Device Properties UUID For _DSD

Oct 2015

Revision 2.0

Contents

Revision history

Revision Number	Description	Revision Date
<1.0>	First revision	July 16, 2014
<2.0>	Added more clarity on rules of usage, and improved example	Oct 20, 2014

1 Introduction

This document specifies the data format associated with UUID

daffd814-6eba-4d8c-8a91-bc9bbf4aa301

(Device Properties UUID) for the _DSD (Device Specific Data) ACPI device configuration object.

1.1 Terms

The following terms are used throughout this document to describe varying aspects of input localization:

ACPI

Advanced Configuration and Power Interface specification.

Device

Hardware component or set of interrelated hardware registers.

Device ID

Plug and Play ID or ACPI ID of a device.

GUID

Globally Unique Identifier. A 128-bit value used to uniquely name entities. A unique GUID can be generated by an individual without the help of a centralized authority. This allows the generation of names that will never conflict, even among multiple, unrelated parties.

UUID

Universal Unique Identifier, GUID.

1.2 Conventions used in this document

1.2.1 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.

`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.

2 Device Properties UUID

2.1 Data Format Definition

The Device Properties UUID

daffd814-6eba-4d8c-8a91-bc9bbf4aa301

defines the data format for the Package (Data Structure) immediately following it as a list of Packages of length 2 (Properties) where the first element of each contained Package (Key) must be a String and the second element of it (Value) must be:

- an Integer,
- a String,
- a Reference, or
- a Package consisting entirely of Integer, String, or Reference objects (and specifically not containing a nested Package).

The list of valid Keys, and the format and interpretation of the corresponding Values, depends on the PNP or ACPI device ID (e.g. _HID) of the Device containing the _DSD.

For instance, the PNP device ID returned by _HID for Device object MDEV below will determine the list of valid Keys and the corresponding Value data formats for that Device object's _DSD.

```
Device (MDEV) {
    Name (_HID, "PNP####")

    Name (_DSD, Package () {
        ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
        Package () {
            Package (2) {...},    // Property 1
            Package (2) {...},    // Property 2
            ...
            Package (2) {...}     // Property n
        }
    })

    ...
}
```

In the context of the Device Properties UUID each Property is a characteristic of the hardware itself or the way it is used in the system in which it is incorporated, as opposed to software configuration data. As such, the list of valid property Keys and Value data formats associated with them must be defined by the device vendor in a way that is independent of the firmware interface to be used on any given platform.

Multiple Properties with the same Key in a single Data Structure associated with the Device Properties UUID are not permitted.

2.2 _HID / _CID Rules for Device Properties UUID

If a new version of a previously existing hardware component is developed, it usually will contain features that were not present in the previous version. The firmware for a platform containing it may still return the old device ID from _HID, but use a new _DSD including the Properties representing the new features. In that case, an existing driver for that component may not know about the new features, but it simply will not refer to the Properties representing them and will continue to work as before. Conversely, the driver may know about the new version of the hardware and will try to use Properties representing the new features in it, but the platform firmware may contain a _DSD that doesn't list those Properties for the device. In that case, the driver will discover that the new Properties are not present and will not try to use the hardware features represented by them.

Either the _HID or the _CID may be used to determine which driver to bind to a device and the driver is then responsible for using only the Properties appropriate for the device ID it has been matched against. If the hardware is different enough as to require Properties incompatible with those defined by another ID, a new ID should be allocated. Moreover, specific device IDs may require _DSD with the Device Properties UUID and with certain minimum set of Property Keys to be present. In those cases, if the _DSD cannot return all of the Properties required by the existing device ID, it is necessary to allocate a new ID for the device. Then, that new ID cannot be regarded as compatible with the previously existing one.

With respect to _CID, if the OS cannot match a driver to the _HID, it may match one based on the device IDs listed by _CID. The presence of _CID indicates that the Data Structure immediately following the Device Properties UUID in the _DSD under the given device object provides all of the necessary Properties defined for each of the device IDs listed by the _CID. In that case, every given Property Key may only be used once, so the Value data type associated with it must be suitable for all of the device IDs returned by _CID. It follows that if each device ID returned by _CID is compatible with the one returned by _HID, they are also compatible with each other.

2.3 Device Properties UUID Rules

Properties described in a Device Properties UUID _DSD are intended to be used in addition to, and not instead of, the existing mechanisms defined by the ACPI specification. For this reason, as a rule, Device Properties should only be used if the ACPI specification does not make direct provisions for handling the underlying use case. ACPI provides a number of generic interfaces to operating systems kernels, and these should continue to function without requiring the kernels to directly parse a device properties _DSD. For example properties that describe how to turn voltage regulators, or clocks, on and off, should not be used, as these should be managed via power resource _ON/_OFF methods, or device _PSx methods.

It is not permitted to use Properties in a data structure associated with the Device Properties UUID to provide the OSPM and device drivers with the same information that can be provided as device resources via _CRS (Current Resource Settings). In case of any conflicts between such Properties and the information returned by _CRS, the latter always takes precedence.

Properties of a Device may depend on the order of the resources provided via _CRS, or the order of the resources of a particular type. For instance, a Property may identify an IRQ via an index in to the IRQ resources provided via _CRS, rather than providing an absolute IRQ number. This avoids duplication between the Properties and _CRS, making it easier to change the resources of a Device in one place.

2.4 Examples

2.4.1 Example valid Property representations

The following examples illustrate valid Property Value data types for the Device Properties UUID.

```
Package (2) {"length", 16}
Package (2) {"device", \_SB.FOO.BAZ}
Package (2) {"sizes", Package (3) {16, 32, 0}}
Package (2) {"labels", Package (4) {"foo", _SB.FOO, "bar", _SB.BAR}}
Package (2) {"default-state", "on"}
```

2.4.2 _DSD dependency on _CRS

The following example illustrates a dependency of Properties returned by _DSD (with the Device Properties UUID) on device resources returned by _CRS. In this particular case, the “gpios” Properties returned by the _DSD for devices LEDH and LEDM contain references to GpioIo resources in the _CRS of device LEDS.

Each of these references consists of a path to the device object containing the _CRS in question and three integer numbers. The first two of these numbers are indexes to the _CRS content. Specifically, they are the index of the GpioIo resource and the index of the pin in that resource’s GPIO pin list pointed to by the given reference, respectively. The fourth number is an additional parameter to be consumed by the driver of the LEDS device.

This means that the “gpios” Property of device LEDH in this example points to the first (index 0) GpioIo resource in the _CRS of device LEDS and to the first (index 0) pin in its GPIO pin list (pin 10). In turn, the “gpios” Property of device LEDM points to the second (index 1) GpioIo resource in the _CRS of device LEDS and to the first (index 0) pin in its GPIO pin list (pin 11).

```
Scope (\_SB.PCI0.LPC)
{
  Device (LEDS)
  {
    Name (_HID, "PNP####")

    Name (_CRS, ResourceTemplate ()
    {
      GpioIo (Exclusive, PullDown, 0, 0, IoRestrictionOutputOnly,
              "\\_SB.PCI0.LPC", 0, ResourceConsumer,,) { 10 }
      GpioIo (Exclusive, PullUp, 0, 0, IoRestrictionInputOnly,
              "\\_SB.PCI0.LPC", 0, ResourceConsumer,,) { 11 }
    })

    Device (LEDH)
    {
      Name (_HID, "PNP####")

      Name (_DSD, Package () {
```

```
            ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
            Package () {
              Package (2) {"label", "Heartbeat"},
              Package (2) {"gpios", Package (4) {
                                      \_SB.PCI0.LPC.LEDS, 0, 0, 1
                                    }},
              Package (2) {"default-trigger", "heartbeat"},
              Package (2) {"default-state", "on"},
              Package (2) {"retain-state-suspended", 1},
            }
          })
        }

        Device (LEDM)
        {
          Name (_HID, "PNP####")

          Name (_DSD, Package () {
            ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
            Package () {
              Package (2) {"label", "MMC0 Activity"},
              Package (2) {"gpios", Package (4) {
                                      \_SB.PCI0.LPC.LEDS, 1, 0, 1
                                    }},
              Package (2) {"default-trigger", "mmc0"},
              Package (2) {"default-state", "on"},
              Package (2) {"retain-state-suspended", 1},
            }
          })
        }
      }
    }
```