Software Just Works on Arm-based Devices

White paper - Version 2

arm SystemReady

Contents

Glossary

This paper uses the following terms and abbreviations:

Term	Meaning
ACPI	Advanced Configuration and Power Interface
ACS	Architecture Compliance Suite
BBR	Base Boot Requirements
BBSR	Base Boot Security Requirements
BMC	Baseboard Management Controller
BSA	Base System Architecture
CSP	Cloud Service Provider
CXL	Compute Express Link
EBBR	A recipe defined in the BBR specification requiring a subset of UEFI. Also used for the Embedded Base Boot Requirements specification
ECR	Engineering Change Request
IBV	Independent BIOS (or Firmware) Vendor
IHV	Independent Hardware Vendor
ISV	Independent Software Vendor
LBBR	A recipe defined in the BBR specification supporting LinuxBoot
OCP	Open Compute Project
ODM	Original Device Manufacturer
OEM	Original Equipment Manufacturer
OSV	Operating System Vendor
SBBR	A recipe defined in the BBR specification requiring UEFI and ACPI. Also used for the previous Server Base Boot Requirements specification
SBMR	Server Base Manageability Requirements

SBSA	BSA Supplement for Servers. Also used for the previous Server Base System Architecture specification
SiP	Silicon Provider
SoC	System on Chip
UEFI	Unified Extensible Firmware Interface

For other terms and abbreviations, visit the Arm Glossary on the Arm Developer website: https://developer.arm.com/support/arm-glossary

Executive Summary

This white paper describes the Arm standards-based approach to system development, and the Arm SystemReady program with its certification process.

Arm SystemReady is a set of standards and a compliance certification program that enable interoperability with generic off-the-shelf operating systems and hypervisors. You will see the term "just works" used throughout this white paper and in other Arm SystemReady materials. When we use the term "just works", we mean the process of enabling operating systems to work out of the box on Arm-based devices, through the adoption of these standards and tools. Because Arm SystemReady standards allows software to just work, our ecosystem can focus on innovation and differentiation.

The audience for this paper includes SoC designers, platform software engineers, OEM and ODM system designers, OS vendors, and end customer procurement teams. Arm SystemReady targets cloud and datacenter, embedded-server, and high-performance IoT devices.

The Arm SystemReady program is a set of tools that Arm provides to the Arm ecosystem. Depending on the needs of the market, these tools can be used to:

- Enable the just works deployment of OS distributions to standards-compliant Arm-based platforms
- Open up markets for Arm-based silicon where OS portability and choice is a barrier to entry, including the infrastructure edge and embedded IoT
- Reduce the cost for deploying software to an Arm-based platform, and the cost of maintaining that software over the device lifetime, by using common OS distributions that support industry standards

SystemReady describes the requirements of an OS distribution, and a certification scheme for demonstrating whether a platform meets those requirements. We do not prescribe how to use SystemReady. Our partners determine what they need, which certification is appropriate to opt into, and how to engage with the standards process.

Introduction

Systems that are designed to just work for the end user – with the ability to install and run generic, off-the-shelf operating systems out of the box – must follow a set of minimum hardware and firmware requirements.

For the Arm ecosystem, this requirement first surfaced in the server segment. The Arm ServerReady compliance certification program provides this "just works" solution for servers, allowing partners to deploy Arm servers with confidence. The program is based on industry standards and is accompanied by a compliance test suite, and a process for certification.

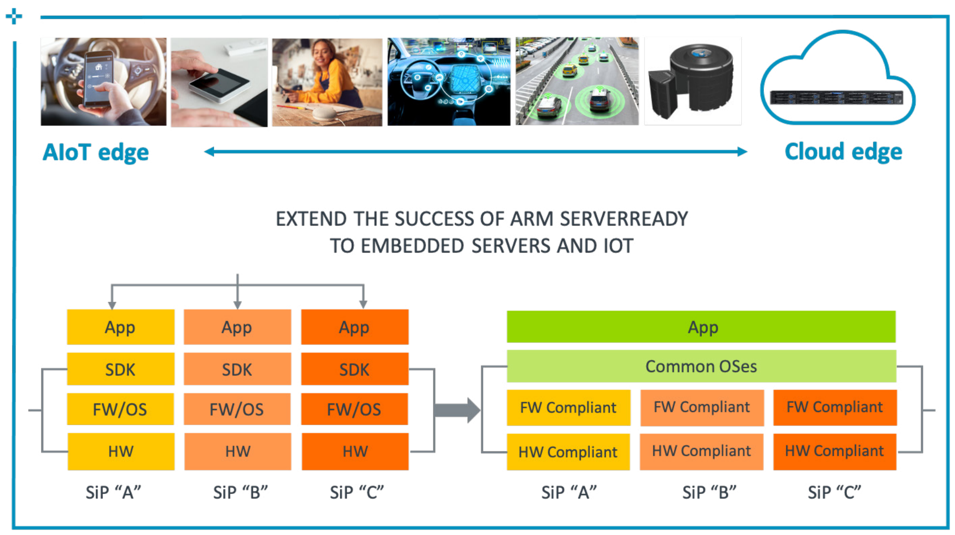
The embedded-server and high-performance IoT segments are different from cloud and data center segments. This is because an operating system is often provided with a device and different standards may be used. However, a standard hardware and firmware platform, with a certification scheme, brings benefits to the embedded-server and IoT ecosystem that are similar to the benefits in the server segment. Platforms that are certified under the Arm SystemReady program are guaranteed to implement a minimum set of hardware and firmware features that an operating system can depend on for deploying the OS image.

Arm SystemReady takes what the Arm ecosystem has learned from Arm ServerReady, and applies it to a broader set of devices in the embedded-server and high-performance IoT ecosystem. If you choose SystemReady compliant platforms, you can:

- Significantly reduce the cost that is associated with adopting a new platform by eliminating custom engineering associated with firmware
- Use industry standard management tools
- Enable OS vendors to provide direct support without the need for a custom kernel build

Figure 1 shows how the Arm SystemReady program standardizes and simplifies system development for the Arm ecosystem, extending from high-performance IoT devices to embedded servers. With Arm SystemReady, an OEM can amortize their OS engineering work across multiple platforms. Independent software vendors (ISVs) can build new businesses around a standard distribution with support and security updates. System integrators can buy certified platforms with confidence, knowing that they can run their chosen OS.

Figure 1: The expansion of the Arm ServerReady program into a wider range of markets



This white paper describes the Arm SystemReady program and the Arm standards-based approach to system development. For the Arm SystemReady certification process, refer to Appendix B of the Arm SystemReady Requirements Specification v1.1.

Arm System Architecture

4.1 Arm standards-based approach

The Arm architecture supports a diverse ecosystem of devices. The infrastructure and IoT space is also diverse, and includes IoT gateways, embedded servers, and cloud and enterprise datacenters. Although diversity is evidence of a healthy ecosystem, unnecessary divergence on Arm-based silicon creates barriers to OS and application portability. Open systems must be deployed, configured, and managed in heterogenous environments. These markets are typically horizontal, where no single vendor owns the entire stack from hardware to operating system. These open systems require support for generic off-the-shelf operating systems. Customers are used to an out-of-the-box experience, where they can take the system out of the shipping box and deploy the OS easily. Or, in the IoT and edge context, customers expect a zero-touch method to provision an OS and application remotely to a machine that is shipped directly from manufacturer to installation point. This zero-touch method often includes OSes that pre-date the silicon, like supporting current and previous versions of an OS. In this space, customizing the OS to work with specific silicon is not usually a viable option.

To unlock value from the embedded-server and high-performance IoT segments, a seamless software experience, which includes an extension of a cloud-native model, becomes essential. Beyond the operational expense savings that are associated with early enterprise IoT deployments, the focus of the IoT segment is expected to shift towards designing more revenue streams, based on application service monetization models. To support these IoT models, the experience of working with systems is frictionless. The process involves landing the end-user choice of operating system or distribution, to working with cloud native stacks managing hundreds of thousands of IoT devices and low-power gateways.

In the cloud and datacenter markets, the industry and the Arm ecosystem have come together in support of the Arm ServerReady program. This program ensures that server-grade silicon and systems adhere to the standards that make these platforms general purpose in nature. With the introduction of the Arm SystemReady program, we extend this vision beyond cloud datacenters. SystemReady provides a similar working experience to the spectrum of infrastructure for the embedded servers and high-performance IoT. This infrastructure includes, for example, systems that are deployed outside of cooled environments, on factory floors, smart buildings, and transport infrastructure, and wireline and wireless networking security and storage gear.

Industry standards offer common rules for hardware and firmware, to enable support for generic off-the-shelf operating systems. These rules establish a contract among the ecosystem players for a baseline framework of interoperability. Participants in the ecosystem are expected to adhere to this common baseline, while innovating on top of it to differentiate and add value in their products for their customers. This standards-based foundation for systems design is critical to the success of the datacenter, embedded-server and IoT gateway deployment.

The Arm standards-based approach focuses on four aspects:

- **Collaboration:** Arm uses a collaborative process, involving companies across the Arm ecosystem. Collectively, these companies form Advisory Committees, with members from OEMs, ODMs, silicon providers (SiPs), OS vendors (OSVs), and software and firmware vendors (ISVs and IBVs), cloud service providers (CSPs), and other hardware and IP vendors.
- **Creation of Arm standards:** Arm standards include the Base System Architecture (BSA) and its market-segment supplements and BBR (Base Boot Requirements). Other standards include the Power State Coordination Interface (PSCI) and the Secure Monitor Call Calling Convention (SMCCC).
- **Participation in industry standards:** Arm participates and provides leadership in various industry standards bodies and groups, ensuring the compatibility of these standards with the Arm architecture. These bodies and groups include the UEFI Forum, PCI SIG, DMTF, OCP, TCG, and the CXL Consortium.
- **Supporting open-source software:** Arm supports open-source software and firmware projects that help accelerate the development and adoption of Arm standards. These projects include Trusted Firmware, TianoCore, UBoot, LinuxBoot, and the Linux kernel.

4.2 Base System Architecture and its supplements

The Base System Architecture (BSA) document specifies a minimum set of CPU and system architecture requirements that are necessary for an OS to boot and run, regardless of the market segments. BSA includes baseline requirements for:

- Processor features
- Memory subsystem features
- GIC and SMMU revision features
- PCIe (Peripheral Component Interconnect Express) integration features
- Base levels for other peripherals, for example UART, USB, and SATA
- Security features
- Power semantics

These requirements are intended to ensure that standard software, or operating systems, operate correctly on machines that are compliant with the BSA.

The server supplement of the BSA is the Server Base System Architecture (SBSA). The SBSA offers several compliance levels. The different levels represent the advancement of the specification over time. When new architectures arrive, new levels are added. Generally, each level includes all the requirements of a previous level. For example, SBSA compliance Level 5 was added to correspond to new features in the Armv8.4-A architecture. Level 6 corresponds to changes in the Armv8.5-A architecture, and Level 7 corresponds to changes in the Armv8.6-A architecture.

In the future, Arm might develop other supplements of the BSA – we call these xBSAs until they have been formally developed – for different market segments. These supplements are deliberately modular, so that partners can use the relevant specifications for the market segment that they are designing for.

Figure 2 shows the relationships between the BSA Specification and the market-specific supplements:

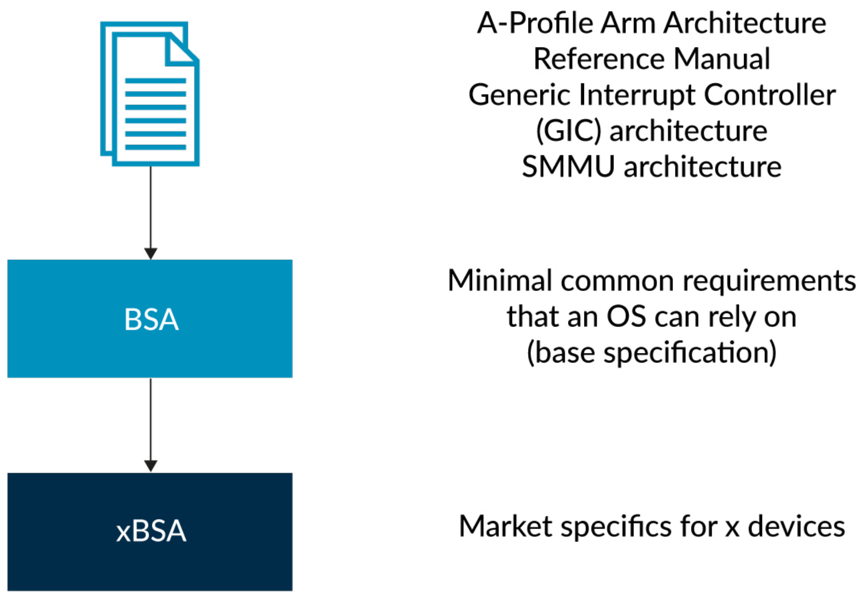

A-Profile Arm Architecture
Reference Manual
Generic Interrupt Controller
(GIC) architecture
SMMU architecture
BSA
Minimal common requirements
that an OS can rely on
(base specification)
xBSA
Market specifics for x devices

Figure 2: BSA specification and its relationship to market-specific supplements

4.3 Base Boot Requirements

The Base Boot Requirements (BBR) document specifies firmware interface requirements that system software, like operating systems and hypervisors, can rely on. Firmware interface requirements include the following specifications:

- UEFI specification
- ACPI specification
- SMBIOS specification
- Other Arm specifications, for example PSCI and SMCCC

The BBR also provides recipes – a recipe meaning a set of requirements – that are tailored to the various generic operating systems:

- **SBBR recipe**
    - Requirements based on the previous SBBR specification
    - PSCI, SMCCC, UEFI, ACPI, SMBIOS
- **EBBR recipe**
    - Requirements based on the EBBR specification
    - PSCI, SMCCC, UEFI
- **LBBR recipe**
    - PSCI, SMCCC, LinuxBoot, ACPI, SMBIOS

Arm may develop other recipes in the future, if necessary.

Architecture Compliance Suites

At Arm, we live by the statement, "There is no specification without verification". This statement is realized by the various Arm compliance programs that help developers to ensure that their hardware is fully compliant with the Arm CPU architecture specifications. In the same spirit, Arm has created the Architecture Compliance Suite (ACS) for the verification of the previous SBSA and SBBR specifications. This ACS test is an essential component of the Arm ServerReady program, and it covers:

- **SBSA hardware requirements**
  - CPU properties
  - System components
  - PCIe integration
  - Based on the PCIe specification
  - Based on standard OS drivers with no quirks enabled

- **SBBR defined firmware requirements**
  - UEFI testing that is based on the UEFI Self-Certification Test (SCT)
  - ACPI and SMBIOS testing that is based on Firmware Test Suite (FWTS)

The test suites are hosted in GitHub and are open source, with a permissive Apache v2 license.

5.1 SBSA Architecture Compliance Suite

The SBSA Architecture Compliance Suite (ACS) is a collection of tests that can be used to verify if a system is compliant to the SBSA specification. Most of the tests are executed from the UEFI shell, by executing the SBSA UEFI shell application. A few tests are executed by running the SBSA ACS Linux application, which in turn depends on the SBSA ACS Linux kernel module.

5.2 Enterprise Architecture Compliance Suite

The Arm Enterprise ACS is a collection of tests that can be used to verify if a system is compliant to the SBSA and SBBR specification. ACS is delivered with tests in source form, and a build script. The output of the build is a bootable Linux OS image that can run all tests that are required by these specifications.

The existing ACS is designed for compliance with previous SBSA and SBBR specifications. Arm is restructuring the ACS for the Arm SystemReady compliance testing, to be used for future compliance of the BSA and its supplements, and for BBR recipes. The existing SBSA ACS and Enterprise ACS are used for the SystemReady SR and ES certification currently.

Arm SystemReady Program

6.1 Arm SystemReady program

The Arm SystemReady program is a natural extension of the Arm ServerReady program. Different market segments may target different sets of operating systems and hypervisors with different hardware and firmware requirements. We use the term band to identify these differences, with a shorthand notation for each band. The bands are:

- SystemReady SR
- SystemReady LS
- SystemReady ES
- SystemReady IR

SystemReady SR is technically identical to ServerReady and continues to bring the exact benefits to the Arm server ecosystem. The additional bands in SystemReady LS, ES, and IR are designed to serve the needs of a broader silicon and software ecosystem. We are defining the bands in consultation with our partners, and we expect that all operating system distributions will find a band that adequately captures their basic requirements for a standards-based Arm platform.

All SystemReady bands will be supported by a common ACS. The new ACS will be modular. It will support testing against different combinations of specifications required by each SystemReady band.

Table 1 describes the different SystemReady bands and the specifications that are required for each band:

Table 1: Arm SystemReady bands

Certification	Specifications		
SystemReady SR	BSA	SBSA	SBBR
SystemReady LS	BSA	SBSA	LBBR
SystemReady ES	BSA	-	SBBR
SystemReady IR	BSA	-	EBBR

SystemReady ES, and IR for 64-bit, have the same hardware requirements.

- SystemReady IR requires devicetree support in addition to the reduced set of UEFI interfaces that are specified in the EBBR specification.
- SystemReady ES requires ACPI and SMBIOS interfaces, in addition to the UEFI interfaces.

SystemReady SR requires additional SBSA compliance for hardware and more stringent UEFI and ACPI requirements for firmware.

Systems that are certified as SystemReady SR meet the requirements for SystemReady ES. There is no need for these systems to be certified as SystemReady ES.

A 32-bit system can be certified as SystemReady IR if it supports devicetree and the EBBR specification. However, because the BSA specification does not cover 32-bit systems, we list the 32-bit systems separately from the 64-bit systems on the Arm SystemReady System Compatibility List (SCL). The System Compatibility Lists for SystemReady SR and SystemReady ES are available on the respective developer.arm.com webpages. The SCL for SystemReady IR will be available on the SystemReady IR developer.arm.com webpage in the future.

The detailed requirements for these bands are defined in the Arm SystemReady Requirements Specification. Appendix A of the specification provides the waiver levels for the current generation of SoCs targeting the embedded-server and IoT markets. These SoCs are known not to be fully BSA-compliant. Hardware or firmware workarounds can be used, however, to achieve the just-works software experience. Similarly, in some cases, the just-works experience can be achieved through OS changes contained in future OS releases. Arm defined waiver levels 0 to 2 for SystemReady ES and IR to accommodate this situation. The use of these levels will be time limited as defined in the Arm SystemReady Requirements Specification.

Table 2 describes the overview of the different technical requirements for each SystemReady band. For the details please refer to the Arm SystemReady Requirements Specification.

	LS (LinuxBoot Server)	IR (IoT)	ES (Embedded Server)	SR (ServerReady)
Firmware Spec	ACPI +SMBIOS	UEFI + Devicetree	UEFI + ACPI + SMBIOS	UEFI + ACPI + SMBIOS
Platform Hardware	64bit Arm	32bit/ 64bit Arm	64bit Arm	64bit Arm
		Can support UEFI SecureBoot and Secure Firmware Update via UEFI Capsule Sevice across (BBSR)		
OS/ Hypervisor	Linux	Linux, etc. →	Generic, off-the-shelf w/ exceptions: RAS, virtualization, etc. →	Generic, off-the-shelf
OS Distro (examples)	Linux	Fedora, openSUSE, Ubuntu, Debian, CBL-Mariner Under investigation: OpenWRT, QNX, VxWorks, Integrity, Yocto, Wind River, Mentor	Windows IoT Enterprise, VMware ESXi, RHEL, SLES, Ubuntu, CentOS, Fedora, openSUSE, Debian, CBL-Mariner, FreeBSD, NetBSD	VMware ESXi, Windows Client/Server, RHEL, SLES, Ubuntu, CentOS, Fedora, openSUSE, Debian, CBL-Mariner, FreeBSD, NetBSD
Hardware Compliance Levels	BSA+SBSA Levels 3 through 6	BSA + No BSA requirements for 32-bit + waivers for existing HW initially	BSA +waivers for existing HW initially	BSA+SBSA Levels 3 through 6
BBR Recipe	LBBR	EBBR	SBBR	SBBR
Certification	Arm SystemReady LS +System Compatibiltiy List	Arm SystemReady IR + System Certification List	Arm SystemReady ES + System Certification List	Arm SystemReady SR + System Certification List

🟩 Common UEFI interfaces.

→ User experience of Software

Table 2: Overview of requirements for each Arm SystemReady band

6.2 Arm SystemReady certification process

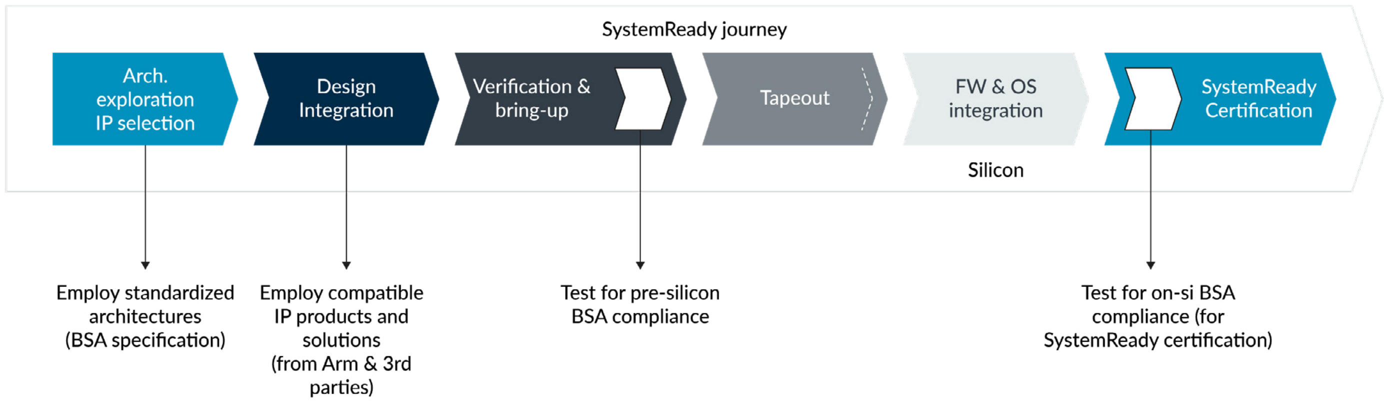
Appendix B of the Arm SystemReady Requirements Specification shows the certification process for the Arm SystemReady program. Some of the significant changes from the original ServerReady program are:

- Partners need to have technical engagement with the dedicated technical leads at Arm.
- Third-party test labs are set up to assist the certification.
- Partners need to make the device available to Arm or the third-party test labs.

Compliant systems that adhere to the Arm SystemReady Terms and Conditions are issued with a compliance certificate. Partners can use the SystemReady logo as part of their product promotion, if they passed the certification process.

Pre-Silicon Compliance Testing

Currently, the Arm SoCs that target the embedded server and IoT markets are mostly not compliant with BSA. This is why pre-silicon BSA compliance is key to SystemReady and software that just works. In addition to design verification, Arm strongly recommends that BSA compliance is tested before silicon tape-out. Pre-silicon BSA compliance is an integral part of the journey to SystemReady.


SystemReady journey
Arch. exploration IP selection
Design Integration
Verification & bring-up
Tapeout
FW & OS integration
SystemReady Certification
Silicon
Employ standardized architectures (BSA specification)
Employ compatible IP products and solutions (from Arm & 3rd parties)
Test for pre-silicon BSA compliance
Test for on-si BSA compliance (for SystemReady certification)

Arm SystemReady pre-silicon compliance testing will be available for silicon partners. Arm is working closely with EDA partners on developing this testing. For silicon partners new to Arm SystemReady, pre-silicon compliance is a key step that makes the final Arm SystemReady certifications a much simpler task.

The Arm SystemReady program aims to make software just work on Arm-based devices. For systems that support technologies like PCIe, this typically means integration with third-party IP. Many Arm SoCs in the current generation are not compliant with PCIe requirements, as identified in the BSA specification. The lack of standard PCIe Enhanced Configuration Access Mechanism (ECAM) support is a good example. In addition, interrupt delivery, UART, USB host controller, and watchdog timer are areas where non-compliance causes a need for vendor-specific drivers. This breaks the just-works vision for software. Until now, the non-compliance is only identified post-silicon. It is very costly to respin the SoC or wait for the next generation of SoCs to be released. Providing a pre-silicon compliance test for SoC designs, before tape-out, is critical for compliance.

Security Extension

The Arm SystemReady program includes a Security Extension that addresses the requirement that OS distributions have for standard security interfaces. Notable security interfaces include UEFI Secure Boot and UEFI secure firmware update using Update Capsules. The security interface requirements are specified in the Base Boot Security Requirements (BBSR) specification. The Security Extension applies to SystemReady SR, ES and IR bands. Arm is extending the Architecture Compliance Suite (ACS) to add test cases that verify compliance with BBSR.

The SystemReady Security Extension certification provides assurance that the security interfaces in scope are implemented according to standards. However, interface compliance does not provide assurance that the underlying platform is secure. In the process of architecting a system, system-level threat modeling should be performed to evaluate threats, risks, and mitigations.

SystemReady Compliance

This section provides an overview of the SystemReady compliance with the Base System Architecture and Base Boot Requirement specifications.

9.1 BSA compliance

To make software just work on Arm-based devices, silicon providers must ensure that SoCs are designed to be BSA-compliant, regardless of the market segment.

If the hardware is BSA-compliant, the OS vendor can target the BSA as a common Arm hardware platform, instead of having to do platform-specific engineering. For open-source operating systems, a large community is working on support for BSA-compliant platforms. This means that upstreamed support is automatic.

If PCIe is supported, PCIe-compliance is critical. PCIe enumeration by software is a complex and challenging area. Arm is working with third-party IP providers to enable IP integration compliance verification before tape-out.

If the SoCs are targeting specific market segments, extra compliance with the segment-specific supplements is needed. For example, compliance with SBSA Supplement is important for the server systems. The end-customer and OS vendor must decide whether they require support for a segment-specific supplement.

The BSA specification covers the 64-bit Arm Application profile.

9.2 BBR compliance

When an SoC is compliant with the BSA, firmware development can be flexible. Different BBR recipes – a recipe meaning a set of requirements – can be supported based on the target OSes.

For example, the same device based on an SoC that is compliant to the BSA and the SBSA Supplement can either provide the firmware using:

- The SBBR recipe if the target OS requires UEFI, ACPI and SMBIOS, or
- The LBBR recipe if LinuxBoot is used instead

Similarly, the same device based on an SoC that is compliant to BSA can either provide the firmware using:

- The SBBR recipe if the target OS requires UEFI, ACPI and SMBIOS, or
- The EBBR recipe if the target OS requires UEFI and devicetree

This is the basis for the current four bands of the Arm SystemReady program:

- SystemReady SR
- SystemReady LS
- SystemReady ES
- SystemReady IR

For devices that are Arm SystemReady SR, ES, or IR certified, there is an extra Security Extension to certify whether the device also supports the UEFI Secure Boot and secure firmware update through Capsule services.

SystemReady SR, ES, and IR require firmware that implements the UEFI ABI. To enable binary portability of operating system images between platforms, an agreed ABI between the OS and the Non-secure world boot loader must exist. Although we could have invented a new standard for the embedded world, UEFI has the benefit of a large ecosystem, including implementations, for example, in U-Boot, that have been demonstrated on embedded devices.

If an OEM or ODM wants to build a vertically integrated platform, based on Arm, that runs its own embedded Linux distribution, or one that is provided by the silicon provider, Arm SystemReady certification is not required. However, the OEM or ODM partner should consider whether it is possible to reduce software engineering cost over the lifetime of the device by using an Arm SystemReady-certified SoC.

Use Cases

The table below provides a list of example use cases for the different SystemReady band certifications.

Use-case	SystemReady solution
I want to build an Arm server SoC for hyperscale or enterprise and my customer will choose and deploy a server OS.	Build SoC and firmware for SystemReady SR certification.
I want to build an IoT gateway based on an Arm SoC. I want to deploy a standard Linux distro that can host containers and application software stacks.	Build SoC and firmware for SystemReady IR certification.
I want to build an enterprise networking platform based on Arm silicon from my supplier. I will build my own OS based on Yocto Linux, but I want UEFI standard firmware and also UEFI Secure boot. I want to update my firmware using UEFI capsule update.	Require SoC and firmware to be SystemReady IR + Security Option. Consider PSA Certified for an independent lab evaluation of the Root of Trust.
I want to land my OS distribution on Arm SoCs. My OS needs ACPI. However, I am prepared to work around hardware that is not quite BSA-compliant to gain access to a broad selection of silicon, originally built for the mobile and embedded markets.	Require SoC and firmware to be SystemReady ES certified.
I want to deploy an Arm server with firmware that supports LinuxBoot	Require server to be SystemReady-LS certified.
I want to build a vertically integrated platform based on Arm that runs my own embedded Linux distribution, or one provided by the SIP.	SystemReady certification is not required. However, consider whether it is possible to reduce my software engineering cost over the lifetime of the device by using a SystemReady device.

Conclusion

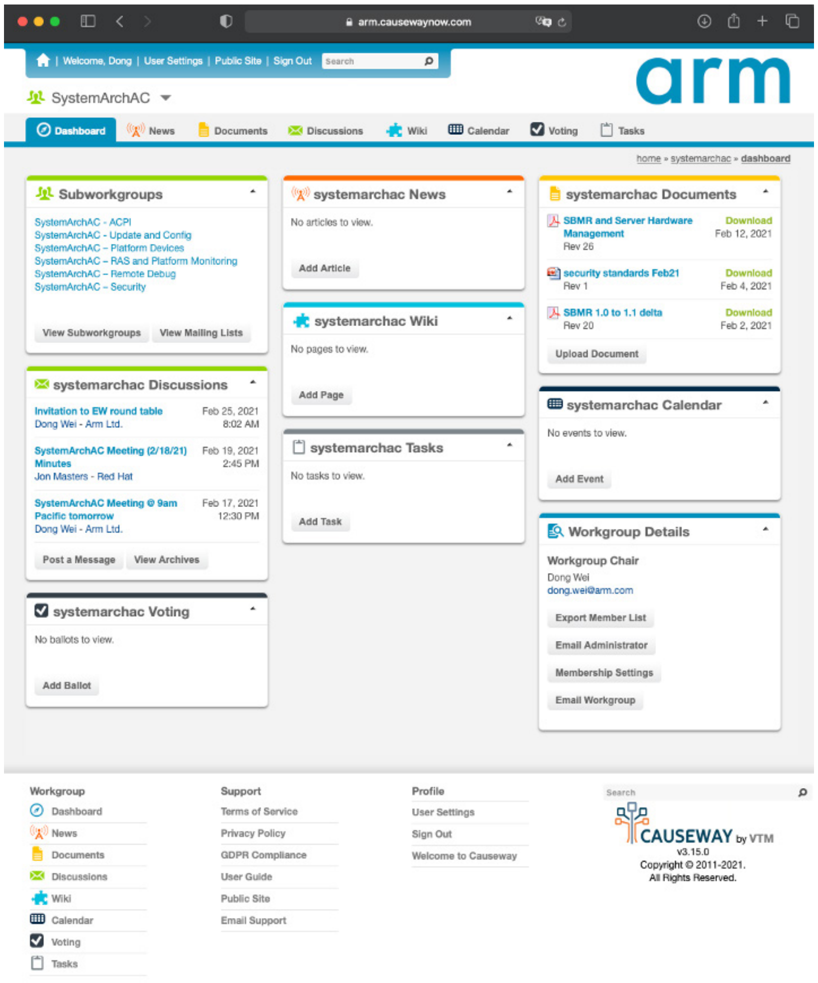
This white paper describes the Arm SystemReady program, its components and process, and the extension from the existing ServerReady compliance program. The paper also describes the Arm standards-based approach and Arm-specific standards that lay the foundation for the SystemReady program. For more information and resources, please refer to the Arm SystemReady website. For any questions, please contact SystemReady@arm.com.

Appendix A

Arm System Architecture Advisory Committee

The Arm ecosystem collaborates on the creation of standards through the Arm System Architecture Advisory Committee (SystemArch AC). This group includes companies across the Arm ecosystem, including OEMs, ODMs, silicon providers, OS vendors, software and firmware vendors, cloud service providers, and other hardware and IP vendors. The committee uses Causeway – an online collaboration platform for developing technical specifications – and communicates in various ways including email, regular conference calls, and biannual gatherings. To communicate with this group, email systemarchac@arm.causewaynow.com.

The following screenshot is the SystemArchAC dashboard on Causeway. It provides an example of how the SystemArch AC uses the collaboration platform:



Subworkgroups

SystemArchAC - ACPI
SystemArchAC - Update and Config
SystemArchAC – Platform Devices
SystemArchAC – RAS and Platform Monitoring
SystemArchAC – Remote Debug
SystemArchAC – Security

View Subworkgroups | View Mailing Lists

systemarchac Discussions

Invitation to EW round table Dong Wei - Arm Ltd.	Feb 25, 2021 8:02 AM
SystemArchAC Meeting (2/18/21) Minutes Jon Masters - Red Hat	Feb 19, 2021 2:45 PM
SystemArchAC Meeting @ 9am Pacific tomorrow Dong Wei - Arm Ltd.	Feb 17, 2021 12:30 PM

Post a Message | View Archives

systemarchac Voting

No ballots to view.

Add Ballot

systemarchac News

No articles to view.

Add Article

systemarchac Wiki

No pages to view.

Add Page

systemarchac Tasks

No tasks to view.

Add Task

systemarchac Documents

SBMR and Server Hardware Management Rev 26	Download Feb 12, 2021
security standards Feb21 Rev 1	Download Feb 4, 2021
SBMR 1.0 to 1.1 delta Rev 20	Download Feb 2, 2021

Upload Document

systemarchac Calendar

No events to view.

Add Event

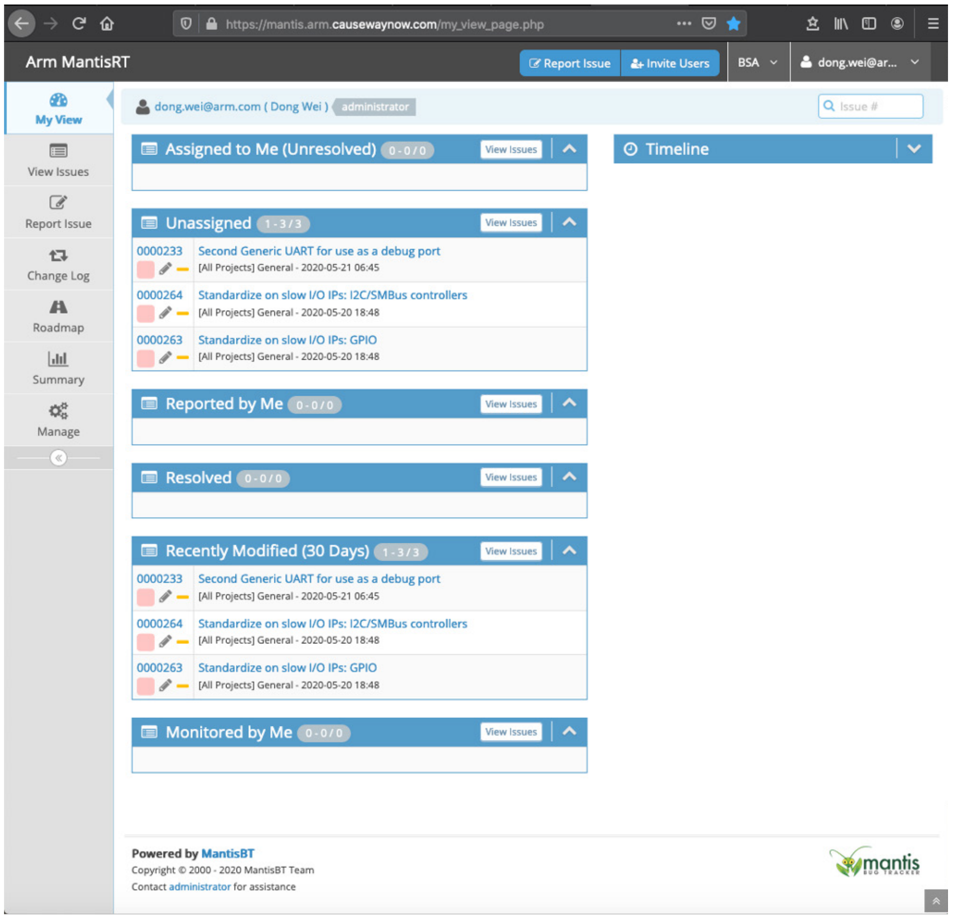
Workgroup Details

Workgroup Chair
Dong Wei
dong.wei@arm.com

Export Member List | Email Administrator | Membership Settings | Email Workgroup

The SystemArchAC uses the Mantis Issue tracker on Causeway to develop the specifications and tracking issues, proposals, and requests. The database is accessible to SystemArchAC member companies using the Causeway login portal. Any SystemArchAC member can raise a request. Issues are described in the SystemArchAC regular meetings. When a change is agreed, it is integrated into the specifications.

The following screenshot shows an example personalized dashboard on Causeway:



dong.wei@arm.com (Dong Wei) administrator

Assigned to Me (Unresolved) 0 - 0 / 0

Unassigned 1 - 3 / 3

0000233 Second Generic UART for use as a debug port
[All Projects] General - 2020-05-21 06:45

0000264 Standardize on slow I/O IPs: I2C/SMBus controllers
[All Projects] General - 2020-05-20 18:48

0000263 Standardize on slow I/O IPs: GPIO
[All Projects] General - 2020-05-20 18:48

Reported by Me 0 - 0 / 0

Resolved 0 - 0 / 0

Recently Modified (30 Days) 1 - 3 / 3

0000233 Second Generic UART for use as a debug port
[All Projects] General - 2020-05-21 06:45

0000264 Standardize on slow I/O IPs: I2C/SMBus controllers
[All Projects] General - 2020-05-20 18:48

0000263 Standardize on slow I/O IPs: GPIO
[All Projects] General - 2020-05-20 18:48

Monitored by Me 0 - 0 / 0

Powered by MantisBT
Copyright © 2000 - 2020 MantisBT Team
Contact administrator for assistance

The SystemArchAC development process is described here:

1. The process starts with private discussions between a partner and Arm, raising an issue or requesting a change. However, the discussion could also start in public conversation between the Arm SystemArchAC community members.
2 An engineering change request (ECR) is submitted on the SystemArchAC Mantis against one of the supported specifications. The ECR includes a problem statement, and justification for change, and description of the requirement. The ECR gets updated to include a proposal for a spec change, which is driven by Arm or other SystemArchAC members.
3 The SystemArchAC community discusses the ECR, and changes are made based on collected feedback from all members.
4 When the SystemArchAC approves the ECR, Arm integrates the change in the specification, and makes it available in the next revision once it is published.

More Advisory Committees may be formed for various market segments, if required. At this time, BSA and its SBSA supplement, BBR, SBMR, and BBSR specifications are developed in the SystemArchAC.