Intel® Firmware Configuration Editor (FCE)

Quick Start Guide

Revision 1.6

August 2017

Table of Contents

1 Introduction

Welcome to the quick start guide for the Intel® Firmware Configuration Editor (FCE). This document is organized as follows:

- **Introduction -** describes the system requirements, special considerations, terminology, and other basic information
- **Quick Start** – how to install and begin using the Intel FCE
- **Constraints and BKMs** – limitations and tips for using the Intel FCE
- **FAQ** – Frequently Asked Questions

1.1 System requirements

To install the Intel FCE, you will need a personal computer with:

- Processor: 1 GHz Intel® Pentium processor or faster
- RAM: 256 MB or more
- Storage: 100 MB or more of available storage space

The Intel FCE application runs on the Microsoft Windows operating system, including the following versions for IA32 and X64-based architectures:

- Windows 10*
- Windows 8/8.1*
- Windows 7*
- Windows XP Professional, SP3*
- Windows Vista Business*

1.2 Special considerations

Keep these considerations in mind when installing and using the Intel FCE:

- The GUID/Name pair defined in the IFR must be consistent with the one stored in the EFI nonvolatile variable store of `.fd`.
- The Intel FCE tools does not support drivers that store HII data in a separate PE/COFF Resource Section as described in the current *UEFI Specification* chapter for *Human Interface Infrastructure Overview*, in the *Design Discussion, Drivers and Applications* section.
- The Intel FCE tool supports only UEFI 2.3 HII and above.

1.3 Terminology and conventions

This quick start guide uses the following terms and conventions:

- C:\MyWorkspace In the example command lines, *C:\MyWorkspace* is used to indicate the directory in which the platform build is launched. When entering commands on your system, make sure to replace C:\MyWorkspace with your local directory name.
- EXAMPLECLIENT.fd In the example command lines, **EXAMPLECLIENT.fd**is used to indicate the build image. When building your own **.fd** image, make sure to replace **EXAMPLECLIENT.fd**with your specific build name.

1.4 For more information

The EDK II source files are located at https://github.com/tianocore/edk2. Related specifications and documentation are at http://www.tianocore.org/docs.

The UEFI specification, including the most recent UEFI platform initialization specifications are located at: http://uefi.org/specifications.

2 Quick Start

This section describes how to install and use the Intel FCE.

2.1 Overview

The Intel Firmware Configuration Editor (Intel FCE) helps you retrieve and change HII configuration (setup) data in firmware device (FD) files: `.fd` files.

The FD file that contains Human Interface Infrastructure (HII) configuration data should be built by the base tools in the build directory. The base tools detect the offset of the HII data, which is stored simply as global data in the PE/COFF image at a build time. The base tools store the offset data in a newly created raw section of firmware file system (FFS) file. Without this configuration, Intel FCE cannot reliably locate HII data.

When run in common mode, Intel FCE manipulates only uncompressed versions of EFI_VARSTORE_IFR and EFI_VARSTORE_IFR_EFI. When run in multi-platform mode, Intel FCE manipulates only the uncompressed versions of UEFI 2.31 EFI_VARSTORE_IFR_EFI.

Intel FCE supports the following question types:

- CHECKBOX
- ORDERED_LIST
- ONE_OF
- NUMERIC
- STRING

2.2 Installing Intel FCE

Before you can use Intel FCE, you must configure the base tools, build the firmware device (FD) image, then install Intel FCE. These simple steps show how to build the FD image and install Intel FCE:

1. Open a Command Prompt window.
2. Change to the directory in which the platform build command is launched (for example, C:\MyWorkspace).
3. Build the FD image.
4. Execute Intel FCE.

You can now perform a variety of tasks using Intel FCE commands.

2.3 Intel FCE command options

You can use Intel FCE to read, update, and verify HII information. Intel FCE also includes a help command:

```
fce.exe                  [read    -i <infd>  [<PlatformId UQI>] ['>' <script>]]
fce.exe  [update        -i <infd>-s <script> -o              <outfd>] [--
         remove|--ignore] [-g <FvNameGuid>]] [-a]
fce.exe                  [verify  -i <infd>  -s <script>]
fce.exe                  [updateq -i <infd>  -o <outfd> <UQI> <Question Type>
<Value>] fce.exe         [[help] | [-?]]
```

2.3.1 Command options

Intel FCE includes the following command options:

read	Extract the HII data from the **<infd>** file.
update	Update the HII data to the **<outfd>** file.
verify	Verify the current platform configuration.
updateq	Update the current platform configuration by command line.
help	Display help information.
<infd>	The name of an existing Firmware Device binary file input.
<PlatformId UQI>	The UQI is required in multi-platform mode to represent a PlatformId question from the VFR files used during binary image creation. It must not be used for common mode.
<outfd>	The name of a Firmware Device binary file output.
<script>	The name of a configuration script.
<UQI>	A hex number followed by an array of hex numbers.
<Question Type>	Any one of ORDERED_LIST, ONE_OF, NUMERIC, STRING, or CHECKBOX.
<Value>	If the <Question Type> is ONE_OF, NUMERIC, or CHECKBOX then **<Value>** will be a single hex number. If the <Question Type> is ORDERED_LIST then **<Value>** will be a single hex number containing the array size followed by an array of that length of hex numbers. If the <Question Type> is STRING then **<Value>** will be a string with double quotation marks.
<FvNameGuid>	GuidValue is one specific FvName guid value.

-i	Import an existing Firmware Device file <infd>.
-o	Create the new Firmware Device with the changes made.
-s	Import config scripts.
>	Redirect the output to a script.
-?	Display help information.
--remove	If one or more of the settings that are updated also exists in NV RAM, remove them only in multi-platform mode.
--ignore	If one or more of the settings that are updated also exists in NV RAM, ignore them only in multi-platform mode.
-g	Specify the FV image to store the multi-platform default setting. If it is missing, the multi-platform default will be inserted into BFV image.
-a	Specify this tool to choose the smaller size between two different storage formats in NV RAM. It's only vaild in multi-platform mode.

2.3.2 Common mode and multi-platform mode

You can run Intel FCE in either of the following modes:

- **Common mode**: Extract the HII data from IFR binary and update it to the EFI variable.
- **Multi-Platform mode**: The default value depends on the PlatformId and DefaultId described in the VFR files. This tool will create the binary file to store default settings at build time for different platforms and modes adding all default settings into BFV as one FFS.

2.4 Using the Intel FCE

This discussion shows how to use the Intel FCE command set to perform different tasks in common mode and multi-platform mode.

2.4.1 Using Intel FCE in common mode

In common mode, Intel FCE lets you extract HII data from an IFR binary and store the UEFI variables in the FD storage space.

2.4.1.1 Extract and Edit HII Data

The followings steps show how to extract data from the `.fd` file and write that data to a script file.

1. Run Intel FCE, passing in the FD file (`EXAMPLECLIENT.fd`) as the input, and redirecting the text output to `Data1.txt`. Enter this command:

   ```
   FCE.exe read -i EXAMPLECLIENT.fd > Data1.txt
   ```

2. Edit the output file `Data1.txt` using a standard text editor (such as Microsoft NotePad*). The edits consist of the values associated with questions. You can also delete all lines in the script that you don't want to change. When editing the file, note that:

   - In the example output file (below), the `0004 0047 0044 0058 0043` between Q and // is the UQI (unique question identifier) string, which is the only identifier for the question.
   - The format is Q <UQI> <Question Type> <Value> // [anything else].
   - `<UQI>` is a hex number followed by an array of hex numbers. The array size is the first number. The UQI identifies a question generically across different platforms. Additional information: UQI is, in fact, the hexadecimal display of a length-and-data UNICODE string: the 0004 is the string length, the 0047 0044 0058 0043 is the only character in the string.
   - `<Question Type>` is any one of ORDERED_LIST, ONE_OF, NUMERIC, STRING, or CHECKBOX.
     - If the `<Question Type>` is ONE_OF, NUMERIC, or CHECKBOX, the value will be a single hex number.
     - If the `<Question Type>` is ORDERED_LIST, the value will be a single hex number containing the array size followed by an array of that length of hex numbers.
     - If the `<Question Type>` is STRING, the value will be a string with double quotation marks.

The edited output file for this step should look similar to the following:

```
Intel(R) Firmware Configuration Editor. (Intel(R) FCE) Version 0.1 Start the
Read Mode:
// Form Set: Intel Test Menu
// Form Set GUID: ec87d643-eba4-4bb5-a1e5-3f3e36b20da9
// Form: Intel Test Menu Q 0004
0047 0044
// 00 = Disabled
// 01 = Enabled                0058 0043 ONE_OF 00 // GDXC
Q 0015 0052 0065
0066
  006F 0072 0020
for                            0073 0065 0072 0076 0065 0020 0031 0036 004D 0020
  TAGEC
// 00 = Disabled               0054 0041 0047 0045 0043 ONE_OF 00 // Reserve 16M
// 01 = Enabled
```

Note: *In the edited output file, the 0004 0047 0044 0058 0043 between Q and // is the UQI string, which is the only identifier for the question.*

3. Save the changed output file to a new file, such as `Data2.txt`.

2.4.1.2 Update the FD file via a script file

Intel FCE lets you use a script file to update the FD file and create a new binary image.

Note: *You can also use a script file to update the FD file and create a new binary image in multi-platform mode. For more information about using script files in multi-platform mode, refer to the discussion on support for multi-platform mode for information, later in this section.*

The following example shows how to use Intel FCE in common mode to update the FD file using a script file. In the example, you modify the value in `Data2.txt` and change the CHECKBOX value from 00 to 01. You could then update the `EXAMPLECLIENT.fd` file to apply the changes to create `Out.fd`, and then burn the file to the ROM.

1. Enter the following command to use a script file to update the FD file:

   ```
   FCE.exe update -i EXAMPLECLIENT.fd -s Data2.txt -o Out.fd
   ```

   The output should look similar to the following:

   ```
   Intel(R) Firmware Configuration Editor. (Intel(R) FCE) Version 0.1
   Start the Update Mode:

   -- Update List --

   [Script line 3522] Update No.1:

   Q  0004 0047 0044 0058 0043 ONE_OF

   [ Update Value From: 0To: 1 ]

   [Script line 3526] Update No.2:

   Q  0015 0052 0065 0073 0065 0072 0076 0065 0020 0031 0036 004d
   0020

   0066 006f 0072 0020 0054 0041 0047 0045 0043 ONE_OF

   [ Update Value From: 0To: 1 ]

   [Results]: 2 questions have been updated successfully in total.

   Congratulations. The output Fd file 'Out.fd' has been completed
   successfully.
   ```

2. Update the target system firmware image with the *Out.fd*.

2.4.1.3 Verify the configuration in the BIOS

You can use Intel FCE to verify the current platform configuration (the binary image `EXAMPLECLIENT.fd`) using the script file `Data2.txt`. To do so, enter this command:

```
FCE.exe verify -i EXAMPLECLIENT.fd -s Data2.txt
```

Any questions that exist in both the script and the current platform and which have different values will be logged to the screen (or redirected to a file). The output will look like the following code, and there will be two differences between `Data2.txt` and `EXAMPLECLIENT.fd`.

```
Intel(R) Firmware Configuration Editor, (Intel(R) FCE) Version 0.1 Start the
Verify Mode:

                    -- Different List --

[Script line 3522] Difference No.1:
[Data2.txt            ]: Q  0004 0047 0044 0058 0043 ONE_OF 1
[EXAMPLECLIENT.fd]: Q          0004 0047 0044 0058 0043 ONE_OF 0

[Script line 3526] Difference No.2:
[Data2.txt            ]: Q  0015 0052 0065 0073 0065 0072 0076 0065 0020 0031
                   0036 004d 0020 0066 006f 0072 0020 0054 0041 0047
                      0045 0043 ONE_OF 1
[EXAMPLECLIENT.fd]: Q          0015 0052 0065 0073 0065 0072 0076 0065 0020
0031
                   0036 004d 0020 0066 006f 0072 0020 0054 0041 0047
                      0045 0043 ONE_OF 0
```

2.4.1.4 Update the binary image via a command line

You can use Intel FCE via a command line interface to update the FD file with a new binary image. This operation is similar to the update operation, but it lets you update the current platform configuration from a command line, instead of reading a script file.

Here is an example of getting the `<UQI> <Question Type> <Value>` from a script question line between the Q and //, and inserting it into the command line:

```
FCE.exe updateq -i EXAMPLECLIENT.fd -o Out.fd 0004 0047 0044 0058 0043
ONE_OF 01
```

The output for that command will look similar to this:

```
Intel(R) Firmware Configuration Editor, (Intel(R) FCE) Version 0.1 Start the
Update Quick Mode:

                     -- Update List --


[Script line 0] Update No.1:
Q                 0004 0047 0044 0058 0043 ONE_OF
[ Update Value From: 0                To: 1 ]
[Results]: 1 question has been updated successfully in total.
Congratulations. The output Fd file 'Out.fd' has been completed successfully.
```

2.4.2 Support for multi-platform mode

When you execute Intel FCE in multi-platform mode, the tool creates a single BIOS image that supports all platforms and modes (normal mode and manufacture mode). In multi-platform mode, Intel FCE creates a BIOS image that (at build time) stores the default values for different platforms and modes, and adds the default settings to the boot firmware volume (BFV) as a single FFS.

During the boot process, drivers (such as PEI, DXE, and SMM) look for or store platform variables as appropriate. If a driver cannot find a variable, it uses a default value.

When a default value needs to be loaded, platform code gets the matched default setting from the storage FFS of the BFV. The platform code then produces the GUID hand-off block (HOB) in which to store the default setting. The PEI variable driver first finds the variable from the GUID HOB, then finds the variable from the EFI nonvolatile variable store of .fd. The DXE variable driver saves all variables from the GUID HOB to the EFI nonvolatile variable store once the variable write is ready for the next boot.

The default value depends on the `PlatformId` and `DefaultId` values described in the VFR files. The default value can be specified in a VFR file using a VFR default value expression, for example:

```
default value=cond(ideqval SETUP_VOLATILE_DATA.PlatId == BoardIdEcolaFalls
? 0x1:0x0), defaultstore = MyStandardDefault
```

The FFS file of this binary is type of EFI_FV_FILETYPE_FREEFORM. The FFS file begins with EFI_FFS_FILE_HEADER, and is then followed by the data section for every `PlatformId` and `DefaultId`. If the variables under different `DefaultId` and `PlatformId` are the same, they will be merged into one data section.

The data section is of type EFI_SECTION_RAW, and begins with EFI_COMMON_SECTION_HEADER. The data section is then followed by the `DefaultId`, `PlatformId` and variable data in turns. The format of a data section looks like this:

```
UINT16: Offset
UINT16: DefaultId1, DataType: PlatformId1 (The data type of platformId is
defined in VFR)
......
UINT16: DefaultIdn, DataType: PlatformIdn VARIABLE_STORE_HEADER
EFI_VARSTORE1
......
EFI_VARSTOREn
```

2.4.2.1 Use Intel FCE in multi-platform mode

These steps below show how to use the multi-platform mode.

1. Run Intel FCE, passing in the FD file (`EXAMPLECLIENT.fd`) as the input, calculating the default value by the values described in `PlatformId UQI`, and redirecting the text output to `Data.txt`.
2. Enter this command:

   ```
   FCE.exe read -i EXAMPLECLIENT.fd 0006 005C 0078 0030 0032 0036 0032 >
   Data.txt
   ```

The “0006 005C 0078 0030 0032 0036 0032” represents the `PlatformId` question in the IFR file. The output will look similar to this:

```
Intel(R) Firmware Configuration Editor, (Intel(R) FCE) Version 0.23


      Start the Read
      Mode:

 // FCEKEY DEFAULT_ID:   0    0    0    0

 //FCEKEY PLATFORM_ID:   0    1    2    3

 //FCEKEY PLATFORM_UQI:  0006 005C 0078 0030 0032 0036 0032

// Form Set: Intel Test Menu
// Form Set GUID: ec87d643-eba4-4bb5-a1e5-3f3e36b20da9
// Form: Intel Test Menu
Q 0006 005C 0078 0043 0030 0030 0030 ONE_OF 00 // GDXC
// 00 = Disabled
// 01 = Enabled
Q 0006 005C 0078 0043 0030 0031 0035 ONE_OF 00 // Reserve 16M for TAGEC
// 00 = Disabled
// 01 = Enabled
```

In the output, you can see the array of `DefaultId` and `PlatformId`: (0, 0), (0, 1), (0, 2) and (0, 3) have the same default values. Also, the current `PlatformId UQI` is 0006 005c 0078 0030 0032 0036 0032.

2.4.2.2 Extract and edit HII data

The procedure for extracting and editing data from an FD file and writing that data to a script file is the same for both common mode and multi-platform mode. An example of extracting and editing HII data is shown earlier in this guide, in the common-mode discussion.

2.4.2.3 Update the FD file via a script file

You can use a script file to update the FD file and create a new binary image in multi-platform mode. For example, you could modify the value in `Data2.txt` and change the CHECKBOX value from 00 to 01. You could then update the `EXAMPLECLIENT.fd` file to apply the changes to create `Out.fd`, and then save all default values as one FFS into all boot firmware volumes of the FD. (A BFV as SecCore.)

Enter this command to use a script file to update the FD file:

```
    FCE.exe update -i EXAMPLECLIENT.fd -s Data2.txt -o Out.fd
```

The output should look similar to this:

```
    Intel(R) Firmware Configuration Editor, (Intel(R) FCE) Version 0.1 Start
    the Update Mode:
                         -- Update List --
    [Script line 3522] Update No.1:
    Q             0004 0047 0044 0058 0043 ONE_OF
    [ Update Value From: 0             To: 1 ]
    [Script line 3526] Update No.2:
    Q             0015 0052 0065 0073 0065 0072 0076 0065 0020 0031 0036 004d 0020
         0066 006f 0072 0020 0054 0041 0047 0045 0043 ONE_OF
    [ Update Value From: 0             To: 1 ]
    [Results]: 2 questions have been updated successfully in total.
    Congratulations. The output Fd file 'Out.fd' has been completed successfully.
```

2.4.2.4 Verify the configuration in the BIOS

You can use Intel FCE to verify the current platform configuration stored in the BFV (the binary image EXAMPLECLIENT.fd) using the script file Data.txt. To do so, enter this command:

```
 FCE.exe verify -i EXAMPLECLIENT.fd -s Data2.txt
```

Any questions that exist in both the script and the current platform and which have different values will be logged to the screen (or redirected to a file).

The output will look like the following code, and there will be two differences between `Data2.txt` and `EXAMPLECLIENT.fd`.

```
 Intel(R) Firmware Configuration Editor, (Intel(R) FCE) Version 0.1 Start the
 Verify Mode:
                      -- Different List --
 [Script line 3522] Difference No.1:
 [Data2.txt                   ]: Q  0004 0047 0044 0058 0043 ONE_OF 1
 [EXAMPLECLIENT.fd]: Q                   0004 0047 0044 0058 0043 ONE_OF 0
 [Script line 3526] Difference No.2:
 [Data2.txt                   ]: Q  0015 0052 0065 0073 0065 0072 0076 0065 0020
 0031
                    0036 004d 0020 0066 006f 0072 0020 0054 0041 0047
                         0045 0043 ONE_OF 1
 [EXAMPLECLIENT.fd]: Q                   0015 0052 0065 0073 0065 0072 0076 0065
 0020 0031
                    0036 004d 0020 0066 006f 0072 0020 0054 0041 0047
                         0045 0043 ONE_OF 0
```

3 Constraints and Best Known Methods (BKMs)

This section describes some key constraints and best known methods (BKMs) for using the Intel FCE.

3.1 Automatic calling of a custom GUIDed tool is not supported

Intel FCE currently does not support automatically calling a custom GUIDed tool for EFI_SECTION_GUID_DEFINED sections in an FFS file that require the custom GUIDed tool. However, you can update the GuidToolDefinitionConf.ini file to add the GUID-NAME mapping relationship that will allow this tool to process the custom EFI_SECTION_GUID_DEFINED section.

Intel FCE currently supports the following standard GUIDed tools:

- a31280ad-481e-41b6-95e8-127f4c984779 TIANO TianoCompress
- ee4e5898-3914-4259-9d6e-dc7bd79403cf LZMA LzmaCompress
- fc1bcdb0-7d31-49aa-936a-a4600d9dd083 CRC32 GenCrc32
- d42ae6bd-1352-4bfb-909a-ca72a6eae889 LZMAF86 LzmaF86Compress

3.2 Variable GUID and name constraints

Intel FCE can be used to configure only the modules which store variables by

GUID and name, as defined in the `.VFR` file. It cannot handle the variables which are not defined in .VFR but which are still set by the UEFI variable service in UEFI drivers as part of UEFI driver model connect operations. If a module

doesn't follow these two preconditions, the Intel FCE will not fail or exit, but

the modification will not take effect in boot.

3.3 Define a PlatformId question in multi-platform mode

The `PlatformId` is defined in the `PlatformId` question. The default value of other questions may depend on the platform value under different `DefaultId` and `PlatformId`. Currently, Intel FCE supports two methods to define this kind of question.

1. Use numeric opcode to define a valid range for `PlatformId`:

```
numeric varid = SETUP_VOLATILE_DATA.PlatId,
          prompt                        = STRING_TOKEN(PLATFORM_ID),    # UQI
value is this string token on UQI language
          help                          =  STRING_TOKEN(PLATFORM_ID),
minimum      = 0,
         maximum

       = 0x34, endnumeric;
```

   In the above example, the PlatformId will be from 0 to 0x34.

2. Use the `OneOf` opcode to define the valid values for `PlatformId`:

```
oneof varid = SETUP_VOLATILE_DATA.PlatId, prompt      =
       STRING_TOKEN(PLATFORM_ID), help   =
       STRING_TOKEN(PLATFORM_ID),

       option text = STRING_TOKEN(ID0), value = 0, flags = 0; option text =
       STRING_TOKEN(ID2), value = 2, flags = 0; option text =
       STRING_TOKEN(ID4), value = 4, flags = 0; option text =
       STRING_TOKEN(ID7), value = 7, flags = 0; option text =
       STRING_TOKEN(ID10), value = 10, flags = 0; option text =
       STRING_TOKEN(ID23), value = 23, flags = 0;

     endoneof;
```

In the above example, the `PlatformId` will be 0, 2, 4, 7, 10, 23.

3.4 Define a DefaultId in multi-platform mode

Intel FCE can parse IFR opcode to get all defined default IDs. Here is an example of a definition of `DefaultId` in VFR:

```
defaultstore MyStandardDefault,
      prompt         =  STRING_TOKEN(STR_STANDARD_DEFAULT_PROMPT),
      attribute      = 0x0000;         // Default ID: 0000 standard default
 defaultstore MyManufactureDefault,
      prompt         =  STRING_TOKEN(STR_MANUFACTURE_DEFAULT_PROMPT),
      attribute      = 0x0001;         // Default ID: 0001 manufacture default
```

3.5 Considerations for the build environment

This quick start guide uses examples of a build environment based on *C:\MyWorkspace*. If your environment is set up differently, remember that Intel FCE depends on the structure of the build environment. If you already have a basetools project, you can use the method described in the previous section to modify the drivers in your workspace.

Note: *In the example command lines in this guide, C:\MyWorkspace is used to indicate the directory in which the platform build is launched. When entering commands on your system, make sure to replace C:\MyWorkspace with your own directory name.*

4 FAQ

This section provides answers to questions commonly asked about using Intel FCE, question fields in the output files, and setup browsers.

4.1 Why does this tool dump out anything from an .fd file?

Intel FCE looks for an HII data offset in the raw section of the firmware file system (FFS) file. If the offset has not yet been generated by the base tools, Intel FCE dumps other data from the binary image. You must update the base tool GenFds in order to generate the offset data and store it in the appropriate location, before Intel FCE can find that offset.

4.2 Why isn’t the value in the setup browser modified correctly?

There could be several reasons the value in the setup browser isn’t modified correctly:

a) The driver doesn't satisfy the precondition. This tool can configure only the modules which store variables by the GUID and name defined in the `.VFR` file.

b) In common mode, the variable defined in .VFR as EFI_VARSTORE_IFR is used as a buffer storage, but has not been stored in the EFI variable zone in system flash.

c) The value is corrected and overlapped in the booting process.

4.3 Why can't the question listed in the script file be found in the current setup browser?

Intel FCE 0.23 and later supports the condition statement calculation. It will not output the questions if the disableif and supressif expressions are constant and are always TRUE. If these two conditions are not matched, the question will be listed in the script. The question will appear only when the conditions in the condition statement have been satisfied.

Here is an example script file:

```
suppressif ideqval SETUP_CPU_FEATURES.XECoreRatioLimitAvailable == 0 OR
        ideqval SETUP_DATA.EnableGv == 0

        OR ideqval SETUP_DATA.TurboMode == 0

        OR NOT ideqval SETUP_CPU_FEATURES.XEorOCSupportAvailable == 2; oneof
  varid                         = SETUP_DATA.EnableXe,
      prompt                    =
      STRING_TOKEN(STR_CPU_XE_PROMPT), help    =
      STRING_TOKEN(STR_CPU_XE_HELP),
      option text = STRING_TOKEN(STR_DISABLED),
                                                               valu
            e = 0, flags = DEFAULT | MANUFACTURING | RESET_REQUIRED;

      option text = STRING_TOKEN(STR_ENABLED),
                                                              va
            lue = 1, flags = RESET_REQUIRED;

  endoneof;
endif;
```

4.4 There is an Update Quick Mode. Why shouldn't I use the updated Question option for that mode?

In your files, there will be only one UQI, but there could be several questions associated with that UQI

The Update Quick Mode always updates the first matched question it encounters, and ignores all other questions that follow. If you want to update more than one question for a single UQI, use a script file instead of the Update Quick Mode.

4.5 What does the UQI represent if the current module doesn't include a UQI definition?

The UQI is translated from the en, en-US, or eng string defined in the `.UNI` file. The Intel FCE algorithm uses the UQI if it exists, and uses en/eng/en-US if the module does not include a UQI. Look at the following examples:

a) **Using UQI.** In this example, the 0006 005C 0078 0030 0030 0033 0043 is translated from the UQI string defined in `UqiList.uni.`

```
#string STR_AMTBX_SETUP_PROMPT        #language uqi  "\x003C"
```

The output should look similar to this:

```
Q 0006 005C 0078 0030 0030 0033 0043 CHECKBOX 00 // Fast Boot
// 0 = Unchecked
// 1 = Checked
```

b) **Translation** from an en/eng/en-US string. In this example, the module does not define a UQI, so the 0004 0042 0049 0053 0054 is translated from the en/eng/en-US package defined in `CPU.uni`.

```
#string STR_BIST_PROMPT               #language eng "BIST"
```

The output should look similar to this:

```
Q 0004 0042 0049 0053 0054 ONE_OF 00 // BIST
// 00 = Disabled
// 01 = Enabled
```

c) **Having no UQI in the module.** In this example, the driver does not include a UQI definition, so the string defined in en/eng/en-US is NULL string.

```
#string STR_EMPTY                 #language en-US ""
```

The output should look similar to this:

```
// [No UQI] NUMERIC 0000 //
// Minimum = 0000
// Maximum = 00F0
// Step    = 0000
```

4.6 Why is an error reported when I use the update feature in multi-platform mode or common mode?

If you try to update an FD that already has UEFI variables stored in its storage space or in the FFS in the boot firmware volume (BFV), you will receive an error, and the FD will not be updated. In multiplatform mode, the error means that there are UEFI variables already stored in the FD storage space. You can use "-- remove" or "--ignore" to decide how to handle this case manually.

- Using "--remove" will clean the variable associated with the IFR in NV RAM while updating the storage FFS in the BFV.
- Using "--ignore" will do nothing to the variable associated with the IFR in NV RAM while updating the storage FFS in the BFV.

In common mode, the error means that the FD already has an FFS in its boot firmware volume.

4.7 Where can I get more information?

See section 1.4 of this document for more information about Intel FCE, plus related specifications and documentation.