EDK II Platform Configuration Database Infrastructure Description

TABLE OF CONTENTS

EDK II Platform Configuration Database Infrastructure Description

EDK II Platform Configuration Database Infrastructure Description

Revision 0.56

12/01/2020 06:07:16

Acknowledgements

Revision History

Revision	Revision History	Date
0.10	Initial Draft	2/6/2006
0.20	Added lots of PCD APIs and the definition for the PCD_IMAGE format.	2/19/2006
0.21	Added VPD support and further explanations of the PCD hierarchy between module/package/platform data collection areas	2/28/2006
0.22	Corrected items in the PCD_DATA encoding to allow for a PCD token to have a GUID entry for DynamicEx access mechanism.	3/02/2006
0.50	Finalized updates and removal of extraneous information. PCD Base Name definitions are covered in a separate specification.	3/23/2006
0.51	Lots of minor edits throughout. Clarifications, text updated, and the .ffs format has gone through several updates.	4/21/2006
0.52	Updated the PCD SePtr routines to handle reflecting correct maximum SizeOfValue.	6/22/2006
	Added GetNextTokenSpace routines for the PCD services	
	Deprecated the PCD FFS definitions - these have been relegated to implementation detail	

EDK II Platform Configuration Database Infrastructure Description

0.53	Removed extraneous non-PCD specific references	6/30/2006
0.54	Change EDK II.	7/13/2006
0.55	Drop the TPL restriction for PCD PPI since TPL is a concept in DXE phase.	4/29/2009
	Add GetNextTokenSpace() service to the PCD PPI and PCD Protocol structures.	
	Update the PCD PPI and Protocol structures to sue the correct field names for the Get/Set services.	
	Update GetNextToken() in the PPI and Protocol to clarify the meaning of a `NULL` Guid parameter that specifies the default token space.	
	Update TPL restriction for the PCD protocol to be <= TPL_CALLBACK.	
0.56	Convert to Gitbook	April 2017

1 INTRODUCTION

1.1 Scope

This document discusses the mechanisms and configuration entries required to make it easy to write portable silicon modules and to port the Framework from platform to platform.

This specification is a "requirements" specification and lists the types of data that need to be abstracted. The set of items requiring abstraction will need to be converted in to various implementations-global build flags, driver specific build flags, protocols, PPIs, data structures, and build files-that form a true porting guide.

This document focuses on data types needed for writing driver modules and includes detailed listings of properties. Porting to a specific platform does not require manually setting all of these values. Some values are derivable from other values and don't need to be set directly. Also, it might be possible to build a more intelligent way to gather the requirements for the user without having them fill out an exhaustive list of properties.

1.2 Target Audience

This document is intended to be the basis for the EDK II (EDK 2.0) activity and is designed to get feedback about what types of data need to be abstracted.

The goal of the document will be to enable the creation of implementation documentation that talks about build and code changes, in addition to the creation of detailed porting guides.

1.3 Related Information

The following publications and sources of information may be useful to you or are referred to by this specification:

- Extensible Firmware Interface Specification Version 1.10, Intel, 2001, http://developer.intel.com/technology/efi.
- Unified Extensible Firmware Interface Specification Version 2.0, Unified EFI, Inc, 2006, http://www.uefi.org.
- Intel(R) Platform Innovation Framework for EFI Specifications, Intel, 2006, http://www.intel.com/technology/framework.
- EDK II Module Development Environment Package Specification, Version 0.51, Intel, 2006, http://www.tianocore.org.
- EDK II Build and Packaging Architecture Specification, Version 0.53, Intel, 2006, http://www.tianocore.org.
- EDK II Module Surface Area Specification, Version 0.51, Intel, 2006, http://www.tianocore.org.
- EDK II Module Development Environment Library Specification, Version 0.58, Intel, 2006, http://www.tianocore.org.
- EDK II Platform Configuration Database Infrastructure Description, Version 0.54, Intel, 2006, http://www.tianocore.org.
- EDK II C Coding Standards Specification, Version 0.51, Intel, 2006, http://www.tianocore.org.

1.4 Terms

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

BDS

Framework Boot Device Selection phase.

BNF

BNF is an acronym for "Backus Naur Form." John Backus and Peter Naur introduced for the first time a formal notation to describe the syntax of a given language.

Component

An executable image. Components defined in this specification support one of the defined module types.

DXE

Framework Driver Execution Environment phase.

DXE SAL

Special class of DXE module that produces SAL Runtime Services. DXE SAL modules differ from DXE Runtime modules in that the DXE Runtime modules support Virtual mode OS calls at OS runtime and DXE SAL modules support intermixing Virtual or Physical mode OS calls.

DXE SMM

Special class of DXE module that is loaded into the System Management Mode memory.

DXE Runtime

Special class of DXE module that provides Runtime Services

EFI

Generic term that refers to one of the versions of the EFI specification: EFI .1.02, EFI 1.10, or UEFI 2.0.

EFI 1.10 Specification

The Extensible Firmware Interface Specification was published by Intel Corporation. Intel has donated the EFI specification to the Unified EFI Forum, and the UEFI now owns future updates of the EFI specification. See UEFI Specification Version 2.0.

Foundation

The set of code and interfaces that glue implementations of the Framework together.

Framework

Intel(R) Platform Innovation Framework for EFI consists of the Foundation, plus other modular components that characterize the portability surface for modular components designed to work on any implementation of the Framework architecture.

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.

HII

Human Interface Infrastructure. This generally refers to the database that contains string, font, and IFR information along with other pieces that use one of the database components.

IFR

Internal Forms Representation. This is the binary encoding that is used for the representation of user interface pages.

Library Class

A library class defines the API or interface set for a library. The consumer of the library is coded to the library class definition. Library classes are defined via a library class .h file that is published by a package. See the EDK II Module Development Environment Library Specification for a list of libraries defined in this package.

Library Instance

An implementation of one or more library classes. See the EDK II Module Development Environment Library Specification for a list of libraries defined in this package.

Module

A module is either an executable image or a library instance. For a list of module types supported by this package, see Module Type.

Module Type

All libraries and components belong to one of the following module types: `BASE`, `SEC`, `PEI_CORE`, `PEIM`, `DXE_CORE`, `DXE_DRIVER`, `DXE_RUNTIME_DRIVER`, `DXE_SMM_DRIVER`, `DXE_SAL_DRIVER`, `UEFI_DRIVER`, or `UEFI_APPLICATION`. These definitions provide a framework that is consistent with a similar set of requirements. A module that is of module type `BASE`, depends only on headers and libraries provided in the MDE, while a module that is of module type `DXE_DRIVER` depends on common DXE components. For a definition of the various module types, see Module Type.

Module Surface Area (MSA)

The MSA is an XML description of how the module is coded. The MSA contains information about the different construction options for the module. After the module is constructed the MSA can describe the interoperability requirements of a module.

Package

A package is a container. It can hold a collection of files for any given set of modules. Packages may be described as one of the following types of modules:

- Source modules, containing all source files and descriptions of a module
- Binary modules, containing EFI Sections or a Framework File System and a description file specific to linking and binary editing of features and attributes specified in a Platform Configuration Database (PCD)
- Mixed modules, with both binary and source modules

Multiple modules can be combined into a package, and multiple packages can be combined into a single package.

Protocol

An API named by a GUID as defined by the EFI specification.

PCD

Platform Configuration Database.

PEI

Pre-EFI Initialization Phase.

PPI

A PEIM-to-PEIM Interface that is named by a GUID as defined by the PEI CIS.

SAL

System Abstraction Layer. A firmware interface specification used on Intel(R) Itanium(R) Processor based systems.

Runtime Services

Interfaces that provide access to underlying platform-specific hardware that might be useful during OS runtime, such as time and date services. These services become active during the boot process but also persist after the OS loader terminates boot services.

SEC

Security Phase is the code in the Framework that contains the processor reset vector and launches PEI. This phase is separate from PEI because some security schemes require ownership of the reset vector.

UEFI Application

An application that follows the UEFI specification. The only difference between a UEFI application and a UEFI driver is that an application is unloaded from memory when it exits regardless of return status, while a driver that returns a successful return status is not unloaded when its entry point exits.

UEFI Driver

A driver that follows the UEFI specification.

UEFI Specification Version 2.0

Current version of the EFI specification released by the Unified EFI Forum. This specification builds on the EFI 1.10 specification and transfers ownership of the EFI specification from Intel to a non-profit, industry trade organization.

Unified EFI Forum

A non-profit collaborative trade organization formed to promote and manage the UEFI standard. For more information, see http://www.uefi.org.

1.5 Conventions Used in this Document

This document uses typographic and illustrative conventions described below.

1.5.1 Data Structure Descriptions

The Intel(R) Architecture processors of the IA-32 family are "little endian" machines. This means that the low-order byte of a multi-byte data item in memory is at the lowest address, while the high-order byte is at the highest address. Processors of the Itanium(R) Processor Family (IPF) 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.

1.5.2 Protocol Descriptions

The protocols described in this document generally have the following format:

Protocol Name

The formal name of the protocol interface.

Summary

A brief description of the protocol interface.

GUID

The 128-bit Globally Unique Identifier (GUID) for the protocol interface.

Protocol Interface Structure

A "C-style" data structure definition containing the procedures and data fields produced by this protocol interface.

Parameters

A brief description of each field in the protocol interface structure.

Description

A description of the functionality provided by the interface, including any limitations and caveats of which the caller should be aware.

Related Definitions

The type declarations and constants that are used in the protocol interface structure or any of its procedures.

1.5.3 Procedure Descriptions

The procedures described in this document generally have the following format:

ProcedureName()

The formal name of the procedure.

Summary

A brief description of the procedure.

Prototype

A "C-style" procedure header defining the calling sequence.

Parameters

A brief description of each field in the procedure prototype.

Description

A description of the functionality provided by the interface, 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 procedure.

Status Codes Returned

A description of any codes returned by the interface. The procedure is required to implement any status codes listed in this table. Additional error codes may be returned, but they will not be tested by standard compliance tests, and any software that uses the procedure cannot depend on any of the extended error codes that an implementation may provide.

1.5.4 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 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 EFI Specification.

1.5.5 Typographic Conventions

The following typographic conventions are used in this document to illustrate programming concepts:

`Code` : This typeface is use to indicate code.

Argument : This typeface is used to indicate arguments.

register: This typeface is used to indicate a processor register.

2 PCD PROTOCOL

Summary

A platform database that contains a variety of current platform settings or directives that can be accessed by a driver or application.

GUID

```
#define PCD_PROTOCOL_GUID \
  { 0x11b34006, 0xd85b, 0x4d0a, { 0xa2, 0x90, 0xd5, 0xa5, 0x71, 0x31, 0xe, 0xf7 }}
```

Protocol Interface Structure

```
typedef struct {
  PCD_PROTOCOL_SET_SKU              SetSku;

  PCD_PROTOCOL_GET8                 Get8;
  PCD_PROTOCOL_GET16                Get16;
  PCD_PROTOCOL_GET32                Get32;
  PCD_PROTOCOL_GET64                Get64;
  PCD_PROTOCOL_GET_POINTER          GetPtr;
  PCD_PROTOCOL_GET_BOOLEAN          GetBool;
  PCD_PROTOCOL_GET_SIZE             GetSize;

  PCD_PROTOCOL_GET_EX_8             Get8Ex;
  PCD_PROTOCOL_GET_EX_16            Get16Ex;
  PCD_PROTOCOL_GET_EX_32            Get32Ex;
  PCD_PROTOCOL_GET_EX_64            Get64Ex;
  PCD_PROTOCOL_GET_EX_POINTER       GetPtrEx;
  PCD_PROTOCOL_GET_EX_BOOLEAN       GetBoolEx;
  PCD_PROTOCOL_GET_EX_SIZE          GetSizeEx;

  PCD_PROTOCOL_SET8                 Set8;
  PCD_PROTOCOL_SET16                Set16;
  PCD_PROTOCOL_SET32                Set32;
  PCD_PROTOCOL_SET64                Set64;
  PCD_PROTOCOL_SET_POINTER          SetPtr;
  PCD_PROTOCOL_SET_BOOLEAN          SetBool;

  PCD_PROTOCOL_SET_EX_8             Set8Ex;
  PCD_PROTOCOL_SET_EX_16            Set16Ex;
  PCD_PROTOCOL_SET_EX_32            Set32Ex;
  PCD_PROTOCOL_SET_EX_64            Set64Ex;
  PCD_PROTOCOL_SET_EX_POINTER       SetPtrEx;
  PCD_PROTOCOL_SET_EX_BOOLEAN       SetBoolEx;

  PCD_PROTOCOL_CALLBACK_ON_SET      CallBackOnSet;
  PCD_PROTOCOL_CANCEL_CALLBACK      CancelCallBackOnSet;
  PCD_PROTOCOL_GET_NEXT_TOKEN       GetNextToken;
  PCD_PROTOCOL_GET_NEXT_TOKENSPACE  GetNextTokenSpace;
} PCD_PROTOCOL;
```

Parameters

SetSku

Establish a current SKU value for the PCD service to use for subsequent data Get/Set requests.

Get8

Retrieve an 8-bit value from the PCD service using the standard platform namespace.

Get16

Retrieve a 16-bit value from the PCD service using the standard platform namespace.

Get32

Retrieve a 32-bit value from the PCD service using the standard platform namespace.

Get64

Retrieve a 64-bit value from the PCD service using the standard platform namespace.

GetPtr

Retrieve a pointer to a value from the PCD service using the standard platform namespace. Can be used to retrieve an array of bytes that represent a data structure, ASCII string, or Unicode string

GetBool

Retrieve a Boolean value from the PCD service using the standard platform namespace.

GetSize

Retrieve the size of a particular PCD Token value using the standard platform namespace.

Get8Ex

Retrieve an 8-bit value from the PCD service using a GUIDed token namespace.

Get16Ex

Retrieve a 16-bit value from the PCD service using a GUIDed token namespace.

Get32Ex

Retrieve a 32-bit value from the PCD service using a GUIDed token namespace.

Get64Ex

Retrieve a 64-bit value from the PCD service using a GUIDed token namespace.

GetPtrEx

Retrieve a pointer to a value from the PCD service using a GUIDed token namespace. Can be used to retrieve an array of bytes that may represent a data structure, ASCII string, or Unicode string

GetBoolEx

Retrieve a Boolean value from the PCD service using a GUIDed token namespace.

GetSizeEx

Retrieve the size of a particular PCD Token value using a GUIDed token namespace.

Set8

Set an 8-bit value in the PCD service using the standard platform namespace.

Set16

Set a 16-bit value in the PCD service using the standard platform namespace.

Set32

Set a 32-bit value in the PCD service using the standard platform namespace.

Set64

Set a 64-bit value in the PCD service using the standard platform namespace.

SetPtr

Set a pointer to a value in the PCD service using the standard platform namespace. Can be used to set an array of bytes that may represent a data structure, ASCII string, or Unicode string

SetBool

Set a Boolean value in the PCD service using the standard platform namespace.

Set8Ex

Set an 8-bit value in the PCD service using a GUIDed token namespace.

Set16Ex

Set a 16-bit value in the PCD service using a GUIDed token namespace.

Set32Ex

Set a 32-bit value in the PCD service using a GUIDed token namespace.

Set64Ex

Set a 64-bit value in the PCD service using a GUIDed token namespace.

SetPtrEx

Set a pointer to a value in the PCD service using a GUIDed token namespace. Can be used to set an array of bytes that may represent a data structure, ASCII string, or Unicode string

SetBoolEx

Set a Boolean value in the PCD service using a GUIDed token namespace.

CallBackOnSet

Establish a notification to alert when a particular PCD Token value is set.

CancelCallBackOnSet

Cancel a previously set notification for a particular PCD Token value.

GetNextToken

Retrieve the next token number that is contained in the PCD name-space.

GetNextTokenSpace

Retrieves the next valid PCD token namespace for a given namespace.

Description

Callers to this protocol must be <= TPL_CALLBACK task priority level. This is the base PCD service API that provides an abstraction for accessing configuration content in the platform. It is a seamless mechanism for extracting information regardless of where the information is stored (such as in VPD, an EFI Variable, or a PCD FFS file,).

This protocol allows access to data through size-granular APIs and provides a mechanism for a firmware component to monitor specific settings and be alerted when a setting is changed.

PCD_PROTOCOL.SetSku()

Summary

Sets the SKU value for subsequent calls to set or get PCD token values.

Prototype

```
typedef
VOID
(EFIAPI *PCD_PROTOCOL_SET_SKU) (
  IN UINTN  SkuId
  );
```

Parameters

SkuId

The SKU value to set.

Description

`SetSku()` sets the SKU Id to be used for subsequent calls to set or get PCD values. `SetSku()` is normally called only once by the system.

For each item (token), the database can hold a single value that applies to all SKUs, or multiple values, where each value is associated with a specific SKU Id. Items with multiple, SKU-specific values are called SKU enabled.

The SKU Id of zero is reserved as a default. The valid ***SkuId*** range is 1 to 255 For tokens that are not SKU enabled, the system ignores any set SKU Id and works with the single value for that token. For SKU-enabled tokens, the system will use the SKU Id set by the last call to `SetSku()`. If no SKU Id is set or the currently set SKU Id isn't valid for the specified token, the system uses the default SKU Id. If the system attempts to use the default SKU Id and no value has been set for that Id, the results are unpredictable.

PCD_PROTOCOL.Get8()

Summary

Retrieves an 8-bit value for a given PCD token.

Prototype

```
typedef
UINT8
(EFIAPI *PCD_PROTOCOL_GET8)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current byte-sized value for a PCD token number. If ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.Get16()

Summary

Retrieves a 16-bit value for a given PCD token.

Prototype

```
typedef
UINT16
(EFIAPI *PCD_PROTOCOL_GET16)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current word-sized value for a PCD token number. If ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.Get32()

Summary

Retrieves a 32-bit value for a given PCD token.

Prototype

```
typedef
UINT32
(EFIAPI *PCD_PROTOCOL_GET32)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current 32-bit sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.Get64()

Summary

Retrieves a 64-bit value for a given PCD token.

Prototype

```
typedef
UINT64
(EFIAPI *PCD_PROTOCOL_GET64)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current byte-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.GetPtr()

Summary

Retrieves a pointer to a value for a given PCD token.

Prototype

```
typedef
VOID *
(EFIAPI *PCD_PROTOCOL_GET_POINTER)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current pointer to the buffer for a PCD token number. Do not make any assumptions about the alignment of the pointer that is returned by this function call. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.GetBool()

Summary

Retrieves a Boolean value for a given PCD token.

Prototype

```
typedef
BOOLEAN
(EFIAPI *PCD_PROTOCOL_GET_BOOLEAN) (
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current Boolean-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.GetSize()

Summary

Retrieves the size of the value for a given PCD token.

Prototype

```
typedef
UINTN
(EFIAPI *PCD_PROTOCOL_GET_SIZE)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current size of a particular PCD token. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.Get8Ex()

Summary

Retrieves an 8-bit value for a given PCD token.

Prototype

```
typedef
UINT8
(EFIAPI *PCD_PROTOCOL_GET_EX_8) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current byte-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.Get16Ex()

Summary

Retrieves a 16-bit value for a given PCD token.

Prototype

```
typedef
UINT16
(EFIAPI *PCD_PROTOCOL_GET_EX_16) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current word-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.Get32Ex()

Summary

Retrieves a 32-bit value for a given PCD token.

Prototype

```
typedef
UINT32
(EFIAPI *PCD_PROTOCOL_GET_EX_32) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current 32-bit sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.Get64Ex()

Summary

Retrieves a 64 -bit value for a given PCD token.

Prototype

```
typedef
UINT64
(EFIAPI *PCD_PROTOCOL_GET_EX_64) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the 64-bit sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.GetPtrEx()

Summary

Retrieves a pointer to a value for a given PCD token.

Prototype

```
typedef
VOID *
(EFIAPI *PCD_PROTOCOL_GET_EX_POINTER) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current pointer to the value for a PCD token number. Do not make any assumptions about the alignment of the pointer that is returned by this function call. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.GetBoolEx()

Summary

Retrieves a Boolean value for a given PCD token.

Prototype

```
typedef
BOOLEAN
(EFIAPI *PCD_PROTOCOL_GET_EX_BOOLEAN) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current BOOLEAN-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PROTOCOL.Set8()

Summary

Sets an 8-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET8) (
  IN UINTN  TokenNumber,
  IN UINT8  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service set the value requested.
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.Set16()

Summary

Sets a 16-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET16) (
  IN UINTN   TokenNumber,
  IN UINT16  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.Set32()

Summary

Sets a 32-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET32) (
  IN UINTN   TokenNumber,
  IN UINT32  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.Set64()

Summary

Sets a 64-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET64) (
  IN UINTN   TokenNumber,
  IN UINT64  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.SetPtr()

Summary

Sets a value of a specified size for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET_POINTER) (
  IN UINTN      TokenNumber,
  IN OUT UINTN  *SizeOfValue,
  IN VOID       *Buffer
  );
```

Parameters

TokenNumber

The PCD token number.

SizeOfValue

A pointer to the length of the value being set for the PCD token. On input, if the ***SizeOfValue*** is greater than the maximum size supported for this ***TokenNumber*** then the output value of ***SizeOfValue*** will reflect the maximum size supported for this ***TokenNumber*** .

Buffer

A pointer to the buffer containing the value to set.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
EFI_SUCCESS	The PCD service has set the value requested
EFI_INVALID_PARAMETER	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
EFI_NOT_FOUND	The PCD service could not find the requested token number.

PCD_PROTOCOL.SetBoolean()

Summary

Sets a Boolean value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET_BOOLEAN) (
  IN UINTN    TokenNumber,
  IN BOOLEAN  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.Set8Ex()

Summary

Sets an 8-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET_EX_8) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINT8           Value
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.Set16Ex()

Summary

Sets a 16-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET_EX_16) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINT16          Value
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.Set32Ex()

Summary

Sets a 32-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET_EX_32) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINT32          Value
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.Set64Ex()

Summary

Sets a 64-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET_EX_64) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINT64          Value
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.SetPtrEx()

Summary

Sets a value of a specified size for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET_POINTER_EX) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINTN           SizeOfValue,
  IN VOID            *Buffer
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

SizeOfValue

The length of the value being set for the PCD token.

Buffer

A pointer to the buffer containing the value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.SetBoolEx()

Summary

Sets a Boolean value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_SET_EX_BOOLEAN) (
  IN CONST EFI_GUI  *Guid,
  IN UINTN          TokenNumber,
  IN BOOLEAN        Value
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.CallbackOnSet()

Summary

Specifies a function to be called anytime the value of a designated token is changed.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_CALLBACK_ON_SET) (
  IN CONST EFI_GUID         *Guid, OPTIONAL
  IN UINTN                  CallBackToken,
  IN PCD_PROTOCOL_CALLBACK  CallBackFunction
  );
```

Parameters

Guid

The 128-bit unique value that designates which namespace to monitor. If `NULL`, use the standard platform namespace.

CallBackToken

The PCD token number to monitor.

CallBackFunction

The function prototype called when the value associated with the ***CallBackToken*** is set.

Related Definitions

```
typedef
VOID
(EFIAPI *PCD_PROTOCOL_CALLBACK) (
  IN EFI_GUID  *Guid, OPTIONAL
  IN UINTN     CallBackToken,
  IN VOID      *TokenData,
  IN UINTN     TokenDataSize
  );
```

Description

Specifies a function to be called anytime the value of a designated token is changed.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has successfully established a call event for the ***CallBackToken*** requested.
`EFI_NOT_FOUND`	The PCD service could not find the referenced token number.

PCD_PROTOCOL.CancelCallback()

Summary

Cancels a previously set callback function for a particular PCD token number.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_CANCEL_CALLBACK) (
  IN CONST EFI_GUID         *Guid, OPTIONAL
  IN UINTN                  CallBackToken,
  IN PCD_PROTOCOL_CALLBACK  CallBackFunction
  );
```

Parameters

Guid

The 128-bit unique value that designates which namespace to monitor. If `NULL`, use the standard platform namespace.

CallBackToken

The PCD token number for which to cancel monitoring.

CallBackFunction

The function prototype that was originally passed to the `CallBackOnSet` function.

Description

Cancels a callback function that was set through a previous call to the `CallBackOnSet` function.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service cancelled the call event associated with the ***CallBackToken***.
`EFI_INVALID_PARAMETER`	The PCD service did not match the ***CallBackFunction*** to one that is currently being monitored.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PROTOCOL.GetNextToken()

Summary

Retrieves the next valid PCD token for a given namespace.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_GET_NEXT_TOKEN) (
  IN CONST EFI_GUID  *Guid, OPTIONAL
  IN OUT UINTN       *TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to retrieve the next token. This is an optional parameter that may be `NULL`. If this parameter is `NULL`, then a request is being made to retrieve tokens from the default token space.

TokenNumber

A pointer to the PCD token number to use to find the subsequent token number.

Description

Retrieves the next valid token number in a given namespace. This is useful since the PCD infrastructure contains a sparse list of token numbers, and one cannot a priori know what token numbers are valid in the database.

If TokenNumber is 0 and Guid is not `NULL`, then the first token from the token space specified by Guid is returned. If TokenNumber is not 0 and Guid is not `NULL`, then the next token in the token space specified by Guid is returned. If TokenNumber is 0 and Guid is `NULL`, then the first token in the default token space is returned. If TokenNumber is not 0 and Guid is `NULL`, then the next token in the default token space is returned. The token numbers in the default token space may not be related to token numbers in token spaces that are named by Guid.

If the next token number can be retrieved, then it is returned in TokenNumber, and `EFI_SUCCESS` is returned. If TokenNumber represents the last token number in the token space specified by Guid, then `EFI_NOT_FOUND` is returned. If TokenNumber is not present in the token space specified by Guid, then `EFI_NOT_FOUND` is returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service retrieved the value requested.
`EFI_NOT_FOUND`	The PCD service could not find data from the requested token number.

PCD_PROTOCOL.GetNextTokenSpace()

Summary

Retrieves the next valid PCD token namespace for a given namespace.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_GET_NEXT_TOKENSPACE) (
  IN OUT CONST EFI_GUID  **Guid
  );
```

Parameters

Guid

An indirect pointer to `EFI_GUID`. On input, it designates a known token namespace from which the search will start. On output, it designates the next valid token namespace on the platform. If `*Guid` is `NULL`, then the GUID of the first token space of the current platform is returned. If the search cannot locate the next valid token namespace, an error is returned and the value of `*Guid` is undefined.

Description

Gets the next valid token namespace for a given namespace. This is useful for traversing the valid token namespaces on a platform.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service retrieved the value requested.
`EFI_NOT_FOUND`	The PCD service could not find the next valid token namespace.

3 PCD PPI

Summary

A platform database that contains a variety of current platform settings or directives that can be accessed by a driver or application.

GUID

```
#define PCD_PPI_GUID \
  { 0x6e81c58, 0x4ad7, 0x44bc,{ 0x83, 0x90, 0xf1, 0x2, 0x65, 0xf7, 0x24, 0x80 }}
```

Protocol Interface Structure

```
typedef struct {
  PCD_PPI_SET_SKU              SetSku;

  PCD_PPI_GET8                 Get8;
  PCD_PPI_GET16                Get16;
  PCD_PPI_GET32                Get32;
  PCD_PPI_GET64                Get64;
  PCD_PPI_GET_POINTER          GetPtr;
  PCD_PPI_GET_BOOLEAN          GetBool;
  PCD_PPI_GET_SIZE             GetSize;

  PCD_PPI_GET_EX_8             Get8Ex;
  PCD_PPI_GET_EX_16            Get16Ex;
  PCD_PPI_GET_EX_32            Get32Ex;
  PCD_PPI_GET_EX_64            Get64Ex;
  PCD_PPI_GET_EX_POINTER       GetPtrEx;
  PCD_PPI_GET_EX_BOOLEAN       GetBoolEx;
  PCD_PPI_GET_EX_SIZE          GetSizeEx;

  PCD_PPI_SET8                 Set8;
  PCD_PPI_SET16                Set16;
  PCD_PPI_SET32                Set32;
  PCD_PPI_SET64                Set64;
  PCD_PPI_SET_POINTER          SetPtr;
  PCD_PPI_SET_BOOLEAN          SetBool;

  PCD_PPI_SET_EX_8             Set8Ex;
  PCD_PPI_SET_EX_16            Set16Ex;
  PCD_PPI_SET_EX_32            Set32Ex;
  PCD_PPI_SET_EX_64            Set64Ex;
  PCD_PPI_SET_EX_POINTER       SetPtrEx;
  PCD_PPI_SET_EX_BOOLEAN       SetBoolEx;

  PCD_PPI_CALLBACK_ON_SET      CallbackOnSet;
  PCD_PPI_CANCEL_CALLBACK      CancelCallback;
  PCD_PPI_GET_NEXT_TOKEN       GetNextToken;
  PCD_PPI_GET_NEXT_TOKENSPACE  GetNextTokenSpace;
} PCD_PPI;
```

Parameters

SetSku

Establish a current SKU value for the PCD service to use for subsequent data Get/Set requests.

Get8

Retrieve an 8-bit value from the PCD service using the standard platform namespace.

Get16

Retrieve a 16-bit value from the PCD service using the standard platform namespace.

Get32

Retrieve a 32-bit value from the PCD service using the standard platform namespace.

Get64

Retrieve a 64-bit value from the PCD service using the standard platform namespace.

GetPtr

Retrieve a pointer to a value from the PCD service using the standard platform namespace. Can be used to retrieve an array of bytes that may represent a data structure, ASCII string, or Unicode string

GetBool

Retrieve a Boolean value from the PCD service using the standard platform namespace.

GetSize

Retrieve the size of a particular PCD Token value using the standard platform namespace.

Get8Ex

Retrieve an 8-bit value from the PCD service using a GUIDed token namespace.

Get16Ex

Retrieve a 16-bit value from the PCD service using a GUIDed token namespace.

Get32Ex

Retrieve a 32-bit value from the PCD service using a GUIDed token namespace.

Get64Ex

Retrieve a 64-bit value from the PCD service using a GUIDed token namespace.

GetPtrEx

Retrieve a pointer to a value from the PCD service using a GUIDed token namespace. Can be used to retrieve an array of bytes that represents a data structure, ASCII string, or Unicode string

GetBoolEx

Retrieve a Boolean value from the PCD service using a GUIDed token namespace.

GetSizeEx

Retrieve the size of a particular PCD Token value using a GUIDed token namespace.

Set8

Set an 8-bit value in the PCD service using the standard platform namespace.

Set16

Set an 8-bit value in the PCD service using the standard platform namespace.

Set32

Set a 32-bit value in the PCD service using the standard platform namespace.

Set64

Set a 64-bit value in the PCD service using the standard platform namespace.

SetPtr

Set a pointer to a value in the PCD service using the standard platform namespace. Can be used to set an array of bytes that represents a data structure, ASCII string, or Unicode string

SetBool

Set a Boolean value in the PCD service using the standard platform namespace.

Set8Ex

Set an 8-bit value in the PCD service using a GUIDed token namespace.

Set16Ex

Set a 16-bit value in the PCD service using a GUIDed token namespace.

Set32Ex

Set a 32-bit value in the PCD service using a GUIDed token namespace.

Set64Ex

Set a 64-bit value in the PCD service using a GUIDed token namespace.

SetPtrEx

Set a pointer to a value in the PCD service using a GUIDed token namespace. Can be used to set an array of bytes that represents a data structure, ASCII string, or Unicode string

SetBoolEx

Set a Boolean value in the PCD service using a GUIDed token namespace.

CallBackOnSet

Establish a notification when a particular PCD Token value is set.

CancelCallBackOnSet

Cancel a previously set notification for a particular PCD Token value.

GetNextToken

Retrieve the next token number that is contained in the PCD name -space.

GetNextTokenSpace

Retrieves the next valid PCD token namespace for a given namespace.

Description

This is the base PCD service API that provides an abstraction for accessing configuration content in the platform. It is a seamless mechanism for extracting information regardless of where the information is stored (such as in VPD, in an EFI Variable, or PCD FFS file).

This PPI provides access to data through size-granular APIs and provides a mechanism for a firmware component to monitor specific settings and be alerted when a setting is changed.

PCD_PPI.SetSku()

Summary

Sets the SKU value for subsequent calls to set or get PCD token values.

Prototype

```
typedef
VOID
(EFIAPI *PCD_PPI_SET_SKU) (
  IN UINTN  SkuId
);
```

Parameters

SkuId

The SKU value to set.

Description

SetSku() sets the SKU Id to be used for subsequent calls to set or get PCD values. SetSku() is normally called only once by the system.

For each item (token), the database can hold a single value that applies to all SKUs, or multiple values, where each value is associated with a specific SKU Id. Items with multiple, SKU-specific values are called SKU enabled.

The SKU Id of zero is reserved as a default. The valid ***SkuId*** range is 1 to 255 for tokens that are not SKU enabled, the system ignores any set SKU Id and works with the single value for that token. For SKU-enabled tokens, the system will use the SKU Id set by the last call to SetSku(). If no SKU Id is set or the currently set SKU Id isn't valid for the specified token, the system uses the default SKU Id. If the system attempts to use the default SKU Id and no value has been set for that Id, the results are unpredictable.

PCD_PPI.Get8()

Summary

Retrieves an 8-bit value for a given PCD token.

Prototype

```
typedef
UINT8
(EFIAPI *PCD_PPI_GET8)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current byte-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.Get16()

Summary

Retrieves a 16-bit value for a given PCD token.

Prototype

```
typedef
UINT16
(EFIAPI *PCD_PPI_GET16)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current word-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.Get32()

Summary

Retrieves a 32-bit value for a given PCD token.

Prototype

```
typedef
UINT32
(EFIAPI *PCD_PPI_GET32)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current 32-bit value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.Get64()

Summary

Retrieves a 64-bit value for a given PCD token.

Prototype

```
typedef
UINT64
(EFIAPI *PCD_PPI_GET64)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current 64-bit value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.GetPtr()

Summary

Retrieves a pointer to a value for a given PCD token.

Prototype

```
typedef
VOID *
(EFIAPI *PCD_PPI_GET_POINTER)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current pointer to the buffer for a PCD token number. Do not make any assumptions about the alignment of the pointer that is returned by this function call. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.GetBool()

Summary

Retrieves a Boolean value for a given PCD token.

Prototype

```
typedef
BOOLEAN
(EFIAPI *PCD_PPI_GET_BOOLEAN) (
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current Boolean-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.GetSize()

Summary

Retrieves the size of the value for a given PCD token.

Prototype

```
typedef
UINTN
(EFIAPI *PCD_PPI_GET_SIZE)(
  IN UINTN  TokenNumber
  );
```

Parameters

TokenNumber

The PCD token number.

Description

Retrieves the current size of the value for a particular PCD token. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.Get8Ex()

Summary

Retrieves an 8-bit value for a given PCD token.

Prototype

```
typedef
UINT8
(EFIAPI *PCD_PPI_GET_EX_8) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates from which namespace to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current byte-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.Get16Ex()

Summary

Retrieves a value for a given PCD token.

Prototype

```
typedef
UINT16
(EFIAPI *PCD_PPI_GET_EX_16) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current word-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.Get32Ex()

Summary

Retrieves a 32-bit value for a given PCD token.

Prototype

```
typedef
UINT32
(EFIAPI *PCD_PPI_GET_EX_32) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current 32-bit value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.Get64Ex()

Summary

Retrieves a 64-bit value for a given PCD token.

Prototype

```
typedef
UINT64
(EFIAPI *PCD_PPI_GET_EX_64) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the 64-bit value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.GetPtrEx()

Summary

Retrieves a pointer to the value for a given PCD token.

Prototype

```
typedef
VOID *
(EFIAPI *PCD_PPI_GET_EX_POINTER) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current pointer to the value for a PCD token number. There should not be any alignment assumptions about the pointer that is returned by this function call. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.GetBoolEx()

Summary

Retrieves a Boolean value for a given PCD token.

Prototype

```
typedef
BOOLEAN
(EFIAPI *PCD_PPI_GET_EX_BOOLEAN) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Description

Retrieves the current Boolean-sized value for a PCD token number. If the ***TokenNumber*** is invalid, the results are unpredictable.

PCD_PPI.Set8()

Summary

Sets an 8-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET8) (
  IN UINTN  TokenNumber,
  IN UINT8  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.Set16()

Summary

Sets a 16-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET16) (
  IN UINTN   TokenNumber,
  IN UINT16  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.Set32()

Summary

Sets a 32-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET32) (
  IN UINTN   TokenNumber,
  IN UINT32  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.Set64()

Summary

Sets a 64-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET64) (
  IN UINTN   TokenNumber,
  IN UINT64  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.SetPtr()

Summary

Sets a value of the specified size for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET_POINTER) (
  IN UINTN      TokenNumber,
  IN OUT UINTN  *SizeOfValue,
  IN VOID       *Buffer
  );
```

Parameters

TokenNumber

The PCD token number.

SizeOfValue

A pointer to the length of the value being set for the PCD token. On input, if the ***SizeOfValue*** is greater than the maximum size supported for this ***TokenNumber*** then the output value of ***SizeOfValue*** will reflect the maximum size supported for this ***TokenNumber*** .

Buffer

A pointer to the buffer containing the value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
EFI_SUCCESS	The PCD service has set the value requested
EFI_INVALID_PARAMETER	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
EFI_NOT_FOUND	The PCD service could not find the requested token number.

PCD_PPI.SetBoolean()

Summary

Sets a Boolean value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET_BOOLEAN) (
  IN UINTN    TokenNumber,
  IN BOOLEAN  Value
  );
```

Parameters

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.Set8Ex()

Summary

Sets an 8-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET_EX_8) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINT8           Value
);
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.Set16Ex()

Summary

Sets a 16-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET_EX_16) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINT16          Value
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.Set32Ex()

Summary

Sets a 32-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET_EX_32) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINT32          Value
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.Set64Ex()

Summary

Sets a 64-bit value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET_EX_64) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINT64          Value
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.SetPtrEx()

Summary

Sets a value of the specified size for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET_EX_POINTER) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN UINTN           SizeOfValue,
  IN VOID            Buffer
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

SizeOfValue

The length of the Value being set for the PCD token.

Buffer

A pointer to the buffer containing the value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.SetBoolEx()

Summary

Sets a Boolean value for a given PCD token.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_SET_EX_BOOLEAN) (
  IN CONST EFI_GUID  *Guid,
  IN UINTN           TokenNumber,
  IN BOOLEAN         Value
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value.

TokenNumber

The PCD token number.

Value

The value to set for the PCD token.

Description

When the PCD service sets a value, it will check to ensure that the size of the value being set is compatible with the Token's existing definition. If it is not, an error will be returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has set the value requested
`EFI_INVALID_PARAMETER`	The PCD service determined that the size of the data being set was incompatible with a call to this function. Use `GetSize()` to retrieve the size of the target data.
`EFI_NOT_FOUND`	The PCD service could not find the requested token number.

PCD_PPI.CallbackOnSet()

Summary

Specifies a function to be called anytime the value of a designated token is changed.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_CALL_BACK_ON_SET) (
  IN CONST EFI_GUID    *Guid, OPTIONAL
  IN UINTN             CallBackToken,
  IN PCD_PPI_CALLBACK  CallBackFunction
  );
```

Parameters

Guid

The 128-bit unique value that designates which namespace to monitor. If `NULL`, use the standard platform namespace.

CallBackToken

The PCD token number to monitor.

CallBackFunction

The function prototype that will be called when the value associated with the ***CallBackToken*** is set.

Related Definitions

```
typedef
VOID
(EFIAPI *PCD_PPI_CALLBACK) (
  IN EFI_GUID  *Guid, OPTIONAL,
  IN UINT      CallBackToken,
  IN VOID      *TokenData,
  IN UINTN     TokenDatSize
  );
```

Description

Specifies a function to be called anytime the value of a designated token is changed.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has successfully established a call event for the ***CallBackToken*** requested.
`EFI_NOT_FOUND`	The PCD service could not find the referenced token number.

PCD_PPI.CancelCallback()

Summary

Cancels a previously set callback function for a particular PCD token number.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_CANCEL_CALL_BACK) (
  IN CONST EFI_GUID    *Guid, OPTIONAL
  IN UINTN             CallBackToken,
  IN PCD_PPI_CALLBACK  CallBackFunction
  );
```

Parameters

Guid

The 128-bit unique value that designates which namespace to monitor. If `NULL`, use the standard platform namespace.

CallBackToken

The PCD token number to cancel monitoring.

CallBackFunction

The function prototype that was originally passed to the `CallBackOnSet` function.

Description

Cancels a callback function that was set through a previous call to the `CallBackOnSet` function.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has cancelled the call event associated with the ***CallBackToken***.
`EFI_INVALID_PARAMETER`	The PCD service did not match the ***CallBackFunction*** to one that is currently being monitored.
`EFI_NOT_FOUND`	The PCD service could not find data the requested token number.

PCD_PPI.GetNextToken()

Summary

Retrieves the next valid PCD token for a given namespace.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PPI_GET_NEXT_TOKEN) (
  IN CONST EFI_GUID  *Guid, OPTIONAL
  IN OUT UINTN       *TokenNumber
  );
```

Parameters

Guid

The 128-bit unique value that designates the namespace from which to extract the value. This is an optional parameter that may be `NULL`. If this parameter is `NULL`, then a request is being made to retrieve tokens from the default token space.

TokenNumber

A pointer to the PCD token number to use to find the subsequent token number.

Description

Retrieves the next valid token number in a given namespace. This is useful since the PCD infrastructure contains a sparse list of token numbers, and one cannot a priori know what token numbers are valid in the database.

If TokenNumber is 0 and Guid is not `NULL`, then the first token from the token space specified by Guid is returned. If TokenNumber is not 0 and Guid is not `NULL`, then the next token in the token space specified by Guid is returned. If TokenNumber is 0 and Guid is `NULL`, then the first token in the default token space is returned. If TokenNumber is not 0 and Guid is `NULL`, then the next token in the default token space is returned. The token numbers in the default token space may not be related to token numbers in token spaces that are named by Guid.

If the next token number can be retrieved, then it is returned in TokenNumber, and `EFI_SUCCESS` is returned. If TokenNumber represents the last token number in the token space specified by Guid, then `EFI_NOT_FOUND` is returned. If TokenNumber is not present in the token space specified by Guid, then `EFI_NOT_FOUND` is returned.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service has retrieved the value requested
`EFI_NOT_FOUND`	The PCD service could not find data from the requested token number.

PCD_PPI.GetNextTokenSpace()

Summary

Retrieves the next valid PCD token namespace for a given namespace.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *PCD_PROTOCOL_GET_NEXT_TOKEN) (
  IN OUT CONST EFI_GUID  **Guid
  );
```

Parameters

Guid

An indirect pointer to `EFI_GUID.` On input, it designates a known token namespace from which the search will start. On output, it designates the next valid token namespace on the platform. If ***Guid* is `NULL`, then the GUID of the first token space of the current platform is returned. If the search cannot locate the next valid token namespace, an error is returned and the value of ***Guid* is undefined.

Description

Gets the next valid token namespace for a given namespace. This is useful for traversing the valid token namespaces on a platform.

Status Codes Returned

Status Code	Description
`EFI_SUCCESS`	The PCD service retrieved the value requested.
`EFI_NOT_FOUND`	The PCD service could not find the next valid token namespace.