Intel® Integrator Toolkit (ITK)

User Guide

Version 6.1.8

May 4, 2018

Contents

Revision History

Revision Number	Description	Revision Date
6.1.0	Initial release.	January 18, 2017
6.1.1	Small bug fix. Updated the validation list (Table 3.2).	January 20, 2017
6.1.2	Fixed issue where factory default SMBIOS values were not detected.	January 25, 2017
6.1.3	Increased maximum SMBIOS string length to 40. Added screenshots.	February 3, 2017
6.1.4	Small bug fixes.	February 8, 2017
6.1.5	Added resolution restrictions to logo file (.JPG). Small bug fixes.	February 15, 2017
6.1.6	Increased supported monitor types for logo file (max 60KB).	March 31, 2017
6.1.8	OA3 key is now deleted properly.	May 4, 2018

1 Introduction

The Intel® Integrator Toolkit (ITK) is a UEFI command line tool that is designed to assist integrators with the process of customizing the BIOS of Intel® NUC or Intel® Compute Stick products.

Section 1.1 gives a brief introduction on how to start using the tool, and Section 2 gives detailed information on how to use certain features. Section 3.2 provides a table with the current validated features for various Intel® NUC or Intel® Compute Stick products.

1.1 Getting Started

This section provides a high-level description of how to start using the Intel® Integrator Toolkit (ITK). Before using this tool, Intel recommends updating the BIOS of the Intel® NUC or Intel® Compute Stick to the latest version (http://downloadcenter.intel.com) and loading the default settings (`F9`).

In order to use the tool, the `ITK6.efi` program must be placed on a FAT32 formatted USB drive. Also, the system that is running this tool (an Intel® NUC or Intel® Compute Stick) must have the internal UEFI shell enabled in the BIOS. This option can be found in in the BIOS options (`F2`) in `Advanced->Boot->Boot Configuration->Boot Devices` or in `Configuration` depending on your product.

Once the internal UEFI shell has been enabled, boot to the shell and navigate (`cd`) to the USB drive's directory that contains the `ITK6.efi` file (for more information on how use the internal UEFI shell, go to Section 1.2). The tool is then run by typing "`ITK6.efi`" along with the various flags and values depending on the feature that is being used.

The full list of features, as well as examples of how they are used, can be found in Section 2.

Note: This tool is only supported in a 64-bit internal UEFI shell.

1.2 Basic UEFI Shell Guide

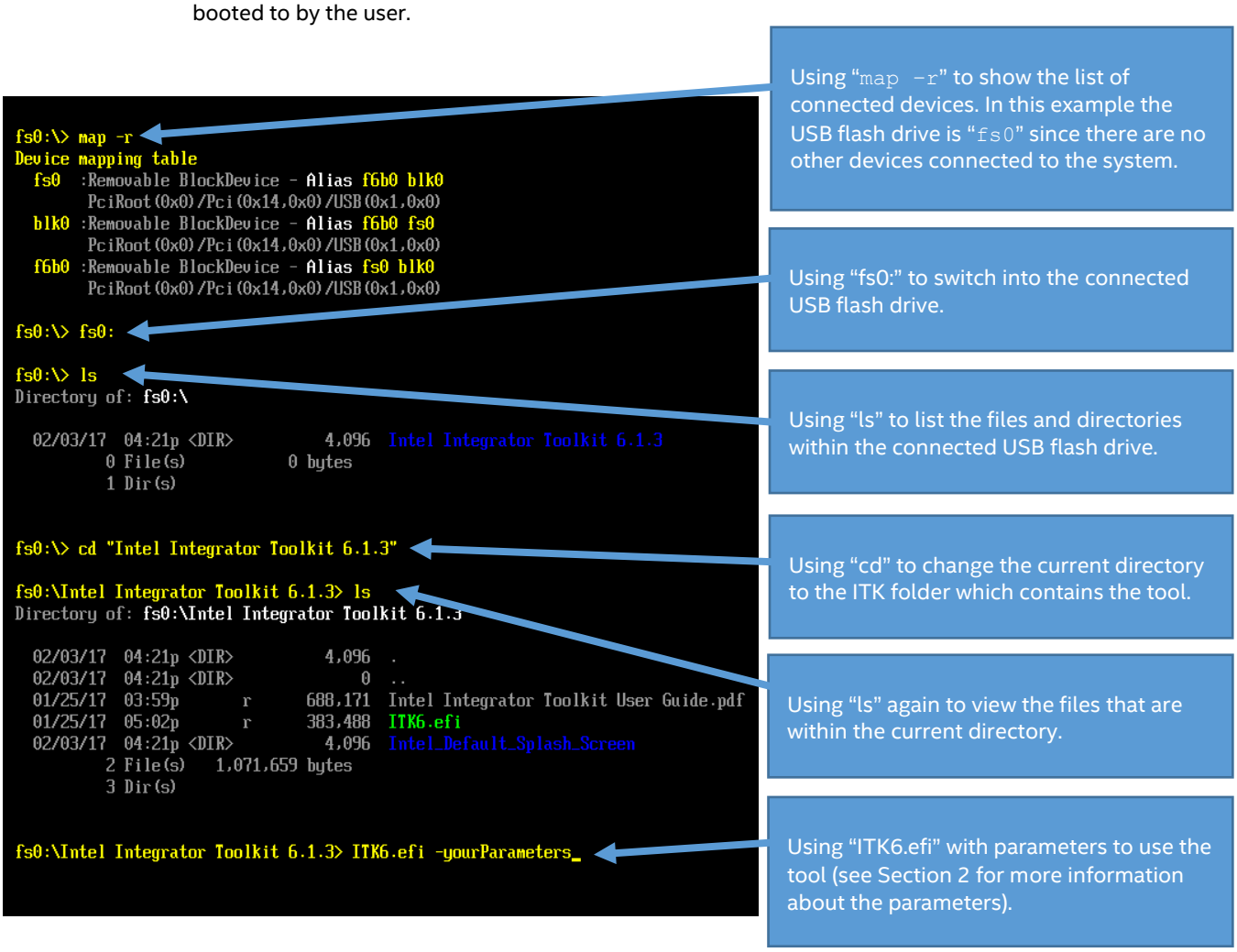
This section provides a basic guide on how to use the internal UEFI shell to run the Intel® Integrator Toolkit. To scroll up or down within the shell, use the "Page Up" or "Page Down" keys. Also, note that the UEFI shell has tab completion capabilities. More information about how to use the EFI shell (including scripting) can be found at: https://software.intel.com/en-us/articles/efi-shells-and-scripting.

Below are some useful commands that can be used to run ITK, as well as a visual walkthrough of some of the commands in a common use case:

- "`map -r`" refreshes the list of devices that are connected to the computer
    - If a USB drive with the `ITK6.efi` was plugged in after the computer booted, this command must be used in order to access that USB drive

- "`fsX:`" changes the current directory to the specified device "`X`"
  - Depending on how many devices are connected to the computer, the "`X`" that corresponds to the USB drive can vary
  - Generally the last "`fsX`" output from the "`map -r`" command will correspond to the USB drive
  - For a USB drive, "`fsX`" should correspond to a "`Removable HardDisk`" which can help discover which device needs to be accessed
- "`ls`" lists the contents of the current directory that you are within
  - This is helpful when trying to locate the `ITK6.efi` file or any other files that are being worked with
- "`cls`" clears the screen
  - This can be useful to speed up text output to the screen on some products

This visual walkthrough assumes that the downloaded ITK folder (unzipped) has been placed onto a flash drive (that has been inserted into the system) and the internal UEFI shell has been booted to by the user.



2 Features and Examples

Intel® Integrator Toolkit contains the following features:

1. System Management BIOS (SMBIOS) Configuration
2. OEM Windows Product Key Injection (OEM Activation 3.0)
3. Custom BIOS Update File Creation with Imported Features (see below)
    a. An existing BIOS update file (.BIO)
    b. The current SMBIOS configuration
    c. The current OEM Windows product key data
    d. The current BIOS custom settings
    e. An imported logo file (.JPG)

A detailed explanation of each feature, as well as examples of how the features are invoked from the command line, can be found in the corresponding section.

Note: Section 3.2 provides a table with the current validated features for various Intel® NUC or Intel® Compute Stick products.

2.1 System Management BIOS (SMBIOS) Configuration

This feature is used to change system identification information that is stored within the BIOS. This includes fields within the System Information and Chassis Information sections within the BIOS as well as several OEM-specific strings.

Note: These SMBIOS configuration settings can be integrated en masse using a custom BIOS flash update file (see Section 2.3).

The below subsections show how an SMBIOS value can be set, deleted or printed. Section 2.1.3 provides a visual walkthrough for a possible use case.

2.1.1 Setting or Deleting an SMBIOS Value

The general syntaxes for setting or deleting an SMBIOS value:

```
ITK6.efi –s –t system –f manufacturer –v myManufacturer

ITK6.efi –s –t system –f manufacturer –d
```

Where:

- "`-s`" is the flag for the SMBIOS Configuration feature
- "`-t`" is the flag for choosing which type of section the setting is within
    - A value of "`system`" is for "System Information"
    - A value of "`chassis`" is for "Chassis Information"
    - A value of "`oem`" is for "OEM-specific Information"

- "`-f`" is the flag for choosing which field the setting is
  - This can be one of many different options depending on the type (more information can be found below in *Table 2.1.1*)
- "`-v`" is the flag for choosing the value for the setting
  - This can either be a user-entered non-empty string of characters, or a number, depending on the field (more information can be found below in *Table 2.1.1*)
  - In order to enter in values with reserved UEFI characters, the input can be surrounded with quotes. For example: `<test>` must be entered as "`<test>`"
  - **The length of the value cannot exceed 40 characters**
- "`-d`" is the flag for deleting an SMBIOS value
  - This flag is used in place of the "`-v`" flag
  - **This sets the specified SMBIOS value to an empty value**

Field options are dependent on the type that has been selected. Also, the value that can be entered depends upon the field option specified. *Table 2.1.1* has been constructed to provide a list of which fields can be changed within each type of section and examples of the command.

Table 2.1.1: SMBIOS Configuration Options with Examples

Field	Types*	Example Command
Manufacturer	`system` and `chassis`	`ITK6.efi -s -t system -f manufacturer -v myMan`
Manufacturer**	`system` and `chassis`	`ITK6.efi -s -t system -f manufacturer -d`
Product Name	`system`	`ITK6.efi -s -t system -f product -v myProduct`
Version	`system` and `chassis`	`ITK6.efi -s -t system -f version -v myVersion`
Serial Number	`system` and `chassis`	`ITK6.efi -s -t system -f serial -v mySerial`
SKU Number	`system`	`ITK6.efi -s -t system -f sku -v mySKU`
Family	`system`	`ITK6.efi -s -t system -f family -v myFamily`
Asset Tag	`chassis`	`ITK6.efi -s -t chassis -f asset -v myAsset`
Chassis Type***	`chassis`	`ITK6.efi -s -t chassis -f type -v 3`
OEM String 1	`oem`	`ITK6.efi -s -t oem -f oem1 -v myString`
OEM String 2	`oem`	`ITK6.efi -s -t oem -f oem2 -v myString`
OEM String 3	`oem`	`ITK6.efi -s -t oem -f oem3 -v myString`

*This column represents the types that are compatible with the corresponding field. Any row that has more than one compatible type can have the corresponding type changed in the example command (e.g. "`-t system`" could be "`-t chassis`" for the first row). Note that type `system` corresponds to "System Information," type `chassis` corresponds to "Chassis Information," and type `oem` corresponds to "OEM-specific Information."

**This example includes the "-d" flag which signifies that the specified SMBIOS value should be deleted.

***Chassis Type must be a number between 1 and 29 which corresponds to a certain pre-defined type as defined by the ACPI spec. A full list of what each number represents can be found in Section 3's *Table 3.1*.

2.1.2 Printing the Current SMBIOS Configuration

The current SMBIOS configuration can be printed by using the following syntax:

```
ITK6.efi -s -p
```

Where:

- "`-s`" is the flag for the SMBIOS Configuration feature
- "`-p`" is the flag for printing the current SMBIOS configuration

A list of the SMBIOS values will then be printed to the screen. If a text value has not been set it will be blank by default. If the SMBIOS value is a numbered value (e.g. the chassis type) it will be a default value depending on the product (e.g. "`3`"). See Table 3.1 for a full list of the meanings for each chassis type value.

2.1.3 Visual Example of Usages

This section provides a visual example of how to use the SMBIOS Configuration feature. This includes: setting, printing, and deleting values of the current SMBIOS configuration.

2.2 OEM Windows Product Key Injection (OEM Activation 3.0)

This feature is used to inject a Windows product key into the BIOS for OEM activation. This tool supports the OEM Activation 3.0 (OA 3.0) process as defined by Microsoft*. The data that this feature injects will also be referred to as OA3 data or the Microsoft Data Management (MSDM) table in either this document or the tool.

The OEM activation step that requires this BIOS injection tool can be found online (https://technet.microsoft.com/en-us/library/dn621894.aspx). The required OEM product key files (.BIN) that this tool injects can be obtained from Microsoft.

Note: This OEM product key data can be integrated en masse using a custom BIOS flash update file (see Section 2.3).

The below subsections give guides on how to set the current OEM product key data, delete it, or print it. Section 2.2.4 provides a visual walkthrough for a possible use case.

2.2.1 Setting the Current OEM Windows Product Key Data

Note: This tool will overwrite any existing OEM product key that is on the device.

The general syntax for setting the current OEM Windows product key data/ OA3 data is:

```
ITK6.efi –o –oid oemID –tid tableID –i myKey.bin
```

Where:

- "`-o`" is the flag for the OEM Windows Product Key Injection feature
- "`-oid`" is the flag for the user's input of the OEM ID that will be associated with the key
  - The OEM ID cannot be blank and can be at most six characters
- "`-tid`" is the flag for the user's input of the Table ID that will be associated with the key
  - The Table ID cannot be blank and can be at most eight characters
- "`-i`" is the flag for choosing the OEM product key file to inject
  - An OEM product key file (.BIN) can be obtained from Microsoft
  - **The OEM product key file must be exactly 49 bytes in size**

2.2.2 Deleting the Current OEM Windows Product Key Data

Note: Once the OEM Windows product key data has been deleted it cannot be recovered.

The syntax for deleting the current OEM Windows product key data (OA3 data) is:

```
ITK6.efi -o -d
```

Where:

- "`-o`" is the flag for the OEM Windows Product Key Injection feature
- "`-d`" is the flag for deleting the current OEM Windows product key data

2.2.3 Printing the Current OEM Windows Product Key Data

The syntax for printing the current OEM Windows product key data (OA3 data) is:

```
ITK6.efi -o -p
```

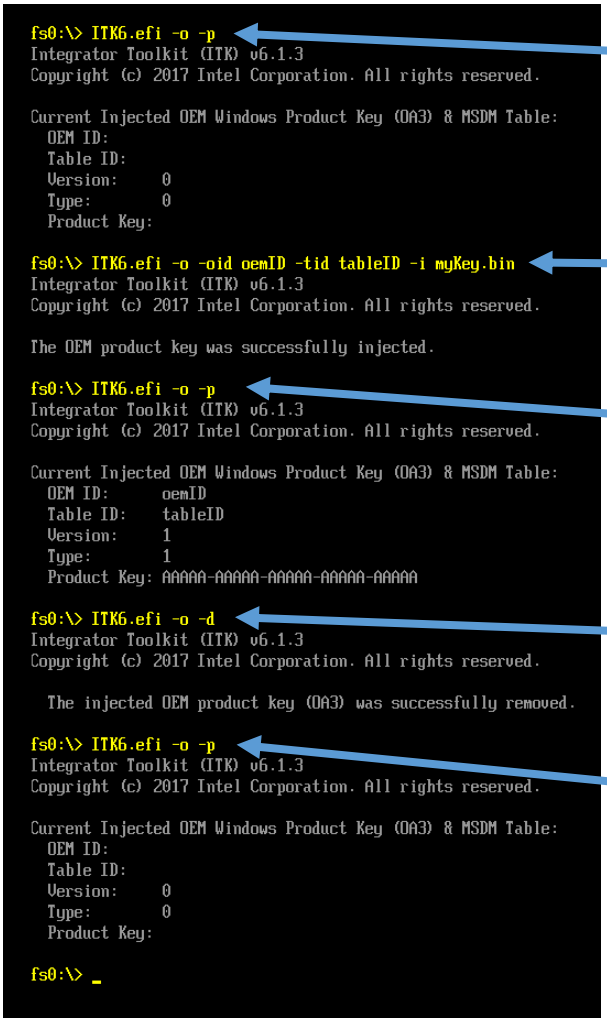
Where:

- "`-o`" is the flag for the OEM Windows Product Key Injection feature
- "`-p`" is the flag for printing the current OEM Windows product key data

Using this feature will print the OEM ID, table ID, version, type, and product key.

2.2.4 Visual Example of Usages

This section provides a visual example of how to use the OEM Windows Product Key Injection feature. This includes: setting, printing, and deleting the current OEM Windows product key data.



Printing the current OEM Windows product key data which is currently empty. (Section 2.2.3)

Setting the current OEM Windows product key data with our OEM/ Table ID as well as our OEM product key file (.BIN). (Section 2.2.1)

Printing the current OEM Windows product key data again to verify our changes. (Section 2.2.3)

Deleting the current OEM Windows product key data with the "-d" flag. (Section 2.2.2)

Printing the current OEM Windows product key data again to verify our changes. (Section 2.2.3)

2.3 Custom BIOS Update File Creation with Imported Features

Note: The custom BIOS creation feature is not supported for Intel® Compute Stick STK1A[x]32SC or Intel® Compute Stick STCK1A[x]FC products.

Note: Portions of this feature are not supported for certain products. See Table 3.2 (Section 3.2) which provides a table with the current features that are supported for various Intel® NUC or Intel® Compute Stick products.

This feature is used to create a custom BIOS flash update file (.BIO) with various imported features. The following subsections will go over what can be added to a custom BIOS update file, examples of how each feature is imported, and some tips to ensure correct functionality. Also, there will be a subsection that gives a step-by-step example of how to create a custom BIOS update file from an existing BIOS update file along with four imported features.

2.3.1 Importing Features: Explanations, Examples, and Tips

Below is an overview of what can be added to the custom BIOS update file. The following subsections will provide a more detailed overview of each feature that can be imported, as well as some notes about possible restrictions. See Section 2.3.2 for a textual and visual walkthrough of a possible use case for importing a download .BIO as well as importing four features to a custom BIOS file.

Initially, a custom BIOS update file can contain one of the below choices (1-2) below:

1. An existing BIOS update file
    a. This updates the BIOS major revision as well as applying the imported features
    b. A BIOS update file can be found at http://downloadcenter.intel.com
2. No existing BIOS update file
    a. Applies just the imported features (does not update the BIOS major revision)
    b. This provides a faster update process

Once it is decided whether an existing BIOS update file will be used or not, the created custom BIOS update file **must have at least one** of the following features (1-4) added to it:

1. The current SMBIOS configuration
2. The current OEM Windows product key data (OA3 data)
3. The current BIOS custom settings
4. An imported logo/ image/ splash screen file (.JPG)

2.3.1.1 Importing an Existing BIOS Update File

Using the flag "`-ib`" will import a downloaded BIOS update file into your custom BIOS update file. Having an imported BIOS update file allows the ability to update the BIOS major revision as well as importing at least one of the features in the below subsections. If there is no imported BIOS update file then the custom BIOS update file will not update the BIOS major revision. The imported BIOS update file should be a .BIO file that corresponds to the product that the custom BIOS update file is being created for. For example, if a user is creating a custom BIOS update file for an Intel® NUC Kit NUC6i5SY[x] then the imported BIOS update file should be for Intel® NUC

Kit NUC6i5SY[x]. The latest BIOS update files for all products can be downloaded at http://downloadcenter.intel.com.

An example of the syntax for creating a custom .BIO which imports a downloaded BIOS update file can be found below:

```
ITK6.efi –b –x myCustomBio.bio –ib DownloadedBio.bio {other}
```

Where:

- "`-b`" is the flag for the BIOS Update File Creation feature
- "`-x`" is the flag for choosing the name of the BIOS update file that will be exported
  - If the current directory already has a file with that filename, there is a choice to either overwrite the existing file or exit the program
  - Note that "`.bio`" must be appended to the end of the filename
- "`-ib`" is the flag for choosing the existing BIOS update file's data that will be imported
  - This flag is optional depending on if the user desires that the target's BIOS major revision will be updated by the custom BIOS update file
  - The imported .BIO file should be the latest BIOS that is available for the product at http://downloadcenter.intel.com
  - Note that "`.bio`" must be appended to the end of the filename
- "`{other}`" is meant to represent the other parameters that the user might add
  - **Note that at least one other feature must be imported into the BIOS update file (see the following sections for the other features that can be added)**

2.3.1.2 Importing the Current SMBIOS Configuration

Using the flag "`-is`" will import the current SMBIOS configuration of the system into the newly created custom BIOS update file. See Section 2.1 for more info on how to configure the current SMBIOS configuration. The flag for this import feature can be combined with others to import multiple features at the same time into a BIOS update file (see Section 2.3.2 for an example). This custom BIOS update file can then be used to update another system to apply the imported SMBIOS configuration.

An example of the syntax for creating a custom .BIO which imports the current SMBIOS configuration can be found below:

```
ITK6.efi –b –x myCustomBio.bio –is
```

Where:

- "`-b`" is the flag for the BIOS Update File Creation feature
- "`-x`" is the flag for choosing the name of the BIOS update file that will be exported
  - If the current directory already has a file with that filename, there is a choice to either overwrite the existing file or exit the program
  - Note that "`.bio`" must be appended to the end of the filename
- "`-is`" is the flag for choosing if the current SMBIOS configuration should be imported
  - To modify the current SMBIOS configuration, see Section 2.1

2.3.1.3 Importing the Current OEM Windows Product Key Data

Using the flag "`-io`" will import the current OEM Windows product key data (OA3.0 data) of the system into the newly created BIOS update file. See Section 2.2 for more info on how to change the current OEM Windows product key data. The flag for this import feature can be combined with others to import multiple features at the same time into a BIOS update file (see Section 2.3.2 for an example). This custom BIOS update file can then be used to update another system to apply the import OEM Windows product key data.

An example of the syntax for creating a custom .BIO which imports the current OEM Windows product key data can be found below:

```
ITK6.efi –b –x myCustomBio.bio –io
```

Where:

- "`-b`" is the flag for the BIOS Update File Creation feature
- "`-x`" is the flag for choosing the name of the BIOS update file that will be exported
  - If the current directory already has a file with that filename, there is a choice to either overwrite the existing file or exit the program
  - Note that "`.bio`" must be appended to the end of the filename
- "`-io`" is the flag for choosing if the current OEM product key data should be imported
  - Ensure that the current OEM product key data is correct before importing it
  - To modify the current OEM product key data, see Section 2.2

2.3.1.4 Importing the Current Custom Settings Configuration

Using the flag "`-ic`" will import the current BIOS custom settings configuration of the system into the newly created BIOS update file. The flag for this import feature can be combined with others to import multiple features at the same time into a BIOS update file (see Section 2.3.2 for an example). This custom BIOS update file can then be used to update another system to apply the BIOS custom settings configuration.

An example of the syntax for creating a custom .BIO which imports the current BIOS custom settings configuration can be found below:

```
ITK6.efi –b –x myCustomBio.bio –ic
```

Where:

- "`-b`" is the flag for the BIOS Update File Creation feature
- "`-x`" is the flag for choosing the name of the BIOS update file that will be exported
  - If the current directory already has a file with that filename, there is a choice to either overwrite the existing file or exit the program
  - Note that "`.bio`" must be appended to the end of the filename
- "`-ic`" is the flag for choosing if the current BIOS custom settings should be imported
  - **See the two bolded notes below about product compatibility information as well as BIOS custom settings limitations**
  - Ensure the current BIOS custom settings are correct before importing them
  - To modify the current BIOS custom settings, press `F2` while booting the unit

Note: A BIOS update file with BIOS custom settings can only be used to update the same product. For example, if the BIOS update file is created from an Intel® NUC Kit NUC6i5SY that file should only be used to update another Intel® NUC Kit NUC6i5SY. Doing otherwise can cause your system to become inoperable.

Note: Not all custom settings can be altered by the custom BIOS update file. For example, the password settings and boot order will not be updated. Additionally, many generations of products have restricted settings that cannot be altered through the BIOS update file process (e.g. some CPU settings). It is always best practice to verify that the settings are being updated correctly on a test unit before mass integration.

2.3.1.5 Importing a Custom Logo/ Image/ Splash Screen File (.JPG)

Using the flag "`-il`" will import a custom logo/ image/ splash screen from a file (.JPG) into the newly created BIOS update file. The flag for this import feature can be combined with others to import multiple features at the same time into a BIOS update file (see Section 2.3.2 for an example). This custom BIOS update file can then be used to update another system to modify the splash screen with the imported image file (.JPG).

Note: Some image editing programs can produce an incompatible .JPEG file. A best known method to fix this issue is to reopen the image with Microsoft Paint and then save the image.

An example of the syntax for creating a custom .BIO which imports the custom logo/ image/ splash screen file (.JPG) can be found below:

```
ITK6.efi –b –x myCustomBio.bio –il myLogo.jpg
```

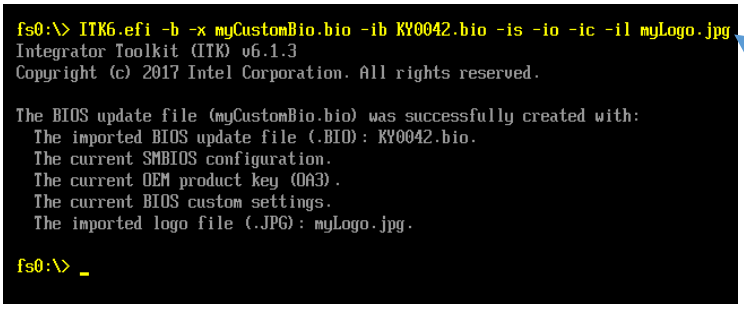
Where:

- "`-b`" is the flag for the BIOS Update File Creation feature
- "`-x`" is the flag for choosing the name of the BIOS update file that will be exported
  - If the current directory already has a file with that filename, there is a choice to either overwrite the existing file or exit the program
  - Note that "`.bio`" must be appended to the end of the filename
- "`-il`" is the flag for choosing the logo file (.JPG) to import
  - **The max size of the file (.JPG) is 60KB**
  - **The minimum resolution is 120 x 120 pixels**
  - **The maximum resolution is 1920 x 1080 pixels**
  - Note that "`.jpg`" must be appended to the end of the filename
  - **In order to revert back to the default logo of your product use one of the included images within the Intel_Default_Splash_Screen folder (e.g. "`-il Intel_Default_Splash_Screen\NUC\Intel_NUC_Default_512x384.jpg`")**

2.3.2 Importing Features: Step-by-step Example

The ideal steps to create a BIOS update file from an existing BIOS update file as well importing all four features, as seen in Section 2.3.1, can be found below with a visual walkthrough (a similar strategy can be used for other configurations albeit with some modified steps):

1. Update the product with the latest BIOS from http://downloadcenter.intel.com
2. Place the downloaded BIOS update file (.BIO) and the logo file (.JPG) onto the USB drive that contains the `ITK6.efi` file
    a. These will imported by the tool in a later step
3. Load the default BIOS settings
    a. Boot into the BIOS (`F2`) and then hit `F9` to load defaults
4. Set the BIOS custom settings to the desired configuration
5. Save the BIOS custom settings (`F10`) and then reboot
6. Verify that the settings are the desired configuration
7. Reboot to the internal UEFI shell
8. Use the tool to set the SMBIOS configuration (Section 2.1)
    a. Verify that the SMBIOS is configured correctly with the print option
9. Use the tool to set the OEM Windows product key data (Section 2.2)
    a. Verify that the OEM product key is configured correctly with the print option
10. Use the tool to create the custom BIOS update file from the existing .BIO file as well as all four imported features with the following syntax:
    ```
    ITK6.efi -b -x myCustomBio.bio -ib DownloadedBio.bio -is -io -ic
             -il myLogo.jpg
    ```
    a. Note that DownloadedBio.bio and myLogo.jpg were acquired in Step 2.
11. Update a different unit with the file
    a. Ensure that the different unit has a different configuration (e.g. BIOS version, SMBIOS, OEM product key data, BIOS custom settings, and logo)
    b. See the first note in Section 2.3.1.4 about product compatibility
12. Verify that the BIOS version, SMBIOS, OEM product key data, BIOS custom settings, and the logo were updated correctly
    a. See the second note in Section 2.3.1.4 about BIOS setting restrictions
13. Now that the custom BIOS update file has been tested it can now be used for mass integration



Creating a custom BIOS update file (`-x myCustomBio.bio`) from an imported BIOS (`-ib KY0042.bio`) that includes the imported features of: the current SMBIOS configuration (`-is`), the current OEM product key data (`-io`), the current BIOS custom settings configuration (`-ic`), and an imported logo (`-il myLogo.jpg`).

(Section 2.3.1)

3 Appendix

3.1 SMBIOS Chassis Type: Values and Meanings

The table below contains translations of chassis type values to their meanings as per the ACPI spec (see Section 2.1).

Table 3.1: Chassis Type: Values and Meanings

Value	Meaning	Value	Meaning
1	None/ Other	**16**	Lunch Box
2	Unknown	**17**	Main Server Chassis
3	Desktop	**18**	Expansion Chassis
4	Low-Profile Desktop	**19**	Sub Chassis
5	Pizza Box	**20**	Expansion Chassis
6	Mini Tower	**21**	Peripheral Chassis
7	Tower	**22**	RAID Chassis
8	Portable	**23**	Rack Mount Chassis
9	Laptop	**24**	Sealed-Case PC
10	Notebook	**25**	Multi-System Chassis
11	Handheld	**26**	Compact PCI
12	Docking Station	**27**	Advanced TCA
13	All-in-one	**28**	Blade
14	Sub-Notebook	**29**	Blade Enclosure
15	Space Saving		

3.2 Validated Products and Features List

Table 3.2 below provides a list of features that are tested to be working for various Intel® NUC or Intel® Compute Stick products. The rows represent the product that was tested, and the columns represent if that feature works for that product. The features include modifying the SMBIOS configuration, modifying the OEM Windows product key data (OA3 data), creating a custom .BIO for updating, and the four features that can be added to a custom .BIO for updating. For example, an Intel® NUC Kit NUC6i7KYK has full functionality, but an Intel® NUC Kit DE3815TYKHE does not have the ability to update a logo from a created custom BIOS update file (.BIO).

Note: Only use features that are tested to be working for your product. Using features that are not tested to be working correctly for a product has the possibility of making your device inoperable.

Note: The custom BIOS creation feature is not supported for Intel® Compute Stick STK1A[x]32SC or Intel® Compute Stick STCK1A[x]FC products.

Table 3.2: Validated Products and Features List

Product Name	Edit SMBIOS Data	Edit OA3 Data	Create Custom .BIO	Add SMBIOS Data	Add OA3 Data	Add Custom Settings	Add Logo File
NUC7i3BN[x]	✓	✓	✓	✓	✓	✓	✓
NUC6CAY[x]	✓	✓	✓	✓	✓	✓	✓
NUC6i7KYK	✓	✓	✓	✓	✓	✓	✓
NUC6i[x]SY[x]	✓	✓	✓	✓	✓	✓	✓
NUC5i[x]RY[x]	✓	✓	✓	✓	✓	✓	✓
NUC5i[x]MY[x]E	✓	✓	✓	✓	✓	✓	✓
NUC5PGYH	✓	✓	✓	✓	✓	✓	✓
NUC5[x]PYH	✓	✓	✓	✓	✓	✓	✓
DE3815TYKHE	✓	✓	✓	✓	✓	✓	×
D[x]0WY[x]	✓	✓	✓	✓	✓	✓	✓
DN2820FY[x]	✓	✓	✓	✓	✓	✓	✓
DC53427HYE*	✓	✓	✓	✓	✓	×	×
DC3217IYE*	✓	✓	✓	✓	✓	×	×
STK2M[x]64CC	✓	✓	✓	✓	✓	×	✓
STK1A[x]32SC	✓	✓	×	×	×	×	×
STCK1A[x]FC	✓	✓	×	×	×	×	×

*This product requires using an EFI shell from a bootable USB drive.