Name

GenCfgOpt.py – The python script that generates UPD txt files for the compiler, header files for the VPD and UPD regions, and generates a Boot Settings File (BSF), all from an EDK II Platform Description (DSC) file.

Synopsis

```
GenCfgOpt UPDTXT PlatformDscFile BuildFvDir [TxtOutFile] [-D Macros]
GenCfgOpt HEADER PlatformDscFile BuildFvDir [InputHFile] [-D Macros]
GenCfgOpt GENBSF PlatformDscFile BuildFvDir BsfOutFile [-D Macros]
```

Description

GenCfgOpt.py is a script that generates configuration options from an EDK II Platform Description (DSC) file. It has three functions. First, it produces a txt file that is used by the compiler that summarizes the UPD section in the DSC file. Second, it generates header files for the UPD and VPD regions. Third, it generates a Boot Settings File (BSF) that can be used by the Intel-provided Binary Configuration Tool (BCT) to provide a graphical user interface for manipulating settings in the UPD and VPD regions.

The GenCfgOpt.py script generates important files that are vital parts of your build process. The UPDTXT and HEADER use cases must be done before the 'build' command; the GENBSF use case may be done at any time. The following sections explain the three use cases.

GenCfgOpt.py UPDTXT

The UPDTXT option creates a text file with all the UPD entries, offsets, size in bytes, and values. GenCfgOpt reads this information from the [PcdsDynamicVpd.Upd] section of the project's DSC file. The DSC file allows you to specify offsets and sizes for each entry, opening up the possibility of introducing gaps between entries. GenCfgOpt fills in these gaps with UPD entries that have the generic names "UnusedUpdSpaceN" where N begins with 0 and increments. The command signature for UPDTXT is:

```
GenCfgOpt UPDTXT PlatformDscFile BuildFvDir [TxtOutFile] [-D Macros]
```

`PlatformDscFile` must be the location of the DSC file for the platform you're building. `BuildFvDir` is the location where the binary will be stored. The optional `TxtOutFile` is a name and location for the output of GenCfgOpt. The default name and location is the <UPD_TOOL_GUID>.txt in the directory specified by `BuildFvDir`. The macro UPD_TOOL_GUID must be defined in the

DSC file or in the optional `Macros` arguments. Each optional macro argument must follow the form "–D <MACRO_NAME>=<VALUE>".

GenCfgOpt checks to see if the UPD txt file has already been created and will only re-create it if the DSC was modified after it was created.

GenCfgOpt.py HEADER

The HEADER option creates two header files in the build folder. Both header files define the _UPD_DATA_REGION and _VPD_DATA_REGION data structs. The first header file is FspUpdVpdInternal.h; the second is FspUpdVpd.h. The first is intended for Intel-internal purposes only, as it names specific variables that Intel may wish to keep from outside customers. The second file, FspUpdVpd.h, labels these variables as "ReservedUpdSpaceN" beginning with N=0. The command signature for HEADER is

```
GenCfgOpt HEADER PlatformDscFile BuildFvDir [InputHFile] [-D Macros]
```

PlatformDscFile and BuildFvDir are described in the previous section. The optional InputHFile is a header file that may contain data definitions that are used by variables in the UPD and VPD regions. This header file must contain the special keywords '!EXPORT EXTERNAL_BOOTLOADER_STRUCT_BEGIN' and '!EXPORT EXTERNAL_BOOTLOADER_STRUCT_END' in comments. Everything between these two keywords will be included in the generated header file.

The mechanism to specify whether a variable appears as "ReservedUpdSpaceN" in the fsp_vpd.h header file is in special commands that appear in the comments of the DSC file. The special commands begin with '!HDR', for header. The following table summarizes the two command options.

HEADER	Use the HEADER command to hide specific variables in the public header file. In your project DSC file, use `!HDR HEADER:{OFF}` at the beginning of the section you wish to hide and `!HDR HEADER:{ON}` at the end.
STRUCT	The STRUCT command allows you to specify a specific data type for a variable. You can specify a pointer to a data struct, for example. You define the data structure in the 'InputHFile' between '!EXPORT EXTERNAL_BOOTLOADER_STRUCT_BEGIN' and '!EXPORT EXTERNAL_BOOTLOADER_STRUCT_END'. Example: `!HDR STRUCT:{MY_DATA_STRUCT*}` You then define MY_DATA_STRUCT in InputHFile.
EMBED	The EMBED command allows you to put one or more UPD data into a specify data structure. You can utilize it as a group of UPD for

example. You must specify a start and an end for the specify data structure.

Example: `!HDR EMBED:{MY_DATA_STRUCT:MyDataStructure:START}`

```
gTokenSpaceGuid.Upd1  | 0x0020 | 0x01 | 0x00
gTokenSpaceGuid.Upd2  | 0x0021 | 0x01 | 0x00
!HDR EMBED:{MY_DATA_STRUCT:MyDataStructure:END}
gTokenSpaceGuid.UpdN  | 0x0022 | 0x01 | 0x00
```

Result:

```
typedef struct {
/** Offset 0x0020
**/
UINT8                   Upd1;
/** Offset 0x0021
**/
UINT8                   Upd2;
/** Offset 0x0022
**/
UINT8                   UpdN;
}  MY_DATA_STRUCT;

typedef struct _UPD_DATA_REGION {
…
/** Offset 0x0020
**/
MY_DATA_STRUCT    MyDataStruct;
…
} UPD_DATA_REGION;
```

GenCfgOpt .py GENBSF

The GENBSF option generates a BSF from the VPD and UPD entries in a package’s DSC file. It does this by parsing special commands found in the comments of the DSC file. They roughly match the keywords that define the different sections of the BSF. The command signature for GENBSF is

```
GenCfgOpt GENBSF PlatformDscFile BuildFvDir BsfOutFile [-D Macros]
```

In this case, the BsfOutFile parameter is required; it should be the relative path to where the BSF should be stored.

Every BSF command in the DSC file begins with ‘!BSF’. The following table summarizes the options that come after ‘!BSF’:

BSF Command	**Description**
PAGES	PAGES maps abbreviations to friendly-text descriptions of the pages in a BSF. Example: `!BSF PAGES:{PG1:”Page 1”, PG2:”Page 2”}`
PAGE	This marks the beginning of a page. Use the abbreviation specified in PAGES command. Example: `!BSF PAGE:{PG1}` All the entries that come after this command are assumed to be on that page, until the next PAGE command
FIND	FIND maps to the BSF ‘Find’ command. It will be placed in the StructDef region of the BSF and should come at the beginning of the UPD and VPD sections of the DSC, immediately before the signatures that mark the beginning of these sections. The content should be the plain-text equivalent of the signature. The signature is usually 8 characters. Example: `!BSF FIND:{PROJSIG1}`
BLOCK	The BLOCK command maps to the BeginInfoBlock section of the BSF. There are two elements: a version number and a plain-text description. Example: `!BSF BLOCK:{NAME:"My platform name", VER:"0.1"}`
NAME	NAME gives a plain-text for a variable. This is the text label that will appear next to the control in BCT. Example: `!BSF NAME:{Variable 0}` If the ‘!BSF NAME’ command does not appear before an entry in the UPD or VPD region of the DSC file, then that entry will not appear in

	the BSF.
TYPE	The TYPE command is used either by itself or with the NAME command. It is usually used by itself when defining an EditNum field for the BSF. You specify the type of data in the second parameter and the range of valid values in the third. Example: `!BSF TYPE:{EditNum, HEX, (0x00,0xFF)}` TYPE appears on the same line as the NAME command when using a combo-box. Example: `!BSF NAME:{Variable 1} TYPE:{Combo}` There is a special ‘None’ type that puts the variable in the StructDef region of the BSF, but doesn’t put it in any Page section. This makes the variable visible to BCT, but not to the end user.
HELP	The HELP command defines what will appear in the help text for each control in BCT. Example: `!BSF HELP:{Enable/disable LAN controller.}`
OPTION	The OPTION command allows you to custom-define combo boxes and map integer or hex values to friendly-text options. Example: `!BSF OPTION:{0:IDE, 1:AHCI, 2:RAID}` Example: `!BSF OPTION:{0x00:0 MB, 0x01:32 MB, 0x02:64 MB}`
FIELD	The FIELD command can be used to define a section of a consolidated PCD such that the PCD will be displayed in several fields via BCT interface instead of one long entry. Example: `!BSF FIELD:{PcdDRAMSpeed:1}`
ORDER	The ORDER command can be used to adjust the display order for the BSF items. By default the order value for a BSF item is assigned to be the UPD/VPD item (Offset * 256). It can be overridden by declaring ORDER command using format ORDER: {HexMajor.HexMinor}. In this case the order value will be (HexMajor*256+HexMinor). The item order value will be used as the sort key during the BSF item display. Example: `!BSF ORDER:{0x0040.01}`

For OPTION and HELP commands, it allows to split the contents into multiple lines by adding multiple OPTION and HELP command lines. The lines except for the very first line need to start with ‘+’ in the content to tell the tool to append this string to the previous one.

For example, the statement

`!BSF OPTION:{0x00:0 MB, 0x01:32 MB, 0x02:64 MB}`

`is equivalent to:`

`!BSF OPTION:{0x00:0 MB, 0x01:32 MB,}`

`!BSF OPTION:{+ 0x02:64 MB}`

The NAME, OPTION, TYPE, and HELP commands can all appear on the same line following the ‘!BSF’ keyword or they may appear on separate lines to improve readability.

License