Arm reference solutions

Avinash Mehta

Sep 21, 2023

TOTAL COMPUTE:

1 Total Compute **3**

1.1 Total Compute: TC1-2021.08.17 3

1.2 Previous releases 19

Warning: This release is now considered obsolete and has been replaced by a newer TC release. Please refer to the latest TC release documentation for more details.

CHAPTER

ONE

TOTAL COMPUTE

Warning: This release is now considered obsolete and has been replaced by a newer TC release. Please refer to the latest TC release documentation for more details.

1.1 Total Compute: TC1-2021.08.17

Total Compute is an approach to moving beyond optimizing individual IP to take a system-level solution view of the SoC that puts use cases and experiences at the heart of the designs.

Total Compute focuses on optimizing Performance, Security, and Developer Access across Arm's IP, software, and tools. This means higher-performing, more immersive, and more secure experiences on devices coupled with an easier app and software development process.

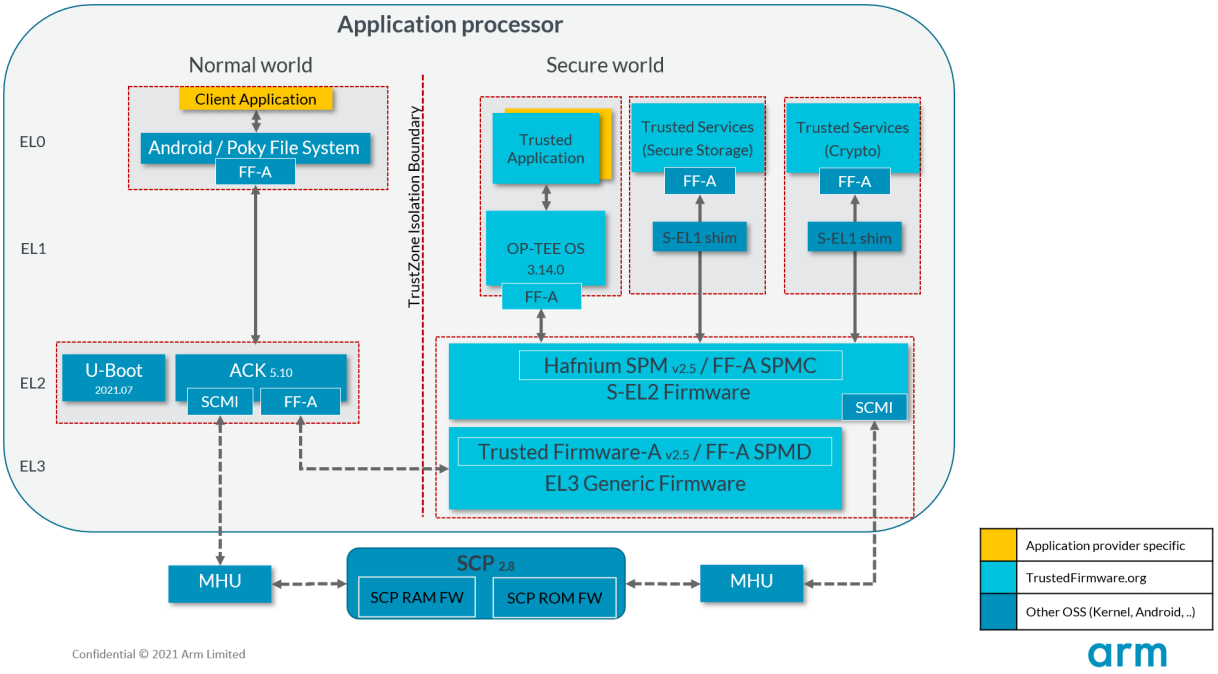
Warning: This release is now considered obsolete and has been replaced by a newer TC release. Please refer to the latest TC release documentation for more details.

1.1.1 Total Compute Platform Software Components

SCP Firmware

The System Control Processor (SCP) is a compute unit of Total Compute and is responsible for low-level system management. The SCP is a Cortex-M3 processor with a set of dedicated peripherals and interfaces that you can extend. SCP firmware supports:

1. Powerup sequence and system start-up
2. Initial hardware configuration
3. Clock management
4. Servicing power state requests from the OS Power Management (OSPM) software



SCP Boot ROM

SCP Boot ROM code is the first software that executes right after a cold reset or Power-on. It performs the following functions:

1. Sets up generic timer, UART console and clocks
2. Initializes the Coherent Interconnect
3. Powers ON primary AP CPU
4. Loads SCP Runtime Firmware

SCP Runtime Firmware

SCP runtime code starts execution after TF-A BL2 has authenticated and copied it from flash. It performs the following functions:

1. Responds to SCMI messages via MHUv2 for CPU power control and DVFS
2. Power Domain management
3. Clock management

Secure Software

Secure software/firmware is a trusted software component that runs in the AP secure world. It mainly consists of AP firmware, Secure Partition Manager and Secure Partitions (OP-TEE, Trusted Services).

AP firmware

The AP firmware consists of the code that is required to boot Total Compute platform up the point where the OS execution starts. This firmware performs architecture and platform initialization. It also loads and initializes secure world images like Secure partition manager and Trusted OS.

Trusted Firmware-A (TF-A) BL1

AP Trusted ROM contains an on-chip trusted ROM that runs the boot code on Total Compute platform. BL1 performs minimal architectural initialization (like exception vectors, CPU initialization) and Platform initialization. It loads the BL2 image and passes control to it.

Trusted Firmware-A (TF-A) BL2

BL2 runs at S-EL1 and performs architectural initialization required for subsequent stages of TF-A and normal world software. It configures the TrustZone Controller and carves out memory region in DRAM for secure and non-secure use. BL2 loads below images:

1. SCP BL2 image
2. EL3 Runtime Software (BL31 image)
3. Secure Partition Manager (BL32 image)
4. Non-Trusted firmware - U-boot (BL33 image)
5. Secure Partitions images (OP-TEE and Trusted Services)

Trusted Firmware-A (TF-A) BL31

BL2 loads EL3 Runtime Software (BL31) and BL1 passes control to BL31 at EL3. In Total Compute BL31 runs at trusted SRAM. It provides below mentioned runtime services:

1. Power State Coordination Interface (PSCI)
2. Secure Monitor framework
3. Secure Partition Manager Dispatcher

Secure Partition Manager

Total Compute enables FEAT S-EL2 architectural extension, and it uses Hafnium as Secure Partition Manager Core (SPMC). BL32 option in TF-A is re-purposed to specify the SPMC image. The SPMC component runs at S-EL2 exception level.

Secure Partitions

Software image isolated using SPM is Secure Partition. Total Compute enables OP-TEE and Trusted Services (crypto, secure storage) as Secure Partitions.

OP-TEE

OP-TEE Trusted OS is virtualized using Hafnium at S-EL2. OP-TEE OS for Total Compute is built with FFA and SEL2 SPMC support. This enables OP-TEE as a Secure Partition running in an isolated address space managed by Hafnium. The OP-TEE kernel runs at S-EL1 with Trusted applications running at S-EL0.

Trusted Services

Trusted Services like Crypto Service and Secure Storage runs as S-EL0 Secure Partitions using a Shim layer at S-EL1. Crypto Service along with S-EL1 Shim layer is built as a single image. The Shim layer forwards FF-A calls from S-EL0 to S-EL2.

U-Boot

TF-A BL31 passes execution control to U-boot bootloader (BL33). U-boot in Total Compute has support for multiple image formats:

1. FitImage format: this contains the Linux kernel and poky ramdisk which are authenticated and loaded in their respective positions in DRAM and execution is handed off to the kernel.
2. Android boot image: This contains the Linux kernel and Android ramdisk. If using Android Verified Boot (AVB) boot.img is loaded from MMC to DRAM, authenticated and then execution is handed off to the kernel.

Kernel

Linux Kernel in Total Compute contains the subsystem-specific features that demonstrate the capabilities of Total Compute. Apart from default configuration, it enables:

1. Arm MHUv2 controller driver
2. Arm FF-A driver
3. OP-TEE driver with FF-A Transport Support
4. Arm FF-A user space interface driver

Android

Total Compute has support for Android Open-Source Project (AOSP), which contains the Android framework, Native Libraries, Android Runtime and the Hardware Abstraction Layers (HALs) for Android Operating system. The Total Compute device profile defines the required variables for Android such as partition size and product packages and has support for 2 different configurations of Android:

1. Nano: This is a stripped-down version to provide the bare minimum for Android Runtime and boot Android to console. It does not have Android UI support.
2. Software rendering: This profile has support for Android UI and boots Android to home screen. It uses SwiftShader to achieve this. Swiftshader is a CPU base implementation of the Vulkan graphics API by Google.

Warning: This release is now considered obsolete and has been replaced by a newer TC release. Please refer to the latest TC release documentation for more details.

1.1.2 Instructions: Obtaining Total Compute software deliverables

- To build the TC1 software stack please refer to *user-guide*
- For the list of changes and features added please refer to *change-log*
- For further details on the latest release and features please refer to *release_notes*

1.1.3 TC Software Stack Overview

The TC1 software consists of firmware, kernel and file system components that can run on the associated FVP. Following are the Software components:

1. SCP firmware – System initialization, Clock and Power control
2. AP firmware – Trusted Firmware-A (TF-A)
3. Secure Partition Manager
4. Secure Partitions
    - OP-TEE Trusted OS
    - Trusted Services with Shim layer
5. U-Boot – loads and verifies the fitImage for poky boot, containing kernel and filesystem or boot Image for Android Verified Boot, containing kernel and ramdisk.
6. Kernel – supports the following hardware features
    - Mailbox hardware unit
    - PAC/MTE/BTI features
7. Android
    - Supports PAC/MTE/BTI features

Total Compute Platform Software Components

Warning: This release is now considered obsolete and has been replaced by a newer TC release. Please refer to the latest TC release documentation for more details.

1.1.4 User Guide

Contents

- *User Guide*
  - *Notice*
  - *Prerequisites*
  - *Syncing and building the source code*
    - *Syncing code*
    - *Board Support Package build*
    - *Android OS build*
  - *Provided components*
    - *Firmware Components*
      - *Trusted Firmware-A*
      - *System Control Processor (SCP)*
      - *U-Boot*
      - *Hafnium*
      - *OP-TEE*
      - *S-EL0 trusted-services*
      - *Linux*
    - *Distributions*
      - *Poky Linux distro*
      - *Android*
    - *Run scripts*
  - *Obtaining the TC1 FVP*
  - *Running the software on FVP*
    - *Running Poky*
    - *Running Android*

Notice

The Total Compute 2021 (TC1) software stack uses the Yocto project to build a Board Support Package (BSP) and a choice of Poky Linux distribution or Android userspace. The Yocto project uses Bitbake to build the software stack.

Prerequisites

These instructions assume that:

- Your host PC is running Ubuntu Linux 18.04 LTS.
- You are running the provided scripts in a `bash` shell environment.

The following utilities must be available on your host PC:

- chrpath
- compression library
- diffstat
- gawk
- makeinfo
- openssl headers
- pip
- repo

To resolve these dependencies, run:

```
sudo apt-get update
sudo apt-get install chrpath gawk texinfo libssl-dev diffstat wget git-core unzip gcc-
↪multilib \
 build-essential socat cpio python python3 python3-pip python3-pexpect xz-utils␣
↪debianutils \
 iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm git-
↪lfs openssl \
 curl lib32ncurses5-dev libz-dev python-pip u-boot-tools m4 zip
```

To get the latest repo tool from google, run the following commands:

```
mkdir -p ~/bin
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
export PATH=~/bin:$PATH
```

If syncing and building android, the minimum requirements for the host machine can be found at https://source.android.com/se

- At least 250GB of free disk space to check out the code and an extra 150 GB to build it. If you conduct multiple builds, you need additional space.
- At least 16 GB of available RAM/swap.
- Git configured properly using "git config" otherwise it may throw error while fetching the code.

Syncing and building the source code

There are two distros supported in the TC1 software stack: poky (a minimal distro containing busybox) and android.

To sync code for poky, please follow the steps in "Syncing code" section for BSP only. To sync code for android, please follow the steps for syncing both BSP and Android.

To build the required binaries for poky, please follow the steps in "Board Support Package build" section only. To build the binaries for Android, please follow the steps in both "Board Support Package build" and "Android OS build" sections.

Syncing code

Create a new folder that will be your workspace, which will henceforth be referred to as <tc1_workspace> in these instructions.

```
mkdir <tc1_workspace>
cd <tc1_workspace>
export TC1_RELEASE=refs/tags/TC1-2021.08.17
```

To sync BSP only without Android, run the repo command.

```
repo init -u https://gitlab.arm.com/arm-reference-solutions/arm-reference-solutions-
↪manifest -m tc1.xml -b ${TC1_RELEASE} -g bsp
repo sync -j$(nproc)
```

To sync both the BSP and Android, run the repo command.

```
repo init -u https://gitlab.arm.com/arm-reference-solutions/arm-reference-solutions-
↪manifest -m tc1.xml -b ${TC1_RELEASE} -g android
repo sync -j$(nproc)
```

Board Support Package build

```
cd <tc1_workspace>/bsp
export DISTRO="poky"
export MACHINE="tc1"
source setup-environment
bitbake tc-artifacts-image
```

The initial clean build will be lengthy, given that all host utilities are to be built as well as the target images. This includes host programs (python, cmake, etc.) and the required toolchain(s).

Once the build is successful, all images will be placed in the <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1 directory.

Note that the BSP includes the Poky Linux distribution, which offers BusyBox-like utilities.

Android OS build

Two profiles are supported:

1. tc1_swr : This supports Android display with swiftshader (software rendering).
2. tc1_nano : This supports headless Android and provides a good runtime environment for testing shell-based applications.

The android images can be built with or without authentication enabled using Android Verified Boot(AVB). AVB build is done in userdebug mode and takes a longer time to boot as the images are verified.

The `build-scripts/tc1/build_android.sh` script in `<tc1_workspace>/android` will patch and build android. This can be passed 2 parameters, `-d` for deciding which profile to build and `-a` for enabling AVB. The following command shows the help menu for the script:

```
cd <tc1_workspace>/android
./build-scripts/tc1/build_android.sh  -h
Incorrect script use, call script as:
<path to build_android.sh> [OPTIONS]
OPTIONS:
-d, --distro                          distro version, values supported [android-nano,␣
↪android-swr]
-a, --avb                             [OPTIONAL] avb boot, values supported [true,␣
↪false], DEFAULT: false
```

The `--avb` option does not influence the way the system boots rather it adds an optional sanity check on the prerequisite images.

As an example, to build android with software rendering and AVB enabled, run the command:

```
./build-scripts/tc1/build_android.sh -d android-swr -a true
```

To build headless android without AVB, run the command:

```
./build-scripts/tc1/build_android.sh -d android-nano
```

Android based stack takes considerable time to build, so start the build and go grab a cup of coffee!

Provided components

Within the Yocto project, each component included in the TC1 software stack is specified as a Bitbake recipe. The TC1 recipes are located at `<tc1_workspace>/bsp/layers/meta-arm/`.

Yocto allows modifying the fetched source code of each recipe component in the workspace, by applying patches. This is however not a convenient approach for developers, since creating patches and updating recipes is time-consuming. To make that easier, Yocto provides the devtool utility. Devtool creates a new workspace, in which you can edit the fetched source code and bake images with the modifications

```
cd <tc1_workspace>/bsp
MACHINE=tc1
DISTRO=poky
. ./conf/setup-environment

# create a workspace for a given recipe component
# recipe-component-name can be of:
```

(continues on next page)

(continued from previous page)

```
# trusted-firmware-a / scp-firmware / u-boot / linux-arm64-ack
devtool modify <recipe-component-name>

# This creates a new workspace for recipe-component-name and fetches source code
# into "<tc1_workspace>/build-poky/workspace/sources/{trusted-firmware-a,scp-firmware,u-
↪boot,linux-arm64-ack}"
# edit the source code in the newly created workspace
# build images with changes on workspace
# recipe-component-name can be of: trusted-firmware-a / scp-firmware / u-boot / linux-
↪arm64-ack
bitbake <recipe-component-name>
```

Firmware Components

Trusted Firmware-A

Based on Trusted Firmware-A

Recipe	<tc1_workspace>/bsp/layers/meta-arm/meta-arm-bsp/recipes-bsp/trusted-firmware-a/trusted-firmware-a-tc1.inc
Files	• <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/bl1-tc1.bin • <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/fip-tc1.bin

System Control Processor (SCP)

Based on SCP Firmware

Recipe	<tc1_workspace>/bsp/layers/meta-arm/meta-arm-bsp/recipes-bsp/scp-firmware/scp-firmware-tc1.inc
Files	• <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/scp_ramfw.bin • <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/scp_romfw.bin

U-Boot

Based on U-Boot gitlab

Recipe	<tc1_workspace>/bsp/layers/meta-arm/meta-arm-bsp/recipes-bsp/u-boot/u-boot-tc1.inc
Files	• <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/u-boot.bin

Hafnium

Based on Hafnium

Recipe	<tc1_workspace>/bsp/layers/meta-arm/meta-arm-bsp/recipes-bsp/hafnium/hafnium-tc1.inc
Files	• <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/hafnium.bin

OP-TEE

Based on OP-TEE

Recipe	<tc1_workspace>/bsp/layers/meta-arm/meta-arm-bsp/recipes-security/optee/optee-os-tc1.inc
Files	• <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/optee/tee-pager_v2.bin

S-EL0 trusted-services

Based on Trusted Services

Recipe	<tc1_workspace>/bsp/layers/meta-tc/recipes-security/trusted-services/secure-partitions_git.bb
Files	• <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/firmware/crypto-sp.bin • <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/firmware/secure-storage.bin

Linux

The recipe responsible for building a 5.10 version of the Android Common kernel (ACK).

Recipe	<tc1_workspace>/bsp/layers/meta-arm/meta-arm-bsp/recipes-kernel/linux/linux-arm-platforms.inc
Files	• <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/Image

Distributions

Poky Linux distro

The layer is based on the poky Linux distribution. The provided distribution is based on BusyBox and built using glibc.

Recipe	<tc1_workspace>/bsp/layers/openembedded-core/meta/recipes-core/images/core-image-minimal.bb
Files	• <tc1_workspace>/bsp/build-poky/tmp-poky/deploy/images/tc1/fitImage-core-image-minimal-tc1-tc1

Android

Android Master (as on May21) is supported in this release with device profiles suitable for TC1 machine configuration. Android is built as a separate project and then booted with the BSP built by Yocto.

Run scripts

Within the `<tc1_workspace>/bsp/run-scripts/` are several convenience functions for testing the software stack. Usage descriptions for the various scripts are provided in the following sections.

Obtaining the TC1 FVP

The TC1 FVP is available to partners for build and run on Linux host environments. Please contact Arm to have access (support@arm.com).

Running the software on FVP

A Fixed Virtual Platform (FVP) of the TC1 platform must be available to run the included run scripts.

The run-scripts structure is as follows:

```
run-scripts
|--tc1
   |--run_model.sh
   |-- ...
```

Ensure that all dependencies are met by running the FVP: `./path/to/FVP_TC1`. You should see the FVP launch, presenting a graphical interface showing information about the current state of the FVP.

The `run_model.sh` script in `<tc1_workspace>/bsp/run-scripts/tc1` will launch the FVP, providing the previously built images as arguments. Run the `run_model.sh` script:

```
./run_model.sh
Incorrect script use, call script as:
<path_to_run_model.sh> [OPTIONS]
OPTIONS:
-m, --model                     path to model
-d, --distro                    distro version, values supported [poky, android-nano,␣
↪android-swr]
-a, --avb                       [OPTIONAL] avb boot, values supported [true, false],␣
↪DEFAULT: false
-t, --tap-interface             [OPTIONAL] enable TAP interface
-e, --extra-model-params        [OPTIONAL] extra model parameters
```

Running Poky

```
./run-scripts/tc1/run_model.sh -m <model binary path> -d poky
```

Running Android

```
If using an android distro, export ANDROID_PRODUCT_OUT variable to point to android out␣
↪directory
for eg. ANDROID_PRODUCT_OUT=<tc1_workspace>/android/out/target/product/tc_swr

For running android with AVB disabled:
 ./run-scripts/tc1/run_model.sh -m <model binary path> -d android-swr
 OR
 ./run-scripts/tc1/run_model.sh -m <model binary path> -d android-nano

For running android with AVB enabled:
 ./run-scripts/tc1/run_model.sh -m <model binary path> -d android-swr -a true
 OR
 ./run-scripts/tc1/run_model.sh -m <model binary path> -d android-nano -a true
```

When the script is run, two terminal instances will be launched. terminal_s0 used for the SCP, TF-A, OP-TEE core logs and terminal_s1 used by TF-A early boot, Hafnium, U-boot and Linux.

Once the FVP is running, the SCP will be the first to boot, bringing the AP out of reset. The AP will start booting from its ROM and then proceed to boot Trusted Firmware-A, Hafnium, OP-TEE then U-Boot, and then Linux and Poky/Android.

When booting Poky the model will boot Linux and present a login prompt. Login using the username `root`. You may need to hit Enter for the prompt to appear.

The OP-TEE and Trusted Services are initialized on both the Android and Poky distribution. But the functionality of OP-TEE and core set of trusted services such as cryptography and secure storage can be invoked only on Poky distribution. For OP-TEE, the TEE sanity test suite can be run using command `xtest`. For Trusted Services, run command `ts-service-test` for Service API level tests and run command `ts-demo` for the demonstration client application.

Warning: This release is now considered obsolete and has been replaced by a newer TC release. Please refer to the latest TC release documentation for more details.

1.1.5 Release notes - 2021.08.17

Contents

- *Release notes - 2021.08.17*
  - *Release tag*
  - *Components*
  - *Hardware Features*
  - *Software Features*
  - *Platform Support*
  - *Known issues or Limitations*
  - *Support*

Release tag

The manifest tag for this release is TC1-2021.08.17

Components

The following is a summary of the key software features of the release:

- Yocto based BSP build supporting Android and Poky distro.
- Trusted firmware-A for secure boot.
- System control processor(SCP) firmware for programming the interconnect, doing power control etc.
- U-Boot bootloader.

- Hafnium for S-EL2 Secure Partition Manager core.
- OP-TEE for Trusted Execution Environment (TEE).
- Crypto and Storage Trusted Services running at S-EL0.

Hardware Features

- Booker CI with Memory Tagging Unit(MTU) support driver in SCP firmware.
- GIC Clayton Initialization in Trusted Firmware-A.
- Mali-D71 DPU and virtual encoder support for display in Linux.
- MHUv2 Driver for SCP and AP communication.
- UARTs, Timers, Flash, PIK, Clock drivers.
- PL180 MMC.
- DynamIQ Shared Unit (DSU) with 8 cores. 1 Makalu ELP + 3 Makalu + 4 Cortex-A510 (R1) cores configuration.

Software Features

- Poky Distribution support.
- Android AOSP Support (May21).
- Android Common Kernel 5.10 with PAC/BTI/MTE
- Trusted Firmware-A & Hafnium v2.5
- OP-TEE 3.14.0
- Support secure boot based on TBBR specification https://developer.arm.com/documentation/den0006/latest
- System Control Processor (SCP) firmware v2.8
- Build system based on Yocto master
- U-Boot bootloader v2021.07
- Power management features: cpufreq and cpuidle.
- SCMI (System Control and Management Interface) support.
- Verified u-boot for authenticating fit image (containing kernel + ramdisk) during poky boot.
- Android Verified Boot (AVB) for authenticating boot and system image during Android boot.
- Software rendering on Android with DRM Hardware Composer offloading composition to Mali D71 DPU.
- Hafnium as Secure Partition Manager (SPM) at S-EL2.
- OP-TEE as Secure Partition at S-EL1, managed by S-EL2 SPMC (Hafnium)
- Arm FF-A driver and FF-A Transport support for OP-TEE driver in Android Common Kernel.
- OP-TEE Support in Poky distribution. This includes OP-TEE client and OP-TEE test suite.
- Crypto and Storage Trusted Services running at S-EL0.
- Trusted Services test suite added to poky distribution.
- Shim Layer at S-EL1 running on top of S-EL2 SPMC (Hafnium) used by Trusted Services running in S-EL0.

Platform Support

- This Software release is tested on TC1 Fast Model platform (FVP). - Supported Fast model version for this release is 11.15.20

Known issues or Limitations

1. At the U-Boot prompt press enter and type "boot" to continue booting else wait for ~15 secs for boot to continue automatically. This is because of the time difference in CPU frequency and FVP operating frequency.
2. OP-TEE test suite xtest, fails for Storage concurrency test case. This issue is under investigation.

Support

For support email: support-arch@arm.com

Warning: This release is now considered obsolete and has been replaced by a newer TC release. Please refer to the latest TC release documentation for more details.

1.1.6 Change Log

Contents

- *Change Log*
  - *Version 2021.08.17*
    - *Features added*

This document contains a summary of the new features, changes and fixes in each release of TC1 software stack.

Version 2021.08.17

Features added

- Memory Tagging Extension (MTE)
- Pointer Authentication Code (PAC)
- Branch Target Identification (BTI)
- Android AOSP to master (May21)
- Android Common Kernel to v5.10
- Trusted Firmware-A & Hafnium to v2.5
- OP-TEE to v3.14.0
- SCP firmware to v2.8

- U-boot to v2021.07
- Yocto to master

Warning: This release is now considered obsolete and has been replaced by a newer TC release. Please refer to the latest TC release documentation for more details.

1.2 Previous releases

This web page provides a list of all the TotalCompute Software Stack releases, cataloged by major version, which can be used for easy historical reference.

1.2.1 TC1 release tags

TC1-2021.08.17

1.2.2 TC0 release tags

TC0-2022.02.25

TC0-2021.07.31

TC0-2021.04.23

TC0-2021.02.09