Working With HP-UX Depots

Bob Campbell
Engineer
Hewlett-Packard

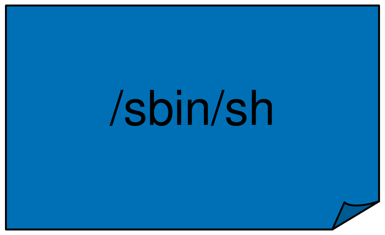
Topics to be covered

- A week of SD-UX in 15 minutes
  - Object types
  - Commands
  - Selection Methods
- Depots in depth
- Building depots
  - Installation/Recovery depots
  - Reactive patching depots
  - Proactive patching depots
- More tasks and examples

SD-UX Object Types

SD-UX Object Types

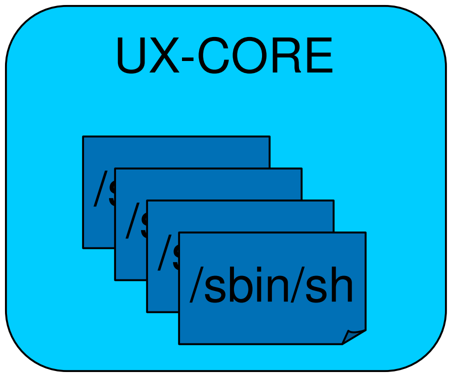
File – Just what you think they are



Files are generally delivered directly into place by SD. Each file object has preset attributes including type, ownership, permissions, modification date, and checksum.

SD-UX Object Types

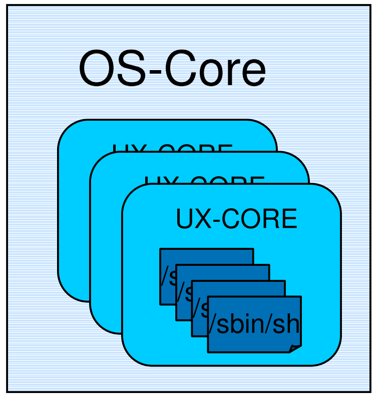
Fileset – Collections of architecture-specific files



Fileset objects contain files and optional control scripts. Multiple instances of a fileset for specific architectures (multi-streaming). The smallest installable object, but cannot exist alone.

SD-UX Object Types

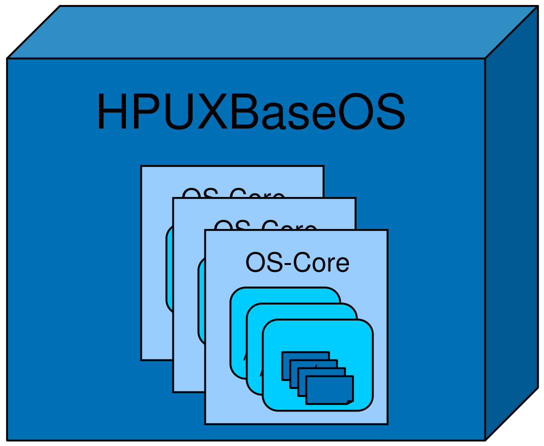
Product – Collections of filesets



Product objects uniquely own one or more filesets and may include optional control scripts. A product containing HP-UX components is built to contain filesets for all supported architectures.

SD-UX Object Types

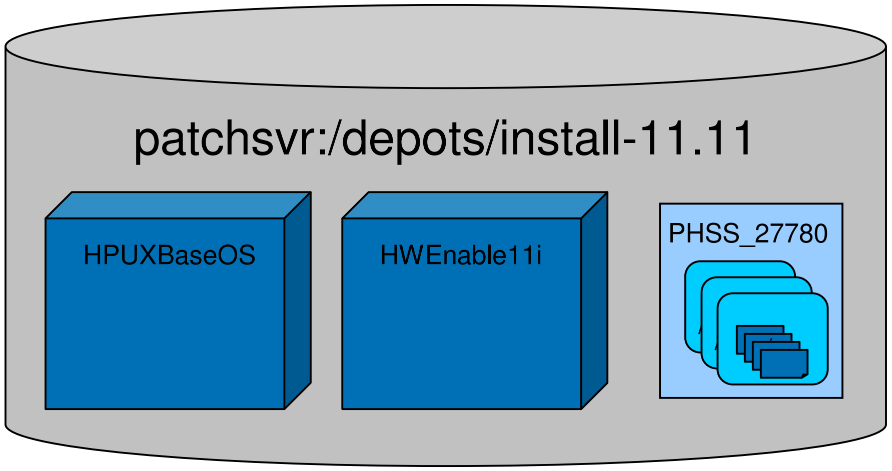
Bundle – Collection of products and/or filesets



Bundles are best thought of as being paper grocery bags. Useful for consolidating collections, but having little value otherwise. Multiple bundles can "contain" a product or fileset.

SD-UX Object Types

Depot – Repository for products and/or bundles

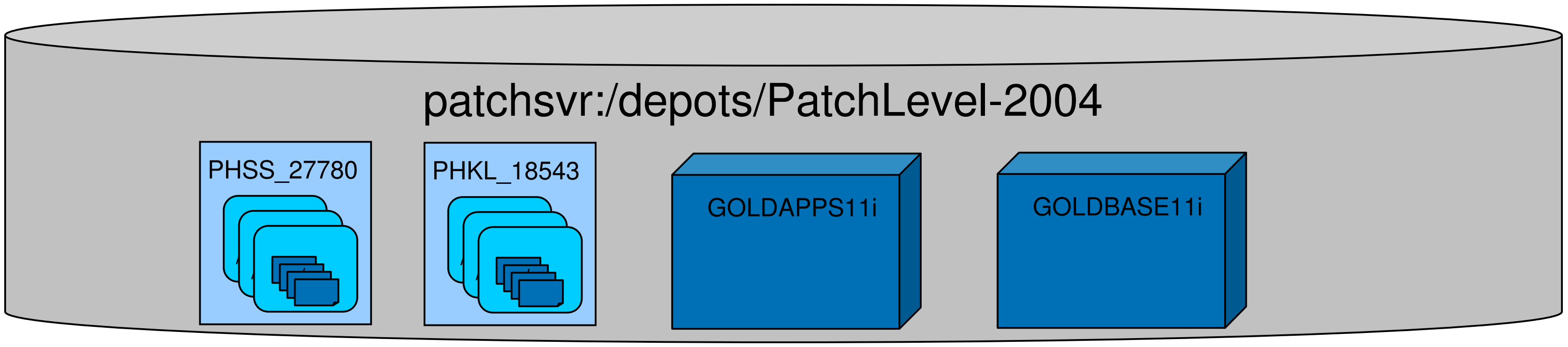


All objects exist in a depot before installation. Depots can be network accessible or local, multi-release or release-specific, compressed or full-sized, restricted or open

SD-UX Commands

swlist(1M)

List SD-UX objects



This command allows users to list depots on systems, bundles and products in depots, files in filesets, and even the attributes of each file.

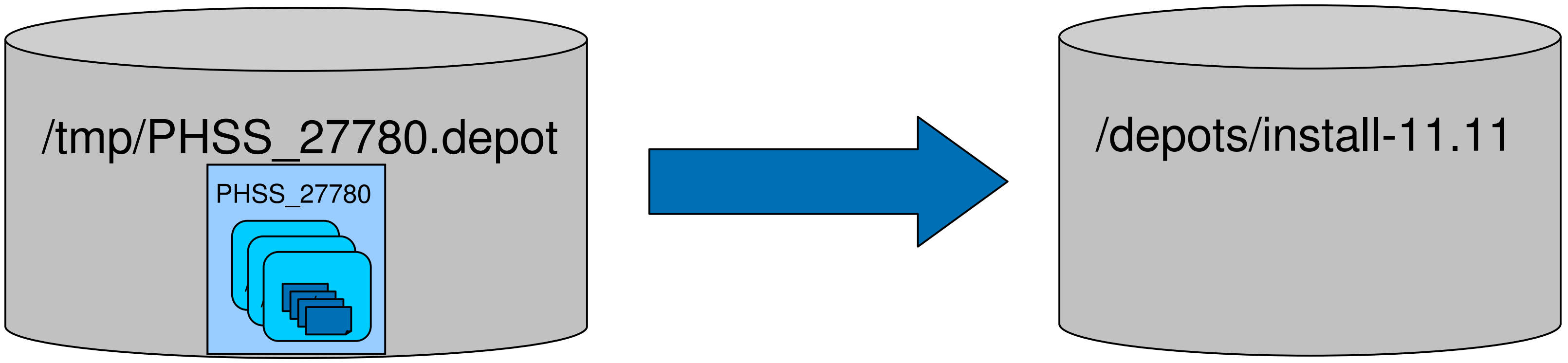
```
swlist -l depot @ patchsvr

swlist -d @ patchsvr:/depots/PatchLevel-2004

swlist -dl file -a cksum PHSS_27780 @ /tmp/PHKL_27780.depot
```

swcopy(1M)

Copy objects between depots

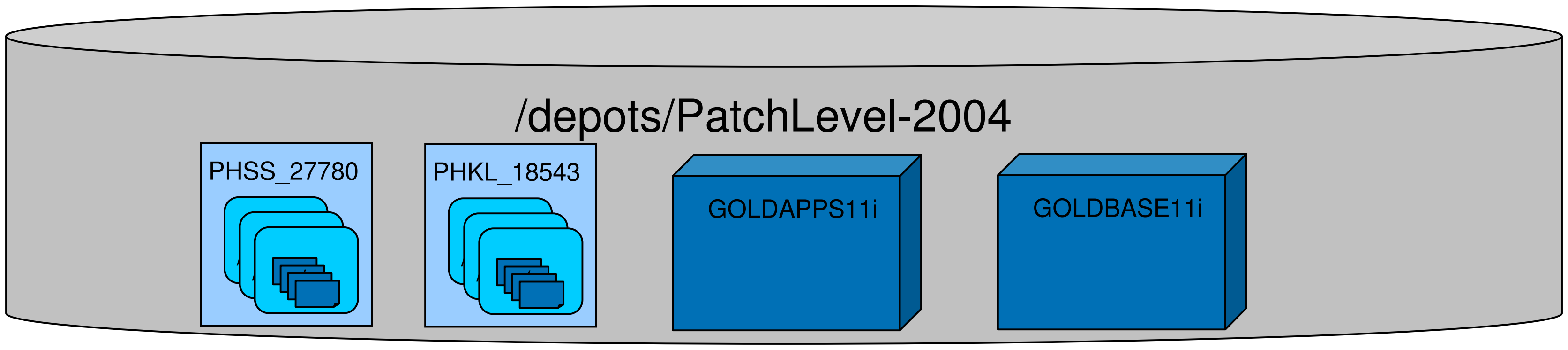


The swcopy command copies software from one depot to another. The target depot can be automatically created if it does not exist and dependencies can be automatically selected. Over 50 options exist!!!

```
swcopy -s /tmp/PHSS_27780.depot \* @ /depots/install-11.11
```

swremove(1M)

Remove objects from depots (or systems)

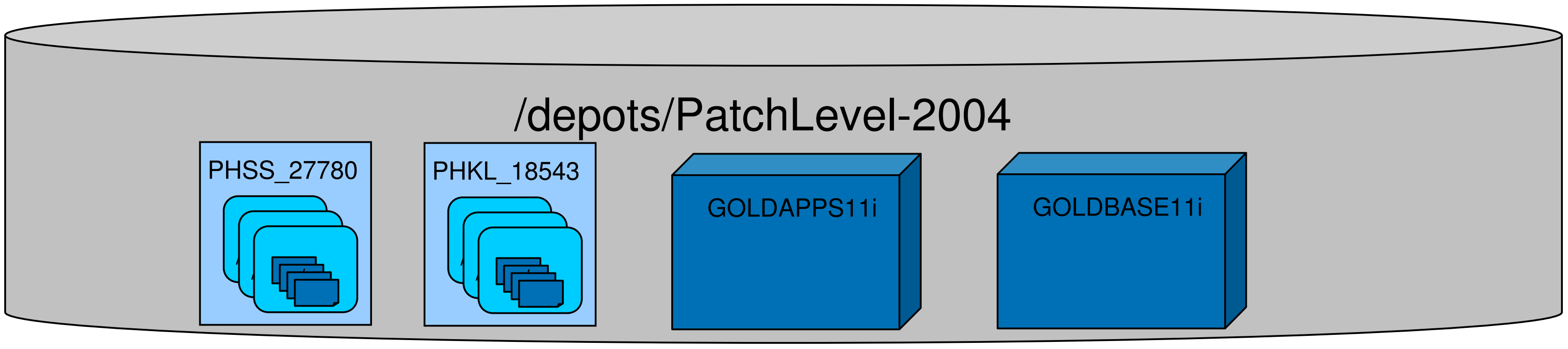


The swremove command deletes software from a depot or system. When all objects are removed the depot is deleted. It is strongly recommended that depots are managed at the product and bundle level (do not remove individual filesets).

```
swremove PHKL_18543 @ /depots/PatchLevel-2004
```

swreg(1M)

Register depots for network access



A depot can always be accessed from the local system, but must be registered to allow remote systems to interact with it.

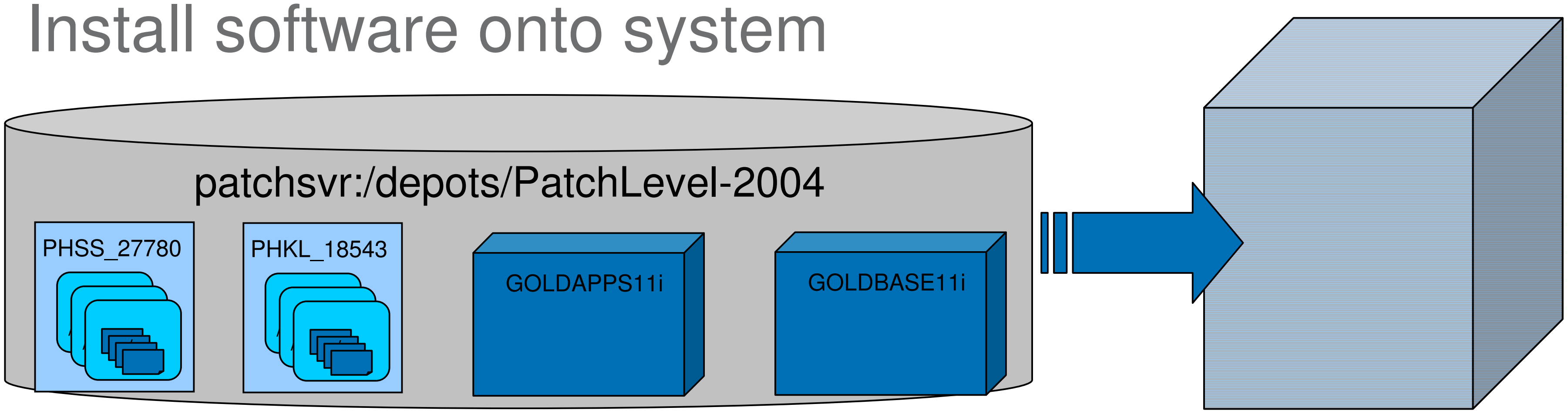
```
swreg -l depot /depots/PatchLevel-2004
```

```
swreg -u -l depot /depots/PatchLevel-2003
```

Unregistering a depot does not remove any of its contents, only visibility!

swinstall(1M)

Install software onto system

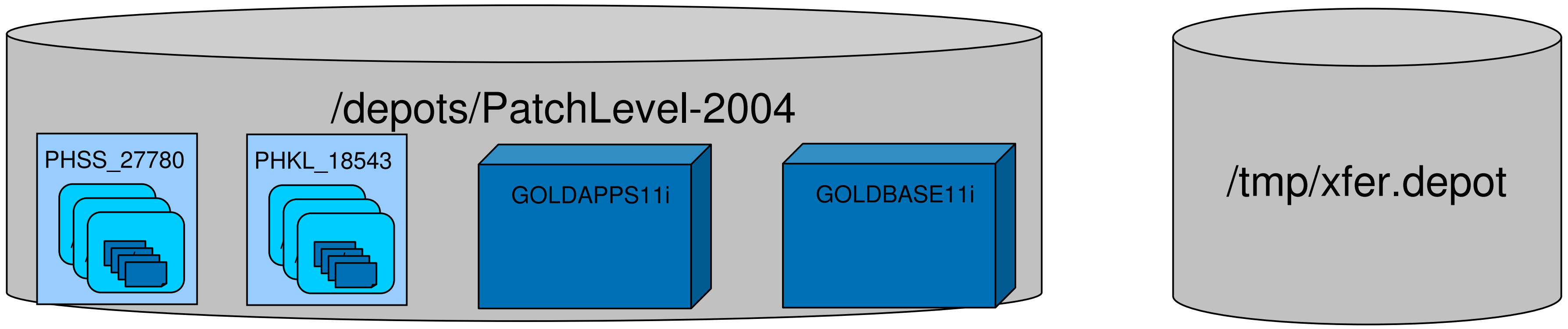


While we manage software in depots, it does not become useful until it is installed onto a system. swinstall is a very complex command that will be used in different ways from different depots.

```
swinstall -s patchsvr:/depots/PatchLevel-2004
          -x autoreboot=true -x patch_match_target=true
```

swpackage(1M)

Create depots from scratch!



Content has to come from somewhere. Before it can be swcopied to new depots an SD object must first be created using swpackage. Can create an ftp-able depot or allow two depots to share files. Most complex area of a complex world!

```
swpackage -s /depots/PatchLevel-2004 -x media_type=tape \* @
/tmp/xfer.depot
```

software package builder

swpackage for the masses!

If the description of swpackage has discouraged you, but you are interested in packaging your own software and applications as SD-UX objects take a look at Software Package Builder (spb)! It provides a graphical interface and integrated knowledge of packaging policies.

For more information see:

http://docs.hp.com/hpux/onlinedocs/5187-4539/repackage_whitepaper.pdf

or download today from:

http://software.hp.com

Selection Methods

Explicit Selection

Calling out by name

The obvious method of selection is to choose by name. A software product can be explicitly selected directly, or by explicitly selecting a bundle containing the product.

```
PHKL_29985
HWEnable11i
```

Explicit selection can include wildcards and/or filters.

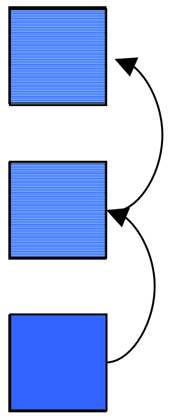
```
GOLD\*
*,c=patch
```

Refer to the sd(5) or swinstall(1m) man pages for full details on using complex software specifications.

Explicit patch selection

You can always change your mind if it is newer

An explicit selection implies a specific desire. If a depot has more than one member of a supersession chain, explicit selection will succeed if newer patches have not been explicitly selected.



Matching Operations

Implicitly give me more of what I already have

Different matching operations are defined for standard products and for patches. Product matching is only used for installation but patch matching is also used to for copying.

In match_target newer versions of products currently installed are automatically selected

In autoselect_patches, a product being installed or copied will automatically have all applicable patches chosen

In patch_match_target, patches for the products already on the system or depot are selected

Dependency Selections

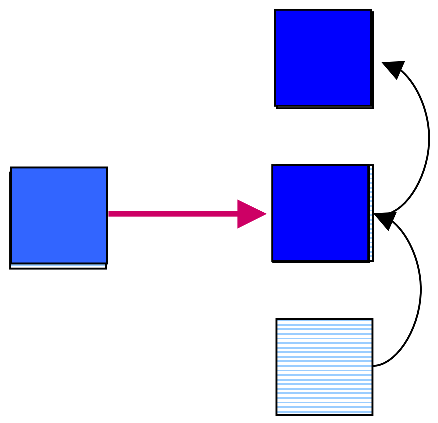
Implicitly give me what I need

As any SD object is selected, other objects listed as dependencies are automatically marked as well. For patches, SD automatically selects the newest available patch to resolve a dependency while recording the oldest patch that can fulfill the requisite.

Selection precedence

You (explicit) always wins

A dependency selection will promote to the newest available patch, but a later explicit selection can chose an older patch as long as the dependency is still met



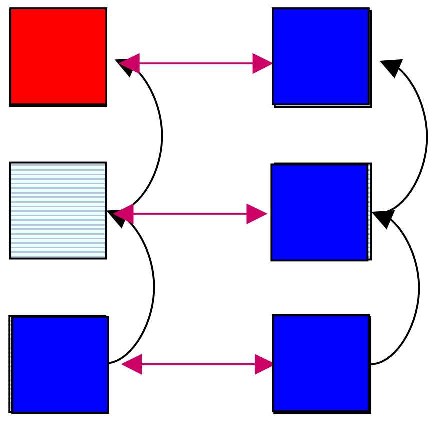
Mixing patch selections

Not always obvious what is going on

An explicit selection overrides an implicit selection. In this example an explicitly selected patch depends on another. This patch has been superseded twice in the depot, but the newest patch in the chain requires a patch that supersedes the original selection.

Moral: Beware mixing methods!

Note: Standard patch bundles avoid this problem!

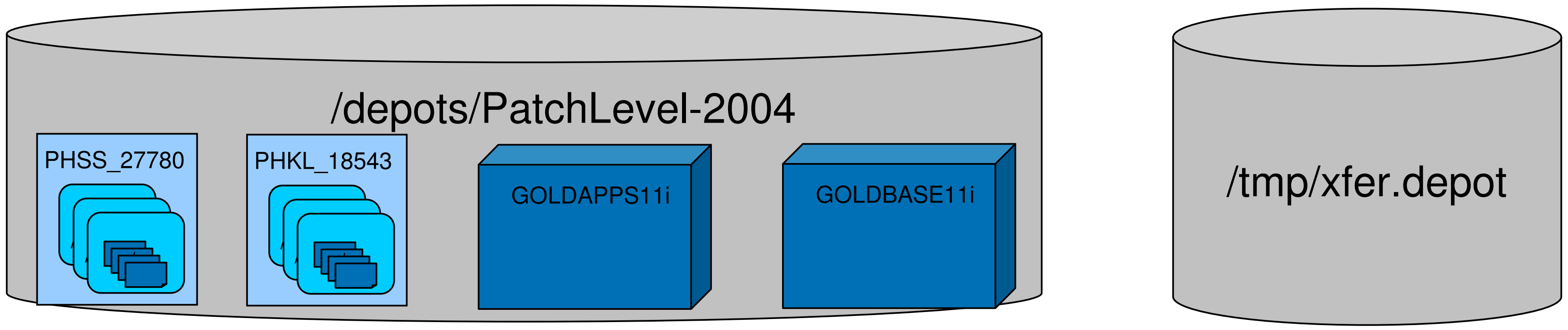


HP WORLD 2004
Solutions and Technology Conference & Expo

SD-UX Depots in Depth

Two depot formats to choose from!

Laid out for all to see or wrapped up in a ball



A directory (or network) depot is created as a directory hierarchy while a tape-style depot exists as a single file in tar(1) format.

Our swpackage example showed how a tape-style depot might be created. The target of a swcopy command will always be a directory depot.

Tape-style depots

Favorite for single patch distribution

```
# ls -l PHNE_28476*
-rw-r--r-- 1 root sys 2938 Mar 24 1:06 PHNE_28476.depot
```

As the name implies, a tape-style depot is built for serial access. While tape use today is rare, a single file format remains useful. In particular, HP-UX patches are available as single-product depots from the IT Resource Center (http://itrc.hp.com) via ftp(1).

The tar(1) or pax(1) commands provide a quick method to check for completeness after transfers:

```
# tar tvf /tmp/PHNE_28476.depot
```

Directory-style depots

A must for multi-system use

```
# ls -l PHNE_28476*
PHNE_28476:
total 16
  dr-x------ 8 root  sys     1024 Jul 12 17:36 PHNE_28476
  dr-x------ 4 root  sys     1024 Jul 12 17:36 catalog
  -rw-r--r-- 1 root  sys     6020 Jul 12 17:36 swagent.log
```

The same patch within a directory depot is split into parts. Each product will have a directory under the depot root with subdirectories for each fileset containing the delivered files. A similar hierarchy exists under the catalog directory for control scripts and IPD information.

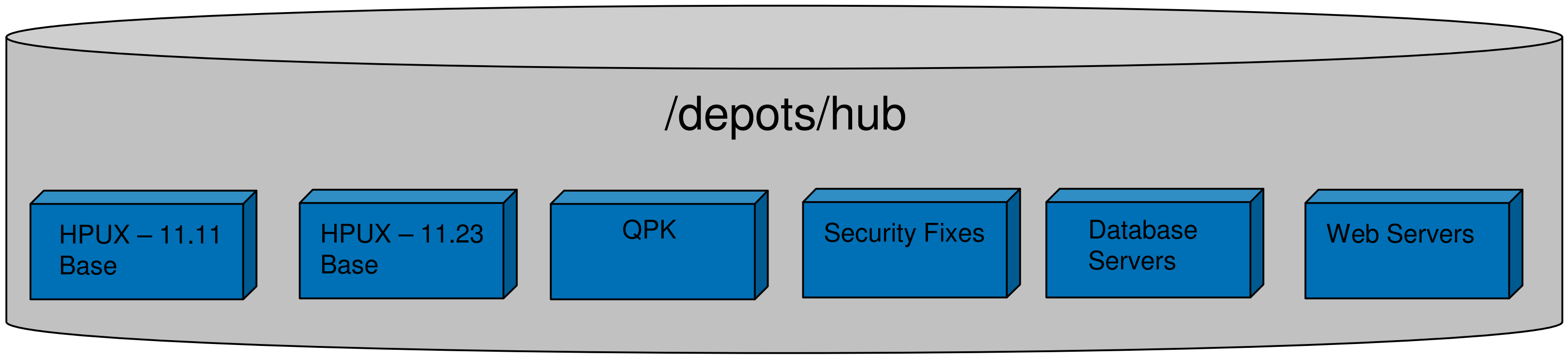
Building Depots

Starting thoughts

- Build for use with simple command lines
  - build for specific releases and revisions
  - normal use should be installation of full content
  - provide all dependencies within depot, always
- Isolate change/leverage testing
  - Stable depots allow the next system to look the same
  - Change is risk, limit impact when possible
- Manage control and access
  - Two managers of one depot leads to conflict
  - A junior admin should not be able to accidentally load a licensed application on his desktop

The Central Hub

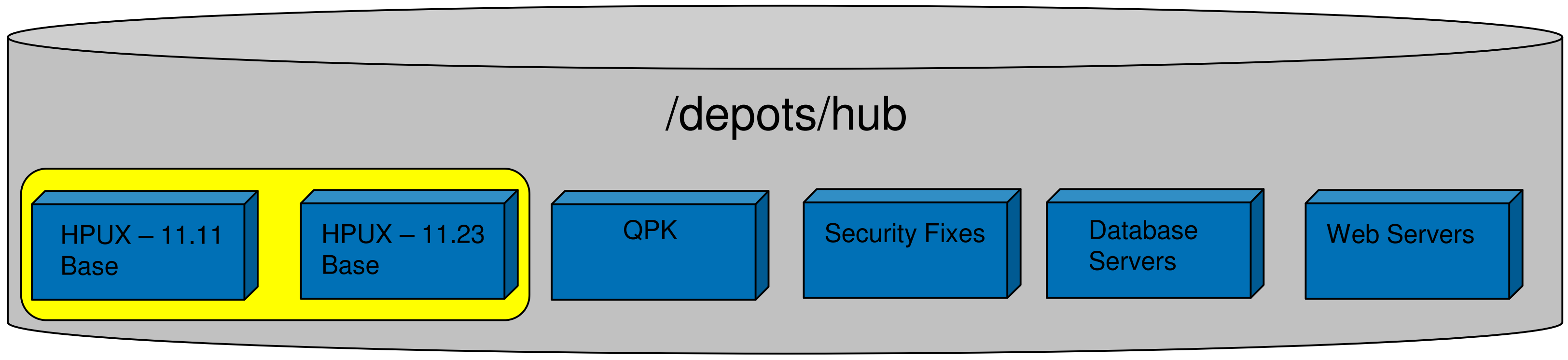
Everything *and* the kitchen sink!



While simple to build, throwing everything into a single depot can pose problems outside of a small & uniform shop. May be useful as an unregistered source of content for smaller depots.

The Central Hub

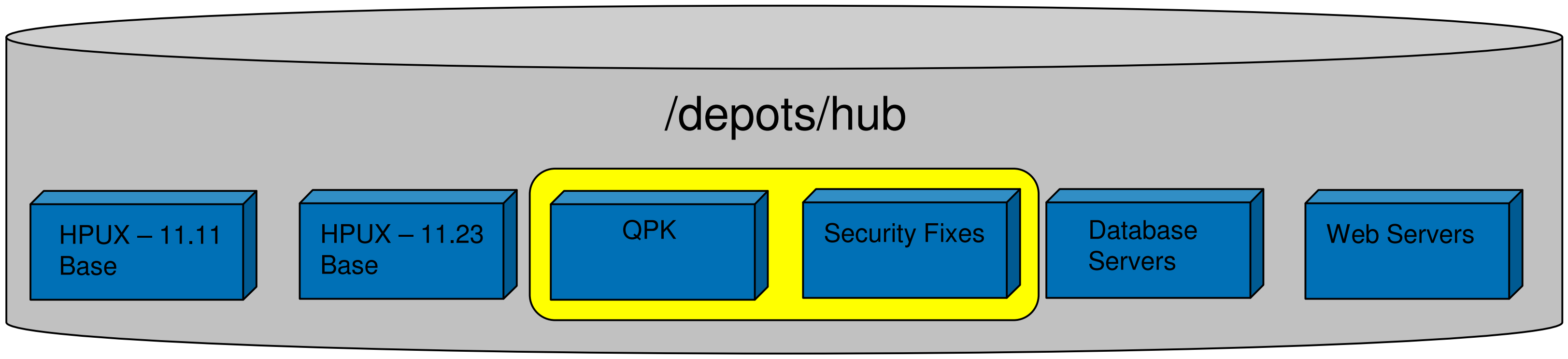
Specific issue #1



Two different HP-UX versions can cause unusual problems with one of the worst being an unintentional and partial OS upgrade

The Central Hub

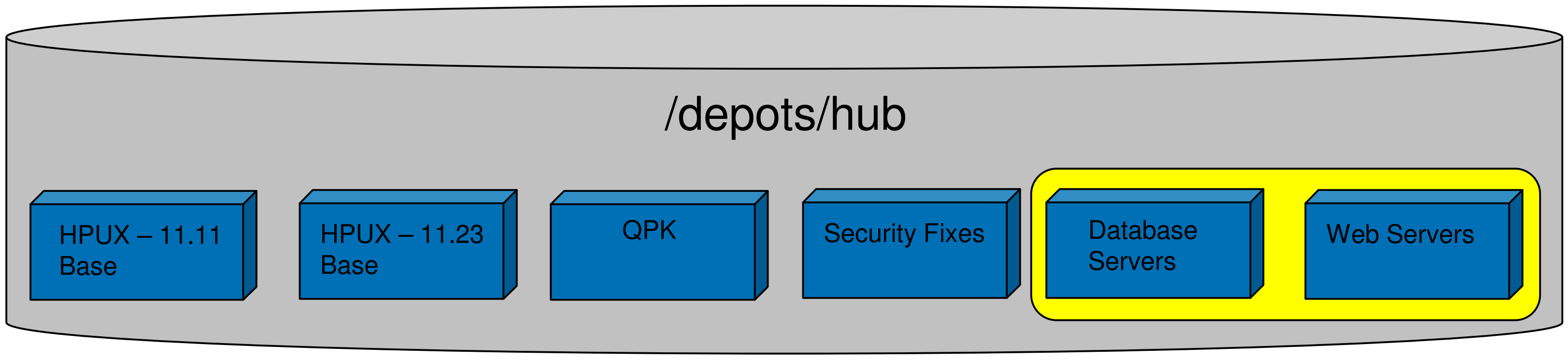
Specific issue #2



Newer patches that are vital to some systems may be a source of instability only on others. The security patches that are critical to an open subnet web server may be unwanted change on a private subnet database server.

The Central Hub

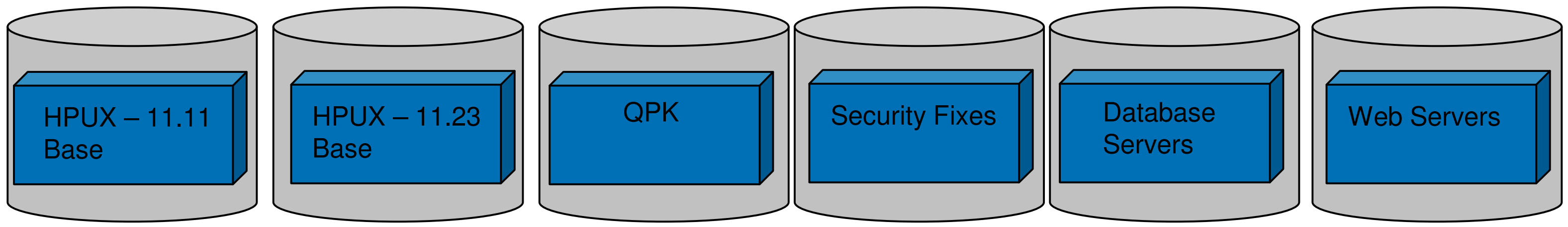
Specific issue #3



Licensed software should be easily accessible to licensed systems. Free availability leads to fun at audit time.

The purpose-driven depot

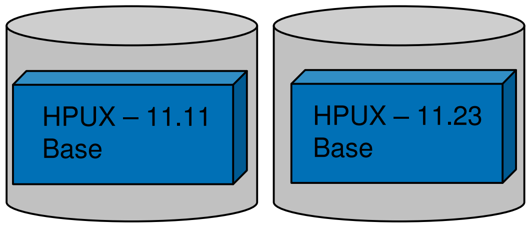
Divide and conquer



While smaller depots imply multiple swinstall sessions, they can still be selectively combined local to the target systems to save time.

The installation depot

A stable base to build upon



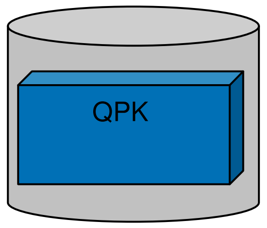
Contains everything required for a cold-install of any system:

- Base HP-UX (HPUXBaseOS/HPUXBaseAux)
- HP-UX Operating Environment (OE)
- All required hardware enablement (drivers, diags, patches)
- Baseline patches

When initially created will require a substantial testing expense. Any change in this depot must be of highest priority across the full organization

The proactive patch depot

An ounce of prevention



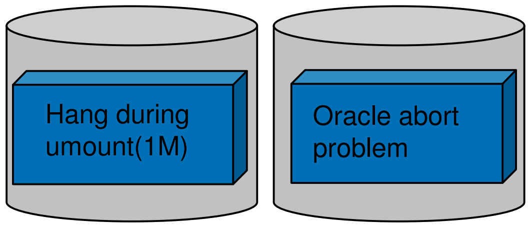
Fixing something that is not broken is a tricky business. The risk of change must be balanced with the incremental improvements contained. Likely contents are:

- Stable patch sets (HP Quality packs)
- High-value patches (Security fixes, warning resolution)
- Locally-proven reactive patches (critical to *you*)

A proactive patch depot could be used to update installation depots, but patch removal would lead to different conditions on older and newer systems.

The reactive patch depot

The medicine cabinet



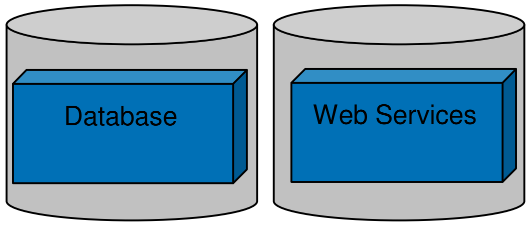
Contains fixes for specific problems. Created initially to install patches on a system experiencing a failure.

- Should contain minimum change to fix a problem
- May be proactively applied to select systems
- Rolled into baseline or proactive depots in indefinite future
- Very tolerant of brand-new (risky) patches

Reactive patches can be grouped into one or more depots if it is desired that all known fixes be loaded after any failure is encountered.

The application depot

Behind the castle walls



There are often distinct groups responsible for system software and applications. Both believe that they are the ones in charge, both should be able to believe it!

- Non-SD packaged applications may require specific fixes
- Change & risk is restricted to application clients
- Can be modified on their own schedules

If application-specific patches are maintained, it should be understood who will be responsible for tracking status (such as warnings and locally known reactive issues).

Tasks and Examples

“Sure, we could add that too...”

Every SD-UX engineer, ever…

HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

interex

shared knowledge • shared power

encompass

AN HP USER GROUP

RECOMMENDED TRAINING VENUE FOR THE

HP Certified Professional