 이 모델은 전역 영역에서 모든 영역을 관리하는 경우에 사용됩니다(예: 영역을 통해 서비스 격리).

- 각 영역은 전역 영역과 별도로 감사됩니다. 이 모델은 각 영역이 별도로 관리되는 경우에 사용됩니다(예: 영역별로 서버 통합).

Oracle Solaris Auditing에 대한 자세한 내용은 **Oracle Solaris 관리: 보안 서비스**의 26 장, "감사(개요)"를 참조하십시오. 감사와 관련한 영역 고려 사항은 **Oracle Solaris 관리: 보안 서비스**의 "Oracle Solaris 영역이 있는 시스템에 대한 감사" 및 **Oracle Solaris 관리: 보안 서비스**의 "영역에서 감사 서비스 구성(작업)"을 참조하십시오. 자세한 내용은 `auditconfig`(1M), `auditreduce`(1M), `usermod`(1M) 및 `user_attr`(4) 매뉴얼 페이지를 참조하십시오.

주 – 임시로 활성화되지만 저장소에서 설정되지 않는 감사 정책을 사용할 수도 있습니다.

자세한 내용은 **Oracle Solaris 관리: 보안 서비스**의 "감사 정책을 변경하는 방법"에 나오는 예제를 참조하십시오.

영역의 코어 파일

`coreadm` 명령은 프로세스의 비정상적인 종료로 인해 생성되는 코어 파일의 이름과 위치를 지정하는 데 사용됩니다. 프로세스가 실행된 영역의 *zonename*을 포함하는 코어 파일 경로는 `%z` 변수를 지정하여 생성할 수 있습니다. 경로 이름은 영역의 루트 디렉토리에 상대적입니다.

자세한 내용은 `coreadm`(1M) 및 `core`(4) 매뉴얼 페이지를 참조하십시오.

비전역 영역에서 DTrace 실행

`dtrace_proc` 및 `dtrace_user` 권한만 필요한 DTrace 프로그램은 비전역 영역에서 실행될 수 있습니다. 비전역 영역에서 사용할 수 있는 권한 집합에 이러한 권한을 추가하려면 `zonecfg limitpriv` 등록 정보를 사용합니다. 자세한 내용은 356 페이지 "DTrace를 사용하는 방법"을 참조하십시오.

`dtrace_proc`를 통해 지원되는 공급자는 `fasttrap` 및 `pid`입니다. `dtrace_user`를 통해 지원되는 공급자는 `profile` 및 `syscall`입니다. DTrace 공급자 및 작업은 영역 범위로만 제한됩니다.

자세한 내용은 337 페이지 "비전역 영역의 권한"을 참조하십시오.

영역이 설치된 Oracle Solaris 시스템 백업 정보

개별 비전역 영역에서 백업을 수행하거나 전역 영역에서 전체 시스템을 백업할 수 있습니다.

루프백 파일 시스템 디렉토리 백업

비전역 영역에서 루프백 파일 시스템(lofs)을 백업하지 마십시오.

비전역 영역에서 read/write 루프백 파일 시스템을 백업 및 복원하는 경우 해당 파일 시스템이 read/write 마운트되는 모든 다른 영역이나 전역 영역에서도 이러한 파일 시스템을 쓸 수 있습니다. 이러한 파일 시스템을 전역 영역에서만 백업 및 복원하여 복사본이 여러 개 생성되는 것을 방지하십시오.

전역 영역에서 시스템 백업

다음과 같은 경우 전역 영역에서 백업을 수행할 수 있습니다.

- 비전역 영역의 구성과 응용 프로그램 데이터를 함께 백업하려는 경우
- 재해 복구 기능이 주요 관심사인 경우. 영역의 루트 파일 시스템과 해당 구성 데이터 및 전역 영역의 데이터를 비롯하여 시스템의 거의 모든 항목을 복원해야 하는 경우 전역 영역에서 백업을 수행해야 합니다.
- 상업용 네트워크 백업 소프트웨어를 사용하는 경우.

주 – 가능하면 모든 상속된 lofs 파일 시스템을 건너뛰도록 네트워크 백업 소프트웨어를 구성해야 합니다. 영역 및 응용 프로그램에서 백업할 데이터를 중지했을 때 백업을 수행해야 하는 경우

시스템의 개별 비전역 영역 백업

다음과 같은 경우 비전역 영역에서 백업을 수행할 수 있습니다.

- 비전역 영역 관리자가 심각하지 않은 오류를 복구하거나 영역에 특정한 응용 프로그램 또는 사용자 데이터를 복원해야 하는 경우.
- 파일 단위로 백업하는 프로그램을 사용할 경우(예: tar 또는 cpio). tar(1) 및 cpio(1) 매뉴얼 페이지를 참조하십시오.
- 영역에서 실행 중인 특정 응용 프로그램 또는 서비스의 백업 소프트웨어를 사용하는 경우. 전역 영역과 비전역 영역 간에 응용 프로그램 환경(예: 디렉토리 경로 및 설치된 소프트웨어)이 다르기 때문에 전역 영역에서 백업 소프트웨어를 실행하기 어려울 수 있습니다.

응용 프로그램이 각 비전역 영역에서 자체 백업 일정에 스냅샷을 수행하고 해당 백업을 전역 영역에서 내보낸 쓰기 가능한 디렉토리에 저장할 수 있는 경우 전역 영역 관리자는 전역 영역에서 백업 전략의 일환으로 이러한 개별 백업을 선택할 수 있습니다.

Oracle Solaris ZFS 백업 만들기

ZFS `send` 명령은 표준 출력에 기록될 ZFS 스냅샷의 스트림 표현을 만듭니다. 기본적으로 전체 스트림이 생성됩니다. 출력을 파일 또는 다른 시스템으로 재지정할 수 있습니다. ZFS `receive` 명령은 표준 입력에 제공한 스트림에 내용이 지정된 스냅샷을 만듭니다. 전체 스트림이 수신된 경우 새 파일 시스템도 생성됩니다. 이 명령으로 ZFS 스냅샷 데이터를 전송하고 ZFS 스냅샷 데이터 및 파일 시스템을 수신할 수 있습니다.

ZFS `send` 및 `receive` 명령 이외에도 `tar` 및 `cpio` 명령 등의 아카이브 유틸리티를 사용하여 ZFS 파일을 저장할 수도 있습니다. 이러한 유틸리티는 ZFS 파일 속성 및 액세스 제어 목록(ACL)을 저장하고 복원합니다. `tar` 및 `cpio` 명령에 적합한 옵션은 매뉴얼 페이지를 참조하십시오.

자세한 내용과 예제는 **Oracle Solaris 관리: ZFS 파일 시스템**의 7 장, “Oracle Solaris ZFS 스냅샷 및 복제 작업”을 참조하십시오.

비전역 영역에서 백업할 항목 결정

비전역 영역의 모든 항목을 백업할 수 있습니다. 영역의 구성은 자주 변경되지 않으므로 응용 프로그램 데이터만 백업할 수도 있습니다.

응용 프로그램 데이터만 백업

응용 프로그램 데이터가 파일 시스템의 특정 부분에 보관되는 경우 이 데이터만 정기적으로 백업할 수 있습니다. 영역의 루트 파일 시스템은 자주 변경되지 않으므로 자주 백업할 필요가 없을 수도 있습니다.

응용 프로그램에서 파일을 배치하는 위치를 결정해야 합니다. 파일을 저장할 수 있는 위치는 다음과 같습니다.

- 사용자의 홈 디렉토리
- `/etc`(구성 데이터 파일의 경우)
- `/var`

응용 프로그램 관리자가 데이터가 저장되는 위치를 알고 있는 경우 시스템을 만들고 영역별 쓰기 가능한 디렉토리를 각 영역에서 사용할 수 있도록 지정할 수 있습니다. 그러면 각 영역에 자체 백업을 저장할 수 있습니다. 전역 관리자 또는 적절한 권한이 부여된 사용자는 이 위치를 시스템에서 백업할 위치 중 하나로 지정할 수 있습니다.

일반 데이터베이스 백업 작업

데이터베이스 응용 프로그램 데이터가 자체 디렉토리에 없는 경우 다음과 같은 규칙이 적용됩니다.

- 데이터베이스가 일관된 상태에 있어야 합니다.

  데이터베이스에 디스크에 비울 내부 버퍼가 있으므로 데이터베이스를 중지해야 합니다. 전역 영역에서 백업을 시작하기 전에 비전역 영역의 데이터베이스를 종료해야 합니다.
- 각 영역에서 파일 시스템 기능을 사용하여 데이터 스냅샷을 만든 다음 전역 영역에서 직접 스냅샷을 백업합니다.

  이 프로세스는 백업 창에 대해 경과되는 시간을 최소화하고 모든 영역에서 백업 클라이언트/모듈이 필요하지 않습니다.

테이프 백업

각 비전역 영역에서 편리한 때에 개별 파일 시스템의 스냅샷을 만들 수 있습니다. 이 때 응용 프로그램이 잠시 중지됩니다. 나중에, 전역 영역에서 각 스냅샷을 백업하였다가 응용 프로그램이 다시 실행된 이후에 테이프에 보관할 수 있습니다.

이 방법은 다음과 같은 이점이 있습니다.

- 필요한 테이프 장치 수가 적습니다.
- 비전역 영역 간의 조정이 필요하지 않습니다.
- 장치를 영역에 직접 지정할 필요가 없으므로 보안이 향상됩니다.
- 일반적으로 이 방법은 전역 영역에서 시스템 관리를 지속합니다.

비전역 영역 복원 정보

전역 영역에서 수행된 백업을 복원할 경우 전역 관리자 또는 적절한 권한이 부여된 사용자는 해당 영역을 다시 설치한 다음 해당 영역의 파일을 복원할 수 있습니다. 여기서는 다음을 가정합니다.

- 복원할 영역의 구성이 백업을 수행할 때의 구성과 동일합니다.
- 백업이 수행된 시간과 영역을 복원할 시간 사이에 전역 영역이 업데이트되지 않았습니다.

그렇지 않으면, 복원 과정에서 수동으로 병합해야 할 일부 파일을 덮어쓸 수 있습니다.

주 – 전역 영역의 모든 파일 시스템이 손실된 경우 전역 영역에서 모든 항목을 복원하면 비전역 영역도 함께 복원됩니다(비전역 영역의 해당 루트 파일 시스템이 백업에 포함되어 있는 경우).

영역이 설치된 시스템에서 사용되는 명령

표 25–3에 식별된 명령은 영역 기능에 대한 기본 관리 인터페이스를 제공합니다.

표 25–3 영역을 관리 및 모니터링하는 데 사용되는 명령

명령 참조	설명
zlogin(1)	비전역 영역에 로그인합니다.
zonename(1)	현재 영역의 이름을 인쇄합니다.
zonestat(1)	영역 리소스 사용을 관찰하는 데 사용됩니다.
zoneadm(1M)	시스템에서 영역을 관리합니다.
zonecfg(1M)	영역 구성을 설정하는 데 사용됩니다.
getzoneid(3C)	영역 ID와 이름 간에 매핑하는 데 사용됩니다.
zones(5)	영역 기능에 대한 설명을 제공합니다.
zcons(7D)	영역 콘솔 장치 드라이버

zoneadmd 데몬은 영역의 가상 플랫폼 매핑을 위한 기본 프로세스입니다. zoneadmd 데몬의 매뉴얼 페이지는 zoneadmd(1M)입니다. 이 데몬은 프로그래밍 인터페이스에 포함되지 않습니다.

다음 표는 rcapd(Resource Capping Daemon)에 사용되는 명령을 보여 줍니다.

표 25–4 rcapd와 함께 사용되는 명령

명령 참조	설명
rcapstat(1)	상한값이 설정된 프로젝트의 리소스 사용률을 모니터합니다.
rcapadm(1M)	rcapd(Resource Capping Daemon)를 구성하고 rcapd가 구성된 경우 rcapd의 현재 상태를 표시하고 리소스 최대 가용량을 사용 또는 사용 안함으로 설정합니다.
rcapd(1M)	리소스 상한값 지원 데몬입니다.

다음 표에 식별된 명령은 영역이 설치된 Oracle Solaris 시스템에서 사용하도록 수정되었습니다. 이러한 명령은 영역에 특정하거나 다른 정보를 제공하는 옵션이 있습니다. 명령은 매뉴얼 페이지 섹션에 나열되어 있습니다.

표 25–5 영역이 설치된 Oracle Solaris 시스템에서 사용하도록 수정된 명령

명령 참조	설명
ipcrm(1)	-z *zone* 옵션을 추가했습니다. 이 옵션은 전역 영역에서 명령을 실행하는 경우에만 유용합니다.
ipcs(1)	-z *zone* 옵션을 추가했습니다. 이 옵션은 전역 영역에서 명령을 실행하는 경우에만 유용합니다.
pgrep(1)	-z *zoneidlist* 옵션을 추가했습니다. 이 옵션은 전역 영역에서 명령을 실행하는 경우에만 유용합니다.
ppriv(1)	-l 옵션과 함께 사용하여 현재 영역에서 사용 가능한 모든 권한을 나열하도록 zone 표현식을 추가했습니다. 또한 - v를 zone 뒤에 사용하여 상세 정보를 출력할 수 있습니다.
priocntl(1)	영역 ID를 *idlist* 및 -i *idtype*에서 사용하여 프로세스를 지정할 수 있습니다. priocntl - i *zoneid* 명령을 사용하여 실행 중인 프로세스를 비전역 영역의 다른 일정 클래스로 이동할 수 있습니다.
proc(1)	-z *zone* 옵션을 ptree에만 추가했습니다. 이 옵션은 전역 영역에서 명령을 실행하는 경우에만 유용합니다.
ps(1)	-o 옵션과 함께 사용되는 인식된 format 이름 목록에 *zonename* 및 *zoneid*를 추가했습니다. 지정된 영역의 프로세스만 나열하도록 -z *zonelist*를 추가했습니다. 영역 이름 또는 영역 ID를 사용하여 영역을 지정할 수 있습니다. 이 옵션은 전역 영역에서 명령을 실행하는 경우에만 유용합니다. 프로세스와 연결된 영역의 이름을 인쇄하도록 -Z를 추가했습니다. 이름은 추가 열 제목 ZONE 아래에 인쇄됩니다.
renice(1)	-i 옵션과 함께 사용되는 유효한 인수 목록을 나열하도록 *zoneid*를 추가했습니다.
sar(1)	풀 기능을 사용하도록 설정한 비전역 영역에서 실행되는 경우 -b, -c -g, -m, - p, -u, -w 및 -y 옵션은 영역이 바인드되는 풀의 프로세서 세트에 있는 프로세서에 대해서만 값을 표시합니다.
auditconfig(1M)	zonename 토큰을 추가했습니다.
auditreduce(1M)	-z *zone-name* 옵션을 추가했습니다. 영역의 감사 로그를 가져오는 기능을 추가했습니다.
coreadm(1M)	프로세스가 실행된 영역을 식별하는 변수 %z를 추가했습니다.
df(1M)	모든 표시된 영역의 마운트를 표시하도록 -Z 옵션을 추가했습니다. 이 옵션은 비전역 영역에는 적용되지 않습니다.

표 25–5 영역이 설치된 Oracle Solaris 시스템에서 사용하도록 수정된 명령 (계속)

명령 참조	설명
dladm(1M)	기본 명령 출력에 영역 열을 추가하는 -Z 옵션을 show 하위 명령에 추가했습니다. 영역 열에는 리소스가 현재 지정된 영역이 표시됩니다.
dlstat(1M)	기본 명령 출력에 영역 열을 추가하는 -Z 옵션을 show 하위 명령에 추가했습니다. 영역 열에는 리소스가 현재 지정된 영역이 표시됩니다.
iostat(1M)	풀 기능을 사용하도록 설정한 비전역 영역에서 실행되는 경우 영역이 바인드되는 풀의 프로세서 세트에 있는 프로세서에 대해서만 정보가 제공됩니다.
ipadm(1M)	인터넷 프로토콜 네트워크 인터페이스 및 TCP/IP 조정 가능 속성을 구성합니다. from-gz 유형은 비전역 영역에만 표시되며, 전역 영역의 비전역 배타적 IP 영역에 대해 구성된 allowed-address 등록 정보를 기반으로 주소가 구성되었음을 나타냅니다. zone 주소 등록 정보는 allowed-address에서 참조하는 모든 주소를 배치해야 하는 영역을 지정합니다. 영역을 공유 IP 영역으로 구성해야 합니다.
kstat(1M)	전역 영역에서 실행된 경우 kstat가 모든 영역에 대해 표시됩니다. 비전역 영역에서 실행된 경우 일치하는 *zoneid*가 있는 kstat만 표시됩니다.
mpstat(1M)	풀 기능을 사용하도록 설정한 비전역 영역에서 실행되는 경우 이 명령은 영역이 바인드되는 풀의 프로세서 세트에 있는 프로세서에 대해서만 라인을 표시합니다.
ndd(1M)	전역 영역에서 사용된 경우 모든 영역에 대한 정보를 표시합니다. 배타적 IP 영역의 TCP/IP 모듈에 대한 ndd는 해당 영역에 대한 정보를 표시합니다.
netstat(1M)	현재 영역에 대해서만 정보를 표시합니다.
nfsstat(1M)	현재 영역에 대해서만 통계를 표시합니다.
poolbind(1M)	*zoneid* 목록을 추가했습니다. 리소스 풀이 있는 영역을 사용하는 방법은 135 페이지 “영역에서 사용되는 리소스 풀”을 참조하십시오.
prstat(1M)	-z *zoneidlist* 옵션을 추가했습니다. -Z 옵션을 추가했습니다. 풀 기능을 사용하도록 설정한 비전역 영역에서 실행되는 경우 영역이 바인드되는 풀의 프로세서 세트에 있는 프로세서에 대해서만 프로세스에서 사용한 최근 CPU 시간 비율(%)이 표시됩니다. -a, -t, -T, - J 및 -Z 옵션의 출력에는 SIZE 열 대신 SWAP가 표시됩니다. 보고된 스왑은 영역의 프로세스 및 tmpfs 마운트에서 사용된 총 스왑입니다. 이 값은 각 영역에 예약된 스왑을 모니터링하도록 지원하며 적합한 zone.max-swap 설정을 선택하는 데 사용될 수 있습니다.
psrinfo(1M)	비전역 영역에서 실행된 경우 영역에 표시된 프로세서에 대한 정보만 표시됩니다.

표 25–5 영역이 설치된 Oracle Solaris 시스템에서 사용하도록 수정된 명령 (계속)

명령 참조	설명
traceroute(1M)	사용법 변경. 비전역 영역에서 지정한 경우 "don't fragment(조각화 안 함)" 비트가 항상 설정되므로 -F 옵션은 적용되지 않습니다.
vmstat(1M)	풀 기능을 사용하도록 설정한 비전역 영역에서 실행되는 경우 영역이 바인드되는 풀의 프로세서 세트에 있는 프로세서에 대해서만 통계를 보고합니다. -p 옵션의 출력과 page, faults 및 cpu 보고서 필드에 적용됩니다.
priocntl(2)	P_ZONEID *id* 인수를 추가했습니다.
processor_info(2)	호출자가 비전역 영역에 있고 풀 기능을 사용하도록 설정했지만 프로세서가 영역이 바인드되는 풀의 프로세서 세트에 없을 경우 오류가 반환됩니다.
p_online(2)	호출자가 비전역 영역에 있고 풀 기능을 사용하도록 설정했지만 프로세서가 영역이 바인드되는 풀의 프로세서 세트에 없을 경우 오류가 반환됩니다.
pset_bind(2)	P_ZONEID를 *idtype*으로 추가했습니다. P_MYID 사양에 대해 선택 가능한 옵션에 영역을 추가했습니다. EINVAL 오류 설명의 유효한 *idtype* 목록에 P_ZONEID를 추가했습니다.
pset_info(2)	호출자가 비전역 영역에 있고 풀 기능을 사용하도록 설정했지만 프로세서가 영역이 바인드되는 풀의 프로세서 세트에 없을 경우 오류가 반환됩니다.
pset_list(2)	호출자가 비전역 영역에 있고 풀 기능을 사용하도록 설정했지만 프로세서가 영역이 바인드되는 풀의 프로세서 세트에 없을 경우 오류가 반환됩니다.
pset_setattr(2)	호출자가 비전역 영역에 있고 풀 기능을 사용하도록 설정했지만 프로세서가 영역이 바인드되는 풀의 프로세서 세트에 없을 경우 오류가 반환됩니다.
sysinfo(2)	PRIV_SYS_CONFIG를 PRIV_SYS_ADMIN으로 변경했습니다.
umount(2)	*file*이 가리키는 파일이 절대 경로가 아닌 경우 ENOENT가 반환됩니다.
getloadavg(3C)	호출자가 비전역 영역에 있고 풀 기능을 사용하도록 설정한 경우 PS_MYID를 psetid로 사용하여 호출할 때와 동일하게 동작합니다.
getpriority(3C)	지정할 수 있는 대상 프로세스에 영역 ID를 추가했습니다. EINVAL 오류 설명에 영역 ID를 추가했습니다.
priv_str_to_set(3C)	호출자의 영역에서 사용 가능한 모든 권한 집합에 "zone(영역)" 문자열을 추가했습니다.
pset_getloadavg(3C)	호출자가 비전역 영역에 있고 풀 기능을 사용하도록 설정했지만 프로세서가 영역이 바인드되는 풀의 프로세서 세트에 없을 경우 오류가 반환됩니다.

표 25–5 영역이 설치된 Oracle Solaris 시스템에서 사용하도록 수정된 명령 (계속)

명령 참조	설명
sysconf(3C)	호출자가 비전역 영역에 있고 풀 기능을 사용하도록 설정한 경우 sysconf(_SC_NPROCESSORS_CONF) 및 sysconf(_SC_NPROCESSORS_ONLN)는 영역이 바인드되는 풀의 프로세서 세트에 있는 총 온라인 프로세서 개수를 반환합니다.
ucred_get(3C)	ucred_getzoneid() 함수를 추가했습니다. 이 함수는 프로세스의 영역 ID 또는 -1(영역 ID를 사용할 수 없는 경우)을 반환합니다.
core(4)	n_type: NT_ZONENAME을 추가했습니다. 이 항목에는 프로세스가 실행된 영역의 이름을 설명하는 문자열이 포함되어 있습니다.
pkginfo(4)	이제 영역을 지원하여 선택적 매개 변수와 환경 변수를 제공합니다.
proc(4)	영역에서 실행 중인 프로세스에 대한 정보를 가져오는 기능을 추가했습니다.
audit_syslog(5)	zonename 감사 정책을 설정한 경우에 사용되는 in<*zone name*> 필드를 추가했습니다.
privileges(5)	프로세스에서 다른 영역의 프로세스를 추적하거나 다른 영역의 프로세스에 신호를 보낼 수 있도록 PRIV_PROC_ZONE을 추가했습니다. zones (5)를 참조하십시오.
if_tcp(7P)	영역 ioctl() 호출을 추가했습니다.
cmn_err(9F)	영역 매개 변수를 추가했습니다.
ddi_cred(9F)	cr에 표시된 사용자 자격 증명에서 영역 ID를 반환하는 crgetzoneid()를 추가했습니다.

◆ ◆ ◆ 26 장

26 Oracle Solaris Zones 관리(작업)

이 장에서는 일반적인 관리 작업을 설명하고 사용 예를 보여 줍니다.

일반적인 영역 관리 항목은 25 장, “Oracle Solaris 영역 관리(개요)”를 참조하십시오.

ppriv 유틸리티 사용

ppriv 유틸리티를 사용하여 영역의 권한을 표시합니다.

▼ 전역 영역에서 Oracle Solaris 권한을 나열하는 방법

ppriv 유틸리티를 -l 옵션과 함께 사용하여 시스템에서 사용 가능한 권한을 나열합니다.

- **프롬프트에서 ppriv -l zone을 입력하여 영역에서 사용 가능한 권한 집합을 보고합니다.**

```
global# ppriv -l zone
```

다음과 유사한 내용이 표시됩니다.

```
contract_event
contract_observer
cpc_cpu
.
.
.
```

▼ 비전역 영역의 권한 집합을 나열하는 방법

ppriv 유틸리티를 -l 옵션 및 표현식 zone과 함께 사용하여 영역의 권한을 나열합니다.

1 **비전역 영역에 로그인합니다. 이 예에서는 *my-zone*이라는 영역을 사용합니다.**

2 **프롬프트에서 ppriv -l zone을 입력하여 영역에서 사용 가능한 권한 집합을 보고합니다.**

```
my-zone# ppriv -l zone
```

다음과 유사한 내용이 표시됩니다.

```
contract_event
contract_identity
contract_observer
file_chown

.
.
.
```

▼ 상세 정보 출력을 사용하여 비전역 영역의 권한 집합을 나열하는 방법

ppriv 유틸리티를 -l 옵션, 표현식 zone 및 -v와 함께 사용하여 영역의 권한을 나열합니다.

1 **비전역 영역에 로그인합니다. 이 예에서는 *my-zone*이라는 영역을 사용합니다.**

2 **프롬프트에서 ppriv - l -v zone을 입력하여 영역에서 사용 가능한 권한 집합과 각 권한에 대한 설명을 보고합니다.**

```
my-zone# ppriv -lv zone
```

다음과 유사한 내용이 표시됩니다.

```
contract_event
        Allows a process to request critical events without limitation.
        Allows a process to request reliable delivery of all events on
        any event queue.
```

```
contract_identity
        Allows a process to set the service FMRI value of a process
        contract template.
contract_observer
        Allows a process to observe contract events generated by
        contracts created and owned by users other than the process's
        effective user ID.
        Allows a process to open contract event endpoints belonging to
        contracts created and owned by users other than the process's
        effective user ID.
file_chown
        Allows a process to change a file's owner user ID.
        Allows a process to change a file's group ID to one other than
        the process' effective group ID or one of the process'
        supplemental group IDs.
.
.
.
```

비전역 영역에서 zonestat 유틸리티 사용

zonestat 유틸리티는 현재 실행 중인 영역의 CPU, 메모리, 네트워크 및 리소스 제어 사용률에 대해 보고합니다. 사용 예도 보여 줍니다.

자세한 내용은 zonestat(1)을 참조하십시오.

zonestat 네트워크 구성 요소는 영역의 PHYS, AGGR, Etherstub 및 SIMNET 데이터 링크에서 가상 네트워크(VNIC) 리소스의 사용을 보여 줍니다. dladm(1M) 및 dlstat(1M) 매뉴얼 페이지에서 설명하는 네트워킹 유틸리티를 사용하여 브릿지 및 터널과 같은 기타 데이터 링크에 대한 정보를 얻을 수 있습니다.

비전역 영역 내에서 모든 zonestat 옵션과 리소스 유형을 호출하여 해당 영역에 대한 통계를 표시할 수도 있습니다.

```
root@zoneA:~# zonestat -z global -r physical-memory 2
```

주 – 비전역 영역에서 zonestat를 사용하면 전역 영역을 포함하여 다른 모든 영역의 결합된 리소스 사용이 전역 영역에서 사용되는 것으로 보고됩니다. zonestat 의 비전역 영역 사용자는 시스템을 공유하는 다른 영역을 인식하지 않습니다.

▼ zonestat 유틸리티를 사용하여 CPU 및 메모리 사용률의 요약을 표시하는 방법

1 관리자로 전환합니다.

2 5초마다 CPU 및 메모리 사용률의 요약을 표시합니다.

```
# zonestat -z global -r physical-memory 5
Collecting data for first interval...
Interval: 1, Duration: 0:00:05
PHYSICAL-MEMORY              SYSTEM MEMORY
mem_default                          2046M
                                ZONE  USED %USED   CAP  %CAP
                             [total] 1020M 49.8%     -     -
                            [system]  782M 38.2%     -     -
                              global  185M 9.06%     -     -

Interval: 2, Duration: 0:00:10
PHYSICAL-MEMORY              SYSTEM MEMORY
mem_default                          2046M
                                ZONE  USED %USED   CAP  %CAP
                             [total] 1020M 49.8%     -     -
                            [system]  782M 38.2%     -     -
                              global  185M 9.06%     -     -
...
```

▼ zonestat 유틸리티를 사용하여 pset에 대해 보고하는 방법

1 관리자로 전환합니다.

2 1분 동안 1초에 한 번씩 기본 pset를 보고합니다.

```
# zonestat -r default-pset 1 1m
Collecting data for first interval...
Interval: 1, Duration: 0:00:01
PROCESSOR_SET                   TYPE  ONLINE/CPUS     MIN/MAX
pset_default            default-pset          2/2         1/-
                                ZONE  USED   PCT   CAP  %CAP   SHRS  %SHR %SHRU
                             [total]  0.02 1.10%     -     -      -     -     -
                            [system]  0.00 0.19%     -     -      -     -     -
                              global  0.01 0.77%     -     -      -     -     -
                                zone1 0.00 0.07%     -     -      -     -     -
                                zone2 0.00 0.06%     -     -      -     -     -

...
Interval: 60, Duration: 0:01:00
PROCESSOR_SET                   TYPE  ONLINE/CPUS     MIN/MAX
pset_default            default-pset          2/2         1/-
                                ZONE  USED   PCT   CAP  %CAP   SHRS  %SHR %SHRU
                             [total]  0.06 3.26%     -     -      -     -     -
```

```
[system]  0.00 0.18%     -     -     -     -     -
  global  0.05 2.94%     -     -     -     -     -
   zone1  0.00 0.06%     -     -     -     -     -
   zone2  0.00 0.06%     -     -     -     -     -
```

▼ zonestat를 사용하여 총사용률 및 고사용률 보고

1. **관리자로 전환합니다.**
2. **3분 동안 10초 간격으로 자동으로 모니터링한 다음 총사용률 및 고사용률에 대한 보고서를 생성합니다.**

```
# zonestat -q -R total,high 10s 3m 3m
Report: Total Usage
    Start: Fri Aug 26 07:32:22 PDT 2011
      End: Fri Aug 26 07:35:22 PDT 2011
    Intervals: 18, Duration: 0:03:00
SUMMARY                  Cpus/Online: 2/2   PhysMem: 2046M  VirtMem: 3069M
                    ---CPU----  --PhysMem-- --VirtMem-- --PhysNet--
                ZONE  USED %PART  USED %USED  USED %USED PBYTE %PUSE
             [total]  0.01 0.62% 1020M 49.8% 1305M 42.5%    14 0.00%
            [system]  0.00 0.23%  782M 38.2% 1061M 34.5%     -     -
              global  0.00 0.38%  185M 9.06%  208M 6.77%     0 0.00%
               test2  0.00 0.00% 52.4M 2.56% 36.6M 1.19%     0 0.00%

Report: High Usage
    Start: Fri Aug 26 07:32:22 PDT 2011
      End: Fri Aug 26 07:35:22 PDT 2011
    Intervals: 18, Duration: 0:03:00
SUMMARY                  Cpus/Online: 2/2   PhysMem: 2046M  VirtMem: 3069M
                    ---CPU----  --PhysMem-- --VirtMem-- --PhysNet--
                ZONE  USED %PART  USED %USED  USED %USED PBYTE %PUSE
             [total]  0.01 0.82% 1020M 49.8% 1305M 42.5%  2063 0.00%
            [system]  0.00 0.26%  782M 38.2% 1061M 34.5%     -     -
              global  0.01 0.55%  185M 9.06%  207M 6.77%     0 0.00%
               test2  0.00 0.00% 52.4M 2.56% 36.6M 1.19%     0 0.00%
```

▼ 배타적 IP 영역의 네트워크 대역폭 사용률을 얻는 방법

zonestat 명령을 -r 옵션 및 network 리소스 유형과 함께 사용하면 각 네트워크 장치의 영역별 사용률이 표시됩니다.

이 절차를 사용하여 각 영역에서 사용되는 VNIC 형식 데이터 링크 대역폭의 양을 봅니다. 예를 들어, e1000g0 아래 표시된 zoneB는 이 영역이 VNIC 형식으로 e1000g0의 리소스를 소비함을 표시합니다. -x 옵션을 추가하여 특정 VNIC를 표시할 수도 있습니다.

1. **루트 관리자가 됩니다.**

2 **network 리소스 유형을 -r option과 함께 zonestat 명령에 사용하여 사용률을 한 번에 표시합니다.**

```
    # zonestat -r network 1 1
Collecting data for first interval...
Interval: 1, Duration: 0:00:01

    NETWORK-DEVICE                 SPEED        STATE        TYPE
    aggr1                        2000mbps           up        AGGR
                   ZONE TOBYTE  MAXBW %MAXBW PRBYTE %PRBYTE POBYTE %POBYTE
                 global  1196K      -      -   710K   0.28%   438K   0.18%

    e1000g0                      1000mbps           up        PHYS
                   ZONE TOBYTE  MAXBW %MAXBW PRBYTE %PRBYTE POBYTE %POBYTE
                [total]  7672K      -      -  6112K   4.89%  1756K   1.40%
                 global  5344K  100m*  42.6%  2414K   1.93%  1616K   1.40%
                  zoneB   992K   100m  15.8%  1336K   0.76%   140K   0.13%
                  zoneA  1336K    50m  10.6%   950K   1.07%      0   0.00%

    e1000g1                      1000mbps           up        PHYS
                   ZONE TOBYTE  MAXBW %MAXBW PRBYTE %PRBYTE POBYTE %POBYTE
                 global   126M      -      -    63M   6.30%    63M   6.30%

    etherstub1                       n/a          n/a   ETHERSTUB
                   ZONE TOBYTE  MAXBW %MAXBW PRBYTE %PRBYTE POBYTE %POBYTE
                [total]  3920K      -      -      0       -      0       -
                 global  1960K  100M*  1.96%      0       -      0       -
                  zoneA  1960K    50M  3.92%      0       -      0       -
```

자세한 정보 **비전역 영역의 명령 예**

비전역 영역에 사용되는 명령:

```
root@zoneA:~# zonestat -r network -x 1 1
```

비전역 영역에서 DTrace 사용

342 페이지 “비전역 영역에서 DTrace 실행”의 설명과 같이 다음 단계를 수행하여 DTrace 기능을 사용합니다.

▼ DTrace를 사용하는 방법

1 **zonecfg limitpriv 등록 정보를 사용하여 dtrace_proc 및 dtrace_user 권한을 추가합니다.**

```
global# zonecfg -z my-zone
zonecfg:my-zone> set limitpriv="default,dtrace_proc,dtrace_user"
zonecfg:my-zone> exit
```

주 – 요구 사항에 따라 두 권한 중 하나를 추가하거나 두 권한을 모두 추가할 수 있습니다.

2 **영역을 부트합니다.**

```
global# zoneadm -z my-zone boot
```

3 **영역에 로그인합니다.**

```
global# zlogin my-zone
```

4 **DTrace 프로그램을 실행합니다.**

```
my-zone# dtrace -l
```

비전역 영역에서 SMF 서비스의 상태 확인

비전역 영역에서 SMF 서비스의 상태를 확인하려면 `zlogin` 명령을 사용합니다.

▼ 명령줄에서 SMF 서비스의 상태를 확인하는 방법

1 **관리자로 전환합니다.**

2 **명령줄에서 다음을 입력하여 사용 안함으로 설정된 서비스를 포함한 모든 서비스를 표시합니다.**

```
global# zlogin my-zone svcs -a
```

참조 자세한 내용은 21 장, “비전역 영역에 로그인(작업)” 및 svcs(1)을 참조하십시오.

▼ 영역 내에서 SMF 서비스의 상태를 확인하는 방법

1 **관리자로 전환합니다.**

2 **영역에 로그인합니다.**

```
global# zlogin my-zone
```

3 **svcs 명령을 -a 옵션과 함께 실행하여 사용 안함으로 설정된 서비스를 포함한 모든 서비스를 표시합니다.**

```
my-zone# svcs -a
```

참조 자세한 내용은 21 장, “비전역 영역에 로그인(작업)” 및 svcs(1)을 참조하십시오.

실행 중인 비전역 영역에서 파일 시스템 마운트

실행 중인 비전역 영역에서 파일 시스템을 마운트할 수 있습니다. 다음 절차가 포함됩니다.

- 전역 관리자 또는 전역 영역에서 해당 권한이 부여된 사용자는 원시 및 블록 장치를 비전역 영역으로 가져올 수 있습니다. 장치를 가져온 후 영역 관리자는 디스크에 액세스할 수 있습니다. 그런 다음 영역 관리자는 디스크에 새 파일 시스템을 만들고 다음 작업 중 하나를 수행할 수 있습니다.
  - 수동으로 파일 시스템 마운트
  - 영역 부트 시 마운트되도록 `/etc/vfstab`에 파일 시스템 배치
- 전역 관리자 또는 해당 권한이 부여된 사용자는 전역 영역에서 비전역 영역으로 파일 시스템을 마운트할 수도 있습니다.

  전역 영역에서 비전역 영역으로 파일 시스템을 마운트하기 전에 비전역 영역이 준비 상태에 있거나 부트되어야 합니다. 그렇지 않으면 다음에 영역을 준비하거나 부트하는 시도가 실패하게 됩니다. 또한 전역 영역에서 비전역 영역으로 마운트된 파일 시스템은 영역이 정지되기 전에 마운트 해제됩니다.

▼ LOFS를 사용하여 파일 시스템을 마운트하는 방법

LOFS 마운트를 사용하여 전역 영역과 비전역 영역 간에 파일 시스템을 공유할 수 있습니다. 이 프로시저는 `zonecfg` 명령을 사용하여 전역 영역 `/export/datafiles` 파일 시스템의 LOFS mount를 `my-zone` 구성에 추가합니다. 이 예에서는 마운트 옵션을 사용자 정의하지 않습니다.

이 절차를 수행하려면 전역 관리자 또는 전역 영역에서 영역 보안 권한 프로파일이 있는 사용자여야 합니다.

1 관리자로 전환합니다.

2 `zonecfg` 명령을 사용합니다.

```
global# zonecfg -z my-zone
```

3 파일 시스템을 구성에 추가합니다.

```
zonecfg:my-zone> add fs
```

4 `my-zone`에서 `/datafiles` 파일 시스템에 대한 마운트 지점을 설정합니다.

```
zonecfg:my-zone:fs> set dir=/datafiles
```

5 전역 영역의 `/export/datafiles`가 `my-zone`에서 `/datafiles`로 마운트되도록 지정합니다.

```
zonecfg:my-zone:fs> set special=/export/datafiles
```

6 **파일 시스템 유형을 설정합니다.**

```
zonecfg:my-zone:fs> set type=lofs
```

7 **지정을 종료합니다.**

```
zonecfg:my-zone:fs> end
```

8 **구성을 확인하고 완결합니다.**

```
zonecfg:my-zone> verify
zonecfg:my-zone> commit
```

자세한 정보 **임시 마운트**

비전역 영역을 재부트하지 않고 전역 영역의 LOFS 파일 시스템 마운트를 추가할 수 있습니다.

```
global# mount -F lofs /export/datafiles /export/my-zone/root/datafiles
```

영역을 부트할 때마다 이 마운트를 수행하려면 `zonecfg` 명령을 사용하여 영역의 구성을 수정해야 합니다.

▼ ZFS 데이터 집합을 비전역 영역에 위임하는 방법

이 절차를 사용하여 ZFS 데이터 집합을 비전역 영역에 위임합니다.

이 절차를 수행하려면 전역 관리자 또는 전역 영역에서 해당 권한이 부여된 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **전역 영역에서 `poolA`라는 기존 ZFS 풀에 `fs2`라는 새 ZFS 파일 시스템을 만듭니다.**

```
global# zfs create poolA/fs2
```

3 **(옵션) `poolA/fs2` 파일 시스템에 대한 `mountpoint` 등록 정보를 `/fs-del/fs2`로 설정합니다.**

```
global# zfs set mountpoint=/fs-del/fs2 poolA/fs2
```

`mountpoint`를 설정할 필요는 없습니다. `mountpoint` 등록 정보를 지정하지 않으면 데이터 세트는 기본적으로 영역 내에서 `/alias`에 마운트됩니다. `mountpoint` 및 `canmount` 등록 정보에 대한 기본값이 아닌 값은 `zfs`(1M) 매뉴얼 페이지에 설명된 대로 이 동작을 변경합니다.

4 **이 파일 시스템에 대한 `mountpoint` 등록 정보의 소스가 이제 `local`인지 확인합니다.**

```
global# zfs get mountpoint poolA/fs2
NAME        PROPERTY    VALUE         SOURCE
poolA/fs2   mountpoint  /fs-del/fs2   local
```

5 **poolA/fs2 파일 시스템을 위임하거나 가칭된 데이터 집합을 지정합니다.**

- **poolA/fs2 파일 시스템을 영역에 위임합니다.**

```
# zonecfg -z my-zone
zonecfg:my-zone> add dataset
zonecfg:my-zone:dataset> set name=poolA/fs2
zonecfg:my-zone:dataset> end
```

- **가칭된 데이터 집합을 지정합니다.**

```
# zonecfg -z my-zone
zonecfg:my-zone> add dataset
zonecfg:my-zone:dataset> set name=poolA/fs2
zonecfg:my-zone:dataset> set alias=delegated
zonecfg:my-zone:dataset> end
```

6 **영역을 재부트하고 모든 poolA 파일 시스템에 대한 zoned 등록 정보를 표시합니다.**

```
global# zfs get -r zoned poolA
NAME      PROPERTY  VALUE   SOURCE
poolA     zoned     off     default
poolA/fs2 zoned     on      default
```

poolA/fs2에 대한 zoned 등록 정보는 on으로 설정됩니다. 이 ZFS 파일 시스템은 비전역 영역에 위임되었고 영역에서 마운트되었으며 영역 관리자의 제어를 받고 있습니다. ZFS는 zoned 등록 정보를 사용하여 한 시점에서 데이터 집합이 비전역 영역으로 위임되었음을 표시합니다.

전역 영역의 특정 파일 시스템에 비전역 영역 액세스 추가

▼ 비전역 영역에서 CD 또는 DVD 매체에 대한 액세스를 추가하는 방법

이 프로시저를 사용하여 비전역 영역에서 CD 또는 DVD 매체에 대한 읽기 전용 액세스를 추가할 수 있습니다. Volume Management 파일 시스템은 전역 영역에서 매체를 장착하는 데 사용됩니다. 그러면 CD 또는 DVD를 사용하여 비전역 영역에서 제품을 설치할 수 있습니다. 이 절차에서는 jes_05q4_dvd라는 DVD를 사용합니다.

1 **관리자로 전환합니다.**

2 **Volume Management 파일 시스템이 전역 영역에서 실행 중인지 여부를 확인합니다.**

```
global# svcs volfs
STATE          STIME    FMRI
online         Sep_29   svc:/system/filesystem/volfs:default
```

3 **(옵션) Volume Management 파일 시스템이 전역 영역에서 실행 중이 아닌 경우 이 파일 시스템을 시작합니다.**

```
global# svcadm volfs enable
```

4 **매체를 삽입합니다.**

5 **드라이브에서 매체를 검사합니다.**

```
global# volcheck
```

6 **DVD가 자동 마운트되는지 여부를 테스트합니다.**

```
global# ls /cdrom
```

다음과 유사하게 표시됩니다.

```
cdrom   cdrom1   jes_05q4_dvd
```

7 **비전역 영역에서 ro,nodevices (읽기 전용 및 장치 없음) 옵션을 사용하여 파일 시스템을 루프백 마운트합니다.**

```
global# zonecfg -z my-zone
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/cdrom
zonecfg:my-zone:fs> set special=/cdrom
zonecfg:my-zone:fs> set type=lofs
zonecfg:my-zone:fs> add options [ro,nodevices]
zonecfg:my-zone:fs> end
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

8 **비전역 영역을 재부트합니다.**

```
global# zoneadm -z my-zone reboot
```

9 **zoneadm list 명령을 -v 옵션과 함께 사용하여 상태를 확인합니다.**

```
global# zoneadm list -v
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris   shared
 1  my-zone  running      /zones/my-zone                 solaris   excl
```

10 **비전역 영역에 로그인합니다.**

```
global#  my-zone
```

11 **DVD-ROM 마운트를 확인합니다.**

```
my-zone# ls /cdrom
```

다음과 유사한 내용이 표시됩니다.

```
cdrom   cdrom1   jes_05q4_dvd
```

12 **제품 설치 설명서의 설명과 같이 제품을 설치합니다.**

13 **비전역 영역을 종료합니다.**

```
my-zone# exit
```

참고 – 비전역 영역에서 /cdrom 파일 시스템을 유지해야 할 수 있습니다. 마운트는 항상 CD-ROM 드라이브의 현재 내용을 반영하거나 드라이브가 비어 있는 경우 빈 디렉토리를 반영합니다.

---

14 **(옵션) 비전역 영역에서 /cdrom 파일 시스템을 제거하려는 경우 다음 절차를 사용합니다.**

```
global# zonecfg -z my-zone
zonecfg:my-zone> remove fs dir=/cdrom
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

영역이 설치된 Oracle Solaris 시스템에 IP 네트워크 다중 경로 사용

▼ 배타적 IP 비전역 영역에서 IP 네트워크 다중 경로를 사용하는 방법

배타적 IP 영역의 IPMP(IP 네트워크 다중 경로)는 전역 영역과 같이 구성됩니다. IPMP를 사용하려면 배타적 IP 영역에 최소 두 개의 zonecfg add net 리소스가 있어야 합니다. IPMP는 이러한 데이터 링크의 영역 내에서 구성됩니다.

하나 이상의 물리적 인터페이스를 IP 다중 경로 그룹 또는 IPMP 그룹으로 구성할 수 있습니다. IPMP를 구성한 후 시스템은 IPMP 그룹에서 인터페이스의 실패를 자동으로 모니터링합니다. 그룹의 인터페이스가 실패하거나 유지 관리를 위해 제거된 경우 IPMP는 실패한 인터페이스의 IP 주소를 자동으로 마이그레이션하거나 페일오버합니다. 이러한 주소의 수신자는 실패한 인터페이스의 IPMP 그룹에서 작동하는 인터페이스입니다. IPMP의 페일오버 구성 요소는 연결을 유지하고 기존 연결의 중단을 방지합니다. 또한 IPMP는 네트워크 트래픽을 IPMP 그룹의 인터페이스 집합에 자동으로 분산하여 전체 네트워크 성능을 향상합니다. 이 프로세스를 부하 분산이라고 합니다.

1 **관리자로 전환합니다.**

2 **Oracle Solaris 관리: 네트워크 인터페이스 및 네트워크 가상화의 "IPMP 그룹 구성"의 설명과 같이 IPMP 그룹을 구성합니다.**

▼ IP 네트워크 다중 경로 기능을 공유 IP 비전역 영역으로 확장하는 방법

이 절차를 사용하여 전역 영역에서 IPMP를 구성하고 IPMP 기능을 비전역 영역으로 확장합니다.

영역을 구성할 때 각 주소 또는 논리적 인터페이스는 비전역 영역과 연관되어야 합니다. 자세한 지침은 219 페이지 “zonecfg 명령 사용” 및 243 페이지 “영역 구성 방법”을 참조하십시오.

이 절차에서는 다음을 수행합니다.

- `bge0` 및 `hme0` 카드가 그룹에서 함께 구성됩니다.
- 주소 192.168.0.1은 비전역 영역 *my-zone*과 연관됩니다.
- `bge0` 카드가 물리적 인터페이스로 설정됩니다. 따라서 IP 주소는 `bge0` 및 `hme0` 카드를 포함하는 그룹에 호스팅됩니다.

실행 중인 영역에서 `ipadm` 명령을 사용하여 연관을 만들 수 있습니다. 자세한 내용은 330 페이지 “공유 IP 네트워크 인터페이스” 및 `ipadm(1M)` 매뉴얼 페이지를 참조하십시오.

이 절차를 수행하려면 전역 관리자 또는 전역 영역에서 해당 권한이 부여된 사용자여야 합니다.

1. **관리자로 전환합니다.**
2. **전역 영역에서 Oracle Solaris 관리: 네트워크 인터페이스 및 네트워크 가상화의 “IPMP 그룹 구성”의 설명과 같이 IPMP 그룹을 구성합니다.**
3. **`zonecfg` 명령을 사용하여 영역을 구성합니다. `net` 리소스를 구성할 때 주소 `192.168.0.1`과 물리적 인터페이스 `bge0`를 영역 *my-zone*에 추가합니다.**

```
zonecfg:my-zone> add net
zonecfg:my-zone:net> set address=192.168.0.1
zonecfg:my-zone:net> set physical=bge0
zonecfg:my-zone:net> end
```

비전역 영역 *my-zone*에는 `bge0`만 표시됩니다.

자세한 정보 bge0가 이후에 실패할 경우

`bge0`가 이후에 실패하고 `bge0` 데이터 주소가 전역 영역의 `hme0`에 페일오버되는 경우 *my-zone* 주소도 마이그레이션됩니다.

주소 `192.168.0.1`이 `hme0`로 이동하는 경우 비전역 영역 *my-zone*에 `hme0`만 표시됩니다. 이 카드는 주소 `192.168.0.1` 와 연관되며 `bge0`는 더 이상 표시되지 않습니다.

배타적 IP 비전역 영역에서 데이터 링크 관리

전역 영역에서 dladm 명령을 사용하여 데이터 링크를 관리합니다.

▼ dladm show-linkprop를 사용하는 방법

dladm 명령을 show-linkprop 하위 명령과 함께 사용하여 실행 중인 배타적 IP 영역에 데이터 링크의 할당을 표시할 수 있습니다.

데이터 링크를 관리하려면 전역 관리자 또는 전역 영역에서 해당 권한이 부여된 사용자여야 합니다.

1 관리자로 전환합니다.

2 시스템에서 데이터 링크의 할당을 표시합니다.

```
global# dladm show-linkprop
```

예 26–1 show-linkprop 하위 명령과 함께 dladm 사용

1. 첫 화면에서 bge0이 지정된 49bge 영역이 부트되지 않았습니다.

```
global# dladm show-linkprop
LINK     PROPERTY        PERM   VALUE        DEFAULT        POSSIBLE
bge0     zone            rw     --           --             --
vsw0     speed           r-     1000         1000           --
vsw0     autopush        rw     --           --             --
vsw0     zone            rw     --           --             --
vsw0     duplex          r-     full         full           half,full
vsw0     state           r-     up           up             up,down
vsw0     adv_autoneg_cap --     --           0              1,0
vsw0     mtu             rw     1500         1500           1500
vsw0     flowctrl        --     --           no             no,tx,rx,bi,pfc,
                                                            auto
...
```

2. 49bge 영역을 부트합니다.

```
global# zoneadm -z 49bge boot
```

3. dladm show-linkprop 명령을 다시 실행합니다. bge0 링크가 이제 49bge에 지정됩니다.

```
global# dladm show-linkprop
LINK     PROPERTY        PERM   VALUE        DEFAULT        POSSIBLE
bge0     zone            rw     49bge        --             --
vsw0     speed           r-     1000         1000           --
vsw0     autopush        rw     --           --             --
vsw0     zone            rw     --           --             --
vsw0     duplex          r-     full         full           half,full
vsw0     state           r-     up           up             up,down
vsw0     adv_autoneg_cap --     --           0              1,0
vsw0     mtu             rw     1500         1500           1500
vsw0     flowctrl        --     --           no             no,tx,rx,bi,pfc,
                                                            auto
...
```

예 26–2 배니티 이름 지정을 사용할 때 데이터 링크 이름과 물리적 위치를 표시하는 방법

장치의 물리적인 위치는 LOCATION 필드에 표시됩니다. 장치에 대한 데이터 링크 이름과 물리적 위치 정보를 보려면 -L 옵션을 사용합니다.

```
global# dladm show-phys -L
LINK         DEVICE       LOCATION
net0         e1000g0      MB
net1         e1000g1      MB
net2         e1000g2      MB
net3         e1000g3      MB
net4         ibp0         MB/RISER0/PCIE0/PORT1
net5         ibp1         MB/RISER0/PCIE0/PORT2
net6         eoib2        MB/RISER0/PCIE0/PORT1/cloud-nm2gw-2/1A-ETH-2
net7         eoib4        MB/RISER0/PCIE0/PORT2/cloud-nm2gw-2/1A-ETH-2
```

▼ dladm을 사용하여 임시 데이터 링크를 지정하는 방법

dladm 명령을 set-linkprop 하위 명령과 함께 사용하여 실행 중인 배타적 IP 영역에 데이터 링크를 임시로 지정할 수 있습니다. zonecfg 명령을 통해 지속적으로 할당해야 합니다.

데이터 링크를 관리하려면 전역 관리자 또는 전역 영역에서 해당 권한이 부여된 사용자여야 합니다.

1 관리자로 전환합니다.

2 dladm set-linkprop를 -t와 함께 사용하여 zoneA라는 실행 중인 영역에 bge0를 추가합니다.

```
global# dladm set-linkprop -t -p zone bge0
LINK     PROPERTY        PERM   VALUE        DEFAULT        POSSIBLE
bge0     zone            rw     zoneA        --             --
```

참고 – -p 옵션은 시스템에서 구문 분석할 수 있는 안정적인 형식을 사용하여 표시를 생성합니다.

▼ dladm reset-linkprop를 사용하는 방법

dladm 명령을 reset-linkprop 하위 명령과 함께 사용하여 bge0 링크 값을 할당 안 됨으로 재설정할 수 있습니다.

1 관리자로 전환합니다.

2 **dladm reset-linkprop을 -t 옵션과 함께 사용하여 bge0 장치의 영역 할당을 실행 취소합니다.**

```
global# dladm reset-linkprop -t -p zone bge0
LINK     PROPERTY        PERM    VALUE        DEFAULT        POSSIBLE
bge0     zone            rw      zoneA        --             --
```

참고 – -p 옵션은 시스템에서 구문 분석할 수 있는 안정적인 형식을 사용하여 표시를 생성합니다.

일반 오류 실행 중인 영역이 장치를 사용하고 있는 경우 재할당이 실패하고 오류 메시지가 표시됩니다. 375 페이지 "배타적 IP 영역에서 장치를 사용하고 있어서 dladm reset-linkprop 실패"를 참조하십시오.

영역이 설치된 Oracle Solaris 시스템에서 Fair Share Scheduler 사용

prctl 명령을 통해 지정된 제한은 지속적이지 않습니다. 시스템을 재부트할 때까지만 제한이 적용됩니다. 영역에 할당을 영구적으로 설정하려면 243 페이지 "영역 구성 방법" 및 255 페이지 "전역 영역의 zone.cpu-shares를 설정하는 방법"을 참조하십시오.

▼ prctl 명령을 사용하여 전역 영역에서 FSS 할당을 설정하는 방법

전역 영역에는 기본적으로 할당 수 1이 지정됩니다. 이 절차를 사용하여 기본 할당을 변경할 수 있습니다. 시스템을 재부트할 때마다 prctl 명령을 통해 할당된 할당을 재설정해야 합니다.

이 절차를 수행하려면 전역 관리자 또는 전역 영역에서 해당 권한이 부여된 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 **prctl 유틸리티를 사용하여 할당 수 2를 전역 영역에 지정합니다.**

```
# prctl -n zone.cpu-shares -v 2 -r -i zone global
```

3 **(선택적) 전역 영역에 지정된 할당 수를 확인하려면 다음을 입력합니다.**

```
# prctl -n zone.cpu-shares -i zone global
```

참조 prctl 유틸리티에 대한 자세한 내용은 prctl(1) 매뉴얼 페이지를 참조하십시오.

▼ 영역에서 zone.cpu-shares 값을 동적으로 변경하는 방법

전역 영역 또는 비전역 영역에서 이 절차를 사용할 수 있습니다.

1 **관리자로 전환합니다.**

2 **`prctl` 명령을 사용하여 `cpu-shares`의 새 값을 지정합니다.**

```
# prctl  -n zone.cpu-shares -r -v value -i zone zonename
```

*idtype*는 *zonename* 또는 *zoneid*입니다. *value*는 새 값입니다.

영역 관리에서 권한 프로파일 사용

이 절에서는 비전역 영역에서 권한 프로파일 사용과 연관된 작업에 대해 설명합니다.

▼ 영역 관리 프로파일을 지정하는 방법

영역 관리 프로파일은 시스템에서 모든 비전역 영역을 관리할 수 있는 권한을 사용자에게 부여합니다.

이 절차를 수행하려면 전역 관리자 또는 전역 영역에서 해당 권한이 부여된 사용자여야 합니다.

1 **수퍼유저가 되거나 동등한 권한을 갖습니다.**
역할에 대한 자세한 내용은 **Oracle Solaris 관리: 보안 서비스**의 "RBAC 초기 구성(작업 맵)"을 참조하십시오.

2 **영역 관리 권한 프로파일이 포함된 역할을 만들고 이 역할을 사용자에게 지정합니다.**

영역이 설치된 Oracle Solaris 시스템 백업

다음 절차를 사용하여 영역의 파일을 백업할 수 있습니다. 영역의 구성 파일도 백업해야 합니다.

▼ ZFSsend를 사용하여 백업을 수행하는 방법

1 **관리자로 전환합니다.**

2 **영역의 zonepath를 얻습니다.**

```
global# zonecfg -z my-zone info zonepath
zonepath: /zones/my-zone
```

3 **zfs list 명령을 사용하여 zonepath 데이터 집합을 얻습니다.**

```
global# zfs list -H -o name /zones/my-zone
rpool/zones/my-zone
```

4 **ZFS 스냅샷을 사용하여 영역의 아카이브를 만듭니다.**

```
global# zfs  snapshot -r rpool/zones/my-zone@snap
global# zfs snapshot -r rpool/zones/my-zone@snap
global# zfs zfs send -rc rpool/zones/my-zone@snap > /path/to/save/archive
global# zfs destroy -r rpool/zones/my-zone@snap
```

다음과 유사하게 표시됩니다.

```
-rwxr-xr-x   1 root     root     99680256 Aug 10 16:13 backup/my-zone.cpio
```

▼ 영역 구성의 복사본을 인쇄하는 방법

비전역 영역 구성의 백업 파일을 만들어야 합니다. 필요한 경우 나중에 백업을 사용하여 영역을 다시 만들 수 있습니다. 영역에 처음 로그인하고 `sysidtool` 질문에 응답한 후 영역 구성의 복사본을 만듭니다. 이 절차에서는 `my-zone`이라는 영역과 `my-zone.config`라는 백업 파일을 사용하여 프로세스를 설명합니다.

1 **관리자로 전환합니다.**

2 **my-zone 영역에 대한 구성을 my-zone.config라는 파일로 인쇄합니다.**

```
global# zonecfg -z my-zone export > my-zone.config
```

비전역 영역 다시 만들기

▼ 개별 비전역 영역을 다시 만드는 방법

필요한 경우 비전역 영역 구성의 백업 파일을 사용하여 비전역 영역을 다시 만들 수 있습니다. 이 절차에서는 `my-zone`이라는 영역과 `my-zone.config`라는 백업 파일을 사용하여 영역 다시 만들기 프로세스를 설명합니다.

1 **관리자로 전환합니다.**

2 **my-zone.config가 my-zone 영역을 다시 만들기 위한 zonecfg 명령 파일로 사용되도록 지정합니다.**

```
global# zonecfg -z my-zone -f my-zone.config
```

3 **영역을 설치합니다.**

```
global# zoneadm -z my-zone install -a /path/to/archive options
```

4 **응용 프로그램 데이터와 같은 영역별 파일을 복원해야 하는 경우 백업에서 새로 만들어진 영역의 루트 파일 시스템으로 수동으로 복원(및 가능한 경우 수동 병합)합니다.**

◆ ◆ ◆ 27 장

27 변경할 수 없는 영역 구성 및 관리

변경할 수 없는 영역은 solaris 비전역 영역에 읽기 전용 파일 시스템 프로파일을 제공합니다.

읽기 전용 영역 개요

읽기 전용 영역 루트가 있는 영역을 변경할 수 없는 영역이라고 합니다. 변경할 수 없는 solaris 영역은 비전역 영역에 대한 root 파일 시스템을 구현하여 영역의 구성을 보존합니다. 이 영역은 추가 제한을 런타임 환경에 추가하여 영역 보안 런타임 경계를 확장합니다. 특정 유지 관리 작업으로 수행되는 경우가 아니면 시스템 경계나 시스템 구성에 대한 수정은 차단되어 있습니다.

MWAC(필수 쓰기 액세스 제어) 커널 정책은 zonecfg file-mac-profile 등록 정보를 통해 파일 시스템 쓰기 권한을 강제 적용하는 데 사용됩니다. 전역 영역에는 MWAC 정책이 적용되지 않으므로 전역 영역은 설치, 이미지 업데이트 및 유지 관리를 위해 비전역 영역의 파일 시스템에 쓸 수 있습니다.

MWAC 정책은 영역이 준비 상태로 전환될 때 다운로드됩니다. 이 정책은 영역 부트 시 사용으로 설정됩니다. 설치 후 어셈블리 및 구성을 수행하려면 임시 쓰기 가능한 루트 파일 시스템 부트 시퀀스가 사용됩니다. 영역의 MWAC 구성에 대한 수정 사항은 영역을 재부트해야만 적용됩니다.

영역 구성, 설치 및 부트에 대한 일반 정보는 17 장, “비전역 영역 계획 및 구성(작업)” 및 19 장, “비전역 영역 설치, 부트, 종료, 정지, 제거 및 복제(작업)”를 참조하십시오.

읽기 전용 영역 구성

zonecfg file-mac-profile 등록 정보

기본적으로 zonecfg file-mac-profile 등록 정보는 비전역 영역에서는 설정되지 않습니다. 영역은 쓰기 가능한 루트 데이터 집합을 갖도록 구성됩니다.

solaris 읽기 전용 영역에서는 읽기 전용 영역 루트를 구성하는 데 file-mac-profile 등록 정보가 사용됩니다. 읽기 전용 루트는 영역 내부에서 런타임 환경에 대한 액세스를 제한합니다.

zonecfg 유틸리티를 통해 file-mac-profile 을 다음 값 중 하나로 설정할 수 있습니다. none을 제외한 모든 프로파일은 /var/pkg 디렉토리와 그 내용을 영역 내부에서 읽기 전용으로 설정합니다.

none	기존 영역 경계를 초과하는 추가 보호가 없는 표준 읽기-쓰기 비전역 영역입니다. 값을 none으로 설정하면 file-mac-profile 등록 정보를 설정하지 않은 것과 같습니다.
strict	읽기 전용 파일 시스템, 예외가 없습니다. ■ IPS 패키지를 설치할 수 없습니다. ■ 영구적으로 사용 가능한 SMF 서비스가 고정되어 있습니다. ■ SMF 매니페스트를 기본 위치에서 추가할 수 없습니다. ■ 구성 파일 기록 및 감사가 고정되어 있습니다. 데이터를 원격으로만 기록할 수 있습니다.
fixed-configuration	/var/* 디렉토리를 업데이트할 수 있습니다. 단, 시스템 구성의 구성 요소를 포함하는 디렉토리는 제외됩니다. ■ IPS 패키지(새 패키지 포함)를 설치할 수 있습니다. ■ 영구적으로 사용 가능한 SMF 서비스가 고정되어 있습니다. ■ SMF 매니페스트를 기본 위치에서 추가할 수 없습니다. ■ 구성 파일 기록 및 감사가 지역적일 수 있습니다. syslog 및 감사 구성이 고정되어 있습니다.
flexible-configuration	/etc/ * 디렉토리에 있는 파일을 수정할 수 있으며 루트의 홈 디렉토리를 변경할 수 있고 /var/* 디렉토리를 업데이트할 수 있습니다. 이 구성은 **System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones** 에 설명된 Oracle Solaris 10 native 희소

루트 영역과 가장 비슷한 기능을 제공합니다. 이 설명서는 Oracle Solaris 10 버전의 설명서입니다.

- IPS 패키지(새 패키지 포함)를 설치할 수 있습니다.
- 영구적으로 사용 가능한 SMF 서비스가 고정되어 있습니다.
- SMF 매니페스트를 기본 위치에서 추가할 수 없습니다.
- 구성 파일 기록 및 감사가 지역적일 수 있습니다. `syslog` 및 감사 구성을 변경할 수 있습니다.

zonecfg add dataset 리소스 정책

`add dataset` 리소스를 통해 영역에 추가된 데이터 집합에는 MWAC 정책이 적용되지 않습니다. 추가 데이터 집합이 위임된 영역에는 이러한 데이터 집합에 대한 모든 권한이 부여됩니다. 플랫폼 데이터 집합을 볼 수 있지만, 관련 데이터와 등록 정보는 영역이 읽기/쓰기로 부트된 경우가 아니면 읽기 전용입니다.

zonecfg add fs 리소스 정책

`add fs` 리소스를 통해 영역에 추가된 파일 시스템에는 MWAC 정책이 적용되지 않습니다. 파일 시스템을 읽기 전용으로 마운트할 수 있습니다.

읽기 전용 영역 관리

디스크 내장 구성은 전역 영역에서만 관리할 수 있습니다. 실행 중인 영역 내에서는 영역이 쓰기 가능하도록 부트된 경우가 아니면 관리가 런타임 상태 설정으로 제한됩니다. 따라서 `svcadm`(1M) 및 `svccfg`(1M) 매뉴얼 페이지에 설명된 SMF 명령을 통한 구성 변경 내용이 디스크 내장 SMF 데이터베이스가 아니라 실시간 임시 SMF 데이터베이스에만 적용될 수 있습니다. 영역의 MWAC 구성에 수정 사항은 영역이 재부트될 때 적용됩니다.

초기 설치 시 또는 업데이트 후에는 `self-assembly-complete` 마일스톤에 도달할 때까지 영역이 임시 읽기-쓰기로 부트됩니다. 그런 다음 영역이 읽기 전용 모드로 재부트됩니다.

zoneadm list -p 표시

구문 분석 가능한 출력에는 R/W 행과 `file-mac-profile` 열이 표시됩니다.

```
global# zoneadm list -p
0:global:running:/:UUID:solaris:shared:-:none
    5:testzone2:running:/export/zones/testzone2:UUID \
        :solaris:shared:R:fixed-configuration
    12:testzone3:running:/export/zones/testzone3:UUID \
        :solaris:shared:R:fixed-configuration
    13:testzone1:running:/export/zones/testzone1:UUID \
        :solaris:excl:W:fixed-configuration
    -:testzone:installed:/export/zones/testzone:UUID \
        :solaris:excl:-:fixed-configuration
```

R 및 W 옵션이 다음과 같이 정의됩니다.

- R은 읽기 전용으로 부트되는 file-mac-profile이 있는 영역을 나타냅니다.
- W는 읽기-쓰기로 부트되는 file-mac-profile이 있는 영역을 나타냅니다.
- –는 실행되고 있지 않거나 file-mac-profile이 없는 영역입니다.

쓰기 가능한 루트 파일 시스템으로 읽기 전용 영역을 부트하기 위한 옵션

zoneadm boot 하위 명령은 전역 영역 관리자가 쓰기 가능 루트 파일 시스템이나 임시 쓰기 가능 루트 파일 시스템으로 읽기 전용 영역을 수동으로 부트할 수 있도록 하는 두 가지 옵션을 제공합니다. 다음 번에 재부트할 때까지 영역이 쓰기 가능 모드입니다.

-w 쓰기 가능 root 파일 시스템으로 영역을 직접 부트합니다.

-W 임시 쓰기 가능 root 파일 시스템으로 영역을 직접 부트합니다. self-assembly-complete 마일스톤에 도달할 때 시스템이 자동으로 재부트됩니다.

재부트하면 영역에 MWAC 정책이 다시 적용됩니다. 이 옵션은 영역의 MWAC 정책이 none일 때 허용됩니다.

비 ROZR 영역의 경우 -W 및 -w 옵션이 둘 다 무시됩니다.

◆ ◆ ◆ 28장

28 그 밖의 기타 Oracle Solaris Zones 문제 해결

이 장에는 영역 문제 해결에 대한 정보가 수록되어 있습니다.

배타적 IP 영역에서 장치를 사용하고 있어서 dladm reset-linkprop 실패

다음 오류 메시지가 표시됩니다.

```
dladm: warning: cannot reset link property 'zone' on 'bge0': operation failed
```

365 페이지 "dladm reset-linkprop를 사용하는 방법"을 참조하십시오. dladm reset-linkprop을 사용하려는 시도가 실패했습니다. 실행 중인 영역 excl에서 장치를 사용 중입니다.

값을 재설정하려면 다음과 같이 합니다.

1.
```
global# ipadm delete-ip bge0
```
2. dladm 명령을 다시 실행합니다.

영역 구성에 지정된 잘못된 권한 집합

영역의 권한 집합에 허용되지 않은 권한이 들어 있거나 필요한 권한이 없거나, 알 수 없는 권한이 포함된 경우, 영역을 확인, 준비 또는 부트하려는 시도가 실패하며 다음과 같은 오류 메시지가 표시됩니다.

```
zonecfg:zone5> set limitpriv="basic"
.
.
.
```

```
global# zoneadm -z zone5 boot
     required privilege "sys_mount" is missing from the zone's privilege set
     zoneadm: zone zone5 failed to verify
```

영역 관리자가 전역 영역에 사용되는 파일 시스템을 중복 마운트

비전역 영역이 먼저 부트될 때 파일 시스템 계층 내 파일이 존재하는 경우 이는 파일 시스템 데이터가 전역 영역에 의해 관리됨을 나타냅니다. 비전역 영역이 설치된 경우 전역 영역 내 패키징 파일 수가 영역 내부에서 중복되었습니다. 이러한 파일은 zonepath 바로 아래에 상주해야 합니다. 파일이 영역에 추가된 ZFS 데이터 집합이나 디스크 장치에서 영역 관리자가 만든 파일 시스템 아래에 상주할 경우 패키징 문제가 발생할 수 있습니다.

영역 로컬 파일 시스템에서 전역 영역에 의해 관리되는 파일 시스템 데이터 저장과 관련된 문제는 ZFS를 예로 설명할 수 있습니다. ZFS 데이터 집합이 비전역 영역으로 위임된 경우 전역 관리자는 해당 데이터 집합을 사용하여 전역 영역에 의해 관리되는 파일 시스템 데이터를 저장해서는 안 됩니다. 이 구성은 올바르게 업그레이드할 수 없습니다.

예를 들어 ZFS 위임 데이터 집합을 /var 파일 시스템으로 사용하면 안 됩니다. Oracle Solaris 운영 체제는 구성 요소를 /var에 설치하는 코어 패키지를 제공합니다. 이러한 패키지는 업그레이드될 때 /var에 액세스해야 하는데, 위임된 ZFS 데이터 집합에 /var가 마운트된 경우에는 액세스가 불가능합니다.

전역 영역에 의해 제어되는 계층 부분 아래에서의 파일 시스템 마운트는 지원됩니다. 예를 들어 빈 /usr/local 디렉토리가 전역 영역에 있는 경우 영역 관리자는 해당 디렉토리 아래에 다른 내용을 마운트할 수 있습니다.

비전역 영역의 /export와 같이 업그레이드 중에 액세스할 필요가 없는 파일 시스템에 대해 위임된 ZFS 데이터 집합을 사용할 수 있습니다.

영역을 부트할 때 표시되는 netmasks 경고

272 페이지 "영역 부트 방법"에 설명된 대로 영역을 부트할 때 다음 메시지가 표시되는 경우:

```
# zoneadm -z my-zone boot
zoneadm: zone 'my-zone': WARNING: hme0:1: no matching subnet
    found in netmasks(4) for 192.168.0.1; using default of
    255.255.255.0.
```

이 메시지는 경고일 뿐이며 명령은 정상적으로 실행되었습니다. 이 메시지는 시스템에서 영역의 구성에 지정된 IP 주소에 사용될 netmask를 찾을 수 없음을 나타냅니다.

이후 재부트 시 경고가 더 이상 표시되지 않도록 하려면 netmasks 데이터베이스가 전역 영역의 /etc/nsswitch.conf 파일에 나열되어 있고 이러한 데이터베이스 중 하나 이상에 my-zone 영역에 사용될 netmasks와 서브넷이 들어 있어야 합니다.

예를 들어, /etc/inet/netmasks 파일과 로컬 NIS 데이터베이스가 전역에서 netmasks를 확인하는 데 사용되는 경우 /etc/nsswitch.conf의 해당 항목은 다음과 같습니다.

```
netmasks: files nis
```

그러고 나면 이후에 사용할 수 있도록 my-zone 영역의 서브넷과 해당 넷마스크 정보를 /etc/inet/netmasks에 추가할 수 있습니다.

netmasks 명령에 대한 자세한 내용은 netmasks(4) 매뉴얼 페이지를 참조하십시오.

영역이 정지되지 않음

영역과 연결된 시스템 상태를 완전히 삭제할 수 없을 경우 정지 작업이 도중에 실패합니다. 이 경우 영역은 실행 중과 설치됨 상태의 중간 상태가 됩니다. 이 상태에서는 활성 사용자 프로세스나 커널 스레드가 없으며 아무것도 만들 수 없습니다. 정지 작업이 실패할 경우 프로세스를 완료하려면 수동으로 조정해야 합니다.

실패하게 되는 가장 일반적인 원인은 시스템에서 모든 파일 시스템의 마운트를 해제하지 못하기 때문입니다. 시스템 상태를 완전히 삭제하는 기존의 Oracle Solaris 시스템 종료와는 달리, 영역은 영역을 부트하는 동안이나 영역 작업 중에 수행된 마운트가 영역이 정지되고 나서 유지되지 않도록 해야 합니다. zoneadm이 영역에서 실행 중인 프로세스가 없음을 확인하더라도 전역 영역의 프로세스가 영역에서 파일을 연 경우 마운트 해제 작업이 실패할 수 있습니다. proc(1) (pfiles 참조) 및 fuser(1M) 매뉴얼 페이지에 설명된 도구를 사용하여 이러한 프로세스를 찾아서 적절한 조치를 취하십시오. 이러한 프로세스를 처리하고 나서 zoneadm halt를 다시 호출하면 영역이 완전히 정지됩니다.

정지할 수 없는 영역의 경우 zoneadm attach -F 옵션을 사용하여 검증 없이 강제 연결함으로써 분리되지 않은 영역을 마이그레이션할 수 있습니다. 대상 시스템이 영역을 호스팅하도록 올바르게 구성되어야 합니다. 잘못된 구성으로 인해 정의되지 않은 동작이 발생할 수 있습니다. 게다가 영역 내에서 파일의 상태를 알 수 없습니다.

제 3 부

Oracle Solaris 10 Zones

Oracle Solaris 9 Zones 는 Oracle Solaris 11 커널에서 실행되는 x86 및 SPARC Solaris 10 10/10(또는 이후에 릴리스된 Oracle Solaris 10 업데이트) 사용자 환경을 호스팅하는 `solaris10` 브랜드 영역입니다. 원래 시스템에 커널 패치 142909-17(SPARC) 또는 142910-17(x86/x64) 이상의 버전을 먼저 설치할 경우 이전 Oracle Solaris 10 릴리스를 사용할 수 있습니다.

◆ ◆ ◆ 29 장

29 Oracle Solaris 10 Zones 소개

BrandZ는 Oracle Solaris 11 환경에서 실행할 수 없는 응용 프로그램을 실행하는 데 사용되는 브랜드 영역을 만들기 위한 프레임워크를 제공합니다. 여기서 설명하는 브랜드는 `solaris10` 브랜드인 Oracle Solaris 10 Zones입니다. 이러한 `solaris10` 브랜드 영역 내에서 실행되는 작업 로드는 Oracle Solaris 커널의 향상된 기능을 이용할 수 있으며 VNIC(가상 NIC) 및 ZFS 중복 제거와 같이 Oracle Solaris 11 릴리스에서만 사용할 수 있는 몇 가지 혁신 기술을 이용합니다.

주 – `solaris10` 브랜드 영역을 지금 만들려면 30 장, “Oracle Solaris 10 시스템 액세스 및 아카이브 만들기”로 이동하십시오.

---

solaris10 브랜드 정보

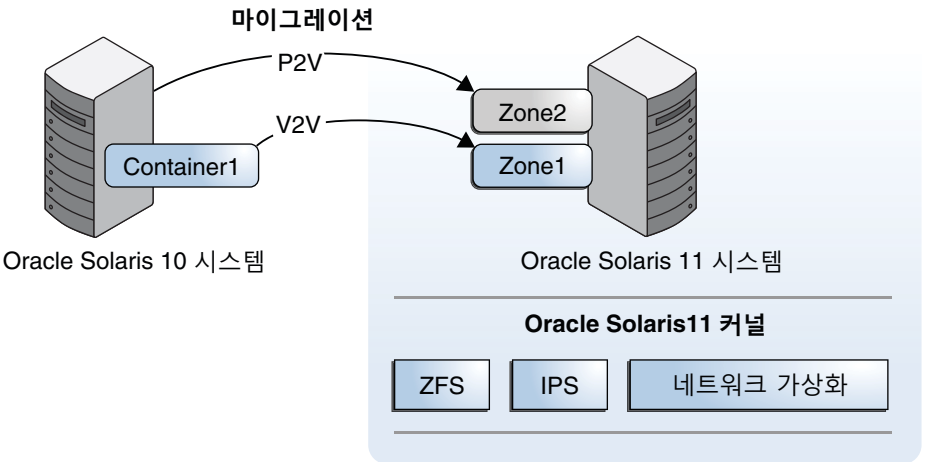
`solaris10(5)` 매뉴얼 페이지에서 설명하는 solaris10 브랜드 영역은 Oracle Solaris 10 10/10 운영 체제 또는 이후 릴리스된 업데이트에서 실행되는 SPARC 및 x86 시스템의 Oracle Solaris 10 응용 프로그램을 위한 완벽한 런타임 환경입니다. Oracle Solaris 9 10/10 이전의 Oracle Solaris 10 릴리스를 실행하고 있는 경우, 원래 시스템에 커널 패치 142909-17(SPARC) 또는 142910-17(x86/x64) 이상의 버전을 처음 설치하면 이전 업데이트 릴리스를 사용할 수 있습니다. 영역을 설치하는 데 사용할 아카이브를 만들기 전에 패치를 설치해야 합니다. Oracle Solaris 10 Zones로 마이그레이션하기 위한 필요 조건은 릴리스의 커널 패치이며 전체 Oracle Solaris 10 9/10 이상의 릴리스가 아닙니다. 패치의 소프트웨어 다운로드 사이트는 My Oracle Support (`https://support.oracle.com`)입니다. "Patches & Updates(패치 및 업데이트)" 탭을 누릅니다. 이 사이트에서 다운로드 지침을 보고 이미지를 다운로드할 수 있습니다. 패치에 대한 추가 정보는 지원 공급자에게 문의하십시오.

브랜드는 Oracle Solaris 11 릴리스가 지원되는 플랫폼으로 정의한 모든 `sun4v`, `sun4u` 및 x86 구조 시스템에서 지원됩니다. 브랜드는 32비트 및 64비트 Oracle Solaris 10 응용 프로그램의 실행을 지원합니다.

브랜드에는 비전역 영역에 Oracle Solaris 10 시스템 이미지를 설치하는 데 필요한 도구가 포함됩니다. Oracle Solaris 10 매체에서 바로 `solaris10` 브랜드 영역을 설치할 수 없습니다. 기존 시스템을 대상 시스템의 비전역 영역으로 직접 마이그레이션하기 위해 P2V(Physical-To-Virtual) 기능이 사용됩니다. `zonep2vchk` 도구는 P2V 프로세스에 필요한 정보를 생성하고 대상 시스템에서 사용하기 위한 `zonecfg` 스크립트를 출력하는 데 사용됩니다. 스크립트는 소스 시스템 구성과 일치하는 영역을 만듭니다. Oracle Solaris 11 시스템에서 `/usr/sbin/zonep2vchk` 스크립트를 복사할 수 있습니다.

브랜드는 기존 Oracle Solaris 10 `native` 영역을 `solaris10` 비전역 영역으로 마이그레이션하는 데 사용되는 도구도 지원합니다. Oracle Solaris 10 `native` 비전역 영역을 `solaris10` 브랜드 영역으로 마이그레이션하기 위한 V2V(가상 대 가상) 프로세스는 P2V와 동일한 아카이브 형식을 지원합니다. 자세한 내용은 31 장, “(선택적) Oracle Solaris 10 Zone으로 **고유** 비전역 영역 마이그레이션”을 참조하십시오.

그림 29–1 Oracle Solaris 10 Zones으로 Oracle Solaris 10 Containers 전환



solaris10 영역 지원

`solaris10` 브랜드 영역은 전체 루트 비전역 영역 모델을 지원합니다. 모든 필수 Oracle Solaris 10 소프트웨어와 추가 패키지는 영역의 전용 파일 시스템에 설치됩니다.

비전역 영역은 자체 ZFS 데이터 집합에 상주해야 하며 ZFS만 지원됩니다. ZFS 데이터 집합은 영역을 설치하거나 연결할 때 자동으로 생성됩니다. ZFS 데이터 집합을 만들 수 없을 경우 영역이 설치되거나 연결되지 않습니다. 영역 경로의 상위 디렉토리도 ZFS 데이터 집합이어야 하며 그렇지 않으면 파일 시스템이 생성되지 않습니다.

`native` Oracle Solaris 10 비전역 영역에서 실행하는 모든 응용 프로그램이나 프로그램은 `solaris10` 브랜드 영역에서도 작동해야 합니다.

영역은 정적으로 링크된 바이너리를 지원하지 않습니다.

주 – 레이블이 사용되는 Oracle Solaris Trusted Extensions 시스템에 `solaris10` 브랜드 영역을 만들고 설치할 수 있지만, 부트되는 브랜드가 `labeled` 브랜드인 **경우** 이 시스템 구성에서는 브랜드 영역만 부트할 수 있습니다. Oracle Solaris 10 시스템에서 Oracle Solaris Trusted Extensions를 사용하는 고객은 인증된 Oracle Solaris 시스템 구성으로 전환해야 합니다.

Oracle Solaris 10 Zones에서 SVR4 패키징 및 패치

solaris10 브랜드 영역에서 패키징 및 패치 사용 정보

SVR4 패키지 메타데이터는 영역 내부에서 사용 가능하며, 패키지 및 패치 명령은 올바르게 작동합니다. 올바르게 작동하려면 아카이브를 만들기 **전에** 패치 119254-75(SPARC) 또는 119255-75(x86/x64) 이상 버전을 Oracle Solaris 10 시스템에 **설치해야 합니다**. 패치의 소프트웨어 다운로드 사이트는 My Oracle Support (`https://support.oracle.com`)입니다. "Patches & Updates(패치 및 업데이트)" 탭을 눌러 다운로드 지침을 보고 이미지를 다운로드합니다. 패치에 대한 추가 정보는 지원 공급자에게 문의하십시오.

영역이 전체 루트 영역이므로 패키지 또는 패치의 커널 구성 요소가 사용되지 않더라도 모든 패키징 및 패치 작업은 성공적입니다. SVR4 패키지는 현재 영역에만 설치됩니다. `solaris10` `native` 영역에서 사용되는 SVR4 패키징에 대한 자세한 내용은 **System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones** 의 "25장, About Packages on an Solaris System With Zones Installed (Overview)" 및 "26장, Adding and Removing Packages and Patches on a Solaris System With Zones Installed (Tasks)"를 참조하십시오. 이 설명서는 Oracle Solaris 10 버전의 설명서입니다.

원격으로 패키지 및 패치 작업 수행 정보

Oracle Solaris 10 Zones 내에서 시작되는 패치 작업의 경우 원격 시스템이 다른 `solaris10` 영역이면 패치 적용 작업이 올바르게 작동합니다. 하지만 원격 시스템이 미니루트이거나 `solaris10` 영역이 아닌 Oracle Solaris 10 시스템이면 이 작업은 정의되지 않은 결과를 생성합니다. 마찬가지로, Oracle Solaris 10 Zones 대신에 미니루트나 시스템에서 Oracle Solaris 10 Zones를 패치하는 데 사용되는 경우 패키지 도구는 정의되지 않은 결과를 생성합니다.

관리자는 패치 작업을 실행할 때 `patchadd` 및 `patchrm` 도구를 사용하여 대체 루트를 지정할 수 있습니다. 이 기능을 사용하여 관리자는 NFS를 통해 `root` 디렉토리가 표시되는 Oracle Solaris 10 미니루트 및 Oracle Solaris 10 물리적 시스템과 같은 원격 시스템을 패치할 수 있습니다.

예를 들어, Oracle Solaris 10 시스템의 루트 디렉토리가 로컬 시스템의 /net/a-system 디렉토리에 NFS 마운트된 경우 로컬 시스템에서 원격 Oracle Solaris 10 시스템을 패치할 수 있습니다.

원격 시스템에 패치 142900-04(또는 이상 버전)를 설치하려면:

```
# patchadd -R /net/a-system 142900-04
```

자세한 내용은 **man pages section 1M: System Administration Commands** 을 참조하십시오.

- patchadd(1M), -R 및 -C 옵션
- patchrm(1M)

NFS 클라이언트인 비전역 영역

영역은 NFS 클라이언트일 수 있습니다. 버전 2, 버전 3 및 버전 4 프로토콜이 지원됩니다. 이 NFS 버전에 대한 자세한 내용은 **Oracle Solaris 관리: 네트워크 서비스**의 "NFS 서비스의 기능"을 참조하십시오.

기본 버전은 NFS 버전 4입니다. 다음 방법 중 하나를 사용하여 클라이언트에서 다른 NFS 버전을 사용하도록 설정할 수 있습니다.

- 영역이 기본적으로 지정된 버전을 사용하도록 /etc/default/nfs를 편집하여 NFS_CLIENT_VERSMAX=*number*를 설정할 수 있습니다. **Oracle Solaris 관리: 네트워크 서비스**의 "NFS 서비스 설정"을 참조하십시오. 작업 맵에서 "/etc/default/nfs 파일을 수정하여 클라이언트에서 다른 버전의 NFS를 선택하는 방법" 절차를 사용합니다.
- 버전 마운트를 수동으로 만들 수 있습니다. 이 방법은 /etc/default/nfs의 내용을 대체합니다. **Oracle Solaris 관리: 네트워크 서비스**의 "NFS 서비스 설정"을 참조하십시오. 작업 맵에서 명령줄을 사용하여 클라이언트에서 다른 버전의 NFS를 선택하는 방법 절차를 사용합니다.

일반 Zones 개념

이 설명서의 제1부 및 제2부에서 설명하는 다음과 같은 리소스 관리 및 영역 개념에 익숙해야 합니다.

- zonep2vchk 도구
- 지원되는 기능과 지원되지 않는 기능
- 관리자가 응용 프로그램에 사용 가능한 시스템 리소스를 사용하는 방법을 제어할 수 있는 리소스 제어
- 영역을 구성, 설치 및 관리하는 데 사용되는 명령, 주로 zonecfg, zoneadm 및 zlogin
- zonecfg 리소스 및 등록 정보 유형

- 전역 영역 및 비전역 영역
- 전체 루트 비전역 모델
- zonecfg 유틸리티를 통해 부여된 인증
- 전역 관리자 및 영역 관리자
- 영역 상태 모델
- 영역 격리 특성
- 권한
- 네트워킹
- 영역 공유 IP 및 배타적 IP 유형
- 영역에 리소스 풀과 같은 기능 사용
- 점유율에 따라 CPU 시간을 할당할 수 있는 예약 클래스인 FSS(Fair Share Scheduler)
- 브랜드 영역의 RSS(Resident Set Size)를 제어하기 위해 전역 영역에서 사용할 수 있는 rcapd(resource capping daemon)

이 릴리스의 Oracle Solaris 10 Zones 정보

작동 제한 사항

/dev/sound 장치를 solaris10 브랜드 영역으로 구성할 수 없습니다.

읽기 전용 영역을 만드는 데 사용하는 file-mac-profile 등록 정보를 사용할 수 없습니다.

quota(1M)에서 설명하는 quota 명령을 사용하여 solaris10 브랜드 영역 내부에서 사용할 UFS 파일 시스템에 대한 쿼터 정보를 검색할 수 없습니다.

Oracle Solaris 10 Zones의 네트워킹

다음 절에서는 Oracle Solaris 10 Zones에서 사용할 수 없거나 Oracle Solaris 10 Zones에서 달라지는 Oracle Solaris 10 네트워킹 구성 요소를 식별합니다.

지원되지 않는 네트워킹 구성 요소

- atun STREAMS 모듈을 사용하는 자동 커널은 지원되지 않습니다.
- 다음 ndd 조정 가능 매개변수는 solaris10 브랜드 영역에서 지원되지 않습니다.
  - ip_squeue_fanout
  - ip_soft_rings_cnt

- ip_ire_pathmtu_interval
- tcp_mdt_max_pbufs

달라지는 네트워킹 기능

배타적 IP 구성을 사용하는 solaris10 브랜드 영역에서는 다음 기능이 물리적 Oracle Solaris 10 시스템과 다릅니다.

- Oracle Solaris 11 릴리스에서는 Mobile IP를 사용할 수 없습니다.
- solaris10 브랜드 영역에서 tcp, udp 또는 icmp 소켓이 열려 있는 경우 autopush 구성이 무시됩니다. 이러한 소켓은 기본적으로 STREAMS 장치 대신 모듈에 매핑됩니다. autopush를 사용하려면 soconfig(1M) 및 sock2path.d(4) 매뉴얼 페이지에서 설명하는 soconfig 및 sock2path.d 유틸리티를 사용하여 이 소켓을 STREAMS 기반 장치에 명시적으로 매핑합니다.
- Oracle Solaris 10 9/10 또는 이전 업데이트를 실행하는 물리적 시스템에서 아카이브된 solaris10 브랜드 영역에서는 VNIC와 같은 /dev/net 링크가 데이터 링크 공급자 인터페이스 라이브러리(libdlpi)에서 지원되지 않습니다. Oracle Solaris 10 8/11에서는 이러한 링크가 지원됩니다. 라이브러리에 대해서는 libdlpi(3LIB) 매뉴얼 페이지에서 설명합니다.

  Oracle Solaris 10 8/11의 libdlpi 라이브러리 또는 libpcap 버전 1.0.0 이상의 라이브러리를 사용하지 않는 응용 프로그램은 VNIC와 같은 /dev/net 링크에 액세스할 수 없습니다.
- Oracle Solaris 10 Zones의 IPMP(IP Network Multipathing)는 Oracle Solaris 11 릴리스를 기반으로 하므로 Oracle Solaris 10 운영 체제의 명령 출력과 비교할 때 ifconfig 명령의 출력에는 차이가 있습니다. 하지만 ifconfig 명령과 IPMP의 문서화된 기능은 변경되지 않았습니다. 따라서 문서화된 인터페이스를 사용하는 Oracle Solaris 10 응용 프로그램은 수정 없이 Oracle Solaris 10 Zones에서 계속 작동합니다.

  다음 예제에서는 데이터 주소 198.162.1.3을 사용하는 IPMP 그룹 ipmp0에 대한 solaris10 브랜드 영역의 ifconfig 명령 출력과 각각 테스트 주소 198.162.1.1 및 198.162.1.2를 사용하는 기본 인터페이스 e1000g1 및 e1000g2를 보여 줍니다.

```
% ifconfig -a
e1000g1:
flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 8
        inet 198.162.1.1 netmask ffffff00 broadcast 198.162.1.255
        ether 0:11:22:45:40:a0
e1000g2:
flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 9
        inet 198.162.1.2 netmask ffffff00 broadcast 198.162.1.255
        ether 0:11:22:45:40:a1
ipmp0: flags=8011000803<UP,BROADCAST,MULTICAST,IPv4,FAILED,IPMP> mtu 68
index 10
        inet 198.162.1.3 netmask ffffff00 broadcast 198.162.1.255
        groupname ipmp0
```

- Oracle Solaris 10 시스템에서 생성되는 표시와는 달리 Oracle Solaris 10 Container의 `ifconfig` 명령은 IP 주소에 대한 기본 인터페이스의 바인딩을 표시하지 않습니다. `arp` 명령을 `-an` 옵션과 함께 사용하여 이 정보를 얻을 수 있습니다.
- IPv6에 대해 인터페이스를 측정하고 주소 구성이 성공하면 인터페이스에 고유의 전역 주소가 지정됩니다. Oracle Solaris 10 시스템에서는 IPMP 그룹의 각 물리적 인터페이스에 고유의 전역 주소가 있으며 IPMP 그룹에는 인터페이스와 같은 수의 전역 주소가 있습니다. Oracle Solaris 10 Zones에서는 IPMP 인터페이스에만 고유의 전역 주소가 있습니다. 기본 인터페이스에는 고유의 전역 주소가 없습니다.
- Oracle Solaris 10 운영 체제와는 달리, IPMP 그룹에 인터페이스가 하나만 있는 경우 테스트 주소와 데이터 주소가 같을 수 없습니다.

`arp`(1M) 및 `ifconfig`(1M) 매뉴얼 페이지와 332 페이지 “배타적 IP 영역의 IP Network Multipathing”를 참조하십시오.

native 비전역 영역이 설치된 경우

Oracle Solaris 10 9/10(또는 이후에 릴리스된 업데이트) 소스 물리적 시스템에 **고유** 영역이 있는 경우 P2V 프로세스의 추가 단계가 수행됩니다. 영역이 중첩되지 않으므로 이 시스템에서 P2V 프로세스를 수행하면 기존 영역이 브랜드 영역 내부에서 사용 불가능하게 됩니다. 영역을 설치할 때 기존 영역이 감지되고, 중첩된 영역을 사용할 수 없으며 디스크 공간을 복구할 수 있음을 알리는 경고가 발생합니다. 먼저 31 장, “(선택적) Oracle Solaris 10 Zone으로 **고유** 비전역 영역 마이그레이션”에서 설명하는 V2V 프로세스를 사용하여 해당 영역을 마이그레이션할 수 있습니다.

이전 릴리스를 실행하는 시스템에서 커널 패치를 적용할 경우 기존 영역을 마이그레이션하기 전에 패치를 적용합니다.

◆ ◆ ◆ 30장

30 Oracle Solaris 10 시스템 액세스 및 아카이브 만들기

이 장에서는 Oracle Solaris 10 10/09(또는 그 이후 릴리스된 업데이트) 시스템에 대한 정보를 얻는 것과 시스템의 아카이브를 만드는 것에 대해 설명합니다. 기존 Oracle Solaris 시스템을 대상 시스템의 비전역 영역으로 직접 마이그레이션하기 위해 P2V(Physical-To-Virtual) 기능이 사용됩니다. 대상 시스템에서 필요한 패키지에 대한 정보도 제공됩니다.

소스 및 대상 시스템 필수 조건

Oracle Solaris 10 패키지 및 패치 도구 사용

Oracle Solaris 10 영역에서 Oracle Solaris 10 패키지 및 패치 도구를 사용하려면 이미지가 생성되기 **전에** 해당 구조에 대해 다음 패치를 설치합니다.

- 119254-75, 119534-24 및 140914-02(SPARC)
- 119255-75, 119535-24 및 140915-02(x86/x64)

P2V 프로세스는 패치 없이 실행할 수 있지만 `solaris10` 브랜드 영역 내에서 패키지와 패치 도구가 제대로 실행되지 않습니다.

대상 시스템에 필수 Oracle Solaris 패키지 설치

시스템에서 Oracle Solaris 10 영역을 사용하려면 Oracle Solaris 11을 실행 중인 시스템에 `pkg:/system/zones/brand/brand-solaris10`이 설치되어 있어야 합니다.

저장소에 대한 자세한 내용은 313 페이지 “Oracle Solaris 11 릴리스를 실행하는 시스템의 이미지 패키징 시스템 소프트웨어”를 참조하십시오.

패키지 설치 지침은 **Oracle Solaris 11 소프트웨어 패키지 추가 및 업데이트**를 참조하십시오.

zonep2vchk 유틸리티를 사용하여 마이그레이션될 시스템 액세스

기존 Oracle Solaris 10 9/10 시스템(또는 이후에 릴리스된 Solaris 10 업데이트)을 Oracle Solaris 11 시스템의 `solaris10` 브랜드 영역으로 직접 마이그레이션할 수 있습니다.

마이그레이션을 시작하기 위해서 `zonep2vchk(1M)`에 설명되어 있는 22 장, "영역 마이그레이션 및 zonep2vchk 도구 정보" 도구를 사용하여 소스 시스템을 검사하고 필요한 정보를 수집합니다. 이 도구는 마이그레이션될 시스템을 평가하여 네트워크 구성을 포함하는 `zonecfg` 템플리트를 생성하기 위해 사용됩니다.

기존 시스템에서 수행된 서비스에 따라 설치 후 전역 관리자나 해당 인증이 부여된 사용자가 영역을 수동으로 사용자 정의해야 할 수 있습니다. 예를 들어, 영역에 지정된 권한을 수정해야 할 수 있습니다. 이 작업을 자동으로 실행되지 않습니다. 또한 영역 내부에서 실행되지 않는 시스템 서비스도 있으므로 모든 Oracle Solaris 10 시스템이 영역 내로 마이그레이션하기에 적합하지는 않습니다.

주 – 마이그레이션될 시스템에 `native` 비전역 영역이 있는 경우, 이러한 영역은 먼저 삭제하거나, 아카이브하여 새 대상 시스템의 영역으로 이동해야 합니다. 희소 루트 영역에 대해서는 준비 상태의 영역에서 아카이브가 수행되어야 합니다. 마이그레이션에 대한 자세한 내용은 31 장, "(선택적) Oracle Solaris 10 Zone으로 **고유** 비전역 영역 마이그레이션"을 참조하십시오. 희소 루트 영역에 대한 자세한 내용은 Oracle Solaris 10 설명서에서 영역 개요를 참조하십시오.

---

zonep2vchk 도구 가져오기

Oracle Solaris 11 시스템에서 Oracle Solaris 10 시스템으로 `/usr/sbin/zonep2vchk` 스크립트를 복사할 수 있습니다. 시스템의 특정 위치에서 `zonep2vchk` 유틸리티를 실행하지 않아도 됩니다.

Oracle Solaris 10 시스템을 영역으로 직접 마이그레이션하기 위한 이미지 만들기

Oracle Solaris Flash 아카이브 도구를 사용하여 영역으로 마이그레이션할 수 있는 설치된 시스템의 이미지를 만들 수 있습니다.

이미지를 만들기 전에 영역에서 실행될 모든 소프트웨어를 시스템에서 완전히 구성할 수 있습니다. 그러면 영역이 설치될 때 설치 프로그램에서 이 이미지를 사용합니다.

▼ flarcreate를 사용하여 이미지를 만드는 방법

ZFS 루트가 있는 시스템에서 `flarcreate(1M)` Oracle Solaris 10 매뉴얼 페이지에 설명되어 있는 `flarcreate` 명령을 사용하여 시스템 이미지를 만들 수 있습니다. 기본적으로 생성되는 `flar`은 **Oracle Solaris 관리: ZFS 파일 시스템**의 "ZFS 데이터 전송 및 수신"에 설명되어 있는 ZFS 전송 스트림입니다.

이 절차 예에서는 NFS를 사용하여 대상 Oracle Solaris 11 시스템에 플래시 아카이브를 배치하지만 어떤 방법을 사용해서든 파일을 이동할 수 있습니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 필요한 권한 프로파일이 있는 사용자여야 합니다.

1. **관리자가 됩니다.**
2. **아카이브될 소스 Oracle Solaris 10 시스템으로 로그인합니다.**
3. **root 디렉토리로 디렉토리를 변경합니다.**

   ```
   # cd /
   ```

4. **소스 시스템에서 `flarcreate`를 사용하여 `s10-system`이라는 플래시 아카이브 이미지 파일을 만들고 이 아카이브를 대상 Oracle Solaris 11 시스템에 배치합니다.**

   ```
   source-system # flarcreate -n s10-system /net/target/export/archives/s10-system.flar
   ```

▼ flarcreate를 사용하여 특정 데이터를 제외하는 방법

아카이브에서 ZFS 데이터 집합 경계에 있지 않은 데이터를 제외하려면 `flarcreate`와 함께 `cpio` 또는 `pax`를 사용해야 합니다. `-L archiver` 옵션을 사용하여 `cpio` 또는 `pax`를 파일을 아카이브하기 위한 방법으로 지정할 수 있습니다.

이 절차 예에서는 NFS를 사용하여 대상 Oracle Solaris 11 시스템에 플래시 아카이브를 배치하지만 어떤 방법을 사용해서든 파일을 이동할 수 있습니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 필요한 권한 프로파일이 있는 사용자여야 합니다.

1. **관리자가 됩니다.**
2. **아카이브할 소스 Oracle Solaris 10 시스템으로 로그인합니다.**
3. **root 디렉토리로 디렉토리를 변경합니다.**

   ```
   # cd /
   ```

4 소스 시스템에서 flarcreate를 사용하여 s10-system이라는 플래시 아카이브 이미지 파일을 만들고 이 아카이브를 대상 Oracle Solaris 11 시스템에 배치합니다.

```
source-system # flarcreate -S -n s10-system -x /path/to/exclude -L cpio /net/target/export/archives/s10-system.flar
Determining which filesystems will be included in the archive...
Creating the archive...
cpio: File size of "etc/mnttab" has
increased by 435
2068650 blocks
1 error(s)
Archive creation complete.
```

참고 – 경우에 따라 `flarcreate`에서 `cpio` 명령의 오류를 표시할 수 있습니다. 가장 일반적으로 `File size of etc/mnttab has increased by 33`(etc/mnttab의 파일 크기가 33만큼 증가했습니다.) 등의 메시지입니다. 이러한 메시지가 로그 파일이나 시스템 상태를 반영하는 파일과 관련된 것이면 무시할 수 있습니다. 모든 오류 메시지를 철저히 검토해야 합니다.

기타 아카이브 생성 방법

다른 방법을 사용하여 아카이브를 만들 수 있습니다. 설치 프로그램에서 다음 아카이브 형식이 허용됩니다.

- `cpio` 아카이브
- `gzip` 압축 `cpio` 아카이브
- `bzip2` 압축 `cpio` 아카이브
- `-x xustar`(XUSTAR) 형식으로 생성된 `pax` 아카이브
- `ufsdump` 레벨 0(전체) 백업

또한 설치 프로그램에서는 파일 권한, 소유권 및 링크를 저장 및 복원하는 아카이브 유틸리티를 사용하여 생성된 파일 디렉토리만 허용됩니다.

자세한 내용은 cpio(1), pax(1), bzip2 (1), gzip(1) 및 ufsdump(1M) 매뉴얼 페이지를 참조하십시오.

주 – P2V를 위한 아카이브 생성에 플래시 아카이브 외의 방법을 사용하는 경우, 아카이브를 만들기 전에 소스 시스템에서 프로세서 종속 `libc.so.1` `lofs`-마운트된 하드웨어 기능(`hwcap`) 라이브러리를 마운트 해제해야 합니다. 그렇지 않으면 아카이브를 사용하여 설치된 영역이 대상 시스템에서 부트되지 않을 수 있습니다. 아카이브를 만든 후 `lofs` 및 마운트 `-O` 옵션을 사용하여 `/lib/libc.so.1` 위에 적절한 하드웨어 기능 라이브러리를 다시 마운트할 수 있습니다.

```
source-system# unmount /lib/libc.so.1
source-system# mount -O -F lofs /lib/libc.so.1
```

호스트 ID 에뮬레이션

독립형 Oracle Solaris 시스템에서 새 시스템의 영역으로 응용 프로그램이 마이그레이션될 때 `hostid`가 새 시스템의 `hostid`로 변경됩니다.

경우에 따라 응용 프로그램이 기존 `hostid`에 의존하여 응용 프로그램 구성을 업데이트할 수 없습니다. 이러한 경우 기존 시스템의 `hostid`를 사용하도록 영역을 구성할 수 있습니다. 243 페이지 “영역 구성 방법”에 설명된 대로 `hostid`를 지정하도록 `zonecfg` 등록 정보를 설정하면 됩니다. 사용되는 값은 기존 시스템에서 실행된 `hostid` 명령의 출력이어야 합니다. 설치된 영역에서 `hostid`를 확인하려는 경우에도 `hostid` 명령을 사용합니다.

호스트 ID에 대한 자세한 내용은 `hostid(1)`을 참조하십시오.

◆ ◆ ◆ 31장

31 (선택적) Oracle Solaris 10 Zone으로 고유 비전역 영역 마이그레이션

이 장에서는 Oracle Solaris 10 9/10(또는 이후 릴리스된 업데이트) 시스템의 **고유** 비전역 영역을 Oracle Solaris 11를 실행 중인 시스템의 Oracle Solaris 10 Zones로 마이그레이션하는 방법을 설명합니다.

마이그레이션하려는 시스템에 `native` 비전역 영역이 있는 경우에만 이 장을 읽으십시오. 먼저 이러한 영역을 아카이브하고 새 대상 시스템의 브랜드 영역으로 이동해야 합니다.

아카이브 고려 사항

Oracle Solaris 10 시스템의 희소 루트 영역은 시스템에서 `solaris10` 브랜드 영역 마이그레이션을 위한 전체 루트 모델로 변환됩니다. V2V 프로세스를 수행하기 전에 희소 루트 영역은 소스 시스템에서 준비 상태에 있어야 합니다. 그러면 아카이브를 만들기 전에 `inherited-pkg-dir` 리소스가 마운트됩니다. 이러한 개념에 대한 자세한 내용은 이 설명서의 Oracle Solaris 10 버전에서 영역 개요를 참조하십시오.

영역의 브랜드는 프로세스의 일부로 변경됩니다.

solaris10 영역 마이그레이션 개요

Oracle Solaris 10 `native` 영역을 `solaris10` 브랜드 영역으로 마이그레이션하기 위한 V2V(가상 대 가상) 프로세스는 P2V와 동일한 아카이브 형식을 지원합니다. 이 프로세스는 한 시스템에서 다른 시스템으로 영역을 마이그레이션하기 위한 기존 인터페이스인 `zoneadm attach` 하위 명령을 사용합니다. `solaris10` 브랜드 `attach` 하위 명령은 `install` 하위 명령의 동일한 옵션에 해당하는 다음 옵션을 사용합니다.

옵션	설명
-a *path*	영역으로 압축을 풀기 위한 아카이브 경로를 지정합니다. 전체 플래시 아카이브 및 pax, cpio, gzip 압축 cpio, bzip 압축 cpio 및 레벨 0 ufsdump가 지원됩니다.
-d *path*	설치를 위한 소스로 파일 트리의 경로를 지정합니다.
-d –	대시 매개 변수와 함께 -d 옵션을 사용하여 기존 디렉토리 레이아웃이 zonepath에서 사용되도록 지정합니다. 따라서 관리자가 설치 전에 zonepath 디렉토리를 수동으로 설정하는 경우, -d – 옵션을 사용하여 이미 있는 디렉토리를 나타낼 수 있습니다.

solaris10 영역 연결 및 분리 정보

대상 시스템에서 영역을 구성한 다음 zoneadm 명령을 detach 및 attach 하위 명령 그리고 -a 옵션과 함께 사용하여 아카이브를 연결하거나 -d 옵션과 함께 사용하여 zonepath를 지정하는 방법으로 solaris10 영역을 Oracle Solaris 호스트로 마이그레이션할 수 있습니다. 이 프로세스는 305 페이지 “영역 마이그레이션 정보” 및 306 페이지 “ZFS 아카이브를 사용하여 비전역 영역을 마이그레이션하는 방법”에 설명되어 있습니다.

solaris10 브랜드 영역 마이그레이션

zonecfg 및 zoneadm 명령을 사용하여 기존 비전역 영역을 한 시스템에서 다른 시스템으로 마이그레이션할 수 있습니다. 영역이 중지되고 현재 호스트에서 분리됩니다. zonepath는 연결된 대상 호스트로 이동합니다.

zoneadm detach 프로세스는 다른 시스템에서 영역을 연결하는 데 필요한 정보를 만듭니다. zoneadm attach 프로세스는 대상 시스템에 영역을 호스팅하기 위한 올바른 구성이 있는지 확인합니다.

여러 가지 방법으로 zonepath를 새 호스트에서 사용 가능하게 만들 수 있으므로 한 시스템에서 다른 시스템으로 zonepath를 실제로 이동하는 작업은 전역 관리자가 수행하는 수동 프로세스입니다.

새 시스템에 연결된 경우 영역은 설치됨 상태가 됩니다.

예 31–1 샘플 attach 명령

```
host2# zoneadm -z zonename attach -a /net/machine_name/s10-system.flar
```

Oracle Solaris 10 시스템에서 기존 영역 마이그레이션

물리적 시스템을 마이그레이션하기 전에 먼저 시스템의 기존 비전역 영역을 아카이브하고 새 대상 시스템의 영역으로 이동해야 합니다.

▼ 기존 native 비전역 영역을 마이그레이션하는 방법

V2V 프로세스를 사용하여 Solaris 10 시스템의 기존 영역을 Oracle Solaris 11 릴리스가 실행되는 시스템의 `solaris10` 브랜드 영역으로 마이그레이션합니다.

1 기존 영역의 구성을 인쇄합니다. 대상 시스템에서 영역을 다시 만들려면 이 정보가 필요합니다.

```
source# zonecfg -z my-zone info
zonename: my-zone
zonepath: /zones/my-zone
brand: native
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
hostid: 1337833f
inherit-pkg-dir:
         dir: /lib
inherit-pkg-dir:
         dir: /platform
inherit-pkg-dir:
         dir: /sbin
inherit-pkg-dir:
         dir: /usr
net:
         address: 192.168.0.90
         physical: bge0
```

2 영역을 정지합니다.

```
source# zoneadm -z my-zone halt
```

영역 내의 응용 프로그램 또는 시스템 데이터가 일관성이 없는 상태로 캡처될 수 있으므로 실행 중인 영역을 아카이브하면 안됩니다.

3 (옵션) 영역이 `inherit-pkg-dir` 설정을 사용하는 희소 루트 영역인 경우 상속된 디렉토리가 아카이브되도록 먼저 영역을 준비합니다.

```
source# zoneadm -s myzone ready
```

4 **zonepath /zones/my-zone 을 사용하여 영역을 아카이브합니다.**

- **영역에 대해 my-zone.cpio.gz라는 gzip 압축된 cpio 아카이브를 만듭니다. 이 영역은 대상 시스템에서도 my-zone이라는 이름이 됩니다.**

```
source# cd /zones
source# find my-zone -print | cpio -oP@ | gzip >/zones/my-zone.cpio.gz
```

- **대상 시스템에서 영역의 이름을 바꾸려는 경우 zonepath 내에서 아카이브를 만듭니다.**

```
source# cd /zones/my-zone
source# find root -print | cpio -oP@ | gzip >/zones/my-zone.cpio.gz
```

5 **다음과 같은 파일 복사를 위한 파일 전송 방식을 사용하여 아카이브를 대상 Oracle Solaris 11 시스템으로 전송합니다.**

- sftp(1) 매뉴얼 페이지에서 설명하는 sftp 명령
- NFS 마운트
- 파일 복사를 위한 기타 파일 전송 방식

6 **대상 시스템에서 영역을 다시 만듭니다.**

```
target# zonecfg -z my-zone
my-zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:my-zone> create -t SYSsolaris10
zonecfg:my-zone> set zonepath=/zones/my-zone
...
```

주 – 영역의 브랜드는 solaris10이어야 하며 원래 영역을 희소 루트 영역으로 구성한 경우에도 영역은 inherit-pkg-dir 설정을 사용할 수 없습니다. inherit-pkg-dir 리소스에 대한 자세한 내용은 Part II, Oracle Solaris Zones를 참조하십시오.

대상 시스템에 다른 하드웨어, 다른 네트워크 인터페이스 또는 영역에서 구성해야 하는 기타 장치나 파일 시스템이 있는 경우 영역의 구성을 업데이트해야 합니다. 16 장, "비전역 영역 구성(개요)" 17 장, "비전역 영역 계획 및 구성(작업)" 및 305 페이지 "영역 마이그레이션 정보"를 참조하십시오.

---

7 **영역의 구성을 표시합니다.**

```
target# zonecfg -z my-zone info
zonename: my-zone
zonepath: /zones/my-zone
brand: solaris10
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
hostid: 1337833f
```

```
net:
        address: 192.168.0.90
        physical: bge0
```

8 소스 시스템에서 만든 아카이브의 영역을 대상 시스템의 /zones 디렉토리로 전송된 아카이브와 연결합니다.

```
target# zoneadm -z my-zone attach -a /zones/my-zone.cpio.gz
```

영역 설치가 성공적으로 완료되면 영역은 부트할 준비가 됩니다.

나중에 사용할 경우를 대비하여 영역의 아카이브를 저장하거나 시스템에서 제거할 수 있습니다.

대상 시스템에서 아카이브를 제거하려면

```
target# rm /zones/myzone.cpio.gz
```

◆ ◆ ◆ 32장

32 solaris10 브랜드 영역 구성

이 장에서는 `Solaris10` 브랜드 영역 구성을 다룹니다.

미리 구성 작업

다음이 필요합니다.

- Oracle Solaris 11 릴리스를 실행 중인 지원되는 SPARC 또는 x86 시스템
- 기본은 `anet` 리소스가 있는 배타적 IP 유형입니다. 네트워크 연결이 필요한 영역의 경우 만들려는 각 공유 IP 영역에 대해 하나 이상의 고유 IPv4 주소가 필요합니다. 물리적 인터페이스도 지정해야 합니다.
- `solaris10` 컨테이너로 마이그레이션하려는 Oracle Solaris 10 10/09(또는 이후에 릴리스된 업데이트) 운영 체제를 실행 중인 시스템. 이전 업데이트는 해당 커널 패치로 마이그레이션할 수 있습니다. 기존 시스템에서 고유한 이미지를 생성할 수 있습니다. 이 프로세스는 390 페이지 “Oracle Solaris 10 시스템을 영역으로 직접 마이그레이션하기 위한 이미지 만들기”에 설명되어 있습니다.

기본적으로 구성에 포함되는 리소스

기본적으로 구성에는 브랜드 영역의 장치, 파일 시스템 및 권한이 포함됩니다.

solaris10 브랜드 영역에서 장치 구성

각 영역에서 지원되는 장치는 해당 브랜드의 매뉴얼 페이지 및 기타 설명서에 설명되어 있습니다. `solaris10` 영역에는 지원되지 않거나 인식할 수 없는 장치를 추가할 수 없습니다. 프레임워크에서 지원되지 않는 장치를 추가하려는 시도를 감지합니다. 영역 구성을 확인할 수 없음을 나타내는 오류 메시지가 표시됩니다.

비전역 영역에서의 장치 고려 사항에 대한 자세한 내용은 333 페이지 “비전역 영역에서 장치 사용”을 참조하십시오.

solaris10 브랜드 영역에 정의된 권한

프로세스는 권한의 일부로 제한됩니다. 권한 제한은 영역에서 다른 영역에 영향을 줄 수 있는 작업을 수행하지 못하도록 방지합니다. 권한 설정은 영역 내에서 권한 있는 사용자의 기능을 제한합니다.

각 브랜드별로 기본값, 필수 기본값, 선택 사항 및 금지된 권한이 정의됩니다. 243 페이지 “영역 구성 방법”의 8단계에 표시된 `limitpriv` 등록 정보를 사용하여 특정 권한을 추가 또는 제거할 수도 있습니다. 표 25–1의 표에 각 영역과 관련된 모든 Solaris 권한 및 각 권한의 상태가 나열되어 있습니다.

권한에 대한 자세한 내용은 `ppriv(1)` 매뉴얼 페이지 및 **시스템 관리 설명서: 보안 서비스**를 참조하십시오.

solaris10 브랜드 영역 구성 프로세스

`zonecfg` 명령을 사용하여 다음을 수행합니다.

- 영역의 브랜드를 설정합니다.
- `solaris10` 영역에 대한 구성을 만듭니다.
- 가상 시스템에서 지정된 리소스 및 등록 정보가 허용되고 내부적으로 일관되는지 여부를 확인하기 위해 구성을 확인합니다.
- 브랜드별 확인을 수행합니다.

`zonep2vchk` 유틸리티를 사용하여 영역 구성을 만들 수 있습니다.

지정된 구성에 대해 `zonecfg verify` 명령을 사용하여 수행되는 검사는 다음을 확인합니다.

- 영역 경로가 지정되어 있는지 확인
- 각 리소스에 대해 필요한 등록 정보가 모두 지정되어 있는지 확인
- 브랜드 요구 사항이 충족되었는지 확인

`zonecfg` 명령에 대한 자세한 내용은 `zonecfg(1M)` 매뉴얼 페이지를 참조하십시오.

대상 영역 구성

Oracle Solaris 11 시스템에 `pkg:/system/zones/brand/brand-solaris10`이 설치되어 있어야 합니다.

`zonecfg` 명령을 사용하여 대상 시스템에 새 영역 구성을 만듭니다.

`zonecfg` 프롬프트는 다음과 같은 형태입니다.

`zonecfg:`*zonename*`>`

파일 시스템과 같이 특정 리소스 유형을 구성할 때 해당 리소스 유형도 프롬프트에 포함됩니다.

`zonecfg:`*zonename*`:fs>`

참고 – `solaris10` 브랜드 영역에서 응용 프로그램을 설치하기 위해 CD 또는 DVD를 사용하게 될 경우, `add fs`를 사용하여 초기에 브랜드 영역을 구성할 때 전역 영역에 CD 또는 DVD 매체에 대한 읽기 전용 액세스를 추가합니다. 그러면 CD 또는 DVD를 사용하여 브랜드 영역에 제품을 설치할 수 있습니다. 자세한 내용은 360 페이지 “비전역 영역에서 CD 또는 DVD 매체에 대한 액세스를 추가하는 방법”을 참조하십시오.

이 절차는 기본 배타적 IP 영역을 구성하는 것에 대해 설명합니다. 227 페이지 “리소스 유형 등록 정보”를 참조하십시오.

▼ 배타적 IP solaris10 브랜드 영역을 구성하는 방법

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 해당 인증이 있는 사용자여야 합니다.

1 관리자로 전환합니다.

2 `s10-zone`이라는 영역으로 배타적 IP `solaris10` 영역을 만듭니다.

```
global# zonecfg -z s10-zone
```

이 영역을 처음으로 구성한 경우 다음 시스템 메시지가 표시됩니다.

```
s10-zone: No such zone configured
Use ’create’ to begin configuring a new zone.
```

3 `SYSsolaris10` 템플리트를 사용하여 새 `solaris10` 영역 구성을 만듭니다.

```
zonecfg:s10-zone> create -t SYSsolaris10
```

`SYSsolaris10` 프로파일은 기본적으로 자동 `anet` 리소스를 포함하는 배타적 IP 영역을 만듭니다.

선택적으로 create를 사용한 다음 브랜드를 설정할 수도 있습니다.

```
create
set brand=solaris10
```

set brand=solaris10을 사용하는 경우 구성에 자동 VNIC를 사용하여 배타적 IP 영역을 추가해야 합니다. anet 리소스를 사용하고 생성될 링크의 기본 링크로 lower-link=auto를 지정합니다. zoneadmd 데몬에서 영역이 부트될 때마다 VNIC가 생성될 링크를 자동으로 선택합니다.

4 이 절차에서 영역 경로 /zones/s10-zone을 설정합니다.

```
zonecfg:s10-zone> set zonepath=/zones/s10-zone
```

5 자동 부트 값을 설정합니다.

```
zonecfg:s10-zone> set autoboot=true
```

true로 설정하는 경우, 전역 영역이 부트될 때 영역이 자동으로 부트됩니다. 기본값은 false입니다. 자동 부트 영역의 경우 영역 서비스 svc:/system/zones:default도 사용으로 설정해야 합니다. svcadm 명령을 사용하여 영역 서비스를 사용으로 설정할 수 있습니다.

6 전역 영역에서 공유되는 ZFS 파일 시스템을 추가합니다.

```
zonecfg:s10-zone> add fs
```

a. 형식을 zfs로 설정합니다.

```
zonecfg:s10-zone:fs> set type=zfs
```

b. 전역 영역에서 마운트할 디렉토리를 설정합니다.

```
zonecfg:s10-zone:fs> set special=share/zone/s10-zone
```

c. 마운트 지점을 지정합니다.

```
zonecfg:s10-zone:fs> set dir=/opt/shared
```

d. 지정을 종료합니다.

```
zonecfg:s10-zone:fs> end
```

이 단계를 두 번 이상 수행하여 둘 이상의 파일 시스템을 추가할 수 있습니다.

7 *tank* 저장소 풀에서 *sales*라는 ZFS 데이터 집합 위임

```
zonecfg:my-zone> add dataset
```

a. ZFS 데이터 집합 *sales*에 대한 경로를 지정합니다.

```
zonecfg:my-zone> set name=tank/sales
```

b. 데이터 집합 지정을 종료합니다.

```
zonecfg:my-zone> end
```

8 **소스 시스템의 hostid가 될 hostid를 설정합니다.**

```
zonecfg:my-zone> set hostid=80f0c086
```

9 **영역에 대한 영역 구성을 확인합니다.**

```
zonecfg:s10-zone> verify
```

10 **영역에 대한 영역 구성을 커밋합니다.**

```
zonecfg:s10-zone> commit
```

11 **zonecfg 명령을 종료합니다.**

```
zonecfg:s10-zone> exit
```

프롬프트에서 명시적으로 commit를 입력하지 않은 경우에도 exit를 입력하거나 EOF가 발생할 때 commit가 자동으로 시도됩니다.

12 **info 하위 명령을 사용하여 브랜드가 solaris10으로 설정되어 있는지 확인합니다.**

```
global# zonecfg -z s10-zone info
```

13 **(옵션) info 하위 명령을 사용하여 hostid를 확인합니다.**

```
global# zonecfg -z s10-zone info hostid
```

다음 순서

참고 – 영역을 구성한 후 영역 구성의 사본을 만들어 두는 것이 좋습니다. 이 백업을 사용하여 나중에 영역을 다시 만들 수 있습니다. 루트 또는 올바른 프로파일의 관리자로 *s10-zone* 영역에 대한 구성을 파일로 인쇄합니다. 이 예에서는 s10-zone.config라는 파일을 사용합니다.

```
global# zonecfg -z s10-zone export > s10-zone.config
```

참조 zonecfg를 사용하여 구성할 수 있는 추가 구성 요소에 대해서는 16 장, “비전역 영역 구성(개요)”을 참조하십시오. 이 설명서는 명령줄 또는 명령 파일 모드에서 zonecfg 명령을 사용하는 것에 대한 정보도 제공합니다. 공유 IP 영역의 경우 zonecfg net 리소스에서 정적 주소가 지정되어야 합니다. ZFS 파일 시스템 추가에 대한 자세한 내용은 **Oracle Solaris 관리: ZFS 파일 시스템**의 “비전역 영역에 ZFS 파일 시스템 추가”를 참조하십시오.

▼ 공유 IP solaris10 브랜드 영역을 구성하는 방법

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 해당 인증이 있는 사용자여야 합니다.

1 **관리자로 전환합니다.**

2 s10-zone이라는 영역으로 공유 IP solaris10 영역을 만듭니다.

```
global# zonecfg -z s10-zone
```

이 영역을 처음으로 구성한 경우 다음 시스템 메시지가 표시됩니다.

```
s10-zone: No such zone configured
Use 'create' to begin configuring a new zone.
```

3 새 solaris10 영역 구성을 만듭니다.

```
zonecfg:s10-zone> create -b
set brand=solaris10
```

주 – IP 유형을 설정하기 위해 **create -t SYSsolaris10-shared-ip**를 사용하지 **마십시오**.

---

4 이 절차에서 영역 경로 /zones/s10-zone을 설정합니다.

```
zonecfg:s10-zone> set zonepath=/zones/s10-zone
```

5 자동 부트 값을 설정합니다.

true로 설정하는 경우, 전역 영역이 부트될 때 영역이 자동으로 부트됩니다. 자동 부트 영역의 경우 영역 서비스 svc:/system/zones:default도 사용으로 설정해야 합니다. 기본값은 false입니다.

```
zonecfg:s10-zone> set autoboot=true
```

6 네트워크 가상 인터페이스를 사용하여 공유 IP 영역을 만듭니다.

```
zonecfg:my-zone> set ip-type=shared

zonecfg:my-zone> add net
```

a. 이 절차에서 네트워크 인터페이스의 physical 장치 유형 bge 장치를 설정합니다.

```
zonecfg:my-zone:net> Set physical=bge0
```

b. 이 절차에서 IP 주소 *10.6.10.233/24*를 설정합니다.

```
zonecfg:my-zone:net> Set address=10.6.10.233/24
```

c. 지정을 종료합니다.

```
zonecfg:my-zone:net> end
```

이 단계를 두 번 이상 수행하여 둘 이상의 네트워크 인터페이스를 추가할 수 있습니다.

7 전역 영역에서 공유되는 ZFS 파일 시스템을 추가합니다.

```
zonecfg:s10-zone> add fs
```

a. 형식을 zfs로 설정합니다.

```
zonecfg:s10-zone:fs> set type=zfs
```

b. 전역 영역에서 마운트할 디렉토리를 설정합니다.

```
zonecfg:s10-zone:fs> set special=share/zone/s10-zone
```

c. 마운트 지점을 지정합니다.

```
zonecfg:s10-zone:fs> set dir=/opt/shared
```

d. 지정을 종료합니다.

```
zonecfg:s10-zone:fs> end
```

이 단계를 두 번 이상 수행하여 둘 이상의 파일 시스템을 추가할 수 있습니다.

8 *tank* 저장소 풀에서 *sales*라는 ZFS 데이터 집합 위임

```
zonecfg:my-zone> add dataset
```

a. ZFS 데이터 집합 *sales*에 대한 경로를 지정합니다.

```
zonecfg:my-zone> set name=tank/sales
```

b. 데이터 집합 지정을 종료합니다.

```
zonecfg:my-zone> end
```

9 소스 시스템의 `hostid`가 될 `hostid`를 설정합니다.

```
zonecfg:my-zone> set hostid=80f0c086
```

10 영역에 대한 영역 구성을 확인합니다.

```
zonecfg:s10-zone> verify
```

11 영역에 대한 영역 구성을 커밋합니다.

```
zonecfg:s10-zone> commit
```

12 `zonecfg` 명령을 종료합니다.

```
zonecfg:s10-zone> exit
```

프롬프트에서 명시적으로 `commit`를 입력하지 않은 경우에도 `exit`를 입력하거나 EOF가 발생할 때 `commit`가 자동으로 시도됩니다.

13 `info` 하위 명령을 사용하여 브랜드가 `solaris10`으로 설정되어 있는지 확인합니다.

```
global# zonecfg -z s10-zone info
```

14 (옵션) `info` 하위 명령을 사용하여 `hostid`를 확인합니다.

```
global# zonecfg -z s10-zone info hostid
```

다음 순서

참고 – 영역을 구성한 후 영역 구성의 사본을 만들어 두는 것이 좋습니다. 이 백업을 사용하여 나중에 영역을 다시 만들 수 있습니다. 루트 또는 올바른 프로파일의 관리자로 *s10-zone* 영역에 대한 구성을 파일로 인쇄합니다. 이 예에서는 `s10-zone.config`라는 파일을 사용합니다.

```
global# zonecfg -z s10-zone export > s10-zone.config
```

참조 `zonecfg`를 사용하여 구성할 수 있는 추가 구성 요소에 대해서는 16 장, “비전역 영역 구성(개요)”을 참조하십시오. 이 설명서는 명령줄 또는 명령 파일 모드에서 `zonecfg` 명령을 사용하는 것에 대한 정보도 제공합니다. 공유 IP 영역의 경우 `zonecfg net` 리소스에서 정적 주소가 지정되어야 합니다. ZFS 파일 시스템 추가에 대한 자세한 내용은 **Oracle Solaris 관리: ZFS 파일 시스템**의 “비전역 영역에 ZFS 파일 시스템 추가”를 참조하십시오.

◆ ◆ ◆ 33장

33 solaris10 브랜드 영역 설치

이 장에서는 solaris10 브랜드 영역 설치를 다룹니다.

영역 설치 이미지

시스템 이미지 유형

- 영역에서 실행할 모든 소프트웨어가 완전히 구성된 Oracle Solaris 시스템의 이미지를 사용할 수 있습니다. 390 페이지 “Oracle Solaris 10 시스템을 영역으로 직접 마이그레이션하기 위한 이미지 만들기”를 참조하십시오.

  zoneadm install -a 명령은 영역의 아카이브가 **아닌** 물리적 시스템의 아카이브를 가져옵니다.
- 물리적 시스템의 이미지 대신 기존 Oracle Solaris 10 native 영역을 사용할 수 있습니다. 31 장, “(선택적) Oracle Solaris 10 Zone으로 **고유** 비전역 영역 마이그레이션”을 참조하십시오.

  zoneadm attach -a 명령은 물리적 시스템의 아카이브가 **아닌** 영역의 아카이브를 가져옵니다.

이미지 sysidcfg 상태

-c를 사용하여 설치 완료 후 영역 구성에 사용할 sysidcfg 파일을 전달할 수 있습니다.

기존 시스템에서 Solaris 10 시스템 아카이브를 만들었고 영역 설치 시 -p(sysidcfg 보존) 옵션을 사용하면 영역의 ID가 이미지 생성에 사용된 시스템의 ID와 같아집니다.

대상 영역 설치 시 -u(sys-unconfig) 및 -c 옵션을 사용하면 생성된 영역에 호스트 이름 또는 이름 서비스가 구성되지 않습니다.

solaris10 브랜드 영역 설치

제2부 및 zoneadm(1M) 매뉴얼 페이지에 설명되어 있는 zoneadm 명령은 비전역 영역 설치 및 관리에 사용되는 기본 도구입니다. zoneadm 명령을 사용하는 작업은 대상 시스템의 전역 영역에서 실행해야 합니다.

아카이브에서 파일의 압축을 해제할 뿐만 아니라 설치 프로세스에서는 호스트에서 영역이 최적으로 실행될 수 있도록 하기 위해 검사, 필요한 사후 처리 및 기타 기능을 수행합니다.

기존 시스템에서 Oracle Solaris 아카이브를 만들었고 영역 설치 시 -p(sysidcfg 보존) 옵션을 사용하면 영역의 ID가 이미지 생성에 사용된 시스템의 ID와 같아집니다.

대상 영역 설치 시 -u(sys-unconfig) 옵션을 사용하면 생성된 영역에 호스트 이름 또는 이름 서비스가 구성되지 않습니다.

주의 – -p 옵션이나 -u 옵션 중 하나를 **사용해야 합니다**. 이러한 두 옵션 중 하나를 지정하지 않은 경우 오류가 발생합니다.

설치 프로그램 옵션

옵션	설명
-a	시스템 이미지를 복사해올 원본 아카이브의 위치입니다. 전체 플래시 아카이브 및 pax, cpio, gzip 압축 cpio, bzip 압축 cpio 및 레벨 0 ufsdump가 지원됩니다.
-c *path*	설치 완료 후 영역 구성에 사용할 sysidcfg 파일을 전달합니다.
-d *path*	시스템 이미지를 복사해올 원본 디렉토리의 위치입니다.
-d –	대시 매개 변수와 함께 -d 옵션을 사용하여 기존 디렉토리 레이아웃이 zonepath에서 사용되도록 지정합니다. 따라서 관리자가 설치 전에 zonepath 디렉토리를 수동으로 설정하는 경우, -d – 옵션을 사용하여 이미 있는 디렉토리를 나타낼 수 있습니다.
-p	시스템 ID를 보존합니다. -p 또는 -u 중 하나를 사용해야 합니다.
-s	확인 없이 설치합니다.
-u	영역을 sys-unconfig합니다. -p 또는 -u 중 하나를 사용해야 합니다. -c를 -u 옵션에 추가로 사용하여 설치 완료 후 영역 구성에 사용할 sysidcfg 파일을 전달할 수 있습니다.

옵션	설명
-v	상세 정보 출력입니다.

-a와 -d 옵션은 동시에 사용할 수 없습니다.

▼ solaris10 브랜드 영역을 설치하는 방법

zoneadm 명령을 install 하위 명령과 함께 사용하여 구성된 solaris10 브랜드 영역을 설치합니다.

Oracle Solaris 10 시스템의 이미지 생성에 대한 자세한 내용은 390 페이지 "Oracle Solaris 10 시스템을 영역으로 직접 마이그레이션하기 위한 이미지 만들기"를 참조하십시오. 이미지를 변경하지 않고 생성된 시스템 이미지에서 sysidcfg ID를 보존하려면 install 하위 명령 뒤에 -p 옵션을 사용합니다. 이미지를 변경하지 않고 생성된 시스템 이미지에서 시스템 ID를 제거하려면 -u 옵션을 사용합니다. 대상 영역에 대해 sys-unconfig가 발생합니다. -c 옵션을 사용하여 설치 완료 후 영역 구성에 사용되는 정보가 담긴 sysidcfg 파일을 포함할 수 있습니다.

절차 예는 물리적으로 설치된 Oracle Solaris 10 시스템에 대해 생성된 아카이브 이미지와 함께 -a 옵션을 사용하는 방법을 보여 줍니다.

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 해당 인증이 있는 사용자여야 합니다.

1 관리자로 전환합니다.

2 zoneadm install 명령을 -a 옵션과 함께 사용하여 구성된 영역 s10-zone을 설치합니다.

```
global# zoneadm -z s10-zone install -a /net/machine_name/s10-system.flar -u
```

설치가 완료되면서 다양한 메시지가 표시됩니다. 여기에 약간의 시간이 걸릴 수 있습니다.

3 (옵션) 오류 메시지가 표시되고 영역 설치에 실패하는 경우, zoneadm list 명령과 -c 및 -v 옵션을 사용하여 영역 상태를 가져옵니다.

```
global# zoneadm list -cv

ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris    shared
 -  s10-zone configured   /zones/s10-zone                solaris10  shared
```

- 상태가 구성된 것으로 나열되어 있으면 메시지에 지정된 대로 수정하고 zoneadm install 명령을 다시 시도합니다.
- 상태가 완료되지 않은 것으로 나열되어 있으며 먼저 다음 명령을 실행합니다.

  ```
  global# zoneadm -z my-zone uninstall
  ```

그런 다음 메시지에 지정된 대로 수정하고 zoneadm install 명령을 다시 시도합니다.

4 설치가 완료되면 list 하위 명령을 -i 및 -v 옵션과 함께 사용하여 설치된 영역을 나열하고 상태를 확인합니다.

```
global# zoneadm list -iv
```

다음과 유사하게 표시됩니다.

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              solaris     shared
 -  s10-zone  installed    /zones/s10-zone               solaris10   shared
```

예 33–1 solaris10 영역 설치

```
# zoneadm -z s10-zone install -u -a /net/machinename/s10_image.flar
      Log File: /var/tmp/s10-zone.install.21207.log
        Source: /net/machinename/s10_image.flar
    Installing: This may take several minutes...
Postprocessing: This may take a minute...

        Result: Installation completed successfully.
      Log File: /zones/s10-zone/root/var/log/s10-zone.install.21207.log
```

일반 오류 설치에 실패하는 경우 로그 파일을 검토합니다. 성공 시 영역 내의 /var/log에 로그 파일이 있습니다. 실패 시 전역 영역의 /var/tmp에 로그 파일이 있습니다.

영역 설치가 인터럽트되거나 실패하는 경우, 영역이 완료되지 않은 상태로 남게 됩니다. uninstall 명령을 -F 옵션과 함께 사용하여 영역을 구성된 상태로 재설정합니다.

◆ ◆ ◆ 34장

34 영역 부트, 로그인 및 영역 마이그레이션

이 장에서는 설치된 영역을 부트하고 zlogin을 사용하여 내부 영역 구성을 완료하는 방법을 설명합니다. 또한 영역을 다른 시스템으로 마이그레이션하는 방법도 다룹니다.

solaris10 브랜드 영역 부트 정보

영역을 부트하면 영역이 실행 중인 상태가 됩니다. 영역은 준비 상태 또는 설치된 상태에서 부트할 수 있습니다. 부트되는 설치된 상태의 영역은 준비 상태에서 실행 중 상태로 투명하게 전환됩니다. 실행 중 상태에 있는 영역에 대해 영역 로그인이 허용됩니다.

초기 부트 후에 처음으로 구성되지 않은 영역에 로그인할 때 내부 영역 구성을 수행합니다.

이미지 sysidcfg 프로파일

기존 시스템에서 Oracle Solaris 10 시스템 아카이브를 만들었고 영역 설치 시 -p(sysidcfg 보존) 옵션을 사용하면 영역의 ID가 이미지 생성에 사용된 시스템의 ID와 같아집니다.

-c 옵션을 사용하여 설치 완료 후 영역 구성에 사용할 sysidcfg 파일을 포함할 수 있습니다. solaris10 영역을 설치하려면 명령줄에서 sysidcfg 파일을 사용합니다. 파일의 전체 경로를 제공해야 합니다.

```
# zoneadm -z s10-zone install -a /net/machine_name/s10-system.flar -u -c /path_to/sysidcfg
```

다음 샘플 sysidcfg 파일은 net0 네트워크 이름과 timezone을 사용하여 배타적 IP 영역을 정적 IP 구성으로 구성합니다.

```
system_locale=C
 terminal=xterm
 network_interface=net0 {
```

```
    hostname=test7
    ip_address=192.168.0.101
    netmask=255.255.255.0
    default_route=NONE
    protocol_ipv6=no
 }
 name_service=NONE
 security_policy=NONE
 timezone=US/Pacific
 timeserver=localhost
 nfs4_domain=dynamic
 root_password=FSPXl81aZ7Vyo
 auto_reg=disable
```

다음 샘플 sysidcfg 파일은 공유 IP 영역을 구성하는 데 사용됩니다.

```
system_locale=C
terminal=dtterm
network_interface=primary {
hostname=my-zone
}
security_policy=NONE
name_service=NIS {
domain_name=special.example.com
name_server=bird(192.168.112.3)
}
nfs4_domain=domain.com
timezone=US/Central
root_password=m4qtoWN
```

다음 샘플 sysidcfg 파일은 배타적 IP 영역을 정적 IP 구성으로 구성하는 데 사용됩니다.

```
system_locale=C
terminal=dtterm
network_interface=primary {
hostname=my-zone
default_route=10.10.10.1
ip_address=10.10.10.13
netmask=255.255.255.0
}
nfs4_domain=domain.com
timezone=US/Central
root_password=m4qtoWN
```

다음 샘플 sysidcfg 파일은 배타적 IP 영역을 DHCP 및 IPv6 옵션으로 구성하는 데 사용됩니다.

```
system_locale=C
terminal=dtterm
network_interface=primary {
dhcp protocol_ipv6=yes
}
security_policy=NONE
name_service=DNS {
domain_name=example.net
name_server=192.168.224.11,192.168.224.33
```

```
}
nfs4_domain=domain.com
timezone=US/Central
root_password=m4qtoWN
```

▼ solaris10 브랜드 영역 내부 구성

프로파일이 제공되지 않을 경우 `zlogin -C`를 처음 사용할 때 구성 도구가 시작됩니다.

이 절차에서 영역 이름은 `s10-zone`입니다.

1. **관리자로 전환합니다.**
2. **한 단말기 창에서, 다음 명령을 사용하여 영역을 부트하기 전에 이 절차에서 영역 콘솔 `s10-zone`에 연결합니다.**

   ```
   # zlogin -C s10-zone
   ```
3. **두번째 창에서, 415 페이지 "solaris10 브랜드 영역을 부트하는 방법"에 설명된 대로 영역을 부트합니다.**

▼ solaris10 브랜드 영역을 부트하는 방법

이 절차를 수행하려면 전역 관리자이거나 전역 영역에서 해당 인증이 있는 사용자여야 합니다.

1. **관리자로 전환합니다.**
2. **`zoneadm` 명령을 `-z` 옵션과 함께, 영역의 이름인 `s10-zone` 그리고 `boot` 하위 명령을 사용하여 영역을 부트합니다.**

   ```
   global# zoneadm -z s10-zone boot
   ```
3. **부트가 완료되면 `list` 하위 명령을 `-v` 옵션과 함께 사용하여 상태를 확인합니다.**

   ```
   global# zoneadm list -v
   ```

   다음과 유사하게 표시됩니다.

   ```
   ID  NAME     STATUS       PATH                           BRAND      IP
    0  global   running      /                              solaris     shared
    1  s10-zone running      /zone/s10-zone                 solaris10   shared
   ```

참조 부트 영역 및 부트 옵션에 대한 자세한 내용은 19 장, "비전역 영역 설치, 부트, 종료, 정지, 제거 및 복제(작업)"를 참조하십시오.

solaris10 브랜드 영역을 다른 호스트로 마이그레이션

`detach` 및 `attach` 하위 명령과 함께 `zoneadm` 명령을 사용하여 `solaris10` 영역을 다른 호스트로 마이그레이션할 수 있습니다. 이 프로세스는 305 페이지 "영역 마이그레이션 정보" 및 306 페이지 "ZFS 아카이브를 사용하여 비전역 영역을 마이그레이션하는 방법"에 설명되어 있습니다.

`zoneadm attach -a` 명령은 물리적 시스템의 아카이브가 **아닌** 영역의 아카이브를 가져옵니다.

용어집

Fair Share Scheduler 할당 수를 기반으로 CPU 시간을 할당할 수 있는 예약 클래스로 FSS라고도 합니다. 할당은 프로젝트에 할당되는 시스템의 CPU 리소스 양을 정의합니다.

FSS Fair Share Scheduler를 참조하십시오.

Oracle Solaris 10 Zones Oracle Solaris 11 릴리스를 실행하는 시스템의 `solaris10` 브랜드 영역에서 실행되는 Solaris 10 응용 프로그램의 완벽한 런타임 환경입니다.

Oracle Solaris Zones 운영 체제 시스템 서비스를 가상화하는 데 사용되는 소프트웨어 분할 기술로, 응용 프로그램을 실행하기 위한 격리된 보안 환경을 제공합니다.

rcapd (resource capping daemon) 리소스 한도가 정의된 프로젝트에서 실행하는 프로세스의 물리적 메모리의 사용량을 조정하는 데몬입니다.

RSS 상주 메모리 페이지 세트 크기를 참조하십시오.

WSS 작업 세트 크기를 참조하십시오.

기본 풀 풀을 사용할 수 있는 경우 시스템에서 만든 풀입니다.

리소스 풀을 참조하십시오.

기본 프로세서 세트 풀을 사용할 수 있는 경우 시스템에서 만든 프로세서 세트입니다.

프로세서 세트를 참조하십시오.

데이터 링크 OSI 프로토콜 스택의 계층 2에 있는 인터페이스로, 시스템에서 STREAMS DLPI(v2) 인터페이스로 표현됩니다. 이 인터페이스는 TCP/IP 등 프로토콜 스택에서 연결될 수 있습니다. Solaris 10 영역 컨텍스트에서는 데이터 링크가 물리적 인터페이스, 통합 또는 VLAN 태그된 인터페이스입니다. 데이터 링크는 예를 들어 NIC 또는 VNIC를 참조할 때 물리적 인터페이스로 참조할 수도 있습니다.

동적 구성 특정 시점에 지정된 시스템에 대한 리소스 풀 프레임워크 내에서 리소스 처리에 대한 정보입니다.

동적 재구성 SPARC 기반 시스템에서 시스템이 실행 중인 동안 하드웨어를 다시 구성하는 기능입니다. DR이라고도 합니다.

로컬 범위 프로세스에 대해 수행되는 로컬 동작으로, 제어값을 초과하려고 시도합니다.

리소스 응용 프로그램 동작을 변경할 목적으로 조작할 수 있는 컴퓨팅 시스템 요소입니다.

리소스 관리 응용 프로그램의 시스템 리소스 사용을 제어할 수 있는 기능입니다.

리소스 분할 영역 리소스의 배타적 부분을 말합니다. 리소스의 모든 분할 영역의 합은 실행 중인 단일 Solaris 인스턴스에서 사용할 수 있는 리소스의 총 크기입니다.

리소스 세트 프로세스 바인드가 가능한 리소스입니다. 대체로 특정 분할 형태를 제공하는 커널 부속 시스템에서 생성된 객체를 나타내는 데 사용됩니다. 리소스 세트의 예로는 예약 클래스 및 프로세서 세트 등이 있습니다.

리소스 소비자 기본적으로 Solaris 프로세스를 의미합니다. 프로젝트와 작업 등의 프로세스 모델 엔티티는 통합 리소스 사용량 측면에서 리소스 사용량을 다루는 방법을 제공합니다.

리소스 제어 리소스 사용량을 프로세스별, 작업별 또는 프로젝트별로 제한합니다.

리소스 풀 시스템 리소스를 분할하는 데 사용되는 구성 방식입니다. 리소스 풀은 분할할 수 있는 리소스 그룹 간의 연관을 나타냅니다.

메모리 상한값 적용 임계치 rcapd(Resource Capping Daemon)에서 상한값 적용을 트리거하는 시스템의 물리적 메모리 사용량 백분율입니다.

분리 세트의 구성원이 겹치지 않고 중복되지 않는 세트 유형입니다.

브랜드 BrandZ 기능의 인스턴스로, 응용 프로그램 실행에 사용되는 고유하지 않은 운영 환경을 포함하는 비전역 영역을 제공합니다.

브랜드 영역 비전역 영역에서 고유하지 않은 응용 프로그램을 실행할 격리된 환경입니다.

블레스 (blessed) Perl에서 객체의 클래스 구성원을 나타내는 데 사용되는 용어입니다.

비전역 영역 Oracle Solaris 운영 체제의 단일 인스턴스 내에서 생성된 가상 운영 체제 환경입니다. Oracle Solaris Zones 소프트웨어 분할 기술은 운영 체제 서비스를 가상화하는 데 사용됩니다.

비전역 영역 관리자 영역 관리자를 참조하십시오.

상주 메모리 페이지 세트 크기 상주하는 메모리 페이지 세트의 크기입니다. 상주 메모리 페이지 세트란 물리적 메모리에 상주하는 페이지 세트를 말합니다.

상한값 시스템 리소스 사용량에 대한 제한입니다.

상한값 설정 시스템 리소스 사용량에 대한 제한을 설정하는 프로세스입니다.

스캐너 자주 사용되지 않은 페이지를 식별하는 커널 스레드입니다. 사용 가능한 메모리의 양이 적은 상태에서는 스캐너가 최근에 사용되지 않은 페이지를 재생 이용합니다.

영역 관리자 영역 관리자의 권한은 비전역 영역으로 제한됩니다.

전역 관리자를 참조하십시오.

영역 상태 비전역 영역의 상태입니다. 영역 상태는 구성됨, 완전하지 않음, 설치됨, 준비, 실행 중 또는 종료 중 상태 중의 하나입니다.

이름 지정 서비스 데이터베이스 이 설명서의 프로젝트 및 작업(개요) 단원에 나오는 LDAP 컨테이너와 NIS 맵에 대한 참조입니다.

읽기 전용 영역 읽기 전용 루트로 구성된 영역입니다.

작업 리소스 관리 시 시간 경과에 따른 일련의 작업을 나타내는 프로세스 집합을 말합니다. 각 작업은 하나의 프로젝트와 연관됩니다.

작업 부하 응용 프로그램 또는 응용 프로그램 그룹의 모든 프로세스를 총칭하는 용어입니다.

작업 세트 크기 작업 세트의 크기입니다. 작업 세트는 처리 주기 중에 프로젝트 작업 부하에서 활발하게 사용하는 페이지 세트를 말합니다.

잠긴 메모리 페이징할 수 없는 메모리입니다.

전역 관리자 루트 사용자 또는 루트 역할을 담당하는 관리자입니다. 전역 영역에 로그인한 경우 전역 관리자나 해당 권한이 부여된 사용자는 시스템 전체를 모니터링하고 제어할 수 있습니다.

영역 관리자를 참조하십시오.

전역 범위 시스템의 모든 리소스 제어에 대한 리소스 제어 값을 적용하는 동작입니다.

전역 영역 모든 Oracle Solaris 시스템에 포함된 영역입니다. 비전역 영역이 사용 중인 경우 전역 영역은 시스템의 기본 영역이자 시스템 전반 관리 제어에 사용되는 영역입니다.

비전역 영역을 참조하십시오.

전체 루트 영역 필요한 모든 시스템 소프트웨어와 추가 패키지가 영역의 개인 파일 시스템에 설치되는 비전역 영역 유형입니다.

정적 풀 구성 관리자가 리소스 풀 기능과 관련하여 시스템을 구성하고자 하는 방법을 나타냅니다.

페이지 아웃 페이지를 물리적 메모리 외부에 있는 영역으로 이동하는 것을 말합니다.

페이지 인 한 번에 한 페이지씩 파일의 데이터를 물리적 메모리로 읽어오는 것을 말합니다.

풀 리소스 풀을 참조하십시오.

풀 데몬 동적 리소스 할당이 필요할 때 활성화되는 `poold` 시스템 데몬입니다.

프로세서 세트 CPU 분리 그룹입니다. 각 프로세서 세트에는 0개 이상의 프로세서가 포함될 수 있습니다. 프로세서 세트 하나는 리소스 풀 구성에서 리소스 요소로 표현됩니다. pset이라고도 합니다.

분리를 참조하십시오.

프로젝트 관련 작업에 대한 네트워크 차원의 관리 식별자입니다.

확장 계정 Solaris 운영 체제에서 작업이나 프로세스를 기반으로 리소스 사용량을 기록할 수 있는 유연한 방법입니다.

힙 프로세스에서 할당된 스크래치 메모리입니다.

색인

V

Z

공

구

권

기