TF-A Tech Forum

Bloblist proposals

Manish Pandey & Madhukar Pappireddy

03-June-2021

Background

- Continuation of ongoing discussion on mechanism to pass dynamic information through boot phases.
- Use list of pre-defined C data structures(blobs) to pass information, let's call it bloblist.
- Blobs can be identified by tags/UUID or mix of both

Information passing through boot phases

- Proposal 1: Always pass a pointer to the device tree blob through the GP register and capture the pointer to the bloblist head as a property of a node that is uniquely identifiable by the downstream boot phase. This needs to define a device tree binding such that producer and consumer agree on the information passed.
- Proposal 2: Pass a pointer to a generic container through the GP register that can be interpreted appropriately by both boot loaders(i.e., producer and consumer of the boot info). This container can either be a dtb or a bloblist which can be simply inferred by checking for a magic header that indicates if the buffer appears to be a flattened device tree.
- Proposal 3: Pass pointer to the bloblist head through GP register and DT can be a blob, which will have a unique tag.

Points raised during tech forum

- During tech forum discussion option 1 and 3 have got more votes

Proposal 1

- Not much change required in FW which are currently using DT to pass information
- DT awareness in early FW is not common
- Backward compatible

Proposal 3

- Using simple C structs can be easily extended to various projects
- Not required to have DT awareness
- Need to check backward compatibility

Complete Boot Flow proposals

- Assuming proposal 1 in previous slide the boot flow will be different, to align with existing differences, among segments
- Client Segment:
  - TF-A creates bloblist and adds base address in NTFW Config DT node
  - U-boot parses and consumes/appends bloblist
  - U-boot appends bloblist address in NW Config DT node.
  - No major change required
- Infra Segment:
  - TF-A creates bloblist and adds base address in NTFW Config DT node
  - edk2 parses and consumes/appends bloblist (edk2's capability to parse DT?)
  - edk2 adds bloblist in System tables passed to Linux(along with ACPI/SMBIOS table)
  - Changes required in edk2 and Linux EFI stub
  - May need to reserve a GUID for bloblist

Points raised during tech forum

- Whether OS will consume any of this information or it should terminate at secondary boot loader(U-boot/edk2) ?
- Since there is no use case which cannot be handled by existing contract between secondary boot loaders and OS, so consensus was to terminate at secondary boot loader until any obvious use case arises.

Identification of bloblist entries

- Use simple 64-bit tags. Reserve tag ranges for platform specific purpose.
- Another suggestion was to use a hybrid approach. Reserve a single tag ID for identifying/constructing a blob that further leverages UUID based identifier. This way, the generic bloblist doesn't need to support UUIDs and can work with tags.
- Even we can distinguish among segments, tag only for Client platforms and Hybrid for Infra platforms.
- Using UUID will be platform choice

Conclusion

- Based on tech forum/internal discussions, it appears that proposal 3 would be more acceptable across various Firmware projects as it uses simple C structs and does not have dependency on DT
- Use x0 to pass pointer to bloblist head
- Compatibility with existing implementations?
  - Existing implementations may already be using x0 for passing information
    - To overcome this problem, create a blob for current information being passed.
  - For existing platforms early Firmware may not be upgradable so it can't be mandated to change usage of x0
    - To overcome this problem, make it a platform choice during transition period.

What next?

- Following changes can be done in core TF-A
  - Define tags for generic blobs
  - Hooks to define platform specific blobs
  - APIs to add/remove/traverse bloblist
  - Migrate current usage of x0 to blobs (under a build flag)
- Demonstration on FVP platform
  - Modify code to repurpose x0 on BL1<->BL2 and BL2<->BL31 boundry
  - Modification to FCONF framework
- Partner platforms
  - Using framework provided by core TF-A, platforms can add their own blobs
  - Extend to use UUID based blobs

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה