# congatec Application Note #34

| | |
|---|---|
| **Affected Products** | Products featuring AptioV UEFI Firmware |
| **Subject** | OEM SMBIOS/DMI tables |
| **Confidential/Public** | public |
| **Author** | HHA |

# Revision History

| Revision | Date (yyyy-mm-dd) | Author | Changes |
|---|---|---|---|
| 1.0 | 2016-10-12 | HHA | Initial release of document |
| | | | |
| | | | |

# Preface

The System Management BIOS (SMBIOS) specification defines data structures, known as DMI tables, (and the access methods) that can be used to read information stored in the BIOS of a computer. On some congatec products, the user (or system designer) can now easily change the contents of the most important SMBIOS data structures. This document explains the underlying mechanisms and how to apply the changes.

## Disclaimer

The information contained within this Application Note, including but not limited to any product specification, is subject to change without notice.

congatec AG provides no warranty with regard to this Application Note or any other information contained herein and hereby expressly disclaims any implied warranties of merchantability or fitness for any particular purpose with regard to any of the foregoing. congatec AG assumes no liability for any damages incurred directly or indirectly from any technical or typographical errors or omissions contained herein or for discrepancies between the product and the Application Note. In no event shall congatec AG be liable for any incidental, consequential, special, or exemplary damages, whether based on tort, contract or otherwise, arising out of or in connection with this Application Note or any other information contained herein or the use thereof.

## Intended Audience

This Application Note is intended for technically qualified personnel. It is not intended for general audiences.

## Electrostatic Sensitive Device

All congatec AG products are electrostatic sensitive devices and are packaged accordingly. Do not open or handle a congatec AG product except at an electrostatic-free workstation. Additionally, do not ship or store congatec AG products near strong electrostatic, electromagnetic, magnetic, or radioactive fields unless the device is contained within its original manufacturer's packaging. Be aware that failure to comply with these guidelines will void the congatec AG Limited Warranty.

## Technical Support

congatec AG technicians and engineers are committed to providing the best possible technical support for our customers so that our products can be easily used and implemented. We request that you first visit our website at www.congatec.com for the latest documentation, utilities and drivers, which have been made available to assist you. If you still require assistance after visiting our website then contact our technical support department by email at support@congatec.com

## Symbols

The following are symbols used in this application note.

**Note**

***Notes call attention to important information that should be observed.***

**Caution**

***Cautions warn the user about how to prevent damage to hardware or loss of data.***

**Warning**

***Warnings indicate that personal injury can occur if the information is not observed.***

## Copyright Notice



## Trademarks

Product names, logos, brands, and other trademarks featured or referred to within this user’s guide, or the congatec website, are the property of their respective trademark holders. These trademark holders are not affiliated with congatec AG, our products, or our website.

## Terminology

| Term | Description |
|---|---|
| UEFI | Unified Extensible Firmware Interface |
| AMI | American Megatrends, Inc - congatec’s BIOS partner |
| Aptio V | AMIs next generation UEFI BIOS |
| SMBIOS | System Management BIOS |
| DMI | Desktop Management Information |

# Contents

# 1 Introduction

The congatec BIOS contains several hard coded DMI tables. In detail, these DMI tables are not simple tables with key-value pairs for each entry. They are compound data structures containing binary, numerical and text string members. The SMBIOS specification explains the details of each type of structure. The Linux tool dmidecode(8) can show the contents of these tables both in an interpreted way or showing the raw data structures. In Windows, the tool RWEverything can do the same. The way how a particular operating system reads and utilizes the tables is beyond the scope of this document.

Newer congatec BIOS releases offer a new OEM feature that allows the content of these tables to be changed by the system integrator. It is possible to overwrite some members of these tables. The modification data is contained in an OEM data module that can be created with the CGUTIL utility from a structured text file. An example text with all currently changeable elements is shown in the appendix. Note that not all BIOS versions support all of the elements in the sample text or allow the ability to modify all of those elements.

# 2 Requirements

OEM SMBIOS DMI table modification requires a congatec embedded BIOS based on AMIs Aptio V UEFI firmware implementation. Only BIOS releases after mid of 2016 may have the support built-in.

In addition to the BIOS support, the only required utility is the congatec system utility CGUTIL revision 1.5.6 or later.

The utility can be downloaded from the congatec website at www.congatec.com and is available for Windows and Linux.

# 3 Defining and editing changes

The text file defining all necessary changes should be generated by editing the example text shown in the appendix. Any entry that has to be changed must be entered into the related line of the text. Any entry that has to remain unchanged must be removed from the text. Currently it is not possible to simply delete string entries from the built in tables by overwriting them with a whitespace. They need to be overwritten with new content.

It is only possible to implement the changes to the tables by using one single OEM data module. When applying a new module, the previous changes will be deleted entirely. Therefore, if the modified data shall contain only static data then the OEM data module can also be included in a pre-programmed BIOS. If the modified data shall contain some static information like manufacturer and type and some dynamic information like serial number, it is necessary to create for each manufactured unit its own data set composed of the static and the dynamic parts and flash it directly to the unit. See appendix for

script examples.

## Comments:

Comments inside the text file are possible. A comment line must start with an '#'. A comment in the line of a data field is not possible.

## UUID:

There are several versions of UUID defined by RFC 4122. The SMBIOS stores the data as an array of bytes. The data must be entered as characters 0-9 and A-F, representing the value of the bytes in hexadecimal notation.

Although RFC4122 recommends network byte order for all fields, the PC industry (including the ACPI, UEFI, and Microsoft specifications) has consistently used little-endian byte encoding for the first three fields: time_low, time_mid, time_hi_and_version. The same encoding should also be used for the SMBIOS representation of the UUID.

The UUID {00112233-4455-6677-8899-AABBCCDDEEFF} therefore must be entered as: UUID = 33221100554477668899AABBCCDDEEFF

## Chassis Type and OEM data:

The ChassisType is defined as an enumeration, defining the type of mechanical enclosure. Bit 7 defines if there is a chassis lock present. See the SMBIOS specification for details.

The ChassisOEM is defined as an OEM specific DWORD. The data must be entered as characters 0-9 and A-F, representing the value of the bytes in hexadecimal notation.

## OEM String:

This type of table allows the ability to store OEM specific strings. Current implementations allow the ability to store up to 5 strings. It is not possible to store empty strings or strings only consisting of a single space character. Therefore the table always begins with the first valid string.

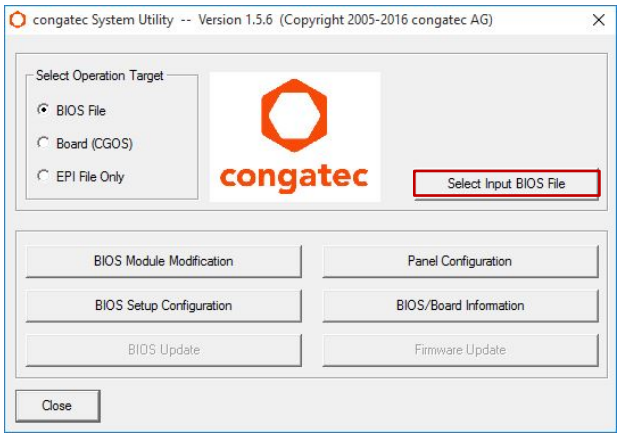
## System Configuration Options:

When available, this type of table is intended to store information about configuration options of the system. It is a list of OEM specific strings. Current implementations allow the ability to store up to 5 strings. It is not possible to store empty strings or strings only consisting of a single space character. Therefore the table always begins with the first valid string.

# 4 Applying changes

Once the text file is fully edited then the OEM data module can be created either with the CGUTILGUI (graphical user interface) or with the CGUTLCMD (command line tool).

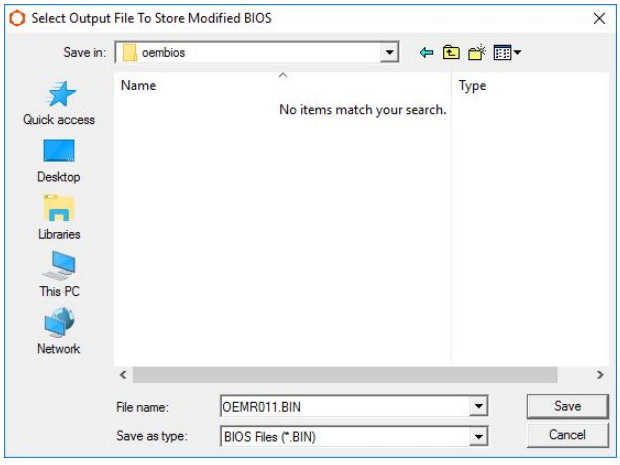
## 4.1 CGUTILGUI




- Open the “congatec System Utility.”

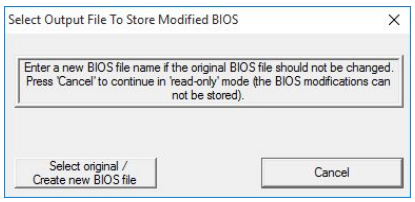
- Select “BIOS File” as operating target.

- Push the button “Select Input BIOS File.”



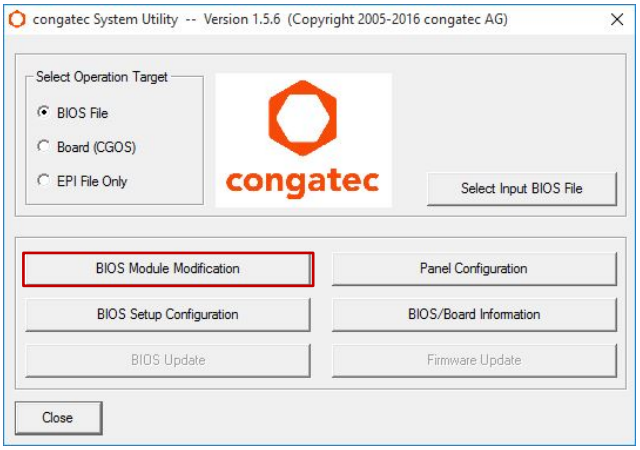

- Browse for your designated BIOS BIN file.

- Click “Select original/Create new BIOS file.”

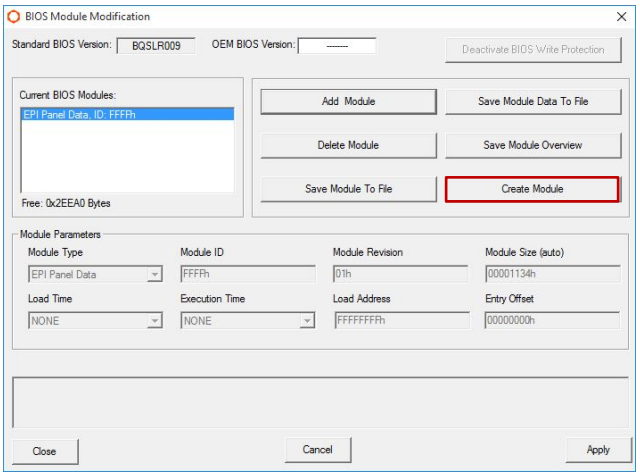



- Enter a file name (in this example “OEMR0011.bin”)

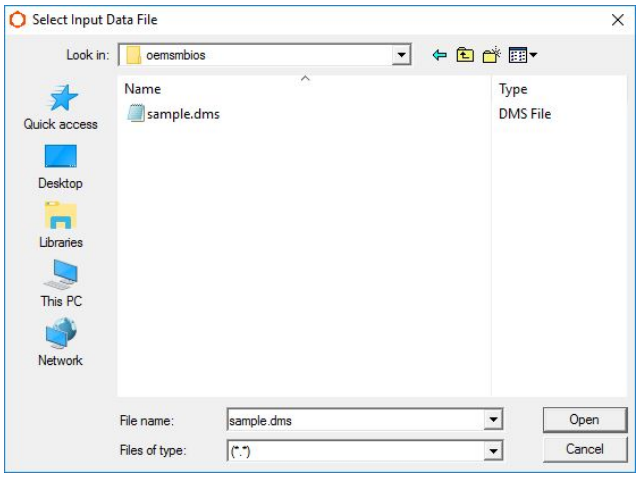
- Changes will be applied only to this file. The original BIOS BIN file will remain unchanged.



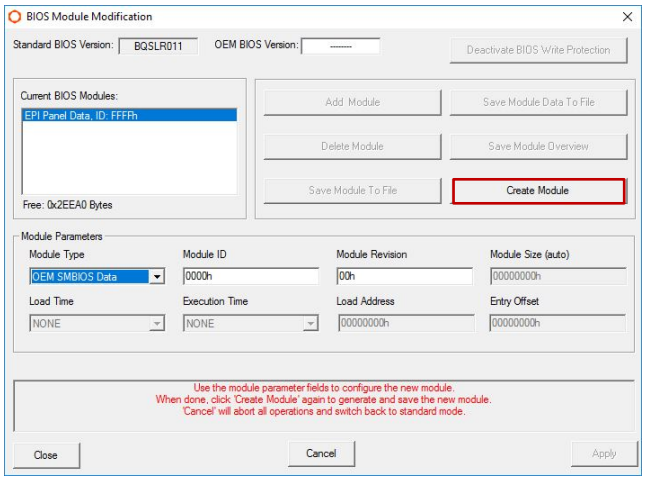

- Click on the "BIOS Module Modification" button.




- Click on the "Create Module" button.



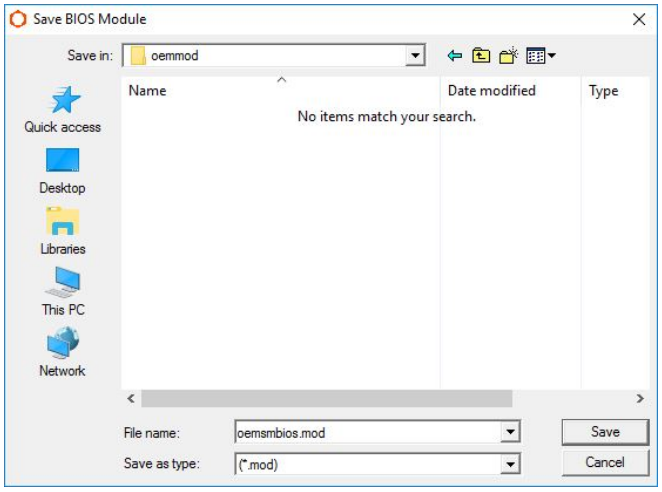

- Browse for your SMBIOS DMI table file (in this example "sample.dms").



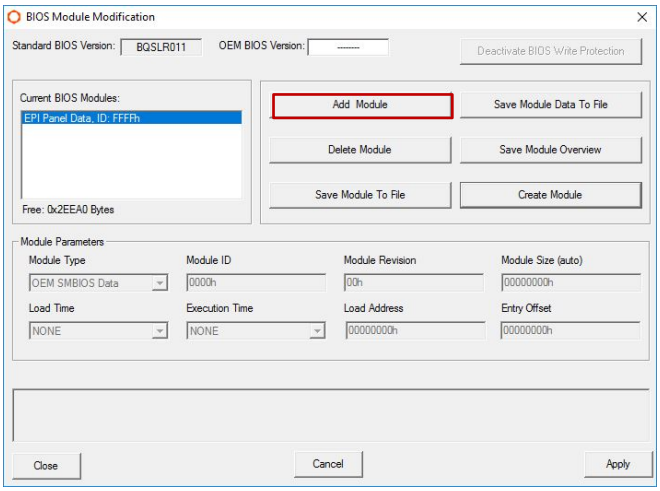

- Choose "OEM SMBIOS Data" from the "Module Type" drop down list.

- Leave the Module ID and Module Revision field unchanged.

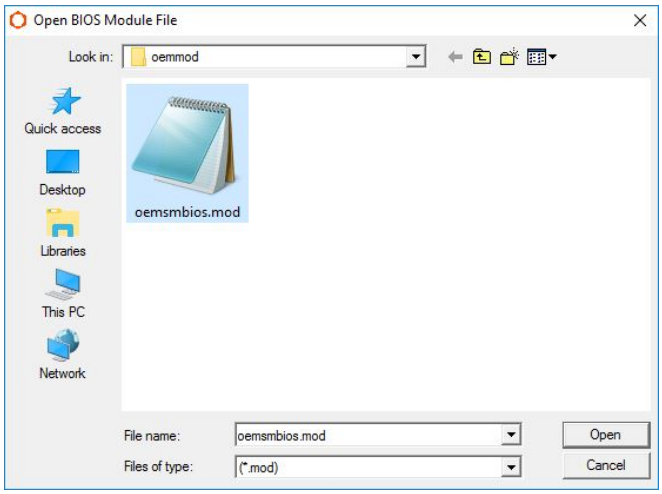
- Once this is completed, click on "Create Module" button.



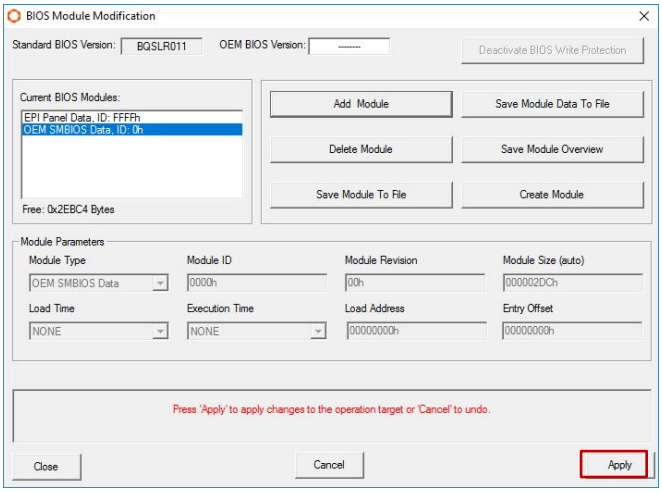

- Chose your designated module file name (in this example "oemsmbios.mod").




- Click on "Add Module" button.




- Browse for the module created earlier ("oemsmbios.mod").




- The designated OEM SMBIOS Data module must be visible in the list of modules.

- Click on "Apply" to save the changes into the BIOS BIN file.

- "Changes successfully applied" will be displayed in red letters in the text box above the "Apply" button.

## 4.2 CGUTLCMD

Create and add the OEM SMBIOS data module with the CGUTLCMD tool.

```
cgutlcmd module /ot:board /create /if:sample.dms /of:sample.mod /t:16
```

**/ot:<>** is the operation target. It is either "board" if the command is executed on the target hardware or the filename of the BIOS file.
**/if:<>** is the input text file defining the changes
**/of:<>** is the OEM data module that will be created
**/t:16** defines the type of the module to be created as OEM SMBIOS Data

Then the OEM SMBIOS module can be inserted using the following command line:

```
cgutlcmd module /ot:board /add /if:sample.mod
```

**/ot:<>** is the operation target. It is either "board" if the BIOS of the target hardware has to be modified or the filename of the BIOS file if a BIOS file has to be modified.

# 5 Additional Information:

| Document | Source |
|---|---|
| CGUTIL Users Guide | www.congatec.com |
| SMBIOS specification | https://www.dmtf.org/standards/smbios |
| RFC 4122 | https://tools.ietf.org/html/rfc4122 |
| RWEverything | http://rweverything.com |
| dmidecode | https://linux.die.net/man/8/dmidecode |

# 6 Appendix:

## 6.1 Sample text file

sample.dms

```
################################################################################
#
# OEM SMBIOS DMI data changes
# Filename: sample.dms
#
################################################################################

[BIOS]
Vendor = NewBiosVendor
Version = NewBiosVersion
ReleaseDate = 12/31/2016

[System]
Manufacturer = NewSystemManufacturer
Product = NewSystemProduct
Version = NewSystemVersion
SerialNum = NewSystemSerialNum
UUID = 33221100554477668899AABBCCDDEEFF
SKU = NewSystemSKU
Family = NewSystemFamily

[BaseBoard]
Manufacturer = NewBaseBoardManufacturer
Product = NewBaseBoardProduct
Version = NewBaseBoardVersion
SerialNum = NewBaseBoardSerialNum
TagNum = NewBaseBoardAssetTag

[Chassis]
Manufacturer = NewChassisManufacturer
ChassisType = 01
Version = NewChassisVersion
SerialNum = NewChassisSerialNum
TagNum = NewChassisAssetTag
ChassisOEM = 00C0DE00

[OemString]
String = congatec AG
String = http://www.congatec.de
String = We simplify
String = the use
String = of embedded technology

[SysConfigOptions]
String = There are no jumpers
String = or any other system controls
String = on the module
String = There are some jumpers
String = on the baseboard

################################################################################
```

## 6.2 Combining static and dynamic information

In the text file with the changes of the OEM SMBIOS tables the order of the entries under a header is not fixed. They are identified only by the tag in the beginning. So it gives the same result if you write:

```
[System]
Manufacturer = NewSystemManufacturer
Product = NewSystemProduct
SerialNum = NewSystemSerialNum
```

or:

```
[System]
SerialNum = NewSystemSerialNum
Manufacturer = NewSystemManufacturer
Product = NewSystemProduct
```

Therefore you can use the table type identifier as a tag after which the dynamic data can be added easily in a script.

For example you start with a file "static.dms" containing:

```
[System]
Manufacturer = NewSystemManufacturer
Product = NewSystemProduct
```

The script will search for the line containing the string "*[System]*" and will insert the line that contains the serial number information after this line.

So after running the script you have as a result the file "local.dms", which combines the static manufacturer and product data and the dynamic serial number data:

```
[System]
SerialNum = MySerialNumberString
Manufacturer = NewSystemManufacturer
Product = NewSystemProduct
```

### 6.2.1 Linux and other Posix compliant operating systems

Using sed(1) in the script:

script

```
#!/bin/sh
SERIAL=MySerialNumberString
sed 's/\[System\]/&\nSerialNum = '"$SERIAL"'/' static.dms > local.dms
```

Note that you have to precede the square brackets with a backslash so that they are not interpreted as control characters and that you have to surround the argument to sed with single quotes for the control characters and with double quotes for the variable.

### 6.2.2 MS Windows

Using a batch file with a loop that walks through the file line by line:

batchfile.bat

```
@echo off
SETLOCAL ENABLEDELAYEDEXPANSION

set serial=MySerialNumberString

set inputFile=static.dms
set outputFile=local.dms
set _strFind=[System]
set _strInsert=SerialNum = %serial%

if exist %outputFile% del %outputFile%
FOR /F "usebackq delims=" %%A IN ("%inputFile%") DO (
  echo  %%A  |  Find  "%_strFind%"  &&  echo  %%A>>"%outputFile%"  &&
echo %_strInsert%>>"%outputFile%"
  IF [!errorlevel!] == [1] echo %%A>>"%outputFile%"
)
```

Make sure to remove the unwanted line break between **`&&`** and **`echo`** when copying the lines out of this document.