Technical Overview of Trusted Firmware-A

Embedded Linux Conference

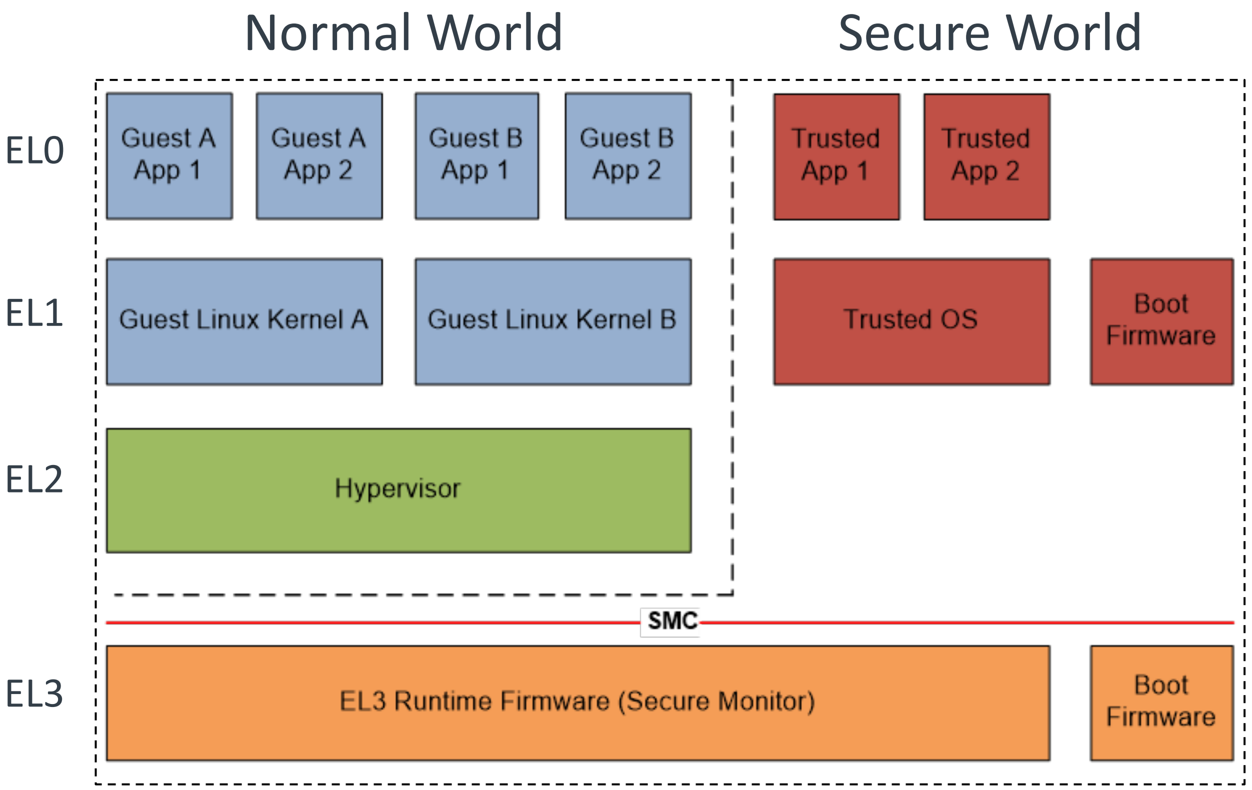
Sandrine Bailleux and Joanna Farley
June 2020

TF-A Project

Foundational Features

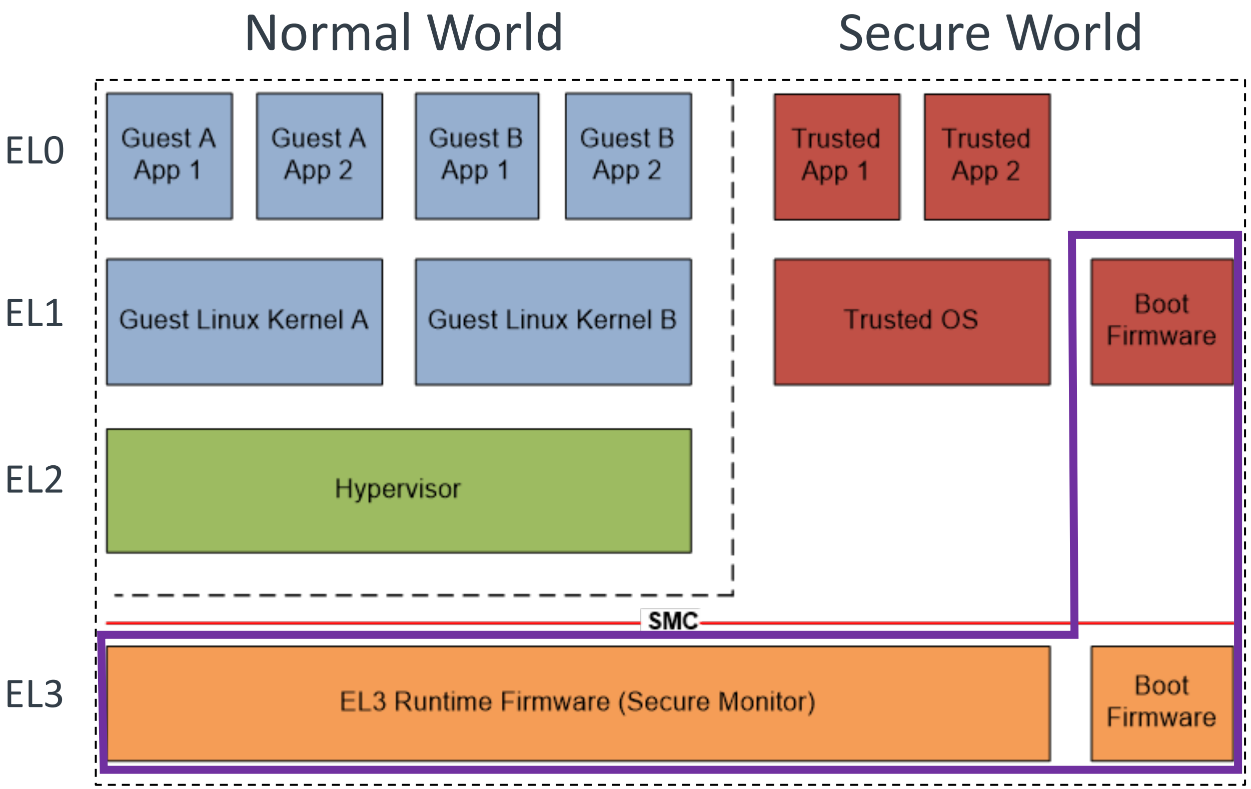
What is Trusted Firmware-A?

- Reference implementation of secure world software (EL3) for Armv7-A and Armv8-A
  - For all Arm Cortex-A & Neoverse processors
  - Across all market segments
- Foundation to build a Trusted Execution Environment (TEE)
- Designed for reuse or porting to other platforms
- 30+ platform ports supported upstream
- 16+ different vendors
- Open source project since October 2013
- BSD-3-Clause license
- Contributions accepted under the term of Developer Certificate of Origin
- Open governance model on trustedfirmware.org
- 6-monthly releases



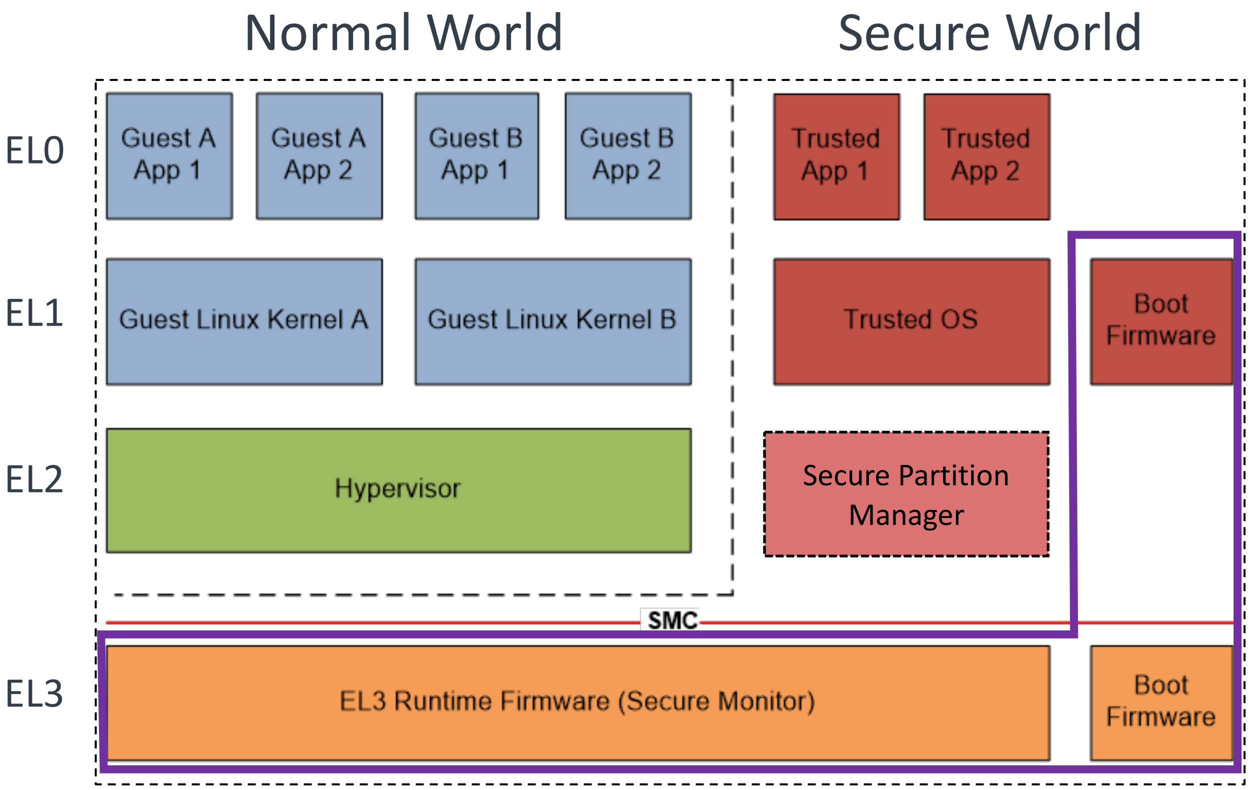
What is Trusted Firmware-A?

- Reference implementation of secure world software (EL3) for Armv7-A and Armv8-A
  - For all Arm Cortex-A & Neoverse processors
  - Across all market segments
- Foundation to build a Trusted Execution Environment (TEE)
- Designed for reuse or porting to other platforms
- 30+ platform ports supported upstream
- 16+ different vendors
- Open source project since October 2013
- BSD-3-Clause license
- Contributions accepted under the term of Developer Certificate of Origin
- Open governance model on trustedfirmware.org
- 6-monthly releases



What is Trusted Firmware-A?

- Reference implementation of secure world software (EL3) for Armv7-A and Armv8-A
  - For all Arm Cortex-A & Neoverse processors
  - Across all market segments
- Foundation to build a Trusted Execution Environment (TEE)
- Designed for reuse or porting to other platforms
- 30+ platform ports supported upstream
- 16+ different vendors
- Open source project since October 2013
- BSD-3-Clause license
- Contributions accepted under the term of Developer Certificate of Origin
- Open governance model on trustedfirmware.org
- 6-monthly releases

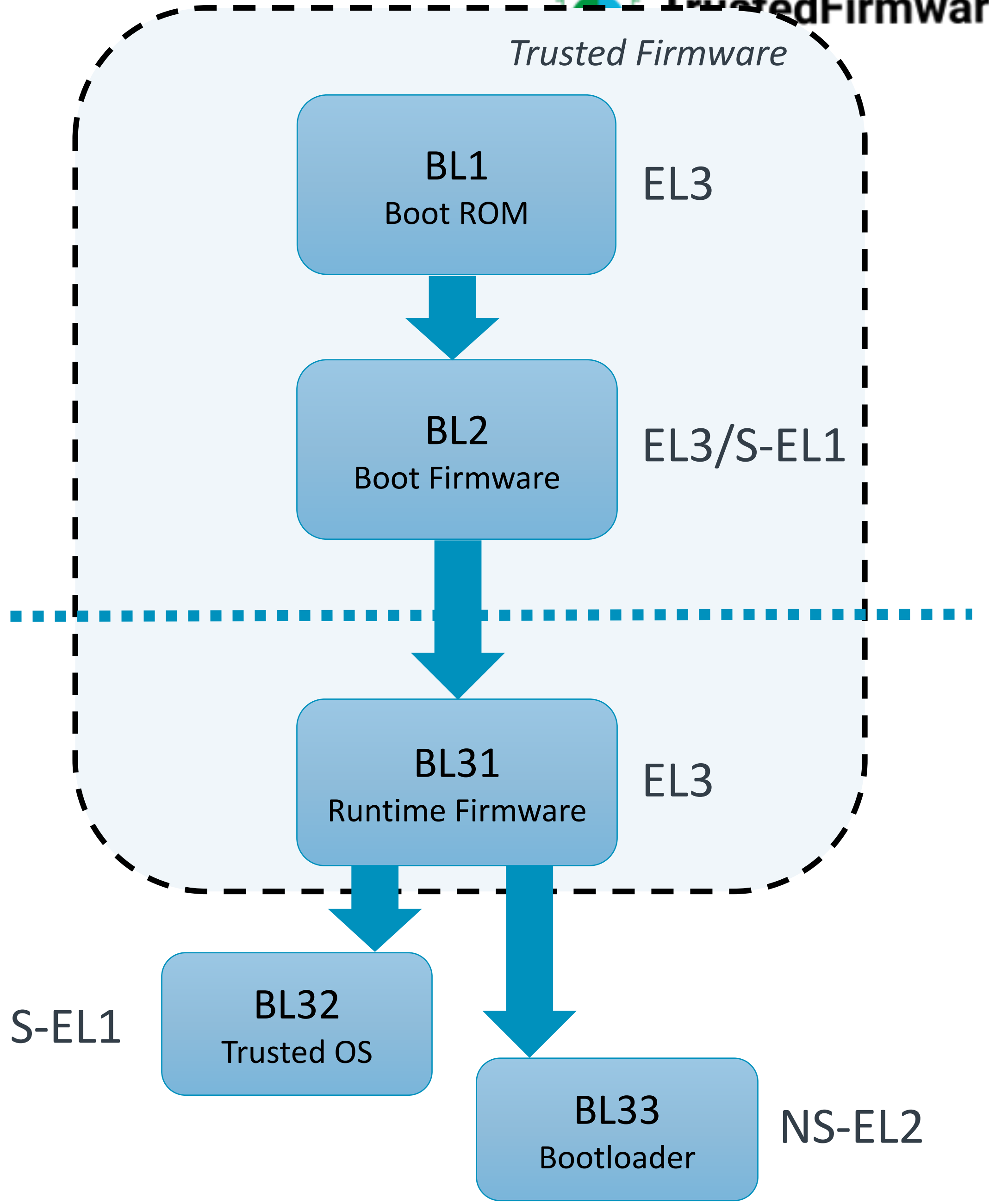


Boot Flow

Several firmware stages

- BL1 and BL2 are transient images
  - Discarded after the boot
- Not used by all platforms
  - Proprietary/custom firmware
  - Existing firmware pre-dating TF-A

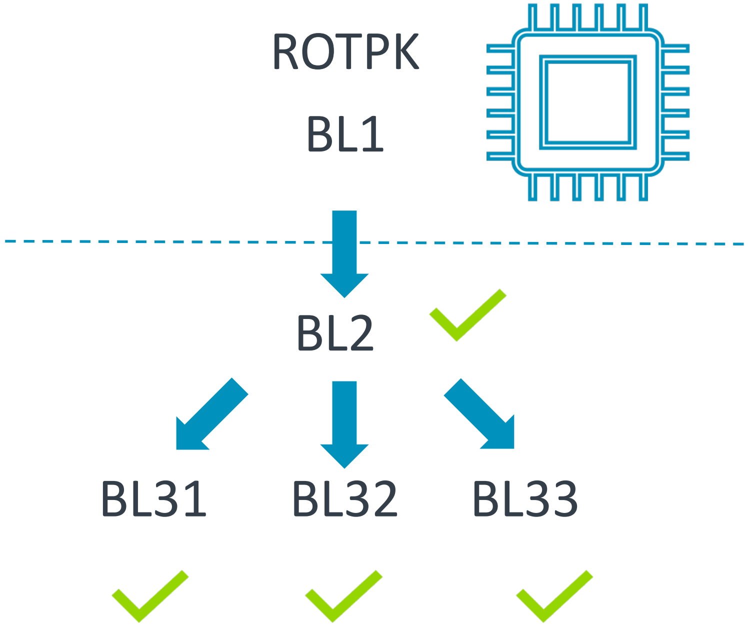
- BL31 is runtime resident
- Provide runtime services...
  - Power management, Arm architectural services, SoC services, board services
- ...to lower exception levels
  - Rich OS
  - Trusted OS (OP-TEE, Android Trusty TEE, NVIDIA TLK,...)



Trusted Boot

Ensuring the integrity of the firmware

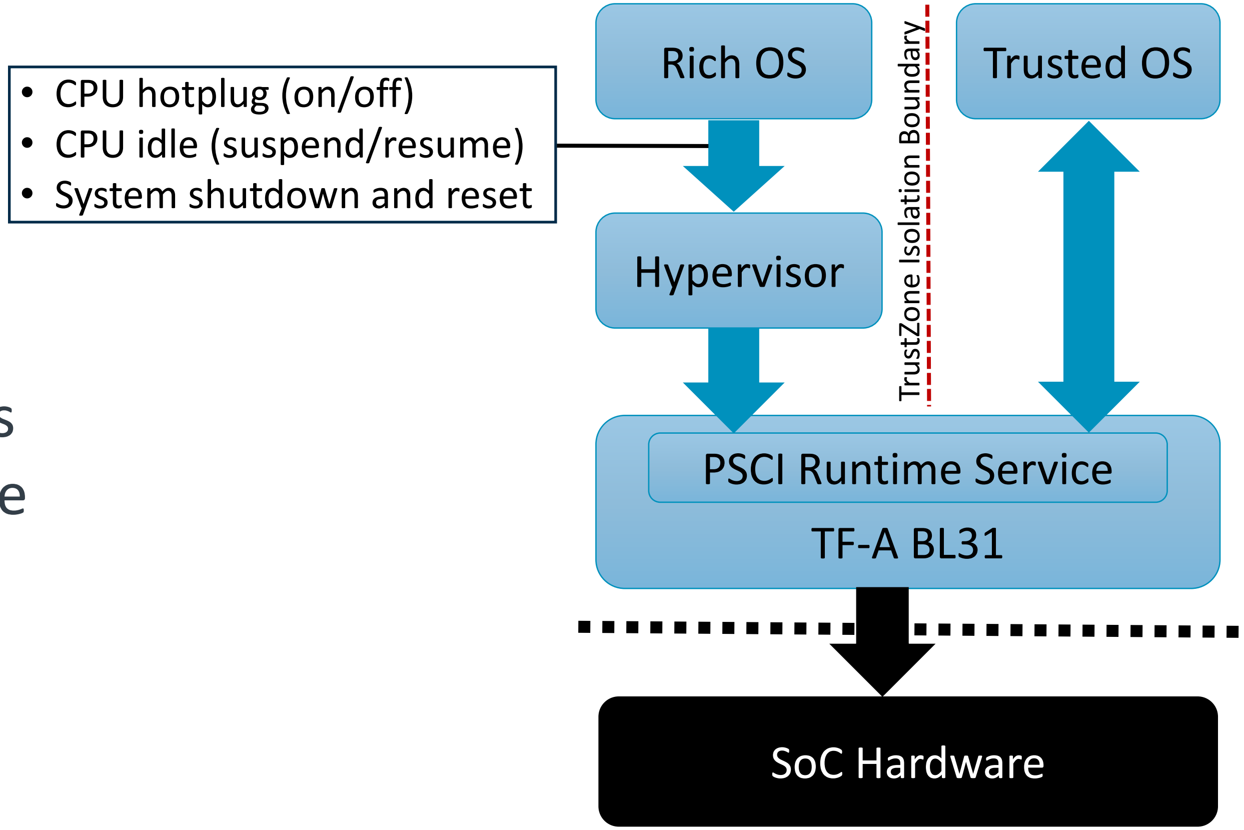
- TBFU (Trusted Boot Firmware Update) Compliant
- Based on a hardware root of trust
  - Immutable root-of-trust public key
  - Immutable secure boot ROM firmware
- Each firmware stage verifies the signature of the next one
  - From ROM firmware (BL1) up to normal world bootloader (BL33)
- Refuse to boot on authentication error
- Optional integration with cryptographic hardware (e.g. Arm CryptoCell-712/713)
- On-going work for multiple signing domains
  - Multiple root-of-trust keys for independent software providers
- Optional firmware encryption for confidentiality/anticloning (e.g. DRM use cases)



Power Management

- Power State Coordination Interface (PSCI) library

- Arbitrate power management requests from Non Secure world with the Secure world notified of these requests



- CPU hotplug (on/off)
- CPU idle (suspend/resume)
- System shutdown and reset

Rich OS

Hypervisor

TrustZone Isolation Boundary

Trusted OS

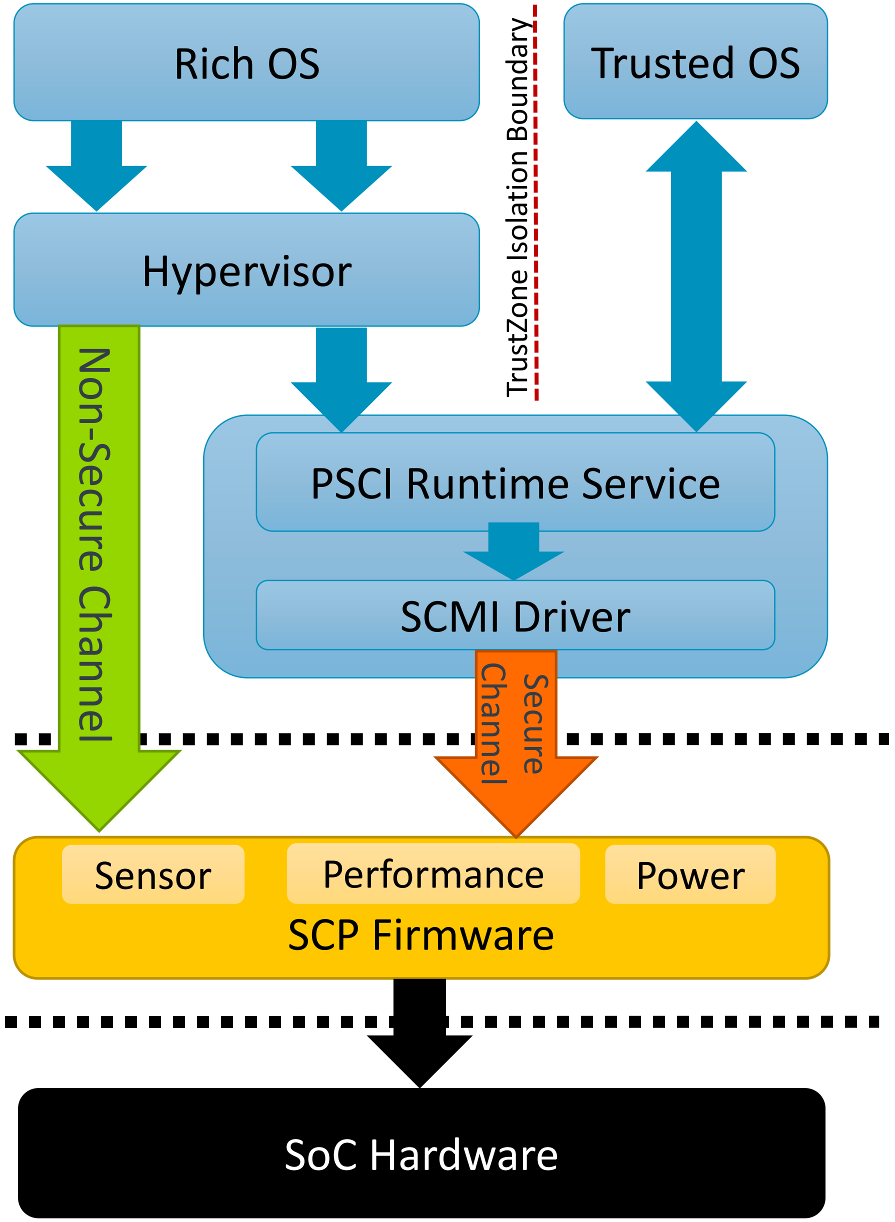
PSCI Runtime Service

TF-A BL31

SoC Hardware

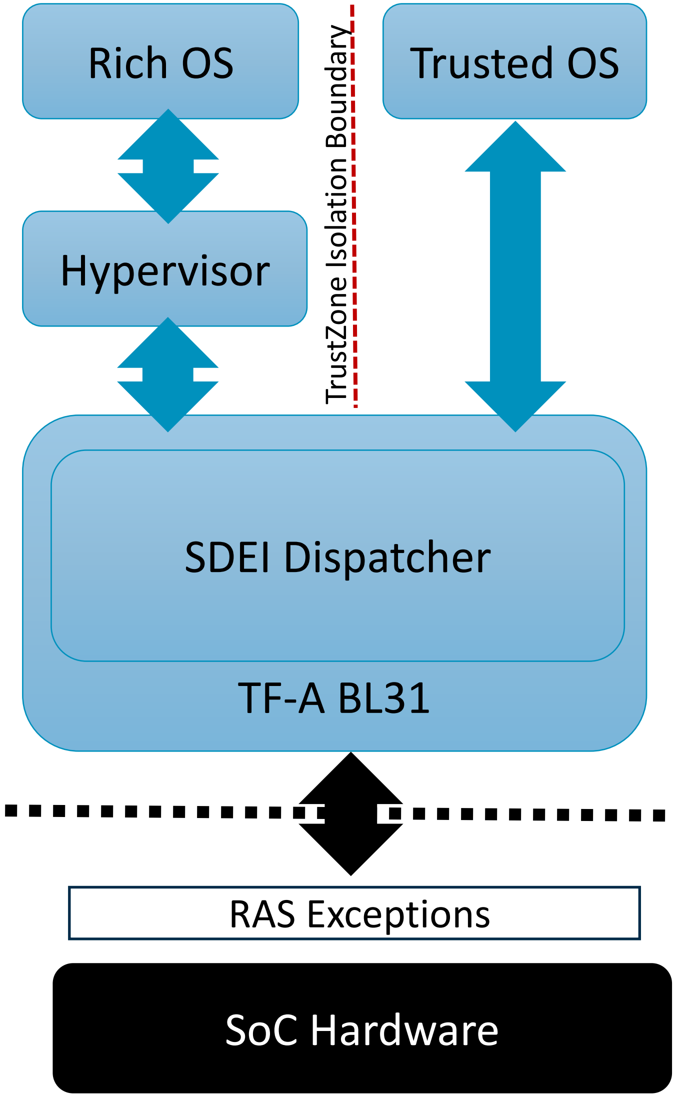
Power Management

- Power State Coordination Interface (PSCI) library
- Arbitrate power management requests from Non Secure world with the Secure world notified of these requests
- System Control and Management Interface (SCMI) driver
  - Standardized interface for power, performance and resource management on a SoC
  - Requires a conforming power controller
    - Arm System Control Processors (SCP)
  - Allows to delegate power management to SCP
  - Enables a platform-agnostic AP firmware



Exception Handling

- Software Delegated Exception Interface (SDEI)
  - Deliver extraordinary System events
  - SDEI Dispatcher implemented in BL31
  - OS or hypervisor register system event callback
  - When triggered be serviced **immediately** by an OS or hypervisor
- Up to 2 priority levels of SDEI events
  - Normal priority
  - Critical priority
- Events can be software or hardware generated
  - Hardware: Interrupts, exceptions
  - Software: Software Generated Interrupts/Events
- Current implemented use case support
  - Platform error handling (RAS)



Armv8 Architecture Enablement

https://developer.arm.com/tools-and-software/open-source-software/firmware/trusted-firmware/trusted-firmware-a/tf-a-architectural-features

FEATURE	TF-A VERSION	ADDITIONAL INFORMATION
Armv8.1-LSE	v1.4 Spinlock	CAS only
Armv8.2-TTCNP	v2.1	Translation table library update
Armv8.2-RAS	v1.5	SDEI, EHF and SPM components
Armv8.2-SPE	v1.4 Lower ELs (Normal world)	Statistical Profiling Extension
Armv8.2-SVE	v1.5 Lower ELs (Normal world)	Scalable Vector Extension
Armv8.3-Pauth	v2.1 Lower ELs (Normal world) v2.2 EL3 and Secure world ELs	
Armv8.4-DIT	v2.1	
Armv8.4-RAS	v1.6	

FEATURE	TF-A VERSION	ADDITIONAL INFORMATION
Armv8.4-TTST	v2.1	
Armv8.4-MPAM	v1.6 Lower ELs (Normal world)	Normal world only
Armv8.4-AMU	v1.5	Enabled for Cortex-A75 and Neoverse-N1, plus all newest Armv8.4 cores
Armv8.4-SecEL2		Ongoing work
Armv8.5-PMU	v2.1	
Armv8.5-SSBS	v2.1	Cortex-A76 and Neoverse-N1
Armv8.5-BTI	v2.2	
Armv8.5-MTE	v2.2 Lower ELs (Normal world)	

Generic Firmware

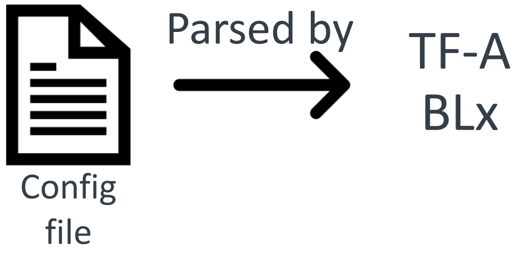
Latest features

Generic Firmware

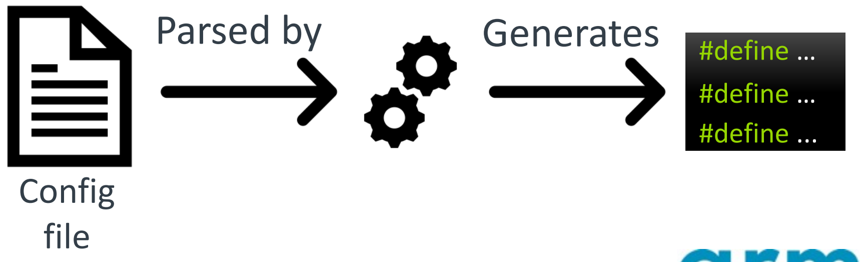
Sample of arm_def.h

```
#define ARM_SHARED_RAM_BASE UL(0x04000000)
#define ARM_SHARED_RAM_SIZE   UL(0x00001000)
#define ARM_IRQ_SEC_SGI_0   8
#define ARM_IRQ_SEC_SGI_1   9
#define ARM_CONSOLE_BAUDRATE 115200
```

- Today: Firmware binaries are tied to a platform
  - Lots of platform header files
  - Built-in platform information (memory map, interrupts, …)
- Goal: A single firmware stack runs across a range of platforms
  - Much like the Linux kernel today
  - By moving all differentiating configuration options to a configuration file
  - Configuration file parsed at boot time for self-configuration



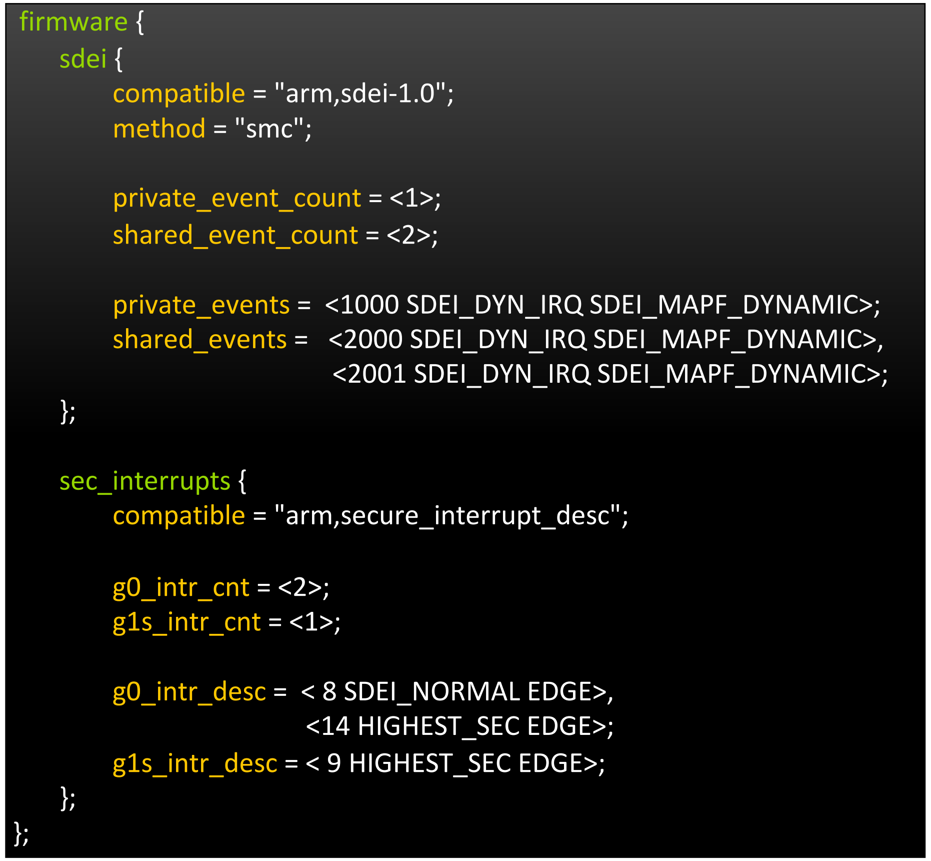
- Not for all market segments (e.g. highly constrained devices)
  - Performance overhead
  - Memory footprint increase
  - More complexity
- Could use config files even for static platform data
  - Tool to convert config files to static platform data *
  - Benefit: Centralize platform data



(*) Not implemented yet.

Configuration Information

- Using DTB format for the config files (libfdt)
  - Might support alternate formats in the future
- Traditional hardware configuration
  - CPU topology
  - Console base address, baudrate, ...
  - Secure watchdog
- Secure firmware features
  - Enable/disable Trusted Boot
  - Configure log level
  - Load address/size of images to load/authenticate
- Modification of configuration as seen by other software
  - Probed runtime memory
  - Secure memory reservation
  - Kernel boot arguments



```
firmware {
    sdei {
        compatible = "arm,sdei-1.0";
        method = "smc";

        private_event_count = <1>;
        shared_event_count = <2>;

        private_events =  <1000 SDEI_DYN_IRQ SDEI_MAPF_DYNAMIC>;
        shared_events =   <2000 SDEI_DYN_IRQ SDEI_MAPF_DYNAMIC>,
                          <2001 SDEI_DYN_IRQ SDEI_MAPF_DYNAMIC>;
    };

    sec_interrupts {
        compatible = "arm,secure_interrupt_desc";

        g0_intr_cnt = <2>;
        g1s_intr_cnt = <1>;

        g0_intr_desc =  < 8 SDEI_NORMAL EDGE>,
                        <14 HIGHEST_SEC EDGE>;
        g1s_intr_desc = < 9 HIGHEST_SEC EDGE>;
    };
};
```

- Configuration of a specific firmware component
  - DDR training parameters
  - TrustZone Controller security policies

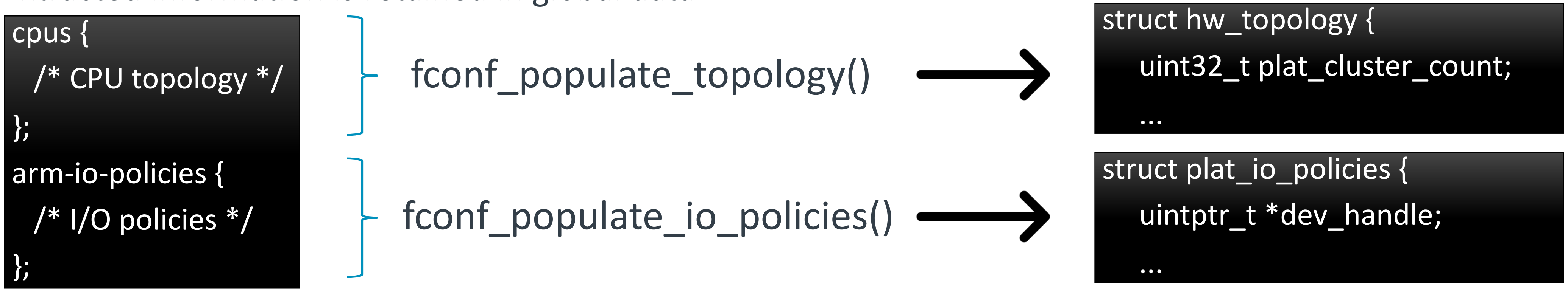
Firmware Configuration Framework (FCONF)

A data abstraction layer to access the configuration data

1. Module registers a callback which extracts configuration data
   - Example: Parse hardware DT to extract platform topology info:

```
FCONF_REGISTER_POPULATOR(HW_CONFIG, topology, fconf_populate_topology);
```

   - All callbacks gathered in a `.fconf_populator` linker section
2. Configuration data is parsed at boot time
   - Every registered callback is called
   - Extracted information is retained in global data



```
cpus {
  /* CPU topology */
};
arm-io-policies {
  /* I/O policies */
};
```

fconf_populate_topology() →

```
struct hw_topology {
    uint32_t plat_cluster_count;
    …
```

fconf_populate_io_policies() →

```
struct plat_io_policies {
    uintptr_t *dev_handle;
    …
```

3. Module queries global configuration data

```
FCONF_GET_PROPERTY(hw_config, topology, plat_cluster_count)
```

FCONF without a Configuration File

A data abstraction layer to access the configuration data

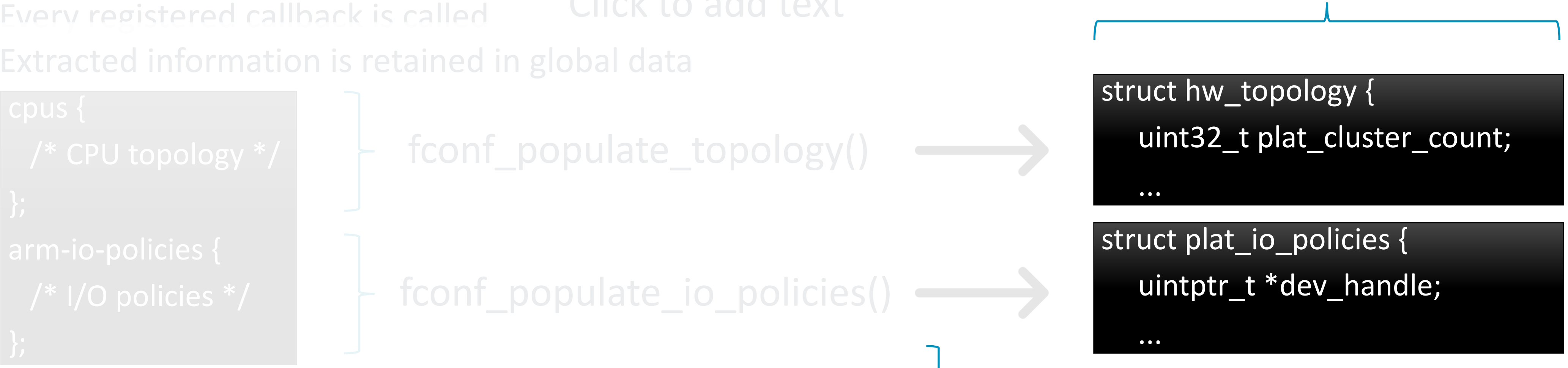
1. Module registers a callback which extracts configuration data
   - Example: Parse hardware DT to extra platform topology info:

   ```
   FCONF_REGISTER_POPULATOR(HW_CONFIG, topology, fconf_populate_topology);
   ```

   - All callbacks gathered in a `.fconf_populator` linker section
2. Configuration data is parsed at boot time
   - Every registered callback is called
   - Extracted information is retained in global data

Click to add text

Provided by platform layer



```
cpus {
  /* CPU topology */
};
arm-io-policies {
  /* I/O policies */
};
```

```
struct hw_topology {
    uint32_t plat_cluster_count;
    ...
```

```
struct plat_io_policies {
    uint_ptr_t *dev_handle;
    ...
```

3. Module queries global configuration data

   ```
   FCONF_GET_PROPERTY(hw_config, topology, plat_cluster_count)
   ```

Does not change, whether config data comes from config file or platform data

arm

Rearchitecturing the Secure World Software

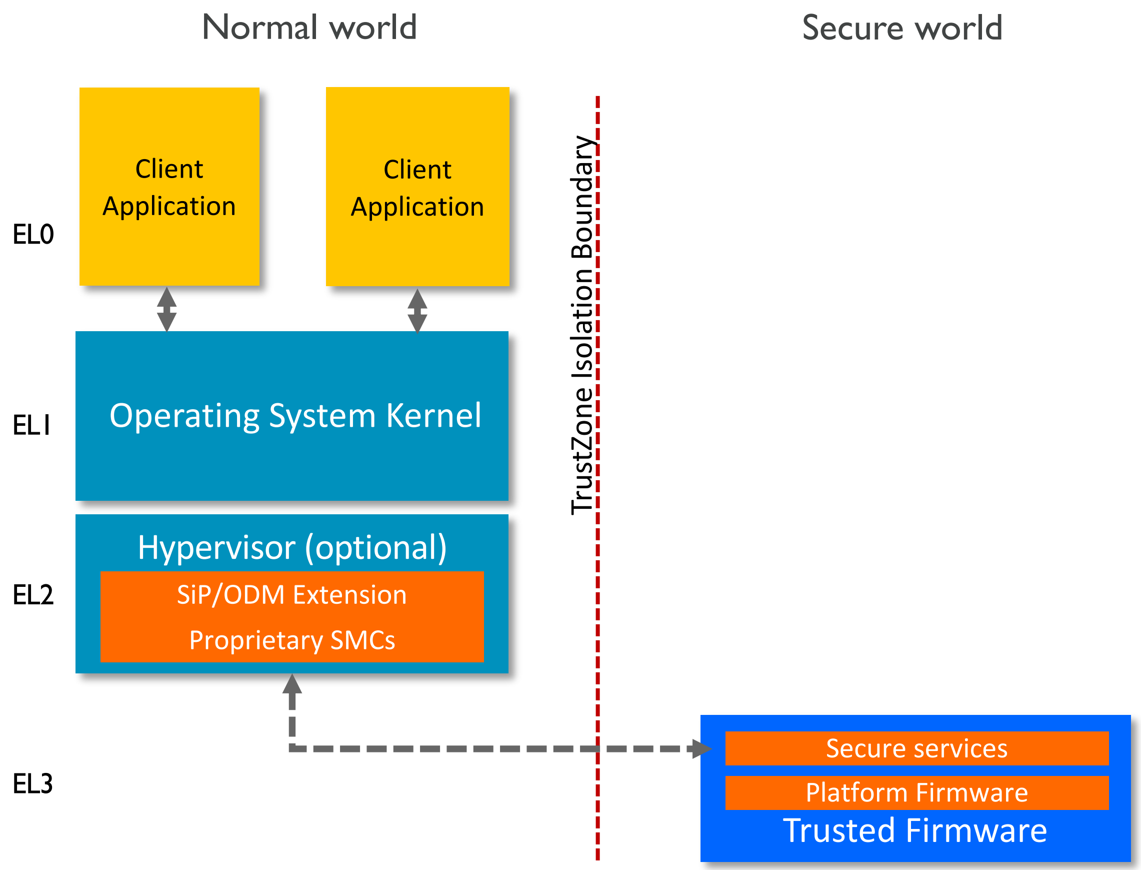
Latest features

Secure World Software Architecture Today

Without a Trusted OS

- EL3 firmware provides lots of services
- Increases code complexity
- Increases attack surface
- Increases fragmentation (platform custom services)

🟩 Application trusted OS specific
🟨 Application provider specific
🟦 Generic software
🟦 TrustedFirmware.org
🟧 Silicon Vendor specific software



Secure services:

- DRM
- Secure payment
- Secure storage
- Crypto

Platform services:

- Trusted boot
- Power management (PSCI)
- Silicon vendor services
- Errata management

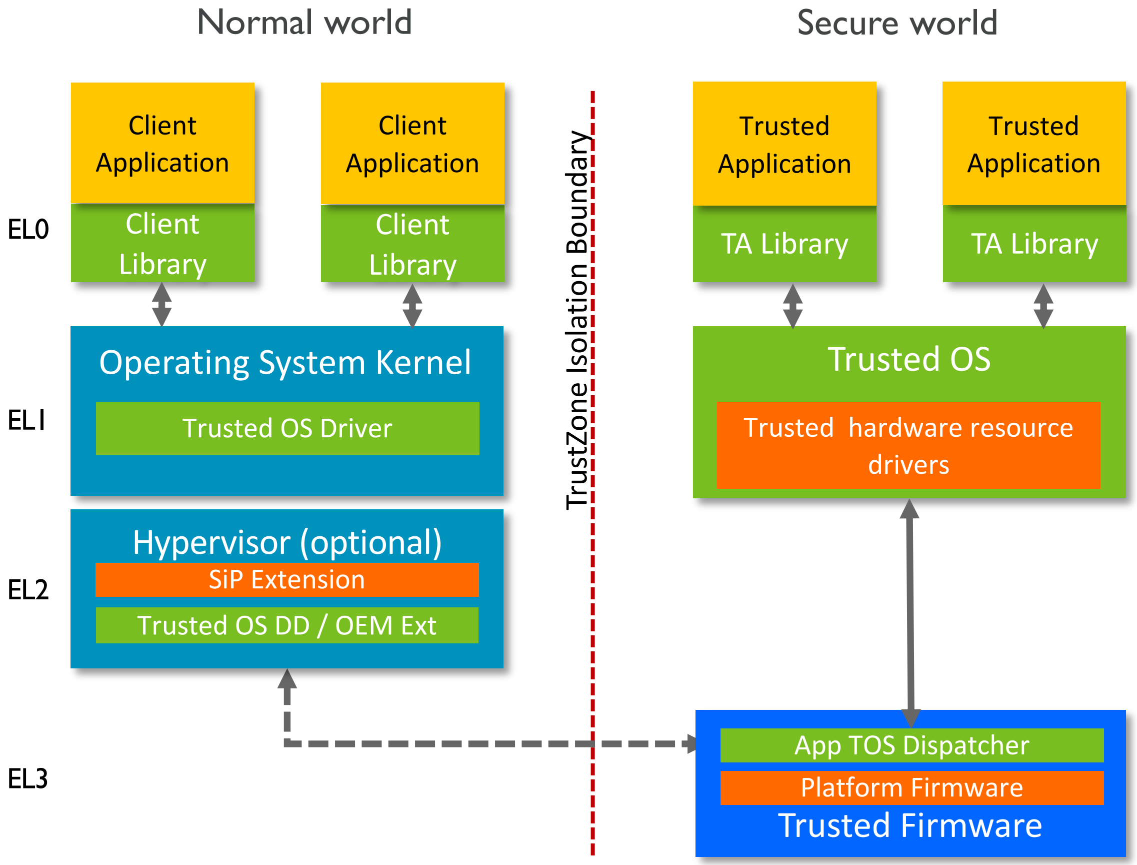
Secure World Software Architecture Today

With a Trusted OS

- 🟩 Application trusted OS specific
- 🟨 Application provider specific
- 🟦 Generic software
- 🟦 TrustedFirmware.org
- 🟧 Silicon Vendor specific software

- Secure services are provided by the Trusted OS
- Platforms services are still in EL3 firmware

- No hardware isolation between S-EL1 and EL3
- Requires some TOS specific components across the software stack



Secure services:

- DRM
- Secure payment
- Secure storage
- Crypto

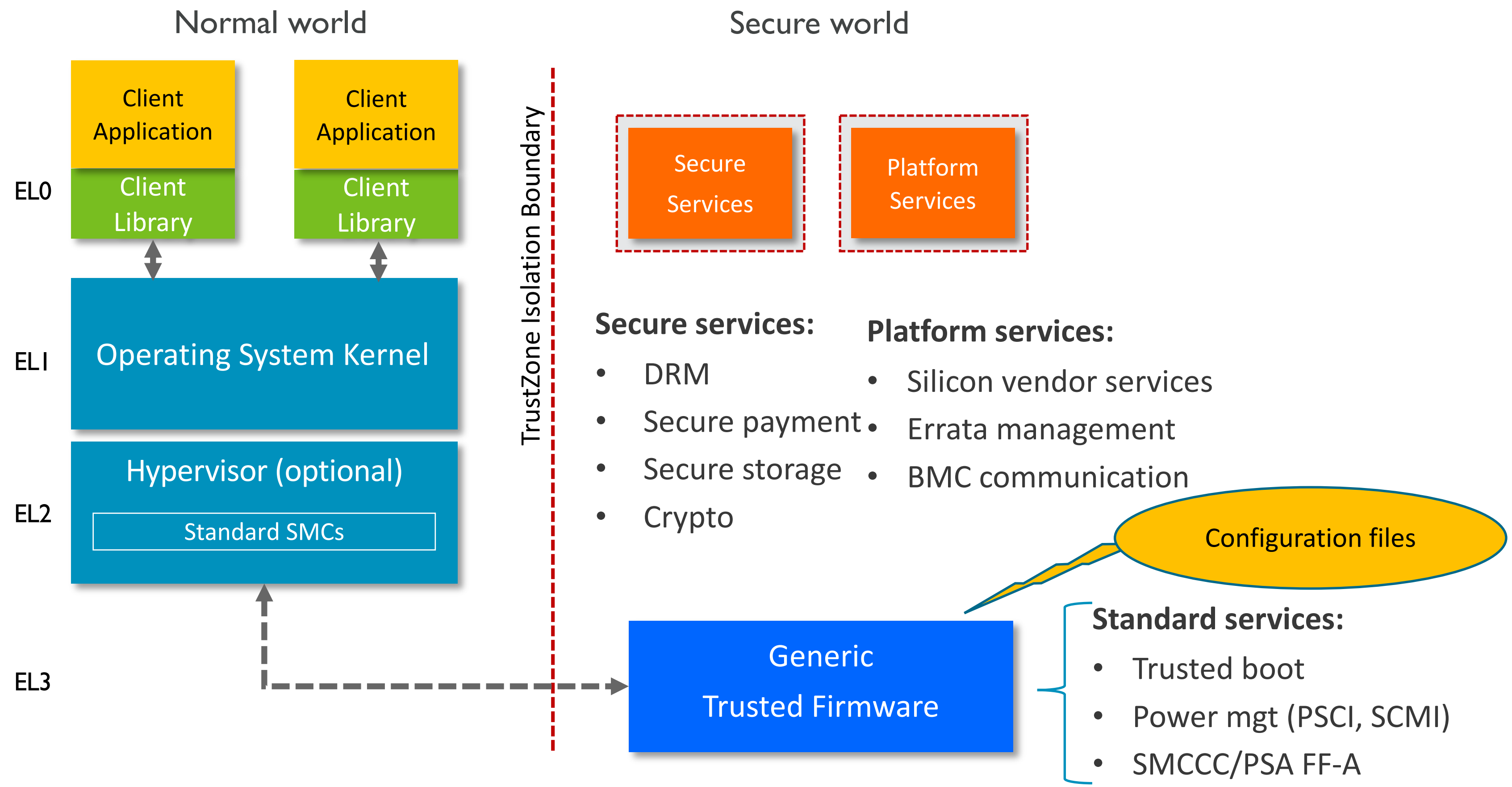
Platform services:

- Trusted boot
- Power management (PSCI)
- Silicon vendor services
- Errata management

Secure World Software Architecture Goal

- 🟩 Application trusted OS specific
- 🟨 Application provider specific
- 🟦 Generic software
- 🟦 TrustedFirmware.org
- 🟧 Silicon Vendor specific software

- Move services upper the exception levels (S-EL0)
- Keep the EL3 firmware minimal
- Reduces firmware attack surface
- Reduces firmware complexity
- Ease auditing and certification
- Allows to have a generic firmware (free of platform specific services)



Normal world

Secure world

EL0

EL1

EL2

EL3

Client Application

Client Library

Client Application

Client Library

Operating System Kernel

Hypervisor (optional)

Standard SMCs

TrustZone Isolation Boundary

Secure Services

Platform Services

Secure services:

- DRM
- Secure payment
- Secure storage
- Crypto

Platform services:

- Silicon vendor services
- Errata management
- BMC communication

Configuration files

Generic Trusted Firmware

Standard services:

- Trusted boot
- Power mgt (PSCI, SCMI)
- SMCCC/PSA FF-A

Leveraging Armv8.4 Secure Virtualization

🟩 Application trusted OS specific

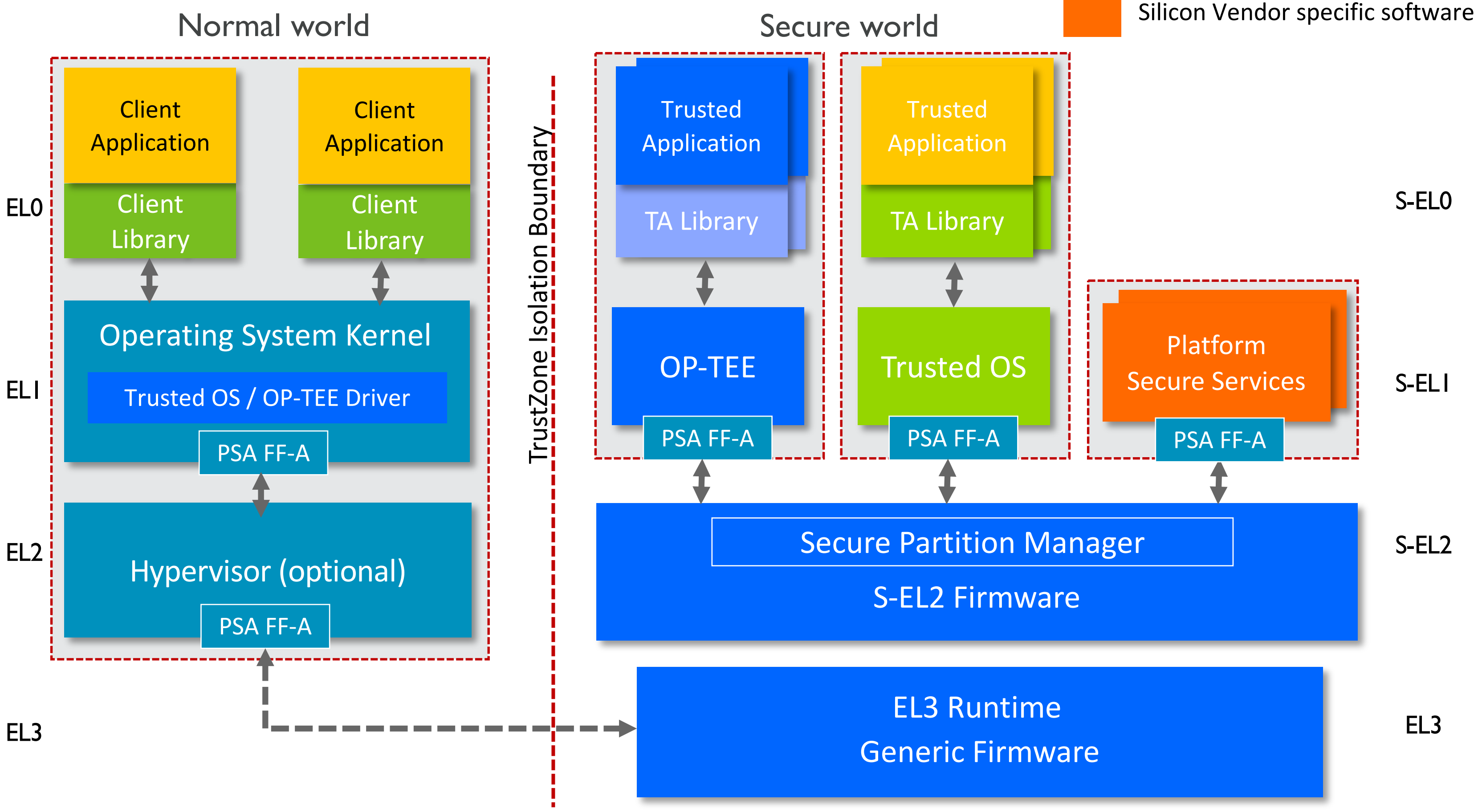
🟨 Application provider specific

🟦 Generic software

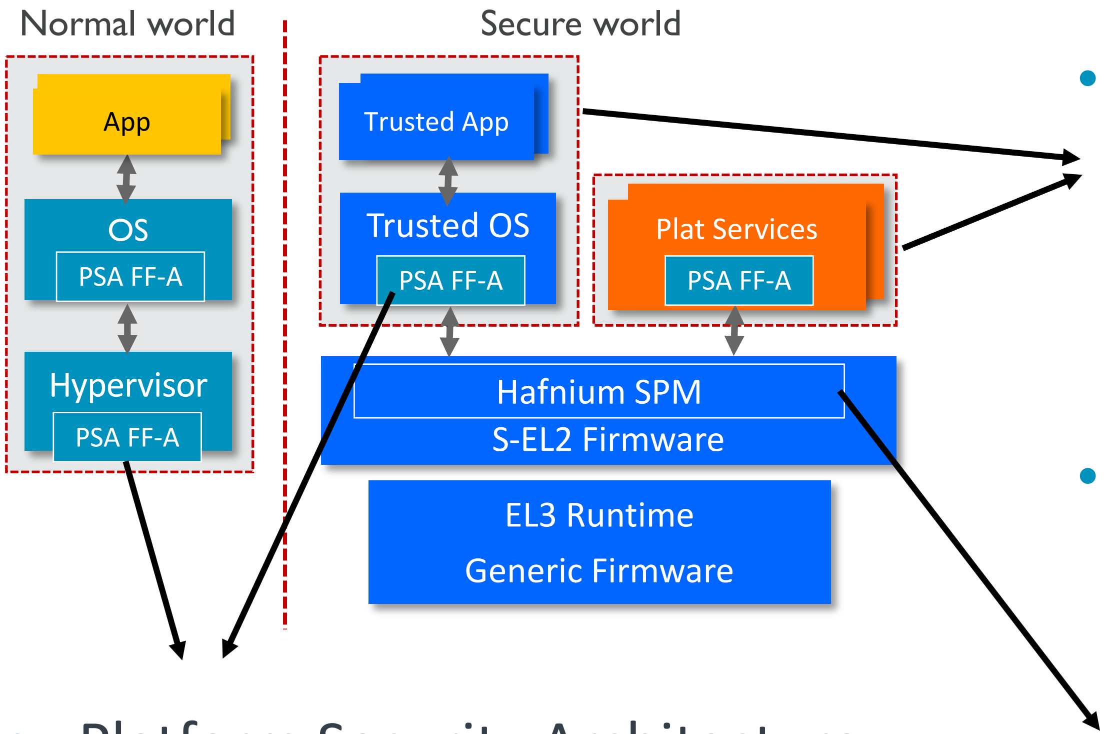
🟦 TrustedFirmware.org

🟧 Silicon Vendor specific software

- Isolation through virtualization in the Secure world
- Standardization of interfaces between Normal and Secure world through Arm PSA FF-A compliance
- Generic Secure Firmware spanning EL3 & S-EL2
- Support for multiple Trusted OSes (isolated from each other)



Secure World Architecture Building Blocks



- Secure Partitions (SP)
  - Mutually distrustful software sandboxes running in the Secure world
  - Isolated execution context and address space
  - Limited access to system resources
- Secure Partition Manager (SPM)
  - Responsible for:
    - Initializing secure partitions at boot time
    - Enabling communication between service requestors and providers
    - Managing runtime requests
  - Enforces principle of least privilege
  - Initial PSA FF-A compliant SPM Dispatcher
  - Hafnium as the reference Secure EL2 SPM of choice
    - Migrated by Google into TrustedFirmware.org
- Platform Security Architecture, Firmware Framework for A-class processors (PSA FF-A)
  - Standard set of interfaces between SPs/SPM
  - Between SPs and Normal world

Useful Project Links

- TF-A mailing list for technical discussions
- TF-A open Tech Forum bi-weekly call
- CGit to browse the source code
- Gerrit server for open reviews
- Documentation
- TF-A Tests suite
- Trustedfirmware.org monthly project status updates
- Trustedfirmware.org board meeting minutes

arm

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