Intel® Platform Innovation Framework for EFI Cache Subclass Specification

Draft for Review

Version 0.9

April 1, 2004

Intel order number xxxxxx-001

Revision History

Revision	Revision History	Date
0.9	First public release.	4/1/04

Contents

1 Introduction

Overview

This specification defines the core code that is required for an implementation of the cache data hub subclass of the Intel® Platform Innovation Framework for EFI (hereafter referred to as the "Framework"). This specification does the following:

- Describes the basic components of the cache data hub subclass and cache subclass data records
- Provides code definitions for type and record definitions for the cache subclass that are architecturally required by the *Intel® Platform Innovation Framework for EFI Architecture Specification*

This specification complies with the System Management BIOS (SMBIOS) Reference Specification, version 2.3.4.

Conventions Used in This Document

This document uses the typographic and illustrative conventions described below.

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 multibyte 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:

STRUCTURE NAME: The formal name of the data structure.

Summary: A brief description of the data structure.

Prototype: A “C-style” type declaration for the data structure.

Parameters: A brief description of each field in the data structure prototype.

Description: A description of the functionality provided by the data structure, including any limitations and caveats of which the caller should be aware.

Related Definitions: The type declarations and constants that are used only by this data structure.

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*.

Typographic Conventions

This document uses the typographic and illustrative conventions described below:

Plain text	The normal text typeface is used for the vast majority of the descriptive text in a specification.
Plain text (blue)	In the online help version of this specification, 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. Note that these links are *not* active in the PDF of the specification.
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`	In the online help version of this specification, words in a `Bold 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. Note that these links are *not* active in the PDF of the specification. Also, these inactive links in the PDF may instead have a `Bold Monospace` appearance that is underlined but in dark red. Again, these links are not active in the PDF of the specification.
`Italic Monospace`	In code or in text, words in *`Italic Monospace`* indicate placeholder names for variable information that must be supplied (i.e., arguments).
`Plain Monospace`	In code, words in a `Plain Monospace` typeface that is a dark red color but is not bold or italicized indicate pseudo code or example code. These code segments typically occur in one or more separate paragraphs.

text text text — In the PDF of this specification, text that is highlighted in yellow indicates that a change was made to that text since the previous revision of the PDF. The highlighting indicates only that a change was made since the previous version; it does not specify what changed. If text was deleted and thus cannot be highlighted, a note in red and highlighted in yellow (that looks like *(Note: text text text.)*) appears where the deletion occurred.

See the master Framework glossary in the Framework Interoperability and Component Specifications help system for definitions of terms and abbreviations that are used in this document or that might be useful in understanding the descriptions presented in this document.

See the master Framework references in the Interoperability and Component Specifications help system for a complete list of the additional documents and specifications that are required or suggested for interpreting the information presented in this document.

The Framework Interoperability and Component Specifications help system is available at the following URL:

http://www.intel.com/technology/framework/spec.htm

2 Design Discussion

Overview

This specification describes a group of data records that have similar characteristics. The data records are records that will be input to the data hub to be consumed by one or more drivers.

This specification complies with the *Intel® Platform Innovation Framework for EFI Data Hub Specification* and it is assumed that the consumer of this specification is well versed in the concept of the data hub. This specification describes in more detail how to implement a driver that will log all the cache-related data records and how to create drivers that will consume this data.

This specification also specifies the format of each data record. Each data record must be capable of being used by current and potential consumers of the data—in other words, the format should also be consumable by all potential agents. The unit of measurement, if applicable, should be the most common unit of measurement for the specific data record, and the range of values for a data record should be usable for the foreseeable future.

Scope

This specification is the contract for cache-memory-related data between the cache memory data producers and the cache memory data consumers. The data referred to here is not the data contained in the cache but the data that describes the cache characteristics. This specification covers all data structures that are cache-related and includes all the different cache levels in the system including those that are directly inside the processor and those physically outside. For multiprocessor (MP) systems, it includes all the cache-related data for all processors in the system.

The data drivers (cache drivers in this case) do not have to declare all the record numbers that are listed in this specification but should declare only those data types that are applicable. If the data is not applicable (for example, if no cache memory exists in the systems), the data consumer should be able to handle these cases.

A specific record defines the semantic context of the data. All records that are related to cache are documented in this specification. The record type is numbered sequentially and a new record type can be appended to the end of the list. If a data record needs to be updated, a new entry to this specification can be added. The data records in this specification comply to the **`EFI_SUBCLASS_TYPE1_HEADER`**.*`Version`* field of 1. Type **`EFI_SUBCLASS_TYPE1_HEADER`** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

At some point when the objectives of this subclass are not accomplished or become inconsistent, an entirely new subclass with a new Globally Unique Identifier (GUID) will be introduced.

Compliancy Requirements

None of the record numbers described in this document are required by the *Extensible Firmware Interface Specification.* Some record number entries may be required by other industry-wide specifications such as the *System Management BIOS (SMBIOS) Reference Specification.*

Some **ENUM** definitions are controlled by the Distributed Management Task Force, Inc. (DMTF) organization. Refer to their Web site at www.dmtf.org/standards/dmi* and select the link to *Master MIF*.

Header Information

Data Record Header

Each data record that is logged or read starts with a standard header of type **EFI_DATA_RECORD_HEADER**. The format of the header is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Specification*.

Cache Subclass Definition

The cache subclass belongs to the data class and is identified as the cache subclass by the GUID. See Cache Subclass Definition in Code Definitions for the definition.

Subclass Header

Each data record that is a member of the data class starts with a standard header of type **EFI_SUBCLASS_TYPE1_HEADER**. The format of the header is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*. This header follows the data record header **EFI_DATA_RECORD_HEADER**, which is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Specification*.

The *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide* provides generic descriptions of the fields in **EFI_SUBCLASS_TYPE1_HEADER**. The following explanations further clarify the descriptions for the cache subclass:

- The *Instance* is the instance number of the cache memory subclass (cache level number) produced by the same *ProducerName* in the system. For example, multiple cache levels exist in a system.
- The *SubInstance* is the instance number of the *RecordType* of the cache memory *Instance*. For example, multiple L1 (or L2, etc.) caches exist in an MP system and each processor has its own L1 cache.

Raw Data

The raw data follows the **EFI_SUBCLASS_TYPE1_HEADER** header and its definition is specific to the *RecordType*. The syntax of the raw data is defined in Data Record Number in Code Definitions. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide.*

Data Record Number

The **EFI_SUBCLASS_TYPE1_HEADER** is followed by a data record. The data record format is specific to a *RecordNumber*.

Type **EFI_SUBCLASS_TYPE1_HEADER** and all generic macros are defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*. **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

See Data Record Number in Code Definitions for the structures and data record numbers of the data records in the cache subclass.

3
Code Definitions

Introduction

This section contains the basic definitions of the data record header fields that are specific to the cache subclass, as well as definitions of cache data records. The following data types and data records are defined in this section:

- **EFI_CACHE_SUBCLASS**
- **EFI_CACHE_SUBCLASS_VERSION**
- **EFI_CACHE_SIZE_DATA**
- **EFI_CACHE_MAXIMUM_SIZE_DATA**
- **EFI_CACHE_SPEED_DATA**
- **EFI_CACHE_SOCKET_DATA**
- **EFI_CACHE_SRAM_TYPE_DATA**
- **EFI_CACHE_SRAM_INSTALL_DATA**
- **EFI_CACHE_ERROR_TYPE_DATA**
- **EFI_CACHE_TYPE_DATA**
- **EFI_CACHE_ASSOCIATIVITY_DATA**
- **EFI_CACHE_CONFIGURATION_DATA**

Header Information

Cache Subclass Definition

EFI_CACHE_SUBCLASS

Summary

The cache subclass belongs to the data class and is identified as the cache subclass by the GUID.

GUID

```
#define EFI_CACHE_SUBCLASS_GUID \
{ 0x7f0013a7, 0xdc79, 0x4b22, 0x80, 0x99, 0x11, 0xf7, 0x5f, 0xdc,
0x82, 0x9d }
```

Class

```
#define        EFI_CACHE_SUBCLASS        EFI_DATA_CLASS_DATA
```

Description

The cache subclass belongs to the data class and is identified as the cache subclass by the GUID.

For this subclass, the values defined above are used as follows:

- **EFI_DATA_RECORD_HEADER**.*DataRecordGuid* = **EFI_CACHE_SUBCLASS_GUID**, which is the GUID that is specific to the cache subclass.
- **EFI_DATA_RECORD_HEADER**.*DataRecordClass* = **EFI_DATA_CLASS_DATA**. The "class" may be equal to the GUID "class" or a superset of the GUID "class."

Type **EFI_DATA_CLASS_DATA** is defined in **EFI_DATA_RECORD_HEADER**, which is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Specification*.

Subclass Version

EFI_CACHE_SUBCLASS_VERSION

Summary

Indicates the version of the cache subclass.

Prototype

```
#define EFI_CACHE_SUBCLASS_VERSION     0x00010000
```

Description

This value indicates the version of the cache subclass. It is used in **EFI_SUBCLASS_TYPE1_HEADER**.*Version*. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Data Record Number

Size

EFI_CACHE_SIZE_DATA

Summary

This data record refers to the size of the cache memory.

Prototype

```
typedef EFI_EXP_BASE2_DATA      EFI_CACHE_SIZE_DATA;
```

Description

This data record refers to the size of the cache memory. The unit of measurement of this data record is in bytes.

Type **EFI_EXP_BASE2_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_CACHE_SIZE_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide.*

Related Definitions

```
//*************************************************************
// Record number
//*************************************************************
#define EFI_CACHE_SIZE_RECORD_NUMBER              0x00000001
```

Maximum Size

EFI_CACHE_MAXIMUM_SIZE_DATA

Summary

This data record refers to the maximum size of the cache memory.

Prototype

```
typedef EFI_EXP_BASE2_DATA      EFI_CACHE_MAXIMUM_SIZE_DATA;
```

Description

This data record refers to the maximum size of the cache memory. The unit of measurement of this data record is in bytes.

Type **EFI_EXP_BASE2_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_CACHE_MAXIMUM_SIZE_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*************************************************************
// Record number
//*************************************************************
#define EFI_CACHE_MAXIMUM_SIZE_RECORD_NUMBER        0x00000002
```

Speed

EFI_CACHE_SPEED_DATA

Summary

This data record refers to the speed of the cache memory.

Prototype

```
typedef EFI_EXP_BASE10_DATA             EFI_CACHE_SPEED_DATA;
```

Summary

This data record refers to the speed of the cache memory. The unit of measurement of this data record is in seconds.

Type **EFI_EXP_BASE10_DATA** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_CACHE_SPEED_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide.*

Related Definitions

```
//*************************************************************
// Record number
//*************************************************************
#define EFI_CACHE_SPEED_RECORD_NUMBER              0x00000003
```

Socket

EFI_CACHE_SOCKET_DATA

Summary

This data record refers to the socket of the cache memory.

Prototype

```
typedef STRING_REF          EFI_CACHE_SOCKET_DATA;
```

Description

This data record refers to the socket of the cache memory. This data record is a string identifying the cache socket.

Type **STRING_REF** is defined in the *Intel® Platform Innovation Framework for EFI Human Interface Infrastructure Specification*.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_CACHE_SOCKET_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*************************************************************
// Record number
//*************************************************************
#define EFI_CACHE_SOCKET_RECORD_NUMBER              0x00000004
```

SRAM Type Supported

EFI_CACHE_SRAM_TYPE_DATA

Summary

This data record refers to the static RAM (SRAM) type of the cache memory supported in the system.

Prototype

```
typedef struct {
  UINT32  Other         :1;
  UINT32  Unknown       :1;
  UINT32  NonBurst      :1;
  UINT32  Burst         :1;
  UINT32  PipelineBurst :1;
  UINT32  Asynchronous  :1;
  UINT32  Synchronous   :1;
  UINT32  Reserved      :25;
} EFI_CACHE_SRAM_TYPE_DATA;
```

Description

This data record refers to the static RAM (SRAM) type of the cache memory supported in the system. This data record is a bit mask.

The type definition structure for **EFI_CACHE_SRAM_TYPE** is in SMBIOS 2.3.4, Table 3.3.8.1, Type 7, Offset 0xB.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_CACHE_SRAM_SUPPORT_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_CACHE_SRAM_SUPPORT_RECORD_NUMBER        0x00000005
```

SRAM Type Installed

EFI_CACHE_SRAM_INSTALL_DATA

Summary

This data record refers to the SRAM type of the cache memory that is installed in the system.

Prototype

```
typedef EFI_CACHE_SRAM_TYPE_DATA     EFI_CACHE_SRAM_INSTALL_DATA;
```

Description

This data record refers to the SRAM type of the cache memory that is installed in the system. This data record is a bit mask.

See SRAM Type Supported for the type definition structure for **EFI_CACHE_SRAM_TYPE_DATA**. This data is the same as in SMBIOS 2.3.4, Table 3.3.8.1, Type 7, Offset 0xD.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_CACHE_SRAM_INSTALL_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*************************************************************
// Record number
//*************************************************************
#define EFI_CACHE_SRAM_INSTALL_RECORD_NUMBER       0x00000006
```

Error Correction Type Supported

EFI_CACHE_ERROR_TYPE_DATA

Summary

This data record refers to the error correction type of the cache memory that the system supports.

Prototype

```
typedef enum {
  EfiCacheErrorOther = 1,
  EfiCacheErrorUnknown = 2,
  EfiCacheErrorNone = 3,
  EfiCacheErrorParity = 4,
  EfiCacheErrorSingleBit = 5,
  EfiCacheErrorMultiBit = 6
} EFI_CACHE_ERROR_TYPE_DATA
```

Description

This data record refers to the error correction type of the cache memory that the system supports. This data record is a bit mask.

The type definition structure for **EFI_CACHE_ERROR_TYPE** is in SMBIOS 2.3.4, Table 3.3.8.2, Type 7, Offset 0x10.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_CACHE_ERROR_SUPPORT_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*******************************************************
// Record number
//*******************************************************
#define EFI_CACHE_ERROR_SUPPORT_RECORD_NUMBER      0x00000007
```

System Cache Type

EFI_CACHE_TYPE_DATA

Summary

This data record refers to the type of cache memory that is in the system.

Prototype

```
typedef enum {
  EfiCacheTypeOther = 1,
  EfiCacheTypeUnknown = 2,
  EfiCacheTypeInstruction = 3,
  EfiCacheTypeData = 4,
  EfiCacheTypeUnified = 5
} EFI_CACHE_TYPE_DATA;
```

Description

This data record refers to the type of cache memory that is in the system. This data record is an enumeration.

The type definition structure for **EFI_CACHE_TYPE_DATA** is in SMBIOS 2.3.4, Table 3.3.8.3, Type 7, Offset 0x11.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_CACHE_TYPE_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*************************************************************
// Record number
//*************************************************************
#define EFI_CACHE_TYPE_RECORD_NUMBER                0x00000008
```

Associativity

EFI_CACHE_ASSOCIATIVITY_DATA

Summary

This data record refers to the associativity of cache memory that is in the system.

Prototype

```
typedef enum {
  EfiCacheAssociativityOther = 1,
  EfiCacheAssociativityUnknown = 2
  EfiCacheAssociativityDirectMapped = 3,
  EfiCacheAssociativity2Way = 4,
  EfiCacheAssociativity4Way = 5,
  EfiCacheAssociativityFully = 6,
  EfiCacheAssociativity8Way = 7,
  EfiCacheAssociativity16Way = 8
} EFI_CACHE_ASSOCIATIVITY_DATA;
```

Description

This data record refers to the associativity of cache memory that is in the system. This data record is an enumeration.

The type definition structure for **EFI_CACHE_ASSOCIATIVITY_DATA** is in SMBIOS 2.3.4, Table 3.3.8.4, Type 7, Offset 0x12.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER**.*RecordType* = **EFI_CACHE_ASSOCIATIVITY_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*************************************************************
// Record number
//*************************************************************
#define EFI_CACHE_ASSOCIATIVITY_RECORD_NUMBER      0x00000009
```

Configuration

EFI_CACHE_CONFIGURATION_DATA

Summary

This data record refers to the current configuration of cache memory that is in the system.

Prototype

```
typedef struct {
  UINT16    Level            :3;
  UINT16    Socketed         :1;
  UINT16    Reserved2        :1;
  UINT16    Location         :2;
  UINT16    Enable           :1;
  UINT16    OperationalMode  :2;
  UINT16    Reserved1        :22;
} EFI_CACHE_CONFIGURATION_DATA;
```

Parameters

Level

Indicates the cache level of this data. See "Related Definitions" below for the data types associated with this parameter.

Socketed

Indicates whether this cache memory is socketed or not. See "Related Definitions" below for the data types associated with this parameter.

Reserved2

Reserved. Should be set to zero and reserved for future use.

Location

Indicates the location of this cache memory. See "Related Definitions" below for the data types associated with this parameter.

Enable

Indicates whether this cache memory is enabled or disabled. See "Related Definitions" below for the data types associated with this parameter.

OperationalMode

Describes the operational mode of the cache. See "Related Definitions" below for the data types associated with this parameter.

Reserved1

Reserved. Should be set to zero and reserved for future use.

Description

This data record refers to the current configuration of cache memory that is in the system. This data record is a structure.

The type definition structure for **EFI_CACHE_TYPE** is in SMBIOS 2.3.4, Type 7, Offset 0x5.

EFI_CACHE_CONFIGURATION_DATA.*Level* contains the same value as *Instance*.

For this data record, **EFI_SUBCLASS_TYPE1_HEADER***.RecordType =* **EFI_CACHE_CONFIGURATION_RECORD_NUMBER**. Type **EFI_SUBCLASS_TYPE1_HEADER** is defined in the *Intel® Platform Innovation Framework for EFI Data Hub Subclass Design Guide*.

Related Definitions

```
//*****************************************************************
// Record number
//*****************************************************************
#define EFI_CACHE_CONFIGURATION_RECORD_NUMBER      0x0000000A


//*****************************************************************
// Constants associated with Level
//*****************************************************************
#define  EFI_CACHE_L1            1
#define  EFI_CACHE_L2            2
#define  EFI_CACHE_L3            3
#define  EFI_CACHE_L4            4
#define  EFI_CACHE_LMAX          EFI_CACHE_L4


//*****************************************************************
// Constants associated with Socketed
//*****************************************************************
#define EFI_CACHE_SOCKETED        1
#define EFI_CACHE_NOT_SOCKETED    0


//*****************************************************************
// EFI_CACHE_LOCATION
//*****************************************************************
typedef enum {
  EfiCacheInternal = 0,
  EfiCacheExternal = 1,
  EfiCacheReserved = 2,
  EfiCacheUnknown = 3,
} EFI_CACHE_LOCATION;
```

```
//*************************************************************
// Constants associated with Enable
//*************************************************************
#define EFI_CACHE_ENABLED        1
#define EFI_CACHE_DISABLED       0


//*************************************************************
// EFI_CACHE_OPERATIONAL_MODE
//*************************************************************
typedef enum {
  EfiCacheWriteThrough = 0,
  EfiCacheWriteBack = 1,
  EfiCacheDynamicMode = 2,
  EfiCacheUnknownMode = 3
} EFI_CACHE_OPERATIONAL_MODE;
```