Unified Extensible Firmware Interface  
Engineering Change Request (ECR)

*Draft for Review*

Title: New protocol EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL & EFI\_PCI\_EXPRESS\_OVERRIDE\_PROTOCOL, and EFI encodings for the PCI Express features and its attributes

Document:

*Ashraf Javeed*

*Submission for Review Date: 12/10/2019*

Review Approval Date: *x/xx/200x*

Submission for Technical Editing Date: *x/xx/200x*

Submission for Draft Review Date: *x/xx/200x*

Verification Date: *x/xx/200x*

Verifier: *firstname lastname, company*

History

|  |  |  |
| --- | --- | --- |
| Revision | Date | Description |
| 0.80 | 01/05/2020 | Revised the ECR proposal, added the new interface GetPolicy() function description |
| 0.75 | 12/10/2019 | Revised ECR proposal favoring new protocol for PCI Express features |
| 0.50 | 05/28/2019 | Initial draft, ECR proposal was for the existing legacy PCI Platform Protocol |

Summary

## Summary of Change

The UEFI PI Specification Volume 5 [[1](#PI_spec_volume)] defines the EFI\_PCI\_PLATFORM\_PROTOCOL & EFI\_PCI\_OVERRIDE\_PROTOCOL, and they are not compatible to acquire device-specific platform policies to initialize the new PCI Express features sets.

The proposal is to define new protocol to support the PCI Express features, which can be used to:-

1. retrieve the device-specific platform policies, to configure PCI Express features during PCI enumeration.
2. It can also be used to notify the platform about the device configuration state (w.r.t. the defined set of PCI Express features) after its initialization.
3. To determine the PCI Express features list that are required to be configured. This support the above two interface features of this protocol; a third interface to test the interoperability between the consumer of the protocol (EDK2 PCI Bus driver) and its producer (the platform) to reach agreement on defined set of PCI Express features to be configured. This interface shall be useful for compatibility, especially in the case of binary modules used in the platform firmware build environment.

This ECR also proposes the EFI encoding for the following PCIe-complaint features defined in the PCI Express Base Specification version 4 Revision 1: -

1. Maximum Payload Size (MPS)
2. Maximum Read Request Size (MRRS)
3. Extended Tag
4. Relax Order Enable
5. No Snoop Enable
6. ASPM support
7. Common Clock Configuration
8. Extended SYNC
9. Atomic Op
10. LTR Enable
11. PTM support
12. Clock Power Management (CPM)
13. L1 PM sub states

Like the PCI Platform Protocol has its alias as the PCI Override Protocol that gives option to be produced either by the platform-specific firmware modules, or the Silicon enabling firmware code respectively; the intent is to define PCI Express Platform Protocol which can be published by the platform-specific drivers, or its alias the PCI Express Override Protocol that can be produced by the silicon-specific driver modules.

## Benefits of the Change

This new protocol shall provide interface to the platform to communicate to EDK2 kernel code about its device-specific platform policies with respect to PCI Express features.

This change would help the PCI Bus driver and/or the PCI Host Bridge driver to acquire device-specific policies from the platform, for any of the PCI devices in the chipset, to initialize and configure the defined set of PCI Express features, during PCI enumeration phase.

## Impact of Change

The platform shall be the producer of this new protocol and the consumer shall be the EDK2 PCI Bus driver.

New header files in the MdePkg/Include/Protocol directory path, to contain the definition of this protocol and its interfaces.

This ECR shall define new GUID for the PCI Express Platform Protocol and the PCI Express Override Protocol, along with additional member field to track the protocol version for backward compatibility in future.

This ECR defines new data structure types to collect device-specific platform policies (like the EFI\_PCI\_PLATFORM\_POLICY [[2](#PI_spec_volume_chapter)] for the PCI Platform Protocol) to represent various PCI Express features, like that of MPS and MRRS, which the PCI Bus driver and PCI Host Bridge drivers can use to identify the override requirements (from the platform). Thus, the data exchange that happens between platform and the EDK2 kernel driver will require translation as per the EFI encodings define in this specification.

There shall be three interface methods in this protocol definition: -

1. Interface to retrieve the device-specific platform polices for the PCI Express features (**GetDevicePolicy**)
2. Interface to notify platform about the device configuration state w.r.t. the same PCI Express features which was configured (**NotifyDeviceState**).
3. Interface to get platform policy, to select from the defined set of PCI Express features of the protocol definition, that needs to be configured for the platform (**GetPolicy**).

The platform shall be responsible for publishing the PCI Express Platform Protocol (or PCI Express Override Protocol by its silicon initialization drivers). The absence of this PCI Express protocol would be considered as no support required to initialize the PCI Express features for the platform.

The EDK2 kernel PCI Bus driver (or PCI Host Bridge) who is responsible for configuring the PCI Express features, is expected to first call the platform firmware (producer) the interface **GetPolicy** to notify the platform on list of PCI Express features it can configure at a system level, based on which the platform is expected to select the list of PCI Express features that needs to be configured for its PCI devices.

The caller than invokes the interface **GetDevicePolicy** for each of the PCI device (discovered during the PCI enumeration), to acquire its device specific policies w.r.t. the PCI Express features. The platform is responsible for providing the device-specific policies for the PCI Express features for each PCI device, as per its EFI encoding standards defined in this protocol.

The EDK2 kernel PCI Bus driver (or PCI Host Bridge) is also responsible for notifying the platform through the interface **NotifyDeviceState**, after its PCI Express configuration, on the actual state corresponding to the PCI Express features for every PCI device. This shall help the platform to support any other collateral to the PCI Port and its endpoint device; like for example, the PCIe Retimer connected between the PCI Root port and its endpoint device, is required to be configured with same value for its data packet size, as the MPS value programmed to its root port and its endpoint device. The platform is responsible to react based on the notification received from the EDK2 PCI Bus driver.

## References

* **Industry Specifications**:-
* [UEFI Platform Initialization Specification Version 1.7](https://uefi.org/sites/default/files/resources/PI_Spec_1_7_final_Jan_2019.pdf)

[1] UEFI PI 1.7 volume 5

[2] UEFI PI 1.7 volume 5 chapter 11.6.1

* [UEFI Specification Version 2.8](https://uefi.org/sites/default/files/resources/UEFI_Spec_2_8_final.pdf)
* [PCI Express Base Specification Revision 4.0 Version 1.0](https://members.pcisig.com/wg/PCI-SIG/document/10912?downloadRevision=active)
* **EDK-2 GitHub References**:-
  + Feature Request: [BZ-1954](https://bugzilla.tianocore.org/show_bug.cgi?id=1954)
  + Edk2-staging branch: :- [UEFI\_PCI\_ENHANCE-2](https://github.com/tianocore/edk2-staging/tree/UEFI_PCI_ENHANCE-2)
  + Code changes Patch submissions: [Patch V7](https://github.com/ashrafj/edk2-staging/commit/cfb29efdc29fd6ae90fb2b2e304243d2dffb0c03)

Detailed Description of the Change  
and Special Instructions

## New protocol definition for the PCI Express Platform Protocol and PCI Express Override Protocol

The new members are highlighted in yellow.

Add new GUIDs and name the PCI Express Platform Protocol as EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL and name the PCI Express Override Protocol as EFI\_PCI\_EXPRESS\_OVERRIDE\_PROTOCOL.

**EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL**

**Summary**

This protocol provides the interface between the PCI bus driver/PCI Host Bridge Resource Allocation driver and a platform-specific driver to describe the unique PCI Express features of a platform. This protocol is optional.

**GUID**

**#define EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL\_GUID \**

**{ 0x787b0367, 0xa945, 0x4d60, 0x8d, 0x34, 0xb9, 0xd1, 0x88, \**

**0xd2, 0xd0, 0xb6)**

**Protocol Interface Structure**

**typedef struct \_EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL {**

**UINT8** *MajorVersion*;

**UINT8** *MinorVersion*;

**EFI\_PCI\_EXPRESS\_GET\_DEVICE\_POLICY** *GetDevicePolicy;*

**EFI\_PCI\_EXPRESS\_NOTIFY\_DEVICE\_STATE** *NotifyDeviceState;*

**EFI\_PCI\_EXPRESS\_GET\_POLICY** *GetPolicy;*

**} EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL;**

**Parameters**

*MajorVersion*

The major version of this PCI Platform Protocol.

*MinorVersion*

The minor version of this PCI Platform Protocol

*GetDevicePolicy*

Retrieves the device policy regarding PCI Express features. See the **GetDevicePolicy()** function description

*NotifyDeviceState*

Notify the device configuration state w.r.t. the PCI Express features. See the **NotifyDeviceState()** function description

*GetPolicy*

Retrieve the platform policy to select the PCI Express features. See the **GetPolicy()** function description

**Description**

The **EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL** is published by a platform-aware driver. This protocol is optional; this protocol is used during the PCI enumeration phase. There cannot be more than one instance of this protocol in the system.

The PCI Bus driver shall use the new interface ***GetPolicy()*** to retrieve the platform policy first, to select which of the defined set of the PCI Express features that needs to be initialized. This shall be called just once to determine the platform request for configuring the PCI Express feature list. An error status returned would be considered as non-compatibility with platform and the PCI Express features shall not be configured by the PCI Bus driver.

The PCI Bus driver shall use the new interface ***GetDevicePolicy()*** to retrieve the device-specific platform policy, and it shall do after the main stages of the PCI enumeration, before it initializes the PCI Express features.

The PCI Bus driver shall use the new interface ***NotifyDeviceState()*** as the notification to the platform, after it has completed the initialization of the PCI Express features.

As per the present definition and specification of this protocol, the major version is 1, and minor version is 1. Any driver utilizing this protocol shall use these versions number to maintain the backward compatibility as per its specification changes in future.

**EFI\_PCI\_EXPRESS\_OVERRIDE\_PROTOCOL**

**Summary**

This protocol provides the interface between the PCI bus driver/PCI Host Bridge Resource Allocation driver and an implementation's driver to describe the unique PCI Express features of a platform. This protocol is optional.

**GUID**

**#define EFI\_PCI\_EXPRESS\_OVERRIDE\_GUID \**

**{ 0xb9d5ea1, 0x66cb, 0x4546, { 0xb0, 0xbb, 0x5c, 0x6d, \**

**0xae, 0xd9, 0x42, 0x47 } }**

**Protocol Interface Structure**

**typedef EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL EFI\_PCI\_EXPRESS\_OVERRIDE\_PROTOCOL;**

**Description**

The PCI Override Protocol is published by an implementation aware driver. This protocol is optional. But it must be called, if present, during PCI enumeration. There cannot be more than one instance of this protocol in the system.

If the PCI bus driver detects the presence of this protocol, it will use the PCI Override Protocol to obtain information about the device-specific platform policy. If the PCI Platform Protocol does not exist or returns an error, then this protocol is called.

The member functions can be used for performing any implementation-specific initialization that is appropriate during the phase.

## Interfaces to the EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL / EFI\_PCI\_EXPRESS\_OVERRIDE\_PROTOCOL

The name of the new interface would be GetDevicePolicy, and to address the PCI device, it will take the associated EFI handle of the Root Bridge IO Protocol, along with the PCI address of type **EFI\_PCI\_ROOT\_BRIDGE\_IO\_PROTOCOL\_PCI\_ADDRESS**, as the input parameter and gives out the **EFI\_PCI\_EXPRESS\_DEVICE\_POLICY** as the output parameter to the PCI Bus driver and the PCI Root Bridge Resource Allocation Protocol drivers to utilize during PCI enumeration to configure the given component.

**EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL.GetDevicePolicy()**

Summary

The PCI Bus driver and PCI Host Bridge Resource Allocation Protocol drivers can call this member function to retrieve the platform policies for the PCI device, regarding the PCI enumeration.

Prototype

**typedef**

**EFI\_STATUS**

**(EFIAPI \* EFI\_PCI\_EXPRESS\_GET\_DEVICE\_POLICY ) (**

**IN CONST EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL \*This,**

**IN EFI\_HANDLE RootBridge,**

**IN EFI\_PCI\_ROOT\_BRIDGE\_IO\_PROTOCOL\_PCI\_ADDRESS PciAddress,**

**IN UINTN Size,**

**IN OUT EFI\_PCI\_EXPRESS\_DEVICE\_POLICY \*PciExPolicy**

**);**

Parameters

This

Pointer to the **EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL** instance.

RootBridge

The associated PCI Root Bridge IO Protocol handle. Type **EFI\_HANDLE** is defined in **InstallProtocolInterface()** in the UEFI 2.1 Specification

PciAddress

The address of the PCI device on the PCI bus. This address can be passed to the **EFI\_PCI\_ROOT\_BRIDGE\_IO\_PROTOCOL** member functions to access the PCI configuration space of the device. See UEFI 2.1 Specification for the definition of **EFI\_PCI\_ROOT\_BRIDGE\_IO\_PROTOCOL\_PCI\_ADDRESS**.

*Size*

The size, in bytes, of the input buffer in the next parameter

*PciExPolicy*

The pointer to platform policy with respect to other PCI features like, the MPS, MRRS, etc. Type **EFI\_PCI\_EXPRESS\_DEVICE\_POLICY** is defined in "Related Definitions" below.

Description

The **GetDevicePolicy()** function retrieves the platform policy for a component regarding PCI enumeration. The PCI bus driver and the PCI Host Bridge Resource Allocation Protocol driver can call this member function to retrieve the device-specific policy.

The **GetDevicePolicy()** memberfunction is meant to return data about other PCI Express features which would be supported by the PCI Bus driver in future; like for example the MPS, MRRS, Extended Tag, ASPM, etc. The details about this PCI Express features can be obtained from the PCI Express Base Specification 4.x. The EFI encodings for these features are defined in the **EFI\_PCI\_EXPRESS\_DEVICE\_POLICY**, see the Related Definition section for this.

This member function will use the associated EFI handle of the PCI Root Bridge IO Protocol, along with the type **EFI\_PCI\_ROOT\_BRIDGE\_IO\_PROTOCOL\_PCI\_ADDRESS** to determine the physical PCI device within the chipset, to return its device-specific platform policies. The caller is responsible to allocate buffer and pass its pointer to this member function, to get its device-specific policy data.

The caller is required to initialize the input buffer with either of two values:-

1. **EFI\_PCI\_EXPRESS\_NOT\_APPLICABLE**: for those PCI Express features which are not supported by the calling driver
2. **EFI\_PCI\_EXPRESS\_\*\_AUTO**: for those PCI Express features which are supported by the calling driver

**Related Definitions**

**typedef struct {**

**EFI\_PCI\_EXPRESS\_MAX\_PAYLOAD\_SIZE** *DeviceCtlMPS***;**

**EFI\_PCI\_EXPRESS\_MAX\_READ\_REQ\_SIZE** *DeviceCtlMRRS***;**

**EFI\_PCI\_EXPRESS\_EXTENDED\_TAG** *DeviceCtlExtTag***;**

**EFI\_PCI\_EXPRESS\_RELAX\_ORDER** *DeviceCtlRelaxOrder***;**

**EFI\_PCI\_EXPRESS\_NO\_SNOOP** *DeviceCtlNoSnoop***;**

**EFI\_PCI\_EXPRESS\_ASPM\_SUPPORT** *LinkCtlASPMState***;**

**EFI\_PCI\_EXPRESS\_COMMON\_CLOCK\_CFG** *LinkCtlCommonClkCfg***;**

**EFI\_PCI\_EXPRESS\_EXTENDED\_SYNCH** *LinkCtlExtSynch***;**

**EFI\_PCI\_EXPRESS\_ATOMIC\_OP** *DeviceCtl2AtomicOp***;**

**EFI\_PCI\_EXPRESS\_LTR** *DeviceCtl2LTR***;**

**EFI\_PCI\_EXPRESS\_PTM** *PTMControl***;**

**EFI\_PCI\_EXPRESS\_CTO\_SUPPORT** *CTOsupport;*

**EFI\_PCI\_EXPRESS\_CPM** *LinkCtlCPM***;**

**EFI\_PCI\_EXPRESS\_L1PM\_SUBSTATES** *L1PMSubstates*;

**} EFI\_PCI\_EXPRESS\_DEVICE\_POLICY;**

The **EFI\_PCI\_EXPRESS\_DEVICE\_POLICY** size is fixed as per its definition corresponding to its version, with each byte field represents one PCI Express feature and its bitmask define the legal combinations to represent all the valid combinations of its attributes, defined in the PCI Express Base Specification.

**///**

**/// EFI encoding used in notification to the platform, for any of the PCI Express**

**/// capabilities feature state, to indicate that it is not a supported feature,**

**/// or its present state is unknown**

**///**

**#define EFI\_PCI\_EXPRESS\_NOT\_APPLICABLE 0xFF**

*DeviceCtlMPS*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature Maximum Payload Size (MPS). Refer to PCI Express Base Specification 4, (chapter 7.3.5.3) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities. Below is it data type and the macro definitions which the driver uses for interpreting the platform policy.

**typedef UINT8 EFI\_PCI\_EXPRESS\_MAX\_PAYLOAD\_SIZE;**

**#define EFI\_PCI\_EXPRESS\_MAX\_PAYLOAD\_SIZE\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_MAX\_PAYLOAD\_SIZE\_128B  0x01 //set to default 128B**

**#define EFI\_PCI\_EXPRESS\_MAX\_PAYLOAD\_SIZE\_256B  0x02 //set to 256B if applicable**

**#define EFI\_PCI\_EXPRESS\_MAX\_PAYLOAD\_SIZE\_512B  0x03 //set to 512B if applicable**

**#define EFI\_PCI\_EXPRESS\_MAX\_PAYLOAD\_SIZE\_1024B 0x04 //set to 1024B if applicable**

**#define EFI\_PCI\_EXPRESS\_MAX\_PAYLOAD\_SIZE\_2048B 0x05 //set to 2048B if applicable**

**#define EFI\_PCI\_EXPRESS\_MAX\_PAYLOAD\_SIZE\_4096B 0x06 //set to 4096B if applicable**

*DeviceCtlMRRS*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature Maximum Read Request Size (MRRS). Refer to PCI Express Base Specification 4, (chapter 7.3.5.4) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities. Below is it data type and the macro definitions which the driver uses for interpreting the platform policy.

**typedef UINT8 EFI\_PCI\_EXPRESS\_MAX\_READ\_REQ\_SIZE;**

**#define EFI\_PCI\_EXPRESS\_MAX\_READ\_REQ\_SIZE\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_MAX\_READ\_REQ\_SIZE\_128B 0x01 //set to default 128B**

**#define EFI\_PCI\_EXPRESS\_MAX\_READ\_REQ\_SIZE\_256B  0x02 //set to 256B if applicable**

**#define EFI\_PCI\_EXPRESS\_MAX\_READ\_REQ\_SIZE\_512B  0x03 //set to 512B if applicable**

**#define EFI\_PCI\_EXPRESS\_MAX\_READ\_REQ\_SIZE\_1024B 0x04 //set to 1024B if applicable**

**#define EFI\_PCI\_EXPRESS\_MAX\_READ\_REQ\_SIZE\_2048B 0x05 //set to 2048B if applicable**

**#define EFI\_PCI\_EXPRESS\_MAX\_READ\_REQ\_SIZE\_4096B 0x06 //set to 4096B if applicable**

*DeviceCtlExtTag*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature Extended Tag field enable and 10-bit Tag Requester enable. Refer to PCI Express Base Specification 4, (chapter 7.3.5.4) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities.

**typedef UINT8 EFI\_PCI\_EXPRESS\_EXTENDED\_TAG;**

**#define EFI\_PCI\_EXPRESS\_EXTENDED\_TAG\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_EXTENDED\_TAG\_5BIT 0x01 //set to default 5-bit**

**#define EFI\_PCI\_EXPRESS\_EXTENDED\_TAG\_8BIT 0x02 //set to 8-bit if applicable**

**#define EFI\_PCI\_EXPRESS\_EXTENDED\_TAG\_10BIT 0x03 //set to 10-bit if applicable**

*LinkCtlASPMState*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature PCIe link’s Active State Power Management (ASPM). Refer to PCI Express Base Specification 4, (chapter 7.3.5.7) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities.

**typedef UINT8 EFI\_PCI\_EXPRESS\_ASPM\_SUPPORT;**

**#define EFI\_PCI\_EXPRESS\_ASPM\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_ASPM\_DISABLE 0x01 //set to default disable state**

**#define EFI\_PCI\_EXPRESS\_ASPM\_L0s\_SUPPORT 0x02 //set to L0s state**

**#define EFI\_PCI\_EXPRESS\_ASPM\_L1\_SUPPORT 0x03 //set to L1 state**

**#define EFI\_PCI\_EXPRESS\_ASPM\_L0S\_L1\_SUPPORT 0x04 //set to L0s and L1 state**

*DeviceCtlRelaxOrder*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature PCIe Device’s Relax Ordering enable/disable. Refer to PCI Express Base Specification 4, (chapter 7.3.5.4) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities.

**typedef UINT8 EFI\_PCI\_EXPRESS\_RELAX\_ORDER;**

**#define EFI\_PCI\_EXPRESS\_RO\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_RO\_DISABLE 0x01 //set to default disable state**

**#define EFI\_PCI\_EXPRESS\_RO\_ENABLE 0x02 //set to enable state**

*DeviceCtlNoSnoop*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature PCIe Device’s No-Snoop enable/disable. Refer to PCI Express Base Specification 4, (chapter 7.3.5.4) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities.

**typedef UINT8 EFI\_PCI\_EXPRESS\_NO\_SNOOP;**

**#define EFI\_PCI\_EXPRESS\_NS\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_NS\_DISABLE 0x01 //set to default disable state**

**#define EFI\_PCI\_EXPRESS\_NS\_ENABLE 0x02 //set to enable state**

*LinkCtlCommonClkCfg*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature PCIe Link’s Clock configuration is common or discrete. Refer to PCI Express Base Specification 4, (chapter 7.3.5.7) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities.

**typedef UINT8 EFI\_PCI\_EXPRESS\_COMMON\_CLOCK\_CFG;**

**#define EFI\_PCI\_EXPRESS\_CLK\_CFG\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_CLK\_CFG\_ASYNCH 0x01 //set to default asynchronous clock**

**#define EFI\_PCI\_EXPRESS\_CLK\_CFG\_COMMON 0x02 //set to common clock**

*LinkCtlExtSynch*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature PCIe link’s Extended Synch enable/disable. Refer to PCI Express Base Specification 4, (chapter 7.3.5.7) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities.

**typedef UINT8 EFI\_PCI\_EXPRESS\_EXTENDED\_SYNCH;**

**#define EFI\_PCI\_EXPRESS\_EXT\_SYNCH\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_EXT\_SYNCH\_DISABLE 0x01 //set to default disable state**

**#define EFI\_PCI\_EXPRESS\_EXT\_SYNCH\_ENABLE 0x02 //set to enable state**

*DeviceCtl2AtomicOp*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature PCIe Device’s AtomicOp Requester enable/disable, as well as its Egress blocking based on the port's routing capability. Refer to PCI Express Base Specification 4, (chapter 7.3.5.16) on how to translate the given data structure as per the PCI hardware terminology. If the data member “**Override**” value is returned as 0 than there is no platform policy to override, other data member values must be ignored. If 1 than other data members will be used as per the device capabilities.

**typedef \_EFI\_PCI\_EXPRESS\_ATOMIC\_OP EFI\_PCI\_EXPRESS\_ATOMIC\_OP;**

**struct** **\_EFI\_PCI\_EXPRESS\_ATOMIC\_OP** {

**//**

**// set to indicate the platform request to override based on below data member**

**// bit fields; clear bit indicates no platform request to override and the other**

**// data member bit fields are not applicable.**

**// Ignored when passed as input parameters.**

**//**

**UINT8 Override:1;**

**//**

**// set to enable the device as the requester for AtomicOp; clear bit to force**

**// default disable state**

**//**

**UINT8 Enable\_AtomicOpRequester:1;**

**//**

**// set to enable the AtomicOp Egress blocking on this port based on its routing**

**// capability; clear bit to force default disable state**

**//**

**UINT8 Enable\_AtomicOpEgressBlocking:1;**

**//**

**// the remaining bits are unused for this feature**

**//**

**UINT8 Reserved:5;**

**};**

*DeviceCtl2LTR*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature PCIe Device’s Latency Tolerance Reporting Mechanism enable/disable. Refer to PCI Express Base Specification 4, (chapter 7.3.5.16) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities.

**typedef UINT8 EFI\_PCI\_EXPRESS\_LTR;**

**#define EFI\_PCI\_EXPRESS\_LTR\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_LTR\_DISABLE 0x01 //set to default disable state**

**#define EFI\_PCI\_EXPRESS\_LTR\_ENABLE 0x02 //set to enable state**

*PTMControl*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature PCIe Device’s Precision Time Measurement (PTM) support enable/disable. Refer to PCI Express Base Specification 4, (chapter 7.9.16) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities.

**typedef UINT8 EFI\_PCI\_EXPRESS\_PTM;**

**#define EFI\_PCI\_EXPRESS\_PTM\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_PTM\_DISABLE 0x01 //set to default disable state**

**#define EFI\_PCI\_EXPRESS\_PTM\_ENABLE 0x02 //set to enable state only**

**#define EFI\_PCI\_EXPRESS\_PTM\_ROOT\_SEL 0x03 //set to root select & enable**

*CTOsupport*

This data variable member is reserved to retrieve the PCI device platform policy for the PCI Express feature PCIe Device’s Completion Timeout (CTO) support disable or set to supported ranges. Refer to PCI Express Base Specification 4, (chapter 7.5.3.16) on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be enabled as per its PCI specification based on the device capabilities.

**typedef UINT8 EFI\_PCI\_EXPRESS\_CTO\_SUPPORT;**

**#define EFI\_PCI\_EXPRESS\_CTO\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_CTO\_DEFAULT 0x01 //set to default range of 5us to 50ms**

**#define EFI\_PCI\_EXPRESS\_CTO\_RANGE\_A1 0x02 //set to range of 50us to 100us**

**#define EFI\_PCI\_EXPRESS\_CTO\_RANGE\_A2 0x03 //set to range of 1ms to 10ms**

**#define EFI\_PCI\_EXPRESS\_CTO\_RANGE\_B1 0x04 //set to range of 16ms to 55ms**

**#define EFI\_PCI\_EXPRESS\_CTO\_RANGE\_B2 0x05 //set to range of 65ms to 210ms**

**#define EFI\_PCI\_EXPRESS\_CTO\_RANGE\_C1 0x06 //set to range of 260ms to 900ms**

**#define EFI\_PCI\_EXPRESS\_CTO\_RANGE\_C2 0x07 //set to range of 1s to 3.5s**

**#define EFI\_PCI\_EXPRESS\_CTO\_RANGE\_D1 0x08 //set to range of 4s to 13s**

**#define EFI\_PCI\_EXPRESS\_CTO\_RANGE\_D2 0x09 //set to range of 17s to 64s**

**#define EFI\_PCI\_EXPRESS\_CTO\_DET\_DISABLE 0x10 //set to CTO detection disable**

*LinkCtlCPM*

This data type is to retrieve the PCI device platform policy for the PCI- compliant feature PCIe link's Clock Power Management (CPM) enable or disable. Refer to PCI Express Base Specification 5 (chapter 7.5.3.7), on how to translate the below EFI encodings as per the PCI hardware terminology. If this data member value is returned as 0 than there is no platform policy to override, this feature would be ignored or disabled/enabled, as per its relationship with other components and its capabilities, as defined in its PCI specification. Below is its data type and the macro definitions which the driver uses for interpreting the platform policy.

**typedef UINT8 EFI\_PCI\_EXPRESS\_CPM;**

**#define EFI\_PCI\_EXPRESS\_CPM\_AUTO 0x00 //No request for override**

**#define EFI\_PCI\_EXPRESS\_CPM\_DISABLE 0x01 //set to default disable state**

**#define EFI\_PCI\_EXPRESS\_CPM\_ENABLE 0x02 //set to enable state**

*L1PMSubstates*

This data type is to retrieve the PCI device platform policy for the PCI- compliant feature PCIe link's L1 PM Substates. Refer to PCI Express Base Specification 5 (chapter 7.8.3.3), on how to translate the given data structure as per the PCI hardware terminology. If the data member "Override" value is 0 than there is no platform policy to override, other data members will be ignored; if 1 than other data members are used for this feature, to align the states based on its capabilities as well as its relationship with other components in the PCI hierarchy. The platform is expected to return any combination from the four L1 PM Substates.

**typedef struct \_EFI\_PCI\_EXPRESS\_L1PM\_SUBSTATES EFI\_PCI\_EXPRESS\_L1PM\_SUBSTATES;**

**struct \_EFI\_PCI\_EXPRESS\_L1PM\_SUBSTATES {**

**//**

**// set to indicate the platform request to override the L1 PM Substates using**

**// the other data member bit fields; bit clear means no request from platform**

**// to override the L1 PM Substates for the device. Ignored when passed as input**

**// parameters.**

**//**

**UINT8 Override:1;**

**//**

**// set to enable the PCI-PM L1.2 state; bit clear to force default disable state**

**//**

**UINT8 Enable\_Pci\_Pm\_L1\_2:1;**

**//**

**// set to enable the PCI-PM L1.1 state; bit clear to force default disable state**

**//**

**UINT8 Enable\_Pci\_Pm\_L1\_1:1;**

**//**

**// set to enable the ASPM L1.2 state; bit clear to force default disable state**

**//**

**UINT8 Enable\_Aspm\_L1\_2:1;**

**//**

**// set to enable the ASPM L1.1 state; bit clear to force default disable state**

**//**

**UINT8 Enable\_Aspm\_L1\_1:1;**

**//**

**// rest of the remaining bits are reserved, not utilized; can be reused in**

**// future to define additional conditions as per PCIe capabilities**

**//**

**UINT8 Reserved:3;**

**};**

**Status Codes Returned**

|  |  |
| --- | --- |
| EFI\_SUCCESS | The function completed successfully, may returns platform policy data for the given PCI component |
| EFI\_UNSUPPORTED | PCI component belongs to PCI topology but not part of chipset to provide the platform policy |
| EFI\_INVALID\_PARAMETER | If any of the input parameters are passed with invalid data |

The name of the second interface would be NotifyDeviceState, primarily meant to notify the platform about the PCI Express features configuration state of a PCI device. It will take its associated EFI handle of the PCI IO Protocol, as the input parameter to identify the PCI device, and it also take the **EFI\_PCI\_EXPRESS\_DEVICE\_CONFIGURATION** as the input parameter to provide the device state.

**EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL.NotifyDeviceState()**

Summary

The PCI Bus driver / PCI Host Bridge Resource Allocation Protocol drivers can call this member function to notify the platform about PCI Express features configuration state of the device.

Prototype

**typedef**

**EFI\_STATUS**

**(EFIAPI \* EFI\_PCI\_EXPRESS\_NOTIFY\_DEVICE\_STATE) (**

**IN CONST EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL \*This,**

**IN EFI\_HANDLE PciDevice,**

**IN UINTN Size,**

**IN EFI\_PCI\_EXPRESS\_DEVICE\_CONFIGURATION \*PciExDeviceConfigure**

**);**

Parameters

This

Pointer to the **EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL** instance.

PciDevice

The associated PCI IO Protocol handle. Type **EFI\_HANDLE** is defined in **InstallProtocolInterface()** in the UEFI 2.1 Specification.

*Size*

The size, in bytes, of the input buffer in next parameter.

*PciExDeviceConfigure*

The pointer to device configuration state with respect to other PCI Express features like, the MPS, MRRS, etc. Type **EFI\_PCI\_EXPRESS\_DEVICE\_CONFIGURATION** is defined in "Related Definitions" below.

Description

The **NotifyDeviceState()** function is to notify about the device configured state regarding PCI enumeration. The PCI bus driver and/or the PCI Host Bridge Resource Allocation Protocol driver can call this member function to notify the device configured state.

The **NotifyDeviceState()** memberfunction is meant to provide data about the PCI Express features which would be supported by the PCI Bus driver in future; like for example the MPS, MRRS, Extended Tag, ASPM, etc. The details about this PCI Express features can be obtained from the PCI Express Base Specification 4.x. The EFI encodings and data types used to report out the present configuration state, in the **EFI\_PCI\_EXPRESS\_DEVICE\_CONFIGURATION** are same as those that were used by the platform to return the device-specific platform policies, in the **EFI\_PCI\_EXPRESS\_DEVICE\_POLICY** (see the "Related Definition" section for this). The difference is that it should return the actual state; without any macros corresponding to **EFI\_PCI\_EXPRESS\_\*\_AUTO**, and for the data types of *DeviceCtl2AtomicOp* and *L1PMSubstates*, its corresponding data member "*Override*" bit field value shall be ignored, will not be applicable. Note that, if the notifying driver does not support any of the PCI Express feature than it shall return its corresponding byte value as **EFI\_PCI\_EXPRESS\_NOT\_APPLICABLE**.

This member function will use the associated EFI handle of the PCI IO Protocol, to determine the physical PCI device within the chipset, to notify its PCI Express configured state. The caller is responsible to allocate buffer and pass its pointer to this member function.

Related Definitions

**typedef EFI\_PCI\_EXPRESS\_DEVICE\_POLICY EFI\_PCI\_EXPRESS\_DEVICE\_CONFIGURATION;**

The **EFI\_PCI\_EXPRESS\_DEVICE\_CONFIGURATION** is an alias of the data type **EFI\_PCI\_EXPRESS\_DEVICE\_POLICY**, used in notifying the platform about the PCI Express features configured states of device, through the NotifyDeviceState interface method.

The EFI encoded macros like **EFI\_PCI\_EXPRESS\_\*\_AUTO**, with the value 0 will not be used to report the PCI feature definite state; similarly, for the data type of DeviceCtl2AtomicOp and L1PMSubstates, its data member "Override" will not be used. For any of the device's PCI features that are not supported, or its state is unknown, it will be given as **EFI\_PCI\_EXPRESS\_NOT\_APPLICABLE**.

**Status Codes Returned**

|  |  |
| --- | --- |
| EFI\_SUCCESS | The function completed successfully, the platform was able to identify the PCI device successfully |
| EFI\_INVALID\_PARAMETER | If any of the input parameters are passed with invalid data; resulting in PCI device not found, or its PCI Express device configuration state provided had invalid data |

The third interface GetPolicy determines which of the PCI Express features that needs to be configured, for which it can expect the device-specific policies in the interface GetDevicePolicy for each PCI device. This interface takes in data type **EFI\_PCI\_EXPRESS\_PLATFORM\_POLICY** for its input buffer, and its size; that would be primarily used to exchange the PCI Express features supported and its initialization requirements between the caller and the platform.

**EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL.GetPolicy()**

Summary

The PCI Bus driver / PCI Host Bridge Resource Allocation Protocol driver can call this member function of this PCI Express Platform Protocol, to determine on the defined set of PCI Express features that shall be initialized.

Prototype

**typedef**

**EFI\_STATUS**

**(EFIAPI \* EFI\_PCI\_EXPRESS\_GET\_POLICY) (**

**IN CONST EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL \*This,**

**IN UINTN Size,**

**IN OUT EFI\_PCI\_EXPRESS\_PLATFORM\_POLICY \*PlatformPolicy**

**);**

Parameters

This

Pointer to the **EFI\_PCI\_EXPRESS\_PLATFORM\_PROTOCOL** instance.

*Size*

The size, in bytes, of the input buffer in next parameter.

*PlatformPolicy*

The list of platform policy for the PCI Express features like, the MPS, MRRS, etc. Type **EFI\_PCI\_EXPRESS\_PLATFORM\_POLICY** is defined in "Related Definitions" below.

Description

The **GetPolicy()** function retrieves the platform policy for the defined set of PCI Express features regarding PCI enumeration. The PCI bus driver and the PCI Host Bridge Resource Allocation Protocol driver can call this member function to retrieve the platform policy.

On input, the caller passes the defined set of the PCI Express features to indicate its supported list. On output, it retrieves the list of PCI Express features that need to be initialized. Based on this data, the **GetDevicePolicy()** function’s output device-specific policy data would be used for those corresponding PCI Express features.

See the “Related Definition” below for the description of the output parameter *PlatformPolicy*. For example, say that the caller indicates first 8 PCI Express features in the list of **EFI\_PCI\_EXPRESS\_PLATFORM\_POLICY**, and platform wants only 5 PCI Express features from that list to be configured by the caller. Thus, based on feature list recorded here, the device-specific policy returned from the **GetDevicePolicy()** for every PCI device in the system; it configures only those 5 PCI Express features accordingly.

The protocol producing driver shall use the size input parameter to determine the length of the buffer of type **EFI\_PCI\_EXPRESS\_PLATFORM\_POLICY**, and to also determine version of the caller. If the size of the input buffer is more than what its supporting version (indicated through the *MajorVersion* and *MinorVersion* data members of the protocol) than it shall return EFI\_INVALID\_PARAMETER.

The error code in return status essential means that the caller cannot initialize any PCI Express features. Thus, this interface would be primary interface for the caller to initialize the PCI Express features for the platform, apart from obtaining the handle of this protocol.

Related Definitions

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// EFI\_PCI\_EXPRESS\_PLATFORM\_POLICY**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**typedef struct {**

//

// For PCI Express feature - Max. Payload Size

//

**BOOLEAN Mps;**

//

// For PCI Express feature - Max. Read Request Size

//

**BOOLEAN Mrrs;**

//

// For PCI Express feature - Extended Tag

//

**BOOLEAN ExtTag;**

//

// For PCI Express feature - Relax Order

//

**BOOLEAN RelaxOrder;**

//

// For PCI Express feature - No-Snoop

//

**BOOLEAN NoSnoop;**

//

// For PCI Express feature - ASPM state

//

**BOOLEAN Aspm;**

//

// For PCI Express feature - Common Clock Configuration

//

**BOOLEAN Ccc;**

//

// For PCI Express feature - Extended Sync

//

**BOOLEAN ExtSync;**

//

// For PCI Express feature - Atomic Op

//

**BOOLEAN AtomicOp;**

//

// For PCI Express feature - LTR

//

**BOOLEAN Ltr;**

//

// For PCI Express feature - PTM

//

**BOOLEAN Ptm;**

//

// For PCI Express feature - Completion Timeout

//

**BOOLEAN Cto;**

//

// For PCI Express feature - Clock Power Management

//

**BOOLEAN Cpm;**

//

// For PCI Express feature - L1 PM Substates

//

**BOOLEAN L1PmSubstates;**

**} EFI\_PCI\_EXPRESS\_PLATFORM\_POLICY;**

The **EFI\_PCI\_EXPRESS\_PLATFORM\_POLICY** is used to exchange feature policy, at a system level. Each Boolean member represents each PCI Express feature defined in the PCI Express Base Specification. The value TRUE on exchange indicates support for this feature configuration, and FALSE indicates not supported, or no configuration required on return data. The order of members represents PCI Express features that are fixed for this protocol definition, and should be aligned with the definition of the **EFI\_PCI\_EXPRESS\_DEVICE\_POLICY**.

**Status Codes Returned**

|  |  |
| --- | --- |
| EFI\_SUCCESS | The function completed successfully, returns platform policy for the PCI Express features |
| EFI\_INVALID\_PARAMETER | If any of the input parameters are passed with invalid data; like the pointer to the protocol is not valid, the size of the input buffer is greater than the version supported, the pointer to buffer is NULL. |