

Running ACS on Arm's Neoverse Reference Design Platforms

Pranav Madhu
Software Engineer, Arm Ltd.
23-Sep-2020



Contents

- What is ServerReady & why is it required?
- Overview of ACS, SBSA & SBBR test suites.
- Getting ACS test suite.
- Running ACS SBSA & SBBR test suites on Neoverse Reference Design.

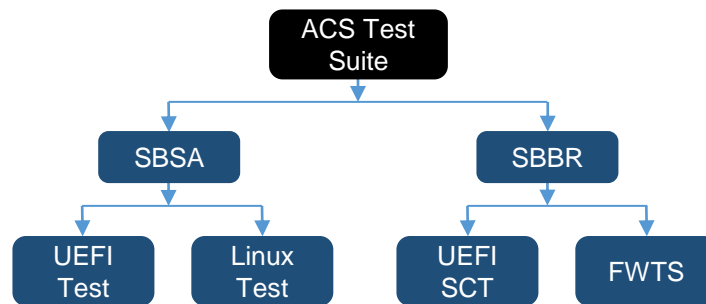
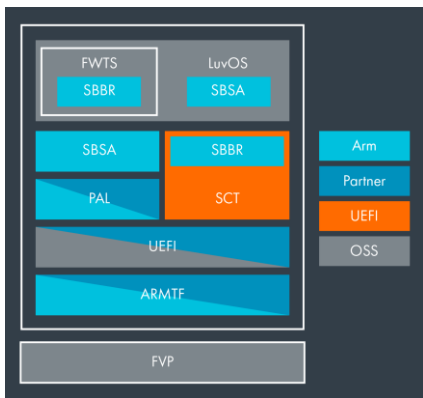
Introduction to ServerReady

- Compliance program, put around standard approach to ARM server.
- Provides a solution for servers that “just works”.
- Has a set of requirement for hardware & firmware.
 - Server Base System Architecture (SBSA): Defines hardware requirements.
 - Server Base Boot Requirements (SBBR): Defines firmware requirements.
- Certifies a platform is good to run with standard OS.



Architectural Compliance Suite (ACS)

- Arm have the ServerReady specification, and this spec have a set of requirements.
- ACS is a set of tests around each one of these requirement.
- Objective is to provide a seamless out of the box experience with booting standard OS.



- ServerReady requirement can be classified as:
 - Hardware requirement
 - Firmware requirement
- Covered in ACS with:
 - SBSA tests
 - SBRR tests

Server Base System Architecture (SBSA)

- SBSA v6 specification:
https://developer.arm.com/documentation/den0029/latest?_ga=2.2681840.1567254736.1582536887-1048184626.1580228297
- Define hardware requirement
- Classify hardware into different level.
 - Level 3 – Level 6
- Tests for
 - CPU properties
 - System Components (GIC, IOMMU, WatchDog & Timers)
 - PCIe Integration
- UEFI shell SBSA tests
- OS SBSA tests

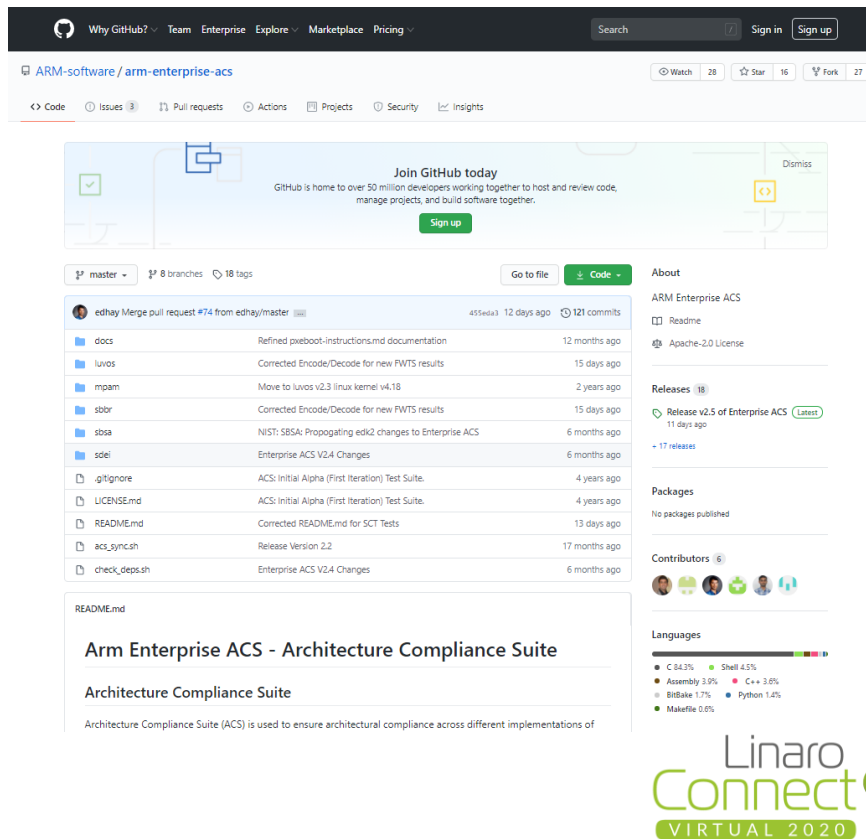
SBSA Level	PE 'A' profile	SMMU	GIC
3	v8.0	v2/v3	v3.0
4	v8.3	v3.0	v3.0
5	v8.4	v3.2	v3.0
6	v8.5	v3.2	v3.0

Server Base Boot Requirements (SBBR)

- SBBR v1.2 specification:
https://developer.arm.com/documentation/den0044/latest?_ga=2.259187914.1567254736.1582536887-1048184626.1580228297
- Define firmware requirements
- Tests for
 - UEFI (UEFI SCT)
 - ACPI (FWTS)
 - SMBIOS (FWTS)
- UEFI shell SBBR test / UEFI-SCT (Self Certification Test)
- OS SBBR test / FWTS (Firmware Test Suite)

Getting ACS Test Suite

- Hosted in GitHub and are open source (Apachev2).
<https://github.com/ARM-software/arm-enterprise-acs>
- Sync the code using 'acs_sync.sh'
- Build the suite from source by running 'luvos/scripts/build.sh' to generate LUV disk image.
<https://github.com/ARM-software/arm-enterprise-acs#acs-build-steps>



The screenshot shows the GitHub repository page for ARM Enterprise ACS. The repository is located at `ARM-software / arm-enterprise-acs`. The page displays a list of files and folders, including `docs`, `luvos`, `mpam`, `sbbr`, `sbbs`, `soel`, `.gitignore`, `LICENSE.md`, `README.md`, `acs_sync.sh`, and `check_deps.sh`. The repository is currently on the `master` branch, with 8 branches and 18 tags. The repository has 121 commits and was last updated 12 days ago. The repository is licensed under Apache-2.0 License. The repository also has 17 releases, with the latest release being `Release v2.5 of Enterprise ACS` (Latest) from 11 days ago. The repository also has 6 contributors and a language distribution chart showing the following languages and their percentages: C (84.3%), Shell (4.5%), Assembly (3.9%), C++ (3.6%), BitBake (1.7%), Python (1.4%), and Makefile (0.6%).

Arm Enterprise ACS - Architecture Compliance Suite

Architecture Compliance Suite

Architecture Compliance Suite (ACS) is used to ensure architectural compliance across different implementations of

Linaro Connect
VIRTUAL 2020

ACS Disk Format

- ACS test suite build generate LUV disk image 'luv-live-image-gpt.img'
- Two Partitions
 - First Partition - for storing test logs
 - Second Partition - contains bootable & test binaries
- Boot the device with ACS disk visible to UEFI BDS stage

```
Disk luv-live-image-gpt.img: 272 MiB, 285212672 bytes, 557056 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 4A13C35F-F8EB-4AF5-BC12-2D225E770A37

Device                Start      End Sectors  Size Type
luv-live-image-gpt.img1  2048 264191 262144 128M Microsoft basic data
luv-live-image-gpt.img2 264192 555007 290816 142M Microsoft basic data
```

luv-live-image-gpt.img disk format

Running SBRR

UEFI-SCT (built on top of UEFI-SCT Framework)

- Boot the system
- Choose 'sbbr/sbsa' from GRUB.
- UEFI startup script launch SCT test automatically.
- To run manually from UEFI shell:
 1. After GRUB, Press 'Esc' to get UEFI shell
 2. Run 'InstallAARCH64.efi', Install SCT to FS2
`FS3:\EFI\BOOT\sbbr> InstallAARCH64.efi`
 3. Run UEFI-SCT test:
`FS2:\SCT> SCT.efi -a -v`

Running FWTS tests

- Boot the system
- Form GRUB choose 'luv'
- FWTS tests and OS context SBSA tests are run automatically on LuvOS.

Running UEFI-SCT (Automatically on Boot)

```
GNU GRUB version 2.02

luv
*sbbr/sbsa
```

```
FS2: Alias(s):HD2b::BLK1:
VenHw(5A96CDDC-6116-4929-B701-3AC2FB1CE228)/HD(1,GPT,3A557DEE-0318-432
5-B114-D24CD953554E,0x800,0x40000)
FS3: Alias(s):HD2c::BLK2:
VenHw(5A96CDDC-6116-4929-B701-3AC2FB1CE228)/HD(2,GPT,9C6D2C41-0006-408
7-BAB2-6E64888E3867,0x40800,0x47000)
BLK4: Alias(s):
VenHw(93E34C7E-B50E-11DF-9223-2443DFD72085,01)
BLK0: Alias(s):
VenHw(5A96CDDC-6116-4929-B701-3AC2FB1CE228)
Press ESC in 5 seconds to skip startup.nsh or any other key to continue.
Shell> echo -off
Press any key to stop the EFI SCT running
add-symbol-file /data_sdb/pr: /workspace/acs/2.5/arm-enterprise-acs/luv/buil
d/tmp/work/aarch64-oe-linux/sbbr/v1.1+gitAUTOINC+b558bad254-r0/edk2/Build/SbbrSc
t/DEBUG_GCC49/AARCH64/SctPkg/Application/StallForKey/StallForKey/DEBUG/StallForK
ey.dll 0xF99380>Loading driver at 0x000F9937000 EntryPoint=0x000F9938000 StallFor
Key.efi
Press any key within 7 seconds
```

- Pressing any key will skip SCT test & continue with remaining ACS test cases

Running UEFI-SCT (Manually from UEFI Shell)

GNU GRUB version 2.02

1

```
luv
*sbbr/sbsa
```

2

Press **ESC** in 5 seconds to skip **startup.nsh** or any other key to continue.

```
Shell> fs3:
FS3:\> cd EFI\BOOT\sbbbr
FS3:\EFI\BOOT\sbbbr> ls
Directory of: FS3:\EFI\BOOT\sbbbr\
09/01/2020 06:17 <DIR>          4,096
09/01/2020 06:17 <DIR>          4,096
09/01/2020 06:17              53,248 InstallAARCH64.efi
09/01/2020 06:17              1,423 sbbbr.nsh
09/01/2020 06:17              818 SbbbrSctStartup.nsh
09/01/2020 06:17 <DIR>          4,096 ARCH64
09/01/2020 06:17              1,505 SctStartup.nsh
4 File(s)          56,994 bytes
3 Dir(s)
```

```
FS3:\EFI\BOOT\sbbbr> InstallAARCH64.efi
```

```
MemoryMapped(0xB,0xE0000000,0xE01FFFFFF)
```

FS0: Alias(s):F0:

```
Fv(89CC2AB6-B847-475F-93E2-819603C3D15A)
```

FS4: Alias(s):F3::BLK3:

```
VenHw(93E34C7E-B50E-11DF-9223-2443DFD72085,00)
```

FS2: Alias(s):HD2b::BLK1:

```
VenHw(5A96CDCD-6116-4929-B701-3AC2FB1CE228)/HD(1,GPT,3A957DEE-0318-4325-B114-D24CD953554E,0x800,0x40000)
```

FS3: Alias(s):HD2c::BLK2:

```
VenHw(5A96CDCD-6116-4929-B701-3AC2FB1CE228)/HD(2,GPT,9C6D2C41-0006-4087-BAB2-6E64888E3867,0x40800,0x47000)
```

BLK4: Alias(s):

```
VenHw(93E34C7E-B50E-11DF-9223-2443DFD72085,01)
```

BLK0: Alias(s):

```
VenHw(5A96CDCD-6116-4929-B701-3AC2FB1CE228)
```

```
Error: Image at 000F79A0000 start failed: Not Found
remove-symbol-file /work/git/edk2/Build/Shell/RELEASE_GCC5/AARCH64/ShellPkg/AppI
lication/Shell/Shell/DEBUG/Shell.d11 0xF79A0240
```

```
1: FS1: (Free Space: 0 MB)
2: FS2: (Free Space: 123 MB)
3: FS3: (Free Space: 2 MB)
4: FS4: (Free Space: 63 MB)
Space Required: 100 MB
Input index of destination FS. 'q' to exit:2
```

```
FS3:\EFI\BOOT\sbbbr> fs2:
```

```
FS2:\> cd SCT
```

```
FS2:\SCT> SCT.efi -a -v
```

3

4

Running FWTS

```
GNU GRUB version 2.02
```

```
*luv  
sbbv/sbsa
```

```
Welcome to Linux UEFI Validation Distribution 2.3
```

```
Running tests...  
cat: /sys/class/dmi/id/product_uuid: No such file or directory  
Running tests...  
Running efivarfs-test ...  
[ 4,340000] [-] INFO  
Running fwts ...  
[ 4,410000] [-] Arm  
Running kernel-efi-warnings ...  
[ 5,630000] [-] kernel-efi-warnings  
[ 5,640000] [+] EFI_illegal_accesses_test... passed  
[ 5,640000] [+] !!!Kernel_FW_BUG... 1 failures  
[ 5,660000] [+] !!!Kernel_FW_WARN... 1 failures  
[ 5,670000] [+] Kernel_FW_INFO... passed  
[ 5,680000] [+] Kernel_MEM_ATTR_TYPE... passed  
[ 5,690000] [+] Kernel_MEM_ATTR_ATTR... passed  
[ 5,700000] [+] Kernel_MEM_ATTR_ALGN... passed  
[ 5,710000] [+] Kernel_MEM_ATTR_DVLP... passed  
[ 5,710000] [+] Kernel_MEM_ATTR_TYMA... passed  
[ 5,720000] [+] Kernel_MEM_ATTR_MISS... passed  
[ 5,730000] [+] Kernel_MCE... skipped
```

Running SBSA

UEFI Test

- Choose 'sbbr/sbsa' from GRUB.
- UEFI startup script launch tests automatically.
- Run manually from UEFI shell:
 1. After GRUB, Press 'Esc' to get UEFI shell
 2. Run 'sbsa.nsh', to start the test and to save result from console.

```
FS2:>FS3:\EFI\BOOT\sbsa\sbsa.nsh
```
 3. Applicable only for first time when the image is executed to install the SCT tests.
 4. To run test just with console logs (not save in file), run the 'Sbsa.efi'.

```
FS3:\EFI\BOOT\sbsa\> Sbsa.efi
```

Linux Test

- Choose 'sbbr/sbsa' from GRUB.
- Boot Linux and run 'sbsa' from commandline once boot is complete.

Running SBSA UEFI test (Automatically on boot)

```
GNU GRUB version 2.02
```

```
linux
linuxefi
*sbbr/sbsa
```

```
Press ESC in 5 seconds to skip startup.nsh or any other key to continue.
```

```
Shell> echo -off
```

```
Press any key to stop the EFI SCT running
```

```
add-symbol-file /data_sdb/pr.../_workspace/acs/2.5/arm-enterprise-acs/luv/build/tmp/work/aarch64-oe-linux/sbbr/v1.1+gitAUTOINC+b558bad254-r0/edk2/Build/SbbrSct/DEBUG_GCC49/AARCH64/SctPkg/Application/StallForKey/StallForKey/DEBUG/StallForKey.dll 0xF99180>Loading driver at 0x000F9917000 EntryPoint=0x000F9918000 StallForKey.efi
```

```
remove-symbol-file /data_sdb/pr.../_workspace/acs/2.5/arm-enterprise-acs/luv/build/tmp/work/aarch64-oe-linux/sbbr/v1.1+gitAUTOINC+b558bad254-r0/edk2/Build/SbbrSct/DEBUG_GCC49/AARCH64/SctPkg/Application/StallForKey/StallForKey/DEBUG/StallForKey.dll 0xF99180;add-symbol-file /data_sdb/pramad01/workspace/acs/2.5/arm-enterprise-acs/luv/build/tmp/work/aarch64-oe-linux/sbsa/1.0+gitAUTOINC+d6c2198725-r0/edk2/Build/Shell/DEBUG_GCC49/AARCH64/AppPkg/Applications/sbsa-acs/uefi_app/SbsaAvsN>Loading driver at 0x000F78FD000 EntryPoint=0x000F78FE000 Sbsa.efi
```

```
SBSA Architecture Compliance Suite
Version 2,5
```

```
Starting tests for level 4 (Print level is 3)
```

Running SBSA UEFI test (Manually from UEFI Shell)

```
GNU GRUB version 2.02

/-----
| luv
| *sbb/sbsa
|-----
```

```
Press ESC in 5 seconds to skip startup.nsh or any other key to continue.
She11> fs2:
FS2:\> FS3:EFI\BOOT\sbsa\sbsa.nsh
FS2:\> echo -off
add-symbol-file /data_sdb/pr ... /workspace/acs/2.5/arm-enterprise-acs/luv/buil
d/tmp/work/aarch64-oe-linux/sbsa/1.0+gitAUTOINC+d6c2198725-r0/edk2/Build/Shell/I
EBUG_GCC49/AARCH64/AppPkg/Applications/sbsa-acs/uefi_app/SbsaAvsNist/DEBUG/Sbsa.
Loading driver at 0x000F78FD000 EntryPoint=0x000F78FE000 Sbsa.efi

SBSA Architecture Compliance Suite
Version 2.5

Starting tests for level 4 (Print level is 3)
```

Run the test with console logs, and save the same into log file

Run the test only with console logs

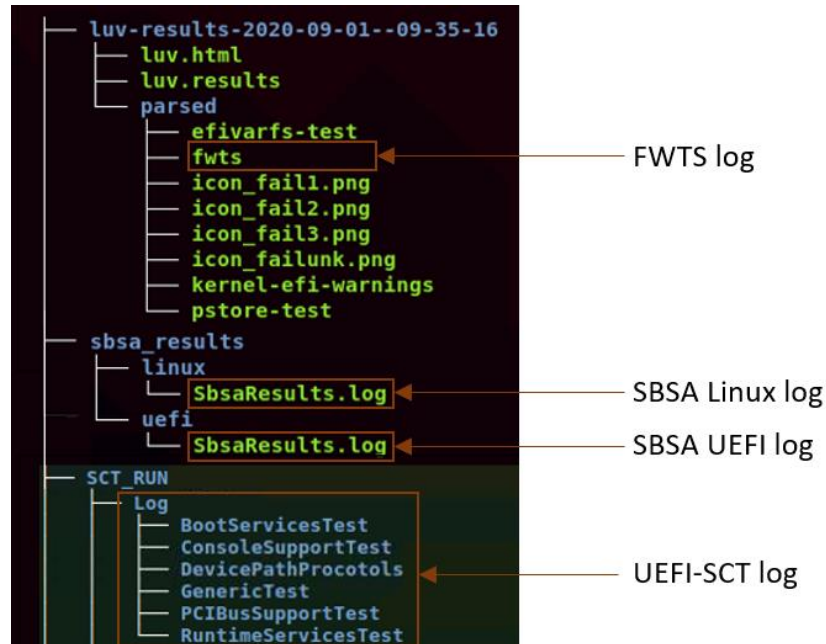
```
Press ESC in 5 seconds to skip startup.nsh or any other key to continue.
She11> fs3:
FS3:\> cd EFI\BOOT\sbsa
FS3:\EFI\BOOT\sbsa> Sbsa.efi
```

Analyzing Result

- Physical platform: Results available at first partition of USB
- FVP: Mount the 'luv-live-image-gpt.img1' to get results

Results:

1. SBSA: <https://git.linaro.org/landing-teams/working/arm/arm-reference-platforms.git/tree/docs/rdn1edge/acs-results/rdn1edge-sbsa.log>
2. FWTS: <https://git.linaro.org/landing-teams/working/arm/arm-reference-platforms.git/tree/docs/rdn1edge/acs-results/rdn1edge-fwts.log>



Debugging ACS Errors

SBSA

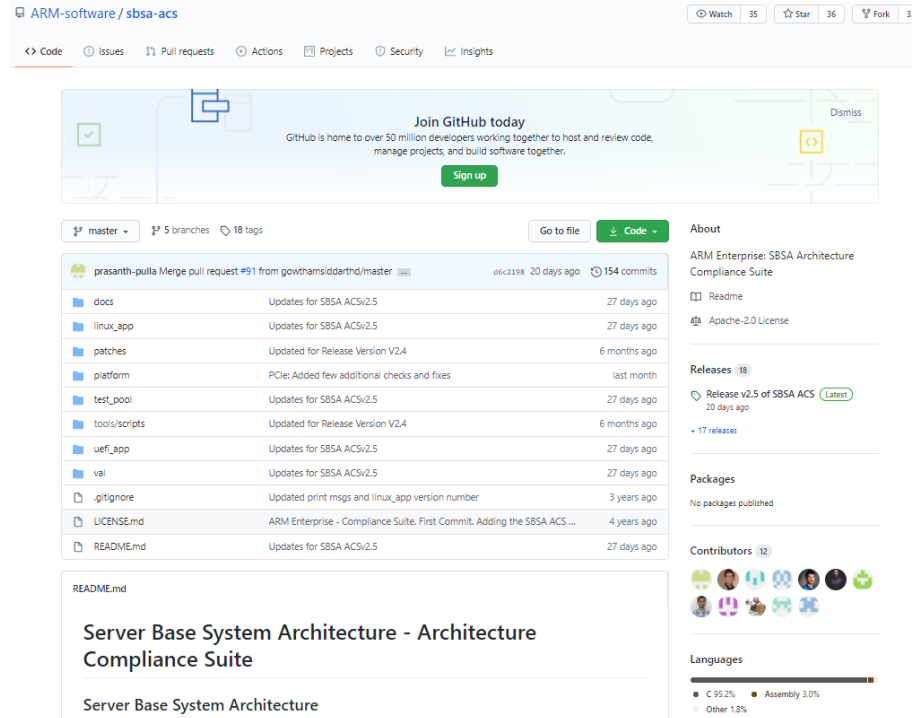
- Figure out Failed/Skipped test cases.
- For UEFI, tests are present in:
`/path/to/arm-enterprise-acs/luv/build/tmp/work/aarch64-oe-linux/sbsa/1.0+gitAUTOINC+<commit-id>-r0/git/test_pool/`
 - Test 201 : Check Counter Frequency
`test_pool/timer_wd/test_t001.c`
 - Test 442 : Check Power Management rules
`test_pool/pcie/test_p042.c`
 - Start from 'payload()' in 'test_*.c' to analyze the test case.
- For Linux:
`meta-luv/recipes-utils/sbsa-acs-app/sbsa-acs-app/`

SBBR

- Figure out Failed/Skipped test cases.
- For SCT:
`build/tmp/work/aarch64-oe-linux/sbbr/v1.1+gitAUTOINC+b558bad254-r0/git/`
- For FWTS:
`build/tmp/work/qemuarm64-oe-linux/fwts/V18.02.00+gitAUTOINC+<commit-id>-r0/git/fwts-test/`

Running Custom SBSA Binary

- Clone <https://github.com/ARM-software/sbsa-acs> to local edk2 source code.
- Compile the Sbsa efi application. Refer here for steps to integrating the app into UEFI build: <https://github.com/ARM-software/sbsa-acs#acs-build-steps---uefi-shell-application>
- Copy the SBSA efi binary to a disk image.
- Boot the device with this disk accessible to UEFI.
- Run the 'Sbsa.efi' from this disk partition.
- Get test result from console log.



ARM-software / sbsa-acs

Watch 35 Star 36 Fork 3

<> Code Issues Pull requests Actions Projects Security Insights

Join GitHub today
GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together.
Sign up

master 5 branches 18 tags Go to file Code

docs	Updates for SBSA ACSv2.5	27 days ago
linux_app	Updates for SBSA ACSv2.5	27 days ago
patches	Updated for Release Version V2.4	6 months ago
platform	PCIe: Added few additional checks and fixes	last month
test_pool	Updates for SBSA ACSv2.5	27 days ago
tools/scripts	Updated for Release Version V2.4	6 months ago
uefi_app	Updates for SBSA ACSv2.5	27 days ago
val	Updates for SBSA ACSv2.5	27 days ago
.gitignore	Updated print msgs and linux_app version number	3 years ago
LICENSE.md	ARM Enterprise - Compliance Suite. First Commit: Adding the SBSA ACS ...	4 years ago
README.md	Updates for SBSA ACSv2.5	27 days ago

README.md

Server Base System Architecture - Architecture Compliance Suite

Server Base System Architecture

Thank you

Accelerating deployment in the Arm Ecosystem

