Oracle® Solaris 11.2 패키지 저장소 복사 및 만들기

부품 번호: E53762-02
2014년 9월

ORACLE

목차

코드 예

이 설명서 사용

- **개요** - Oracle Solaris IPS(이미지 패키징 시스템) 기능을 사용하여 소프트웨어 패키지 저장소를 만들고, 복사, 액세스 설정, 업데이트 및 유지 관리하는 방법을 설명합니다.
- **대상** - 소프트웨어를 설치 및 관리하거나 소프트웨어를 설치 및 관리하는 다른 사용자를 지원하는 시스템 관리자입니다.
- **필요한 지식** - Oracle Solaris SMF(서비스 관리 기능) 기능 경력과 NFS 및 웹 서버 관리 경력

제품 설명서 라이브러리

이 제품에 대한 최신 정보 및 알려진 문제는 설명서 라이브러리(http://www.oracle.com/pls/topic/lookup?ctx=E56343)에서 확인할 수 있습니다.

Oracle 지원 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.

피드백

http://www.oracle.com/goto/docfeedback에서 이 설명서에 대한 피드백을 보낼 수 있습니다.

♦ ♦ ♦ 1 장

1 이미지 패키징 시스템 패키지 저장소

Oracle Solaris 11 소프트웨어는 IPS(이미지 패키징 시스템) 패키지로 배포됩니다. IPS 패키지는 IPS 게시자가 채우는 IPS 패키지 저장소에 저장됩니다.

이 설명서에서는 Oracle Solaris IPS(이미지 패키징 시스템) 기능을 사용하여 소프트웨어 패키지 저장소를 만드는 방법을 설명합니다. IPS 도구를 사용하면 기존 저장소를 쉽게 복사하거나 고유 패키지에 대해 고유한 저장소를 만들고, 저장소에서 패키지를 쉽게 업데이트할 수 있습니다. 저장소 사용자에게 파일 인터페이스나 HTTP 또는 HTTPS 인터페이스를 제공할 수 있습니다. 또한 이 설명서에서는 자동으로 저장소를 업데이트하는 방법 및 저장소를 복제하는 방법을 설명하며 캐싱, 로드 균형 조정, HTTPS 액세스 구성 등의 Apache 웹 서버 구성을 보여 줍니다.

이 장의 내용은 다음과 같습니다.

- 내부 용도로 로컬 IPS 패키지 저장소를 만들어야 하는 이유
- 패키지 저장소를 만들기 위한 모범 사례
- 저장소 호스트를 위한 시스템 요구 사항

로컬 IPS 저장소

로컬 IPS 저장소가 필요할 수 있는 이유는 다음과 같습니다.

- **성능 및 보안.** 클라이언트 시스템이 인터넷으로 이동하여 새 소프트웨어 패키지를 검색하거나 기존 패키지를 업데이트하지 못하도록 해야 할 수 있습니다.
- **변경 제어.** 오늘 수행한 설치 작업을 내년에도 동일하게 수행할 수 있도록 해야 할 수 있습니다. 시스템을 업데이트할 수 있는 버전을 쉽게 제어하려고 합니다.
- **사용자 정의 패키지.** 사용자 정의 IPS 패키지를 제공하려고 합니다.

로컬 IPS 패키지 저장소를 만들고 사용하기 위한 모범 사례

다음 모범 사례에 따라 저장소 가용성을 유지 관리하고 오류를 최소화합니다.

모든 SRU(Support Repository Update)의 컨텐츠를 모두 포함합니다.

모든 지원 업데이트로 로컬 저장소가 업데이트되도록 유지합니다. 지원 업데이트에는 보안 업데이트 및 기타 중요한 수정 사항이 포함되어 있습니다. Oracle Solaris OS 패키지 저장소의 각 부 버전 및 업데이트는 전체 패키지 세트로 릴리스됩니다. SRU는 변경된 패키지만 포함된 스파스 업데이트로 릴리스됩니다.

- 지원 업데이트의 패키지 하위 세트를 저장소에 추가하지 마십시오. 지원 업데이트의 모든 컨텐츠를 로컬 저장소에 추가합니다.
- 지원 업데이트를 건너뛰지 마십시오. 해당하는 모든 지원 업데이트를 각 저장소에 누적합니다.
- Oracle 게시자가 제공한 패키지를 제거하지 마십시오.
- `svc:/application/pkg/mirror` SMF(서비스 관리 기능) 서비스를 사용하여 오라클 고객 지원 센터 저장소에서 로컬 마스터 저장소를 자동으로 업데이트합니다. 지침은 인터넷에서 자동으로 저장소를 복사하는 방법 [20]을 참조하십시오.

사용자는 설치할 `entire` 통합 패키지 버전을 지정하여 저장소의 최신 버전보다 이전 버전으로 업데이트할 수 있습니다. “Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트”의 4 장, “Oracle Solaris 이미지 업데이트 또는 업그레이드”를 참조하십시오.

저장소를 업데이트할 때마다 확인합니다.

저장소의 컨텐츠 또는 등록 정보 값을 변경할 때마다 `pkgrepo verify` 명령을 사용합니다. `pkgrepo verify` 명령은 저장소 컨텐츠의 다음 속성이 올바른지 확인합니다.

- 파일 체크섬.
- 파일 사용 권한. 저장소 파일 및 디렉토리와 저장소까지의 경로를 검사하여 `pkg5srv` 사용자가 저장소 컨텐츠를 읽을 수 있는지 확인합니다.
- 패키지 매니페스트 사용 권한.
- 패키지 매니페스트 컨텐츠.
- 패키지 서명.

공유 위치에 저장소를 만듭니다.

공유 위치는 BE(부트 가능 환경)에 없는 위치입니다. 공유 위치의 예로 `/var/share` 및 `/export`가 있습니다. 공유 위치에 저장소를 만들면 다음과 같은 이점이 있습니다.

- 다른 기존 BE에서 저장소를 쉽게 사용할 수 있습니다.
- 업그레이드를 통해 또는 기존 BE를 복제하여 새 BE를 만드는 경우 저장소 복사본을 여러 개 구현하여 공간을 낭비하지 않습니다.
- 다른 BE에서 이미 수행한 저장소 업데이트를 다시 적용하는 데 시간과 I/O 리소스를 낭비하지 않습니다.

비전역 영역을 사용 중인 경우 게시자가 전역 영역에 구성되지 않은 경우에도 비전역 영역에 구성된 게시자의 모든 위치를 전역 영역에서 액세스할 수 있어야 합니다.

해당 ZFS 파일 시스템에 각 저장소를 만듭니다.

개별 ZFS 파일 시스템을 사용하면 다음을 수행할 수 있습니다.

- 향상된 성능을 얻을 수 있습니다.

- 파일 시스템 특성을 개별적으로 설정할 수 있습니다. 예를 들어, 저장소를 업데이트할 때 성능 향상을 위해 `atime`을 `off`로 설정합니다. `atime` 등록 정보는 파일을 읽을 때 파일의 액세스 시간을 업데이트할지 여부를 제어합니다. 이 등록 정보를 off로 설정하면 파일을 읽을 때 쓰기 트래픽이 발생하지 않습니다.
- 리소스 사용을 관리할 수 있습니다. 각 저장소 데이터 세트에 적합한 디스크 할당량을 지정하여 큰 저장소 업데이트가 풀의 모든 공간을 사용하지 않도록 합니다. 이 모범 사례는 인터넷에서 자동으로 저장소를 복사하는 방법 [20]의 설명에 따라 자동으로 업데이트를 수행하는 경우에 특히 중요합니다.
- 스냅샷을 만들 수 있습니다.

저장소를 업데이트할 때마다 스냅샷을 작성합니다.

다음과 같은 이점을 얻으려면 저장소를 업데이트할 때마다 저장소 파일 시스템의 스냅샷을 작성합니다.

- 스냅샷에서 이전 버전의 저장소로 롤백합니다.
- 스냅샷에서 저장소를 업데이트하여 사용자 방해를 최소화합니다.

고가용성을 제공합니다.

- 각기 다른 위치에 저장소 복제본을 유지 관리합니다. 지침은 "여러 개의 동일한 로컬 저장소 유지 관리" [32]를 참조하십시오.
- 캐싱, 로드 균형 조정 및 여러 저장소 제공을 위해 웹 서버를 구성합니다. 자세한 내용은 5장. 웹 서버 뒤에서 저장소 서버 실행을 참조하십시오.

로컬 저장소의 보안을 설정합니다.

지침은 "HTTPS 저장소 액세스 구성" [51]을 참조하십시오.

시스템 요구 사항

IPS 패키지 저장소를 호스트하는 시스템은 x86 기반 시스템 또는 SPARC 기반 시스템일 수 있습니다.

운영 체제

Oracle Solaris 11 11/11을 실행하는 저장소 서버는 모든 Oracle Solaris 11 업데이트 패키지를 지원합니다.

디스크 공간

Oracle Solaris 11.2 릴리스 저장소의 복사본을 호스트하려면 저장소 서버에 16GB의 여유 공간이 있어야 합니다.

모범 사례는 모든 지원 업데이트로 로컬 저장소가 업데이트되도록 유지하는 것이므로 매년 지원 업데이트에 추가 공간의 10-15GB를 사용하도록 계획합니다. 물론, Oracle Solaris Studio 또는 Oracle Solaris Cluster 같은 추가 소프트웨어는 패키지 저장소에 추가 공간이 필요합니다.

하나의 시스템에 두 개 이상의 IPS 저장소가 호스트되는 경우 각 저장소를 개별적으로 롤백하고 복구할 수 있도록 각 저장소를 개별적인 ZFS 파일 시스템으로 만듭니다.

저장소 관리 권한

패키지 저장소를 만들고 구성하는 데 필요한 권한을 얻으려면 다음 방법 중 하나를 사용합니다. 필요한 프로파일 또는 역할을 확인하는 방법을 포함하여 프로파일 및 역할에 대한 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안 "을 참조하십시오.

권한 프로파일

`profiles` 명령을 사용하여 지정 받은 권한 프로파일을 나열합니다. 다음 프로파일은 로컬 패키지 저장소를 유지 관리하는 데 유용합니다.

ZFS File System Management

이 권한 프로파일을 사용하면 `zfs` 명령을 실행할 수 있습니다.

Software Installation

이 권한 프로파일을 사용하면 `pkg` 명령을 실행할 수 있습니다.

Service Management

이 권한 프로파일을 사용하면 `svccfg` 등의 SMF 명령을 실행할 수 있습니다.

역할

`roles` 명령을 사용하여 지정 받은 역할을 나열합니다. `root` 역할이 있는 경우 `root` 암호와 함께 `su` 명령을 사용하여 `root` 역할을 가정할 수 있습니다.

sudo 명령

사이트의 보안 정책에 따라 사용자 암호와 함께 `sudo` 명령을 사용하여 권한이 있는 명령을 실행할 수 있는 경우도 있습니다.

♦ ♦ ♦ 2 장

2 IPS 패키지 저장소 복사

이 장에서는 Oracle Solaris IPS 패키지 저장소 복사본을 만드는 두 가지 방법을 설명합니다. 매체 또는 Oracle Solaris 다운로드 사이트의 저장소 파일을 사용하거나 인터넷에서 수동 또는 자동으로 저장소 컨텐츠를 검색할 수 있습니다. 어떤 경우든지 공유 위치에 로컬 패키지 저장소에 대한 개별 ZFS 파일 시스템을 먼저 만듭니다. 저장소가 만들어진 후 저장소를 확인하고 스냅샷을 작성합니다.

이 장에서는 저장소 복사와 관련된 성능 및 문제 해결 정보도 제공합니다.

저장소 복사에 대한 성능 고려 사항

Oracle Solaris 다운로드 사이트에서 저장소 파일을 다운로드하는 경우 또는 “인터넷에서 저장소 복사” [18]에 표시된 `pkgrecv` 명령을 사용하여 인터넷 위치에서 저장소 컨텐츠를 검색하는 경우 전송 성능을 향상시키기 위해 다음 구성을 고려하십시오.

- ZFS 저장소 풀 용량이 80% 미만인지 확인합니다. `zpool list` 명령을 사용하여 풀 용량을 확인합니다.
- 프록시를 사용 중인 경우 프록시 성능을 확인합니다.
- 다량의 메모리를 사용하는 응용 프로그램을 닫습니다.
- 임시 디렉토리에서 적절한 여유 공간을 사용할 수 있는지 확인합니다. 작업 중 `pkgrecv` 명령은 `$TMPDIR`을 임시 저장소 디렉토리로 사용합니다. `TMPDIR`이 설정되지 않은 경우 `pkgrecv`는 이 임시 저장소에 `/var/tmp`를 사용합니다. `$TMPDIR` 또는 `/var/tmp`에 수행 중인 `pkgrecv` 작업 크기에 충분한 여유 공간이 있는지 확인합니다.
- `pkgrecv` 명령을 사용하여 큰 저장소를 복사하는 경우 `--clone` 옵션을 사용하는 것이 좋습니다. `--clone` 옵션을 사용하면 속도가 향상되고 메모리 사용량이 감소합니다. 로컬 IPS 패키지 저장소를 복제하는 방법 [33]을 참조하십시오.
- `pkgrecv` 명령을 사용하여 큰 저장소를 만들거나 업데이트하는 경우 대상 저장소에 SSD를 사용하는 것이 좋습니다. 패키지 검색이 완료된 후 필요에 따라 저장소를 이동할 수 있습니다.

로컬 패키지 저장소 문제 해결

다음 방법은 문제를 방지하거나 발생할 수 있는 문제의 원인을 찾는 데 유용할 수 있습니다.

- 저장소 소스 파일을 확인합니다. `.zip` 파일을 사용하여 저장소를 만드는 경우 `zip` 파일에서 저장소를 복사하는 방법 [14]에 설명된 대로 체크섬을 사용하여 시스템의 파일이 올바른지 확인합니다.
- 설치된 저장소를 확인합니다. `pkgrepo verify` 명령을 사용하여 설치된 저장소를 확인합니다.

  `pkgrepo verify`에서 보고되는 사용 권한 문제는 다음과 같습니다.

  - 파일 사용 권한. 파일 시스템 기반 저장소에 대한 디렉토리 및 파일 사용 권한 문제를 방지하려면 `pkg5srv` 사용자에게 저장소를 읽을 권한이 있는지 확인합니다.
  - 디렉토리 사용 권한. 저장소의 모든 디렉토리에 실행 권한이 있는지 확인합니다.

  `pkgrepo verify` 명령에서 다른 유형의 오류를 보고하는 경우 `pkgrepo fix` 명령을 사용하여 오류를 수정하십시오. 자세한 내용은 `pkgrepo`(1) 매뉴얼 페이지를 참조하십시오.
- 게시자 원본을 확인합니다. 각 이미지에서 각 게시자의 원본을 적절하게 설정했는지 확인합니다. 설치된 패키지를 업데이트하거나, 설치된 패키지에 따라 달라지는 패키지를 설치하거나, 비전역 영역을 설치하려면 게시자 원본으로 설정하는 저장소에 적어도 게시자를 설정할 이미지에 설치된 것과 동일한 소프트웨어가 포함되어 있어야 합니다. 사용자가 파일 인터페이스를 사용하여 패키지를 검색하도록 설정하는 방법 [23]의 3단계를 참조하십시오. 게시자 설정 및 패키지 설치 문제 해결에 대한 자세한 내용은 “Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트 ”를 참조하십시오.
- 웹 서버 구성을 확인합니다. 저장소에 액세스하도록 Apache 웹 서버를 구성하는 경우 인코딩된 슬래시를 디코딩하지 않도록 웹 서버를 구성합니다. “필요한 Apache 구성 설정” [46]의 지침을 참조하십시오. 인코딩된 슬래시를 디코딩하면 "패키지를 찾을 수 없음" 오류가 발생할 수 있습니다.
- 비전역 영역에서만 액세스할 수 있는 저장소를 만들지 마십시오. 게시자가 전역 영역에 구성되지 않은 경우에도 비전역 영역에 구성된 게시자의 모든 위치를 전역 영역에서 액세스할 수 있어야 합니다.

파일에서 저장소 복사

이 절에서는 하나 이상의 저장소 파일에서 Oracle Solaris 패키지 저장소의 로컬 복사본을 만드는 방법을 설명합니다. 저장소 파일은 매체에 있거나 Oracle Solaris 다운로드 사이트에서 사용할 수 있습니다. 저장소 파일은 `zip` 파일 또는 `iso` 파일일 수 있습니다.

▼ `zip` 파일에서 저장소를 복사하는 방법

1. **새 저장소에 대한 ZFS 파일 시스템을 만듭니다.**

공유 위치에 저장소를 만듭니다. 저장소 파일 시스템을 만들 때 atime을 off로 설정합니다. "로컬 IPS 패키지 저장소를 만들고 사용하기 위한 모범 사례" [9]를 참조하십시오.

```
$ zfs create -o atime=off rpool/export/IPSpkgrepos
$ zfs create rpool/export/IPSpkgrepos/Solaris
$ zfs get atime rpool/export/IPSpkgrepos/Solaris
NAME                              PROPERTY  VALUE  SOURCE
rpool/export/IPSpkgrepos/Solaris  atime     off    inherited from rpool/export/IPSpkgrepos
```

2. **패키지 저장소 파일을 가져옵니다.**

   시스템 설치 이미지를 다운로드한 곳과 동일한 위치에서 Oracle Solaris IPS 패키지 저장소 .zip 파일을 다운로드하거나 매체 패킷에서 저장소 DVD를 찾습니다. .zip 파일과 함께 install-repo.ksh 스크립트와 .txt 파일(README 및 체크섬 파일)을 다운로드합니다.

```
$ ls
install-repo.ksh           sol-11_2-ga-repo-3of4.zip
README-zipped-repo.txt     sol-11_2-ga-repo-4of4.zip
sol-11_2-ga-repo-1of4.zip  sol-11_2-ga-repo.txt
sol-11_2-ga-repo-2of4.zip
```

3. **스크립트 파일이 실행 파일인지 확인합니다.**

```
$ chmod +x install-repo.ksh
```

4. **저장소 설치 스크립트를 실행합니다.**

   저장소 설치 스크립트 install-repo.ksh는 지정된 디렉토리에 각 저장소 .zip 파일의 압축을 풉니다. 선택적으로 스크립트는 다음과 같은 추가 작업을 수행합니다.

   - 다운로드한 .zip 파일의 체크섬을 확인합니다. -c 옵션을 지정하여 체크섬을 확인하지 않을 경우 저장소 설치 스크립트를 실행하기 전에 수동으로 체크섬을 확인합니다. 다음 digest 명령을 실행하고 그 출력을 .md5 파일의 해당 체크섬과 비교합니다.

     ```
     $ digest -a md5 file
     ```

   - 지정된 대상에 저장소가 이미 포함되어 있는 경우 기존 컨텐츠에 저장소 컨텐츠를 추가합니다.
   - 최종 저장소를 확인합니다. -v 옵션을 지정하여 저장소를 확인하지 않을 경우 저장소 설치 스크립트를 실행한 후 pkgrepo 명령의 info, list 및 verify 하위 명령을 사용하여 저장소를 확인합니다.
   - 마운트 및 배포할 ISO 이미지 파일을 만듭니다. -I 옵션을 사용하여 .iso 파일을 만드는 경우 .iso 파일 및 .iso 파일을 사용하는 방법을 설명하는 README 파일이 지정된 대상 디렉토리에 있습니다.

5. **저장소 컨텐츠를 확인합니다.**

   이전 단계에서 -v 옵션을 지정하지 않은 경우 pkgrepo 명령의 info, list 및 verify 하위 명령을 사용하여 저장소가 올바르게 복사되었는지 확인합니다. pkgrepo verify 명령에서 오류를 보고하는 경우 pkgrepo fix 명령을 사용하여 오류를 수정하십시오. 자세한 내용은 pkgrepo(1) 매뉴얼 페이지를 참조하십시오.

6. **새 저장소의 스냅샷을 작성합니다.**

```
$ zfs snapshot rpool/export/IPSpkgrepos/Solaris@sol-11_2_0
```

예 2-1 zip 파일에서 새 저장소 만들기

이 예에서는 zip 파일의 압축을 풀 때까지 저장소가 존재하지 않습니다. 이 스크립트는 다음 옵션을 사용할 수 있습니다.

-s	선택 사항. .zip 파일이 있는 디렉토리의 전체 경로를 지정합니다. 기본값: 현재 디렉토리.
-d	필수. 저장소를 저장할 디렉토리의 전체 경로를 지정합니다.
-i	선택 사항. 이 저장소를 채우는 데 사용할 파일을 지정합니다. 소스 디렉토리에 .zip 파일 세트가 여러 개 포함되어 있을 수 있습니다. 기본값: 소스 디렉토리에서 사용 가능한 최신 이미지.
-c	선택 사항. .zip 파일의 체크섬과 지정된 파일의 체크섬을 비교합니다. 인수 없이 -c를 지정하는 경우 사용되는 기본 파일은 소스 디렉토리의 -i 이미지에 대한 .md5 파일입니다.
-v	선택 사항. 최종 저장소를 확인합니다.
-I	선택 사항. 소스 디렉토리에 저장소의 ISO 이미지를 만듭니다. 또한 mkiso.log 로그 파일을 소스 디렉토리에 그대로 둡니다.
-h	선택 사항. 사용법 메시지를 표시합니다.

```
$ ./install-repo.ksh -d /export/IPSpkgrepos/Solaris -c -v -I
Comparing checksums of downloaded files...done. Checksums match.
Uncompressing sol-11_2-ga-repo-1of4.zip...done.
Uncompressing sol-11_2-ga-repo-2of4.zip...done.
Uncompressing sol-11_2-ga-repo-3of4.zip...done.
Uncompressing sol-11_2-ga-repo-4of4.zip...done.
Repository can be found in /export/IPSpkgrepos/Solaris.
Initiating repository verification.
Building ISO image...done.
ISO image and instructions for using the ISO image are at:
/tank/downloads/sol-11_2-ga-repo.iso
/tank/downloads/README-repo-iso.txt
$ ls /export/IPSpkgrepos/Solaris
COPYRIGHT         NOTICES           pkg5.repository   publisher         README-iso.txt
```

저장소 재구성 및 확인에 시간이 걸릴 수 있지만 "저장소를 찾을 수 있음" 메시지가 표시된 후 저장소 컨텐츠를 검색할 수 있습니다.

예 2-2 zip 파일에서 기존 저장소에 추가

이 예에서는 저장소 zip 파일의 컨텐츠가 기존 패키지 저장소의 컨텐츠에 추가됩니다.

```
$ pkgrepo -s /export/IPSpkgrepos/Solaris info
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4764     online           2014-03-18T05:30:57.221021Z
$ ./install-repo.ksh -d /export/IPSpkgrepos/Solaris -c -v -I
IPS repository exists at destination /export/IPSpkgrepos/Solaris
Current version: 0.175.2.0.0.35.0
Do you want to add to this repository? (y/n) y
Comparing checksums of downloaded files...done. Checksums match.
Uncompressing sol-11_2-ga-repo-1of4.zip...done.
Uncompressing sol-11_2-ga-repo-2of4.zip...done.
Uncompressing sol-11_2-ga-repo-3of4.zip...done.
Uncompressing sol-11_2-ga-repo-4of4.zip...done.
Repository can be found in /export/IPSpkgrepos/Solaris.
Initiating repository rebuild.
Initiating repository verification.
Building ISO image...done.
ISO image and instructions for using the ISO image are at:
/tank/downloads/sol-11_2-ga-repo.iso
/tank/downloads/README-repo-iso.txt
$ pkgrepo -s /export/IPSpkgrepos/Solaris info
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-06-02T18:11:55.640930Z
```

▼ iso 파일에서 저장소를 복사하는 방법

1. **새 저장소에 대한 ZFS 파일 시스템을 만듭니다.**

   공유 위치에 저장소를 만듭니다. 저장소 파일 시스템을 만들 때 `atime`을 `off`로 설정합니다. "로컬 IPS 패키지 저장소를 만들고 사용하기 위한 모범 사례" [9]를 참조하십시오.

   ```
   $ zfs create -o atime=off rpool/export/IPSpkgrepos
   $ zfs create rpool/export/IPSpkgrepos/Solaris
   $ zfs get atime rpool/export/IPSpkgrepos/Solaris
   NAME                              PROPERTY  VALUE  SOURCE
   rpool/export/IPSpkgrepos/Solaris  atime     off    inherited from rpool/export/IPSpkgrepos
   ```

2. **패키지 저장소 이미지 파일을 가져옵니다.**

   예 2-1. "zip 파일에서 새 저장소 만들기"에 설명된 대로 `-I` 옵션을 사용하여 저장소 `.zip` 파일로부터 `.iso` 파일을 만듭니다.

3. **이미지 파일을 마운트합니다.**

   저장소 `.iso` 파일을 마운트하여 컨텐츠에 액세스합니다.

   ```
   $ mount -F hsfs /path/sol-11_2-repo.iso /mnt
   ```

   저장소 서버 시스템이 다시 시작될 때마다 `.iso` 이미지를 다시 마운트할 필요가 없도록 하려면 다음 단계의 설명에 따라 저장소 파일 컨텐츠를 복사합니다.

4. **저장소 컨텐츠를 새 위치에 복사합니다.**

각 저장소 액세스 성능을 향상시키고 시스템이 다시 시작될 때마다 .iso 이미지를 다시 마운트할 필요가 없도록 하려면 /mnt/repo/에서 저장소 파일을 ZFS 파일 시스템으로 복사합니다. 이 복사 작업은 rsync 명령이나 tar 명령을 통해 수행할 수 있습니다.

- **rsync 명령을 사용합니다.**

  rsync 명령을 사용할 경우 repo 디렉토리에 있는 파일과 하위 디렉토리를 복사하려면 /mnt/repo가 아니라 /mnt/repo/(후행 슬래시 문자 포함)로 지정해야 합니다. rsync(1) 매뉴얼 페이지를 참조하십시오.

  ```
  $ rsync -aP /mnt/repo/ /export/IPSpkgrepos/Solaris
  ```

- **tar 명령을 사용합니다.**

  다음 예와 같이 tar 명령을 사용하면 마운트된 파일 시스템의 저장소를 저장소 ZFS 파일 시스템에 빠르게 복사할 수 있습니다.

  ```
  $ cd /mnt/repo; tar cf - . | (cd /export/IPSpkgrepos/Solaris; tar xfp -)
  ```

5. **이미지 파일을 마운트 해제합니다.**

   여전히 /mnt 디렉토리에 있지 않은지 확인합니다.

   ```
   $ umount /mnt
   ```

6. **새 저장소 컨텐츠를 확인합니다.**

   pkgrepo 명령의 info, list 및 verify 하위 명령을 사용하여 저장소가 올바르게 복사되었는지 확인합니다. pkgrepo verify 명령에서 오류를 보고하는 경우 pkgrepo fix 명령을 사용하여 오류를 수정하십시오. 자세한 내용은 pkgrepo(1) 매뉴얼 페이지를 참조하십시오.

7. **새 저장소의 스냅샷을 작성합니다.**

   ```
   $ zfs snapshot rpool/export/IPSpkgrepos/Solaris@sol-11_2_0
   ```

인터넷에서 저장소 복사

이 절에서는 인터넷 위치에서 저장소를 복사하여 Oracle Solaris 릴리스 패키지 저장소의 로컬 복사본을 만드는 방법을 설명합니다. 첫번째 절차에서는 명령줄에서 복사 명령을 실행하는 방법을 보여 줍니다. 두번째 절차에서는 SMF 서비스를 사용하여 자동으로 저장소를 복사 및 업데이트하는 방법을 보여 줍니다.

▼ 인터넷에서 명시적으로 저장소를 복사하는 방법

1. **새 저장소에 대한 ZFS 파일 시스템을 만듭니다.**

공유 위치에 저장소를 만듭니다. 저장소 파일 시스템을 만들 때 `atime`을 `off`로 설정합니다. "로컬 IPS 패키지 저장소를 만들고 사용하기 위한 모범 사례" [9]를 참조하십시오.

```
$ zfs create -o atime=off rpool/export/IPSpkgrepos
$ zfs create rpool/export/IPSpkgrepos/Solaris
$ zfs get atime rpool/export/IPSpkgrepos/Solaris
NAME                              PROPERTY  VALUE  SOURCE
rpool/export/IPSpkgrepos/Solaris  atime     off    inherited from rpool/export/IPSpkgrepos
```

2. **필요한 저장소 기반구조를 만듭니다.**

   저장소를 복사할 수 있도록 필요한 pkg(5) 저장소 기반구조를 만듭니다. 이전 방법에서 사용한 이미지 파일에는 저장소 기반구조가 포함되어 있으므로 이 단계가 필요하지 않습니다. 이 방법의 설명에 따라 `pkgrecv` 명령을 사용하여 저장소 컨텐츠를 복사하는 경우 저장소 기반구조를 만든 후 저장소 컨텐츠를 해당 기반구조로 복사해야 합니다. pkg(5) 및 pkgrepo(1) 매뉴얼 페이지를 참조하십시오.

   ```
   $ pkgrepo create /export/IPSpkgrepos/Solaris
   ```

3. **저장소 컨텐츠를 새 위치에 복사합니다.**

   `pkgrecv` 명령을 사용하여 저장소를 복사합니다. 이 작업은 네트워크 성능에 영향을 줄 수 있습니다. 이 작업을 완료하는 데 필요한 시간은 네트워크 대역폭 및 연결 속도에 따라 달라집니다. "저장소 복사에 대한 성능 고려 사항" [13]도 참조하십시오. 이 저장소를 나중에 업데이트할 경우 변경 사항만 전송되며 프로세스 시간이 줄어들 수 있습니다.

   다음 명령은 `-s` 옵션으로 지정된 패키지 저장소의 모든 패키지 버전을 `-d` 옵션으로 지정된 저장소로 모두 검색합니다. 보안 사이트에서 복사하는 경우 필요한 SSL 인증서와 키가 설치되어 있는지 확인하고 필요한 인증서 및 키 옵션을 지정합니다.

   ```
   $ pkgrecv -s https://pkg.oracle.com/solaris/support -d /export/IPSpkgrepos/Solaris \
   --key /path-to-ssl_key --cert /path-to-ssl_cert '*'
   ```

   `-m` 및 `--clone` 옵션에 대한 자세한 내용은 pkgrecv(1) 매뉴얼 페이지를 참조하십시오. 이 목적으로 `-m latest` 옵션을 사용하면 안됩니다. 너무 희소한 저장소를 사용하면 사용자가 이미지를 업데이트하려고 할 때 오류가 발생할 수 있습니다.

4. **새 저장소 컨텐츠를 확인합니다.**

   `pkgrepo` 명령의 `info`, `list` 및 `verify` 하위 명령을 사용하여 저장소가 올바르게 복사되었는지 확인합니다. `pkgrepo verify` 명령에서 오류를 보고하는 경우 `pkgrepo fix` 명령을 사용하여 오류를 수정하십시오. 자세한 내용은 pkgrepo(1) 매뉴얼 페이지를 참조하십시오.

5. **새 저장소의 스냅샷을 작성합니다.**

   ```
   $ zfs snapshot rpool/export/IPSpkgrepos/Solaris@sol-11_2_0
   ```

▼ 인터넷에서 자동으로 저장소를 복사하는 방법

기본적으로 svc:/application/pkg/mirror SMF 서비스는 이 이미지에 정의된 solaris 게시자 원본에서 /var/share/pkg/repositories/solaris로 정기적인 pkgrecv 작업을 수행합니다. 이 pkgrecv 작업은 매달 하루 오전 2:30에 시작됩니다. 이 기본 동작을 변경하려면 이 절차의 설명에 따라 서비스를 구성합니다.

이 서비스를 성공적으로 실행할 때마다 저장소 카탈로그가 새로 고쳐집니다. 검색 색인을 작성하기 위해 저장소를 새로 고칠 필요는 없습니다.

이 서비스는 정기적으로 실행되므로 저장소가 만들어지고 계속 업데이트됩니다. 이 문서에 표시된 수동 저장소 업데이트 지침을 사용할 필요는 없습니다.

다른 시스템은 solaris 게시자 옵션을 자동으로 업데이트된 이 저장소나 이 저장소의 복제본으로 설정할 수 있습니다. 하나의 시스템에만 인터넷 게시자 원본이 있고 mirror 서비스를 실행하여 자동으로 업데이트를 받으면 됩니다.

1. **게시자 원본을 설정합니다.**

   기본적으로 mirror 서비스는 /에 루트 지정된 이미지에 구성된 solaris 게시자에서 패키지를 전송합니다. mirror 서비스 구성에 게시자 원본을 직접 지정할 수는 없지만 이 정보를 검색할 이미지 루트를 구성할 수 있습니다. 해당 이미지 루트에서 pkg set-publisher를 사용하여 미러 저장소에 대한 pkgrecv 전송 소스로 사용할 게시자 원본을 구성합니다.

   a. **(선택 사항) 이미지 루트를 설정합니다.**

      미러 서비스에 사용하려는 게시자 구성이 이 이미지에 사용하려는 게시자 구성과 다른 경우 다음 예와 같이 BE에 포함되지 않은 공유 위치에 사용자 이미지를 만들고 mirror 서비스의 config/ref_image 등록 정보 값을 새 이미지로 재설정합니다. mirror 서비스는 config/ref_image 이미지의 게시자 구성을 사용합니다.

      ```
      $ svccfg -s pkg/mirror:default setprop config/ref_image = /var/share/pkg/
      mirror_svc_ref_image
      $ pkg image-create /var/share/pkg/mirror_svc_ref_image
      ```

   b. **(선택 사항) 게시자를 설정합니다.**

      solaris 게시자뿐 아니라 다른 게시자의 패키지로 미러 저장소를 업데이트하려는 경우 ha-cluster 및 solarisstudio 게시자 추가를 보여 주는 다음 예와 같이 mirror 서비스의 config/publishers 등록 정보 값을 재설정합니다.

      ```
      $ svccfg -s pkg/mirror:default setprop config/publishers = solaris,ha-
      cluster,solarisstudio
      ```

   c. **게시자 원본을 설정합니다.**

      이 서비스는 정기적으로 실행되므로 일반 업데이트를 제공하는 저장소로 게시자 원본을 설정해야 합니다. Oracle 제품의 경우 게시자 원본을 지원 저장소로 설정하여 SRU(Support Repository Update)를 검색하는 것이 좋습니다. 다음 예에서 -R 옵션은

대체 이미지 루트에 게시자를 구성하는 경우에만 필요합니다. 원본 URI에 따라 `-k` 및 `-c` 옵션이 필요하지 않을 수도 있습니다.

```
$ pkg -R /var/share/pkg/mirror_svc_ref_image set-publisher \
-g https://pkg.oracle.com/solaris/support/ -k ssl_key -c ssl_cert solaris
$ pkg -R /var/share/pkg/mirror_svc_ref_image set-publisher \
-g https://pkg.oracle.com/ha-cluster/support/ -k ssl_key -c ssl_cert ha-cluster
$ pkg -R /var/share/pkg/mirror_svc_ref_image set-publisher \
-g https://pkg.oracle.com/solarisstudio/support/ -k ssl_key -c ssl_cert solarisstudio
```

다음 명령 중 하나를 사용하여 새 이미지에 구성된 게시자를 확인합니다.

```
$ pkg -R /var/share/pkg/mirror_svc_ref_image publisher
$ pkg -R /var/share/pkg/mirror_svc_ref_image publisher solaris ha-cluster
 solarisstudio
```

2. **(선택 사항) 미러 서비스의 기타 등록 정보를 구성합니다.**

   서비스가 실행되는 시간, 미러 저장소 위치 등 `mirror` 서비스의 기타 등록 정보를 수정할 수도 있습니다.

   미러 중인 게시자 원본이 업데이트되는 시간과 더 일치하도록 서비스 실행 시간을 변경할 수도 있습니다. 서비스 실행 시간을 변경하려면 `config/crontab_period` 등록 정보 값을 수정합니다.

   미러 저장소의 위치를 변경하려면 `config/repository` 등록 정보 값을 수정합니다. 미러 저장소의 위치를 변경하는 경우 공유 위치에 저장소를 유지합니다. "로컬 IPS 패키지 저장소를 만들고 사용하기 위한 모범 사례" [9]를 참조하십시오. 기본 위치 `/var/share/pkg/repositories/solaris`는 BE에 포함되지 않은 공유 위치입니다.

3. **미러 서비스를 사용으로 설정합니다.**

   `svcs mirror` 명령을 사용하여 `mirror` 서비스의 상태를 확인합니다.

   - **서비스가 사용 안함으로 설정되었으며 이 서비스를 사용하려고 합니다.**

     a. **구성을 변경한 경우 서비스 인스턴스를 새로 고칩니다.**

        `mirror` 서비스의 구성을 변경한 경우 이전 단계의 `svccfg setprop` 명령과 같이 서비스를 새로 고쳐 변경된 값을 실행 중인 스냅샷에 커밋합니다. `svcprop -p config mirror` 명령의 출력에 원하는 값이 표시되지 않는 경우 `svccfg -s mirror:default listprop config` 명령의 출력에 원하는 값이 표시되는지 확인합니다. `svcadm refresh mirror:default` 또는 `svccfg -s mirror:default refresh`를 사용하여 변경된 값을 서비스의 실행 중인 스냅샷에 커밋합니다. `svcprop -p config mirror` 명령을 다시 사용하여 원하는 구성 방식으로 서비스가 구성되었는지 확인합니다.

     b. **서비스 인스턴스를 사용으로 설정합니다.**

        다음 명령을 사용하여 미러 서비스를 사용으로 설정합니다.

        ```
        $ svcadm enable mirror:default
        ```

`svcs mirror` 명령을 사용하여 `mirror` 서비스가 온라인 상태인지 확인합니다. `config/crontab_period` 등록 정보에 설정된 시간에 서비스가 실행됩니다.

- **서비스가 온라인 상태이며 지금 서비스를 실행하려고 합니다.**

  서비스가 온라인 상태인 경우 서비스를 새로 고쳐 즉시 실행합니다. `svc-pkg-mirror` 메소드와 `pkgrecv` 명령이 `pkg5srv` 사용자에 의해 실행됩니다.

- **서비스가 온라인 상태이며 이 서비스를 사용하지 않으려고 합니다.**

  `svcadm disable mirror` 명령을 사용하여 이 서비스를 사용 안함으로 설정합니다. 마스터 저장소를 유지 관리하기 위해 한 시스템에서만 이 서비스를 실행할 수도 있습니다. 다른 시스템에서는 이 서비스를 사용 안함으로 설정하는 것이 좋습니다.

- **서비스가 유지 관리 중이거나 성능 저하 상태입니다.**

  `svcs -xvL mirror` 명령을 사용하여 문제 진단 및 해결을 위한 추가 정보를 가져옵니다.

4. **저장소 컨텐츠를 확인합니다.**

   `mirror` 서비스 실행이 완료된 후 `pkgrepo` 명령의 `info`, `list` 및 `verify` 하위 명령을 사용하여 저장소가 올바르게 복사 또는 업데이트되었는지 확인합니다. `pkgrepo verify` 명령에서 오류를 보고하는 경우 `pkgrepo fix` 명령을 사용하여 오류를 수정하십시오. 자세한 내용은 `pkgrepo`(1) 매뉴얼 페이지를 참조하십시오.

   `mirror` 서비스의 `config/crontab_period` 등록 정보 값을 검사하여 서비스가 언제 실행되는지 확인합니다. 서비스가 실행되는 동안 `svcs -p mirror` 명령은 서비스 상태를 `online*`으로 표시하며 이 서비스에서 시작한 프로세스를 보여 줍니다. 저장소를 확인하기 전에 서비스 상태가 `online`으로 표시되고 서비스와 연관된 프로세스가 없을 때까지 기다립니다.

5. **새 저장소의 스냅샷을 작성합니다.**

   ```
   $ zfs snapshot rpool/VARSHARE/pkg/repositories/solaris@sol-11_2_0
   ```

다음 순서 여러 게시자의 컨텐츠를 동시에 복사하지 않으려는 경우도 있습니다. 하나의 `config/publishers` 등록 정보에 여러 게시자를 설정하는 대신 `pkg/mirror` 서비스의 여러 인스턴스를 만들 수 있습니다. 예를 들어, `config/publishers` 등록 정보를 `default` 인스턴스의 경우 `solaris`로 설정하고, 새 `pkg/mirror:ha-cluster` 인스턴스의 경우 `ha-cluster`로 설정하고, 새 `pkg/mirror:solarisstudio` 인스턴스의 경우 `solarisstudio`로 설정할 수 있습니다. 마찬가지로 `config/crontab_period`를 각 인스턴스마다 다르게 설정할 수 있습니다. 이 절차와 같이 각 게시자의 컨텐츠를 한 저장소에 저장하거나 각 `pkg/mirror` 인스턴스에 대해 개별 `config/repository` 값을 설정할 수 있습니다.

참조 SMF 명령에 대한 자세한 내용은 “Oracle Solaris 11.2의 시스템 서비스 관리 ”를 참조하십시오.

♦ ♦ ♦ 3 장

3 저장소에 대한 액세스 제공

이 장에서는 클라이언트가 파일 인터페이스 또는 HTTP 인터페이스를 사용하여 로컬 저장소에 있는 패키지를 검색할 수 있도록 설정하는 방법에 대해 설명합니다. 하나의 저장소에 두 가지 액세스 유형을 모두 구성할 수 있습니다.

사용자가 파일 인터페이스를 사용하여 패키지를 검색하도록 설정

이 절에서는 로컬 네트워크의 디렉토리에서 로컬 저장소 패키지를 제공하는 방법을 설명합니다.

▼ 사용자가 파일 인터페이스를 사용하여 패키지를 검색하도록 설정하는 방법

1. **NFS 공유를 구성합니다.**
   클라이언트가 NFS를 사용하여 로컬 저장소에 액세스하도록 설정하려면 NFS 공유를 만들고 게시합니다.

   ```
   $ zfs share -o share.nfs=on rpool/export/IPSpkgrepos%ipsrepo
   ```

   설정할 수 있는 추가 `share.nfs` 등록 정보 등의 자세한 내용은 `zfs_share`(1M) 매뉴얼 페이지를 참조하십시오.

2. **공유가 게시되었는지 확인합니다.**
   다음 테스트 중 하나를 사용하여 공유가 게시되었는지 확인합니다.

   - **공유 파일 시스템 테이블에서 저장소를 검색합니다.**

     ```
     $ grep repo /etc/dfs/sharetab
     /export/IPSpkgrepos     ipsrepo nfs     sec=sys,rw
     ```

   - **원격 시스템에서 저장소에 액세스할 수 있는지 확인합니다.**

     ```
     $ dfshares solaris
     ```

```
RESOURCE                                  SERVER ACCESS  TRANSPORT
   solaris:/export/IPSpkgrepos           solaris  -       -
```

3. **게시자 원본을 설정합니다.**

   클라이언트 시스템이 로컬 파일 저장소에서 패키지를 가져올 수 있도록 하려면 게시자에 대한 원본을 설정합니다.

   a. **게시자 이름을 결정합니다.**

      다음 명령을 사용하여 저장소의 게시자 이름을 결정합니다.

```
$ pkgrepo info -s /export/IPSpkgrepos/Solaris
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-04-02T18:11:55.640930Z
```

   b. **이 게시자 원본의 적합성을 확인합니다.**

      설치된 패키지를 업데이트하거나, 설치된 패키지에 따라 달라지는 패키지를 설치하거나, 비전역 영역을 설치하려면 게시자 원본으로 설정하는 저장소에 적어도 게시자를 설정할 이미지에 설치된 것과 동일한 소프트웨어가 포함되어 있어야 합니다. 저장소에 이전 또는 최신 소프트웨어가 포함될 수도 있지만 이미지에 설치된 것과 동일한 소프트웨어가 있어야 합니다.

      다음 명령은 지정된 저장소가 이 이미지에 적합하지 않은 게시자 원본임을 보여 줍니다.

```
$ pkg list entire
NAME (PUBLISHER)      VERSION                    IFO
entire                0.5.11-0.175.2.0.0.36.0    i--
$ pkgrepo list -Hs http://pkg.oracle.com/solaris/release
 entire@0.5.11-0.175.2.0.0.36.0
pkgrepo list: The following pattern(s) did not match any packages:
        entire@0.5.11-0.175.2.0.0.36.0
```

      다음 명령은 지정된 저장소가 이 이미지에 적합한 게시자 원본임을 보여 줍니다.

```
$ pkgrepo list -Hs /export/IPSpkgrepos/Solaris entire@0.5.11-0.175.2.0.0.36.0
solaris       entire       0.5.11,5.11-0.175.2.0.0.36.0:20140401T190148Z
```

   c. **게시자 원본을 설정합니다.**

      이전 단계의 저장소 위치 및 게시자 이름으로 다음 명령을 실행하여 게시자 원본을 설정합니다.

```
$ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/Solaris/ solaris
```

      `-G '*'` — `solaris` 게시자에 대한 모든 기존 원본을 제거합니다.

      `-M '*'` — `solaris` 게시자에 대한 모든 기존 미러를 제거합니다.

      `-g` — 새로 만든 로컬 저장소의 URI를 `solaris` 게시자에 대한 새 원본으로 추가합니다.

게시자 구성에 대한 자세한 내용은 “Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트 ”의 “게시자 구성”을 참조하십시오.

다른 이미지의 게시자 원본을 재설정하는 경우 적합성 테스트를 다시 수행합니다. 다른 이미지에는 다른 버전의 소프트웨어가 설치되어 있을 수 있으며 이 저장소를 사용하지 못할 수도 있습니다. 다른 시스템에 있는 이미지의 게시자 원본을 재설정하는 경우 `-g` 인수에 전체 경로를 사용합니다.

사용자가 HTTP 인터페이스를 사용하여 패키지를 검색하도록 설정

이 절에서는 패키지 저장소 서버를 사용하여 로컬 저장소 패키지를 제공하는 방법을 설명합니다.

▼ 사용자가 HTTP 인터페이스를 사용하여 패키지를 검색하도록 설정하는 방법

패키지 저장소 서버 `pkg.depotd`는 패키지 저장소에 포함된 데이터에 대한 네트워크 액세스를 제공합니다. `svc:/application/pkg/server` SMF 서비스는 `pkg.depotd` 데몬을 호출합니다. 클라이언트가 HTTP를 사용하여 로컬 저장소에 액세스하도록 설정하기 위해 이 절차에서는 `pkg/server` 서비스를 구성하는 방법을 보여 줍니다. 서비스의 `default` 인스턴스를 구성할 수 있습니다. 이 절차에서는 새 인스턴스를 만들고 구성하는 방법을 보여 줍니다.

1. **저장소 서버 인스턴스를 만듭니다.**
   `add` 하위 명령을 사용하여 `solaris`라는 `pkg/server` 서비스의 새 인스턴스를 추가합니다.

   ```
   $ svccfg -s pkg/server add solaris
   ```

2. **저장소의 경로를 설정합니다.**
   이 서비스 인스턴스가 저장소 데이터를 찾을 수 있는 경로를 설정합니다.

   ```
   $ svccfg -s pkg/server:solaris setprop pkg/inst_root=/export/IPSpkgrepos/Solaris
   ```

3. **(선택 사항) 포트 번호를 설정합니다.**
   저장소 서버 인스턴스가 수신 패키지 요청을 수신 대기할 포트 번호를 설정합니다. 기본적으로 `pkg.depotd`는 포트 80에서 연결을 수신합니다. 포트를 변경하려면 `pkg/port` 등록 정보를 재설정합니다.

   ```
   $ svccfg -s pkg/server:solaris setprop pkg/port=81
   ```

4. **(선택 사항) 기타 등록 정보를 설정합니다.**
   `pkg/server` 등록 정보의 전체 목록은 `pkg.depotd`(1M) 매뉴얼 페이지를 참조하십시오.

여러 서비스 등록 정보를 설정하려면 다음 명령을 사용하여 모든 등록 정보를 한 번에 편집합니다. 변경한 행의 시작 부분에서 주석 표시자(#)를 제거하는 것에 주의하십시오.

```
$ svccfg -s pkg/server:solaris editprop
```

5. **저장소 서비스를 시작합니다.**

   패키지 저장소 서버 서비스를 다시 시작합니다.

```
$ svcadm refresh pkg/server:solaris
$ svcadm enable pkg/server:solaris
```

6. **저장소 서버가 작동 중인지 테스트합니다.**

   저장소 서버가 작동 중인지 확인하려면 `localhost` 위치에서 브라우저 창을 엽니다. 기본적으로 `pkg.depotd`는 포트 80에서 연결을 수신합니다. 포트를 변경한 경우 `localhost:`*port_number* 위치에서 브라우저 창을 엽니다.

7. **게시자 원본을 설정합니다.**

   클라이언트 시스템이 로컬 파일 저장소에서 패키지를 가져올 수 있도록 하려면 게시자에 대한 원본을 설정합니다.

   a. **게시자 이름을 결정합니다.**

      다음 명령을 사용하여 저장소의 게시자 이름을 결정합니다.

```
$ pkgrepo info -s /export/IPSpkgrepos/Solaris
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-04-02T18:11:55.640930Z
```

   b. **이 게시자 원본의 적합성을 확인합니다.**

      설치된 패키지를 업데이트하거나, 설치된 패키지에 따라 달라지는 패키지를 설치하거나, 비전역 영역을 설치하려면 게시자 원본으로 설정하는 저장소에 적어도 게시자를 설정할 이미지에 설치된 것과 동일한 소프트웨어가 포함되어 있어야 합니다. 저장소에 이전 또는 최신 소프트웨어가 포함될 수도 있지만 이미지에 설치된 것과 동일한 소프트웨어가 있어야 합니다.

      다음 명령은 지정된 저장소가 이 이미지에 적합하지 않은 게시자 원본임을 보여 줍니다.

```
$ pkg list entire
NAME (PUBLISHER)      VERSION                  IFO
entire                0.5.11-0.175.2.0.0.36.0  i--
$ pkgrepo list -Hs http://pkg.oracle.com/solaris/release
 entire@0.5.11-0.175.2.0.0.36.0
pkgrepo list: The following pattern(s) did not match any packages:
        entire@0.5.11-0.175.2.0.0.36.0
```

      다음 명령은 지정된 저장소가 이 이미지에 적합한 게시자 원본임을 보여 줍니다.

```
$ pkgrepo list -Hs http://localhost:81/ entire@0.5.11-0.175.2.0.0.36.0
solaris       entire       0.5.11,5.11-0.175.2.0.0.36.0:20140401T190148Z
```

c. **게시자 원본을 설정합니다.**

게시자 원본을 다음 값 중 하나로 설정합니다.

- `pkg/inst_root` 위치

```
$ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/Solaris/ solaris
```

- `pkg/port` 위치

```
$ pkg set-publisher -G '*' -M '*' -g http://localhost:81/ solaris
```

`-G '*'` `solaris` 게시자에 대한 모든 기존 원본을 제거합니다.

`-M '*'` `solaris` 게시자에 대한 모든 기존 미러를 제거합니다.

`-g` 새로 만든 로컬 저장소의 URI를 `solaris` 게시자에 대한 새 원본으로 추가합니다.

게시자 구성에 대한 자세한 내용은 "Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트 "의 "게시자 구성"을 참조하십시오.

다른 이미지의 게시자 원본을 재설정하는 경우 적합성 테스트를 다시 수행합니다. 다른 이미지에는 다른 버전의 소프트웨어가 설치되어 있을 수 있으며 이 저장소를 사용하지 못할 수도 있습니다.

참조

- "웹 서버 액세스를 사용하여 여러 저장소 제공" [39]에서는 여러 위치나 단일 위치에서 여러 저장소를 제공하는 방법을 설명합니다.
- "하나의 도메인 아래의 다중 저장소" [49]에서는 서로 다른 접두어로 하나의 도메인 이름 아래에서 여러 저장소를 실행하는 방법을 설명합니다.
- "HTTPS 저장소 액세스 구성" [51]에서는 보안 저장소 액세스를 구성하는 방법을 설명합니다.

♦ ♦ ♦ 4 장

4 로컬 IPS 패키지 저장소 유지 관리

이 장에서는 IPS 저장소에서 패키지를 업데이트하고, 저장소의 등록 정보를 설정하거나 업데이트하고, 보조 원본의 저장소에 패키지를 추가하는 방법을 설명합니다.

로컬 저장소 업데이트

이 절의 절차는 IPS 패키지 저장소 업데이트에 대한 다음 모범 사례를 보여 줍니다.

- 해당 릴리스에 대한 모든 지원 업데이트로 각 저장소가 업데이트되도록 유지합니다. 지원 업데이트에는 보안 업데이트 및 기타 중요한 수정 사항이 포함되어 있습니다.
  - 지원 업데이트에서 적용할 특정 수정 사항을 선택하지 마십시오. 지원 업데이트의 패키지 하위 세트를 저장소에 추가하지 마십시오. 지원 업데이트의 모든 컨텐츠를 로컬 저장소에 추가합니다. `pkgrecv` 명령의 기본 동작은 모든 패키지의 버전을 모두 검색하는 것입니다.
  - 지원 업데이트를 건너뛰지 마십시오. 해당하는 모든 지원 업데이트를 각 저장소에 누적합니다.

  사용자는 설치할 `entire` 통합 패키지 버전을 지정하여 저장소의 최신 버전보다 이전 버전으로 업데이트할 수 있습니다. “Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트 ”의 4 장, “Oracle Solaris 이미지 업데이트 또는 업그레이드”를 참조하십시오.
- 저장소 복사본을 업데이트합니다. 이 모범 사례는 저장소가 업데이트되는 동안 시스템이 저장소에 액세스하지 않도록 합니다. 저장소를 업데이트하기 전에 저장소 스냅샷을 만들고, 스냅샷을 복제하고, 업데이트를 수행한 후 원래 저장소를 업데이트된 복제본으로 바꿉니다.

동일한 컨텐츠로 패키지 저장소의 여러 복사본을 유지 관리하는 경우 다음 절차에 따라 동일한 저장소 중 하나를 업데이트합니다. 이 마스터 저장소에서 추가 저장소를 업데이트하는 절차는 “여러 개의 동일한 로컬 저장소 유지 관리” [32]를 참조하십시오.

▼ 로컬 IPS 패키지 저장소를 업데이트하는 방법

참고 - `svc:/application/pkg/mirror` SMF 서비스를 사용하여 정기적으로 저장소를 업데이트하는 경우에는 이 절차를 수행할 필요가 없습니다. `mirror` 서비스 사용에 대한 지침은 인터넷에서 자동으로 저장소를 복사하는 방법 [20]을 참조하십시오.

1. **패키지 저장소의 ZFS 스냅샷을 만듭니다.**

   업데이트할 저장소의 최신 스냅샷이 있는지 확인합니다.

```
$ zfs list -t all -r rpool/export/IPSpkgrepos/Solaris
NAME                                          USED  AVAIL  REFER  MOUNTPOINT
rpool/export/IPSpkgrepos/Solaris             17.6G  78.4G    34K  /export/IPSpkgrepos/Solaris
rpool/export/IPSpkgrepos/Solaris@initial         0      -  17.6G  -
```

   저장소 스냅샷이 이미 있는 경우 `zfs diff` 명령을 사용하여 스냅샷이 저장소 데이터 세트와 같은지 확인합니다.

```
$ zfs diff rpool/export/IPSpkgrepos/Solaris@initial
$
```

   `zfs diff` 명령이 출력을 생성하지 않는 경우 스냅샷이 상위 데이터 세트와 같으며 해당 스냅샷을 업데이트에 사용할 수 있습니다.

   `zfs diff` 명령이 출력을 생성하는 경우 또는 저장소의 스냅샷이 없는 경우 인터넷에서 명시적으로 저장소를 복사하는 방법 [18]의 6단계과 같이 새 스냅샷을 작성합니다. 이 새로운 스냅샷을 업데이트에 사용합니다.

2. **패키지 저장소의 ZFS 복제본을 만듭니다.**

   저장소 스냅샷을 복제하여 업데이트할 수 있는 저장소 복사본을 만듭니다.

```
$ zfs clone rpool/export/IPSpkgrepos/Solaris@initial rpool/export/IPSpkgrepos/Solaris_tmp
$ zfs list -r rpool/export/IPSpkgrepos/Solaris/
NAME                                          USED  AVAIL  REFER  MOUNTPOINT
rpool/export/IPSpkgrepos/Solaris             17.6G  78.4G    34K  /export/IPSpkgrepos/Solaris
rpool/export/IPSpkgrepos/Solaris@initial         0      -  17.6G  -
rpool/export/IPSpkgrepos/Solaris_tmp          76K  78.4G  17.6G  /export/IPSpkgrepos/
Solaris_tmp
```

3. **패키지 저장소의 ZFS 복제본을 업데이트합니다.**

   파일 또는 HTTP 위치에서 원래 저장소를 만든 것과 동일한 방식으로 파일 또는 HTTP 위치에서 저장소를 업데이트할 수 있습니다.

   - **`zip` 파일에서 업데이트합니다.**

     예 2-2. “`zip` 파일에서 기존 저장소에 추가”를 참조하십시오. 지정된 대상에 패키지 저장소가 이미 포함되어 있는 경우 `zip` 파일의 컨텐츠가 기존 저장소의 컨텐츠에 추가됩니다.

- **ISO 파일에서 업데이트합니다.**

  a. **ISO 이미지를 마운트합니다.**

  ```
  $ mount -F hsfs ./sol-11_2-incr-repo.iso /mnt
  ```

  b. **ISO 파일 컨텐츠를 저장소 복제본에 복사합니다.**

  iso 파일에서 저장소를 복사하는 방법 [17]에 표시된 대로 rsync 또는 tar을 사용합니다.

  ```
  $ rsync -aP /mnt/repo/ /export/IPSpkgrepos/Solaris_tmp
  ```

  c. **ISO 이미지를 마운트 해제합니다.**

- **저장소에서 업데이트합니다.**

  다른 저장소의 컨텐츠를 저장소 복제본에 복사합니다. 보안 사이트에서 복사하는 경우 필요한 SSL 인증서와 키가 설치되어 있는지 확인하고 필요한 인증서 및 키 옵션을 지정합니다.

  ```
  $ pkgrecv -s https://pkg.oracle.com/solaris/support \
  -d /export/IPSpkgrepos/Solaris_tmp \
  --key /path-to-ssl_key --cert /path-to-ssl_cert '*'
  ```

  pkgrecv 명령에 대한 자세한 내용은 pkgrecv(1) 매뉴얼 페이지를 참조하십시오. 변경된 패키지만 업데이트되므로 저장소를 업데이트하는 시간은 원래 저장소를 채우는 시간보다 훨씬 짧을 수 있습니다. "저장소 복사에 대한 성능 고려 사항" [13]의 성능 팁을 참조하십시오.

  pkgrecv 작업이 중단된 경우 "중단된 패키지 수신 재개" [32]의 지침을 따르십시오.

4. **작업 저장소를 업데이트된 복제본으로 바꿉니다.**

   ```
   $ svcadm disable -st pkg/server:solaris
   $ zfs promote rpool/export/IPSpkgrepos/Solaris_tmp
   $ zfs rename rpool/export/IPSpkgrepos/Solaris rpool/export/IPSpkgrepos/Solaris_old
   $ zfs rename rpool/export/IPSpkgrepos/Solaris_tmp rpool/export/IPSpkgrepos/Solaris
   ```

   svcadm 명령에 대한 자세한 내용은 svcadm(1M) 매뉴얼 페이지를 참조하십시오.

5. **업데이트된 저장소를 확인합니다.**

   pkgrepo verify 명령을 사용하여 업데이트된 저장소를 확인합니다. pkgrepo verify 및 pkgrepo fix 명령에 대한 자세한 내용은 pkgrepo(1) 매뉴얼 페이지를 참조하십시오.

6. **새 패키지를 카탈로그화하고 검색 색인을 업데이트합니다.**

   새로 업데이트된 저장소에 있는 새 패키지를 카탈로그화하고 모든 검색 색인을 업데이트합니다.

```
$ pkgrepo refresh -s rpool/export/IPSpkgrepos/Solaris
```

7. **ZFS 복제본을 패키지 저장소의 새로 업데이트된 복제본으로 만듭니다.**

```
$ zfs snapshot rpool/export/IPSpkgrepos/Solaris@S11U2SRU1
```

8. **SMF 서비스를 다시 시작합니다.**

   HTTP 인터페이스를 통해 저장소를 제공하는 경우 SMF 서비스를 다시 시작합니다. 서비스를 다시 시작할 때 적합한 서비스 인스턴스를 지정해야 합니다.

```
$ svcadm restart pkg/server:solaris
```

9. **이전 저장소를 제거합니다.**

   업데이트된 저장소가 제대로 작동하는 경우 이전 저장소를 제거할 수 있습니다.

```
$ zfs destroy rpool/export/IPSpkgrepos/Solaris_old
```

중단된 패키지 수신 재개

pkgrecv 작업이 중단된 경우 -c 옵션을 사용하여 이미 다운로드한 컨텐츠를 검색하고 컨텐츠 다운로드를 재개합니다. *cache_dir* 값은 다음 예에 표시된 것처럼 전송이 중단되었을 때 정보 메시지에 제공됩니다.

```
PROCESS                    ITEMS       GET (MB)        SEND (MB)
...
pkgrecv: http protocol error: code: 503 reason: Service Unavailable
URL: 'https://pkg.oracle.com/solaris/support/file/file_hash

pkgrecv: Cached files were preserved in the following directory:
        /var/tmp/pkgrecv-fOGaIg
Use pkgrecv -c to resume the interrupted download.
$ pkgrecv -c /var/tmp/pkgrecv-fOGaIg \
-s https://pkg.oracle.com/solaris/support -d /export/IPSpkgrepos/Solaris_tmp \
--key /path/to/ssl_key --cert /path/to/ssl_cert '*'
Processing packages for publisher solaris ...
Retrieving and evaluating 156 package(s)...
```

여러 개의 동일한 로컬 저장소 유지 관리

다음 목적을 충족하려면 동일한 컨텐츠로 패키지 저장소의 여러 복사본을 유지 관리하는 것이 좋습니다.

- 각기 다른 노드에서 복사본을 유지 관리하여 저장소의 가용성을 늘립니다.

- 많은 사용자가 있거나 사용자가 먼 거리에 분산되어 있는 경우 저장소 액세스 성능을 향상시킵니다.

로컬 IPS 패키지 저장소를 업데이트하는 방법 [30] 절차에 따라 패키지 저장소 중 하나를 업데이트합니다. 그런 다음 로컬 IPS 패키지 저장소를 복제하는 방법 [33] 절차에 따라 처음 업데이트한 저장소에서 동일한 저장소를 추가로 업데이트합니다. 이러한 두 절차는 비슷하지만 `pkgrecv` 명령을 사용하는 방식에 중요한 차이점이 있습니다. 복제 절차의 `pkgrecv` 작업은 원본 저장소 파일을 정확하게 복사하며 결과는 다음과 같습니다.

- 복제된 저장소 카탈로그의 시간 기록은 원본 저장소 카탈로그의 시간 기록과 정확히 같습니다. 저장소가 로드 균형 조정된 경우 로드 밸런서가 노드 간에 클라이언트를 전환할 때 문제를 방지하려면 모든 저장소의 카탈로그가 정확히 같아야 합니다. 로드 균형 조정에 대한 자세한 내용은 "로드 균형 조정 구성" [50]을 참조하십시오.
- 대상 저장소에는 있지만 원본 저장소에 없는 패키지는 대상 저장소에서 제거됩니다. 스파스 저장소만 포함된 복사본을 만들려는 경우가 아니면 복제 작업의 소스로 스파스 저장소를 사용하지 마십시오.

▼ 로컬 IPS 패키지 저장소를 복제하는 방법

이러한 단계에 대한 자세한 내용은 로컬 IPS 패키지 저장소를 업데이트하는 방법 [30]을 참조하십시오.

1. **대상 저장소를 복사합니다.**
   대상 저장소의 최신 스냅샷이 있는지 확인합니다. 이 스냅샷의 ZFS 복제본을 만듭니다.

2. **대상 저장소의 복사본을 업데이트합니다.**
   `pkgrecv` 명령을 사용하여 이전에 업데이트된 로컬 패키지 저장소를 대상 저장소 복사본에 복제합니다. `pkgrecv` 복제 작업에 대한 자세한 내용은 `pkgrecv`(1) 매뉴얼 페이지를 참조하십시오.

   ```
   $ pkgrecv -s /net/host1/export/IPSpkgrepos/Solaris \
   -d /net/host2/export/IPSpkgrepos/Solaris_tmp --clone
   ```

3. **작업 대상 저장소를 업데이트된 복제본으로 바꿉니다.**

4. **업데이트된 저장소를 확인합니다.**
   `pkgrepo verify` 명령을 사용하여 업데이트된 대상 저장소를 확인합니다.

5. **새로 업데이트된 저장소의 스냅샷을 작성합니다.**

6. **SMF 서비스를 다시 시작합니다.**
   HTTP 인터페이스를 통해 저장소를 제공하는 경우 SMF 서비스를 다시 시작합니다. 서비스를 다시 시작할 때 적합한 서비스 인스턴스를 지정해야 합니다.

7. **이전 저장소를 제거합니다.**
   업데이트된 저장소가 제대로 작동하는 경우 이전 저장소를 제거합니다.

참조 HTTP 인터페이스를 통해 저장소를 제공하는 경우 다음과 같은 관련 설명서를 참조하십시오.

- "웹 서버 액세스를 사용하여 여러 저장소 제공" [39]에서는 서로 다른 포트에서 실행 중인 여러 `pkg.depotd` 데몬을 사용하여 여러 저장소를 제공하는 방법을 설명합니다.
- "하나의 도메인 아래의 다중 저장소" [49]에서는 서로 다른 접두어로 하나의 도메인 이름 아래에서 여러 저장소를 실행하는 방법을 설명합니다.

저장소 등록 정보 확인 및 설정

이 절에서는 IPS 저장소에 대한 정보를 표시하는 방법 및 저장소 등록 정보 값을 변경하는 방법을 설명합니다.

전체 저장소에 적용되는 등록 정보 보기

다음 명령은 로컬 저장소에 의해 알려진 패키지 게시자의 목록을 표시합니다. STATUS(상태) 열은 게시자의 패키지 데이터를 현재 처리 중인지 여부를 나타냅니다.

```
$ pkgrepo info -s /export/IPSpkgrepos/Solaris
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4506     online           2013-07-11T23:32:46.379726Z
```

다음 명령은 전체 저장소에 적용되는 등록 정보를 표시합니다. 저장소 등록 정보의 전체 목록 및 등록 정보 값 지정을 비롯한 설명은 pkgrepo(1) 매뉴얼 페이지를 참조하십시오.

```
$ pkgrepo get -s /export/IPSpkgrepos/Solaris
SECTION    PROPERTY                    VALUE
publisher  prefix                      solaris
repository check-certificate-revocation False
repository signature-required-names     ()
repository trust-anchor-directory       /etc/certs/CA/
repository version                     4
```

`publisher/prefix`

기본 게시자의 이름입니다. 저장소에는 여러 게시자의 패키지가 포함될 수 있지만 하나의 게시자만 기본 게시자로 설정할 수 있습니다. 이 기본 게시자 이름은 다음과 같은 목적으로 사용됩니다.

- `pkg` 명령에서 패키지 FMRI에 게시자가 지정되지 않은 경우 패키지를 식별하려면
- `pkgsend`(1) 명령을 사용하여 패키지가 저장소에 게시되고 패키지 매니페스트에 게시자가 지정되지 않은 경우 패키지에 게시자를 지정하려면

repository/check-certificate-revocation

인증서를 확인하기 위한 플래그입니다. True로 설정된 경우 pkgrepo verify 명령은 인증서 발급 후 인증서가 해지되었는지 확인하려고 합니다. 이 값은 "Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트 "의 "추가 이미지 등록 정보" 및 pkg(1) 매뉴얼 페이지에 설명된 check-certificate-revocation 이미지 등록 정보 값과 일치해야 합니다.

repository/signature-required-names

패키지의 서명을 검증하는 동안 인증서의 공통 이름으로 표시되어야 하는 이름 목록입니다. 이 목록은 pkgrepo verify 명령에서 사용됩니다. 이 값은 "Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트 "의 "서명된 패키지에 필요한 이미지 등록 정보" 및 pkg(1) 매뉴얼 페이지에 설명된 signature-required-names 이미지 등록 정보 값과 일치해야 합니다.

repository/trust-anchor-directory

이 저장소에서 패키지의 신뢰 앵커가 포함된 디렉토리의 절대 경로 이름입니다. 기본값은 /etc/certs/CA/입니다. 이 값은 "Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트 "의 "추가 이미지 등록 정보" 및 pkg(1) 매뉴얼 페이지에 설명된 trust-anchor-directory 이미지 등록 정보 값과 일치해야 합니다.

repository/version

저장소의 형식 버전입니다. 이 값은 "저장소 등록 정보 값 수정" [37]의 pkgrepo set 명령을 사용하여 설정할 수 없습니다. 이 값은 pkgrepo create 명령을 사용하여 설정할 수 있습니다. 기본적으로 버전 4 저장소가 만들어집니다. 버전 4 저장소는 다중 게시자에 대한 패키지 저장을 지원합니다.

저장소 게시자 등록 정보 보기

다음 명령은 로컬 저장소의 solaris 게시자에 대한 등록 정보를 표시합니다. 괄호는 값이 값 목록일 수 있음을 나타냅니다.

```
$ pkgrepo get -p solaris -s /export/IPSpkgrepos/Solaris
PUBLISHER SECTION    PROPERTY         VALUE
solaris   publisher  alias
solaris   publisher  prefix           solaris
solaris   repository collection-type  core
solaris   repository description      ""
solaris   repository legal-uris       ()
solaris   repository mirrors          ()
solaris   repository name             ""
solaris   repository origins          ()
solaris   repository refresh-seconds  ""
solaris   repository registration-uri ""
solaris   repository related-uris     ()
```

`publisher/prefix`

`-p` 옵션에 지정된 게시자 이름입니다. `-p` 옵션이 지정되지 않은 경우 이 값은 이전 절의 설명처럼 이 저장소에 대한 기본 게시자의 이름입니다.

`repository/collection-type`

이 저장소의 패키지 유형입니다. 값이 `core`이면 이 저장소에 저장소의 패키지가 선언한 모든 종속성이 포함됩니다. 값이 `supplemental`이면 이 저장소에 저장소의 패키지가 선언한 모든 종속성이 포함되지 않습니다.

`repository/description`

이 저장소의 목적과 컨텐츠입니다. HTTP 인터페이스에서 이 저장소를 사용할 수 있는 경우 이 값은 기본 페이지 위쪽의 About(정보) 섹션에 표시됩니다.

`repository/legal-uris`

저장소에 대한 법적 정보를 제공하는 문서의 위치 목록입니다.

`repository/mirrors`

이 저장소와 동일한 패키지 컨텐츠가 포함된 저장소의 위치 목록입니다.

`repository/name`

이 저장소의 이름입니다. HTTP 인터페이스에서 이 저장소를 사용할 수 있는 경우 이 값은 기본 페이지 맨 위와 창 제목에 표시됩니다.

`repository/origins`

이 저장소와 동일한 패키지 컨텐츠 및 메타 데이터가 포함된 저장소의 위치 목록입니다.

`repository/refresh-seconds`

클라이언트가 이 저장소에서 업데이트된 패키지 데이터 확인 간에 기다릴 시간(초)입니다.

`repository/registration-uri`

이 저장소에 액세스하는 데 필요한 자격 증명을 얻는 데 사용해야 하는 리소스의 위치입니다.

`repository/related-uris`

관심을 가질 수 있는 다른 패키지가 포함된 저장소의 위치 목록입니다.

다음 명령은 `pkg.oracle.com` 저장소의 지정된 *section/property*에 대한 정보를 표시합니다.

```
$ pkgrepo get -p solaris -s http://pkg.oracle.com/solaris/release \
repository/name repository/description
PUBLISHER SECTION    PROPERTY    VALUE
solaris   repository description This\ repository\ serves\ the\ Oracle\ Solaris\ 11\ Package\
 repository.
solaris   repository name        Oracle\ Solaris\ 11\ Package\ Repository
```

저장소 등록 정보 값 수정

"저장소 게시자 등록 정보 보기" [35]는 로컬 저장소의 solaris 게시자에 대해 저장소 이름 및 설명 등록 정보 값이 설정되지 않았음을 보여 줍니다. HTTP 인터페이스에서 이 저장소를 사용할 수 있고 브라우저에서 이 저장소의 컨텐츠를 보는 경우 기본 이름만 표시되고 설명은 표시되지 않습니다. 이러한 값을 설정하면 게시자 repository/name 값이 페이지 위쪽 및 페이지 제목으로 표시되고 게시자 repository/description 값이 이름 바로 아래의 About(정보) 섹션에 표시됩니다. 이러한 값을 설정할 때 -p 옵션을 사용하여 게시자를 하나 이상 지정해야 합니다. 이 저장소에 두 개 이상 게시자의 컨텐츠가 포함된 경우 각 게시자에 대해 다른 값을 설정하거나 -p all을 지정할 수 있습니다.

```
$ pkgrepo set -p solaris -s /export/IPSpkgrepos/Solaris \
repository/description="Local copy of the Oracle Solaris 11 repository." \
repository/name="Oracle Solaris 11"
$ pkgrepo get -p solaris -s /export/IPSpkgrepos/Solaris repository/name repository/
description
PUBLISHER SECTION    PROPERTY         VALUE
solaris   repository description      Local\ copy\ of\ the\ Oracle\ Solaris\ 11\ repository.
solaris   repository name             Oracle\ Solaris\ 11
```

로컬 저장소 사용자 정의

pkgrecv 명령을 사용하여 저장소에 패키지 및 해당 게시자 데이터를 추가할 수 있습니다. pkgrepo 명령을 사용하여 저장소에서 패키지 및 게시자를 제거할 수 있습니다.

저장소에 패키지 추가

저장소에 게시자를 추가할 수 있습니다. 예를 들어, 하나의 저장소에서 solaris, ha-cluster 및 solarisstudio 패키지를 유지 관리할 수 있습니다.

사용자 정의 패키지를 추가하는 경우 사용자 정의 게시자 이름 아래에 해당 패키지를 게시합니다. 사용자 정의 패키지를 solaris 등의 기존 게시자로 게시하지 마십시오. 게시자가 지정되지 않은 패키지를 게시하는 경우 해당 패키지는 저장소에 대한 기본 게시자로 추가됩니다. 올바른 기본 게시자를 사용하여 테스트 저장소에 사용자 정의 패키지를 게시합니다. 그런 다음 pkgrecv 명령을 사용하여 생산 저장소에 패키지 및 해당 게시자 정보를 추가합니다. 지침은 "Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.2 "의 "Publish the Package"를 참조하십시오.

다음 예에서는 isvpub 게시자 데이터 및 ISVproducts.p5p 패키지 아카이브의 모든 패키지가 로컬 저장소에 추가됩니다. 패키지 아카이브는 게시자 정보와 해당 게시자가 제공한 하나 이상의 패키지가 포함된 파일입니다. "Packaging and Delivering Software With the

Image Packaging System in Oracle Solaris 11.2 ”의 “Deliver as a Package Archive File”을 참조하십시오. 대부분의 pkgrepo 작업은 패키지 아카이브에 사용할 수 없습니다. 패키지 아카이브에는 패키지만 포함되고 저장소 구성은 포함되어 있지 않습니다. 그러나 pkgrepo list 및 pkgrepo contents 명령은 패키지 아카이브에서 작동합니다. pkgrepo contents 명령은 “저장소의 패키지 검사” [39]에서 설명합니다.

pkgrepo list 출력에서는 게시자가 이 이미지에서 검색 순서가 가장 높은 게시자가 아니기 때문에 표시됩니다.

```
$ pkgrepo -s /tmp/ISVproducts.p5p list
PUBLISHER NAME                                    O VERSION
isvpub    isvtool                                   1.1,5.11:20131120T021902Z
isvpub    isvtool                                   1.0,5.11:20131120T010105Z
```

다음 pkgrecv 명령은 원본 저장소에서 모든 패키지를 검색합니다. 검색할 패키지 이름을 나열하거나 '*' 이외의 패턴을 지정하는 경우 -r 옵션을 지정하여 필요한 종속성 패키지를 모두 검색해야 합니다.

```
$ pkgrecv -s /tmp/ISVproducts.p5p -d /export/IPSpkgrepos/Solaris '*'
Processing packages for publisher isvpub ...
Retrieving and evaluating 2 package(s)...
PROCESS        ITEMS      GET (MB)      SEND (MB)
Completed        2/2       0.0/0.0          0.0/0
```

저장소 컨텐츠를 변경한 후 저장소를 새로 고치고 이 저장소에 대해 구성된 패키지 저장소 서버 서비스 인스턴스를 다시 시작합니다.

```
$ pkgrepo -s /export/IPSpkgrepos/Solaris refresh -p isvpub
Initiating repository refresh.
$ svcadm refresh pkg/server:solaris
$ svcadm restart pkg/server:solaris
```

다음 pkgrepo info 명령은 검색된 두 패키지가 동일한 패키지의 서로 다른 버전이므로 한 패키지를 표시합니다. pkgrepo list 명령은 두 패키지를 모두 표시합니다.

```
$ pkgrepo -s /export/IPSpkgrepos/Solaris info
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-01-02T19:19:06.983979Z
isvpub    1        online           2014-03-20T23:24:37.196773Z
$ pkgrepo -s /export/IPSpkgrepos/Solaris list -p isvpub
PUBLISHER NAME                                    O VERSION
isvpub    isvtool                                   1.1,5.11:20131120T021902Z
isvpub    isvtool                                   1.0,5.11:20131120T010105Z
```

pkg set-publisher 명령을 사용하여 isvpub 게시자에 대한 새 저장소 위치를 추가합니다.

HTTP 인터페이스에서 이 저장소를 사용할 수 있고 브라우저에서 이 저장소의 컨텐츠를 보는 경우 위치에 게시자를 지정하여 이 새로운 패키지를 볼 수 있습니다. 예를 들어, http://localhost:81/isvpub/를 지정할 수 있습니다.

저장소의 패키지 검사

"저장소에 패키지 추가" [37]에 표시된 `pkgrepo info` 및 `pkgrepo list` 명령 외에도 `pkgrepo contents` 명령을 사용하여 저장소의 패키지 컨텐츠를 검사할 수 있습니다.

단일 패키지의 경우 `pkgrepo contents` 명령의 출력은 `pkg contents -m` 명령의 출력과 같습니다. `pkgrepo contents` 명령은 지정된 저장소의 각 일치 패키지에 대한 출력을 표시하고 `pkg contents` 명령은 이 이미지에서 설치할 수 있는 일치 패키지 버전에 대한 출력만 표시합니다. `-t` 옵션을 지정하면 `pkgrepo contents` 명령은 지정된 작업만 표시합니다.

다음 예에서는 이 패키지의 한 버전만 지정된 저장소에 있으므로 패키지 버전을 지정할 필요가 없습니다. 이 패키지에는 Oracle Database 12 설치와 작업에 필요한 Oracle Solaris 패키지 세트를 제공하는 `depend` 작업이 포함되어 있습니다.

```
$ pkgrepo -s http://pkg.oracle.com/solaris/release/ \
contents -t depend oracle-rdbms-server-12cR1-preinstall
depend fmri=x11/library/libxi type=group
depend fmri=x11/library/libxtst type=group
depend fmri=x11/session/xauth type=group
depend fmri=compress/unzip type=require
depend fmri=developer/assembler type=require
depend fmri=developer/build/make type=require
```

저장소에서 패키지 제거

Oracle 게시자가 제공한 패키지를 제거하지 마십시오. "Oracle Solaris 11.2의 소프트웨어 추가 및 업데이트 "에서는 원하는 패키지만 설치하고 원하지 않는 패키지를 설치하지 않는 방법을 보여 줍니다.

`pkgrepo remove` 명령을 사용하여 Oracle 게시자가 제공하지 않은 패키지를 제거할 수 있습니다. `pkgrepo remove-publisher` 명령을 사용하여 게시자 및 해당 게시자가 제공한 모든 패키지를 제거할 수 있습니다. 자세한 내용은 **pkgrepo**(1) 매뉴얼 페이지를 참조하십시오. 로컬 IPS 패키지 저장소를 업데이트하는 방법 [30]의 설명에 따라 이러한 작업은 저장소 복사본에서 수행해야 합니다.

웹 서버 액세스를 사용하여 여러 저장소 제공

이 절의 절차에서는 "사용자가 HTTP 인터페이스를 사용하여 패키지를 검색하도록 설정" [25]에 제공된 정보를 확장하여 여러 저장소를 제공하도록 지원하는 방법을 보여 줍니다.

다음 방법은 HTTP 액세스를 사용하여 여러 개의 IPS 패키지 저장소를 제공하는 두 가지 방법입니다. 두 방법 모두 고유한 저장소 경로를 사용하여 pkg/server 서비스의 추가 인스턴스를 만드는 것으로 시작합니다.

- 여러 위치. 사용자가 개별 위치에서 페이지를 표시하여 각 저장소에 액세스합니다.
- 단일 위치. 사용자가 한 위치에서 모든 저장소에 액세스합니다.

여러 위치에 대한 액세스 제공 외에도 "저장소에 패키지 추가" [37]와 같이 단일 저장소에서 여러 게시자의 패키지를 제공할 수 있습니다.

▼ 개별 위치에서 여러 저장소를 제공하는 방법

이 예에서는 Solaris 저장소 외에도 SolarisStudio 저장소가 존재합니다. Solaris 저장소는 pkg/server 서비스의 solaris 인스턴스에 지정된 대로 포트 81을 사용하여 http://localhost/에서 액세스할 수 있습니다. "사용자가 HTTP 인터페이스를 사용하여 패키지를 검색하도록 설정" [25]을 참조하십시오.

1. **새 저장소 서버 인스턴스를 만듭니다.**

   svccfg 명령의 add 하위 명령을 사용하여 pkg/server 서비스의 새 인스턴스를 추가합니다.

   ```
   $ svccfg -s pkg/server add studio
   ```

2. **새 인스턴스가 추가되었는지 확인합니다.**

   ```
   $ svcs pkg/server
   STATE  STIME    FMRI
   online 14:54:16 svc:/application/pkg/server:default
   online 14:54:20 svc:/application/pkg/server:studio
   online 14:54:20 svc:/application/pkg/server:solaris
   ```

3. **저장소의 경로를 설정합니다.**

   이 서비스 인스턴스가 저장소 데이터를 찾을 수 있는 경로를 설정합니다.

   ```
   $ svccfg -s pkg/server:studio setprop pkg/inst_root=/export/IPSpkgrepos/SolarisStudio
   ```

4. **(선택 사항) 새 인스턴스에 대한 포트 번호를 설정합니다.**

   ```
   $ svccfg -s pkg/server:studio setprop pkg/port=82
   ```

5. **(선택 사항) Apache 프록시 기준을 설정합니다.**

   pkg/proxy_base 설정 예는 "단순 접두어가 지정된 프록시 구성" [48]을 참조하십시오.

6. **저장소 이름과 설명을 설정합니다.**

   저장소 이름과 설명이 "저장소 등록 정보 값 수정" [37]에 표시된 대로 설정되었는지 확인합니다.

7. **저장소 서비스를 시작합니다.**
   패키지 저장소 서버 서비스를 다시 시작합니다.

   ```
   $ svcadm refresh pkg/server:studio
   $ svcadm enable pkg/server:studio
   ```

8. **저장소 서버가 작동 중인지 테스트합니다.**
   `http://localhost:82/` 위치에서 브라우저 창을 엽니다.
   포트 번호를 설정하지 않은 경우 기본값은 80입니다. `http://localhost:80/` 또는 `http://localhost/`에서 저장소를 봅니다.
   포트 번호가 다른 `pkg/server` 인스턴스에서도 사용 중인 경우 새 패키지를 보려면 위치에 게시자 이름을 추가합니다. 예를 들어, `http://localhost:81/solarisstudio/`에서 저장소를 봅니다.

9. **게시자 원본을 설정합니다.**
   게시자 원본을 다음 값 중 하나로 설정합니다.

   - `pkg/inst_root` 위치

     ```
     $ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/SolarisStudio/ \
     solarisstudio
     ```

   - `pkg/port` 위치

     ```
     $ pkg set-publisher -G '*' -M '*' -g http://localhost:82/ solarisstudio
     ```

참조 `http://pkg.example.com/solaris` 및 `http://pkg.example.com/studio`와 같이 서로 다른 접두어로 하나의 도메인 이름 아래에 여러 저장소를 실행하는 방법은 "하나의 도메인 아래의 다중 저장소" [49]를 참조하십시오.

▼ 단일 위치에서 여러 저장소를 제공하는 방법

이 절차의 많은 단계는 이전 절차의 단계와 같습니다. 자세한 내용은 이전 절차를 참조하십시오.

1. **새 저장소 서버 인스턴스를 만듭니다.**

2. **저장소의 경로를 설정합니다.**
   특정 `pkg/depot` 인스턴스에서 관리하는 각 `pkg/server` 인스턴스에는 고유한 `pkg/inst_root` 값이 있어야 합니다.

3. **새 인스턴스에 대한 `readonly` 등록 정보를 확인합니다.**
   `pkg/readonly` 등록 정보의 기본값은 `true`입니다. 이 값이 변경된 경우 값을 `true`로 재설정합니다.

```
$ svcprop -p pkg/readonly pkg/server:studio
true
```

4. **새 인스턴스에 대한 `standalone` 등록 정보를 설정합니다.**
   기본적으로 `pkg/standalone` 등록 정보 값은 `true`입니다. `pkg/standalone` 등록 정보가 `false`로 설정된 모든 `pkg/server` 인스턴스는 `pkg/depot` 서비스 인스턴스에 의해 동일한 위치에서 제공될 수 있습니다.

   ```
   $ svccfg -s pkg/server:studio
   svc:/application/pkg/server:studio> setprop pkg/standalone=false
   svc:/application/pkg/server:studio> refresh
   svc:/application/pkg/server:studio> select solaris
   svc:/application/pkg/server:solaris> setprop pkg/standalone=false
   svc:/application/pkg/server:solaris> refresh
   svc:/application/pkg/server:solaris> exit
   $
   ```

   `pkg/standalone` 등록 정보가 `false`로 설정된 각 `pkg/server` 인스턴스에 대해 `pkg/inst_root` 등록 정보 값이 고유한지 확인합니다.

5. **(선택 사항) `pkg/depot` 인스턴스에 대한 포트 번호를 설정합니다.**
   기본적으로 `svc:/application/pkg/depot:default` 서비스의 포트 번호는 80입니다. 이 포트 번호는 이 `pkg/depot` 인스턴스에서 관리될 모든 `pkg/server` 인스턴스에 대한 포트 번호와 같을 수 있습니다. 포트 번호를 변경하려면 `pkg/depot:default`의 `config/port` 등록 정보를 설정합니다.

6. **`pkg/depot` 인스턴스를 다시 시작합니다.**

   ```
   $ svcadm refresh pkg/depot:default
   $ svcadm restart pkg/depot:default
   ```

7. **저장소 서버가 작동 중인지 테스트합니다.**
   사용자가 `http://localhost:80/` 위치를 열면 `http://localhost/solaris` 저장소가 `solaris` 게시자와 함께 나열되고 `http://localhost/studio` 저장소가 `solarisstudio` 게시자와 함께 나열됩니다.
   한 저장소가 여러 게시자에 대한 패키지를 제공하는 경우 모든 게시자가 나열됩니다. 예를 들어, `http://localhost/solaris` 저장소가 `solaris` 및 `isvpub` 게시자와 함께 나열될 수도 있습니다.

8. **게시자 원본을 설정합니다.**
   게시자 원본을 다음 값 중 하나로 설정합니다.

   - 고유한 `pkg/inst_root` 위치.

     ```
     $ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/SolarisStudio/ \
     solarisstudio
     ```

   - `config/port` 값과 `pkg/server` 인스턴스 이름으로 정의된 위치입니다.

```
$ pkg set-publisher -G '*' -M '*' -g http://localhost:80/studio/ solarisstudio
```

다음 순서 pkg/depot 인스턴스에서 관리되는 저장소 컨텐츠를 변경하는 경우 "로컬 저장소 업데이트" [29] 및 "로컬 저장소 사용자 정의" [37]의 설명에 따라 다음 단계를 둘 다 수행하십시오.

- 저장소에서 pkgrepo refresh를 실행합니다.
- pkg/depot 인스턴스에서 svcadm restart를 실행합니다.

각 인스턴스가 하나 이상의 저장소를 호스트하는 pkg/depot 서비스의 추가 인스턴스를 만들 수 있습니다.

pkg/server 및 pkg/depot 서버 인스턴스를 구성하는 대신 독립형 구성을 생성하려면 pkg.depot-config(1M) 매뉴얼 페이지를 참조하십시오.

♦ ♦ ♦ 5 장

5 웹 서버 뒤에서 저장소 서버 실행

Apache 웹 서버 인스턴스 뒤에서 저장소 서버를 실행하면 다음과 같은 이점이 있습니다.

- 하나의 도메인 이름으로 여러 저장소를 호스트할 수 있습니다. pkg(5) 저장소 서버는 로컬 네트워크 또는 인터넷에서 저장소에 대한 액세스를 쉽게 제공할 수 있게 해줍니다. 하지만 저장소 서버는 하나의 도메인 이름 또는 정교한 접두어 아래에서 여러 저장소를 제공하도록 지원하지 않습니다. 하나의 도메인 이름 아래에서 여러 저장소를 호스트하려면 웹 프록시 뒤에서 저장소 서버를 실행하십시오.
- 성능 및 가용성이 향상됩니다. 웹 프록시 뒤에서 저장소 서버를 실행하면 여러 저장소 간에 로드 균형 조정을 사용으로 설정하고 컨텐츠 캐싱을 사용으로 설정하여 서버의 성능 및 가용성을 향상시킬 수 있습니다.
- 보안 저장소 서버를 제공할 수 있습니다. 클라이언트 인증서를 지원하는 SSL(Secure Sockets Layer) 프로토콜 사용 Apache 인스턴스 뒤에서 저장소 서버를 실행합니다.

저장소 서버 Apache 구성

이 장의 예에서는 Apache 웹 서버를 프록시 소프트웨어로 사용합니다. `svc:/network/http:apache22` 서비스를 사용으로 설정하여 Apache 웹 서버를 활성화합니다. 자세한 내용은 Apache HTTP Server Version 2.2 Documentation을 참조하십시오.

이 예에 표시된 원칙을 모든 프록시 서버 소프트웨어에 적용할 수 있어야 합니다.

Oracle Solaris 11.2 OS에는 `/etc/apache2/2.2`의 기본 `httpd.conf` 파일을 제공하는 `web/server/apache-22` 패키지의 Apache 웹 서버가 포함되어 있습니다. 일반적으로 다음 명령을 사용하여 `httpd.conf` 파일을 찾을 수 있습니다.

```
$ pkg search -Hl -o path ':file:path:*httpd.conf'
etc/apache2/2.2/httpd.conf
etc/apache2/2.2/original/httpd.conf
```

필요한 Apache 구성 설정

Apache 웹 서버 인스턴스 뒤에서 패키지 저장소를 실행하는 경우 인코딩된 슬래시를 디코딩하지 않도록 `httpd.conf` 파일에 다음 설정을 포함합니다.

```
AllowEncodedSlashes NoDecode
```

슬래시는 계층적 패키지 이름을 표현하는 데 사용되므로 패키지 이름에 URL 인코딩된 슬래시가 포함될 수 있습니다. 예를 들어, 패키지 이름 `pkg://solaris/developer/build/make`는 웹 서버에 대한 `http://pkg.oracle.com/solaris/release/manifest/0/developer%2Fbuild%2Fmake`가 됩니다. 이러한 슬래시가 디렉토리 구분 기호로 해석되지 않도록 하려면 Apache에 `%2F` 인코딩된 슬래시를 디코딩하지 않도록 지시합니다.

이 설정을 생략하면 `404 Not Found` 오류가 발생할 수 있으며 검색 기능이 저하될 수 있습니다.

권장되는 일반 Apache 구성 설정

다음 설정은 성능과 보안에 영향을 줍니다.

회선을 통한 메타 데이터 크기를 줄입니다.

HTTP 클라이언트는 HTTP 요청에서 압축된 데이터를 수신할 수 있음을 서버에 알릴 수 있습니다. Apache DEFLATE 필터를 사용으로 설정하면 카탈로그 및 매니페스트와 같은 메타 데이터의 전송 크기를 크게 줄일 수 있습니다. 카탈로그 및 매니페스트와 같은 메타 데이터는 종종 90%까지 압축됩니다.

```
AddOutputFilterByType DEFLATE text/html application/javascript text/css text/plain
```

파이프라인된 추가 요청을 허용합니다.

클라이언트가 연결을 닫지 않고도 더 많은 수의 파이프라인된 요청을 수행하도록 허용하려면 `MaxKeepAliveRequests` 값을 늘립니다.

```
MaxKeepAliveRequests 10000
```

응답에 대한 최대 대기 시간을 설정합니다.

프록시 시간 초과는 백엔드 저장소가 응답할 때까지 Apache가 기다리는 시간을 설정합니다. 대부분의 작업에서는 30초 정도가 적합합니다. 검색할 때 결과 수가 너무 많으면 시간이 상당히 오래 걸릴 수 있습니다. 이러한 검색을 수용하기 위해서는 시간 초과 값을 더 높게 설정해야 할 수 있습니다.

```
ProxyTimeout 30
```

전달 프록싱을 사용 안함으로 설정합니다.

전달 프록싱이 사용 안함으로 설정되었는지 확인합니다.

```
ProxyRequests Off
```

저장소 서버에 대한 캐싱 구성

캐싱 프록시 뒤에서 저장소 서버를 설정하는 최소 구성이 필요합니다. 카탈로그 속성 파일(“카탈로그 속성 파일에 대한 캐시 고려 사항” [47] 참조) 및 저장소 검색 결과(“검색을 위한 캐시 고려 사항” [48] 참조)를 제외하고 제공된 모든 파일은 고유하므로 필요한 경우 무기한 캐시해도 안전합니다. 또한 모든 저장소 응답에는 캐시 파일이 실수로 사용되지 않는 것을 방지하기 위해 적합한 HTTP 헤더를 포함합니다.

Apache를 캐싱 프록시로 구성하는 방법에 대한 자세한 내용은 Caching Guide를 참조하십시오.

`CacheRoot` 지시어를 사용하여 디렉토리가 캐시된 파일을 포함하도록 지정합니다. 지정된 디렉토리를 Apache 프로세스가 쓸 수 있는지 확인합니다. Apache가 이 디렉토리에 쓸 수 없는 경우 명시적인 오류 메시지가 출력되지 않습니다.

```
CacheRoot /tank/proxycache
```

Apache는 특정 디렉토리에 대해 캐싱을 사용으로 설정합니다. 다음 지시어와 같이 저장소 서버가 서버의 모든 컨텐츠를 캐시하도록 할 수도 있습니다.

```
CacheEnable disk /
```

`CacheMaxFileSize` 지시어를 사용하여 캐시할 최대 파일 크기를 설정합니다. 1MB의 Apache 기본값은 대부분의 저장소에서 너무 작을 수 있습니다. 다음 지시어는 캐시되는 파일의 최대 크기를 1GB로 설정합니다.

```
CacheMaxFileSize 1000000000
```

기본 파일 시스템에서 성능을 최적화하기 위해 디스크 내장 캐시의 디렉토리 구조를 조정합니다. ZFS 데이터 세트에서 다중 디렉토리 레벨은 하나의 디렉토리에 있는 파일 수보다 성능에 더 많은 영향을 줍니다. 따라서 각 디렉토리에 파일 수가 많은 하나의 디렉토리 레벨을 구성하십시오. `CacheDirLevels` 및 `CacheDirLength` 지시어를 사용하여 디렉토리 구조를 제어합니다. `CacheDirLevels`를 1로 설정합니다. `CacheDirLength`를 디렉토리 수와 디렉토리당 파일 수 간에 적절한 균형을 이룰 수 있는 값으로 설정합니다. 아래에 설정된 값 2는 4096개의 디렉토리를 생성합니다. 자세한 내용은 Disk-based Caching 설명서를 참조하십시오.

```
CacheDirLevels 1
CacheDirLength 2
```

카탈로그 속성 파일에 대한 캐시 고려 사항

저장소 카탈로그 속성 파일(`catalog.attrs`)에는 저장소 카탈로그의 현재 상태가 포함됩니다. 이 파일은 캐싱을 보장하도록 충분히 클 수 있습니다. 하지만 백엔드 저장소의 카탈로그

가 변경된 경우 이 파일이 사용되지 않을 수 있습니다. 다음 두 가지 방법 중 하나를 사용하여 이 문제를 해결할 수 있습니다.

- 이 파일을 캐시하지 않습니다. 이 솔루션은 추가 트래픽이 중요한 고려 사항이 아닌 고대역폭 환경에서 저장소 서버를 실행하는 경우에 가장 적합합니다. 다음 `httpd.conf` 파일 부분은 `catalog.attrs` 파일을 캐시하지 않도록 지정하는 방법을 보여 줍니다.

  ```
  <LocationMatch ".*/catalog.attrs">
          Header set Cache-Control no-cache
  </LocationMatch>
  ```

- 백 엔드 저장소의 카탈로그가 업데이트될 때마다 캐시에서 이 파일을 정렬합니다.

검색을 위한 캐시 고려 사항

패키지 저장소를 검색하면 요청에 따라 사용자 정의 응답이 생성됩니다. 따라서 검색 결과는 캐시하는 데 적합하지 않습니다. 저장소 서버는 검색 결과가 캐시에서 사용되지 않는 것을 방지하기 위해 적합한 HTTP 헤더를 설정합니다. 하지만 캐싱으로 얻을 수 있는 예상되는 대역폭 절감 효과는 크지 않습니다. 다음 `httpd.conf` 파일 부분은 검색 결과를 캐시하지 않도록 지정하는 방법을 보여 줍니다.

```
<LocationMatch ".*/search/\d/.*">
        Header set Cache-Control no-cache
</LocationMatch>
```

단순 접두어가 지정된 프록시 구성

이 예에서는 로드 균형 조정되지 않은 저장소 서버에 대한 기본 구성을 보여 줍니다. 이 예에서는 `http://pkg.example.com/myrepo`를 `internal.example.com:10000`에 연결합니다.

이 예에서 설명되지 않은 다른 등록 정보 설정에 대한 지침은 “웹 서버 액세스를 사용하여 여러 저장소 제공” [39]을 참조하십시오.

저장소 서버를 액세스할 수 있는 URL의 이름을 지정하는 `pkg/proxy_base` 설정을 사용하여 저장소 서버를 구성합니다. 다음 명령을 사용하여 `pkg/proxy_base`를 설정합니다.

```
$ svccfg -s pkg/server add repo
$ svccfg -s pkg/server:repo setprop pkg/proxy_base = astring: http://pkg.example.com/
myrepo
$ svcadm refresh pkg/server:repo
$ svcadm enable pkg/server:repo
```

`pkg`(5) 클라이언트는 네트워크 작업을 수행할 때 저장소 서버에 대해 20개의 병렬 연결을 엽니다. 저장소 스레드 수가 특정 시점에 서버에 대한 예상 연결 수와 일치하는지 확인합니다. 다음 명령을 사용하여 저장소당 스레드 수를 설정합니다.

```
$ svccfg -s pkg/server:repo setprop pkg/threads = 200
$ svcadm refresh pkg/server:repo
$ svcadm restart pkg/server:repo
```

nocanon을 사용하여 URL의 정규화를 억제합니다. 검색이 올바르게 작동하려면 이 설정이 중요합니다. 또한 백엔드 연결 수를 저장소 서버가 제공하는 스레드 수로 제한합니다. 다음 `httpd.conf` 파일 부분은 하나의 저장소 서버를 프록시하는 방법을 보여 줍니다.

```
Redirect /myrepo http://pkg.example.com/myrepo/
ProxyPass /myrepo/ http://internal.example.com:10000/ nocanon max=200
```

Oracle Solaris SSL 커널 프록시 및 SSL을 사용하여 웹 서버 통신을 암호화하고 속도를 향상시키는 방법에 대한 자세한 내용은 "Oracle Solaris 11.2의 네트워크 보안 "의 3 장, "웹 서버 및 Secure Sockets Layer 프로토콜"을 참조하십시오.

하나의 도메인 아래의 다중 저장소

프록시 뒤에서 저장소 서버를 실행하는 가장 중요한 이유는 하나의 도메인 이름에서 서로 다른 접두어로 여러 저장소를 쉽게 실행하기 위해서입니다. "단순 접두어가 지정된 프록시 구성" [48]의 예를 쉽게 확장하여 여러 저장소를 지원할 수 있습니다.

이 예에서는 하나의 도메인 이름에서 세 개의 서로 다른 접두어가 세 개의 서로 다른 패키지 저장소에 연결되어 있습니다.

- `http://pkg.example.com/repo_one`은 `internal.example.com:10000`에 연결됩니다.
- `http://pkg.example.com/repo_two`는 `internal.example.com:20000`에 연결됩니다.
- `http://pkg.example.com/xyz/repo_three`는 `internal.example.com:30000`에 연결됩니다.

pkg(5) 저장소 서버는 SMF 관리 서비스입니다. 따라서 동일한 호스트에서 여러 저장소 서버를 실행하려면 단순히 새로운 서비스 인스턴스를 만들기만 하면 됩니다.

```
$ svccfg -s pkg/server add repo1
$ svccfg -s pkg/server:repo1 setprop pkg/property=value
$ ...
```

이전 예와 같이 각 저장소 서버는 200개의 스레드로 실행됩니다.

```
Redirect /repo_one http://pkg.example.com/repo_one/
ProxyPass /repo_one/ http://internal.example.com:10000/ nocanon max=200

Redirect /repo_two http://pkg.example.com/repo_two/
ProxyPass /repo_two/ http://internal.example.com:20000/ nocanon max=200

Redirect /xyz/repo_three http://pkg.example.com/xyz/repo_three/
ProxyPass /xyz/repo_three/ http://internal.example.com:30000/ nocanon max=200
```

로드 균형 조정 구성

Apache 로드 밸런서 뒤에서 저장소 서버를 실행해야 할 수 있습니다. 로드 균형 조정의 한 가지 이점은 저장소 가용성을 증가시킨다는 것입니다. 이 절에서는 로드 균형 조정의 두 가지 예를 보여 줍니다.

저장소가 로드 균형 조정된 경우 로드 밸런서가 노드 간에 클라이언트를 전환할 때 문제를 방지하려면 모든 저장소의 카탈로그가 정확히 같아야 합니다. 카탈로그가 정확히 일치하게 하려면 "여러 개의 동일한 로컬 저장소 유지 관리" [32]의 설명에 따라 로드 균형 조정에 참여하는 저장소를 복제합니다.

로드 균형 조정된 단일 저장소 서버

이 예에서는 `http://pkg.example.com/myrepo`를 `internal1.example.com:10000` 및 `internal2.example.com:10000`에 연결합니다.

"단순 접두어가 지정된 프록시 구성" [48]과 같이 적합한 `proxy_base` 설정을 사용하여 저장소 서버를 구성합니다.

각 저장소가 실행 중인 스레드 수를 로드 밸런서 설정에 있는 저장소 수로 나눈 값으로 백엔드 연결 수를 제한합니다. 그렇지 않으면 Apache가 저장소에 대해 사용 가능한 것보다 많은 수의 연결을 열고 작동이 정지되어 성능이 저하될 수 있습니다. `max=` 매개변수를 사용하여 각 저장소에 대한 최대 병렬 연결 수를 지정합니다. 다음 예에서는 각각 200개의 스레드를 실행하는 두 개의 저장소를 보여 줍니다. 저장소 스레드 수를 설정하는 방법에 대한 예는 "단순 접두어가 지정된 프록시 구성" [48]을 참조하십시오.

```
<Proxy balancer://pkg-example-com-myrepo>
        # depot on internal1
        BalancerMember http://internal1.example.com:10000 retry=5 max=100

        # depot on internal2
        BalancerMember http://internal2.example.com:10000 retry=5 max=100
</Proxy>

Redirect /myrepo http://pkg.example.com/myrepo/
ProxyPass /myrepo/ balancer://pkg-example-com-myrepo/ nocanon
```

로드 균형 조정된 단일 저장소 서버 및 로드 균형 조정되지 않은 단일 저장소 서버

이 예에는 로드 균형 조정된 저장소 서버 설정 및 로드 균형 조정되지 않은 저장소 서버 설정을 호스트하는 저장소 서버에 대해 `httpd.conf` 파일에 추가해야 하는 모든 지시어가 포함되어 있습니다.

이 예에서는 하나의 도메인 이름에서 두 개의 서로 다른 접두어가 세 개의 서로 다른 패키지 저장소에 연결되어 있습니다.

- `http://pkg.example.com/repo_one`은 `internal1.example.com:10000` 및 `internal2.example.com:10000`에 연결됩니다.
- `http://pkg.example.com/repo_two`는 `internal1.example.com:20000`에 연결됩니다.

```
AddOutputFilterByType DEFLATE text/html application/javascript text/css text/plain

AllowEncodedSlashes NoDecode

MaxKeepAliveRequests 10000

ProxyTimeout 30

ProxyRequests Off

<Proxy balancer://pkg-example-com-repo_one>
        # depot on internal1
        BalancerMember http://internal1.example.com:10000 retry=5 max=100

        # depot on internal2
        BalancerMember http://internal2.example.com:10000 retry=5 max=100
</Proxy>

Redirect /repo_one http://pkg.example.com/repo_one/
ProxyPass /repo_one/ balancer://pkg-example-com-repo_one/ nocanon
Redirect /repo_two http://pkg.example.com/repo_two/
ProxyPass /repo_two/ http://internal.example.com:20000/ nocanon max=200
```

HTTPS 저장소 액세스 구성

모든 클라이언트가 HTTP를 통해 패키지를 제공하도록 구성된 저장소에서 패키지를 다운로드할 수 있습니다. 액세스를 제한해야 하는 경우도 있습니다. 저장소 액세스를 제한하는 한 가지 방법은 클라이언트 인증서를 지원하는 SSL 사용 Apache 인스턴스 뒤에서 저장소 서버를 실행하는 것입니다.

SSL을 사용하면 다음과 같은 이점이 있습니다.

- 클라이언트와 서버 간에 암호화된 패키지 데이터 전송을 보장합니다.
- 클라이언트가 서버에 제공하는 인증서를 기준으로 저장소 액세스 권한을 부여할 수 있습니다.

보안 저장소 서버를 설정하려면 사용자 정의 인증서 체인을 만들어야 합니다.

1. 인증서 체인의 헤드인 CA(인증 기관)를 만듭니다.
2. 저장소에 액세스하도록 허용된 클라이언트에게 이 CA의 인증서를 발급합니다.

CA 복사본 한 개는 저장소 서버에 저장됩니다. 클라이언트가 서버에 인증서를 제공할 때마다 서버의 CA에서 클라이언트 인증서를 검증하여 액세스 권한을 부여할지 확인합니다.

이 절에서는 인증서 체인을 만들고 클라이언트 인증서를 확인하도록 Apache 프론트 엔드를 구성하는 다음 단계를 설명합니다.

- 키 저장소 만들기
- 클라이언트 인증서에 대한 인증 기관 만들기
- Apache 구성 파일에 SSL 구성 추가
- 자체 서명된 서버 인증 기관 만들기
- PKCS12 키 저장소 만들기

Oracle Solaris의 Apache 웹 서버 권한에 대한 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안 "의 "확장 권한을 사용하여 리소스 잠금"을 참조하십시오.

키 저장소 만들기

인증서와 키를 관리하려면 키 저장소를 만듭니다. 키 저장소는 CA, CA 키 및 클라이언트 인증서와 키를 저장합니다.

키 저장소 관리에 사용되는 도구는 `pktool`입니다. 자세한 내용은 `pktool`(1) 매뉴얼 페이지를 참조하십시오.

`pktool`의 기본 키 저장소 위치는 `/var/user/`*username*입니다. 여기서 *username*은 현재 시스템 사용자의 이름입니다. 여러 사용자가 키 저장소를 관리하는 경우 이 키 저장소 기본 위치는 문제가 될 수 있습니다. 또한 인증서 혼동을 방지하려면 IPS 패키지 저장소 관리에 전용 키 저장소가 있어야 합니다. IPS 패키지 저장소의 `pktool` 키 저장소에 대해 사용자 정의 위치를 설정하려면 환경 변수 `SOFTTOKEN_DIR`을 설정합니다. 필요에 따라 `SOFTTOKEN_DIR` 변수를 재설정하여 여러 키 저장소를 관리합니다.

다음 명령을 사용하여 키 저장소에 대한 디렉토리를 만듭니다. 여러 사용자가 키 저장소를 관리해야 하는 경우 소유자, 그룹 및 사용 권한을 적절하게 설정합니다.

```
$ mkdir /path-to-keystore
$ export SOFTTOKEN_DIR=/path-to-keystore
```

키 저장소 액세스는 `pktool` 명령을 호출할 때마다 입력해야 하는 문장암호로 보호됩니다. 새로 만든 키 저장소의 기본 문장암호는 `changeme`입니다. `changeme` 문장암호를 보다 안전한 문장암호로 변경해야 합니다.

다음 명령을 사용하여 키 저장소에 대한 문장암호(PIN)를 설정합니다.

```
$ pktool setpin
Enter token passphrase: changeme
Create new passphrase:
Re-enter new passphrase:
```

```
Passphrase changed.
$ ls /path-to-keystore
pkcs11_softtoken
```

클라이언트 인증서에 대한 인증 기관 만들기

CA는 인증서 체인의 최상위 레벨 인증서입니다. CA는 클라이언트 인증서를 생성하고 클라이언트가 저장소에 액세스하기 위해 제공하는 인증서를 검증해야 합니다.

타사 CA는 VeriSign 등의 일부 신뢰할 수 있는 회사에서 관리됩니다. 이 신뢰할 수 있는 관리를 사용하면 클라이언트가 해당 CA 중 하나에서 서버의 ID를 확인할 수 있습니다. 이 절의 예에는 저장소 서버의 ID 확인이 포함되어 있지 않습니다. 이 예에서는 클라이언트 인증서 확인만 보여 줍니다. 따라서 이 예에서는 자체 서명된 인증서를 사용하여 CA를 만들고 타사 CA를 사용하지 않습니다.

CA에는 CN(일반 이름)이 필요합니다. 하나의 저장소만 실행하는 경우 CN을 조직 이름(예: "Oracle Software Delivery")으로 설정하는 것이 좋습니다. 여러 저장소가 있는 경우 각 저장소에 해당 CA가 있어야 합니다. 이 경우 CA를 만들 저장소를 고유하게 식별하는 이름으로 CN을 설정합니다. 예를 들어, 릴리스 저장소와 지원 저장소가 있는 경우 릴리스 CA의 인증서만 릴리스 저장소에 대한 액세스를 허용하고 지원 CA의 인증서만 지원 저장소에 대한 액세스를 허용합니다.

키 저장소에서 인증서를 식별하려면 인증서에 대한 설명 레이블을 설정합니다. 모범 사례는 인증서 레이블을 *CN*_ca로 설정하는 것입니다. 여기서 *CN*은 인증서 CN입니다.

다음 명령을 사용하여 CA 인증서를 만듭니다. 여기서 *name*은 인증서 CN이고 *CAlabel*은 인증서 레이블입니다.

```
$ pktool gencert label=CAlabel subject="CN=name" serial=0x01
```

CA는 키 저장소에 저장됩니다. 다음 명령을 사용하여 키 저장소 컨텐츠를 표시합니다.

```
$ pktool list
```

"Apache 구성 파일에 SSL 구성 추가" [55]의 설명에 따라 Apache를 구성하는 경우 키 저장소에서 CA 인증서를 추출해야 합니다. 다음 명령을 사용하여 `ca_file.pem`이라는 파일로 CA 인증서를 추출합니다.

```
$ pktool export objtype=cert label=CAlabel outformat=pem \
outfile=ca_file.pem
```

저장소 액세스에 사용되는 클라이언트 인증서 만들기

CA를 생성한 후 클라이언트 인증서를 생성할 수 있습니다.

인증서 서명 요청 생성

클라이언트 인증서를 생성하려면 CSR(인증서 서명 요청)을 생성합니다. CSR에는 서버에 안전하게 전달해야 하는 모든 정보가 포함되어 있습니다.

클라이언트가 자신이 발급한 유효한 인증서를 가지고 있는지만 확인하려는 경우에는 정보를 인코딩할 필요가 없습니다. 클라이언트가 해당 인증서를 서버에 제공하면 서버가 CA에서 인증서를 검증하여 클라이언트 인증서가 사용자에 의해 생성되었는지 확인합니다. 그러나 SSL에는 CSR에 대한 `subject`가 필요합니다. 다른 정보를 서버에 전달할 필요가 없는 경우 `subject`를 인증서가 발급된 국가로 설정하면 됩니다. 예를 들어, `subject`를 `C=US`로 설정할 수 있습니다.

모범 사례는 클라이언트의 사용자 이름을 인증서에 인코딩하여 서버가 클라이언트를 식별할 수 있게 하는 것입니다. 사용자 이름은 저장소 액세스 권한을 부여할 사용자의 이름입니다. 이 목적으로 CN을 사용할 수 있습니다. “인증서 키 추출” [55]의 설명에 따라 최종 인증서에 대한 키를 찾고 추출할 수 있도록 이 CSR에 대한 레이블을 지정합니다.

다음 명령을 사용하여 CSR을 생성합니다.

```
$ pktool gencsr subject="C=US,CN=username" label=label format=pem \
outcsr=cert.csr
```

다음 OpenSSL 명령을 사용하여 `cert.csr` 파일에서 CSR을 검사합니다.

```
$ openssl req -text -in cert.csr
```

CSR에 서명

인증서를 만들려면 CA가 CSR에 서명해야 합니다. CSR에 서명하려면 다음 정보를 제공합니다.

- “클라이언트 인증서에 대한 인증 기관 만들기” [53]와 같이 `gencert` 명령을 사용하여 CA를 만들 때 `subject`에 사용한 것과 동일한 문자열로 인증서 `issuer`를 설정합니다.
- 16진수 일련 번호로 설정합니다. 이 예에서는 CA 일련 번호가 `0x01`로 설정되었으므로 첫번째 클라이언트 인증서에 일련 번호 `0x02`가 지정되어야 합니다. 새 클라이언트 인증서를 생성할 때마다 일련 번호를 증분시킵니다.

  각 CA와 종속 클라이언트 인증서에는 해당 일련 번호 세트가 있습니다. 키 저장소에 여러 CA가 구성되어 있는 경우 클라이언트 인증서 일련 번호를 올바르게 설정해야 합니다.
- `signkey`를 키 저장소의 CA 레이블로 설정합니다.
- `outcert`를 인증서 파일의 이름으로 설정합니다. 모범 사례는 액세스할 저장소를 따서 인증서와 키의 이름을 지정하는 것입니다.

다음 명령을 사용하여 CSR에 서명합니다.

```
$ pktool signcsr signkey=CAlabel csr=cert.csr  \
serial=0x02 outcert=reponame.crt.pem issuer="CN=name"
```

reponame.crt.pem 파일에 인증서가 생성됩니다. 다음 OpenSSL 명령을 사용하여 인증서를 검사합니다.

```
$ openssl x509 -text -in reponame.crt.pem
```

인증서 키 추출

키 저장소에서 이 인증서에 대한 키를 추출합니다. "인증서 서명 요청 생성" [54]에서 gencsr을 실행하여 CSR을 생성할 때 지정한 것과 동일한 레이블 값으로 label을 설정합니다. 다음 명령을 사용하여 키 저장소에서 키를 내보냅니다.

```
$ pktool export objtype=key label=label outformat=pem \
outfile=reponame.key.pem
```

SSL로 보호된 저장소에 액세스해야 하는 클라이언트 시스템에 인증서와 키를 전송합니다.

클라이언트 시스템이 보호된 저장소에 액세스하도록 설정

SSL로 보호된 저장소에 액세스하려면 클라이언트 시스템에 인증서 및 키 복사본이 있어야 하며 게시자 구성에 인증서와 키를 지정해야 합니다.

각 클라이언트 시스템에 인증서(*reponame*.crt.pem)와 키(*reponame*.key.pem)를 복사합니다. 예를 들어, 각 클라이언트의 /var/pkg/ssl 디렉토리에 복사할 수 있습니다.

다음 명령을 사용하여 게시자 구성에 생성된 인증서와 키를 지정합니다.

```
$ pkg set-publisher -k reponame.key.pem -c reponame.crt.pem \
-p https://repolocation
```

SSL 인증은 HTTPS 저장소 URI에 대해서만 지원됩니다. 파일 저장소 URI에 대한 SSL 인증은 지원되지 않습니다.

Apache 구성 파일에 SSL 구성 추가

저장소에 클라이언트 인증서 기반 인증을 사용하려면 먼저 "저장소 서버 Apache 구성" [45]의 설명에 따라 일반 저장소 서버 Apache 구성을 설정합니다. 그런 다음 httpd.conf 파일의 끝에 다음 SSL 구성을 추가합니다.

```
# Let Apache listen on the standard HTTPS port
Listen 443

# VirtualHost configuration for request on port 443
<VirtualHost 0.0.0.0:443>
        # DNS domain name of the server, needs to match your server certificate
        ServerName pkg-sec.example.com
```

```
# enable SSL
SSLEngine On

# Location of the server certificate and key.
# You either have to get one from a certificate signing authority like
# VeriSign or create your own CA for testing purposes (see "Creating a
# Self-Signed CA for Testing Purposes")
SSLCertificateFile /path/to/server.crt
SSLCertificateKeyFile /path/to/server.key

# Intermediate CA certificate file. Required if your server certificate
# is not signed by a top-level CA directly but an intermediate authority
# Comment out this section if you are using a test certificate or your
# server certificate doesn't require it.
# For more info:
# http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslcertificatechainfile
SSLCertificateChainFile /path/to/ca_intermediate.pem

# CA certs for client verification.
# This is where the CA certificate created in step 3 needs to go.
# If you have multiple CAs for multiple repos, just concatenate the
# CA certificate files
SSLCACertificateFile /path/to/ca_cert.pem

# If the client presents a certificate, verify it here. If it doesn't,
# ignore.
# This is required to be able to use client-certificate based and
# anonymous SSL traffic on the same VirtualHost.
# This statement could also go into the <Location> tags but putting it
# here avoids re-negotiation which can cause security issues with older
# servers/clients:
# http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2009-3555
SSLVerifyClient optional

<Location /repo>
        SSLVerifyDepth 1
        # This is the SSL requirement for this location.
        # Requirements can be made based on various information encoded
        # in the certificate. Two variants are the most useful for use
        # with IPS repositories:
        # a) SSLRequire ( %{SSL_CLIENT_I_DN_CN} =~ m/reponame/ )
        #    only allow access if the CN in the client certificate matches
        #    "reponame", useful for different certificates for different
        #    repos
        #
        # b) SSLRequire ( %{SSL_CLIENT_VERIFY} eq "SUCCESS" )
        #    grant access if clients certificate is signed by one of the
        #    CAs specified in SSLCACertificateFile
        SSLRequire ( %{SSL_CLIENT_VERIFY} eq "SUCCESS" )

        # proxy request to depot running at internal.example.com:12345
        ProxyPass http://internal.example.com:12345 nocanon max=500
</Location>
```

```
</VirtualHost>
```

자체 서명된 서버 인증 기관 만들기

테스트 목적에는 타사 CA 대신 자체 서명된 서버 CA(인증 기관)를 사용할 수 있습니다. Apache에 대한 자체 서명된 서버 CA를 만드는 단계는 "클라이언트 인증서에 대한 인증 기관 만들기" [53]의 설명에 따라 클라이언트 인증서에 대한 CA를 만드는 단계와 유사합니다.

다음 명령을 사용하여 서버 CA를 만듭니다. `subject`를 서버의 DNS 이름으로 설정합니다.

```
$ pktool gencert label=apacheCA subject="CN=apachetest" \
serial=0x01
```

다음 명령을 사용하여 서버 CA에 대한 CSR을 만듭니다. 여러 이름으로 서버에 액세스할 수 있거나 해당 IP 주소로 직접 사용할 수 있게 하려는 경우 OpenSSL 설명서의 Subject Alternative Name의 설명에 따라 `subjectAltNames` 지시어를 사용합니다.

```
$ pktool gencsr label=apache subject="CN=pkg-sec.internal.example.com" \
altname="IP=192.168.1.1,DNS=pkg-sec.internal.example.com" \
format=pem outcsr=apache.csr
```

다음 명령을 사용하여 CSR에 서명합니다. `SSLCertificateFile`에 대해 `server.crt`를 사용합니다.

```
$ pktool signcsr signkey=apacheCA csr=apache.csr serial=0x02 \
outcert=server.crt issuer="CN=apachetest"
```

다음 명령을 사용하여 키를 추출합니다. `SSLCertificateKeyFile`에 대해 `server.key`를 사용합니다.

```
$ pktool export objtype=key label=apache outformat=pem \
outfile=server.key
```

클라이언트가 이 서버 키를 허용하게 하려면 클라이언트 시스템의 허용된 CA 디렉토리에 CA 인증서(`apacheCA`)를 추가하고 `ca-certificates` 서비스를 다시 시작하여 필요한 OpenSSL 링크를 만듭니다.

다음 명령을 사용하여 CA 인증서를 추출합니다.

```
$ pktool export label=apacheCA objtype=cert outformat=pem \
outfile=test_server_ca.pem
```

클라이언트 시스템의 CA 인증서 디렉토리에 CA 인증서를 복사합니다.

```
$ cp /path-to/test_server_ca.pem /etc/certs/CA/
```

CA 인증서 서비스를 다시 시작합니다.

```
$ svcadm refresh ca-certificates
```

계속하기 전에 새 CA 인증서가 연결되었는지 확인합니다. 새로 고치면 ca-certificate 서비스가 /etc/openssl/certs 디렉토리의 링크를 재작성합니다. 다음 명령을 실행하여 새 CA 인증서가 연결되었는지 확인합니다.

```
$ ls -l /etc/openssl/certs | grep test_server_ca.pem
lrwxrwxrwx   1 root     root          40 May  1 09:51 e89d96e0.0 -> ../../certs/CA/
test_server_ca.pem
```

해시 값 e89d96e0.0은 인증서 주체를 기반으로 하기 때문에 다를 수도 있습니다.

Firefox를 사용하여 보안 저장소에 액세스할 PKCS12 키 저장소 만들기

"저장소 액세스에 사용되는 클라이언트 인증서 만들기" [53]에서 만든 PEM 인증서는 pkg 클라이언트를 사용하여 보안 저장소에 액세스하는 데 작동합니다. 그러나 BUI(브라우저 사용자 인터페이스)에 액세스하려면 인증서와 키를 Firefox가 가져올 수 있는 형식으로 변환해야 합니다. Firefox는 PKCS12 키 저장소를 허용합니다.

다음 OpenSSL 명령을 사용하여 Firefox에 대한 PKCS12 키 저장소를 만듭니다.

```
$ openssl pkcs12 -export -in /path-to/certificate.pem \
-inkey /path-to/key.pem -out name.p12
```

방금 만든 PKCS12 키 저장소를 가져오려면 다음 Firefox 메뉴, 탭 및 버튼을 선택합니다. Edit(편집) > Preferences(기본 설정) > Advanced(고급) > Encryption(암호화) > View certificates(인증서 보기) > Authorities(기관) > Import(가져오기).

한 번에 하나씩 인증서를 가져옵니다.

전체 보안 저장소 예

이 예에서는 repo1, repo2 및 repo3이라는 세 개의 보안 저장소를 구성합니다. repo1 및 repo2 저장소는 전용 인증서로 구성됩니다. 따라서 repo1에 대한 인증서는 repo2에서 작동하지 않고 repo2에 대한 인증서는 repo1에서 작동하지 않습니다. repo3 저장소는 두 인증서 중 하나를 허용하도록 구성됩니다.

이 예에서는 Apache 인스턴스에 적합한 서버 인증서를 이미 사용할 수 있다고 가정합니다. Apache 인스턴스에 대한 서버 인증서가 없는 경우 "자체 서명된 서버 인증 기관 만들기" [57]에서 테스트 인증서 만들기 지침을 참조하십시오.

https://pkg-sec.example.com/repo1, https://pkg-sec.example.com/repo2 및 https://pkg-sec.example.com/repo3 아래에 세 개의 저장소가 설정됩니다. 이러한 저장소는 http://internal.example.com의 포트 10001, 10002 및 10003에 각각 설정된 저장소 서버를 가리킵니다. "키 저장소 만들기" [52]의 설명에 따라 SOFTTOKEN_DIR 환경 변수가 올바르게 설정되었는지 확인합니다.

▼ 보안 저장소를 구성하는 방법

1. **repo1에 대한 CA 인증서를 만듭니다.**

```
$ pktool gencert label=repo1_ca subject="CN=repo1" serial=0x01
$ pktool export objtype=cert label=repo1_ca outformat=pem \
outfile=repo1_ca.pem
```

2. **repo2에 대한 CA 인증서를 만듭니다.**

```
$ pktool gencert label=repo2_ca subject="CN=repo2" serial=0x01
$ pktool export objtype=cert label=repo2_ca outformat=pem \
outfile=repo2_ca.pem
```

3. **결합된 CA 인증서 파일을 만듭니다.**

```
$ cat repo1_ca.pem > repo_cas.pem
$ cat repo2_ca.pem >> repo_cas.pem
$ cp repo_cas.pem /path-to-certs
```

4. **사용자 myuser가 repo1 저장소에 액세스하도록 허용하는 클라이언트 인증서/키 쌍 하나를 만듭니다.**

```
$ pktool gencsr subject="C=US,CN=myuser" label=repo1_0001 format=pem \
outcsr=repo1_myuser.csr
$ pktool signcsr signkey=repo1_ca csr=repo1_myuser.csr  \
serial=0x02 outcert=repo1_myuser.crt.pem issuer="CN=repo1"
$ pktool export objtype=key label=repo1_0001 outformat=pem \
outfile=repo1_myuser.key.pem
$ cp repo1_myuser.key.pem /path-to-certs
$ cp repo1_myuser.crt.pem /path-to-certs
```

5. **사용자 myuser가 repo2 저장소에 액세스하도록 허용하는 클라이언트 인증서/키 쌍 하나를 만듭니다.**

```
$ pktool gencsr subject="C=US,CN=myuser" label=repo2_0001 format=pem \
outcsr=repo2_myuser.csr
$ pktool signcsr signkey=repo2_ca csr=repo2_myuser.csr  \
serial=0x02 outcert=repo2_myuser.crt.pem issuer="CN=repo2"
$ pktool export objtype=key label=repo2_0001 outformat=pem \
outfile=repo2_myuser.key.pem
$ cp repo2_myuser.key.pem /path-to-certs
$ cp repo2_myuser.crt.pem /path-to-certs
```

6. **Apache를 구성합니다.**
   `httpd.conf` 파일의 끝에 다음 SSL 구성을 추가합니다.

```
# Let Apache listen on the standard HTTPS port
Listen 443

<VirtualHost 0.0.0.0:443>
        # DNS domain name of the server
```

```
        ServerName pkg-sec.example.com

        # enable SSL
        SSLEngine On

        # Location of the server certificate and key.
        # You either have to get one from a certificate signing authority like
        # VeriSign or create your own CA for testing purposes (see "Creating a
        # Self-Signed CA for Testing Purposes")
        SSLCertificateFile /path/to/server.crt
        SSLCertificateKeyFile /path/to/server.key

        # Intermediate CA certificate file. Required if your server certificate
        # is not signed by a top-level CA directly but an intermediate authority.
        # Comment out this section if you don't need one or if you are using a
        # test certificate
        SSLCertificateChainFile /path/to/ca_intermediate.pem

        # CA certs for client verification.
        # This is where the CA certificate created in step 3 needs to go.
        # If you have multiple CAs for multiple repos, just concatenate the
        # CA certificate files
        SSLCACertificateFile /path/to/certs/repo_cas.pem

        # If the client presents a certificate, verify it here. If it doesn't,
        # ignore.
        # This is required to be able to use client-certificate based and
        # anonymous SSL traffic on the same VirtualHost.
        SSLVerifyClient optional

        <Location /repo1>
                SSLVerifyDepth 1
                SSLRequire ( %{SSL_CLIENT_I_DN_CN} =~ m/repo1/ )
                # proxy request to depot running at internal.example.com:10001
                ProxyPass http://internal.example.com:10001 nocanon max=500
        </Location>

        <Location /repo2>
                SSLVerifyDepth 1
                SSLRequire ( %{SSL_CLIENT_I_DN_CN} =~ m/repo2/ )
                # proxy request to depot running at internal.example.com:10002
                ProxyPass http://internal.example.com:10002 nocanon max=500
        </Location>

        <Location /repo3>
                SSLVerifyDepth 1
                SSLRequire ( %{SSL_CLIENT_VERIFY} eq "SUCCESS" )
                # proxy request to depot running at internal.example.com:10003
                ProxyPass http://internal.example.com:10003 nocanon max=500
        </Location>

</VirtualHost>
```

7. **repo1에 대한 액세스를 테스트합니다.**

```
$ pkg set-publisher -k /path-to-certs/repo1_myuser.key.pem \
-c /path-to-certs/repo1_myuser.crt.pem \
-p https://pkg-sec.example.com/repo1/
```

8. **repo2에 대한 액세스를 테스트합니다.**

```
$ pkg set-publisher -k /path-to-certs/repo2_myuser.key.pem \
-c /path-to-certs/repo2_myuser.crt.pem \
-p https://pkg-sec.example.com/repo2/
```

9. **repo3에 대한 액세스를 테스트합니다.**

repo1 인증서를 사용하여 repo3에 대한 액세스를 테스트합니다.

```
$ pkg set-publisher -k /path-to-certs/repo1_myuser.key.pem \
-c /path-to-certs/repo1_myuser.crt.pem \
-p https://pkg-sec.example.com/repo3/
```

repo2 인증서를 사용하여 repo3에 대한 액세스를 테스트합니다.

```
$ pkg set-publisher -k /path-to-certs/repo2_myuser.key.pem \
-c /path-to-certs/repo2_myuser.crt.pem \
-p https://pkg-sec.example.com/repo3/
```

색인