TF-A Tech Forum

Passing dynamic information through boot phases

Manish Pandey, Madhukar Pappireddy

May 2021

Information passing through boot phases

- Static
  - Configuration data known at compilation
  - Exist for quite some time and standardized across projects
  - Existing implementation
    - Device tree/ACPI tables
    - FCONF in TF-A
- Dynamic
  - Discovered at run time by probing a device, reading physical registers etc
  - Used in individual projects(no standardization)
  - Existing implementation
    - Modifying DT at runtime
    - C structures: HOB(uefi), Bloblist (u-boot), aux_params(coreboot/TF-A)
    - SMC Calls ?
- These two use cases are quite different and should not be mixed.

Use case

- Dynamically generated information produced by early boot phases and consumed by later boot phases. Required only once during boot process.
- Goal is to standardize the way dynamic information is passed between boot phases, at least for a given segment".
- Why can't we use existing solutions
  - Modifying DT at runtime: less efficient, dependency on DT library in project. (currently being used for measured boot).
  - SMC Calls: Overkill/Security implications, as the information is required only once.
  - HOB/Bloblist/aux_list: Not standard(across projects) but a good reference point to start with.
- We can start with standardizing C data structures usage
- Start with TF-A(bottom up), agreement with other projects

C data structure implementations

- HOB
  - UEFI PI spec describes it for transitioning between the PEI and DXE boot phases
  - Edk2 implements GUID based HOB list
- Bloblist(U-boot)
  - List of blobs of data
  - Each record information has a 32-bit tag value
  - Auto-allocating enums
- Aux_list(TF-A)
  - List of blobs of data
  - 64-bit tag value (32 bits currently used)
  - Explicit values for each entry and a range for 'local' use

Tags or UUIDs?

- Tags
  - Simple and avoids bloated data structures with UUID
  - Though we can reserve tag ranges but still there is possibility of tag collision
- UUIDs
  - Complex data structure and increased code size
  - No collision, parallel development
- Hybrid approach (Proposed)
  - Reserve a tag for data structures using UUID
  - Constraint Firmware can use only tags
  - Richer Firmware can leverage UUID

What next

- Standardization on Physical register use to pass base of C data structure list.
- Implement bloblist in platform specific code(in TF-A) and other projects.
- TF-A code can be moved from platform specific to generic code (can be generalized on per-segment basis).
- Further considerations
  - Multiple bloblists?
  - Checksum to finish bloblist generation at each stage?

References

- TF-A mailing list discussion: https://lists.trustedfirmware.org/pipermail/tf-a/2021-April/001069.html
- trusted-substrate call recording: https://linaro-org.zoom.us/rec/share/zjfHeMIumkJhirLCVQYTHR6ftaqyWvF_0klgQnHTqzgA5Wav0qOO8n7SAM0yj-Hg.mLyFkVJNB1vDKqw_ Passcode: IPn+5q%
- Edk2 HOB: https://github.com/tianocore/edk2/blob/master/MdePkg/Include/Library/HobLib.h
- U-boot bloblist: https://github.com/u-boot/u-boot/blob/master/doc/README.bloblist
- TF-A aux list: https://review.trustedfirmware.org/plugins/gitiles/TF-A/trusted-firmware-a/+/refs/heads/master/include/export/lib/bl_aux_params/bl_aux_params_exp.h

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