

4.5. **_OSC – A Mechanism for Exposing PCI Express Capabilities Supported by an Operating System**

_OSC is an object that is used by OSPM to query the capabilities of a device (as defined in ACPI) and to communicate to the platform the feature support or capabilities provided by a device's driver. This object is a child object of the device in the ACPI namespace. Device specific objects are evaluated after _OSC invocation. For a complete description of _OSC method, refer to the ACPI 3.0 Specification. The ACPI 3.0 Specification can be found at <http://www.acpi.info/>.

4.5.1. **_OSC Interface for PCI Host Bridge Devices**

The _OSC interface defined in this section applies only to “Host Bridge” ACPI devices that originate PCI, PCI-X, or PCI Express hierarchies. These ACPI devices must have a _HID of (or a _CID including) either EISAID(“PNP0A03”) or EISAID(“PNP0A08”).

For a host bridge device that originates a PCI Express hierarchy, the _OSC interface defined in this section is required. For a host bridge device that originates a PCI/PCI-X bus hierarchy, inclusion of an _OSC object is optional.

The _OSC interface for a PCI/PCI-X/PCI Express hierarchy is identified by the Universal Unique Identifier (UUID) 33db4d5b-1ff7-401c-9657-7441c03dd766. A revision ID of 1 encompasses fields defined in this section of this revision of this specification comprised of three DWORDs, including the first DWORD described by the generic ACPI definition of _OSC.

The first DWORD in the _OSC Capabilities Buffer contains bits that are generic to _OSC. These include status and error information.

The second DWORD in the _OSC Capabilities Buffer is the Support Field. Bits defined in the Support Field provide information regarding operating system supported features. Contents in the Support Field are passed one way; the operating system will disregard any changes to this field when returned.

The third DWORD in the _OSC Capabilities Buffer is the Control Field. Bits defined in the Control Field are used to submit requests by the operating system for control/handling of the associated feature, typically (but not excluded to) those features that utilize native interrupts or events handled by an operating system-level driver. If any bits in the Control Field are returned cleared (masked to zero) by the _OSC control method, the respective feature is designated unsupported by the platform and must not be enabled by the operating system. Some of these features may be controlled by platform firmware prior to operating system boot or during runtime for a legacy operating system, while others may be disabled/inoperative until native operating system support is available. System firmware must only mask a Control Field bit to zero if it has explicit knowledge that the feature will not work properly under native operating system control, due to platform errata or other incompatibilities. Since _OSC applies to the entire hierarchy originated by a PCI Host Bridge, and system firmware cannot generally comprehend the features and capabilities of all devices that may be hot-plugged into a system, lack of explicit support of a feature in the system firmware is not a reason to mask a Control Field bit to zero. For example, unless system firmware

has knowledge that the system contains hardware that does not work properly with SHPC or PCI Express Native Hot Plug software, it must set both the PCI Express Native Hot Plug control and SHPC Native Hot Plug control bits, even if the system does not explicitly support hot plug. It is the operating system’s responsibility to determine whether a slot in the hierarchy is hot pluggable by examining the status of the slots based on the PCI Express Base Specification and the PCI SHPC Specification.

For the above reason, system firmware must always assume that PCI-X features are supported beneath a PCI Express hierarchy unless it has explicit knowledge to the contrary, and must not mask _OSC Control Field bits that apply only to PCI/PCI-X to zero in PCI Express hierarchies. However, if a bit in the _OSC Control Field applies to PCI Express, system firmware must mask this bit to zero in a PCI/PCI-X hierarchy.

If the _OSC control method is absent from the scope of a host bridge device, then the operating system must not enable or attempt to use any features defined in this section for the hierarchy originated by the host bridge. Doing so could contend with platform firmware operations or produce undesired results.

It is recommended that a machine with multiple host bridge devices should report the same capabilities for all host bridges of the same type and also negotiate control of the features described in the Control Field in the same way for all host bridges of the same type. That is, for a machine with multiple host bridge devices supporting both PCI/PCI-X and PCI Express hierarchies, this recommendation only applies to the same type of host bridge devices, PCI/PCI-X host bridge devices can have one capabilities setting and PCI Express hot bridge devices can have another.

Table Error! No text of specified style in document.-1: Interpretation of the _OSC Support Field

Support Field Bit Offset	Interpretation
0	<p>Extended PCI Config operation regions supported</p> <p>The operating system sets this bit to 1 if it supports ASL accesses through PCI Config operation regions to extended configuration space (offsets greater than 0xFF). Otherwise, the operating system sets this bit to 0.</p>
1	<p>Active State Power Management supported</p> <p>The operating system sets this bit to 1 if it natively supports configuration of Active State Power Management registers in PCI Express devices. Otherwise, the operating system sets this bit to 0.</p>
2	<p>Clock Power Management Capability supported</p> <p>The operating system sets this bit to 1 if it supports the Clock Power Management Capability and will enable this feature during a native hot plug insertion event if supported by the newly added device. Otherwise, the operating system sets this bit to 0.</p> <p>Note: The Clock Power Management Capability is defined in an errata to the <i>PCI Express Base Specification, Revision 1.0</i>.</p>

Support Field Bit Offset	Interpretation
3	<p>PCI Segment Groups supported</p> <p>The operating system sets this bit to 1 if it supports PCI Segment Groups as defined by the _SEG object and access to the configuration space of devices in PCI Segment Groups as described by this specification. Otherwise, the operating system sets this bit to 0.</p>
4	<p>MSI supported</p> <p>The operating system sets this bit to 1 if it supports configuration of devices to generate message-signaled interrupts, either through the MSI Capability or the MSI-X Capability. Otherwise, the operating system sets this bit to 0.</p>
5-31	Reserved

Table Error! No text of specified style in document.-2: Interpretation of the _OSC Control Field, Passed in via Arg3

Control Field Bit Offset	Interpretation	Interdependencies
0	<p>PCI Express Native Hot Plug control</p> <p>The operating system sets this bit to 1 to request control over PCI Express native hot plug. If the operating system successfully receives control of this feature, it must track and update the status of hot plug slots and handle hot plug events as described in the PCI Express Base Specification (including determining whether the associated slot(s) support hot plug).</p>	<p>Yes, refer to 4.5.2.4</p>
1	<p>SHPC Native Hot Plug control</p> <p>The operating system sets this bit to 1 to request control over PCI/PCI-X Standard Hot-Plug Controller (SHPC) hot plug. If the operating system successfully receives control of this feature, it must track and update the status of hot plug slots and handle hot plug events as described in the SHPC Specification (including determining whether the associated slot(s) support hot plug).</p>	
2	<p>PCI Express Native Power Management Events control</p> <p>The operating system sets this bit to 1 to request control over PCI Express native power management event interrupts (PMEs). If the operating system successfully receives control of this feature, it must handle power management events as described in the PCI Express Base Specification.</p>	<p>Yes, refer to 4.5.2.4</p>
3	<p>PCI Express Advanced Error Reporting control</p> <p>The operating system sets this bit to 1 to request control over PCI Express Advanced Error Reporting. If the operating system successfully receives control of this feature, it must handle error reporting through the Advanced Error Reporting Capability as described in the PCI Express Base Specification; further, the operating system retains control of AER across power transitions for S1, S2, S3 system power states.</p>	<p>Yes, refer to 4.5.2.4</p>
4	<p>PCI Express Capability Structure control</p> <p>The operating system sets this bit to 1 to request control over the PCI Express Capability structures (standard and extended) defined in the <i>PCI Express Base Specification, Revision 1.1</i>. These capability structures are the PCI Express Capability, the Virtual Channel Extended Capability, the Power Budgeting Extended Capability, the Advanced Error Reporting Extended Capability, and the Serial Number Extended Capability. If the operating system successfully receives control of this feature, it is responsible for configuring the registers in all PCI Express Capabilities in a manner that complies with the PCI Express Base Specification. Additionally, the operating system is responsible for saving and restoring all PCI Express Capability register settings across power transitions to and from S1, S2, S3 system power states when register context may have been lost.</p>	<p>Yes, refer to 4.5.2.4</p>

Control Field Bit Offset	Interpretation	Interdependencies
5-31	Reserved	

Table Error! No text of specified style in document.-3: Interpretation of the _OSC Control Field, Returned Value

Control Field Bit Offset	Interpretation	Interdependencies
0	<p>PCI Express Native Hot Plug control</p> <p>The firmware sets this bit to 1 to grant control over PCI Express native hot plug interrupts. If firmware allows the operating system control of this feature, it means that the firmware has made the necessary configurations to ensure that all hot plug events are routed to device interrupts as described in the PCI Express Base Specification (e.g., switched from SCI/GPE interrupt mechanism used in the legacy hot plug mechanisms), regardless of whether there are hot pluggable slots down the hierarchy. It is the operating system’s responsibility to determine whether a slot is hot pluggable by examining the status of the slots based on the PCI Express Base Specification. Additionally, after control is transferred to the operating system, firmware must not update the state of hot plug slots, including the state of the indicators and power controller. If control of this feature was requested and denied or was not requested, firmware returns this bit set to 0.</p>	<p>Yes, refer to 4.5.2.4</p>
1	<p>SHPC Native Hot Plug control</p> <p>The firmware sets this bit to 1 to grant control over PCI/PCI-X Standard Hot-Plug Controller (SHPC) hot plug. If firmware allows the operating system control of this feature, it means that the firmware has made the necessary configurations to ensure that all hot plug events are routed to device interrupts as described in the SHPC 1.0 (e.g., switched from SCI/GPE interrupt mechanism used in the legacy hot plug mechanisms), regardless of whether there are hot pluggable slots down the hierarchy. It is operating system’s responsibility to determine whether a slot is hot pluggable by examining the status of the slots based on the PCI SHPC Specification. Additionally, after control is transferred to the operating system, firmware must not update the state of hot plug slots, including the state of the indicators and power controller. If control of this feature was requested and denied or was not requested, firmware returns this bit set to 0.</p> <p>System firmware must always assume that PCI-X features are supported beneath a PCI Express hierarchy unless it has explicit knowledge to the contrary, and must not mask this bit to zero in PCI Express hierarchy.</p>	

Control Field Bit Offset	Interpretation	Interdependencies
2	<p>PCI Express Native Power Management Events control</p> <p>The firmware sets this bit to 1 to grant control over control over PCI Express native power management event interrupts (PMEs). If firmware allows the operating system control of this feature, then in the context of the <code>_OSC</code> method, it must ensure that all PMEs are routed to root port interrupts as described in the PCI Express Base Specification. Additionally, after control is transferred to the operating system, firmware must not update the PME Status field in the Root Status register or the PME Interrupt Enable field in the Root Control register. If control of this feature was requested and denied or was not requested, firmware returns this bit set to 0.</p>	<p>Yes, refer to 4.5.2.4</p>
3	<p>PCI Express Advanced Error Reporting control</p> <p>The firmware sets this bit to 1 to grant control over PCI Express Advanced Error Reporting. If firmware allows the operating system control of this feature, then in the context of the <code>_OSC</code> method, it must ensure that error messages are routed to device interrupts as described in the PCI Express Base Specification. Additionally, after control is transferred to the operating system, firmware must not modify the Advanced Error Reporting Capability. If control of this feature was requested and denied or was not requested, firmware returns this bit set to 0.</p>	<p>Yes, refer to 4.5.2.4</p>
4	<p>PCI Express Capability Structure control</p> <p>The firmware sets this bit to 1 to grant control over the PCI Express Capability. If the firmware does not grant control of this feature, firmware must handle configuration of the PCI Express Capability Structure.</p> <p>If firmware grants the operating system control of this feature, any firmware configuration of the PCI Express Capability may be overwritten by an operating system configuration, depending on operating system policy.</p>	<p>Yes, refer to 4.5.2.4</p>
5-31	Reserved	

4.5.2. Rules for Evaluating `_OSC`

This section defines when and how the operating system must evaluate `_OSC` as well as restrictions on firmware implementations.

4.5.2.1. Query Flag

If the Query Support Flag (Capabilities DWORD 1, bit 0) is set by the operating system when evaluating `_OSC`, no hardware settings are permitted to be changed by firmware in the context of the `_OSC` call. It is strongly recommended that the operating system evaluate `_OSC` with the Query

Support Flag set until `_OSC` returns the Capabilities Masked bit clear, to negotiate the set of features to be granted to the operating system for native support. A platform may require a specific combination of features to be supported natively by an operating system before granting native control of a given feature.

4.5.2.2. Evaluation Conditions

The operating system must evaluate `_OSC` under the following conditions:

- During initialization of any driver that provides native support for features described in the section above. These features may be supported by one or many drivers, but should only be evaluated by the main bus driver for that hierarchy. Secondary drivers must coordinate with the bus driver to install support for these features. Drivers may not relinquish control of features previously obtained; i.e., bits set in Capabilities DWORD3 after the negotiation process must be set on all subsequent negotiation attempts.
- When a Notify(<device>, 8) is delivered to the PCI Host Bridge device.
- Upon resume from S4. Platform firmware will handle context restoration when resuming from S1-S3.

4.5.2.3. Sequence of `_OSC` Calls

The following rules govern sequences of calls to `_OSC` that are issued to the same host bridge and occur within the same boot.

- The operating system is permitted to evaluate `_OSC` an arbitrary number of times.
- If the operating system declares support of a feature in the Status Field in one call to `_OSC`, then it must preserve the set state of that bit (declaring support for that feature) in all subsequent calls.
- If the operating system is granted control of a feature in the Control Field in one call to `_OSC`, then it must preserve the set state of that bit (requesting that feature) in all subsequent calls.
- Firmware may not reject control of any feature it has previously granted control to.

There is no mechanism for the operating system to relinquish control of a feature previously requested and granted.

4.5.2.4. Dependencies Between `_OSC` Control Bits

Because handling of hot-plug events, power management events, and advanced error reporting all require the modification of PCI Express Capability registers, the operating system is required to claim control over the PCI Express Capability (bit 4 of the Control field) in conjunction with claiming control over PCI Express Native Hot Plug, PCI Express Native Power Management Events, or PCI Express Advanced Error Reporting (bits 0, 2, and 3 of the Control field). If the operating system attempts to claim control of any of these features without also claiming control over the PCI Express Capability, the firmware is required to refuse control of the feature being illegally claimed and mask the corresponding bit.



IMPLEMENTATION NOTE

Some operating systems may require the platform to grant control over all of PCI Express Native Hot Plug, PCI Express Native Power Management Events, PCI Express Advanced Error Reporting and the PCI Express Capability (bits 0, 2, 3, and 4) in an “ALL or NOTHING” manner. If control of any one of these capabilities is denied by the platform, such operating systems may choose not to claim control of any of these features. Platforms that support these operating systems need to take this into account.

4.5.3. ASL Example

A sample _OSC implementation for a mobile system incorporating a PCI Express hierarchy is shown below:

```
Device(PCI0) // Root PCI bus
{
    Name(_HID,EISAID("PNP0A08")) // PCI Express Root Bridge
    Name(_CID,EISAID("PNP0A03")) // Compatible PCI Root Bridge

    Name(SUPP,0) // PCI _OSC Support Field value
    Name(CTRL,0) // PCI _OSC Control Field value

    Method(_OSC,4)
    { // Check for proper UUID
        If(LEqual(Arg0,ToUUID("33DB4D5B-1FF7-401C-9657-7441C03DD766")))
        {
            // Create DWORD-addressable fields from the Capabilities Buffer
            CreateDWordField(Arg3,0,CDW1)
            CreateDWordField(Arg3,4,CDW2)
            CreateDWordField(Arg3,8,CDW3)

            // Save Capabilities DWORD2 & 3
            Store(CDW2,SUPP)
            Store(CDW3,CTRL)

            // Only allow native hot plug control if the OS supports:
            // * ASPM
            // * Clock PM
            // * MSI/MSI-X
            If(LNotEqual(And(SUPP, 0x16), 0x16))
            {
                And(CTRL,0x1E,CTRL) // Mask bit 0 (and undefined bits)
            }

            // Always allow native PME, AER (no dependencies)

            // Never allow SHPC (no SHPC controller in this system)
            And(CTRL,0x1D,CTRL)
        }
    }
}
```

PCI FIRMWARE SPECIFICATION, REV. 3.0

```
If(Not(And(CDW1,1))) // Query flag clear?
{ // Disable GPEs for features granted native control.
  If(And(CTRL,0x01)) // Hot plug control granted?
  {
    Store(0,HPCE) // clear the hot plug SCI enable bit
    Store(1,HPCS) // clear the hot plug SCI status bit
  }
  If(And(CTRL,0x04)) // PME control granted?
  {
    Store(0,PMCE) // clear the PME SCI enable bit
    Store(1,PMCS) // clear the PME SCI status bit
  }
  If(And(CTRL,0x10)) // OS restoring PCI Express cap structure?
  { // Set status to not restore PCI Express cap structure
    // upon resume from S3
    Store(1,S3CR)
  }
}

If(LNotEqual(Arg1,One))
{ // Unknown revision
  Or(CDW1,0x08,CDW1)
}

If(LNotEqual(CDW3,CTRL))
{ // Capabilities bits were masked
  Or(CDW1,0x10,CDW1)
}
// Update DWORD3 in the buffer
Store(CTRL,CDW3)
Return(Arg3)
} Else {
  Or(CDW1,4,CDW1) // Unrecognized UUID
  Return(Arg3)
}
} // End _OSC

// End PCI0
```
