EDK II Image Description File .IDF File Format Specification

TABLE OF CONTENTS

EDK II Image Description File .IDF File Format Specification

1 Introduction

- 1.1 Related Information
- 1.2 Terms
- 1.3 Conventions used in this document

2 HII Image Packs

- 2.1 Defintions
- 2.2 Example

EDK II Image Description File .IDF File Format Specification

EDK II Image Description File .IDF File Format Specification

Revision 1.00

04/30/2025 09:38:17

Acknowledgements

Revision History

Revision	Description	Date
1.0	Initial Release	April 2017
	#477 Document to describe *.idf file format	

1 INTRODUCTION

This document describes file format for Image Description files that are used to create HII Image Packages introduced in the Unified Extensible Firmware Interface Specification, Version 2.1.

1.1 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 2.1, Unified EFI, Inc., 2007, http://www.uefi.org.
- EDK II C Coding Standards Specification, Version 0.51, Intel, 2006, https://edk2.tianocore.org/source/browse/edk2/trunk/docs

1.2 Terms

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

BDS

Framework Boot Device Selection phase.

BNF

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

Component

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

DXE SAL

Framework Driver Execution Environment phase. 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

EFI

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

EFI 1.10 Specification

Intel Corporation published the Extensible Firmware Interface Specification. Intel donated the EFI specification to the Unified EFI Forum, and the UEFI now owns future updates of the EFI specification. See UEFI Specifications.

Foundation

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

Framework

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

GUID

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

HII

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

IFR

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

Library Class

A library class defines the API or interface set for a library. The consumer of the library is coded to the library class definition. Library classes are defined via a library class .h file that is published by a package. See the EDK 2.0 Module Development Environment Library Specification for a list of libraries defined in this package.

Library Instance

An implementation of one or more library classes. See the EDK 2.0 Module Development Environment Library Specification for a list of library defined in this package.

Module

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

Module Type

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

Module Surface Area (MSA)

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

Package

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

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

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

Protocol

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

PCD

Platform Configuration Database.

PEI

Pre-EFI Initialization Phase.

PPI

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

SAL

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

Runtime Services

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

SEC

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

UEFI Application

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

UEFI Driver

A driver that follows the UEFI specification.

UEFI Specification Version 2.0

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

Unified EFI Forum

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

1.3 Conventions used in this document

This document uses the typographic and illustrative conventions described below.

1.3.1 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 First In First Out (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 Interface Specification.

1.3.2 Typographic conventions

This document uses the typographic and illustrative conventions described below:

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. USE ONLY IF YOU MAKE AN ACTUAL CROSS-REFERENCE LINK.
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. or as a definition heading (GlossTerm)
Italic	In text, an Italic typeface can be used as emphasis to introduce a new term or to indicate a manual or specification name.
`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.
`Monospace (blue)`	Words in a `Monospace` typeface that is underlined and in blue indicate an active hyperlink to the code definition for that function or type definition. Click on the word to follow the hyperlink. USE ONLY IF YOU MAKE AN ACTUAL HYPERLINK.
Italic Bold	In code or in text, words in Italic Monospace indicate placeholder names for variable information that must be supplied (i.e., arguments).

2 HII IMAGE PACKS

EDK II Image Description files used for creating HII Image Packs have the following format in EBNF:

```
<ImageDefFileFormat> :: = <CommentLines>*
                          <Content>+
<CommentLines>       :: = "//" <AsciiString> <EOL>
<AsciiString>        :: = [ <TS>* <AsciiChars>* ]*
<AsciiChars>         ::= (0x21 - 0x7E)
<EOL>                ::= <TS> 0x0D 0x0A
<TabSpace>           ::= {<Tab>} {<Space>}
<TS>                 ::= <TabSpace>*
<MTS>                ::= <TabSpace>+
<Content>            ::= {<BlankLine>} {<ImageLine>}
<BlankLine>          ::= <EOL>
<ImageLine>          ::= <TS> "#image" <ImageID> [<TRANS>] <Filename> <EOL>
<ImageID>            ::= <MTS> (a-zA-Z)(a-zA-Z0-9_)*
<TRANS>              ::= <MTS> "TRANSPARENT"
<Filename>           :: =<MTS> <Word> ["." <Extension>]
<Word>               ::= (a-zA-Z0-9_)(a-zA-Z0-9_-,)*
<Extension>          ::= (a-zA-Z0-9_-)+
```

2.1 Definitions

Filename

Alphanumeric characters with optional period ".", dash "-" and/or underscore "_" characters. A period character may not be followed by another period character. No whitespace characters are permitted.

2.2 Example

```
//  The Image definition file
//
// Copyright (c) 2017, 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 that 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.
//

#image IMG_LOGO       TRANSPARENT  Logo.bmp
#image IMG_FULL_LOGO               Logo.jpg
#image IMG_OEM_LOGO                Logo.png
```