ACPI Tables

The expectations of individual ACPI tables are discussed in the list that follows.

If a section number is used, it refers to a section number in the ACPI specification where the object is defined. If "Signature Reserved" is used, the table signature (the first four bytes of the table) is the only portion of the table recognized by the specification, and the actual table is defined outside of the UEFI Forum (see Section 5.2.6 of the specification).

For ACPI on arm64, tables also fall into the following categories:

- Required: DSDT, FADT, GTDT, MADT, MCFG, RSDP, SPCR, XSDT
- Recommended: BERT, EINJ, ERST, HEST, PCCT, SSDT
- Optional: BGRT, CPEP, CSRT, DBG2, DRTM, ECDT, FACS, FPDT, IBFT, IORT, MCHI, MPST, MSCT, NFIT, PMTT, RASF, SBST, SLIT, SPMI, SRAT, STAO, TCPA, TPM2, UEFI, XENV
- Not supported: BOOT, DBGP, DMAR, ETDT, HPET, IVRS, LPIT, MSDM, OEMx, PSDT, RSDT, SLIC, WAET, WDAT, WDRT, WPBT

Table	Usage for ARMv8 Linux
BERT	Section 18.3 (signature == "BERT") **Boot Error Record Table** Must be supplied if RAS support is provided by the platform. It is recommended this table be supplied.
BOOT	Signature Reserved (signature == "BOOT") **simple BOOT flag table** Microsoft only table, will not be supported.
BGRT	Section 5.2.22 (signature == "BGRT") **Boot Graphics Resource Table** Optional, not currently supported, with no real use-case for an ARM server.
CPEP	Section 5.2.18 (signature == "CPEP") **Corrected Platform Error Polling table** Optional, not currently supported, and not recommended until such time as ARM-compatible hardware is available, and the specification suitably modified.
CSRT	Signature Reserved (signature == "CSRT") **Core System Resources Table** Optional, not currently supported.
DBG2	Signature Reserved (signature == "DBG2") **DeBuG port table 2** License has changed and should be usable. Optional if used instead of earlycon=<device> on the command line.
DBGP	Signature Reserved (signature == "DBGP") **DeBuG Port table** Microsoft only table, will not be supported.
DSDT	Section 5.2.11.1 (signature == "DSDT") **Differentiated System Description Table** A DSDT is required; see also SSDT. ACPI tables contain only one DSDT but can contain one or more SSDTs, which are optional. Each SSDT can only add to the ACPI namespace, but cannot modify or replace anything in the DSDT.
DMAR	Signature Reserved (signature == "DMAR") **DMA Remapping table** x86 only table, will not be supported.
DRTM	Signature Reserved (signature == "DRTM") **Dynamic Root of Trust for Measurement table** Optional, not currently supported.

Table	Usage for ARMv8 Linux
ECDT	Section 5.2.16 (signature == "ECDT") **Embedded Controller Description Table** Optional, not currently supported, but could be used on ARM if and only if one uses the GPE_BIT field to represent an IRQ number, since there are no GPE blocks defined in hardware reduced mode. This would need to be modified in the ACPI specification.
EINJ	Section 18.6 (signature == "EINJ") **Error Injection table** This table is very useful for testing platform response to error conditions; it allows one to inject an error into the system as if it had actually occurred. However, this table should not be shipped with a production system; it should be dynamically loaded and executed with the ACPICA tools only during testing.
ERST	Section 18.5 (signature == "ERST") **Error Record Serialization Table** On a platform supports RAS, this table must be supplied if it is not UEFI-based; if it is UEFI-based, this table may be supplied. When this table is not present, UEFI run time service will be utilized to save and retrieve hardware error information to and from a persistent store.
ETDT	Signature Reserved (signature == "ETDT") **Event Timer Description Table** Obsolete table, will not be supported.
FACS	Section 5.2.10 (signature == "FACS") **Firmware ACPI Control Structure** It is unlikely that this table will be terribly useful. If it is provided, the Global Lock will NOT be used since it is not part of the hardware reduced profile, and only 64-bit address fields will be considered valid.
FADT	Section 5.2.9 (signature == "FACP") **Fixed ACPI Description Table** Required for arm64. The HW_REDUCED_ACPI flag must be set. All of the fields that are to be ignored when HW_REDUCED_ACPI is set are expected to be set to zero. If an FACS table is provided, the X_FIRMWARE_CTRL field is to be used, not FIRMWARE_CTRL. If PSCI is used (as is recommended), make sure that ARM_BOOT_ARCH is filled in properly - that the PSCI_COMPLIANT flag is set and that PSCI_USE_HVC is set or unset as needed (see table 5-37). For the DSDT that is also required, the X_DSDT field is to be used, not the DSDT field.
FPDT	Section 5.2.23 (signature == "FPDT") **Firmware Performance Data Table** Optional, not currently supported.
GTDT	Section 5.2.24 (signature == "GTDT") **Generic Timer Description Table** Required for arm64.
HEST	Section 18.3.2 (signature == "HEST") **Hardware Error Source Table** ARM-specific error sources have been defined; please use those or the PCI types such as type 6 (AER Root Port), 7 (AER Endpoint), or 8 (AER Bridge), or use type 9 (Generic Hardware Error Source). Firmware first error handling is possible if and only if Trusted Firmware is being used on arm64. Must be supplied if RAS support is provided by the platform. It is recommended this table be supplied.
HPET	Signature Reserved (signature == "HPET") **High Precision Event timer Table** x86 only table, will not be supported.
IBFT	Signature Reserved (signature == "IBFT") **iSCSI Boot Firmware Table** Microsoft defined table, support TBD.

Table	Usage for ARMv8 Linux
IORT	Signature Reserved (signature == "IORT") **Input Output Remapping Table** arm64 only table, required in order to describe IO topology, SMMUs, and GIC ITSs, and how those various components are connected together, such as identifying which components are behind which SMMUs/ITSs. This table will only be required on certain SBSA platforms (e.g., when using GICv3-ITS and an SMMU); on SBSA Level 0 platforms, it remains optional.
IVRS	Signature Reserved (signature == "IVRS") **I/O Virtualization Reporting Structure** x86_64 (AMD) only table, will not be supported.
LPIT	Signature Reserved (signature == "LPIT") **Low Power Idle Table** x86 only table as of ACPI 5.1; starting with ACPI 6.0, processor descriptions and power states on ARM platforms should use the DSDT and define processor container devices (_HID ACPI0010, Section 8.4, and more specifically 8.4.3 and 8.4.4).
MADT	Section 5.2.12 (signature == "APIC") **Multiple APIC Description Table** Required for arm64. Only the GIC interrupt controller structures should be used (types 0xA - 0xF).
MCFG	Signature Reserved (signature == "MCFG") **Memory-mapped ConFiGuration space** If the platform supports PCI/PCIe, an MCFG table is required.
MCHI	Signature Reserved (signature == "MCHI") **Management Controller Host Interface table** Optional, not currently supported.
MPST	Section 5.2.21 (signature == "MPST") **Memory Power State Table** Optional, not currently supported.
MSCT	Section 5.2.19 (signature == "MSCT") **Maximum System Characteristic Table** Optional, not currently supported.
MSDM	Signature Reserved (signature == "MSDM") **Microsoft Data Management table** Microsoft only table, will not be supported.
NFIT	Section 5.2.25 (signature == "NFIT") **NVDIMM Firmware Interface Table** Optional, not currently supported.
OEMx	Signature of "OEMx" only **OEM Specific Tables** All tables starting with a signature of "OEM" are reserved for OEM use. Since these are not meant to be of general use but are limited to very specific end users, they are not recommended for use and are not supported by the kernel for arm64.
PCCT	Section 14.1 (signature == "PCCT) **Platform Communications Channel Table** Recommend for use on arm64; use of PCC is recommended when using CPPC to control performance and power for platform processors.
PMTT	Section 5.2.21.12 (signature == "PMTT") **Platform Memory Topology Table** Optional, not currently supported.
PSDT	Section 5.2.11.3 (signature == "PSDT") **Persistent System Description Table** Obsolete table, will not be supported.

Table	Usage for ARMv8 Linux
RASF	Section 5.2.20 (signature == "RASF") **RAS Feature table** Optional, not currently supported.
RSDP	Section 5.2.5 (signature == "RSD PTR") **Root System Description PoinTeR** Required for arm64.
RSDT	Section 5.2.7 (signature == "RSDT") **Root System Description Table** Since this table can only provide 32-bit addresses, it is deprecated on arm64, and will not be used. If provided, it will be ignored.
SBST	Section 5.2.14 (signature == "SBST") **Smart Battery Subsystem Table** Optional, not currently supported.
SLIC	Signature Reserved (signature == "SLIC") **Software LIcensing table** Microsoft only table, will not be supported.
SLIT	Section 5.2.17 (signature == "SLIT") **System Locality distance Information Table** Optional in general, but required for NUMA systems.
SPCR	Signature Reserved (signature == "SPCR") **Serial Port Console Redirection table** Required for arm64.
SPMI	Signature Reserved (signature == "SPMI") **Server Platform Management Interface table** Optional, not currently supported.
SRAT	Section 5.2.16 (signature == "SRAT") **System Resource Affinity Table** Optional, but if used, only the GICC Affinity structures are read. To support arm64 NUMA, this table is required.
SSDT	Section 5.2.11.2 (signature == "SSDT") **Secondary System Description Table** These tables are a continuation of the DSDT; these are recommended for use with devices that can be added to a running system, but can also serve the purpose of dividing up device descriptions into more manageable pieces. An SSDT can only ADD to the ACPI namespace. It cannot modify or replace existing device descriptions already in the namespace. These tables are optional, however. ACPI tables should contain only one DSDT but can contain many SSDTs.
STAO	Signature Reserved (signature == "STAO") **_STA Override table** Optional, but only necessary in virtualized environments in order to hide devices from guest OSs.
TCPA	Signature Reserved (signature == "TCPA") **Trusted Computing Platform Alliance table** Optional, not currently supported, and may need changes to fully interoperate with arm64.
TPM2	Signature Reserved (signature == "TPM2") **Trusted Platform Module 2 table** Optional, not currently supported, and may need changes to fully interoperate with arm64.
UEFI	Signature Reserved (signature == "UEFI") **UEFI ACPI data table** Optional, not currently supported. No known use case for arm64, at present.
WAET	Signature Reserved (signature == "WAET") **Windows ACPI Emulated devices Table** Microsoft only table, will not be supported.

Table	Usage for ARMv8 Linux
WDAT	Signature Reserved (signature == "WDAT") **Watch Dog Action Table** Microsoft only table, will not be supported.
WDRT	Signature Reserved (signature == "WDRT") **Watch Dog Resource Table** Microsoft only table, will not be supported.
WPBT	Signature Reserved (signature == "WPBT") **Windows Platform Binary Table** Microsoft only table, will not be supported.
XENV	Signature Reserved (signature == "XENV") **Xen project table** Optional, used only by Xen at present.
XSDT	Section 5.2.8 (signature == "XSDT") **eXtended System Description Table** Required for arm64.

ACPI Objects

The expectations on individual ACPI objects that are likely to be used are shown in the list that follows; any object not explicitly mentioned below should be used as needed for a particular platform or particular subsystem, such as power management or PCI.

Name	Section	Usage for ARMv8 Linux
_CCA	6.2.17	This method must be defined for all bus masters on arm64 - there are no assumptions made about whether such devices are cache coherent or not. The _CCA value is inherited by all descendants of these devices so it does not need to be repeated. Without _CCA on arm64, the kernel does not know what to do about setting up DMA for the device. NB: this method provides default cache coherency attributes; the presence of an SMMU can be used to modify that, however. For example, a master could default to non-coherent, but be made coherent with the appropriate SMMU configuration (see Table 17 of the IORT specification, ARM Document DEN 0049B).
_CID	6.1.2	Use as needed, see also _HID.
_CLS	6.1.3	Use as needed, see also _HID.
_CPC	8.4.7.1	Use as needed, power management specific. CPPC is recommended on arm64.
_CRS	6.2.2	Required on arm64.
_CSD	8.4.2.2	Use as needed, used only in conjunction with _CST.
_CST	8.4.2.1	Low power idle states (8.4.4) are recommended instead of C-states.
_DDN	6.1.4	This field can be used for a device name. However, it is meant for DOS device names (e.g., COM1), so be careful of its use across OSes.
_DSD	6.2.5	To be used with caution. If this object is used, try to use it within the constraints already defined by the Device Properties UUID. Only in rare circumstances should it be necessary to create a new _DSD UUID. In either case, submit the _DSD definition along with any driver patches for discussion, especially when device properties are used. A driver will not be considered complete without a corresponding _DSD description. Once approved by kernel maintainers, the UUID or device properties must then be registered with the UEFI Forum; this may cause some iteration as more than one OS will be registering entries.
_DSM	9.1.1	Do not use this method. It is not standardized, the return values are not well documented, and it is currently a frequent source of error.
_GL	5.7.1	This object is not to be used in hardware reduced mode, and therefore should not be used on arm64.
_GLK	6.5.7	This object requires a global lock be defined; there is no global lock on arm64 since it runs in hardware reduced mode. Hence, do not use this object on arm64.
_GPE	5.3.1	This namespace is for x86 use only. Do not use it on arm64.
_HID	6.1.5	This is the primary object to use in device probing, though _CID and _CLS may also be used.
_INI	6.5.1	Not required, but can be useful in setting up devices when UEFI leaves them in a state that may not be what the driver expects before it starts probing.
_LPI	8.4.4.3	Recommended for use with processor definitions (_HID ACPI0010) on arm64. See also _RDI.

Name	Section	Usage for ARMv8 Linux
_MLS	6.1.7	Highly recommended for use in internationalization.
_OFF	7.2.2	It is recommended to define this method for any device that can be turned on or off.
_ON	7.2.3	It is recommended to define this method for any device that can be turned on or off.
_OS	5.7.3	This method will return "Linux" by default (this is the value of the macro ACPI_OS_NAME on Linux). The command line parameter acpi_os=<string> can be used to set it to some other value.
_OSC	6.2.11	This method can be a global method in ACPI (i.e., _SB._OSC), or it may be associated with a specific device (e.g., _SB.DEV0._OSC), or both. When used as a global method, only capabilities published in the ACPI specification are allowed. When used as a device-specific method, the process described for using _DSD MUST be used to create an _OSC definition; out-of-process use of _OSC is not allowed. That is, submit the device-specific _OSC usage description as part of the kernel driver submission, get it approved by the kernel community, then register it with the UEFI Forum.
_OSI	5.7.2	Deprecated on ARM64. As far as ACPI firmware is concerned, _OSI is not to be used to determine what sort of system is being used or what functionality is provided. The _OSC method is to be used instead.
_PDC	8.4.1	Deprecated, do not use on arm64.
_PIC	5.8.1	The method should not be used. On arm64, the only interrupt model available is GIC.
_PR	5.3.1	This namespace is for x86 use only on legacy systems. Do not use it on arm64.
_PRT	6.2.13	Required as part of the definition of all PCI root devices.
_PRx	7.3.8-11	Use as needed; power management specific. If _PR0 is defined, _PR3 must also be defined.
_PSx	7.3.2-5	Use as needed; power management specific. If _PS0 is defined, _PS3 must also be defined. If clocks or regulators need adjusting to be consistent with power usage, change them in these methods.
_RDI	8.4.4.4	Recommended for use with processor definitions (_HID ACPI0010) on arm64. This should only be used in conjunction with _LPI.
_REV	5.7.4	Always returns the latest version of ACPI supported.
_SB	5.3.1	Required on arm64; all devices must be defined in this namespace.
_SLI	6.2.15	Use is recommended when SLIT table is in use.
_STA	6.3.7, 7.2.4	It is recommended to define this method for any device that can be turned on or off. See also the STAO table that provides overrides to hide devices in virtualized environments.
_SRS	6.2.16	Use as needed; see also _PRS.
_STR	6.1.10	Recommended for conveying device names to end users; this is preferred over using _DDN.
_SUB	6.1.9	Use as needed; _HID or _CID are preferred.
_SUN	6.1.11	Use as needed, but recommended.
_SWS	7.4.3	Use as needed; power management specific; this may require specification changes for use on arm64.
_UID	6.1.12	Recommended for distinguishing devices of the same class; define it if at all possible.

ACPI Event Model

Do not use GPE block devices; these are not supported in the hardware reduced profile used by arm64. Since there are no GPE blocks defined for use on ARM platforms, ACPI events must be signaled differently.

There are two options: GPIO-signaled interrupts (Section 5.6.5), and interrupt-signaled events (Section 5.6.9). Interrupt-signaled events are a new feature in the ACPI 6.1 specification. Either - or both - can be used on a given platform, and which to use may be dependent of limitations in any given SoC. If possible, interrupt-signaled events are recommended.

ACPI Processor Control

Section 8 of the ACPI specification changed significantly in version 6.0. Processors should now be defined as Device objects with _HID ACPI0007; do not use the deprecated Processor statement in ASL. All multiprocessor systems should also define a hierarchy of processors, done with Processor Container Devices (see Section 8.4.3.1, _HID ACPI0010); do not use processor aggregator devices (Section 8.5) to describe processor topology. Section 8.4 of the specification describes the semantics of these object definitions and how they interrelate.

Most importantly, the processor hierarchy defined also defines the low power idle states that are available to the platform, along with the rules for determining which processors can be turned on or off and the circumstances that control that. Without this information, the processors will run in whatever power state they were left in by UEFI.

Note too, that the processor Device objects defined and the entries in the MADT for GICs are expected to be in synchronization. The _UID of the Device object must correspond to processor IDs used in the MADT.

It is recommended that CPPC (8.4.5) be used as the primary model for processor performance control on arm64. C-states and P-

states may become available at some point in the future, but most current design work appears to favor CPPC.

Further, it is essential that the ARMv8 SoC provide a fully functional implementation of PSCI; this will be the only mechanism supported by ACPI to control CPU power state. Booting of secondary CPUs using the ACPI parking protocol is possible, but discouraged, since only PSCI is supported for ARM servers.

ACPI System Address Map Interfaces

In Section 15 of the ACPI specification, several methods are mentioned as possible mechanisms for conveying memory resource information to the kernel. For arm64, we will only support UEFI for booting with ACPI, hence the UEFI GetMemoryMap() boot service is the only mechanism that will be used.

ACPI Platform Error Interfaces (APEI)

The APEI tables supported are described above.

APEI requires the equivalent of an SCI and an NMI on ARMv8. The SCI is used to notify the OSPM of errors that have occurred but can be corrected and the system can continue correct operation, even if possibly degraded. The NMI is used to indicate fatal errors that cannot be corrected, and require immediate attention.

Since there is no direct equivalent of the x86 SCI or NMI, arm64 handles these slightly differently. The SCI is handled as a high priority interrupt; given that these are corrected (or correctable) errors being reported, this is sufficient. The NMI is emulated as the highest priority interrupt possible. This implies some caution must be used since there could be interrupts at higher privilege levels or even interrupts at the same priority as the emulated NMI. In Linux, this should not be the case but one should be aware it could happen.

ACPI Objects Not Supported on ARM64

While this may change in the future, there are several classes of objects that can be defined, but are not currently of general interest to ARM servers. Some of these objects have x86 equivalents, and may actually make sense in ARM servers. However, there is either no hardware available at present, or there may not even be a non-ARM implementation yet. Hence, they are not currently supported.

The following classes of objects are not supported:

- Section 9.2: ambient light sensor devices
- Section 9.3: battery devices
- Section 9.4: lids (e.g., laptop lids)
- Section 9.8.2: IDE controllers
- Section 9.9: floppy controllers
- Section 9.10: GPE block devices
- Section 9.15: PC/AT RTC/CMOS devices
- Section 9.16: user presence detection devices
- Section 9.17: I/O APIC devices; all GICs must be enumerable via MADT
- Section 9.18: time and alarm devices (see 9.15)
- Section 10: power source and power meter devices
- Section 11: thermal management
- Section 12: embedded controllers interface
- Section 13: SMBus interfaces

This also means that there is no support for the following objects:

Name	Section	Name	Section
_ALC	9.3.4	_FDM	9.10.3
_ALI	9.3.2	_FIX	6.2.7
_ALP	9.3.6	_GAI	10.4.5
_ALR	9.3.5	_GHL	10.4.7
_ALT	9.3.3	_GTM	9.9.2.1.1
_BCT	10.2.2.10	_LID	9.5.1
_BDN	6.5.3	_PAI	10.4.4
_BIF	10.2.2.1	_PCL	10.3.2
_BIX	10.2.2.1	_PIF	10.3.3
_BLT	9.2.3	_PMC	10.4.1
_BMA	10.2.2.4	_PMD	10.4.8
_BMC	10.2.2.12	_PMM	10.4.3
_BMD	10.2.2.11	_PRL	10.3.4
_BMS	10.2.2.5	_PSR	10.3.1
_BST	10.2.2.6	_PTP	10.4.2

Name	Section	Name	Section
_BTH	10.2.2.7	_SBS	10.1.3
_BTM	10.2.2.9	_SHL	10.4.6
_BTP	10.2.2.8	_STM	9.9.2.1.1
_DCK	6.5.2	_UPD	9.16.1
_EC	12.12	_UPP	9.16.2
_FDE	9.10.1	_WPC	10.5.2
_FDI	9.10.2	_WPP	10.5.3