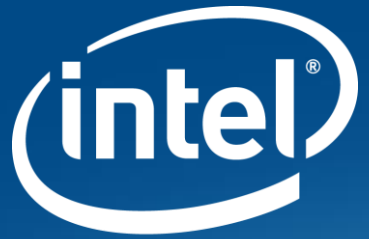


Software

TPM2: KERNEL DRIVER TO EVENT-DRIVEN APPLICATIONS

Jarkko Sakkinen <jarkko.sakkinen@intel.com>

Philip Tricca <philip.b.tricca@intel.com>

AGENDA

- What it is, what it does, why you should care
- Architecture / component model
- TPM2 Resource Management
- Kernel Driver
  - Status: Resource Management & Event log
- User Space Plumbing
  - APIs & Async I/O
  - Resource Management
- Aligning kernel & user space resource management

TRUSTED PLATFORM MODULE V2.0

What it is, what it does, why you should care

- Crypto co-processor
  - Key protection, generation, entropy / RNG & usage policy
  - Crypto functions: asymmetric, symmetric, hash, hmac
  - Algorithm agility: flexible support for new algorithms
  - Crypto decelerator: not fast
- Software measurements / “measured boot”
  - Tamper resistant (software) hash chain, “extend” operation == rolling hash
  - Rolling hash records software execution history
  - Useful for reporting on platform software state & policy decisions

TCG TPM2 SOFTWARE STACK: DESIGN

System API (SYS)

- 1:1 mapping to TPM2 commands
- No
  - file IO
  - crypto
  - heap

Enhanced SAPI (ESYS)

- Spec public
- No implementation yet
- Additional utility functions
- Provides Cryptographic functions for sessions
- No file IO
- Requires heap

Feature API (FAPI)

- Spec in draft form
- No implementation yet
- File IO
- Requires heap
- Must be able to do retries
- Context based state
- Must support static linking

User

TPM Command Transmission Interface (TCTI)

- Abstract command / response mechanism
- Decouple APIs driving TPM from command transport / IPC
- No crypto
- No heap, file I/O

TPM Access Broker and Resource Manager (TAB/RM)

- Power management
- Potentially no file IO – depends on power mgmt.
- Abstract Limitations of TPM Storage
- No crypto

TPM Device Driver

- Device Interface (CRB / polling)
- Pre-boot log handoff

Kernel

2017 TCG WORK

Refactoring existing specs, lots of progress on ESAPI

- Combined TCTI / SAPI spec separated into independent specs w/ new versions of each
- New API for converting (aka “marshalling”) between C types and byte-stream representation

Public Review open:

- TCTI v1.0: https://trustedcomputinggroup.org/wp-content/uploads/TSS_TCTI_v1.0_r04_Public-Review.pdf
- SAPI v1.1: https://trustedcomputinggroup.org/wp-content/uploads/TSS_SAPI_v1.1_r21_Public_Review.pdf
- Type Marshalling v1.0: https://trustedcomputinggroup.org/wp-content/uploads/TSS_Marshaling_v1.0_r03_Public-Review.pdf
- TPM2 Access Broker & Resource Manager v1.0: https://trustedcomputinggroup.org/wp-content/uploads/TSS-TAB-and-Resource-Manager-ver1.0-rev16_Public_Review.pdf

TPM2 RESOURCE MANAGEMENT

TPMs are very resource constrained

- Designed for low cost, promote adoption
- RAM on the order of “a few kilobytes”
- Typically able to load ~3 RSA 2048 keys simultaneously

Scarce resources must be shared

- TPM supports commands specific to object and session management:
  - ContextLoad, ContextSave & FlushContnext
- Resource Management: Saving & Loading contexts

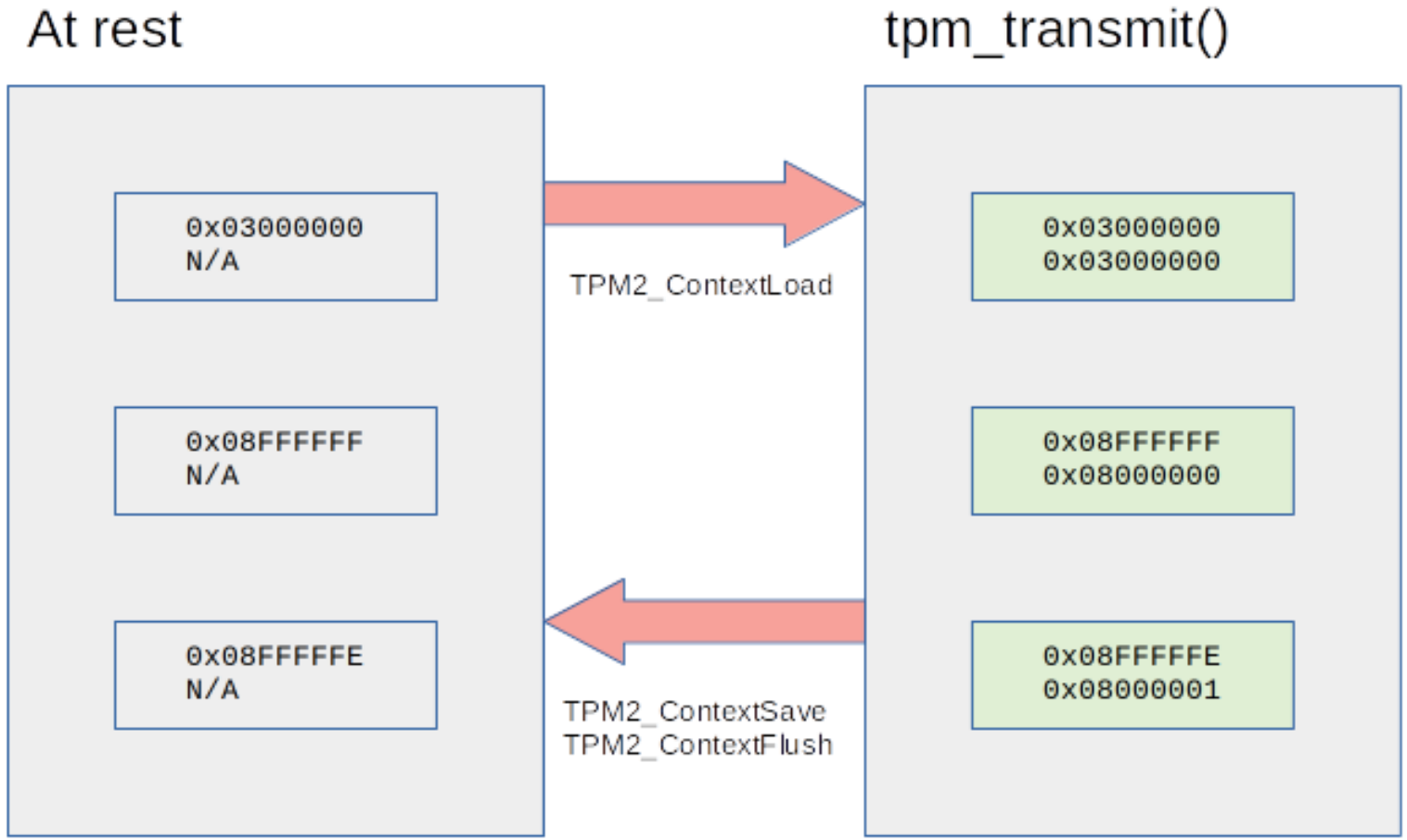
Isolation through Resource Management

- Associate objects (keys, session) with owner
- Prevent access by non-owner

NEW KERNEL FEATURES SINCE LSS 2016

TPM 2.0 resource manager
TPM 2.0 event log
ARM64 support for tpm_crb

IN-KERNEL RM: THE BASIC IDEA



IN-KERNEL RM: SWAPPING TRANSIENT CONTEXTS

A handle is linked to a context only as long as it is loaded.

- Assigned when context is loaded
- Reclaimed / recycled after Context is flushed

A context must be always explicitly flushed with TPM2_FlushContext.

- Saved contexts for transient objects remain resident in the TPM

Handles are “virtualized”

- Virtual handle space starting from 0x08FFFFFF
- Assigned in escending order and substitute them for commands and responses.

IN-KERNEL RM: SWAPPING SESSION CONTEXTS

Handle of a session never changes on its lifetime.

- Context may be saved
- Tracking info remains in TPM

When a session is swapped

- It is saved, not flushed
- Flush removes tracking info, session must be recreated to reload

When connection is closed

- Explicitly flush associated sessions
- TPM space is removed

SOFTWARE STACK ARCHITECTURE

Enhanced System API (ESAPI) & Feature API (FAPI):

- Specifications are currently available for public review, no implementation yet

System API (SAPI):

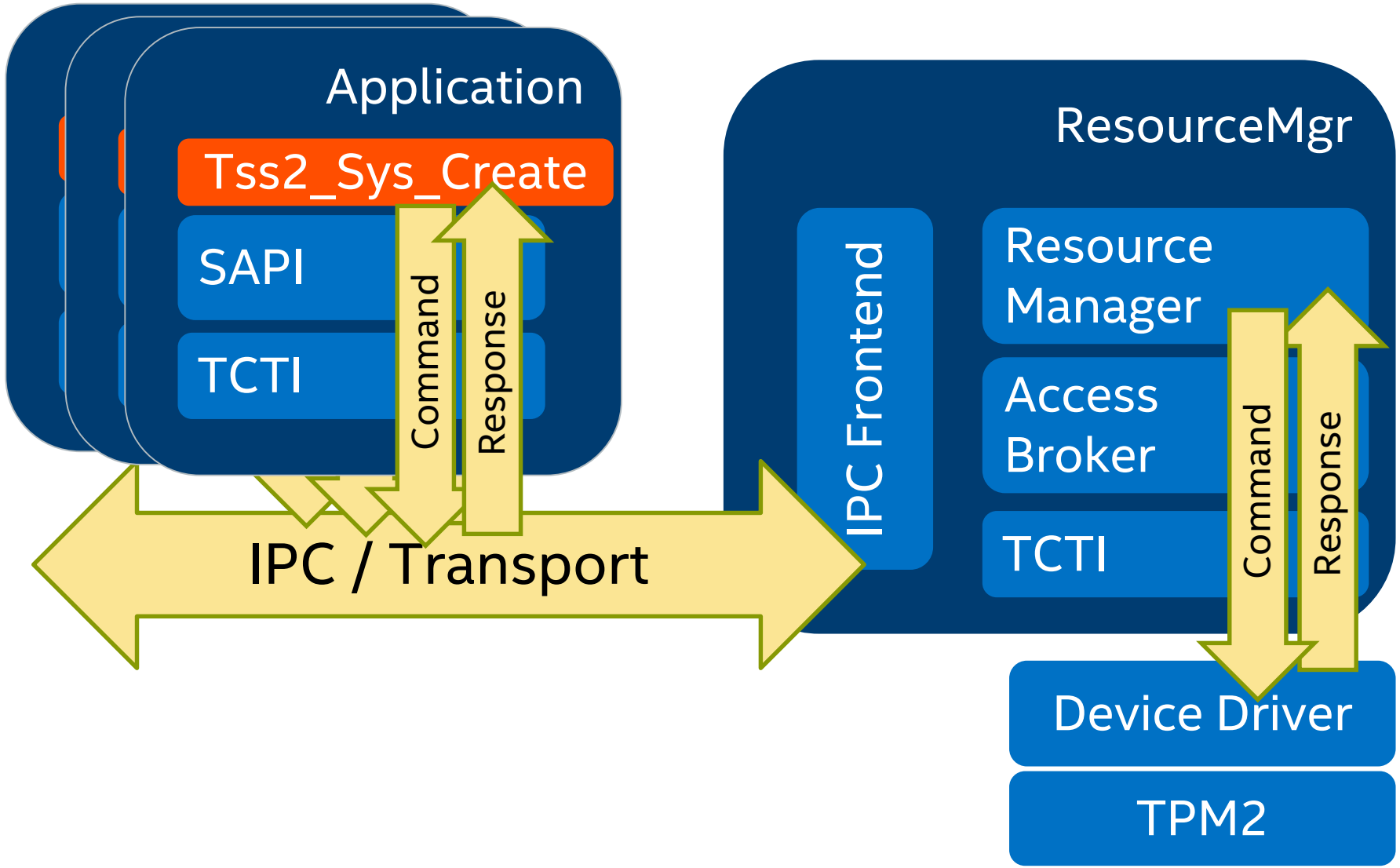
- 1:1 mapping to TPM2 commands
- No: file IO, crypto, heap
- Async & synchronous APIs

TPM2 Command Transmission Interface (TCTI):

- Decouple APIs driving TPM from command transport / IPC
- ‘receive’ function supports async w/ timeout / polling interface

TPM2 Access Broker & Resource Management (TAB/RM)

TPM2 TSS COMPONENTS



USERSPACE LIBRARIES: STATUS

https://github.com/01org/tpm2-tss

Progress since last LPC

- Establishing project structure / process
- Removed POC resource mgmt. daemon

Async I/O & event driven programming frameworks

- Support for async programming models
- Prototype code available for Glib / GIO using Gsource

Lots of interest in language bindings

- Rust bindings from Doug Goldstein @ StarLabs: https://crates.io/crates/tss-sapi
- Rumors of Python bindings but no OSS implementation yet

Currently in planning / next steps

- TCTI dlopen-able interface
- PKCS#11 module in planning
- UEFI & SGX TCTIs?

RESOURCE MANAGEMENT IN USER SPACE

https://github.com/01org/tpm2-abrmd

User space daemon integrated with Linux infrastructure (systemd, dbus)

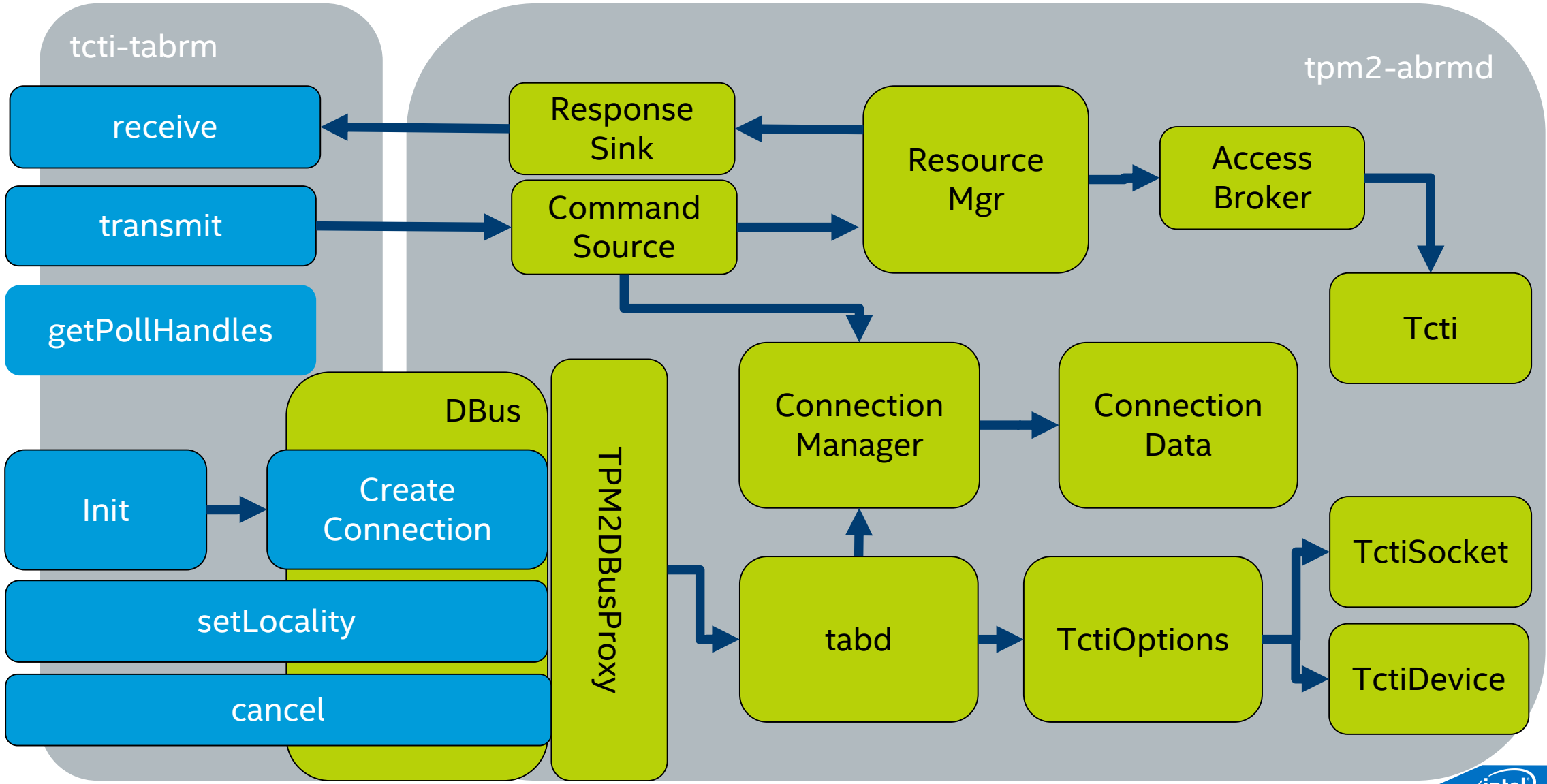
Necessary for supporting

- Existing customers on < 4.12 kernels
- Async I/O
- Remote connections: prototyping TCP / IP / TLS

Prototyping policy layers

- TPM command black list (per-user command sets?)
- Per-connection / process resource caps

TPM2-ABRMD: COMPONENT MODEL



ALIGNING KERNEL & USER-SPACE RM

Goal: get best of both worlds

Need to isolate user-space AND kernel space TPM objects

- Cannot be done from user-space
- Still not done in kernel RM but now possible

Reduces the need for IPC

- IPC becomes File I/O
- Kernel driver doesn't support async / poll
- Remote connections via TCP / IP & TLS belong in user-space

Policy requirements not well understood yet

- No policy interface in kernel
- Easy to prototype in user-space
- Prevent resource exhaustion / DoS