Intel® Platform Innovation Framework for EFI ACPI Table Storage Specification

Draft for Review

Version 0.9

April 1, 2004

Intel order number xxxxxx-001

Revision History

Revision	Revision History	Date
0.9	First public release.	4/1/04

Contents

1 Introduction

Overview

This specification describes a format for storing ACPI tables in an implementation of the Intel® Platform Innovation Framework for EFI (hereafter referred to as the "Framework"). The *Advanced Configuration and Power Interface Specification* (hereafter referred to as the "ACPI specification") defines a set of tables that describe platform configuration and power management. The platform Driver Execution Environment (DXE) driver must provide these tables to the operating system. These tables are largely static in nature and platform specific. This specification does the following:

- Describes the basic components of ACPI table storage
- Provides code definitions for ACPI table storage that are architecturally required by the *Intel® Platform Innovation Framework for EFI Architecture Specification*

See Industry Specifications in the Framework master help system for the URL for the ACPI specification.

Scope

This specification provides a design for storing ACPI tables that comply with the ACPI specification, revisions 1.0b and 2.0, in a Framework-compliant firmware device. This method relies on existing firmware volume (FV) and Firmware File System (FFS) designs. See the following specifications for more information on FV and FFS designs:

- *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*
- *Intel® Platform Innovation Framework for EFI Firmware File System Specification*

Conventions Used in This Document

This document uses the typographic and illustrative conventions described below.

Data Structure Descriptions

Intel® processors based on 32-bit Intel® architecture (IA-32) are “little endian” machines. This distinction means that the low-order byte of a multibyte data item in memory is at the lowest address, while the high-order byte is at the highest address. Processors of the Intel® Itanium® processor family may be configured for both “little endian” and “big endian” operation. All implementations designed to conform to this specification will use “little endian” operation.

In some memory layout descriptions, certain fields are marked *reserved*. Software must initialize such fields to zero and ignore them when read. On an update operation, software must preserve any reserved field.

The data structures described in this document generally have the following format:

STRUCTURE NAME:	The formal name of the data structure.
Summary:	A brief description of the data structure.
Prototype:	A “C-style” type declaration for the data structure.
Parameters:	A brief description of each field in the data structure prototype.
Description:	A description of the functionality provided by the data structure, including any limitations and caveats of which the caller should be aware.
Related Definitions:	The type declarations and constants that are used only by this data structure.

Pseudo-Code Conventions

Pseudo code is presented to describe algorithms in a more concise form. None of the algorithms in this document are intended to be compiled directly. The code is presented at a level corresponding to the surrounding text.

In describing variables, a *list* is an unordered collection of homogeneous objects. A *queue* is an ordered list of homogeneous objects. Unless otherwise noted, the ordering is assumed to be First In First Out (FIFO).

Pseudo code is presented in a C-like format, using C conventions where appropriate. The coding style, particularly the indentation style, is used for readability and does not necessarily comply with an implementation of the *Extensible Firmware Interface Specification*.

Typographic Conventions

This document uses the typographic and illustrative conventions described below:

Plain text	The normal text typeface is used for the vast majority of the descriptive text in a specification.
Plain text (blue)	In the online help version of this specification, any plain text that is underlined and in blue indicates an active link to the cross-reference. Click on the word to follow the hyperlink. Note that these links are *not* active in the PDF of the specification.
Bold	In text, a **Bold** typeface identifies a processor register name. In other instances, a **Bold** typeface can be used as a running head within a paragraph.
Italic	In text, an *Italic* typeface can be used as emphasis to introduce a new term or to indicate a manual or specification name.
`BOLD Monospace`	Computer code, example code segments, and all prototype code segments use a **`BOLD Monospace`** typeface with a dark red color. These code listings normally appear in one or more separate paragraphs, though words or segments can also be embedded in a normal text paragraph.

Bold Monospace	In the online help version of this specification, words in a **Bold Monospace** typeface that is underlined and in blue indicate an active hyperlink to the code definition for that function or type definition. Click on the word to follow the hyperlink. Note that these links are *not* active in the PDF of the specification. Also, these inactive links in the PDF may instead have a **Bold Monospace** appearance that is underlined but in dark red. Again, these links are not active in the PDF of the specification.
Italic Monospace	In code or in text, words in *Italic Monospace* indicate placeholder names for variable information that must be supplied (i.e., arguments).
`Plain Monospace`	In code, words in a `Plain Monospace` typeface that is a dark red color but is not bold or italicized indicate pseudo code or example code. These code segments typically occur in one or more separate paragraphs.
text text text	In the PDF of this specification, text that is highlighted in yellow indicates that a change was made to that text since the previous revision of the PDF. The highlighting indicates only that a change was made since the previous version; it does not specify what changed. If text was deleted and thus cannot be highlighted, a note in red and highlighted in yellow (that looks like *(Note: text text text.)*) appears where the deletion occurred.

See the master Framework glossary in the Framework Interoperability and Component Specifications help system for definitions of terms and abbreviations that are used in this document or that might be useful in understanding the descriptions presented in this document.

See the master Framework references in the Interoperability and Component Specifications help system for a complete list of the additional documents and specifications that are required or suggested for interpreting the information presented in this document.

The Framework Interoperability and Component Specifications help system is available at the following URL:

http://www.intel.com/technology/framework/spec.htm

2
Design Discussion

ACPI Table Storage Terms

The following terms are used throughout this specification or else are not widely used in other Framework specifications. See the master Glossary in the master help system for definitions of additional terms.

ACPI

Advanced Configuration and Power Interface.

APIC

Advanced Programmable Interrupt Controller.

BOOT

Simple Boot Flag Table.

DBGP

Debug Port Table.

DSDT

Differentiated System Description Table.

ECDT

Embedded Controller Boot Resources Table.

ETDT

Event Timer Description Table.

FACS

Firmware ACPI Control Structure.

FADT

Fixed ACPI Description Table.

FFS

Firmware File System.

FV

Firmware volume.

MADT

Multiple APIC Description Table.

OEM

Original equipment manufacturer.

OEMx

OEM Specific Information Tables.

PSDT

Persistent System Description Table.

SBST

Smart Battery Specification Table.

SLIT

System Locality Information Table.

SPCR

Serial Port Console Redirection Table.

SRAT

Static Resource Affinity Table.

SPMI

Server Platform Management Interface Table.

SSDT

Secondary System Description Table.

Overview

This specification describes how ACPI tables are stored by the firmware. Each table is stored in a single raw data section that complies with the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*. The sum of the sections is a single Firmware File System (FFS) file with the free-form file type, **`EFI_FV_FILETYPE_FREEFORM`**. This file may be stored in any firmware volume (FV) that is accessible during the DXE phase. The file is not required to be accessible during other Framework phases.

The ACPI platform driver will load the tables from the firmware. The file can be located using the **`EFI_FIRMWARE_VOLUME_PROTOCOL`** services; see the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification* for more details. The ACPI platform driver will add tables using the **`EFI_ACPI_SUPPORT_PROTOCOL`** services; see the *Intel® Platform Innovation Framework for EFI ACPI Specification* for more details.

Rationale

Because the ACPI tables are largely static, in the interest of code space and driver flexibility, the ACPI tables may be built prior to their use. This specification defines a nonvolatile file format for storing the tables.

Requirements

ACPI tables will be stored in an FV-compliant manner. The *Intel® Platform Innovation Framework for EFI Firmware Volume Specification* describes file format and construction. The ACPI platform driver requires information about the file and format to load the initial tables. This specification describes one possible format for storing the initial tables.

ACPI tables will be stored in an ACPI 1.0b– or 2.0–compliant format. The tables will be complete except for clearly identified fields that must be updated prior to booting the operating system.

ACPI Table Storage

Introduction

ACPI tables are defined in source code. A special utility will extract the tables from the object code that is created by the compiler. The extracted code will be wrapped in a raw data section and concatenated into a single FV file. See the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification* for more information on raw sections.

ACPI Table Storage File

The ACPI table storage file is fully FFS compliant. The file is a number of sections of type **`EFI_SECTION_RAW`**. Sections can be encapsulated as defined in the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*. The GUID that identifies the file as an ACPI table storage file is defined in ACPI Table Storage File in Code Definitions.

FADT Section

The Fixed ACPI Description Table (FADT) section contains a raw data section header, as defined in the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*, followed by an FADT structure, which is defined in the ACPI 2.0 specification. No padding is needed to meet section alignment because the section header and FADT structure sizes are both 4-byte multiples. The FADT section can be identified using the Signature field of the FADT structure. The following fields in the FADT structure may require updating during the DXE phase:

- Checksum
- FIRMWARE_CTRL
- DSDT
- X_FIRMWARE_CTRL
- X_DSDT

Exactly one FADT section is required in an ACPI table storage file.

FACS Section

The Firmware ACPI Control Structure (FACS) section contains a raw data section header, as defined in the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*, followed by an FACS structure, which is defined in the ACPI 2.0 specification. No padding is needed to meet section alignment because the section header and FACS structure sizes are both 4-byte multiples. The FACS section can be identified using the Signature field of the FACS structure. The following fields in the FACS structure may require updating during the DXE phase:

- Hardware Signature

Exactly one FACS section is required in an ACPI table storage file.

DSDT Section

The Differentiated System Description Table (DSDT) section contains a raw data section header, as defined in the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*, followed by a DSDT structure, which is defined in the ACPI 2.0 specification. The section may contain additional padding to satisfy section alignment requirements. The DSDT section can be identified using the Signature field of the DSDT structure. The following fields in the DSDT structure may require updating during the DXE phase:

- Length
- Checksum

Exactly one DSDT section is required in an ACPI table storage file.

SSDT Section

The Secondary System Description Table (SSDT) section contains a raw data section header, as defined in the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*, followed by an SSDT structure, which is defined in the ACPI 2.0 specification. The section may contain additional padding to satisfy section alignment requirements. The SSDT section can be identified using the Signature field of the SSDT structure. The following fields in the SSDT structure may require updating during the DXE phase:

- Checksum

Any number of SSDT sections may be present in an ACPI table storage file.

PSDT Section

The Persistent System Description Table (PSDT) is obsolete in ACPI 2.0. A PSDT section will look and be treated like a Secondary System Description Table (SSDT) section.

MADT Section

The Multiple APIC Description Table (MADT) section contains a raw data section header, as defined in the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*, followed by an MADT structure, which is defined in the ACPI 2.0 specification. No padding is needed to meet section alignment because the section header and MADT structure sizes are both 4-byte multiples. The MADT section can be identified using the Signature field of the MADT structure. The following fields in the MADT structure may require updating during the DXE phase:

- Checksum

If a MADT section is required, only one MADT section is allowed in an ACPI table storage file.

Other ACPI Sections

Other tables that are defined or referenced in the ACPI 2.0 specification may also be included as long as they contain a standard ACPI 2.0 system description table header. The following are some possible tables that could be included:

- Simple Boot Flag Table (BOOT)
- Debug Port Table (DBGP)
- Embedded Controller Boot Resources Table (ECDT)
- Event Timer Description Table (ETDT)
- OEM Specific Information Tables (OEMx)
- Smart Battery Specification Table (SBST)
- System Locality Information Table (SLIT)
- Serial Port Console Redirection Table (SPCR)
- Static Resource Affinity Table (SRAT)
- Server Platform Management Interface Table (SPMI)

The section must begin with a raw data section header, as defined in the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*, followed immediately by the table. The entire section must be padded to ensure the section length is a multiple of four bytes. Core and platform ACPI drivers must update the Checksum field if required. All other table fields will remain unchanged. The ACPI platform driver must expose these tables to the ACPI support driver and copy them when requested. This specification places no additional limits to the number and type of other ACPI sections.

3
Code Definitions

Introduction

This section contains the basic definitions for storing ACPI tables in a Framework implementation. The following data types are defined in this section:

- **EFI_ACPI_TABLE_STORAGE_GUID**

ACPI Table Storage File

EFI_ACPI_TABLE_STORAGE_GUID

Summary

The ACPI table storage file is fully FFS compliant. The file is a number of sections of type **EFI_SECTION_RAW**. Sections can be encapsulated as defined in the *Intel® Platform Innovation Framework for EFI Firmware Volume Specification*. Following is the GUID that identifies the file as an ACPI table storage file.

GUID

```
#define EFI_ACPI_TABLE_STORAGE_GUID \
{ 0x7e374e25, 0x8e01, 0x4fee, 0x87, 0xf2, 0x39, 0xc, 0x23, 0xc6,
0x6, 0xcd }
```