EDK II Package Declaration (DEC) File Format Specification

January 2016
Revision 1.25

Acknowledgements

Revision History

Revision	Revision History	Date
1.0	Initial release.	December 2007
1.1	Updated based on errata	August 2008
1.2	Updated based on enhancement requests	June 2009
1.21	Updated based on errata and for enhancement requests • Standardized the format for common content. • Added support for @Keyword Doxygen tag • Added support for @ModuleType Doxygen tags • Added support for @ValidList, @DefaultValue and @ValidRange Doxygen tags for PCDs • Added PKG_UNI_FILE element to [defines] section	January 2010
1.22	Errata and grammatical editing	April 2010

1.22 w/ Errata A	Updates: • Updated to support UEFI version 2.3.1 and updated spec release dates in Introduction • Clarify UEFI's PI Distribution Package Specification • Standardize Common data definitions for all specifications • Grammatical, formatting and spelling changes • Replaced "should" with wording saying that it is "recommended" • Added EBNF for <Extension> • Added scoping rules for Macros, clarified MACRO summary • Added an example of a binary only DEC file • Removed references to system environment variables in the Macros section • Specifically state where <MACROVAL> can be used, and where it is prohibited; specifically state that MACROVAL entries are expanded where they are used; clarify that MACROS are only expanded, not evaluated during initial parsing of the DEC file • Added table that shows that every part of a path name can be replaced by a MACROVAL • Clarify that C data arrays must be byte arrays for PCD value fields; prohibit C format and Registry Format GUID structures in VOID* PCD value fields • Update non-zero number is True, only 0 is consideered False • Prohibit specifying items as architecturally specific and also common • Changed <RegistryGuid> to <RegistryFormatGUID> in 3.4 • Defined <GuidValue> as a <CFormatGUID> for this release (need to allow registry format in a future release) • Update the [Includes], [Guids], [Protocols], [PPIs], [LibraryClasses] and PCD sections to allow an empty section • Updated the format for <QuotedString>, <CString> and <UnicodeString> • Update PATH related EBNF • Add <MacroDefinition> to [Defines], [Includes] and [LibraryClasses] sections • Provide rules in 2.2.6 for how macros can be shared between different subsections	December 2011
1.22 w/ Errata B	Updates: • Added a + after <Express> in the DoxComment definition of PCDs, as more than one expression can be specified in the UEFI PI Distribution Package Specification. • Added text describing the use of <HexDigit> for error numbers as well as how they are scoped.	June 2012

1.22 w/ Errata B (cont.)	Updates: • Updated UEFI/PI Spec version in chapter 1.3 to include Errata letters. • In Section 3.10 modified the optional error number in DoxComment definitions for PCDs from <HexDigit>+ to <ErrorCode> and defined <ErrorCode> to be of type <NumValUint32>; also added a "**|**" after the value to separate the error code from numeric values • Added AsBuilt entries for Abstract and Description • Clarified that the file must use the DOS end-of-line character sequence, **0x0D 0x0A**	June 2012
1.22 w/ Errata C	Updates: • Updated UEFI/PI Specification version support in chapter 1 • Modified examples to correct previous errors • Removed errors from text • Updated examples • Modified EBNF to prevent using the architectural modifier of common with any other architecture. Ensure that wording specifically states that the architecture modifiers are not case sensitive. • Add description of PCD processing rules in section 3.10 • Allow registry format GUID values in GUIDs, Protocols, PPIs sections instead of requiring C format GUID values (which will continue to be allowed) • The error codes are scoped to the TokenSpaceGUID, not to the PCD • Added reference to the EDK II Build Specification for PCD processing rules.	August 2013
1.24	Updates; • Change revision number of this specification from 1.22 to 1.24 • Updated **DEC_SPECIFICATION** to **0x00010017** • Added additional parameter definitions for clarification of the comment content for PCDs • Added the **PACKAGE_UNI_FILE** entry to the **[Defines]** section • Added reserved TianoCore user extension, for **"ExtraFiles"** • Added PCD comment type for # [Error] which is used to map an error code for a given token space to a specific string.	August 2014
1.24 w/ Errata A	Updates: • Changed DEC_SPECIFICATION to 0x00010018 and allow specifying it as a decimal, i.e., 1.24. • Updated specification dates and added two new specifications in section 1.2 • Removed expression EBNF as it has been replaced by the EDK II Expression Syntax Specification.	December 2014
1.24 w/ Errata B	Updates: • Update link to the EDK II Specifications, fixed the name of the Multi-String .UNI File Format Specification	March 2015

1.24 w/ Errata C	Updates: Clarify that #include statements are not permitted in UNI file specified in the PACKAGE_UNI_FILE entry	August 2015
1.25	Updates: • Specification revision to 1.25 • Revised WORKSPACE wording for updated build system that can handle packages located outside of the WORKSPACE directory tree (refer to the TianoCore.org/ EDKII website for additional instructions on setting up a development environment).	January 2015

Contents

Tables

1 Introduction

This document describes the EDK II Declaration (DEC) file format. This format was designed to support building packaging and distribution of EDK II modules, as well as for building platforms and modules using the EDK II build infrastructure. EDK II declaration files may be created during installation of a distribution that follows the UEFI Platform Initialization Distribution Package Specification. They may also be created manually.

The EDK II Build Infrastructure supports generation of UEFI 2.5 and PI 1.4 (Unified EFI, Inc.) compliant binary images.

This version of the specification clarifies (or enables) existing content, no new content has been introduced.

1.1 Overview

This document describes the format for DEC files with the following goals:

Compatible

The DEC Format must maintain backward compatibility with any existing DEC file formats. This means that the changes made to this specification must not require changes to existing DEC files.

Simplified platform build and configuration

One goal of this format is to simplify the build setup and configuration for a given platform. It was also designed to simplify the process of developing EDK II firmware components.

1.2 Terms

BaseTools

The BaseTools are the tools required for an EDK II build.

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.

DEC

EDK II Package Declaration File. This file declares information about what is provided in the package. An EDK II package is a collection of like content.

DEPEX

Module dependency expressions that describe runtime process restrictions.

Dist

This refers to a distribution package that conforms to the UEFI Platform Initialization Distribution Package Specification.

DSC

EDK II Platform Description File. This file describes what and how modules, libraries and components are to be built, as well as defining library instances which will be used when linking EDK II modules.

DXE

Framework Driver Execution Environment phase.

DXE SAL

A 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

A 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

EBNF

Extended "Backus-Naur Form" meta-syntax notation with the following additional constructs: square brackets "[…]" surround optional items, suffix "*" for a sequence of zero or more of an item, suffix "+" for one or more of an item, suffix "?" for zero or one of an item, curly braces "{…}" enclosing a list of alternatives, and super/subscripts indicating between n and m occurrences.

EDK

Extensible Firmware Interface Development Kit, the original implementation of the Intel® Platform Innovation Framework for EFI Specifications developed in 2007.

EDK II

EFI Development Kit, version II that provides updated firmware module layouts and custom tools, superseding the original EDK.

EDK Compatibility Package (ECP)

The EDK Compatibility Package (ECP) provides libraries that will permit using most existing EDK drivers with the EDK II build environment and EDK II platforms.

EFI

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

FDF

EDK II Flash definition file. This file is used to define the content and binary image layouts for firmware images, update capsules and PCI option ROMs.

FLASH

This term is used throughout this document to describe one of the following:

- An image that is loaded into a hardware device on a platform - traditional ROM image
- An image that is loaded into an Option ROM device on an add-in card
- A bootable image that is installed on removable, bootable media, such as a Floppy, CD-ROM or USB storage device.

- An image that is contains update information that will be processed by OS Runtime services to interact with EFI Runtime services to update a traditional ROM image.
- A UEFI application that can be accessed during boot (at an EFI Shell Prompt), prior to hand-off to the OS Loader.

Foundation

The set of code and interfaces that glue implementations of EFI together.

Framework

Intel® 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 EFI architecture.

GUID

Globally Unique Identifier. A 128-bit value used to name entities uniquely. 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. GUID values can be registry format (8-4-4-4-12) or C data structure format.

GUID also refers to an API named by a GUID.

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.

HOB

Hand-off blocks are key architectural mechanisms that are used to hand off system information in the early pre-boot stages.

IFR

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

INF

EDK II Module Information File. This file describes how the module is coded. For EDK, this file describes how the component or library is coded as well as providing some basic build information.

Source INF - An EDK II Module Information file that contains content in a [Sources] section and it does not contain a [Binaries] section. If the [Binaries] section is empty or the only entries in the [Binaries] section are of type DISPOSABLE, then the [Binaries] section is ignored.

Binary INF - An EDK II Module Information file that has a [Binaries] section and does not contain a [Sources] section or the [Sources] section is empty.

Mixed INF - An EDK II Module Information file that contains content in both [Sources] and [Binaries] sections and there are entries in the [Binaries] section are not of type DISPOSABLE

AsBuilt INF - An EDK II Module Information file generated by the EDK II build system when building source content (listed in a [Sources] section).

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.

Library Instance

An implementation of one or more library classes.

Module

A module is either an executable image or a library instance.

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. The EDK II build system also permits modules of type **USER_DEFINED**. These modules will not be processed by the EDK II Build system. See Table 2.

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.

PCD

Platform Configuration Database.

PEI

Pre-EFI Initialization Phase.

PEIM

An API named by a GUID.

PPI

A PEIM-to-PEIM Interface that is named by a GUID.

Protocol

An API named by a GUID.

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.

SAL

System Abstraction Layer. A firmware interface specification used on Intel® Itanium® Processor based systems.

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.

SKU

Stock Keeping Unit.

SMM

System Management Mode. A generic term for the execution mode entered when a CPU detects an SMI. The firmware, in response to the interrupt type, will gain control in physical

mode. For this document, "SMM" describes the operational regime for IA32 and x64 processors that share the OS-transparent characteristics.

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 2.4

Current UEFI version.

UEFI Platform Initialization Distribution Package Specification Version 1.0

The current version of this specification includes Errata B.

UEFI Platform Initialization Specification 1.3

Current version of the PI specification.

Unified EFI Forum

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

VFR

Visual Forms Representation.

VPD

Vital Product Data that is read-only binary configuration data, typically located within a region of a flash part. This data would typically be updated as part of the firmware build, post firmware build (via patching tools), through automation on a manufacturing line as the 'FLASH' parts are programmed or through special tools.

1.3 Related Information

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

- *Unified Extensible Firmware Interface Specification*, Version 2.5, Unified EFI, Inc, 2015, http://www.uefi.org.
- *Platform Initialization Specification*, Version 1.4, Unified EFI, Inc., 2015, http://www.uefi.org.
- *UEFI Platform Initialization Distribution Package Specification*, Version 1.0 with Errata B, Unified EFI, Inc., 2014, http://www.uefi.org.
- *Intel® Platform Innovation Framework for EFI Specifications*, Intel, 2007, http://www.intel.com/technology/framework/.
- https://github.com/tianocore/tianocore.github.io/wiki/EDK-II-Specifications
  - *EDK II Module Writers Guide,* Intel, 2010.
  - *EDK II User Manual*, Intel, 2010.
  - *EDK II C Coding Standard*, Intel, 2015.
  - *EDK II Build Specification,* Intel, 2016.
  - *EDK II DSC Specification*, Intel, 2016.
  - *EDK II FDF Specification*, Intel, 2016.
  - *EDK II INF Specification*, Intel, 2016.
  - *Multi-String UNI File Format Specification,* Intel, 2016.

- *EDK II Expression Syntax Specification,* Intel, 2015.
- *VFR Programming Language*, Intel, 2015.
- *UEFI Packaging Tool (UEFIPT) Quick Start,* Intel, 2015.
- *EDK II Platform Configuration Database Infrastructure Descriptions,* Intel, 2009.

- *INI file*, Wikipedia, http://en.wikipedia.org/wiki/INI_file.
- *C Now - C Programming Information*, Langston University, Tulsa Oklahoma, J.H. Young, 1999-2011, http://c.comsci.us/syntax/expression/ebnf.html.

1.4 Target Audience

This document is intended for persons doing EFI development and support for different platforms, as well as development and support for distributable modules. It is most likely only of interest in the event that there is a problem with a build or if a developer needs to perform special customizations of a build for a platform.

1.5 Conventions Used in this Document

This document uses typographic and illustrative conventions described below.

1.5.1 Data Structure Descriptions

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

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

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

Summary:
A brief description of the data structure.

Prototype:
An EBNF-type declaration for the data structure.

Example:
Sample data structure using the prototype.

1.5.2 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 *Extensible Firmware Specification*.

1.5.3 Typographic Conventions

This document uses the typographic and illustrative conventions described below:

Typographic Convention	Typographic convention description
Plain text	The normal text typeface is used for the vast majority of the descriptive text in a specification.
Plain text (blue)	Any plain text that is underlined and in blue indicates an active link to the cross-reference. Click on the word to follow the hyperlink.
Bold	In text, a **Bold** typeface identifies a processor register name. In other instances, a **Bold** typeface can be used as a running head within a paragraph.
Italic	In text, an *Italic* typeface can be used as emphasis to introduce a new term or to indicate a manual or specification name.
`BOLD Monospace`	Computer code, example code segments, and all prototype code segments use a `BOLD Monospace` typeface with a dark red color. These code listings normally appear in one or more separate paragraphs, though words or segments can also be embedded in a normal text paragraph.
`Bold Monospace`	Words in a `Bold Monospace` typeface that is underlined and in blue indicate an active hyper link to the code definition for that function or type definition. Click on the word to follow the hyper link.
$(VAR)	This symbol VAR defined by the utility or input files.
`Italic Monospace`	In code or in text, words in *`Italic Monospace`* indicate placeholder names for variable information that must be supplied (i.e., arguments).

Note: *Due to management and file size considerations, only the first occurrence of the reference on each page is an active link. Subsequent references on the same page will not be actively linked to the definition and will use the standard, non underlined* `BOLD Monospace` *typeface. Find the first instance of the name (in the underlined* `BOLD Monospace` *typeface) on the page and click on the word to jump to the function or type definition.*

The following typographic conventions are used in this document to illustrate the Extended Backus-Naur Form.

[item]	Square brackets denote the enclosed item is optional.
{item}	Curly braces denote a choice or selection item, only one of which may occur on a given line.
<item>	Angle brackets denote a name for an item.
(range-range)	Parenthesis with characters and dash characters denote ranges of values, for example, (a-zA-Z0-9) indicates a single alphanumeric character, while (0-9) indicates a single digit.
"item"	Characters within quotation marks are the exact content of an item, as they must appear in the output text file.
?	The question mark denotes zero or one occurrences of an item.

*	The star character denotes zero or more occurrences of an item.
+	The plus character denotes one or more occurrences of an item.
item$^{\{n\}}$	A superscript number, n, is the number occurrences of the item that must be used. Example: (0-9)^{8} indicates that there must be exactly eight digits, so 01234567 is valid, while 1234567 is not valid.
item$^{\{n,\}}$	A superscript number, n, within curly braces followed by a comma “,” indicates the minimum number of occurrences of the item, with no maximum number of occurrences.
item$^{\{,n\}}$	A superscript number, n, within curly braces, preceded by a comma “,”indicates a maximum number of occurrences of the item.
item$^{\{n,m\}}$	A super script number, n, followed by a comma “,“ and a number, m, indicates that the number of occurrences can be from n to m occurrences of the item, inclusive.

2
DEC File Overview

This document describes the format of EDK II package declaration (DEC) files. The DEC files are used by the EDK II utilities that parse EDK II DSC and EDK II INF files to generate *AutoGen.c* and *AutoGen.h* files for the EDK II build infrastructure.

There are no new features or format introduced in this specification.

This version of the specification reflects changes to the EDK II reference build system that has been updated to support builds using EDK II Packages that are located in directories outside of the directory specified by the system environment variable, WORKSPACE. Refer to the TianoCore.org web-site for more information on the EDK II build system.

An EDK II Package (directory) is a directory that contains an EDK II package declaration (DEC) file. Only one DEC file is permitted per directory. EDK II Packages cannot be nested within other EDK II Packages.

A DEC file describes content for a collection of 'like' modules located in a directory tree. Each collection of modules must be in a unique directory tree in the **WORKSPACE**, and must contain only one DEC file.

EDK II modules are located in subdirectories below the directory containing this file. If a module is a Library, the module directory must be created in the "*Library*" subdirectory. An "*Include*" subdirectory is also required, with the header file for the Library class placed in the sub-directory *Include/Library/*, and be called *LibraryClassName.h*. Additional content for the *Include* directory may include header files and potentially another *Industry Standard* directory.

Note: *Path and Filename elements within the DEC are case-sensitive in order to support building on UNIX style operating systems. Names that are used in C code are case sensitive as well as MACRO names used as short-cuts within the DEC file. Use of "..", "./" and "../" in path and filename elements is prohibited.*

Note: *GUID values are used during runtime to uniquely map the C names of PROTOCOLS, PPIS, PCDS and other variable names.*

Note: *The examples in this document use a backslash "\" character when the example line does not fit in between the margins. This character is not permitted in the actual DEC file, as all valid entries must appear on the same line.*

Note: *The total path and file name length is limited by the operating system and third party tools. It is recommended that for EDK II builds that the WORKSPACE (or directories listed in the PACKAGES_PATH system environment variable) directory be either a directory under a subst drive in Windows (s:/build as an example) or be located in either the /opt directory or in the user's / home/username directory for Linux and OS/X.*

2.1 Usage Overview

The DEC file supports EDK II module builds. The file is used to define specific information that will be shared between different EDK II Modules. There are eight possible major section types in the DEC file, Defines, Includes, LibraryClasses, Guids,

Protocols, Ppis, PCD and UserExtensions.

Within a DEC file, comments are encouraged, with the hash "#" character identifying a comment. All text after a comment character must be ignored by any parsing tool. Comment characters can be at the start of a line, or after a data element (there must be one or more white space characters between the data element and the comment character. Examples:

```
# this is a comment line
[includes.common] # This is also a valid comment.

[includes.common # This is not valid]
```

The last example is not valid, as the section header data element format is `[text]` with the square brackets included as part of the data element.

Note: *All EDK II DEC files MUST use the forward slash character for all directory paths specified.*

The remainder of this chapter discusses the different section content usage.

2.2 Declaration File Format

This section covers the content for the EDK II DEC files.

2.2.1 Section Entries

To simplify parsing, the EDK II meta-data files continue using the INI format. This style was introduced for EDK meta-data files, when only the Windows tool chains were supported. It was decided that for compatibility purposes, that INI format would continue to be used. EDK II formats differ from the defacto format in that the semi-colon ";" character cannot be used to indicate a comment.

Leading and trailing space/tab characters must be ignored.

Duplicate section names must be merged by tools.

This declaration file consists of sections delineated by section tags enclosed within square [] brackets. Section tag entries are case-insensitive. The different sections and their usage are described below. The text of a given section can be used for multiple section names by separating the section names with a comma. For example:

```
[Includes.X64, includes.IPF]
```

The content below each section heading is processed by the parsing utilities in the order that they occur in the file. The precedence for processing these architecture section tags is from right to left, with sections defining an architecture having a higher precedence than a section which uses common (or no architecture extension) as the architecture.

It is not permissible to have an architectural modifier in the same section tag as an entry with the common architectural modifier. Specifying entries as only for IA32 and also valid for all other architectures (`[Includes.common, Includes.IA32]`) is not valid.

Note: *Content such as filenames, directory names, MACROs and C variable names within a section IS case sensitive. IA32, Ia32 and ia32 within a section are processed as separate items.IA32, Ia32*

and ia32 within a section in a directory or file name are processed as separate items. (Refer to Naming Conventions below for more information on directory and/or file naming.)

Sections are terminated by the start of another section or the end of the file.

Comments are not permitted between square brackets of a section specifier.

Duplicate sections (two sections with identical section tags) will be merged by tools, with the second section appended to the first.

If architectural modifiers are used in the section tag, the section is merged by tools with content from common sections (if specified) with the architectural section appended to the first, into an architectural section. For example, given the following:

```
[Includes]
Includes/
[Includes.IA32]
Includes/Ia32
[Includes.X64]
Includes/X64
```

After the first pass of the tools, when building the module for IA32, the source files will logically be:

```
[Includes.IA32]
Includes/
Includes/Ia32
```

When building the module for X64, the source files will logically be:

```
[Includes.X64]
Includes/
Includes/X64
```

The **[Defines]** section tag prohibits use of architectural modifiers. All other sections can specify architectural modifiers.

2.2.2 Comments

The hash **#** character indicates comments in the Declaration (DEC) file. In line comments terminate the processing of a line. In line comments must be placed at the end of the line, and may not be placed within the section (**[,]**) tags.

Only **gPkgTSGuid.PcdFoo|TRUE|BOOLEAN|0x00000015** in the following example is processed by tools; the remainder of the line is ignored:

```
gPkgTSGuid.PcdFoo|TRUE|BOOLEAN|0x00000015    # EFI_COMPUTING_UNIT_MEMORY
```

Note: *Blank lines and lines that start with the hash # character must be ignored by tools.*

Hash characters appearing within a quoted string are permitted, with the string being processed as a single token. The following example must handle the quoted string as single element by tools.

```
UI = "# Copyright 2007, No Such, LTD. All rights reserved."
```

The line extension character, "\", cannot be used to extend a comment. Like the comment character stops processing of a line, comments are always terminated by the end of line. Doxygen tags are permitted in comment blocks preceding individual GUID, Protocol, PPI and PCD entries. These tags are used as containers for content required by the UEFI Packaging specification, and may also be used by tools. Each section will define the valid Doxygen tags which apply.

If a hash "**#**" character is required in a value field, the value field must be encapsulated by double quotation marks.

<CommentBlock> Entries

Various elements in the DEC file have a recommended format for comment information regarding the header files, module types an item supports and other information. These special comment blocks are processed by tools used to create a distribution package of the code that conforms to the UEFI Distribution Package (UDP) Specification. Tools used to install a distribution package that conforms to the UDP must add appropriate type information in these comment blocks. The comment block formats are specified in chapter 3 of this document.

The general format of these comment blocks in the **[Guids]**, **[Protocols]** and **[Ppis]** sections is:

```
"##" Path/To/HeaderFile.h
GUID_C_Name = <GUID> ["##" <ModuleTypeList>] ["#" <HelpText>]
```

Example:

```
## Include/Guid/GlobalVariable.h
gEfiGlobalVariableGuid = {0x8BE4DF61,0x93CA,0x11D2, \
  {0xAA,0x0D,0x00,0xE0,0x98,0x03,0x2B,0x8C}}

## Include/Protocol/DebugPort.h
gEfiDebugPortDevicePathGuid = {0xEBA4E8D2, 0x3858, 0x41EC, \
  {0xA2, 0x81, 0x26, 0x47, 0xBA, 0x96, 0x60, 0xD0}} ## UEFI_DRIVER

# Guid for EFI_DISK_INFO_PROTOCOL.Interface to specify Ide interface.
## Include/Protocol/DiskInfo.h
gEfiDiskInfoIdeInterfaceGuid = {0x5E948FE3, 0x26D3, 0x42B5, \
  {0xAF, 0x17, 0x61, 0x02, 0x87, 0x18, 0x8D, 0xEC}} ## DXE_DRIVER,
UEFI_DRIVER

## Include/Guid/SmmCommunicate.h
gSmmCommunicateHeaderGuid = {0xf328e36c, 0x23b6, 0x4a95, \
  {0x85, 0x4b, 0x32, 0xe1, 0x95, 0x34, 0xcd, 0x75} \
  } ## SMM_CORE, DXE_SMM_DRIVER
```

Note: *In the above example, the line has been extended so that the field is continued to the last line (the "}" character) so that the comment at the end of the line can be processed correctly by the Intel(r) UEFI Packaging Tool.*

Additional informational help text is also defined in the **<CommentBlock>** tag. The format defined for comment blocks that are at the end of lines listed in all of the examples must not continue on following lines. If the **<CommentBlock>** information is long, the information is allowed to be split into multiple comment lines that immediately precede the element. For example:

```
## Include/Guid/DebugAgentGuid.h
## PEIM, DXE_DRIVER, DXE_SMM_DRIVER
# MdeModule Debug Agent GUID
gEfiDebugAgentGuid = \
{0x865a5a9b,0xb85d,0x474c,{0x84,0x55,0x65,0xd1,0xbe,0x84,0x4b,0xe2}}
```

Unlike GUIDs, Protocols and PPIs, the PCD entries are not associated with a header file, so the general format is:

```
## CommentBlock
[# CommentBlockCont]
Entry [<ShortSingleCommentBlock>]
```

Example:

```
## DXE_DRIVER, DXE_SMM_DRIVER # S3 support
# The PCD is used to specify memory size with page number for a
# pre-allocated ACPI NVS memory to be used by PEI in S3 phase.
# The default size 32K.
# When changing the value of this PCD, the platform developer should
# make sure the memory size is large enough to meet PEI requirement in
# S3 phase.
gEfiTModPkgTokenSpaceGuid.PcdS3AcpiReservedMemorySize | \
  0x8000 | UINT32 | 0x30000007
```

2.2.3 Valid Entries

Processing of the line is terminated if a comment is encountered or by the end of the line. Entries in this file (not comments) are not allowed to span multiple lines.

Items in quotation marks are treated as a single token and have the highest precedence. All expressions must be written using in-fix notation (operators are written between the operands). Parenthesis surrounding groups of operands and operators are recommended to determine the order in which operations are to be performed to remove ambiguity. All other processing occurs from left to right.

In the following example, B - C is processed first, then result is added to A followed by adding 2; finally 3 is added to the result.

```
(A + (B - C) + 2) + 3
```

In the next example, A + B is processed first, then C + D is processed and finally the two results are added.

```
(A + B) + (C + D)
```

Space and tab characters are permitted around field separators.

2.2.4 Naming Conventions

The EDK II build infrastructure is supported under Microsoft* Windows*, Linux* and MAC OS/X* operating systems. As a result of multiple environment support, all directory and file names must be treated as case sensitive.

- The use of special characters in directory names and file names is restricted to the dash, underscore, and period characters, respectively "-", "_", and ".".

- Period characters must not be followed by another period character. File and Directory names must not start with "./", "." or "..".
- Directory names and file names must not contain space or tab characters.
- Directory Names must only contain alphanumeric, underscore, dash and period characters (period characters must not be sequential) and it is recommended that they start with an alpha character.
- All files must reside in the directory containing the DEC file or in sub-directories of the directory containing the DEC file.
- It is recommended that filenames start with an alpha character.
- All EDK II directories that are architecturally dependent must use a name with only the first character capitalized followed by lower case characters or numeric characters. Ia32, Ipf, X64 and Ebc are valid architectural directory names. IA32, IPF and EBC are not acceptable directory names, and may cause build breaks. From a build tools perspective, an IA32 directory name is not equivalent to Ia32 or ia32.
- When an architecture is used in a directory name, the directory content must be listed in a section that uses the matching architecture modifier. If a common section contains filenames that have directories with architecture modifiers, the file will be processed for all architectures, not just the architecture specified in the directory name.
- Absolute paths are not permitted in EDK II DEC files. All paths specified are relative to the EDK II package directory containing the DEC file.

Space Characters in filenames: The build tools must be able to process the tool definitions file: tools_def.txt (describing the location and flags for compiler and user defined tools), which may contain space characters in paths on Windows* systems.

The EDK II Coding Style specification covers naming conventions for use within C Code files, and as well as specifying the rules for directory and file names. This section is meant to highlight those rules as they apply to the content of the DEC files.

Architecture keywords (IA32, IPF, X64 and EBC) are used by build tools and in meta-data files for describing alternate threads for processing of files. These keywords must not be used for describing directory paths. Additionally, directory names with architectural names (Ia32, Ipf, X64 and Ebc) do not automatically cause the build tools or meta-data files to follow these alternate paths. Directories and Architectural Keywords are similar in name only.

For clarity, this specification will use all upper case letters when describing architectural keywords, and the directory names with only the first letter in upper case.

All directory paths within EDK II DEC files must use the "/" forward slash character to separate directories as well as directories from filenames.

Example:

```
C:/Work/Edk2/edksetup.bat
```

File names must also follow the same naming convention required for directories. No white space characters are permitted. The special characters permitted in directory names are the only special characters permitted in file names.

Absolute paths or relative paths outside of the directory the DEC file resides must not be used when specifying directories or filenames in any section of the DEC file.

2.2.5 !include Statements

The `!include` statement is NOT permitted in DEC files.

2.2.6 Macro Statements

Macro statements are permitted in the EDK II DEC files. Macro statements assign a Value to a Variable Name, and are only valid during the processing of the DEC specifying the value. Macro statements are local to the file - global macro values are not permitted. Use of system environment variables is also prohibited in value fields; they may appear in comments, however during the build, comment content is ignored. This decision was made in order to support UEFI's PI Distribution Package Specification requirements.

Macro Definition statements that appear within a section of the file (other than the `[Defines]` section) are scoped to the section they are defined in. If the Macro statement is within the `[Defines]` section, then the Macro is common to the entire file, with local definitions taking precedence (if the same MACRO name is redefined in subsequent sections, then the MACRO value is local to only that section).

Any defined MACRO definitions will be expanded by tools when they encounter the entry in the section. All macros are local to the DEC file (this is a requirement for UEFI distribution of source and binary content).

The macro statements are positional, in that only statements following a macro definition are permitted to use the macro – a macro cannot be used before it has been defined.

Macros defined the `[Defines]` section are common to all sections.

Macros defined in a common architectural section may be used in the architecturally modified sections of the same section type. Macros defined in architectural sections cannot be used in other architectural sections, nor can they be used in the common section.

Macro expansion is done at the time the macro is used.

Example

```
[LibraryClasses.common]
# Can use $(MDE)
# Cannot use either $(SMM) or $(SAL)
DEFINE MDE = Include/Library
BaseLib|$(MDE)/BaseLib.inf

[LibraryClasses.X64, LibraryClasses.IA32]
# Can use $(MDE) and local $(SMM)
# Cannot use $(SAL)
DEFINE SMM = Include/Library
SmmLib|$(SMM)/SmmLib.h

[LibraryClasses.IPF]
# Can use $(MDE) from the common section and local $(SAL)
# Cannot use $(SMM)
DEFINE SAL = Include/Library
SalLib|$(SAL)/SalLib.h
PalLib|$(MDE)/PalLib.h
```

In the previous example, the directory and filename for a library instance is the header file that can be used for all modules that provide the library implementations that conform to the definitions in the file.

2.2.7 PCD Names

Unique PCD names are defined as PCD Token Space Guid C name and the PCD C name - separated by a period “.” character:

PcdTokenSpaceGuidCName.PcdCName

The PCD's Name **(PcdName)** is defined as PCD Token Space Guid C name and the PCD C name separated by a period "**.**" character. PCD C names are used in C code and must follow the C variable name rule.

2.2.8 Conditional Directive Statements (!if...)

Conditional statements are NOT permitted in EDK II DEC files.

2.3 EDK II DEC Format

EDK II DEC files can be created by package installation tools using the UEFI Distribution Package description files that accompany a distribution package. They may also be created manually.

All content (except section tag names) within the EDK II DEC file is case-sensitive.

2.4 [Defines] Usage

This is a required section.

The **[Defines]** section is used to track the package's GUID and version, which allows

multiple copies of the same package with different versions to be processed by tools. Architectural modifiers are not permitted in the **[Defines]** section. If major changes to a package occur, the GUID value of the package must also change.

The DEC_SPECIFICATION of existing DEC files does not need to be updated unless content in the file has been updated to match new content specified by this revision of the specification. Additionally, the package's version major number may change. Minor changes require incrementing the package's version minor number. The **PACKAGE_UNI_FILE** entry points to a Unicode file containing localization strings. The use of the #include statement in this file is prohibited. The file path (if present) is relative to the directory containing the DEC file.

The parsing utilities process any local symbol assignments defined in this section. Note that the sections are processed in the order listed in the DEC file, and later assignments of these local symbols override previous assignments.

This section will use only one section header:

[Defines]

The format for entries in this section is:

Name = Value

The following is an example of this section.

```
[Defines]
DEC_SPECIFICATION  = 0x00010019
PACKAGE_NAME       = MdePkg
PACKAGE_GUID       = 1E73767F-8F52-4603-AEB4-F29B510B6766
PACKAGE_VERSION    = 1.02
PACKAGE_UNI_FILE   = MdePkg.uni
```

2.5 [Includes] Usage

This is an optional section.

The **[Includes]** section is used to identify the "standard" location "include directories" provide by this EDK II package. The **[Includes]** contains a list of package relative directory names. These directories contain sub-directories or header files. If the Package directory contains the directories, and the *Include*, *Include/Ppis*, *Include/Protocol* and *Include/Guid*, and header files exist in each of these directories, then all four directories will be listed in this section.

This list of directories is used by the build tools to create the list of standard directory locations required by compilers.

Also included in this section are the directories containing headers that may be required for individual EDK II module types. Refer to Appendix, "*EDK II Module Types*", for a list of the valid types.

Refer to the **[Includes]** definition later in this document for a complete description of this section and its contents.

The **[Includes]** section uses one of the following section definitions:

[Includes.common] [Includes.IA32] [Includes.X64] [Includes.IPF] [includes.EBC] [Includes]

The format for entries in this section is one field, with an optional comment "**#**" field as shown below:

```
Package_Relative/path  # Comment such as Keyword List
```

The relative path is relative to the directory the DEC file is in. Use of "..", "./" and "../" in the directory path is not permitted.

Caution: *Do not list individual files in the* **[Includes]** *section.*

2.6 [Guids] Usage

This is an optional section.

This section is used to define the GUID Value for Guid C Names.

This section uses one of the following section definitions:

[Guids] [Guids.IA32] [Guids.X64] [Guids.IPF] [Guids.EBC]
[Guids.common] [Guids.IA32, Guids.X64] [Guids.X64, Guids.IPF]

Format for the entries in this section is two fields with an equal "=" character separating the fields as shown below.

```
GuidCName = {C Format Guid Value}  # Comment
```

The Comment section can be used to identify the list of supported module types.

2.7 [Protocols] Usage

This is an optional section.

This section is used to define the GUID Value for Protocol C Names.

This section use ones of the following section definitions:

[Protocols] [Protocols.IA32] [Protocols.X64] [Protocols.IPF]
[Protocols.EBC] [Protocols.common]

Format for the entries in this section is two fields with an equal "=" character separating the fields as shown below.

```
ProtocolCName = {C Format Guid Value}  # Comment
```

The Comment section can be used to identify the list of supported module types.

2.8 [Ppis] Usage

This is an optional section.

This section is used to define the GUID Value for PPI C Names.

This section use ones of the following section definitions:

[Ppis] [Ppis.IA32] [Ppis.X64] [Ppis.IPF] [Ppis.EBC] [Ppis.common]

Format for the entries in this section is two fields with an equal "=" character separating the fields as shown below.

```
PpiCName = {C Format Guid Value}  # Comment
PpiCname1 = RegistryFormatGUID    # Comment
```

The Comment section can be used to identify the list of supported module types.

2.9 [LibraryClasses] Usage

This is an optional section.

This section is used to define the headers associated with the new EDK II library classes. A library class is declared and its associated header file specified in this section. The module library instances that satisfy a Library Class must use the Library Class Header file, without modification.

Refer to the **[LibraryClasses]** definition later in this document for a complete description of this section and its contents.

This section uses one of the following section definitions:

[LibraryClasses.common] [LibraryClasses.IA32] [LibraryClasses.X64] [LibraryClasses.IPF] [LibraryClasses.EBC] [LibraryClasses]

Format for the entries in this section is two fields, with a pipe "|" character as the field separator, as shown below.

```
LibraryClassName | Relative/path/and/header_filename.h
```

The relative path is relative to the directory the DEC file is in. Use of "..", "../" or "./" in the directory path is prohibited.

2.10 PCD Usage

These are optional sections.

These sections are used to declare basic information about PCDs. Refer to the *PI Specification* as well as the EDK II Platform Configuration Database Infrastructure Descriptions document for additional information regarding PCDs. Only the DynamicEx access methods are defined in the *PI Specification*; the remaining types are specific to this EDK II Build System.

Generally, PCDs using the FixedAtBuild and PatchableInModule access methods are used to set static configuration elements that can be determined at build time (or modified prior to inserting a module into a flash image). PCDs using the Dynamic and DynamicEx access methods are used for configuration knobs that may be manipulated by setup screens (or other methods) during the boot process.

Information in this section is used to create entries in the *AutoGen.c* and *AutoGen.h* files for EDK II modules.

The PCD is used for configuration when the PCD value is produced and consumed by drivers during execution, the value may be user configurable from setup or the value is produced by the platform in a specified area. It is associated with modules that are released in source code. The PatchableInModule and DynamicEx PCD access methods are associated with modules that are released as binary only modules. The FeatureFlag PCD is used to enable some code paths.

PCDs are usually defined by a specification that defines the name, token number, token space GUID and datum type. A default value and valid access methods may also be given in the industry specification. If a developer needs to create a new PCD, they can, following the conventions listed in the PI specification. Only PCDs that will be shared between multiple users need to be defined in published architectural specs. If a PCD is only going to be used by a single organization, then a new PCD can be created within the organization, keeping all modules that use the PCD internal to the organization.

Every PCD (**PcdName**) is identified by a two part definition - the PCD's Token Space Guid CName and the PCD CName. These two parts are separated by a period "." character. Together, these two parts make up the first field in a PCD Entry. When the AutoGen code is created, the value of the Token Space GUID and the token value are used to uniquely identify the

PCD.

Refer to the PCD Sections definition later in this document for a complete description of this section and its contents.

This section resembles one of the following section definitions:

```
[PcdsFeatureFlag] [PcdsFeatureFlag.common] [PcdsFixedAtBuild.IA32]
[PcdsPatchableInModule.X64] [PcdsDynamic.IPF] [PcdsDynamicEx.EBC]
```

The EDK II build system supports five PCD access methods: FeatureFlag, FixedAtBuild, PatchableInModule, Dynamic and DynamicEx. These indicate access methods for get/set operations. The PcdsDynamicEx method is defined by the *PI Specification*. A PCD may be listed under multiple PCD access method sections, except FeatureFlag PCDs. Listing a PCD in multiple sections indicates that modules have been coded to use in any one of the non-FeatureFlag access methods.

It recommended that modules use either the FeatureFlag PCD or use the flexible (INF file's [Pcd] section) for access. If a module is coded for only one type of access, such as FixedAtBuild, then it can only use the FixedAtBuild access method in a platform, and therefore, it must not be listed in any other section types in this file. Likewise, if the module is coded as a DynamicEx form, then it can only be listed in the DynamicEx section. If a module is coded for the Dynamic access method, then the platform integrator would be able to choose how they want to use the PCD. It can then be specified to use either a FixedAtBuild, PatchableInModule, Dynamic or DynamicEx access method in the platform description (DSC) file. Using the Dynamic access method in module code is the most flexible method, as platform integrators may chose a to use a different type (such as fixed) for a given platform without modifying the module's INF file or the code for the module.

The EDK II build tools will automatically generate a const definition for the FeatureFlag and FixedAtBuild PCDs, while the PatchableInModule and both Dynamic forms will have a volatile definition generated.

The two recommended access methods that are commonly specified in modules INF files are: FeatureFlag (list in [FeaturePcd] sections) and Dynamic (listed in [PCD] sections). For modules that will be distributed as binary modules, PCDs in those modules that will be exposed by the binary must use PatchableInModule or DynamicEx access mehtods.

It is recommended that PCDs be listed in PcdsFixedAtBuild, PcdsPatchableInModule, PcdsDynamic and PcdsDynamicEx sections in the DEC file.

Module developers need to check what sections a specific PCD is listed in, in order to code the module using the correct access type. Also, PCDs may have different default values for different architectures.

The format for entries in the PCDs sections is four fields, with a pipe "|" character as the field separator, as shown below:

```
# Comment
TokenSpaceGuidCname.PcdCname|DefaultValue|DatumType|Token
```

The Comment section can be used to identify the list of supported module types as well as to contain conditional test statements for acceptable values.

Default values listed in this file can be overridden by the default values specified in INF files (provided all INF files use the same value for a PCD) or by values specified in the DSC or FDF files of a platform.

The Token value is used programmatically in code. PCD Token numbers must be unique to a Token Space GUID name space. The two PCD drivers use the token number to locate a PCD's value.

FeatureFlag PCDs

FeatureFlag PCDs can only be listed in FeatureFlag access method sections in the EDK II meta-data files. The datum type of a FeatureFlag PCD must be BOOLEAN.

VOID* PCD DatumType

The declarations in this file do not include a maximum datum size for the "**VOID***" PCDs. It is recommended that the platform integrator allocate space for the content, rather than depend on letting tools compute the maximum value based on the greater of the lengths from the values in the DEC, DSC and INF files. However, if the platform integrator does not specify a size in the DSC file, the data size is calculated by the tools to be the greatest length of all values specified for this PCD listed in the DEC, INF, FDF and DSC files.

2.11 [UserExtensions] Usage

This is an optional section.

The EDK II user extensions section allows for extending the DEC files with custom processing. The format for a user extension is:

[UserExtensions.$(UserID).$(Identifier)]

Data elements under the section header are not required; this is an optional section. Content of in this section is free form.

The EDK II build tools do not use this section and will ignore all content within a **[UserExtensions]** section.

The following is an example of a User Extensions section.

```
[userextensions.NoSuchCorp."Script_1.0"]
  NoSuch.bat
```

2.11.1 [UserExtensions.TianoCore."ExtraFiles"] Section

The EDK II **[UserExtensions.TianoCore."ExtraFiles"]** section allow for distributing extraneous files that are associated with a package. Files listed in this section are not processed by EDK II build tools. These files must exist in the directory or sub-directories of the directory containing the DEC file.

Note: *The Intel® UEFI Packaging Tool will parse this section and for all files listed in this file, add the file to the package distribution using the UEFI Distribution Package Distribution.*

The section header must be:

[UserExtensions.TianoCore."ExtraFiles"]

Having data elements under the section header is not required.

The following is an example of a **[UserExtensions.TianoCore."ExtraFiles"]** section:

```
[UserExtensions.TianoCore."ExtraFiles"]
Readme.txt
UserManual.pdf
```

3
EDK II DEC File Format

This section of the document describes the EDK II DEC sections using an Extended Backus-Naur Form.

3.1 General Rules

The general rules for EDK II INI style documents follow.

Note: *Path and Filename elements within the DEC are case-sensitive in order to support building on UNIX style operating systems. Additionally, names that are C variables or used as a macro are case sensitive. Other elements such as section tags or hex digits, in the DEC file are not case-sensitive. The use of "..", "../" and "./" in paths and filenames is strictly prohibited.*

- Only one DEC file is permitted per directory.
- Text in section tags (text within square brackets) is not case sensitive.
- A section terminates with either another section definition or the end of the file.
- Entries terminate with either a comment or the end of line character sequence.
- To append comment information to any item, the comment must start with a hash "#" character.
- All comments terminate with the end of line character sequence.
- Any comment not associated with a defined comment format is considered a global comment.
- Global comments must be separated from formatted comments with a blank line.
- Comment lines cannot be extended using the line extension character.
- Field separators for lines that contain more than one field are pipe "|" characters. This character was selected to reduce the possibility of having the field separator character appear in a string, such as a filename or text string.

Note: *The only notable exception is the PcdName which is a combination of the PcdTokenSpaceGuidCName and the PcdCName that are separated by the period "." character. This notation for a PCD name is used to uniquely identify the PCD.*

- When processing numeric values, either integer or hex, leading zeros specified in the entry may be ignored. For example, `0x00000000000000000000001` can be a valid value for a **`UINT8`** data type, as the actual value is `1`.

3.1.1 Backslash

The backslash "\" character is used in this document when an example entry cannot fit between the margins of this file. It must not be used in the DEC file to extend an entry.

3.1.2 White space characters

Whitespace (space and tab) characters are permitted between token and field separator

elements for all entries.

Whitespace characters are not permitted between the PcdTokenSpaceGuidCName and the dot, nor are they permitted between the dot and the PcdCName.

3.1.3 Paths for filenames

Note that for specifying the path for a file name, if the path value starts with a dollar sign "$" character, a local MACRO is being specified. White space characters are not permitted in path names.

Caution: *The use of "..", "./" and "../" in a path element is prohibited.*

For all EDK II DEC files, the directory path must use the forward slash character for separating directories. For example, MdePkg/Include/ is Valid.

Unless otherwise noted, all file names and paths must be relative to the directory where the DEC file is located.

3.2 Package Declaration (DEC) Definitions

This section defines content and format of a package declaration file. The **[Defines]** section must appear before any other section except the **<Header>** comment block. (The header, when specified, is always the first section of a DEC file.) The remaining sections may be specified in any order within the DEC file.

Summary

The EDK II Package Declaration (DEC) file has the following format (using the EBNF).

```
<EDK_II_DEC>  ::=  <Header>?
                   <Defines>
                   <Includes>*
                   <LibraryClass>*
                   <Guids>*
                   <Protocols>*
                   <Ppis>*
                   <Pcd>*
                   <UserExtensions>*
```

3.2.1 Common Definitions

Summary

The following are common definitions used by multiple section types in EDK II meta-data documents. Not all of the definitions below pertain to entries in the DEC file (for example, **<Expression>** statements are not permitted).

Prototype

```
<Word>            ::=  (a-zA-Z0-9_)(a-zA-Z0-9_-.)* Alphanumeric
                       characters with optional period ".", dash "-"
                       and/or underscore "_" characters.
                       A period character may not be followed by
                       another period character.
                       No white space characters are permitted.

<SimpleWord>      ::=  (a-zA-Z0-9)(a-zA-Z0-9_-)* A word that cannot
                       contain a period character.

<ToolWord>        ::=  (A-Z)(a-zA-Z0-9)* Alphanumeric characters.
                       white space characters are not permitted.

<FileSep>         ::=  "/"

<Extension>       ::=  (a-zA-Z0-9_-)+ One or more alphanumeric
                       characters.

<File>            ::=  <Word> ["." <Extension>]

<PATH>            ::=  [<MACROVAL> <FileSep>] <RelativePath>

<RelativePath>    ::=  <DirName> [<FileSep> <DirName>]*

<DirName>         ::=  {<Word>} {<MACROVAL>}

<FullFilename>    ::=  <PATH> <FileSep> <File>

<Filename>        ::=  [<PATH> <FileSep>] <File>

<Chars>           ::=  (a-zA-Z0-9_)

<Digit>           ::=  (0-9)

<NonDigit>        ::=  (a-zA-Z_)

<Identifier>      ::=  <NonDigit> <Chars>*

<CName>           ::=  <Identifier>  # A valid C variable name.

<AsciiChars>      ::=  (0x21 - 0x7E)

<CChars>          ::=  [{0x21} {(0x23 - 0x5B)} {(0x5D - 0x7E)}
                       {<EscapeSequence>}]*

<DblQuote>        ::=  0x22

<EscapeSequence>  ::=  "\" {"n"} {"t"} {"f"} {"r"} {"b"} {"0"} {"\"}
                       {<DblQuote>}
```

```
<TabSpace>          ::=  {<Tab>} {<Space>}
<TS>                ::=  <TabSpace>*
<MTS>               ::=  <TabSpace>+
<Tab>                 ::=  0x09
<Space>               ::=  0x20
<CR>                  ::=  0x0D
<LF>                  ::=  0x0A
<CRLF>                ::=  <CR> <LF>
<WhiteSpace>          ::=  {<TS>} {<CR>} {<LF>} {<CRLF>}
<WS>                  ::=  <WhiteSpace>*
<Eq>                  ::=  <TS> "=" <TS>
<FieldSeparator>      ::=  "|"
<FS>                  ::=  <TS> <FieldSeparator> <TS>
<Wildcard>            ::=  "*"
<CommaSpace>          ::=  "," <Space>*
<Cs>                  ::=  "," <Space>*
<AsciiString>         ::=  [ <TS>* <AsciiChars>* ]*
<EmptyString>         ::=  <DblQuote><DblQuote>
<CFlags>              ::=  <AsciiString>
<PrintChars>          ::=  {<TS>} {<CChars>}
<QuotedString>        ::=  <DblQuote> <PrintChars>* <DblQuote>
<CString>             ::=  ["L"] <QuotedString>
<NormalizedString>    ::=  <DblQuote> [{<Word>} {<Space>}]+ <DblQuote>
<GlobalComment>       ::=  <WS> "#" [<TS> <AsciiString>] <EOL>+
<Comment>             ::=  "#" <TS> <AsciiString> <EOL>+
```

```
<UnicodeString>     ::=  "L" <QuotedString>

<HexDigit>          ::=  (a-fA-F0-9)

<HexByte>           ::=  {"0x"} {"0X"} [<HexDigit>] <HexDigit>

<HexNumber>         ::=  {"0x"} {"0X"} <HexDigit>+

<HexVersion>        ::=  "0x" [0]* <Major> <Minor>

<Major>             ::=  <HexDigit>? <HexDigit>? <HexDigit>?
                         <HexDigit>

<Minor>             ::=  <HexDigit> <HexDigit> <HexDigit> <HexDigit>

<DecimalVersion>    ::=  {"0"} {(1-9) [(0-9)]*} ["." (0-9)+]

<VersionVal>        ::=  {<HexVersion>} {(0-9)+ "." (0-99)}

<GUID>              ::=  {<RegistryFormatGUID>} {<CFormatGUID>}

<RegistryFormatGUID>  ::=  <RHex8> "-" <RHex4> "-" <RHex4> "-"
                           <RHex4> "-" <RHex12>

<RHex4>             ::=  <HexDigit> <HexDigit> <HexDigit> <HexDigit>

<RHex8>             ::=  <RHex4> <RHex4>

<RHex12>            ::=  <RHex4> <RHex4> <RHex4>

<RawH2>             ::=  <HexDigit>? <HexDigit>

<RawH4>             ::=  <HexDigit>? <HexDigit>? <HexDigit>?
                         <HexDigit>

<OptRawH4>          ::=  <HexDigit>? <HexDigit>? <HexDigit>?
                         <HexDigit>?

<Hex2>              ::=  {"0x"} {"0X"} <RawH2>

<Hex4>              ::=  {"0x"} {"0X"} <RawH4>

<Hex8>              ::=  {"0x"} {"0X"} <OptRawH4> <RawH4>

<Hex12>             ::=  {"0x"} {"0X"} <OptRawH4> <OptRawH4> <RawH4>

<Hex16>             ::=  {"0x"} {"0X"} <OptRawH4> <OptRawH4>
                         <OptRawH4> <RawH4>

<CFormatGUID>       ::=  "{" <Hex8> <CommaSpace> <Hex4> <CommaSpace>
                         <Hex4> <CommaSpace> "{"
```

```
                                <Hex2> <CommaSpace> <Hex2> <CommaSpace>
                                <Hex2> <CommaSpace> <Hex2> <CommaSpace>
                                <Hex2> <CommaSpace> <Hex2> <CommaSpace>
                                <Hex2> <CommaSpace> <Hex2> "}" "}"

<CArray>                ::=  "{" {<NList>} {<CArray>} "}"

<NList>                 ::=  <HexByte>
                             [<CommaSpace> <HexByte>]*

<RawData>               ::=  <TS> <Number> [<Cs> <Number> [<EOL> <TS>]]*

<Integer>               ::=  {(0-9)} {(1-9)(0-9)+}

<Number>                ::=  {<Integer>} {<HexNumber>}

<TRUE>                  ::=  {"TRUE"} {"true"} {"True"} {"0x1"}
                             {"0x01"} {"1"}

<FALSE>                 ::=  {"FALSE"} {"false"} {"False"} {"0x0"}
                             {"0x00"} {"0"}

<BoolType>              ::=  {<TRUE>} {<FALSE>}

<MACRO>                 ::=  (A-Z)(A-Z0-9_)*

<MACROVAL>              ::=  "$(" <MACRO> ")"

<PcdName>               ::=  <TokenSpaceGuidCName> "." <PcdCName>

<PcdCName>              ::=  <CName>

<TokenSpaceGuidCName>   ::=  <CName>

<UINT8>                 ::=  {"0x"} {"0X"} (\x0 - \xFF)

<UINT16>                ::=  {"0x"} {"0X"} (\x0 - \xFFFF)

<UINT32>                ::=  {"0x"} {"0X"} (\x0 - \xFFFFFFFF)

<UINT64>                ::=  {"0x"} {"0X"} (\x0 - \xFFFFFFFFFFFFFFFF)

<UINT8z>                ::=  {"0x"} {"0X"} <HexDigit> <HexDigit>

<UINT16z>               ::=  {"0x"} {"0X"} <HexDigit> <HexDigit>
                                   <HexDigit> <HexDigit>

<UINT32z>               ::=  {"0x"} {"0X"} <HexDigit> <HexDigit>

                             <HexDigit> <HexDigit> <HexDigit> <HexDigit>
                             <HexDigit> <HexDigit>
```

```
<UINT64z>              ::=  {"0x"} {"0X"} <HexDigit> <HexDigit>
                            <HexDigit> <HexDigit> <HexDigit> <HexDigit>
                            <HexDigit> <HexDigit> <HexDigit> <HexDigit>
                            <HexDigit> <HexDigit> <HexDigit> <HexDigit>
                            <HexDigit> <HexDigit>

<ShortNum>             ::=  (0-255)

<IntNum>               ::=  (0-65535)

<LongNum>              ::=  (0-4294967295)

<LongLongNum>          ::=  (0-18446744073709551615)

<NumValUint8>          ::=  {<ShortNum>} {<UINT8>}

<NumValUint16>         ::=  {<IntNum>} {<UINT16>}

<NumValUint32>         ::=  {<LongNum>} {<UINT32>}

<NumValUint64>         ::=  {<LongLongNum>} {<UINT64>}

<ModuleType>           ::=  {"BASE"} {"SEC"} {"PEI_CORE"} {"PEIM"}
                            {"DXE_CORE"} {"DXE_DRIVER"} {"SMM_CORE"}
                            {"DXE_RUNTIME_DRIVER"} {"DXE_SAL_DRIVER"}
                            {"DXE_SMM_DRIVER"} {"UEFI_DRIVER"}
                            {"UEFI_APPLICATION"} {"USER_DEFINED"}

<ModuleTypeList>       ::=  <ModuleType> [" " <ModuleType>]*

<IdentifierName>       ::=  <TS> {<MACROVAL>} {<PcdName>} <TS>


<Boolean>               ::=  {<BoolType>} {<Expression>}

<EOL>                   ::=  <TS> 0x0D 0x0A

<OA>                    ::=  (a-zA-Z)(a-zA-Z0-9)*

<arch>                  ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {<OA>}
```

Note: *When using the characters "\|" or "\|\|" in an expression, the expression must be encapsulated in open "(" and close ")" parenthesis.*

Note: *Comments may appear anywhere within a DEC file, provided they follow the rules that a comment may not be enclosed within Section headers, and that in line comments must appear at the end of a statement.*

Parameters

Expression

Expression syntax is defined the EDK II Expression Syntax Specification.

ExpressionVal

An expression that evaluates to a number that fits the datum types of the other elements, For example, if one of the other elements is a **UINT8**, then the expression must evaluate to a value that is byte. It is recommended that all parameters be typecast to **UINT64** values before any operation performed, then the result can be tested to ensure that the datum size of the result is correct. PCD names can be used as parameters, and the value of the PCD is used for evaluation purposes. Any result that exceeds the size of the datum type must break the build.

Note: *Circular references (self-references are tight circular references) must cause a build break.*

UnicodeString

When the **<UnicodeString>** element (these characters are string literals as defined by the C99 specification: `L"string`", not actual Unicode characters) is included in a value, the build tools may be required to expand the ASCII string between the quotation marks into a valid **UCS-2 character** format string. The build tools parser must treat all content between the field separators (excluding white space characters around the field separators) as ASCII literal content when generating the *AutoGen.c* and *AutoGen.h* files.

Comments

Strings that appear in comments may be ignored by the build tools. An ASCII string matching the format of the ASCII string defined by **<UnicodeString>** (`L"Foo"` for example,) that appears in a comment must never be expanded by any tool.

CFlags

CFlags refers to a string of valid arguments appended to the command line of any third party or provided tool. It is not limited to just a compiler executable tool. MACRO values that appear in quoted strings in CFlags content must not be expanded by parsing tools.

OA

Other Architecture - One or more user defined target architectures, such as ARM or PPC. The architectures listed here must have a corresponding entry in the EDK II meta-data file, *Conf/tools_def.txt*.

FileSep

FileSep refers to either the back slash "\" or forward slash "/" characters that are used to separate directory names. All EDK II DEC files must use the "**/**" forward slash character when specifying the directory portion of a filename. Microsoft operating systems, that normally use a back slash character for separating directory names, will interpret the forward slash character correctly.

CArray

All C data arrays used in PCD value fields must be byte arrays. The C format GUID style is a special case that is permitted in some fields that use the **<CArray>** nomenclature.

EOL

The DOS End Of Line: "0x0D 0x0A" character sequence must be used for all EDK II meta-data files. All *Nix based tools can properly process the DOS EOL characters. Microsoft based tools cannot process the *Nix style EOL characters.

3.2.2 MACROs

Use of MACRO statements is optional.

Summary

Macro statements are characterize by a **DEFINE** line. Macro statements in DEC files are only permitted to describe a path (shortcut name). If the Macro statement is within the **[Defines]** section, then the Macro is common to the entire file, with local definitions taking precedence (if the same MACRO name is used in subsequent sections, then the MACRO value is local to only that section.)

Macro statements in comments must also be ignored by parsing tools.

Macros may not be referenced before they are defined.

A previously defined macro is permitted to be used as `$(MACRO)` in the right side of a different Macro (in the value) statement.

Macro names must not use the name of the tokens defined in this file, such as `PACKAGE_GUID` is defined as a token in the **[Defines]** section of this document, and therefore cannot be used as the **<MACRO>** name in a **<MacroDefinition>** statements, **<MACRO>** variable.

If the tools encounters a macroval, as in `$(MACRO)`, that is not defined, the build tools must break.

Prototype

```
<MacroDefinition>  ::= <TS> "DEFINE" <TS> <MACRO> <Eq> [<Value>] <EOL>

<Value>            ::= {<PATH>} {<Filename>}
```

Examples

```
DEFINE GEN_SKU_DIR = MyPlatformPkg/GenPei
DEFINE SKU1_Dir = MyPlatformPkg/Sku1/Pei
DEFINE LIB = Include/Library
DEFINE PROTO_HDRS = $(LIB)/Protocol
```

Parameters

<PATH>

Any part of the path line can be replaced by a MACRO as shown in the following table.

Table 1. MACRO Usages

MACRO DEFINITION	MACRO USAGE
DEFINE MY_MACRO = test1	$(MY_MACRO)/test2/test3.inf
DEFINE MY_MACRO = test1/	$(MY_MACRO)test2/test3.inf
DEFINE MY_MACRO = test3.inf	test1/test2/$(MY_MACRO)
DEFINE MY_MACRO = test3	test1/test2/$(MY_MACRO).inf
DEFINE MY_MACRO = test1/test2/test3.inf	$(MY_MACRO)

3.2.3 Conditional Statements

The conditional statements are not permitted anywhere within the DEC file.

3.2.4 !include Statement

The !include statement is not permitted in an EDK II DEC file.

3.2.5 Special Comment Blocks

This section defines special format comment blocks that contain information about this package. These command blocks are not required.

The UEFI Distribution Package Specification states the for a given PCD Token Space, Error numbers must be unique. Since there may be multiple PCD that may have identical error types, an error number with an associated error message may be shared. The syntax described here can be used by tools to map these error numbers by a token space GUID C name. These comment blocks must appear before content might be used (in later sections of the DEC file).

Prototype

```
<ErrNoBlock> ::= <TS> "#" <EOL>
                 <TS> "#" <MTS> "[Error." <TSpaceGuidCName> "]" <EOL>
                 [<TS> "#" <MTS> <UINT32> <FS> <AsciiString> <EOL>]+
                 <TS> "#" <EOL>+
```

Example

```
#
# [Error.gEfiIntelFrameworkModulePkgTokenSpaceGuid]
#   0x80000001 | Invalid value provided.
#   0x80000002 | Reserved bits must be set to zero.
#
```

3.3 Header Comment Section

This is an optional section for EDK II packages that will not be distributed using tools that create UEFI Distribution Packages. It is required for EDK II packages that will be distributed using tools that create UEFI Distribution Packages.

Summary

The Copyright and License notices for the DEC file are in the comments that start the file. The format for the comment section is:

```
## @file
# Abstract
#
# Description
#
# Copyright
#
# License
#
##
```

This information can be derived from an XML Distribution package file (UEFI Packaging Specification) or from a developer creating a new package declaration (DEC) document.

Prototype

```
<Header>       ::=  <SourceHeader>
                    [<BinaryHeader>]

<SourceHeader> ::=  <Comment>*
                    "##" [<Space>] <Space> "@file" [<TS> <File>] <EOL>
                    [<Abstract>]
                    [<Description>]
                    <Copyright>+
                    "#" <EOL>
                    <License>
                    "##" <EOL>

<Filename>     ::=  <Word> "." <Extension>

<Abstract>     ::=  "#" <AsciiString> <EOL>
                    ["#" <EOL>]

<Description>  ::=  ["#" <AsciiString> <EOL>]+
                    ["#" <EOL>]

<Copyright>    ::=  "#" <TabSpace> <CopyName> <Date> "," <CompInfo>

<CopyName>     ::=  ["Portions" <MTS>] "Copyright (c)" <MTS>

<Date>         ::=  <Year> [<TS> {<DateList>} {<DateRange>}]

<Year>         ::=  "2" (0-9)(0-9)(0-9)

<DateList>     ::=  <CommaSpace> <Year> [<CommaSpace> <Year>]*

<DateRange>    ::=  "-" <TS> <Year>

<CompInfo>     ::=  (0x20 - 0x7e)* <MTS> <Arr>

<Arr>          ::=  "All rights reserved." [<TS> "<BR>"] <EOL>

<License>      ::=  ["#" <TS> <AsciiString> <EOL>]+
                    ["#" <EOL>]

<BinaryHeader> ::=  "##" <Space> <Space> "@BinaryHeader" <EOL>
                    <BinaryAbstract>
                    "#" <EOL>
                    <BinDescription>
                    "#" <EOL>
                    <Copyright>+
                    "#" <EOL>
                    <BinaryLicense>
                    "#" <EOL>
                    "##" <EOL>
```

```
<Filename>       ::=  <Word> "." <Extension>

<BinaryAbstract> ::=  "#" <TS> <AsciiString> <EOL>

<BinDescription> ::=  ["#" <TS> <AsciiString> <EOL>]+

<Copyright>      ::=  "#" <MTS> <CopyName> <Date> "," <CompInfo>

<CopyName>       ::=  ["Portions" <MTS>] "Copyright (c)" <MTS>

<Date>           ::=  <Year> [<TS> {<DateList>} {<DateRange>}]

<Year>           ::=  "2" (0-9)(0-9)(0-9)

<DateList>       ::=  <CommaSpace> <Year> [<CommaSpace> <Year>]*

<DateRange>      ::=  "-" <TS> <Year>

<CompInfo>       ::=  (0x20 - 0x7e)* <MTS> <Arr>

<Arr>            ::=  "All rights reserved." [<TS> "<BR>"] <EOL>

<BinaryLicense>  ::=  ["#" <TS> <AsciiString> <EOL>]+
```

Parameters

Abstract

A brief one line description of what the package provides.

BinaryAbstract

A brief one line description of what the packages provides that may be different from a source abstract.

Description

A detailed description of what the package provides.

BinaryDescription

A detailed description of what the package provides that may be different from a source description.

Copyright

The copyright date should be modified if there is a functional change to the source code. Since binaries are constructed from source, the binary file uses the same copyright date as the source DEC, however tools are not required to ensure that the dates are identical.

License

One or more licenses that the package with source code is released under.

BinaryLicense

One or more licenses that the binary package is released under that may be different from the licenses used for distributing the package with source code.

Example

```
## @file
# Framework Module Development Environment Industry Standards
#
# This Package provides headers and libraries that conform to
# EFI/Framework Industry standards.
#
# Copyright (c) 2006 - 2007, Intel Corporation.
#
#  All rights reserved.
#  This program and the accompanying materials are licensed and made
#  available under the terms and conditions of the BSD License which
#  accompanies this distribution.
#  The full text of the license may be found at
#  http://opensource.org/licenses/bsd-license.php
#
#  THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS,
#  WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR
#  IMPLIED.
#
##
```

3.4 [Defines] Section

This section is required.

Summary

This describes the **[Defines]** section, which is required in all DEC files. This file is created during installation of a UEFI distribution package or by the developer and is an input to the EDK II build tool parsing utilities. Elements may appear in any order within this section.

The code for this specification is **"00010019**" and new versions of this specification must increment the minor (0019) portion of the specification code. This value may also be specified as a decimal value, 1.25.

Existing DEC files are not required to update the **DEC_SPECIFICATION** version value. This value may be used by tools to identify any new functionality introduced by this specification version.

Tools are allowed to use the **PACKAGE_GUID** and **PACKAGE_VERSION** values for testing dependencies. This is particularly import for creating UEFI Distribution Packages, as they are required fields in the distribution package description file.

Note that comments may be appended to define statements as well as located anywhere within the **[Defines]** section.

Prototype

```
<defines>      ::=  "[Defines]" <EOL>
                    <TS> "DEC_SPECIFICATION" <Eq> <SpecVer> <EOL>
                    <TS> "PACKAGE_NAME" <Eq> <UiNameType> <EOL>
                    <TS> "PACKAGE_GUID" <Eq> <RegistryFormatGUID> <EOL>
                    <TS> "PACKAGE_VERSION" <Eq> <DecimalVersion> <EOL>
                    [<TS> "PACKAGE_UNI_FILE" <Eq> <Filename> <EOL>]
                    <MacroDefinition>*

<UiNameType>   ::=  <Word>

<SpecVer>      ::=  {<HexVersion>} {(0-9))+ "." (0-9)+}
```

Parameters

UiNameType

The **PACKAGE_NAME** value may be used for creating directories.

DecimalVersion

This is a decimal number, and if not specified is assumed to be 0. Alpha characters are not permitted.

SpecVer

For new DEC files, the version value must be set to 0x00010019. Tools that process this version of the DEC file can successfully process earlier versions of the DEC file (this is a backward compatible update). There is no requirement to change the value in existing DEC files if no other content changes. This may also be specified as decimal value, 1.25.

Filename

Filenames listed in the **[Defines]** section must be relative to the directory the DEC file is in. Use of "..", "." and "../" in the directory path is not permitted. Use of an absolute path is not permitted. The file name specified in the **PACKAGE_UNI_FILE** entry must be a Unicode file with an extension of .uni, .UNI or .Uni.

Example

```
[DEFINES]
  DEC_SPECIFICATION  =  0x00010019
  PACKAGE_NAME       =  MdePkg
  PACKAGE_GUID       =  5e0e9358-46b6-4ae2-8218-4ab8b9bbdcec
  PACKAGE_VERSION    =  0.3
  PACKAGE_UNI_FILE   = MdePkg.uni
```

3.5 [Includes] Sections

These sections are optional.

Summary

Defines the **[Includes]** section tag in the DEC files. This section lists the "standard"

include locations, not file names, provided in the package. Each include **<PATH>** entry listed in a section must be unique to the section (duplicate entries within a section are not permitted).

A path entry listed in an architectural section must not be listed in the common section.

The included path must not end with the **<FileSep>** character.

All paths must be relative to the directory that contains the DEC file. Use of absolute paths, or **WORKSPACE** relative paths is prohibited.

It is permissible to use a **<MACROVAL>** entry in this section provided the above rules are followed.

The '**common**' architecture modifier in a section tag must not be combined with other architecture type modifiers; doing so will result in a build break.

Prototype

```
<Include>        ::=  "[Includes" [<com_attribs>] "]" <EOL>
                      <IncEntries>*

<com_attribs>    ::=  {".common"} {<attribs>}

<attribs>        ::=  <attrs> ["," <TS> "Includes" <attrs>]*

<attrs>          ::=  "." <arch>

<IncEntries>     ::=  {<MacroDefinition>} {<HdrFile>}

<HdrFile>        ::=  <CommentBlock>*
                      <TS> <PATH>
                      {<CommentBlock>} {<EOL>}

<CommentBlock>   ::=  <TS> "##" <TS> <ModuleTypeList> <CmtOrEol>

<CmtOrEol>       ::=  {<Comment>} {<EOL>}
```

Parameters

PATH

Path statements listed in this section must be relative to the directory that contains the DEC file. Use of "..", "../" or "./" in the directory path is prohibited.

Restrictions

It is NOT permissible to list an include directory under common and under a specific architecture. It is permissible to specify include directory entries under all architectures except "**common**" if different include directories are required for different architectures.

Example

```
##############################################################################
#
# Include section - list of Include Paths relative to the DEC file that
#                   are provided by this package.
#                   Comments are used for Keywords and Module Types.
#
##############################################################################

[Includes.common]
  Include              # Includes for all processor architectures

[Includes.IA32]
  Include/Ia32         # Includes specific to IA32

[Includes.X64]
  Include/X64          # Includes specific to X64

[Includes.IPF]
  Include/Ipf          # Includes specific to IA64

[Includes.EBC]
  Include/Ebc          # Includes specific to EBC

[Includes.ARM]
  Include/Arm          # Includes specific to ARM
```

3.6 [Guids] Sections

These sections are optional.

Summary

Defines the **[Guids]** section tag of the DEC files.

GUID entries listed in architectural sections are not permitted to be listed in the common architectural section.

The '**common**' architecture modifier in a section tag must not be combined with other architecture type modifiers; doing so will result in a build break.

The use of a **<MACROVAL>** element in this section is prohibited.

Each GUID entry must be listed only once per section.

Prototype

```
<Guids>            ::=  "[Guids" [<com_attribs>] "]" <EOL>
                        <GuidEntries>*

<com_attribs>      ::=  {".common"} {<attribs>}

<attribs>          ::=  <attrs> ["," <TS> "Guids" <attrs>]*

<attrs>            ::=  "." <arch>

<GuidEntries>      ::=  [<GuidComment>]
                        <TS> <CName> <Eq> <CFormatGUID>
                        {<CommentBlock>} {<EOL>}

<GuidComment>      ::=  [<Description>]
                        <TS> "##" <TS> <GuidHeaderFile> <EOL>

<GuidValue>        ::=  <CFormatGUID>

<GuidHeaderFile>   ::=  <PATH> <Word> ".h"

<Description>      ::=  <TS> "##" <TS> <AsciiString> <EOL>
                        [<TS> "#" <TS> <AsciiString> <EOL>]*

<CommentBlock>     ::=  <TS> "##" <TS> <ModuleTypeList>
                        {<Comment>} {<EOL>}
```

Parameters

GuidHeaderFile

Path to the GUID header file statement listed in comments in this section must be relative to the directory that contains the DEC file. Use of "..", "../" or "./" in the directory path is prohibited.

Restrictions

It is not permissible to list a GUID entry under common and under a specific architecture. It is permissible to specify GUID entries under all architectures except "common" if different GUID values may be required for different architectures.

Example

```
##################################################################################
#
# Global Guid Definition section - list of Global Guid C Name
#                                  Data Structures that are provided by
#                                  this package.
#
##################################################################################

[Guids.common]
  gPcdHobGuid                     = { 0x582E7CA1, 0x68CD, 0x4D44, \
    { 0xB4, 0x3B, 0xF2, 0x98, 0xED, 0x58, 0x7B, 0xA6 }}
  gEfiWinNtPassThroughGuid        = { 0xCC664EB8, 0x3C24, 0x4086, \
    { 0xB6, 0xF6, 0x34, 0xE8, 0x56, 0xBC, 0xE3, 0x6E }}
  gEfiWinNtCPUSpeedGuid           = { 0xD4F29055, 0xE1FB, 0x11D4, \
    { 0xBD, 0x0D, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiWinNtCPUModelGuid           = { 0xBEE9B6CE, 0x2F8A, 0x11D4, \
    { 0xBD, 0x0D, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiWinNtMemoryGuid             = { 0x99042912, 0x122A, 0x11D4, \
    { 0xBD, 0x0D, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiWinNtConsoleGuid            = { 0xBA73672C, 0xA5D3, 0x11D4, \
    { 0xBD, 0x00, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiWinNtUgaGuid                = { 0xAB248E99, 0xABE1, 0x11D4, \
    { 0xBD, 0x0D, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiWinNtGopGuid                = { 0x4e11e955, 0xccca, 0x11d4, \
    { 0xbd, 0x0d, 0x00, 0x80, 0xc7, 0x3c, 0x88, 0x81 }}
  gEfiWinNtSerialPortGuid         = { 0x0C95A93D, 0xA006, 0x11D4, \
    { 0xBC, 0xFA, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiWinNtFileSystemGuid         = { 0x0C95A935, 0xA006, 0x11D4, \
    { 0xBC, 0xFA, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiWinNtPhysicalDisksGuid      = { 0x0C95A92F, 0xA006, 0x11D4, \
    { 0xBC, 0xFA, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiWinNtVirtualDisksGuid       = { 0x0C95A928, 0xA006, 0x11D4, \
    { 0xBC, 0xFA, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiEdkNt32PkgTokenSpaceGuid    = { 0x0D79A645, 0x1D91, 0x40a6, \
    { 0xA8, 0x1F, 0x61, 0xE6, 0x98, 0x2B, 0x32, 0xB4 }}
```

3.7 [Protocols] Sections

These sections are optional.

Summary

Defines the **`[Protocols]`** section tag. This is a list of the global PROTOCOL C Names and their C Guid values that are declared in the EDK II package (.DEC) file.

Protocol entries listed in architectural sections are not permitted to be listed in the

common architectural section.

The '**common**' architecture modifier in a section tag must not be combined with other architecture type modifiers; doing so will result in a build break.

The use of a **<MACROVAL>** element in this section is prohibited.

Each Protocol entry must be listed only once per section.

Prototype

```
<Protocols>          ::=  "[Protocols" [<com_attribs>] "]" <EOL>
                          <ProtocolEntries>*

<com_attribs>        ::=  {".common"} {<attribs>}

<attribs>            ::=  <attrs> ["," <TS> "Protocols" <attrs>]8

<attrs>              ::=  "." <arch>

<ProtocolEntries>    ::=  [<ProtocolComment>]
                          <TS> <CName> <Eq> <CFormatGUID>
                          {<CommentBlock>} {<EOL>}

<ProtocolComment>    ::=  [<Description>]
                          <ProtoHdrFile>
<ProtoHdrFile>       ::=  <TS> "##" <TS> <PATH> <Word> ".h"

<Description>        ::=  <TS> "##" <TS> <AsciiString> <EOL>
                          [<TS> "#" <TS> <AsciiString> <EOL>]*

<CommentBlock>       ::=  [<TS> "##" <TS> <ModuleTypeList>]
                          {<Comment>} {<EOL>}
```

Parameters

ProtocolHeaderFile

Path to the Protocol header file statement listed in comments in this section must be relative to the directory that contains the DEC file. Use of "..", "../" or "./" in the directory path is prohibited.

Restrictions

It is NOT permissible to list a Protocol entry under common and under a specific architecture. It is permissible to specify Protocol entries under all architectures except "**common**" if different Guid values may be required for different architectures.

Example

```
##################################################################################
#
# Global Protocols Definition section - list of Global Protocols C Name
#                                       Data Structures that are provided by
#                                       this package.
#
##################################################################################

[Protocols.common]
  gEfiWinNtThunkProtocolGuid     = { 0x58C518B1, 0x76F3, 0x11D4, \
    { 0xBC, 0xEA, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
  gEfiWinNtIoProtocolGuid        = { 0x96EB4AD6, 0xA32A, 0x11D4, \
    { 0xBC, 0xFD, 0x00, 0x80, 0xC7, 0x3C, 0x88, 0x81 }}
```

3.8 [PPIs] Sections

These sections are optional.

Summary

Defines the optional **[Ppis]** section tag. This is a list of the global PPI C Names that are referenced in the EDK II package's module C code.

PPI entries listed in architectural sections are not permitted to be listed in the common architectural section.

The '**common**' architecture modifier in a section tag must not be combined with other architecture type modifiers; doing so will result in a build break.

The use of a **<MACROVAL>** element in this section is prohibited.

Each PPI entry must be listed only once per section.

Prototype

```
<Ppis>           ::=  "[Ppis" [<com_attribs>] "]" <EOL>
                      <PpiEntries>*

<com_attribs>    ::=  {".common"} {<attribs>}

<attribs>        ::=  <attrs> ["," <TS> "Ppis" <attrs>]*

<attrs>          ::=  "." <arch>

<PpiEntries>     ::=  [<PpiComment>]
                      <TS> <CName> <Eq> <CFormatGUID>
                      {<CommentBlock>} {<EOL>}

<PpiComment>     ::=  [<Description>]
                      <TS> "##" <TS> <PpiHeaderFile> <EOL>

<Description>    ::=  <TS> "##" <TS> <AsciiString> <EOL>
                      [<TS> "#" <TS> <AsciiString> <EOL>]*

<PpiHeaderFile>  ::=  <PATH> <Word> ".h"

<CommentBlock>   ::=  [<TS> "##" <TS> <ModuleTypeList>]
                      {<Comment>} {<EOL>}
```

Parameters

PpiHeaderFile

Path to the PPI header file statement listed in comments in this section must be relative to the directory that contains the DEC file. Use of "..", "../" or "./" in the directory path is prohibited.

Restrictions

It is NOT permissible to list a PPI entry under common and under a specific architecture. It is permissible to specify PPI entries under all architectures except "common" if different Guid values may be required for different architectures.

Example

```
##############################################################################
#
# Global Ppis Definition section - list of Global Ppis C Name
#                                  Data Structures that are provided by
#                                  this package.
#
##############################################################################

[Ppis.common]
  gPeiNtThunkPpiGuid              = { 0x98C281E5, 0xF906, 0x43DD, \
    { 0xA9, 0x2B, 0xB0, 0x03, 0xBF, 0x27, 0x65, 0xDA }}
  gNtPeiLoadFilePpiGuid           = { 0xFD0C65EB, 0x0405, 0x4CD2, \
    { 0x8A, 0xEE, 0xF4, 0x00, 0xEF, 0x13, 0xBA, 0xC2 }}
  gNtFwhPpiGuid                   = { 0x4E76928F, 0x50AD, 0x4334, \
    { 0xB0, 0x6B, 0xA8, 0x42, 0x13, 0x10, 0x8A, 0x57 }}
  gPeiNtAutoScanPpiGuid           = { 0x0DCE384D, 0x007C, 0x4BA5, \
    { 0x94, 0xBD, 0x0F, 0x6E, 0xB6, 0x4D, 0x2A, 0xA9 }}
```

3.9 [LibraryClasses] Sections

These sections are optional.

Summary

Defines the **[LibraryClasses]** tag in the DEC files.

The **[LibraryClasses]** section maps the location, relative to the DEC file, of library ClassName to the header file for the library class specified by the ClassName.

ClassName entries listed in architectural sections are not permitted to be listed in the common architectural section.

The '**common**' architecture modifier in a section tag must not be combined with other architecture type modifiers; doing so will result in a build break.

All paths must be relative to the directory that contains the DEC file. Use of absolute paths, or **WORKSPACE** relative paths is prohibited.

It is permissible to use a **<MACROVAL>** entry in this section provided the above rules are followed.

Each ClassName entry must be listed only once per section.

Prototype

```
<LibraryClasses>  ::=  "[LibraryClasses" [<com_attribs>] "]" <EOL>
                       <LcEntries>*

<com_attribs>     ::=  {".common"} {<attribs>}

<attribs>         ::=  <attrs> ["," <TS> "LibraryClasses" <attrs>]*

<attrs>           ::=  "." <arch>

<LcEntries>       ::=  {<LcEntry>} {<MacroDefinition>}

<LcEntry>         ::=  [<LcDoxygenHelp>]
                       <TS> <ClassName> <FS> <Filename>
                       {<CommentBlock>} {<EOL>}

<LcDoxygenHelp>   ::=  <TS> "##" <TS> "@LibraryClasses" <TS>
                       <AsciiString> <EOL>

<ClassName>       ::=  <ToolWord>
                         # A User Defined Keyword consisting of
                         # alphanumeric characters. No special
                         # characters are permitted.

<Filename>        ::=  <PATH> <File> ".h"

<CommentBlock>    ::=  <TS> "##" <TS> <ModuleTypeList>
                       {<Comment>} {<EOL>}
```

Parameters

Filename

Path portion of the header file statements in this section must be relative to the directory that contains the DEC file. Use of "..", "../" or "./" in the directory path is prohibited.

Restrictions

It is NOT permissible to list a Library Class entry under common and under a specific architecture. It is permissible to specify Library Class entries under all architectures except `"common"` if different header filename values are required for different architectures.

Example

```
##################################################################################
#
# Library Class Header section - list of Library Class header
#                                files that are provided by
#                                this package.
#
##################################################################################

[LibraryClasses.common]
  UefiRuntimeServicesTableLib | \
    Include/Library/UefiRuntimeServicesTableLib.h
  UefiLib | Include/Library/UefiLib.h
  UefiDriverModelLib | Include/Library/UefiDriverModelLib.h
  UefiDriverEntryPoint | Include/Library/UefiDriverEntryPoint.h
  UefiDecompressLib | Include/Library/UefiDecompressLib.h
  UefiBootServicesTableLib | Include/Library/UefiBootServicesTableLib.h
  TimerLib|Include/Library/TimerLib.h
  SmbusLib|Include/Library/SmbusLib.h
  ResourcePublicationLib|Include/Library/ResourcePublicationLib.h
  PostCodeLib|Include/Library/PostCodeLib.h
  ReportStatusCodeLib|Include/Library/ReportStatusCodeLib.h
  PrintLib|Include/Library/PrintLib.h
  PerformanceLib|Include/Library/PerformanceLib.h
  PeiServicesTablePointerLib| \
    Include/Library/PeiServicesTablePointerLib.h
  PeimEntryPoint|Include/Library/PeimEntryPoint.h
  PeiServicesLib|Include/Library/PeiServicesLib.h
  PeiCoreEntryPoint|Include/Library/PeiCoreEntryPoint.h
  PeCoffLib|Include/Library/PeCoffLib.h

  BaseLib|Include/Library/BaseLib.h
[LibraryClasses.IA32]
  UefiApplicationEntryPoint | \
    Include/Library/UefiApplicationEntryPoint.h # UEFI_APPLICATION
[LibraryClasses.X64]
  UefiApplicationEntryPoint| \
    Include/Library/UefiApplicationEntryPoint.h # UEFI_APPLICATION
[LibraryClasses.IPF]
  UefiApplicationEntryPoint| \
    Include/Library/UefiApplicationEntryPoint.h # UEFI_APPLICATION
[LibraryClasses.EBC]
  UefiApplicationEntryPoint| \
    Include/Library/UefiApplicationEntryPoint.h # UEFI_APPLICATION
```

3.10 PCD Sections

These are optional sections. However, if modules in this package's directory tree use a PCD that is not declared in other DEC files, then the package creator must list the PCD in this file.

Summary

Defines the PCDs section tags in the DEC files.

There are five defined PCD access method types. Do not confuse these access types with the data types of the PCDs. The five access method types are: **PcdsFeatureFlag**, **PcdsFixedAtBuild**, **PcdsPatchableInModule**, **PcdsDynamic** and **PcdsDynamicEx**.

The PCD is used to define potential values that a module might be coded against, and if a module uses the access methods for **PcdsDynamic**, then the platform can define the final usage. PCDs listed as **PcdsDynamic** or **PcdsDynamicEx** will only have one value in the final binary image - the PEI and DXE PCD Drivers that maintain these values use a single database for all architectures, with a unique value for a PCD (identified by the Token Space GUID value and the Token Number).

If a PCD is only listed under a **PcdsFixedAtBuild** or **PcdsPatchableInModule**, then modules are restricted, and cannot be coded to use either of the dynamic PCD access methods.

Using the "**common**" architectural modifier is exactly the same as specifying a PCD section type without an architectural modifier. A PCD listed in an INF or DSC file which uses an architectural modifier may be listed in the '**common**' section in the DEC file; it is not required to list PCDs in architectural sections in the DEC. If a PCD is only listed under a section with an architectural modifier that is not 'common', then it may only be used by modules build for that specific architecture.

Each PCD entry must be listed only once per section. If a PCD is listed more than once within one section, the last entry takes precedence.

The PCD values in this file are the default values. Default values listed in architectural sections override default values specified in a 'common' architectural section. The EDK II build system allows these values may be overridden by default values in the INF files (provided all INF files use the same default value for PCDs that are used by multiple modules) and by values specified in the DSC or FDF file. A PCD's Datum type cannot be changed for different access methods, only one datum type is permitted.

PCD entries may be put into any or all PCD section types except **PcdsFeatureFlag** sections.

The use of a **<MACROVAL>** element in this section is prohibited.

The Token number specified in a PCD entry is used in code (auto-generated files by the EDK II build system), along with the Token Space GUID value to uniquely identify a given PCD. The Token number must be identical for every entry of a PCD in this file (using a different token number for a PCD listed in **PcdsDynamic** and **PcdsDynamicEx** sections for example, will result in a build break).

PCDs listed in **PcdsFeatureFlag** sections must only be listed in **PcdsFeatureFlag** sections.

Prototype

```
<PCDs>           ::=  <PcdSections>*

<PcdSections>    ::=  {<PcdFeature>} {<PcdOther>}

<PcdFeature>     ::=  <FFSectionTag> <FFEntries>*

<FFEntries>      ::=  ["##" <TS> <PcdDescription>]
                      [<TS> <Prompt>]
                      [{<List>} {<Express>}]
                      <TS> <PcdBool>

<FFSectionTag>   ::= "[" "PcdsFeatureFlag" [<com_FFattribs>} "]" <EOL>

<com_FFattribs> ::=  {".common"} {<FFattribs>}

<FFattribs>      ::=  <attrs> ["," <TS> "PcdsFeatureFlag" <attrs>]*

<PcdOther>       ::=  "[" <PcdType> [<com_or_attribs>] "]" <EOL>
                      <PcdEntries>*

<PcdType>        ::=  {"PcdsPatchableInModule"} {"PcdsFixedAtBuild"}
                      {"PcdsDynamic"} {"PcdsDynamicEx"}

<com_or_attribs> ::=  {<com_attribs>} {<attribs>}

<com_attribs>    ::=  [".common"] ["," <TS> <PcdType> [".common"]]*

<attribs>        ::=  <attrs> ["," <TS> <PcdType> <attrs>]*

<attrs>          ::=  "." <arch>

<PcdEntries>     ::=  ["##" <TS> <PcdDescription>]
                      [<TS> <Prompt>]
                      [<DoxComment>]
                      <PcdEntry>

<PcdEntry>       ::=  <TS> {<PcdBool>} {<PcdNumEntry>} {<PcdPtr>}

<PcdNumEntry>    ::=  {<Pcd8>} {<Pcd16>} {<Pcd32>} {<Pcd64>}

<PcdBool>        ::=  <PcdName> <FS> <BoolPcd> <FS> <Token> <CbOrEol>

<BoolPcd>        ::=  <BoolType> <FS> "BOOLEAN"

<CbOrEol>        ::=  {<CommentBlock>} {<EOL>}

<Pcd8>           ::=  <PcdName> <FS> <PcdUint8> <FS> <Token> <CbOrEol>

<PcdUint8>       ::=  <NumValUint8> <FS> "UINT8"
```

```
<Pcd16>          ::=  <PcdName> <FS> <PcdUint16> <FS> <Token> <CbOrEol>

<PcdUint16>      ::=  <NumValUint16> <FS> "UINT16"

<Pcd32>          ::=  <PcdName> <FS> <PcdUint32> <FS> <Token> <CbOrEol>

<PcdUint32>      ::=  <NumValUint32> <FS> "UINT32"

<Pcd64>            ::=  <PcdName> <FS> <PcdUint64> <FS> <Token> <CbOrEol>

<PcdUint64>        ::=  <NumValUint64> <FS> "UINT64"

<PcdPtr>           ::=  <PcdName> <FS> <PcdPtrVal> <FS> <Token> <CbOrEol>

<PcdPtrVal>        ::=  <PtrVal> <FS> "VOID*"

<PtrVal>           ::=  {<CString>} {<CArray>}

<Token>            ::=  <NumValUint32>

<DoxComment>       ::=  <TS> {<Range>+} {<List>} {<Express>+}

<Prompt>           ::=  "#" <TS> "@Prompt <MTS> <AsciiString> <EOL>

<Range>            ::=  "#" <TS> "@ValidRange" <TS> <ERangeValues> <EOL>

<ERangeValues>     ::=  [<ErrorCode> <TS>] <RangeValues>

<List>           ::=  "#" <TS> "@ValidList" <TS> <EValidValueList> <EOL>

<EValidValueList> ::=  [<ErrorCode> <TS>] <ValidValueList>

<Express>          ::=  "#" <TS> "@Expression" <TS> <EExpression> <EOL>

<EExpression>      ::=  [<ErrorCode> <TS>] <Expression>

<ErrorCode>        ::=  <NumValUint32> <FS>

<PcdDescription>   ::=  <AsciiString> <EOL>
                        [<TS> "#" <AsciiString> <EOL>]*

<CommentBlock>     ::=  <TS> "##" <TS> <ModuleTypeList>
                        {<Comment>} {<EOL>}

<RangeValues>      ::=  {<ValidRange>} {<RangeExpress>}

<RngOp>            ::=  <TS> {"AND"} {"and"} {"OR"} {"or"} <TS>

<RangeExpress>     ::=  {"(" <ValidRng> ")" <RngOp> "(" <ValidRng> ")"}
                        {"NOT" <TS> "(" <ValidRng> ")"}
```

```
<ValidRng>          ::=  ["NOT" <TS>] "(" <ValidRangeIn> ")"

<ValidRangeIn>      ::=  ["NOT" <TS>] {<Integer> <TS> "-" <TS> <Integer>}
                         {<HexValue> <TS> "-" <TS> <HexValue>}
                         {"LT" <TS> {<Integer>} {<HexValue>}}
                         {"GT" <TS> {<Integer>} {<HexValue>}}
                         {"LE" <TS> {<Integer>} {<HexValue>}}
                         {"GE" <TS> {<Integer>} {<HexValue>}}
                         {"XOR" <TS> {<Integer>} {<HexValue>}}
                         {"EQ" <TS> {<Integer>} {<HexValue>}}

<ValidValueList>     ::=  <Number> ["," <TS> <Number>]*
```

Parameters

Expression

The expression in the **@Expression** entry must evaluate to True in order for the value to be valid.

RangeExpressions

This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (LT, LE, GT, GE.) The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted. Parentheses are recommended for clarity. Since there may be different error numbers associated with valid ranges (one for less than, another for greater than) more than one ValidRange entry is permitted.

HexDigit

This value, if specified corresponds to the ErrorNumber of an Error Message defined in a UEFI PI Distribution Package. Each error number is related to a single error message (translations of message text are not considered as separate error messages), with the text of error messages included in the Unicode file specified in the **PACKAGE_UNI_FILE** element in the **[Defines]** section.

TokenNumber

Each PCD declared within a token space (defined by the Token Space GUID) must be assigned a unique 32-bit value. This token number and an optional token space GUID are used in code for accessing a PCD's value.

ErrorCode

This value, if specified corresponds to the ErrorNumber of an Error Message defined in a UEFI PI Distribution Package. Each error number is related to a single error message.

Token

Each PCD declared within a token space (defined by the Token Space GUID) must be assigned a unique 32-bit value. This token number and an optional token space GUID are used in code for accessing a PCD’s value.

Restrictions

It is permissible to list a PCD entry under common and under a specific architecture.

It is permissible to specify PCD entries under all architectures except "**common**" if different default values may be required for different architectures. The tools must use

architectural specific recommended values over recommended values listed in the common sections. This technique is permitted to allow unique recommendations for individual architectures.

It is not permissible to list command modifier and an architectural modifier in a section header. The following example is not permitted:

```
[PcdsPatchableInModule.IA32, PcdPatchableInModule.common]
```

It is permissible to specify multiple architectures for like PcdType items in the same section header. For example:

```
[PcdsFeatureFlag.IA32, PcdsFeatureFlag.X64]
```

It is permissible to mix PcdsPatchableInModule, PcdsFixed, PcdsDynamic and PcdsDynamicEx PcdType elements within an architecture section. For example:

```
[PcdsPatchableInModule.IA32, PcdsFixedAtBuild.IA32]
```

The **PcdsFeatureFlag** PcdType may not be mixed with any other PcdType elements in the section header. The following example is NOT VALID:

```
[PcdsFeatureFlag.IA32, PcdsFixedAtBuild.IA32]
```

While allowed by this specification, it is not recommended to mix different PcdType.architecture values in a single section. The following example is valid, but not recommended:

```
[PcdsDynamicEx.IA32, PcdsFixedAtBuild.X64, PcdPatchableInModule.IPF]
```

Refer to the *PI Specification* for more information.

Example

```
##############################################################################
#
#
# PCD Declarations section - list of all PCDs Declared by this package
#                            Only this package should be providing the
#                            declaration, other packages should not.
#
##############################################################################
#

[PcdsFeatureFlag.common]
  ## If TRUE, the component name protocol will not be installed.
  gEfiMdePkgTokenSpaceGuid.PcdComponentNameDisable | FALSE | \
    BOOLEAN | 0x0000000d

  ## If TRUE, the driver diagnostics protocol will not be installed.
  gEfiMdePkgTokenSpaceGuid.PcdDriverDiagnosticsDisable | FALSE | \
    BOOLEAN | 0x0000000e

[PcdsFixedAtBuild.common]
  ## Indicates the maximum length of unicode string
  gEfiMdePkgTokenSpaceGuid.PcdMaximumUnicodeStringLength | \
    1000000 | UINT32 | 0x00000001

  ## Indicates the maximum length of ascii string
  gEfiMdePkgTokenSpaceGuid.PcdMaximumAsciiStringLength | 1000000 | \
    UINT32 | 0x00000002

  ## Indicates the maximum node number of linked list
  gEfiMdePkgTokenSpaceGuid.PcdMaximumLinkedListLength | 1000000 | \
    UINT32 | 0x00000003

[PcdsFixedAtBuild.common, PcdsPatchableInModule.common]
  ## This flag is used to control the printout of DebugLib
  gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel | 0x80000000 | \
    UINT32 | 0x00000006

  ## Indicates the allowable maximum number in extract handler table
  gEfiMdePkgTokenSpaceGuid.PcdMaximumGuidedExtractHandler | 0x10 | \
    UINT32 | 0x00000025

[PcdsFixedAtBuild.IPF]
  ## This flag is used to control the printout of DebugLib
  gEfiMdePkgTokenSpaceGuid.PcdIoBlockBaseAddressForIpf | \
```

```
    0x0ffffc00000 | UINT64 | 0x0000000c

[PcdsFixedAtBuild, PcdsPatchableInModule, PcdsDynamic, PcdsDynamicEx]
  ## This value is used to set the base address of pci express hierarchy
  gEfiMdePkgTokenSpaceGuid.PcdPciExpressBaseAddress | 0xE0000000 | \
    UINT64 | 0x0000000a

  ## Default current ISO 639-2 language: English & French
  gEfiMdePkgTokenSpaceGuid.PcdUefiVariableDefaultLangCodes | \
    "engfraengfra" | VOID* | 0x0000001c

  ## Default current ISO 639-2 language: English
  gEfiMdePkgTokenSpaceGuid.PcdUefiVariableDefaultLang | "eng" | \
    VOID* | 0x0000001d
```

Note: *In the above example, the backslash character is used to show a line continuation for readability. Use of a backslash character in the actual DEC file is not permitted.*

3.11 [UserExtensions] Sections

These sections are optional.

Summary

Defines the optional EDK II DEC file **[UserExtensions]** tag. The build tools must have an a priori knowledge of how to process any items in this section. EDK II build tools ignore this section.

Each **[UserExtensions]** section must have a unique set of **UserId** and **IdString** values. This means that the same **UserId** can be used in more than one section, provided the **IdString** values are different. The same **IdString** values can be used if the **UserId** values are different. The same **IdString** and **UserId** values can be used if specifying architecture specific content.

The use of a **<MACROVAL>** element in this section is prohibited.

The **"common**" architecture modifier in a section tag must not be combined with other architecture type modifiers; doing so may result in a build break.

Prototype

```
<UserExtensions>  ::=  "[UserExtensions" <com_attribs> "]" <EOL>
                       <statements>*

<com_attribs>     ::=  {<UserId> <IdString> [".common"]} {<atttribs>}

<attribs>         ::=  <attrs> ["," <TS> ""UserExtensions" <attrs>]*

<attrs>           ::=  <UserId> <IdString> ["." <arch>]

<UserId>          ::=  "." (a-zA-Z)(a-zA-Z0-9_.)*

<IdString>        ::=  "." {<NormalizedString>} {<SimpleWord>}

<statements>      ::= Content is build tool chain specific.
```

Parameters

UserId

Words that contain period "." must be encapsulated in double quotation marks.

IdString

Normalized strings that contain period "." or space characters must be encapsulated in double quotation marks. It is recommended that the IdString start with a letter.

Example

```
[UserExtensions.NoSuchCorp."MyScript_1"]
MyBatch.bat
```

3.11.1 [UserExtensions.TianoCore."ExtraFiles"] Section

This is an optional section.

Defines the optional EDK II DEC file **[UserExtensions.TianoCore."ExtraFiles"]** section tag. The EDK II build tools must not process any files listed in this section.

Summary

This section is used by the Intel® UEFI Packaging Tool that is distributed as part of the EDK II BaseTools, to locate files listed under this section header and add them to the UEFI distribution package. When installing a UEFI distribution package, these files will be installed in the package's directory tree.

Prototype

```
<UserExtensions>    ::= "[UserExtensions" <TcEf> "]" <EOL> <FileNames>*

<TcEf>              ::= ".TianoCore." <DblQuote> "ExtraFiles" <DblQuote>

<FileNames>         ::= <TS> [<RelativePath>] <File> <EOL>
```

Parameters

FileNames

Paths listed in the filename elements of the this section must be relative to the directory the DEC file resides in. Use of "..", "." and "../" in the directory path is not permitted.

Example

```
[UserExtensions.TianoCore."ExtraFiles"]
  Readme.txt
```

Appendix A
DEC Examples

The following are examples of EDK II package declaration (DEC) files. Line extension characters are not permitted in the DEC file, they are used here for readability.

The first example shows a DEC file for a package that provides only library class definitions and library instances. The second example shows a DEC file that provide library class definitions, library instances and drivers. The third example shows the format of a DEC file for binary only content within the directory structure.

Note: *In the all of the following examples, the backslash "\" character at the end of a line is used to show a line continuation for readability. Use of a backslash character in the actual DEC file is not permitted.*

A.1 EDK II IntelFrameworkPkg Example

```
## @file
# Intel Framework Module Package.
#
# This package contains the definitions and module implementation
# which follows Intel EFI Framework Specification.
#
# Copyright (c) 2007 - 2016, Intel Corporation. All rights reserved.<BR>
#
# This program and the accompanying materials are licensed and made
# available under the terms and conditions of the BSD License
# which accompanies this distribution. The full text of the license may
# be found at:
#   http://opensource.org/licenses/bsd-license.php
#
# THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS,
# WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR
# IMPLIED.
#
##

[Defines]
  DEC_SPECIFICATION              = 0x00010019
  PACKAGE_NAME                   = IntelFrameworkModulePkg
  PACKAGE_UNI_FILE               = IntelFrameworkModulePkg.uni
  PACKAGE_GUID                   = 88894582-7553-4822-B484-624E24B6DECF
  PACKAGE_VERSION                = 0.92

[Includes]
  Include                        # Root include for the package

[LibraryClasses]
  ##  @libraryclass  Platform BDS library definition about platform
  #                  specific behavior.
  PlatformBdsLib|Include/Library/PlatformBdsLib.h

  ##  @libraryclass  Generic BDS library definition, include the data
  #                  structure and function.
  GenericBdsLib|Include/Library/GenericBdsLib.h

[Guids]
  ## IntelFrameworkModule package token space guid
  #  Include/Guid/IntelFrameworkModulePkgTokenSpace.h
  gEfiIntelFrameworkModulePkgTokenSpaceGuid = { 0xD3705011, 0xBC19, \
    0x4af7, { 0xBE, 0x16, 0xF6, 0x80, 0x30, 0x37, 0x8C, 0x15 }}
```

```
## GUID identifies Data Hub records logged by Status Code Runtime
#  Protocol.
#  Include/Guid/DataHubStatusCodeRecord.h
gEfiDataHubStatusCodeRecordGuid = { 0xD083E94C, 0x6560, 0x42E4, { \
  0xB6, 0xD4, 0x2D, 0xF7, 0x5A, 0xDF, 0x6A, 0x2A }}

## GUID indicates the tiano custom compress/decompress algorithm.
#  Include/Guid/TianoDecompress.h
gTianoCustomDecompressGuid     = { 0xA31280AD, 0x481E, 0x41B6, \
  { 0x95, 0xE8, 0x12, 0x7F, 0x4C, 0x98, 0x47, 0x79 }}

## GUID indicates the LZMA custom compress/decompress algorithm.
#  Include/Guid/LzmaDecompress.h
gLzmaCustomDecompressGuid      = { 0xEE4E5898, 0x3914, 0x4259, \
  { 0x9D, 0x6E, 0xDC, 0x7B, 0xD7, 0x94, 0x03, 0xCF }}

## Include/Guid/AcpiVariable.h
gEfiAcpiVariableCompatiblityGuid   = { 0xc020489e, 0x6db2, 0x4ef2, \
  { 0x9a, 0xa5, 0xca, 0x6,  0xfc, 0x11, 0xd3, 0x6a }}

## Include/Guid/LegacyBios.h
gEfiLegacyBiosGuid                 = { 0x2E3044AC, 0x879F, 0x490F, \
  { 0x97, 0x60, 0xBB, 0xDF, 0xAF, 0x69, 0x5F, 0x50 }}

## Include/Guid/LegacyDevOrder.h
gEfiLegacyDevOrderVariableGuid     = { 0xa56074db, 0x65fe, 0x45f7, \
  { 0xbd, 0x21, 0x2d, 0x2b, 0xdd, 0x8e, 0x96, 0x52 }}

## Include/Guid/CapsuleDataFile.h
gEfiUpdateDataFileGuid             = { 0x283fa2ee, 0x532c, 0x484d, \
  { 0x93, 0x83, 0x9f, 0x93, 0xb3, 0x6f, 0xb, 0x7e }}

## Include/Guid/BlockIoVendor.h
gBlockIoVendorGuid                 = { 0xcf31fac5, 0xc24e, 0x11d2, \
  {0x85, 0xf3, 0x0, 0xa0, 0xc9, 0x3e, 0xc9, 0x3b }}

## Include/Guid/BdsHii.h
gFrontPageFormSetGuid              = { 0x9e0c30bc, 0x3f06, 0x4ba6, \
  {0x82, 0x88, 0x9, 0x17, 0x9b, 0x85, 0x5d, 0xbe }}
gBootManagerFormSetGuid            = { 0x847bc3fe, 0xb974, 0x446d, \
  {0x94, 0x49, 0x5a, 0xd5, 0x41, 0x2e, 0x99, 0x3b }}
gDeviceManagerFormSetGuid          = { 0x3ebfa8e6, 0x511d, 0x4b5b, \
  {0xa9, 0x5f, 0xfb, 0x38, 0x26, 0xf, 0x1c, 0x27 }}
gDriverHealthFormSetGuid           = { 0xf76e0a70, 0xb5ed, 0x4c38, \
  {0xac, 0x9a, 0xe5, 0xf5, 0x4b, 0xf1, 0x6e, 0x34 }}
```

```
  gBootMaintFormSetGuid              = { 0x642237c7, 0x35d4, 0x472d, \
    {0x83, 0x65, 0x12, 0xe0, 0xcc, 0xf2, 0x7a, 0x22 }}
  gFileExploreFormSetGuid            = { 0x1f2d63e1, 0xfebd, 0x4dc7, \
    {0x9c, 0xc5, 0xba, 0x2b, 0x1c, 0xef, 0x9c, 0x5b }}

  ## Include/Guid/BdsLibHii.h
  gBdsLibStringPackageGuid           = { 0x3b4d9b23, 0x95ac, 0x44f6, \
    {0x9f, 0xcd, 0xe, 0x95, 0x94, 0x58, 0x6c, 0x72 }}

  ## Include/Guid/LastEnumLang.h
  gLastEnumLangGuid                  = { 0xe8c545b, 0xa2ee, 0x470d, \
    {0x8e, 0x26, 0xbd, 0xa1, 0xa1, 0x3c, 0xa, 0xa3 }}

  ## Include/Guid/HdBootVariable.h
  gHdBootDevicePathVariablGuid       = { 0xfab7e9e1, 0x39dd, 0x4f2b, \
    {0x84, 0x8, 0xe2, 0xe, 0x90, 0x6c, 0xb6, 0xde }}

[Protocols]
  ## Vga Mini port binding for a VGA controller
  #  Include/Protocol/VgaMiniPort.h
  gEfiVgaMiniPortProtocolGuid    = { 0xc7735a2f, 0x88f5, 0x4882, \
    { 0xae, 0x63, 0xfa, 0xac, 0x8c, 0x8b, 0x86, 0xb3 }}

  ## ISA I/O Protocol is used to perform ISA device Io/Mem operations.
  #  Include/Protocol/IsaIo.h
  gEfiIsaIoProtocolGuid          = { 0x7ee2bd44, 0x3da0, 0x11d4, \
    { 0x9a, 0x38, 0x0, 0x90, 0x27, 0x3f, 0xc1, 0x4d }}

  ## ISA Acpi Protocol is used to operate and communicate with ISA
  #  device.
  #  Include/Protocol/IsaAcpi.h
  gEfiIsaAcpiProtocolGuid        = { 0x64a892dc, 0x5561, 0x4536, \
    { 0x92, 0xc7, 0x79, 0x9b, 0xfc, 0x18, 0x33, 0x55 }}

  ## PS/2 policy protocol abstracts the specific platform initialization
  #  and setting.
  #  Include/Protocol/Ps2Policy.h
  gEfiPs2PolicyProtocolGuid      = { 0x4DF19259, 0xDC71, 0x4D46, \
    { 0xBE, 0xF1, 0x35, 0x7B, 0xB5, 0x78, 0xC4, 0x18 }}

  ## OEM Badging Protocol defines the interface to get the OEM badging
  # image with the dispaly attribute.
  #  Include/Protocol/OEMBadging.h
  gEfiOEMBadgingProtocolGuid     = { 0x170E13C0, 0xBF1B, 0x4218, \
    { 0x87, 0x1D, 0x2A, 0xBD, 0xC6, 0xF8, 0x87, 0xBC }}
```

```
  ## Include/Protocol/ExitPmAuth.h
  gExitPmAuthProtocolGuid        = { 0xd088a413, 0xa70, 0x4217, \
    { 0xba, 0x55, 0x9a, 0x3c, 0xb6, 0x5c, 0x41, 0xb3 }}

#
# [Error.gEfiIntelFrameworkModulePkgTokenSpaceGuid]
#   0x80000001 | Invalid value provided.
#   0x80000002 | Reserved bits must be set to zero.
#

[PcdsFeatureFlag]
  ## Indicates if OEM device is enabled as StatusCode report device.
  #  It is only used in Framework StatusCode implementation. <BR><BR>
  #   TRUE  - Enable OEM device.<BR>
  #   FALSE - Disable OEM device.<BR>
  # @Prompt Report StatusCode via OEM Device.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseOEM|FALSE| \
    BOOLEAN|0x00010024

  ## Indicates if StatusCode report is loged into DataHub.<BR><BR>
  #   TRUE  - Log StatusCode report into DataHub.<BR>
  #   FALSE - Does not log StatusCode report into DataHub.<BR>
  # @Prompt Log StatusCode into DataHub.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseDataHub| \
    FALSE|BOOLEAN|0x00010029

  ## Indicates if Serial device uses half hand shake.<BR><BR>
  #   TRUE  - Serial device uses half hand shake.<BR>
  #   FALSE - Serial device doesn't use half hand shake.<BR>
  # @Prompt Enable Serial device Half Hand Shake.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.\
    PcdIsaBusSerialUseHalfHandshake|FALSE|BOOLEAN|0x00010043

  ## Indicates if Legacy support is needed for ACPI S3 Save.<BR><BR>
  #   TRUE  - Support Legacy OS with S3 boot.<BR>
  #   FALSE - Does not support Legacy OS with S3 boot.<BR>
  # @Prompt Turn on Legacy Support in S3 Boot.
gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdPlatformCsmSupport \
    |TRUE|BOOLEAN|0x00010044

  ## Indicates if PS2 keyboard does a extended verification during start.
  #  Extended verification will take some performance. It can be set to
  #  FALSE for boot performance.<BR><BR>
  #   TRUE  - Turn on PS2 keyboard extended verification.<BR>
  #   FALSE - Turn off PS2 keyboard extended verification.<BR>
  # @Prompt Turn on PS2 Keyboard Extended Verification.
```

```
  gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
    PcdPs2KbdExtendedVerification|TRUE|BOOLEAN|0x00010045

  ## Indicates if Framework Acpi Support protocol is installed.<BR><BR>
  #   TRUE  - Install Framework Acpi Support protocol.<BR>
  #   FALSE - Doesn't install Framework Acpi Support protocol.<BR>
  # @Prompt Enable Framework Acpi Support.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
    PcdInstallAcpiSupportProtocol|TRUE|BOOLEAN|0x00010046


  ## Indicates if PS2 mouse does a extended verification during start.
  #  Extended verification will take some performance. It can be set to
  #  FALSE for boot performance.<BR><BR>
  #   TRUE  - Turn on PS2 mouse extended verification. <BR>
  #   FALSE - Turn off PS2 mouse extended verification. <BR>
  # @Prompt Turn on PS2 Mouse Extended Verification
  gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
    PcdPs2MouseExtendedVerification|TRUE|BOOLEAN|0x00010047

  ## Indicates if only Boot logo is showed and all message output is
  #  disabled in BDS.<BR><BR>
  #   TRUE  - Only Boot Logo is showed in boot.<BR>
  #   FALSE - All messages and Boot Logo are showed in boot.<BR>
  # @Prompt Enable Boot Logo only.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdBootlogoOnlyEnable| \
    FALSE|BOOLEAN|0x00010048

[PcdsFixedAtBuild, PcdsPatchableInModule]
  ## FFS filename to find the default BMP Logo file.
  # @Prompt FFS Name of Boot Logo File.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdLogoFile |{ 0x99, 0x8b, \
    0xB2, 0x7B, 0xBB, 0x61, 0xD5, 0x11, 0x9A, 0x5D, 0x00, 0x90, 0x27, \
    0x3F, 0xC1, 0x4D }|VOID*|0x40000003

  ## FFS filename to find the shell application.
  # @Prompt FFS Name of Shell Application.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdShellFile|{ 0xB7, 0xD6, \
    0x7A, 0xC5, 0x15, 0x05, 0xA8, 0x40, 0x9D, 0x21, 0x55, 0x16, \
    0x52, 0x85, 0x4E, 0x37 }|VOID*|0x40000004

  ## ISA Bus features to support DMA, SlaveDMA and ISA Memory. <BR><BR>
  #  BIT0 indicates if DMA is supported<BR>
  #  BIT1 indicates if only slave DMA is supported<BR>
  #  BIT2 indicates if ISA memory is supported<BR>
  #  Other BITs are reseved and must be zero.
```

```
  #  If more than one features are supported, the different BIT will be
  #  enabled at the same time.
  # @Prompt ISA Bus Features.
  # @Expression 0x80000002 | \
  (gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
    PcdIsaBusSupportedFeatures & 0xF8) == 0
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.\
    PcdIsaBusSupportedFeatures|0x05|UINT8|0x00010040

[PcdsDynamic, PcdsDynamicEx]
  ## Indicates if the machine has completed one boot cycle before.
  #  After the complete boot, BootState will be set to FALSE.<BR><BR>
  #   TRUE  - The complete boot cycle has not happened before.<BR>
  #   FALSE - The complete boot cycle has happened before.<BR>
  # @Prompt Boot State Flag.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdBootState|TRUE| \
    BOOLEAN|0x0001002f

  ## Timeout value for displaying progressing bar in before boot OS.
  #  According to UEFI 2.0 spec, the default TimeOut should be 0xffff.
  #  This PCD should be set as HII type PCD by platform integrator mapped
  # to UEFI global variable L"TimeOut".
  # @Prompt Boot Timeout (s).
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdPlatformBootTimeOut| \
    0xffff|UINT16|0x40000001

  ## Error level for hardware recorder.
  #  If value 0, platform does not support feature of hardware error
  #  record.
  #  This PCD should be set as HII type PCD by platform integrator mapped
  #  to UEFI global variable L"HwErrRecSupport".
  # @Prompt Error Level For Hardware Recorder.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
    PcdHardwareErrorRecordLevel|0|UINT16|0x40000002

[PcdsPatchableInModule, PcdsDynamic, PcdsDynamicEx]
  #  The 4 PCDs below are used to specify the video resolution and text
  #  mode of text setup.
  #  To make text setup work in this resolution,
  #  PcdVideoHorizontalResolution, PcdVideoVerticalResolution,
  #  PcdConOutColumn and PcdConOutRow in MdeModulePkg.dec should be
  #  created as PcdsDynamic or PcdsDynamicEx
  #  in platform DSC file. Then BDS setup will update these PCDs defined
  # in MdeModulePkg.dec and reconnect console drivers (GraphicsConsole,
  # Terminal, Consplitter) to make the video resolution and text mode
  # work for text setup.
```

```
  ## Specify the video horizontal resolution of text setup.
  # @Prompt Video Horizontal Resolution of Text Setup.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
    PcdSetupVideoHorizontalResolution|800|UINT32|0x50000001

  ## Specify the video vertical resolution of text setup.
  # @Prompt Video Vertical Resolution of Text Setup.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
    PcdSetupVideoVerticalResolution|600|UINT32|0x50000002

  ## Specify the console output column of text setup.
  # @Prompt Console Output Column of Text Setup.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdSetupConOutColumn| \
    80|UINT32|0x50000003

  ## Specify the console output row of text setup.
  # @Prompt Console Output Row of Text Setup.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdSetupConOutRow|25| \
    UINT32|0x50000004

[PcdsFixedAtBuild, PcdsDynamic, PcdsDynamicEx, PcdsPatchableInModule]
  ## I/O Base address of floppy device controller.
  # @Prompt I/O Base Address of Floppy Device Controller.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdFdcBaseAddress| \
    0x3f0|UINT16|0x30000000

  ## Indicates if BiosVideo driver will switch to 80x25 Text VGA Mode
  #  when exiting boot service.<BR><BR>
  #   TRUE  - Switch to Text VGA Mode.<BR>
  #   FALSE - Does not switch to Text VGA Mode.<BR>
  # @Prompt Switch to Text VGA Mode on UEFI Boot.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
    PcdBiosVideoSetTextVgaModeEnable|FALSE|BOOLEAN|0x30000001

  ## Indicates if BiosVideo driver will check for VESA BIOS Extension
  #  service support.<BR><BR>
  #   TRUE  - Check for VESA BIOS Extension service.<BR>
  #   FALSE - Does not check for VESA BIOS Extension service.<BR>
  # @Prompt Enable Check for VESA BIOS Extension Service.
  gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
    PcdBiosVideoCheckVbeEnable|TRUE|BOOLEAN|0x30000002

  ## Indicates if BiosVideo driver will check for VGA service support.
  #  NOTE: If both PcdBiosVideoCheckVbeEnable and
  #  PcdBiosVideoCheckVgaEnable are set to FALSE,
```

```
 #  that means Graphics Output protocol will not be installed, the VGA
 #  miniport protocol will be installed instead.<BR><BR>
 #   TRUE  - Check for VGA service.<BR>
 #   FALSE - Does not check for VGA service.<BR>
 # @Prompt Enable Check for VGA Service.
 gEfiIntelFrameworkModulePkgTokenSpaceGuid.\
   PcdBiosVideoCheckVgaEnable|TRUE|BOOLEAN|0x30000003

 ## Indicates if memory space for legacy region will be set as
 #  cacheable.<BR><BR>
 #   TRUE  - Set cachebility for legacy region.<BR>
 #   FALSE - Does not set cachebility for legacy region.<BR>
 # @Prompt Enable Cachebility for Legacy Region.
 gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
   PcdLegacyBiosCacheLegacyRegion|TRUE|BOOLEAN|0x00000004

## Specify memory size with bytes to reserve EBDA below 640K for OPROM.
## The value should be a multiple of 4KB.
# @Prompt Reserved EBDA Memory Size.
# @Expression 0x80000001 | \
(gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdEbdaReservedMemorySize \
< 0xA0000) AND \
((gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdEbdaReservedMemorySize \
& 0x1000) == 0)
 gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdEbdaReservedMemorySize| \
   0x8000|UINT32|0x30000005

 ## Specify memory size with page number for a pre-allocated ACPI NVS
 #  memory to be used
 #  by PEI in S3 phase. The default size 32K. When changing the value of
 #  this PCD, the platform developer should make sure the memory size is
 #  large enough to meet PEI requiremnt in S3 phase.
 # @Prompt Reserved S3 Boot ACPI Memory Size.
 gEfiIntelFrameworkModulePkgTokenSpaceGuid. \
   PcdS3AcpiReservedMemorySize|0x8000|UINT32|0x30000006

 ## Specify memory size for boot script executor stack usage in S3
 #  phase.
 #  The default size 32K. When changing the value of this PCD, the
 # platform developer should make sure the memory size is large enough
 # to meet boot script executor requiremnt in S3 phase.
 # @Prompt Reserved S3 Boot Script Stack ACPI Memory Size.
 gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdS3BootScriptStackSize| \
   0x8000|UINT32|0x30000007

 ## Specify the end of address below 1MB for the OPROM.
```

```
#  The last shadowed OpROM should not exceed this address.
# @Prompt Top Address of Shadowed Legacy OpROM.
# @Expression 0x80000001 | \
 gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdEndOpromShadowAddress \
 < 0x100000
gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdEndOpromShadowAddress| \
  0xdffff|UINT32|0x30000008

## Specify the low PMM (Post Memory Manager) size with bytes below 1MB.
#  The value should be a multiple of 4KB.
# @Prompt Low PMM (Post Memory Manager) Size.
# @Expression 0x80000001 | \
(gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdLowPmmMemorySize <
0x100000) AND \
((gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdLowPmmMemorySize & \
0x1000) == 0)
gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdLowPmmMemorySize| \
  0x10000|UINT32|0x30000009

## Specify the high PMM (Post Memory Manager) size with bytes above 1MB.
#  The value should be a multiple of 4KB.
# @Prompt High PMM (Post Memory Manager) Size.
# @Expression 0x80000001 | \
 (gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdHighPmmMemorySize & \
 0x1000) == 0
 gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdHighPmmMemorySize| \
   0x400000|UINT32|0x3000000a
```

A.2 EDK II EmulatorPkg Example

```
## @file
#
# This is the Emu Emulation Environment Platform
#
# Copyright (c) 2008 - 2016, Intel Corporation. All rights reserved.<BR>
# Portions copyright (c) 2011, Apple Inc. All rights reserved.
#
# This program and the accompanying materials are licensed and made
# available under the terms and conditions of the BSD License which
# accompanies this distribution.
# The full text of the license may be found at
#   http://opensource.org/licenses/bsd-license.php
# THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS,
# WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR
# IMPLIED.
#
##

[Defines]
  DEC_SPECIFICATION  = 0x00010019
  PACKAGE_NAME       = EmulatorPkg
  PACKAGE_GUID       = 36E48BD7-7D92-5A47-A2CD-513F072E3300
  PACKAGE_VERSION    = 0.1

[Includes]
  Include

[LibraryClasses]
  ThunkPpiList|Include/Library/ThunkPpiList.h
  ThunkProtocolList|Include/Library/ThunkProtocolList.h
  EmuThunkLib|Include/Library/EmuThunkLib.h
  KeyMap|Include/Library/KeyMapLib.h
  PpiListLib|Include/Library/PpiListLib.h
  SmbiosLib|Include/Library/SmbiosLib.h

[Protocols]
  gEmuThunkProtocolGuid          = { 0x5CF32E0B, 0x8EDF, 0x2E44, \
    { 0x9C, 0xDA, 0x93, 0x20, 0x5E, 0x99, 0xEC, 0x1C } }
  gEmuIoThunkProtocolGuid        = { 0x453368F6, 0x7C85, 0x434A, \
    { 0xA9, 0x8A, 0x72, 0xD1, 0xB7, 0xFF, 0xA9, 0x26 } }
  gEmuGraphicsWindowProtocolGuid = { 0x30FD316A, 0x6728, 0x2E41, \
    { 0xA6, 0x90, 0x0D, 0x13, 0x33, 0xD8, 0xCA, 0xC1 } }
  gEmuThreadThunkProtocolGuid    = { 0x3B1E4B7C, 0x09D8, 0x944F, \
    { 0xA4, 0x08, 0x13, 0x09, 0xEB, 0x8B, 0x44, 0x27 } }
```

```
  gEmuBlockIoProtocolGuid        = { 0x6888A4AE, 0xAFCE, 0xE84B, \
    { 0x91, 0x02, 0xF7, 0xB9, 0xDA, 0xE6, 0xA0, 0x30 } }
  gEmuSnpProtocolGuid            = { 0xFD5FBE54, 0x8C35, 0xB345, \
    { 0x8A, 0x0F, 0x7A, 0xC8, 0xA5, 0xFD, 0x05, 0x21 } }

[Ppis]
  gEmuThunkPpiGuid               = { 0xE113F896, 0x75CF, 0xF640, \
    { 0x81, 0x7F, 0xC8, 0x5A, 0x79, 0xE8, 0xAE, 0x67 } }

[Guids]
  gEmulatorPkgTokenSpaceGuid  = { 0x4F792E68, 0xE8C8, 0x794E, { 0xB1, \
    0xD8, 0x37, 0x03, 0xF3, 0xF2, 0xD5, 0xA5 } }
  gEmuSystemConfigGuid       = { 0xF8626165, 0x6CEB, 0x924A, { 0xBA, \
    0xFC, 0xF1, 0x3A, 0xB9, 0xD6, 0x57, 0x28 } }
  gEmuVirtualDisksGuid       = { 0xf2ba331a, 0x8985, 0x11db, { 0xa4, \
    0x06, 0x00, 0x40, 0xd0, 0x2b, 0x18, 0x35 } }
  gEmuPhysicalDisksGuid      = { 0xf2bdcc96, 0x8985, 0x11db, { 0x87, \
    0x19, 0x00, 0x40, 0xd0, 0x2b, 0x18, 0x35 } }

[PcdsFeatureFlag]
  ## If TRUE, if symbols only load on breakpoints and gdb entry
  gEmulatorPkgTokenSpaceGuid.PcdEmulatorLazyLoadSymbols|TRUE|BOOLEAN| \
    0x00020000

[PcdsFixedAtBuild]
  gEmulatorPkgTokenSpaceGuid.PcdEmuFlashNvStorageVariableBase|0x0| \
    UINT64|0x00001014
  gEmulatorPkgTokenSpaceGuid.PcdEmuFlashNvStorageFtwSpareBase|0x0| \
    UINT64|0x00001015
  gEmulatorPkgTokenSpaceGuid.PcdEmuFlashNvStorageFtwWorkingBase|0x0| \
    UINT64|0x00001016
  gEmulatorPkgTokenSpaceGuid.PcdEmuFdBaseAddress|0x0|UINT64|0x00001017
  gEmulatorPkgTokenSpaceGuid.PcdEmuFlashNvStorageEventLogBase|0x0| \
    UINT64|0x0000100e
  gEmulatorPkgTokenSpaceGuid.PcdEmuFlashNvStorageEventLogSize|0x0| \
    UINT32|0x0000100f
  gEmulatorPkgTokenSpaceGuid.PcdEmuFlashFvRecoveryBase|0x0|UINT64| \
    0x00001010
  gEmulatorPkgTokenSpaceGuid.PcdEmuFlashFvRecoverySize|0x0|UINT32| \
    0x00001011
  gEmulatorPkgTokenSpaceGuid.PcdEmuFirmwareFdSize|0x0|UINT32| \
    0x00001012
  gEmulatorPkgTokenSpaceGuid.PcdEmuFirmwareBlockSize|0|UINT32|0x00001013

  ## Number of Application Processors (APs) in the system 0 means
  # Uniprocessor mode
```

```
  gEmulatorPkgTokenSpaceGuid.PcdEmuApCount|L"0"|VOID*|0x00001019

  ## Magic page to implement PEI Services Table Pointer Lib
  gEmulatorPkgTokenSpaceGuid.PcdPeiServicesTablePage|0x1003000000| \
    UINT64|0x0000101b

  ## Size of the packet filter
  gEmulatorPkgTokenSpaceGuid.PcdNetworkPacketFilterSize|524288| \
    UINT32|0x0000101c

[PcdsFixedAtBuild, PcdsPatchableInModule]
  gEmulatorPkgTokenSpaceGuid.PcdEmuBootMode|1|UINT32|0x00001006
  gEmulatorPkgTokenSpaceGuid.PcdEmuFirmwareVolume| \
    L"..\\Fv\\Fv_Recovery.fd"|VOID*|0x00001009
  gEmulatorPkgTokenSpaceGuid.PcdEmuMemorySize| \
    L"64!64"|VOID*|0x0000100c

  #
  # filename[:[R|F][O|W]][:BlockSize]
  # filename can be a device node, like /dev/disk1
  # R - Removable Media F - Fixed Media
  # O - Write protected W - Writable
  #   Default is Fixed Media, Writable
  # For a file the default BlockSize is 512, and can be overridden via
  # BlockSize, for example 2048 for an ISO CD image. The block size for a
  # device comes from the device and is not configurable.
  # Device Size comes from file or device.
  # On Mac OS X you can use Disk Utility to create .dmg files and mount
  # them like disks
  gEmulatorPkgTokenSpaceGuid.PcdEmuVirtualDisk|L"disk.dmg:FW"| \
    VOID*|0x00001001

  gEmulatorPkgTokenSpaceGuid.PcdEmuGop|L"GOP Window"|VOID*|0x00001018
  gEmulatorPkgTokenSpaceGuid.PcdEmuFileSystem| \
    L".!../../../../../EdkShellBinPkg/bin/ia32/Apps"|VOID*|0x00001004
  gEmulatorPkgTokenSpaceGuid.PcdEmuSerialPort| \
    L"/dev/ttyS0"|VOID*|0x00001002
gEmulatorPkgTokenSpaceGuid.PcdEmuNetworkInterface|L"en0"|VOID*| \
    0x0000100d

  gEmulatorPkgTokenSpaceGuid.PcdEmuCpuModel| \
    L"Intel(R) Processor Model"|VOID*|0x00001007
  gEmulatorPkgTokenSpaceGuid.PcdEmuCpuSpeed|L"3000"|VOID*|0x00001008
  gEmulatorPkgTokenSpaceGuid.PcdEmuMpServicesPollingInterval|0x100| \
    UINT64|0x0000101a
```

A.3 ShellBinPkg.dec

The following is an example of a DEC file where only binary modules are provided; no libraries or source code is present in the package’s directory tree.

```
## @file
# UEFI 2.0 Shell Binary Package
#
# This package contains binary shell application that follows
# UEFI specification and UEFI Shell 2.0 specification.
#
# Copyright (c) 2011 - 2016, Intel Corporation. All rights reserved.<BR>
#
#    This program and the accompanying materials are licensed and made
#    available under the terms and conditions of the BSD License which
#    accompanies this distribution.
#    The full text of the license may be found at:
#      http://opensource.org/licenses/bsd-license.php
#    THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS"
#    BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER
#    EXPRESS OR IMPLIED.
#
##

[Defines]
  DEC_SPECIFICATION  = 0x00010019
  PACKAGE_NAME       = ShellBinPkg
  PACKAGE_GUID       = 4B34AD9D-1324-41e5-8B1D-359AA7BCA62C
  PACKAGE_VERSION    = 0.1
```

A.4 UefiCpuPkg.dec

```
### @file  UefiCpuPkg.dec
# This Package provides UEFI compatible CPU modules and libraries.
#
# Copyright (c) 2007 - 2016, Intel Corporation. All rights reserved.<BR>
#
# This program and the accompanying materials are licensed and made
# available under the terms and conditions of the BSD License which
# accompanies this distribution.
# The full text of the license may be found at:
#   http://opensource.org/licenses/bsd-license.php
#
# THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS,
# WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR
# IMPLIED.
#
##

[Defines]
```

```
  DEC_SPECIFICATION              = 0x00010019
  PACKAGE_NAME                   = UefiCpuPkg
  PACKAGE_UNI_FILE               = UefiCpuPkg.uni
  PACKAGE_GUID                   = 2171df9b-0d39-45aa-ac37-2de190010d23
  PACKAGE_VERSION                = 0.2

[Includes]
  Include

[LibraryClasses]
  ##  @libraryclass  Defines some routines that are generic for IA32
  #                  family CPU to be UEFI specification compliant.
  ##
  UefiCpuLib|Include/Library/UefiCpuLib.h

[LibraryClasses.IA32, LibraryClasses.X64]
  ##  @libraryclass  Provides functions to manage MTRR settings on IA32
  #                  and X64 CPUs.
  ##
  MtrrLib|Include/Library/MtrrLib.h

  ##  @libraryclass  Provides functions to manage the Local APIC on IA32
  #                  and X64 CPUs.
  ##
  LocalApicLib|Include/Library/LocalApicLib.h

[Guids]
  gUefiCpuPkgTokenSpaceGuid      = { 0xac05bf33, 0x995a, 0x4ed4, \
    { 0xaa, 0xb8, 0xef, 0x7a, 0xe8, 0xf, 0x5c, 0xb0 }}

#
# [Error.gUefiCpuPkgTokenSpaceGuid]
#   0x80000001 | Invalid value provided.
#

[PcdsFixedAtBuild, PcdsPatchableInModule]
  ## This value is the CPU Local Apic base address, which aligns the
  #  address on a 4-KByte boundary.
  #  @Prompt Configure base address of CPU Local Apic
  #  @Expression  0x80000001 | \
  (gUefiCpuPkgTokenSpaceGuid.PcdCpuLocalApicBaseAddress & 0xfff) == 0
  gUefiCpuPkgTokenSpaceGuid.PcdCpuLocalApicBaseAddress|0xfee00000| \
    UINT32|0x00000001
```

Appendix B
EDK II Module Types

Table 2. EDK II Module Types

MODULE_TYPE	Supported Architecture Types	Description
BASE	Any	Modules or Libraries can be ported to any execution environment. This module type is intended to be used by silicon module developers to produce source code that is not tied to any specific execution environment.
SEC	Any	Modules of this type are designed to start execution at the reset vector of a CPU. They are responsible for preparing the platform for the PEI phase.
PEI_CORE	Any	This module type is used by PEI Core implementations that are compliant with the PI Specification.
PEIM	Any	This module type is used by PEIMs that are compliant with PI Specification.
DXE_CORE	Any	This module type is used by DXE Core implementations that are compliant with the PI Specification.
DXE_DRIVER	Any	This module type is used by DXE Drivers that are compliant with the PI Specification.
DXE_RUNTIME_DRIVER	Any	This module type is used by DXE Drivers that are compliant to the PI Specification. These modules execute in both boot services and runtime services environments.
DXE_SAL_DRIVER	IPF	This module type is used by DXE Drivers that can be called in physical mode before SetVirtualAddressMap() is called and either physical mode or virtual mode after SetVirtualAddressMap() has been called. This module type is only available for IPF processor types.
DXE_SMM_DRIVER	IA32, X64	This module type is used by DXE Drivers that are loaded into SMRAM.
SMM_CORE	Any	This is the SMM core.
UEFI_DRIVER	Any	This module type is used by UEFI Drivers that are compliant with the EFI 1.10 and UEFI specifications. These modules provide services in the boot services execution environment. UEFI Drivers that return EFI_SUCCESS are not unloaded from memory. UEFI Drivers that return an error are unloaded from memory.
UEFI_APPLICATION	Any	This module type is used by UEFI Applications that are compliant with the EFI 1.10 and EFI 2.0 specifications. UEFI Applications are always unloaded when they exit.