introduction to hp-ux patch management

bob campbell
customer solutions operation

february 25, 2001

hp-ux 11i patch management

agenda

- a brief history of patch
- 11.x patch attributes
- swlist & patch state
- working with 10.x depots
- enforced patch dependencies
- fun with category tags
- known problems

a brief history of patch

disclaimer #1

the difference between a defect and a feature is dependant on which cubicle you are sitting in

a brief history of patch

disclaimer #2

the actual facts should never get in the way of an entertaining story

a brief history of patch

hp-ux 10.0 is released

- we say goodbye to Motorola-based systems
- DUX clusters are replaced by NFS diskless clusters
- software distributor is introduced
- the release shirts are passed out
- engineers ask a simple question…

the question

how do i patch my product?

software distributor & patching

software distributor…

- is designed to manage software revisions
- allows for software to be split into small partitions
- is designed to support multiple platforms

the hp patch process…

- is based upon supersession
- needs to manage multiple patch streams within a single ancestor
- has an infrastructure that predates hp-ux 10.0

the hp-ux 10.x patch

- not managed by SD-UX
- only identified by naming conventions
- requires 6 scripts per patch
- delivers content using a single fileset
- evolved during the 10.x releases

the hp-ux 10.x patch
(an object with issues)

- updates to ancestors can cause orphaned patches
- patches must be installed separately from products
- no superseded patches in depots
- the IPD was often inaccurate
- new files find interesting homes
- patch dependencies not enforced

the hp-ux 11.0 patch

- SD manages patches directly
- all information native to IPD
- scripts for customizations only
- patches and products play nice
- depot clutter not a problem
- new options and attributes galore!

patch attributes

data associated with an SD-UX object, providing information and controlling behavior

attributes can be displayed using the “`-a`” option of the `swlist` command

11.x patch attributes

- **`is_patch`**

  the attribute that defines an object as a patch

- **`is_sparse`**

  marks a patch fileset as incomplete without ancestor

- **`supersedes`**

  a list of all previous patch filesets replaced by current fileset

- **`superseded_by`**

  the specific fileset that was installed and caused fileset to be superseded

- **`applied_to`**

  ancestor fileset which patch was installed against

- **`applied_patches`**

  list of patch filesets that have modified product fileset

11.x patch attributes

- **`category_tag`**

  marking attributes that can be used in filtering selections
- **`readme`**

  the full contents of the patch original text file
- **`layout_version`**

  the revision of the external data specification
- **`data_model_revision`**

  the version of internal data structures
- **`patch_state`**

  the installation state of the patch fileset

the hp-ux 11.0 patch

(better, but not perfect)

- `swlist` did not change
- 10.x depots, 11.x servers
- still can't depend on dependencies
- new and interesting failure modes

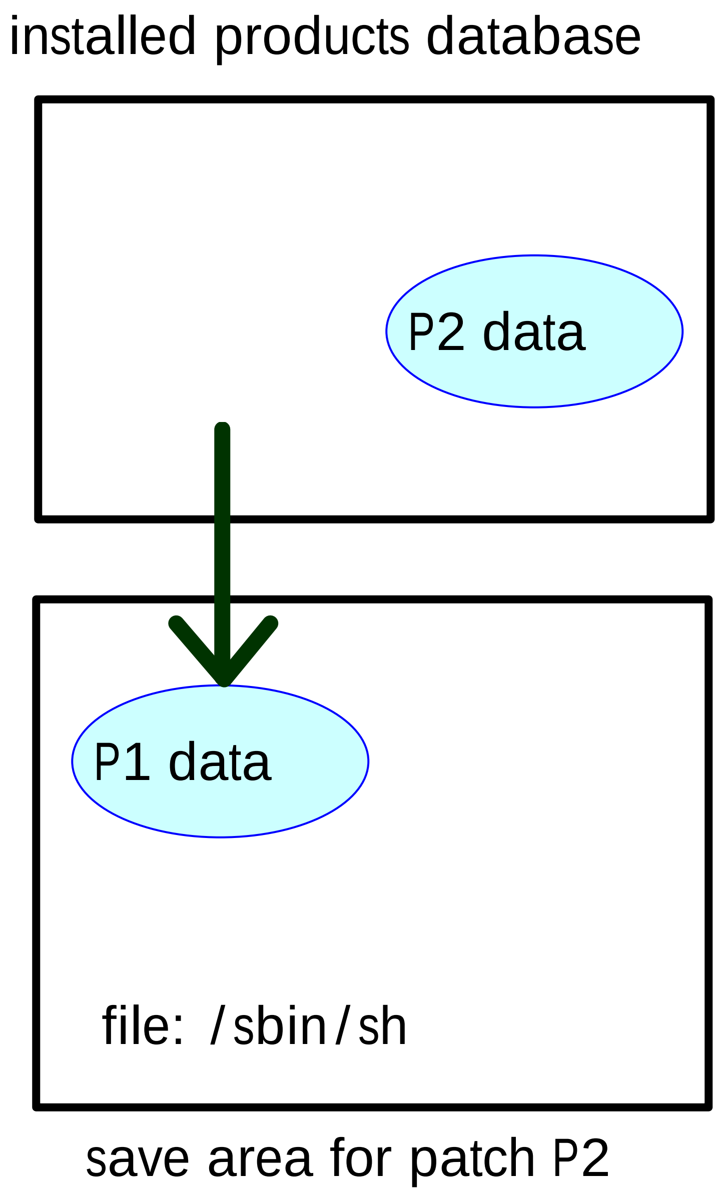
`swlist`

you thought that the `swlist` command was used to display the software currently on the system…..

almost.

`swlist` is actually a command used to list the contents of the Installed Products Database

an hp-ux 10.x patch managed the installed products database
(P2 supersedes P1)

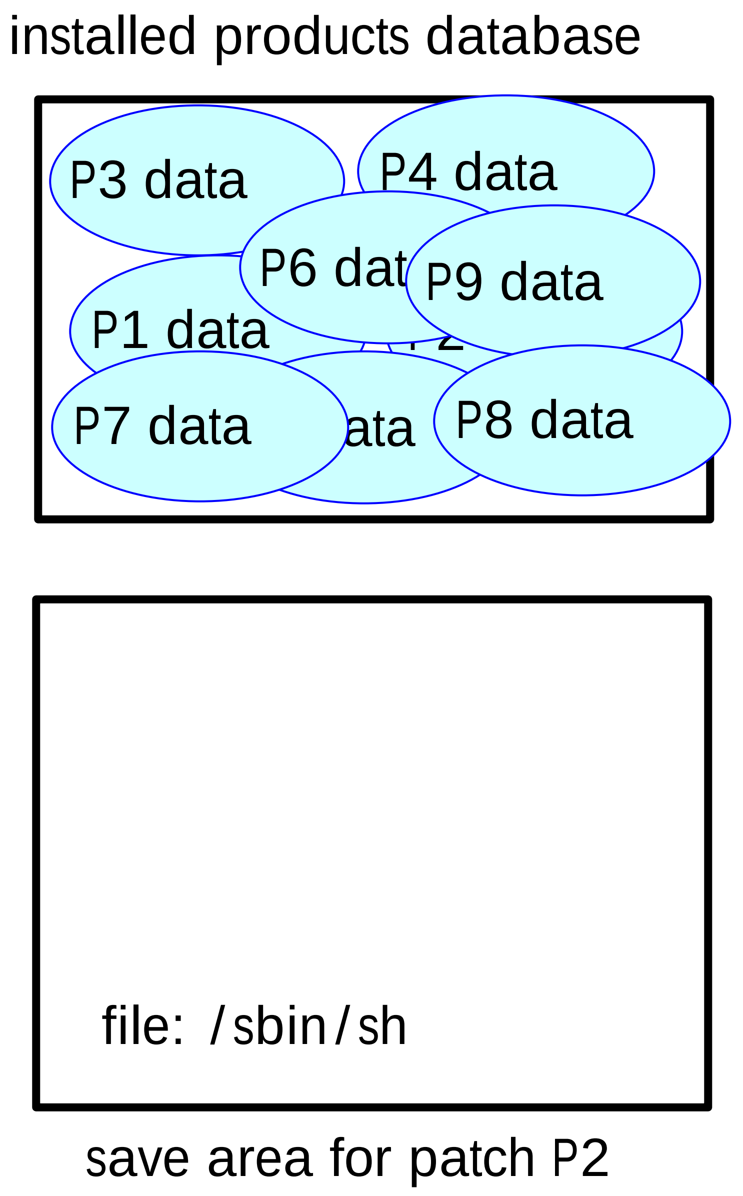


`swlist`

on hp-ux 10.x systems, the swlist command only listed the active patches because all superseded patch information had been removed from the IPD

SD had no knowledge of or access to information regarding superseded patches all hp-ux 11.x patch information is stored in the installed products database

(P2 supersedes P1)



swlist, 11.0 style

while swlist did not change, the IPD most certainly did. with 9 patches installed in each environment (P1 thru P9)

```
hp-ux 10.20:   # swlist -l product P\*

                  P9        1.0   English Dreamtime Patch

hp-ux 11.00:   # swlist -l product P\*

                  P1        1.0   English Dreamtime Patch
                  P2        1.0   English Dreamtime Patch
                  P3        1.0   English Dreamtime Patch
                  P4        1.0   English Dreamtime Patch
                  P5        1.0   English Dreamtime Patch
                  P6        1.0   English Dreamtime Patch
                  P7        1.0   English Dreamtime Patch
                  P8        1.0   English Dreamtime Patch
                  P9        1.0   English Dreamtime Patch
```

altered states

the question “*what is the condition of my patch*” can be answered within two different contexts:

- product context

  treated as a product, has the delivered software been properly installed and configured? This question is resolved by the `state` attribute.

- patch context

  treated as a patch, is the delivered software currently active on my system? This information is provided within the `patch_state` attribute.

state

configured: the software has been correctly installed and configured

installed: the software has been delivered, but the configure script has not run (many patches do not require configuration)

corrupt: SD-UX has detected an error and the software is in an unknown state

transient: the software is in an unknown state, but SD-UX has no idea how it got here

`patch_state`

active: software delivered in an active patch is currently in place and directly busable

superseded: the software within the patch was delivered to the system, but has been replaced by the content of a newer patch

committed: the save area associated with the fileset has been deleted and patch removal and rollback has been disabled

committed/superseded: the patch has been both committed and superseded (hp-ux 11.11 or hp-ux 11.0 with PHCO_22526)

`swlist`

OK, by using `patch_state` I can see what is active on my system, but isn't there a better way than using `grep`?

swlist

an option has been introduced to control the behavior of `swlist`

```
swlist -l product \
       -x show_superseded_patches=false *,c=patch
```

hp-ux `11.11` sets `show_superseded_patches` to false by default. `PHCO_22526` introduces the option to hp-ux `11.0`, but the default behavior remains unchanged.

with `PHCO_22526` installed, the default behavior can be changed by adding the line:

```
      swlist.show_superseded_patches=false
```

to the file `/var/adm/sw/defaults`

hp-ux 10.x depots
(a tale of two attributes)

data_model_revision: an internal data structure format definition (you can’t play with it, don’t bother trying)

- **2.10**

  model used by hp-ux 10.20
- **2.40**

  model used by hp-ux 11.0

layout_version: the externally visible data revision. currently two values are defined:

- **0.8**

  The original definition used by all hp-ux 10.x software
- **1.0**

  fully consistent with to the POSIX / IEEE 1387.2 standard, used by all hp-ux 11.x software

why POSIX?

- patch attributes
- category tags
- fileset architecture
- vendor defined attributes (VDAs)

the problem with **`layout_version`** (patch-22)

hp-ux 10.x SD-UX commands will complain about the mismatch for every object with an unsupported `data_model_revision`

unless explicitly told not to using the "`-x layout_version=0.8`" option, any hp-ux 11.0 SD-UX command will automatically convert depots to be POSIX compliant

the `swremove` command did not include a "`layout_version`" option

`layout_version`
(the good news)

- hp-ux 11.11 and hp-ux 11.0 systems running PHCO_22526 will no longer modify the existing depot configuration
- once clean depots are set up, they should remain clean during normal use

my depots are already a bit confused with respect to their actual `layout_revision`

`layout_version`

(the bad news)

what the 10.x clients will say

```
WARNING: Invalid value defined for the keyword "data_model_revision",
         at line 3.  The value "2.40" is not a supported data model
         revision.  This release of the Software Distributor supports
         the following values:

                 2.20 2.10 2.00

         The SD product needs to be updated using the instructions in
         the "Managing HP-UX With SD-UX" manual.  Continuing to read
         the file "/var/tmp/AAAa01165/catalog/INDEX" using the "2.20"
         data model semantics. New attributes associated with the
         "2.40" format will be ignored.  Unrecognized attributes may
         result in subsequent ERROR or WARNING messages.  If errors
         result from using the "2.20" semantics, you must update SD to
         a version that supports the "2.40" "data_model_revision".
WARNING: Ignoring unknown keyword "category" at line 7.
WARNING: Ignoring unknown keyword "category_tag" at line 82.
WARNING: Ignoring unknown keyword "directory" at line 84.
WARNING: Ignoring unknown keyword "is_locatable" at line 85.
```

this can be fixed using **`swmodify`**, right?

Wrong…

While swmodify can reset the value for `layout_version`, there is no such option for `data_model_revision`

the errors will still be displayed

will this put my 10.x systems at any risk?

Probably not…

all of the data_model_revision 0.8 information is still available. the 1.0 information generates messages, but should not have any impact on operations

that said, that is not how I recommend you run your shop…

swcopy to the rescue!

both the `layout_version` and `data_model_revision` can be corrected in a single operation using the following `swcopy` command:

```
   swcopy    -s /BAD_DEPOT
             -x layout_version=0.8
             -x enforce_dependencies=false
             -x compress_files=true
                   \* @ /GOOD_DEPOT
```

all of the contents of *BAD_DEPOT* are now converted to `layout_version` 0.8 as well as `data_model_revision` 2.10 and safely stored in *GOOD_DEPOT*

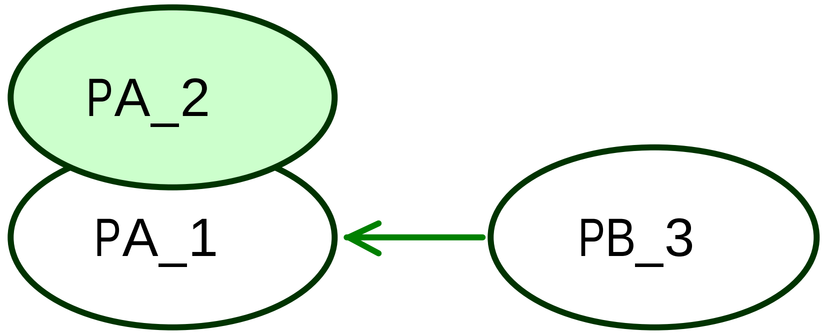
enforced patch dependencies

the patch PB_3 has a dependency on content first delivered by PA_1

patch PA_1 has been superseded by the newer patch PA_2

cumulative patching requires PA_2 to meet any requirements of PB_3

SD-UX has never understood the ability of a superseding patch to fulfill a dependency, until now!



SD requisites
(3 types defined)

requisites are stored in an attribute with a label that matches the type of the requisite

- corequisites

  dependency must be present, but load order is not important

- prerequisites

  dependency must load before requisite holder, even in common swinstall session

- exrequisites (not implemented)

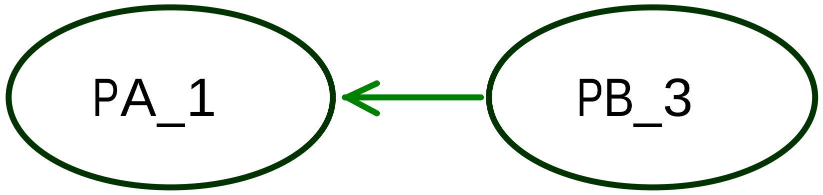
  target and requisite holder are mutually exclusive, only one may be present on system or selected for installation

SD requisites

- usually enforced between two filesets
- are not architecturally filtered
- are not generally supported before hp-ux `11.11`
- starting in hp-ux `11.11`, all dependencies without requisites must be marked using the `manual_dependencies` category tag

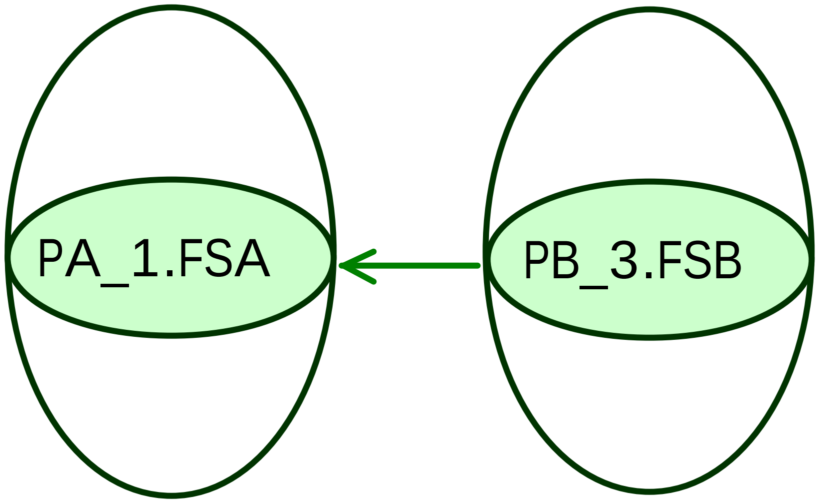
enforced patch dependencies

the standard view of patch dependencies are of a patch to patch relationship



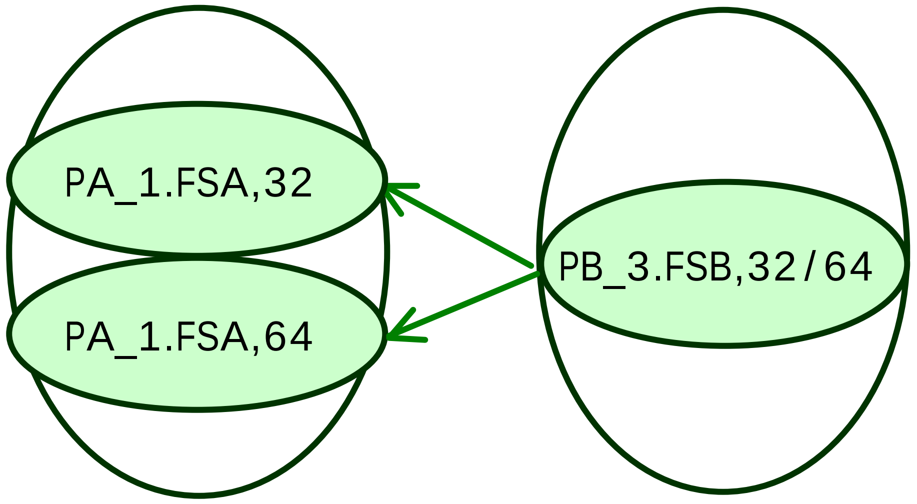
enforced patch dependencies

in reality, SD-UX generally manages requisites between two filesets



enforced patch dependencies

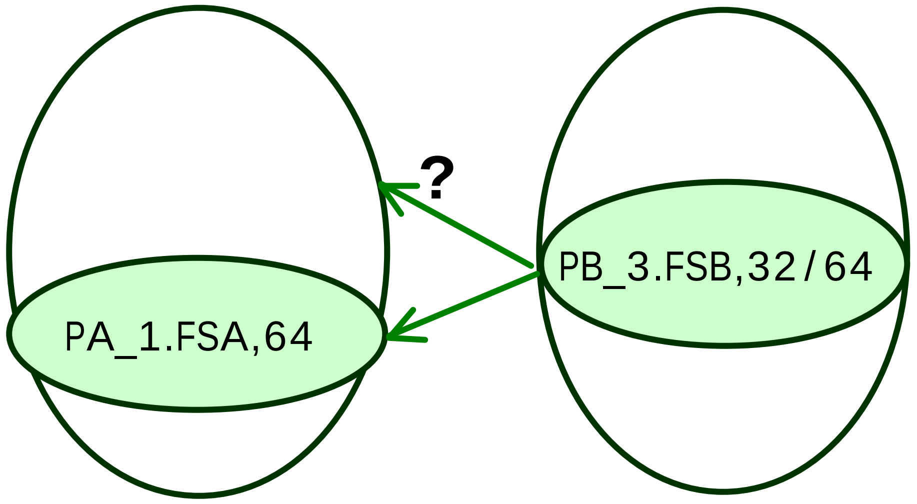
and one fileset can exist for more than one architecture, causing the requisite to be in the form A or B



enforced patch dependencies

a few cases exist where a requisite cannot be architecturally balanced.

as requisites cannot be filtered architecturally, this poses a problem



dummy requisite specifications

when the architecture of the requisite is a subset of the requisite holder, the base fileset `OS-Core.CORE2-KRN` will be used to fill the gaps

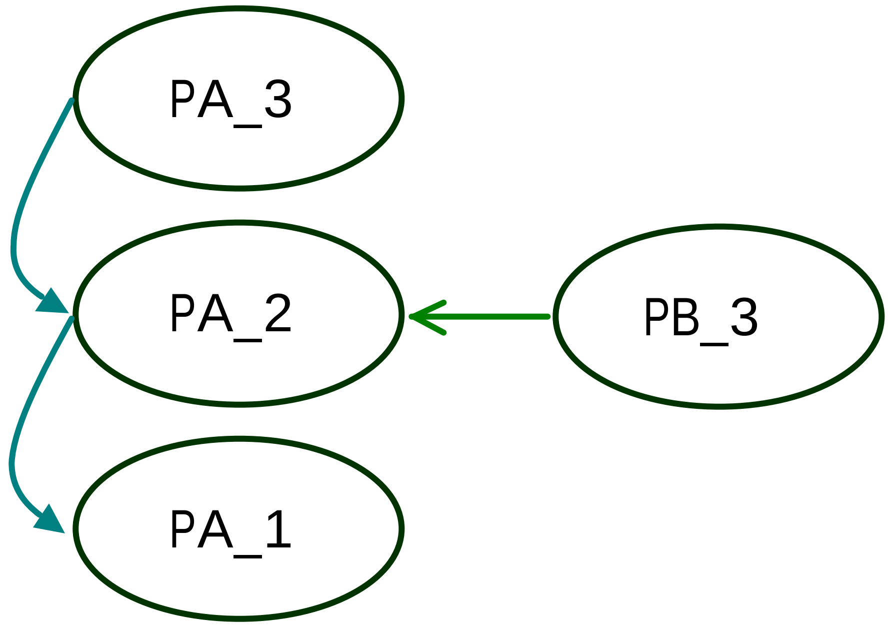
```
corequisite
   PA_1,fa=64 |
   OS-Core.CORE2-KRN,fa=32
```

(abridged for our benefit)

life in the depot

if a depot has several patches from a single patch supersession chain, dependencies will automatically be promoted to the top of the chain

another patch may be explicitly selected, but must meet the needs of the requisite



category tags

- descriptive labels that can be used to mark bundles, products, and filesets
- default values applied to all patch products and filesets
- tags can be added and deleted with the swmodify command
- trust me, on hp-ux 11.0 systems, load PHCO_22526 before you create your own category tags

category tags
(so what?)

- can be used as a filter within any software specification
- can be used to filter patches during matching operations

NOTE: using patch filters outside of the `swlist` command is not recommended unless patch dependencies are enforced or manually verified!

predefined patch category tags

- patch
- general_release / special_release
- critical
  - panic
  - halts_system
  - memory_leak
  - corruption
- defect_repair / enhancement
- hardware_enablement
- manual_dependencies

category tag examples

list all patches on system (regardless of naming conventions!)

```
    swlist -l product *,c=patch
```

install all patches in depot that match my system and have been marked with the tag `myApp` (other patches may be included to resolve dependencies)

```
    swinstall -s MyServer:/MyDepot
            -x patch_match_target=true
            -x patch_filter=*,c=myApp
```

mark all `PHKL_*` patches in depot with the tag `kernel`

```
    swmodify -d -a category_tag=kernel \
                PHKL_\* @ /MyDepot
```

list all category types in depot

```
    swlist -d -l category @ MyServer:/MyDepot
```

known problems

As this slide set is being created, only the following issues are of special note:

- split patches
- orphaned patches

split patches

- 11.x patches often consist of several different filesets
- some ancestor filesets may be installed or upgraded after the patch was initially installed
- the condition of having a patch only partially installed is known as a **split patch**
- while a subsequent installation will load only the missing fileset, you must notice that it is missing!

split patches
(the good news)

- patch dependencies will not result in a split patch condition
- an option has been added to allow split patches:

  `allow_split_patches=true`

  the first customer to determine any valid reason for using this option will win an HP Ft Collins hat or shirt!

orphaned patches

- if an hp-ux 11.x patch is loaded without an ancestor, it is an **orphaned patch**
- with native SD patch support, this condition is a sign of a confused IPD
- as of this time, only one way to cause this condition is known

the way

1. start with a depot that has a new version of a product as well as a patch for an older version
2. find a system that has the older version of the product, but does not have the patch
3. install both the new product and the patch at the same time

   step 3 could happen if both `match_target` and `patch_match_target` are selected