A|B Update

Broadcom Proprietary and Confidential

Release Note: This document can be shared with customers.

abu - a|b system update integration

Terminology: "legacy boot" - this term refers to the boot process in place prior to the "a|b system" support.

Applicability: the feature applies by default to all armv8 (onward) designs (aka. non legacy designs), including in particular: 7268, 7271, 72604, 7278.

The feature is integrated in android-n (16.4 release) onward.

Note: the feature could be backported to 'avko' (and by extension armv7 legacy designs), however it requires a foul proof path for automatic upgrade through OTA from legacy OTA based mode to a|b update mode and while such mechanism could be put in place, at the present time, there is no plan to do so.

CONTENTS

Key Points

- a|b processing is applicable to "boot" and "system" partitions content.
    - the primary partition is labelled (suffixed) "**_i**"
    - the secondary partition is labelled (suffixed) "**_e**"
    - at beginning of time, when nothing is yet registered locally on the device regarding partition setup, the "_i" partition is used as primary.
        - this is also a way to 'reset' the processing of the a|b mode by simply invalidating the "eio" partition content on the device, as example by pushing a bogus content which would invalidate the magic hash causing the bootloader to reset the partition content.
- The root file system content from ramdisk.img is now folded into the system.img which is the default root file system.
    - during normal boot, the bootloader would skip init'ing ramdisk to kernel and instead let the device-mapper map the system.img root file system.
    - **consequently, if you are making changes which target ramdisk content (such as init scripts, "init" binary, selinux policy, etc...), you will now need to flash the system.img partition instead of the boot.img partition to apply those changes.**
- For recovery mode, the recovery uses the ramdisk from the boot.img, so there is no recovery image per say.
    - the ramdisk from the boot.img is effectively the ramdisk-recovery.img in this case, it is packaged during the build.
    - in simple terms, the boot.img in the a|b update model is effectively the recovery.img of the legacy model.
    - the recovery mode in a|b model is very limited usage, in particular it is mostly used for factory reset or reboot into bootloader. there is no other usage defined specifically by android, however a specific device implementation may choose to enhance this mode with other functionalities, this is out of scope of this document.
- Compared to the legacy partition layout, the main changes to the partition table for a|b update are:
    - removing the recovery partition, no longer needed since the new boot.img contains the (limited) recovery support.
    - reducing the cache partition:
        - no longer needed for OTA update support.
        - required for certification support due to some certification tests expecting to find a "/cache".
    - legacy (recovery,boot) partitions are now the boot redundant pair boot_i|boot_e.
    - legacy (cache,system) partitions have been concatenated and then split half way to become the system redundant pair system_i|system_e.
    - ... because of this scheme, it should be possible to move a legacy system to an a|b system without needing to wipe out the overall emmc content and in particular be able to preserve userdata content seamlessly.

- **In order to fit within the restrictions of a 8GB emmc footprint design (proofed design), the redundant system_[ie] partitions are mandated to move to squashfs file system instead of ext4; however should the target device allow larger flash footprint to be available, it is recommended to continue using ext4 instead (better performance and smaller ota possible).**
- **Starting with Android-O**, a new set of redundant partition is made available: vendor.
    - the vendor_i|vendor_e partitions are carved out of the existing system_i|system_e partitions which are reduced in size consequently.

Integration Overview

To support the a|b update system, one use a partition to pass information between bootloader and main application (through the "boot_control" HAL).

This partition carries the information about the number of alternate slots, whether the slots can be booted, etc...

In our integration, the partition is called "**eio**"(TM), and is used as follows:

- Android BSU boots up and detects presence of "eio" partition.
    - if not present, fallback to legacy boot.
- Read content from "eio"; if empty, create an initial set of data:
    - slot index 0 is always suffixed "_i"; this is the default partition that will be invoked if there is no other indication in the "eio" partition that an alternate should be used instead.
    - slot index 1 is always suffixed "_e"
        - verify presence of two suffixed partitions for "boot" and "system".
        - additionally we could check presence of valid boot.img header for the boot partition to populate some of the "eio" data, this is not done at the present time.
    - failure to determine that the partition is properly setup results in abort of the "eio" processing.
    - fallback to legacy boot attempt
        - note: there is actually no guarantee the device would boot, in most cases it would just end up failing to launch anything beyond BSU boot step.
- Select the slot to boot.
    - update the boot command line with the information relative to a|b boot.
- Launch selected slot.

Bootloader Environment Configuration

In order to properly make use of this mechanism, **we require that the device configures the FB_DEVICE_TYPE as a bootloader variable**, this is to ensure that the commander block can be

located, and further processing can be derived form its content, in most devices with a single emmc such as 7268|7271|avko, setting this on BOLT command line should do:

```
setenv -p FB_DEVICE_TYPE flash0
```

Failure to do so will lead the bootloader to revert to legacy system assumption (because the "eio" commander cannot be found), which in turns may lead to undefined behavior as we cannot guarantee the content of the boot and system partitions are adequate nor can we assume the system is fully setup for legacy vs. a|b-boot, so typically a complete failure to boot will arise from such, though you may also end up in a recovery mode depending on whether you have setup other bootloader variables.

For the same purpose of booting properly in the correct mode, it is recommended, but not necessary, to remove **the ANDROID_BOOT_IMG and ANDROID_RECOVERY_IMG bootloader variables:**

```
unsetenv ANDROID_BOOT_IMG
unsetenv ANDROID_RECOVERY_IMG
```

Normal Boot

Unless you have enabled "verified boot" and the boot is actually successful (see "avb" information), we do not enter verified boot state even though the expectations are that the device mapper android extension runs a verity path.

- it is possible to bypass the android verity device mapper enforcement of dm-verity of the system.img by pretending to be an engineering image (regardless of the actual android image mode that was built in), this particular mode is achieved by passing additional kernel command line information targeting the android device mapper extension and it is the de-facto default mode of operations we use.

Upon successful mount of the system.img rootfs by the kernel device mapper, its /init is launched and the device boots android as traditionally expected.

Recovery Boot

When booting in recovery mode, the ramdisk from the boot.img is being loaded and boot continues from there in recovery mode as usual in such mode of operations.

New Fastboot Command

A new set of fastboot based command is available to work with the a|b update scheme. Each commend which requires a <slot-suffix> input would expect this value to be either "i" or "e", any other suffixes would be rejected.

Commands are:

- set_active:<slot-suffix>
  - force the slot to be booted to be the one referenced; this also reset the boot failure state of such slot if it was failed to boot before.
- has-slot:<partition>
  - queries whether the <partition> is an a|b update one, meaning effectively is a slot based one.
- current-slot
  - returns the active slot that will be booted.
- slot-suffixes
  - will always return "i" and "e", the slot suffixes known to this integration.
- slot-successful:<slot-suffix>
  - queries whether the slot was booted successfully, i.e. has been marked as successful, which means it was booted once at least up to the update_verifier (part of the system image support).
- slot-unbootable:<slot-suffix>
  - queries whether the slot was marked as non-bootable.
- slot-retry-count:<slot-suffix>
  - queries the number of times the slot will be attempted to boot successfully before marking it as non bootable, this number will decrease as the slow fails to boot repeatedly.

Flashing Partitions with Fastboot

There is virtually no change required. The default fastboot behavior is to detect which slot needs to be flashed on a a|b system by querying which is the active slot on the device. As such, the same commands used to flash a legacy device can be used to flash the active partitions of an a|b update device.

However, it is also possible to use the new [--slot] option. This option takes the slot name to be used for flashing, keyword 'all' can also be used, but it would cause the flashing to take twice the time since each pair of primary|secondary partitions would be flashed. The slot value used for the [--slot] option is the device slot suffix.

Below are examples which illustrates how you would use the slot option to flash the equivalent of the primary boot.img in legacy mode and in the new a|b mode,

```
legacy or slot detection:                    fastboot -u flash boot
boot.img
```

```
a|b update (slot specific):                    fastboot -u --slot e flash
boot boot.img

a|b update (forcing a slot old'skool):         fastboot -u flash boot_i
boot.img
```

Help - My Device Does Not Boot

You may occasionally be in a state where the device refuses to boot; this would happen especially when there are no valid partition to boot from detected (marked as such).

A few classic scenario leading to such situation would be:

- you have failed to fully boot an image to android: say boot to kernel alone or crash booting android early.
- you have upgraded a device from pre-O bootloader to a O bootloader (the commander block has been changed).

So what to do in this case... typically the device would boot in fastboot mode by default when such situation happens, but if you are using fastboot over tcp, the device may just be stuck in BOLT. regardless, make sure you boot into bootloader mode.

Next, issue the following command:

```
fastboot set_active a
```

This command will mark the first partition|slot (thus _i) to be attempted to boot in the next reboot cycle.

Help - I Am Not Flashing to the Expected Partition

Similarly to the above case where the device does not know what partition to boot from because none is found valid, you may also find that you are not flashing to the expected slot because the slot has changed.

A simple way to always come back to a known state is to issue the command:

```
fastboot set_active a
```

This will reset the commander block to use the first slot (thus _i) for all further operations, including flashing and booting.

Applying OTA for a|b update

Generating the OTA Package

The same mechanism used for legacy ota packages can apply to the a|b update mode; in particular, the ota package (full or incremental) can be generated using the following commands:

```
1) make -j48 PRODUCT-<my-device>-userdebug showcommands dist DIST_DIR=<dist
repo>
2) ./build/tools/releasetools/ota_from_target_files dist/<my-device>-
target_files-<type>.${USER}.zip full-ota.zip

or, if using a BASELINE for increment:

2) ./build/tools/releasetools/ota_from_target_files -i BASELINE-<my-device>-
target_files-<type>.${USER}.zip dist/<my-device>-target_files-<type>.${USER}.zip
incremental-ota.zip
```

The size of an ota package may vary based on the content and file system mode used.

In particular, when using squashfs instead of ext4, it is not possible to generate a block based ota package since the file system is compressed; therefore the ota package size overall will be larger than the one seen in a pure ext4 based incremental block ota under such configuration.

It is left up to the discretion of the device implementation to decide what best suits the need depending on the hardware configuration.

- on android N for the reference devices implementing a|b updates, squashfs mode is used, therefore limiting the ability to produce 'small sized' ota.
- on android O (onward) the reference devices implementing a|b updates are now using ext4 mode, therefore block based ota can be used, producing smaller packages.

Setting Up Server

You need a http server in order to host the image to run through the update engine client support. On ubuntu (typical android development environment), you may install the apache2 web server.

```
sudo apt-get install apache2
```

Once installed, the default location available to drop ota images would be: "**/var/www/html**"

Applying Update

Assuming you have a http server on your host machine, take the generated ota and unzip it on the host server. This should in particular produce a couple of packages:

```
-rw-rw-r-- 1 pierre pierre 466836562 Jan  1  2009 payload.bin
-rw-rw-r-- 1 pierre pierre       154 Jan  1  2009 payload_properties.txt
```

The payload.bin is the actual payload information to update; payload_properties.txt is the metadata header content associated with the payload.

Now on the device running the a|b mode, simply invoke the update engine client tool:

```
update_engine_client --payload=http://<server-ip>/payload.bin --update --
headers="FILE_HASH=Rbneb79V1QpAUmfm19jlQRhB0NeMVWOFz/Q6lj49ygY=
FILE_SIZE=466836562
METADATA_HASH=2UsI2+YnFXvRmtKvzbpQ/u1LlLPzX56bVKD/uQu4VOI=
METADATA_SIZE=21294
"
```

Where the [--headers] option content references the exact content of the payload_properties.txt.

Note: the [--headers] parameter needs to have the proper syntax as seen in the example above with newline between parameters <type>=<value> pair, or the client would abort due to inability to read the proper set.

The update client tool will apply the payload.bin update into the non active slot; upon successful completion of this step, a manual reboot is necessary to trigger the boot commander actions to boot into the newly flashed partition.

You can follow progress update of the operation on the target device using those logcat:

```
logcat -s bcm-bootc:v update_engine:v
```

Example: Running update_engine_client

For reference, here is an example log you may see when applying an update using the update_engine_client on the device.

If things go well, the last log seen at the end of the process (as highlighted) should be the update engine asking for system reboot in order for the update to apply; that is the device to reboot in the newly assigned 'current slot' which would be the opposite slot from the active one at the time of the update run.

```
03-22 07:51:58.943  2541  2541 I bcm-bootc: getCurrentSlot(0)
03-22 07:51:58.943  2541  2541 I update_engine:
[0322/075158:INFO:update_attempter_android.cc(199)] Using this install plan:
03-22 07:51:58.944  2541  2541 I update_engine:
[0322/075158:INFO:install_plan.cc(71)] InstallPlan: new_update, payload type:
unknown, source_slot: A, target_slot: B, url:
http://10.136.18.101/cypressd/payload.bin, payload size: 466834493, payload
```

```
hash: EhmgWWQceppAMwKMnipv/b22mHnp5hrvEycTvQC12Bg=, metadata size: 21294,
metadata signature: , hash_checks_mandatory: true, powerwash_required: false
03-22 07:51:58.944  2541  2541 W update_engine:
[0322/075158:WARNING:hardware_android.cc(126)] STUB: Assuming OOBE is complete.
03-22 07:51:58.944  2541  2541 I update_engine:
[0322/075158:INFO:cpu_limiter.cc(71)] Setting cgroup cpu shares to  2
03-22 07:51:58.944  2541  2541 E update_engine:
[0322/075158:ERROR:utils.cc(199)] 0 == writer.Open(path, O_WRONLY | O_CREAT |
O_TRUNC, 0600) failed: No such file or directory
03-22 07:51:58.944  2541  2541 E update_engine:
[0322/075158:ERROR:cpu_limiter.cc(74)] Failed to change cgroup cpu shares to 2
using /sys/fs/cgroup/cpu/update-engine/cpu.shares
03-22 07:51:58.944  2541  2541 I update_engine:
[0322/075158:INFO:update_attempter_android.cc(379)] Marking booted slot as good.
03-22 07:51:58.944  2541  2541 I bcm-bootc: markBootSuccessful(0)
03-22 07:51:58.947  2541  2541 I update_engine:
[0322/075158:INFO:update_attempter_android.cc(394)] Scheduling an action
processor start.
03-22 07:51:58.947  2541  2541 I update_engine:
[0322/075158:INFO:action_processor.cc(46)] ActionProcessor: starting
InstallPlanAction
03-22 07:51:58.947  2541  2541 I update_engine:
[0322/075158:INFO:action_processor.cc(116)] ActionProcessor: finished
InstallPlanAction with code ErrorCode::kSuccess
03-22 07:51:58.947  2541  2541 I update_engine:
[0322/075158:INFO:action_processor.cc(143)] ActionProcessor: starting
DownloadAction
03-22 07:51:58.947  2541  2541 I update_engine:
[0322/075158:INFO:install_plan.cc(71)] InstallPlan: new_update, payload type:
unknown, source_slot: A, target_slot: B, url:
http://10.136.18.101/cypressd/payload.bin, payload size: 466834493, payload
hash: EhmgWWQceppAMwKMnipv/b22mHnp5hrvEycTvQC12Bg=, metadata size: 21294,
metadata signature: , hash_checks_mandatory: true, powerwash_required: false
03-22 07:51:58.947  2541  2541 I update_engine:
[0322/075158:INFO:download_action.cc(178)] Marking new slot as unbootable
03-22 07:51:58.947  2541  2541 I bcm-bootc: setSlotAsUnbootable(1,current:0)
03-22 07:51:58.949  2541  2541 I update_engine:
[0322/075158:INFO:multi_range_http_fetcher.cc(45)] starting first transfer
03-22 07:51:58.949  2541  2541 I update_engine:
[0322/075158:INFO:multi_range_http_fetcher.cc(73)] starting transfer of range
0+466834493
03-22 07:51:58.949  2541  2541 I update_engine:
[0322/075158:INFO:libcurl_http_fetcher.cc(94)] Starting/Resuming transfer
03-22 07:51:58.958  2541  2541 I update_engine:
[0322/075158:INFO:libcurl_http_fetcher.cc(106)] Using proxy: no
03-22 07:51:58.958  2541  2541 I update_engine:
[0322/075158:INFO:libcurl_http_fetcher.cc(237)] Setting up curl options for HTTP
03-22 07:51:58.960  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(196)] Completed 0/? operations,
1136/466834493 bytes downloaded (0%), overall progress 0%
03-22 07:51:58.962  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(536)] Manifest size in payload matches
expected value from Omaha
03-22 07:51:58.962  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(1396)] Verifying metadata hash signature
using public key: /etc/update_engine/update-payload-key.pub.pem
03-22 07:51:58.962  2541  2541 I update_engine:
[0322/075158:INFO:payload_verifier.cc(93)] signature blob size = 264
03-22 07:51:58.963  2541  2541 I update_engine:
[0322/075158:INFO:payload_verifier.cc(112)] Verified correct signature 1 out of
1 signatures.
03-22 07:51:58.963  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(1439)] Metadata hash signature matches
value in Omaha response.
03-22 07:51:58.965  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(1459)] Detected a 'full' payload.
03-22 07:51:58.966  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(374)] PartitionInfo old boot sha256:  size:
0
```

```
03-22 07:51:58.966  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(374)] PartitionInfo new boot sha256:
jTasb96DQoY9u1KjnKLfEzJCLuOZJ+AE8e8tIXKGl5k= size: 21073920
03-22 07:51:58.966  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(374)] PartitionInfo old system sha256:
 size: 0
03-22 07:51:58.966  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(374)] PartitionInfo new system sha256:
7WrR01G8iz3x3NEbizeBj9MhdmEZe7cs1ChJO/gZlpw= size: 769654784
03-22 07:51:58.976  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(359)] Applying 11 operations to partition
"boot"
03-22 07:51:58.986  2541  2541 I update_engine:
[0322/075158:INFO:delta_performer.cc(647)] Starting to apply update payload
operations
cypressd:/ # 03-22 07:52:01.825  2541  2541 I update_engine:
[0322/075201:INFO:delta_performer.cc(359)] Applying 367 operations to partition
"system"
03-22 07:52:09.272  2541  2541 I update_engine:
[0322/075209:INFO:delta_performer.cc(196)] Completed 31/378 operations (8%),
56035960/466834493 bytes downloaded (12%), overall progress 10%
03-22 07:52:19.798  2541  2541 I update_engine:
[0322/075219:INFO:delta_performer.cc(196)] Completed 61/378 operations (16%),
112041584/466834493 bytes downloaded (24%), overall progress 20%
03-22 07:52:34.251  2541  2541 I update_engine:
[0322/075234:INFO:delta_performer.cc(196)] Completed 95/378 operations (25%),
168067200/466834493 bytes downloaded (36%), overall progress 30%
03-22 07:52:47.195  2541  2541 I update_engine:
[0322/075247:INFO:delta_performer.cc(196)] Completed 129/378 operations (34%),
217862280/466834493 bytes downloaded (46%), overall progress 40%
03-22 07:52:59.906  2541  2541 I update_engine:
[0322/075259:INFO:delta_performer.cc(196)] Completed 162/378 operations (42%),
270776936/466834493 bytes downloaded (58%), overall progress 50%
03-22 07:53:13.086  2541  2541 I update_engine:
[0322/075313:INFO:delta_performer.cc(196)] Completed 197/378 operations (52%),
325145192/466834493 bytes downloaded (69%), overall progress 60%
03-22 07:53:22.171  2541  2541 I update_engine:
[0322/075322:INFO:delta_performer.cc(196)] Completed 227/378 operations (60%),
380689608/466834493 bytes downloaded (81%), overall progress 70%
03-22 07:53:33.744  2541  2541 I update_engine:
[0322/075333:INFO:delta_performer.cc(196)] Completed 258/378 operations (68%),
434390992/466834493 bytes downloaded (93%), overall progress 80%
03-22 07:53:45.392  2541  2541 I update_engine:
[0322/075345:INFO:delta_performer.cc(196)] Completed 318/378 operations (84%),
456460240/466834493 bytes downloaded (97%), overall progress 90%
03-22 07:53:52.057  2541  2541 I update_engine:
[0322/075352:INFO:delta_performer.cc(196)] Completed 378/378 operations (100%),
466834493/466834493 bytes downloaded (100%), overall progress 100%
03-22 07:53:52.058  2541  2541 I update_engine:
[0322/075352:INFO:delta_performer.cc(1336)] Extracted signature data of size 264
at 466812671
03-22 07:53:52.061  2541  2541 I update_engine:
[0322/075352:INFO:multi_range_http_fetcher.cc(111)] terminating transfer
03-22 07:53:52.062  2541  2541 I update_engine:
[0322/075352:INFO:multi_range_http_fetcher.cc(171)] Received transfer
terminated.
03-22 07:53:52.062  2541  2541 I update_engine:
[0322/075352:INFO:multi_range_http_fetcher.cc(123)] TransferEnded w/ code 206
03-22 07:53:52.062  2541  2541 I update_engine:
[0322/075352:INFO:multi_range_http_fetcher.cc(157)] Done w/ all transfers
03-22 07:53:52.194  2541  2541 I update_engine:
[0322/075352:INFO:delta_performer.cc(1596)] Verifying payload using public key:
/etc/update_engine/update-payload-key.pub.pem
03-22 07:53:52.194  2541  2541 I update_engine:
[0322/075352:INFO:payload_verifier.cc(93)] signature blob size = 264
03-22 07:53:52.194  2541  2541 I update_engine:
[0322/075352:INFO:payload_verifier.cc(112)] Verified correct signature 1 out of
1 signatures.
03-22 07:53:52.194  2541  2541 I update_engine:
[0322/075352:INFO:delta_performer.cc(1633)] Payload hash matches value in
```

```
payload.
03-22 07:53:52.194  2541  2541 I update_engine:
[0322/075352:INFO:action_processor.cc(116)] ActionProcessor: finished
DownloadAction with code ErrorCode::kSuccess
03-22 07:53:52.194  2541  2541 I update_engine:
[0322/075352:INFO:action_processor.cc(143)] ActionProcessor: starting
FilesystemVerifierAction
03-22 07:53:52.195  2541  2541 I update_engine:
[0322/075352:INFO:filesystem_verifier_action.cc(157)] Hashing partition 0 (boot)
on device /dev/block/by-name/boot_e
03-22 07:53:52.689  2541  2541 I update_engine:
[0322/075352:INFO:filesystem_verifier_action.cc(248)] Hash of boot:
jTasb96DQoY9u1KjnKLfEzJCLuOZJ+AE8e8tIXKGl5k=
03-22 07:53:52.694  2541  2541 I update_engine:
[0322/075352:INFO:filesystem_verifier_action.cc(157)] Hashing partition 1
(system) on device /dev/block/by-name/system_e
03-22 07:54:10.694  2541  2541 I update_engine:
[0322/075410:INFO:filesystem_verifier_action.cc(248)] Hash of system:
7WrR01G8iz3x3NEbizeBj9MhdmEZe7cs1ChJO/gZlpw=
03-22 07:54:10.732  2541  2541 I update_engine:
[0322/075410:INFO:action_processor.cc(116)] ActionProcessor: finished
FilesystemVerifierAction with code ErrorCode::kSuccess
03-22 07:54:10.732  2541  2541 I update_engine:
[0322/075410:INFO:action_processor.cc(143)] ActionProcessor: starting
PostinstallRunnerAction
03-22 07:54:10.732  2541  2541 I bcm-bootc: setActiveBootSlot(1): reset stats
03-22 07:54:10.733  2541  2541 I update_engine:
[0322/075410:INFO:postinstall_runner_action.cc(341)] All post-install commands
succeeded
03-22 07:54:10.733  2541  2541 I update_engine:
[0322/075410:INFO:action_processor.cc(116)] ActionProcessor: finished last
action PostinstallRunnerAction with code ErrorCode::kSuccess
03-22 07:54:10.733  2541  2541 I update_engine:
[0322/075410:INFO:update_attempter_android.cc(282)] Processing Done.
03-22 07:54:10.735  2541  2541 I update_engine:
[0322/075410:INFO:update_attempter_android.cc(291)] Update successfully applied,
waiting to reboot.
```

Reboot the device at this point and notice that the active slot has switched from its prior run (which means if running _i the new active slot would become _e, and vice-versa). The active slot information is seen on the boot log early on when bsu runs and selects the proper slot to boot.