# Triage Tool Manual

## GIT for Industry

| | |
|---|---|
| **Version:** | **4.5.0** |
| **Date:** | **31st May 2021** |
| **Author:** | **Smart DCC Ltd.** |
| **Classification:** | **DCC Public** |

# Document Control

### Revision History

| Revision Date | Summary of Changes | Changes Marked | Version Number |
|---|---|---|---|
| 02/09/2016 | First version. | N/A | 1.0.0RC1 |
| 16/09/2016 | Changes to section 2.1 | N/A | 1.0.0RC1 |
| 16/09/2016 | Document review and approval | N/A | 1.0.0RC1 |
| 06/10/2016 | Added log file parser | N/A | 1.0.0RC2 |
| 11/11/2016 | Added parsing of HAN logs | N/A | 1.0.0 |
| 08/08/2017 | Update | N/A | 1.3.8E |
| 25/08/2017 | Update | N/A | 2.0.0 RC1 |
| 30/11/2017 | Update | N/A | 2.0.0 RC2 |
| 09/02/2018 | Update | N/A | 2.0.0 |
| 27/03/2018 | Update | N/A | 2.0.1 |
| 30/04/2018 | Update | N/A | 2.0.2 |
| 28/05/2018 | Update | N/A | 2.0.3 |
| 05/07/2018 | Update | N/A | 2.1.0 |
| 09/10/2018 | Update | N/A | 2.1.1 |
| 16/01/2019 | Updated with GBCS 2 use cases. | N/A | 2.1.2 |
| 06/03/2019 | Update | N/A | 2.1.3 |
| 24/04/2019 | Update | N/A | 2.1.4 |
| 17/06/2019 | Update | N/A | 2.1.5 |
| 06/09/2019 | Update | N/A | 2.1.6 |
| 13/09/2019 | Update | N/A | 3.0.0 |
| 25/10/2019 | Update | N/A | 3.0.1 |
| 14/11/2019 | Update | N/A | 3.0.2 |
| 09/01/2020 | Update | N/A | 3.1.0 |
| 31/01/2020 | Update | N/A | 3.1.1 |
| 13/03/2020 | Update | N/A | 3.2.0 |
| 23/04/2020 | Update | N/A | 3.2.1 |
| 08/05/2020 | Update | N/A | 3.3.0 |
| 19/06/2020 | Update | N/A | 4.0.0 |
| 26/06/2020 | Update | N/A | 4.0.1 |
| 09/07/2020 | Update | N/A | 4.1.0 |

| Revision Date | Summary of Changes | Changes Marked | Version Number |
|---|---|---|---|
| 24/07/2020 | Update | N/A | 4.2.0 |
| 31/07/2020 | Update | N/A | 4.2.1 |
| 12/08/2020 | Update | N/A | 4.2.2 |
| 18/08/2020 | Update | N/A | 4.2.3 |
| 31/08/2020 | Update | N/A | 4.2.4 |
| 10/09/2020 | Update | N/A | 4.2.5 |
| 15/09/2020 | Update | N/A | 4.2.6 |
| 18/09/2020 | Update | N/A | 4.2.7 |
| 06/10/2020 | Update | N/A | 4.2.8 |
| 26/11/2020 | Update | N/A | 4.2.9 |
| 28/01/2021 | Update | N/A | 4.3.0 |
| 31/03/2021 | Update | N/A | 4.4.0 |
| 31/05/2021 | Update | N/A | 4.5.0 |

### Reviewers

| Name | Title / Responsibility | Release Date | Version Number |
|---|---|---|---|
| Karim Kanso | Critical Software Senior Engineer | 11/11/2016 | 1.0.0 |
| André Oliveira | Critical Software Software Engineer | 11/11/2016 | 1.0.0 |
| Dinis Paes | Critical Software Technical Manager | 11/11/2016 | 1.0.0 |
| Dinis Paes | Critical Software Technical Manager | 08/08/2017 | 1.3.8E |
| Dinis Paes | Critical Software Technical Manager | 25/08/2017 | 2.0.0 RC1 |
| Dinis Paes | Critical Software Technical Manager | 30/11/2017 | 2.0.0 RC2 |
| Pedro Almeida | Critical Software GFI Team Leader | 09/02/2018 | 2.0.0 |
| Helder Martins | Critical Software GFI Team Member | 27/03/2018 | 2.0.1 |
| Helder Martins | Critical Software GFI Team Member | 30/04/2018 | 2.0.2 |
| Helder Martins | Critical Software GFI Team Member | 28/05/2018 | 2.0.3 |
| Helder Martins | Critical Software GFI Team Member | 05/07/2018 | 2.1.0 |
| Helder Martins | Critical Software GFI Team Member | 09/10/2018 | 2.1.1 |
| Pedro Trindade | Critical Software GFI Team Member | 16/01/2019 | 2.1.2 |
| Francisco Pinto | Critical Software GFI Team Leader | 06/03/2019 | 2.1.3 |
| Francisco Pinto | Critical Software GFI Team Leader | 24/04/2019 | 2.1.4 |

| Name | Title / Responsibility | Release Date | Version Number |
|---|---|---|---|
| Francisco Pinto | Critical Software GFI Team Leader | 17/06/2019 | 2.1.5 |
| Helder Martins | Critical Software GFI Team Member | 06/09/2019 | 2.1.6 |
| Helder Martins | Critical Software GFI Team Member | 13/09/2019 | 3.0.0 |
| Helder Martins | Critical Software GFI Team Member | 25/10/2019 | 3.0.1 |
| Fábio Silva | Critical Software GFI Team Member | 14/11/2019 | 3.0.2 |
| Helder Martins | Critical Software GFI Team Member | 09/01/2020 | 3.1.0 |
| Helder Martins | Critical Software GFI Team Member | 31/01/2020 | 3.1.1 |
| Helder Martins | Critical Software GFI Team Member | 13/03/2020 | 3.2.0 |
| Francisco Pinto | Critical Software GFI Team Leader | 23/04/2020 | 3.2.1 |
| Francisco Pinto | Critical Software GFI Team Leader | 08/05/2020 | 3.3.0 |
| Nuno Besteiro | Critical Software GFI Team Member | 19/06/2020 | 4.0.0 |
| Nuno Besteiro | Critical Software GFI Team Member | 26/06/2020 | 4.0.1 |
| Nuno Besteiro | Critical Software GFI Team Member | 09/07/2020 | 4.1.0 |
| Helder Martins | Critical Software GFI Team Member | 24/07/2020 | 4.2.0 |
| Nuno Besteiro | Critical Software GFI Team Member | 31/07/2020 | 4.2.1 |
| Francisco Pinto | Critical Software GFI Team Leader | 12/08/2020 | 4.2.2 |
| Helder Martins | Critical Software GFI Team Member | 18/08/2020 | 4.2.3 |
| Helder Martins | Critical Software GFI Team Member | 31/08/2020 | 4.2.4 |
| Francisco Pinto | Critical Software GFI Team Leader | 10/09/2020 | 4.2.5 |
| Daniel Oliveira | Critical Software GFI Team Member | 15/09/2020 | 4.2.6 |
| Francisco Pinto | Critical Software GFI Team Leader | 18/09/2020 | 4.2.7 |
| Daniel Oliveira | Critical Software GFI Team Member | 06/10/2020 | 4.2.8 |
| Francisco Pinto | Critical Software GFI Team Leader | 26/11/2020 | 4.2.9 |
| Francisco Pinto | Critical Software GFI Team Leader | 28/01/2021 | 4.3.0 |
| Helder Martins | Critical Software GFI Team Member | 31/03/2021 | 4.4.0 |
| Pedro Marques | Critical Software GFI Team Member | 31/05/2021 | 4.5.0 |

### Approvals

| Name | Signature | Title / Responsibility | Release Date | Version Number |
|---|---|---|---|---|
| Dinis Paes | | Critical Software Technical Manager | 11/11/2016 | 1.0.0 |
| Dinis Paes | | Critical Software Technical Manager | 08/08/2017 | 1.3.8E |

| Name | Signature | Title / Responsibility | Release Date | Version Number |
|---|---|---|---|---|
| Dinis Paes | | Critical Software Technical Manager | 25/08/2017 | 2.0.0 RC1 |
| Dinis Paes | | Critical Software Technical Manager | 30/11/2017 | 2.0.0 RC2 |
| Pedro Almeida | | Critical Software GFI Team Leader | 09/02/2018 | 2.0.0 |
| Pedro Almeida | | Critical Software GFI Team Leader | 27/03/2018 | 2.0.1 |
| Pedro Almeida | | Critical Software GFI Team Leader | 30/04/2018 | 2.0.2 |
| Pedro Almeida | | Critical Software GFI Team Leader | 28/05/2018 | 2.0.3 |
| Pedro Almeida | | Critical Software GFI Team Leader | 05/07/2018 | 2.1.0 |
| Pedro Almeida | | Critical Software GFI Team Leader | 09/10/2018 | 2.1.1 |
| Pedro Almeida | | Critical Software GFI Team Leader | 16/01/2019 | 2.1.2 |
| Pedro Almeida | | Critical Software Technical Manager | 06/03/2019 | 2.1.3 |
| Pedro Almeida | | Critical Software Technical Manager | 24/04/2019 | 2.1.4 |
| Pedro Almeida | | Critical Software Technical Manager | 17/06/2019 | 2.1.5 |
| Pedro Almeida | | Critical Software Technical Manager | 06/09/2019 | 2.1.6 |
| Pedro Almeida | | Critical Software Technical Manager | 13/09/2019 | 3.0.0 |
| Pedro Almeida | | Critical Software Technical Manager | 25/10/2019 | 3.0.1 |
| Pedro Almeida | | Critical Software Technical Manager | 14/11/2019 | 3.0.2 |
| Pedro Almeida | | Critical Software Technical Manager | 09/01/2020 | 3.1.0 |
| Pedro Almeida | | Critical Software Technical Manager | 31/01/2020 | 3.1.1 |
| Pedro Almeida | | Critical Software Technical Manager | 13/03/2020 | 3.2.0 |
| Pedro Almeida | | Critical Software Technical Manager | 23/04/2020 | 3.2.1 |
| Pedro Almeida | | Critical Software Technical Manager | 08/05/2020 | 3.3.0 |
| Pedro Almeida | | Critical Software Technical Manager | 19/06/2020 | 4.0.0 |
| Pedro Almeida | | Critical Software Technical Manager | 26/06/2020 | 4.0.1 |
| Pedro Almeida | | Critical Software Technical Manager | 09/07/2020 | 4.1.0 |
| Pedro Almeida | | Critical Software Technical Manager | 24/07/2020 | 4.2.0 |
| Francisco Pinto | | Critical Software GFI Team Leader | 31/07/2020 | 4.2.1 |
| Pedro Almeida | | Critical Software Technical Manager | 12/08/2020 | 4.2.2 |
| Pedro Almeida | | Critical Software Technical Manager | 18/08/2020 | 4.2.3 |
| Pedro Almeida | | Critical Software Technical Manager | 31/08/2020 | 4.2.4 |
| Pedro Almeida | | Critical Software Technical Manager | 10/09/2020 | 4.2.5 |
| Pedro Almeida | | Critical Software Technical Manager | 15/09/2020 | 4.2.6 |
| Pedro Almeida | | Critical Software Technical Manager | 18/09/2020 | 4.2.7 |
| Pedro Almeida | | Critical Software Technical Manager | 06/10/2020 | 4.2.8 |

| Name | Signature | Title / Responsibility | Release Date | Version Number |
|---|---|---|---|---|
| Pedro Almeida | | Critical Software Technical Manager | 26/11/2020 | 4.2.9 |
| Pedro Almeida | | Critical Software Technical Manager | 28/01/2021 | 4.3.0 |
| Pedro Almeida | | Critical Software Technical Manager | 31/03/2021 | 4.4.0 |
| Pedro Almeida | | Critical Software Technical Manager | 31/05/2021 | 4.5.0 |

# Contents


**1 Introduction ..... 9**

1.1 Objective ..... 9

1.2 Document structure ..... 9

**2 Overview ..... 10**

2.1 Obtaining Tool ..... 10

2.2 Obtaining Support ..... 11

2.3 Supported Functions ..... 11

2.3.1 Message Validation ..... 11

2.3.2 Payload Decryption ..... 14

2.3.3 Working with Multiple Messages ..... 16

2.3.4 Working with HAN logs ..... 17

2.3.5 Error Reporting ..... 19

**3 HEX ..... 23**

3.1 Examples ..... 23

**4 PCAP ..... 24**

**5 DCF ..... 27**

**6 Log File ..... 28**

6.1 Examples ..... 28

**7 Glossary ..... 31**

# Table of Figures

Figure 1 – Starting Triage Tool ..... 10
Figure 2 – Nominal Tool Output ..... 14
Figure 3 – No Credentials ..... 14
Figure 4 – Message Decryption (Simple) ..... 15
Figure 5 – Decryption MAC Fail ..... 15
Figure 6 – Message Decryption (Advanced) ..... 16
Figure 7 – Brief Option ..... 16
Figure 8 – Message Grouping ..... 17
Figure 9 – Working with HAN logs ..... 18
Figure 10 – HAN log parse errors ..... 19
Figure 11 – Message with Error in Header ..... 20
Figure 12 – Message with Error in Payload ..... 20
Figure 13 – Out of Order GBT Messages ..... 21
Figure 14 – Duplicated Messages ..... 22
Figure 15 – Decoding a PCAP file of a WAN capture ..... 25
Figure 16 – Decoding a PCAP file of a HAN capture ..... 26
Figure 17 – Decoding a DCF file ..... 27
Figure 18 – Log File Invocation ..... 29
Figure 19 – Log File Partial Parse ..... 30
Figure 20 – Log File Parse Fail ..... 30

# 1 Introduction

## 1.1 Objective

This document is a User Manual for the GFI Triage Tool intended to provide detailed technical information for standard and advanced user operation and configuration.

## 1.2 Document structure

Section 1 (Introduction) presents a general description of this document's contents.

Section 2 (Overview) presents a description of the triage tool operation.

Section 3 (HEX) presents the HEX input format, used for reading single GBCS messages.

Section 4 (PCAP) presents the PCAP input format, used for reading WAN and HAN logs.

Section 5 (DCF) presents the DCF (Daintree) input format, used for reading HAN logs.

Section 6 (Log File) presents the Log File input format, used for reading CH log files.

Section 7 (Glossary) presents a list of definitions and acronyms used throughout this document.

# 2 Overview

GIT for Industry (GFI) is a software tool, provided by Smart DCC, for anybody that wishes to check whether their interpretation of the Great Britain Specification Companion for smart meters (GBCS) is consistent with Smart DCC's. At the time of writing, GFI supports Use Cases for GBCS v4.0 Draft 2 over a ZigBee HAN.

The GFI Triage Tool is a tool built over GFI to leverage the GBCS message checking functionality of GFI to support issue triage during systems integration. The tool is capable of examining GBCS payloads and WAN and HAN log files to extract GBCS messages and display them in a meaningful way. In addition, decryption functionality is available.

## 2.1 Obtaining Tool

The tool is distributed as part of the main GFI package as is ready to be used. It is installed as `/opt/gfi/bin/gfi-triage-tool`.

The triage tool is provided with a command line only interface. That is, no GUI is available for the tool, and the user should be confident with using a terminal environment in Ubuntu/Linux. Below are the steps for opening a terminal window on the GFI platform:

1) Open the GFI Command Line icon located in the desktop.

2) Enter the text: "`./bin/gfi-triage-tool`", See Figure 1.

3) Append the relevant parameters to the tool, such as the log files and any credentials needed for decrypting the payloads. The parameters and valid log file formats are detailed in Section 2.3.

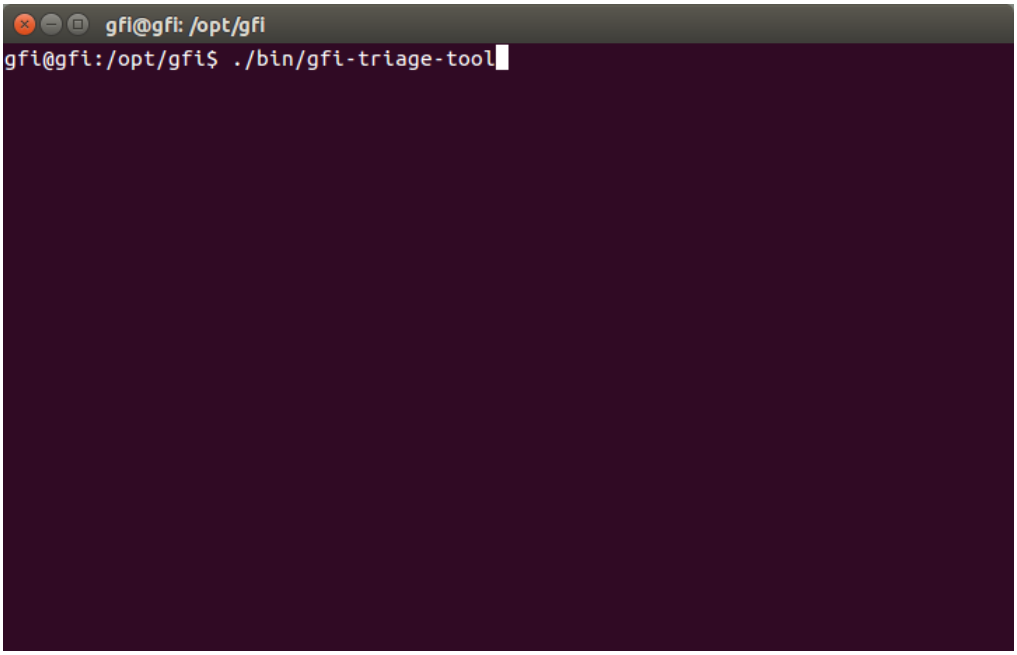


**Figure 1 – Starting Triage Tool**

## 2.2 Obtaining Support

All support for the tool is provided through the GFI helpdesk at DCC. To contact the helpdesk, please use the following email address: `gfisupport@smartdcc.co.uk`

## 2.3 Supported Functions

The current release of GFI Triage Tool supports two main functions. First, it supports GBCS message syntactical verification and then display. Secondly, it supports identification, extraction and decryption of encrypted blocks in the message payloads.

### 2.3.1 Message Validation

In the case of the first function – message validation – the following formats are supported:

- **HEX**
  Singular GBCS messages extracted into hex files. See Section 3 for more details.

- **PCAP**
  Files that contain WAN captures (e.g. ethernet frames with IP packets) or HAN captures (802.15.4 MAC frames with Zigbee payloads) with multiple GBCS messages. See Section 4 for more details.

- **DCF**
  Text files in Daintree format that contain HAN captures (802.15.4 MAC frames with Zigbee payloads) with multiple GBCS messages. See Section 5 for more details.

- **Log File**
  Text files that contain GBCS messages. Currently EDMI CH log files are supported out of the box. Some of the parameters are tuneable, thus it is possible that other similar text log file formats are supported. See Section 6 for more information.

These files are offered to the program by command line arguments. Its usage is:

```
Usage: gfi-triage-tool [<file 1> ... <file n>] [options]
Options:
  --version                     Show version information.
  --brief                       Show messages in brief format.
  --group                       Group related messages.
  --private-key=[EUI64,]<file>  Specify the KA private key.
  --certificate=[EUI64,]<file>  Specify the KA certificate.
  --nwk-key=<32hexchars>        Specify a Zigbee NWK key.
  --aps-key=<32hexchars>        Specify a Zigbee APS key.
  --pcap-file=<file>            Append pcap file to process queue.
  --hex-file=<file>             Append hex file to process queue.
  --dcf-file=<file>             Append Daintree file to process queue.
  --log-file=<file>             Append log file to process queue.
  --log-regex-delimiter=<regex> Regex to delimit log messages.
  --log-columns=<number>        Number of hex columns log messages.
  --csv-file=<file>             Append csv with Use Cases information.
Where:
  EUI64 is the id of the party's credentials in hex format. If
  omitted, then a default value of 0 is used, and is interpreted as
  a default credential to use.
```

The tool, by default, will load the samples-FEATURES.csv file located on /opt/gfi/conf. If the file is not present on that directory it is possible to set the correct file location with `--csv-file=<file>` parameter.

It is possible to provide multiple heterogeneous files in a single invocation of the tool. The tool will decide how to handle each file based on the file name extension, i.e. a file with name `payload.hex` will be treated as a HEX format, a file with name `capture.pcap` will be treated as a PCAP format, a file with name `capture.dcf` will be trated as a DCF format, and a file with name `report.log` will be trated as a Log File format. If the extension is not known, or the file does not exist this will be treated as an error.

In the case where the extension used for the file does not align with the GFI Triage Tool, it is possible to specify the file type using the `--`*`type`*`-file=` command line parameters. That is, for a HEX file, the parameter would be: `--hex-file=/path/to/hex.file`

While processing log files, the tool will attempt to parse the GBCS headers, and identify the GBCS payload. It will then perform syntactical verification of the payload. That is, it will verify the structure of the message in accordance with the underlying datatypes. For example, in the case of an ASN.1 message which consists of *tag*, *length* and *value* tuples, it will check the *length* and *value* are in accordance with the *tag*, but it will not check the value of the *value* is correct with respect to GBCS. Semantical verification of the payloads is not performed by the tool; that is the payloads are not verified against ZCL or COSEM message templates in GBCS.

The triage tool supports the following message headers:

- MAC Header
- Grouping Header
- Message Routing Header
- GBT Header (when encapsulated by Message Routing Header)

As the purpose of the tool is to help triage issues on a SMETS 2 network, full cryptographic verification of the messages is not required as it is catered for by the end systems. As such, neither the MAC nor digital signature are verified to be correct, beyond they are the correct size, i.e. syntactically correct.

In the case the tool identifies a message successfully, it will print basic information regarding that message to the terminal. Table 1 identifies the information that is printed for each message class. In addition to this, warnings will be displayed if originator counters are repeated and out-of-order GBT messages are identified.

| Field | Normal | Supplementary | Supplementary + Counter | Supplementary + KA | GBT |
|---|---|---|---|---|---|
| Time Stamp | If available in source log file. | | | | |
| Message Code | ✓ | ✓ | ✓ | ✓ | ✓ |
| Message Type (CRA Flag) | ✓ | ✓ | ✓ | ✓ | ✓ |
| Business Originator ID | ✓ | ✓ | ✓ | ✓ | ✓ |
| Business Target ID | ✓ | ✓ | ✓ | ✓ | ✓ |
| Originator Counter | ✓ | ✓ | ✓ | ✓ | ✓ |
| Supplementary Remote Party ID | | ✓ | ✓ | ✓ | |
| Supplementary Remote Party Counter | | ✓ | ✓ | ✓ | |
| Supplementary Originator Counter | | | ✓ | | |
| Block Control | | | | | ✓ |
| Block Number | | | | | ✓ |
| Block Number Acknowledgement | | | | | ✓ |
| Message Type (GBT) | | | | | ✓ |

**Table 1 – Displayed Fields**

As an example of the application of the tool to a HEX message and the corresponding output, as depicted in Figure 2. The messages used for these screenshots are available from the RTDS 1.3.2 data pack, available from DCC. The tool will first iterate over the files provided on the command line and extract the GBCS messages, then the tool will display the messages to the terminal.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool ECS40_6.2.7_SINGLE_SUCCESS_COMMAND_GBCS.HEX
Processing [ECS40_6.2.7_SINGLE_SUCCESS_COMMAND_GBCS.HEX] ... ok, 1 messages identified
-----------------------------------------------------------------
Message Code .....................: 004e
Message Type .....................: Command
Business Originator ID ...........: 90b3d51f30010000
Business Target ID ...............: 00db1234567890a0
Originator Counter ...............: 00000000000003ea
Payload Parse ....................: Success
-----------------------------------------------------------------

gfi@gfi:/opt/gfi$
```

**Figure 2 – Nominal Tool Output**

### 2.3.2 Payload Decryption

The GFI Triage Tool supports decryption of messages, providing the necessary credentials are provided. First, consider the situation when the credentials have not been provided. If an encrypted block of data is encountered the tool will print a warning that it could not identify the required credentials. This is depicted in Figure 3.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool GCS16a_4.6.1_SUCCESS_RESPONSE_GBCS.HEX
Processing [GCS16a_4.6.1_SUCCESS_RESPONSE_GBCS.HEX] ... ok, 1 messages identified
-----------------------------------------------------------------
Message Code .....................: 0077
Message Type .....................: Response
Business Originator ID ...........: 00db1234567890a1
Business Target ID ...............: 90b3d51f30010000
Originator Counter ...............: 00000000000003ee
Payload Parse ....................: Success
Warning: Failed to identify a certificate to decrypt message.
Warning: Failed to identify a certificate to decrypt message.
-----------------------------------------------------------------

gfi@gfi:/opt/gfi$
```

**Figure 3 – No Credentials**

As the message in Figure 3 was from RTDS, it is using standard DCC provided SMKI credentials. To be precise, it will be using ZAZ1 credentials and the associate device certificates. Thus, looking at the originator and target IDs, it is clear that this is originating from the device 00db1234567890a1 (in the following this is referred to as A1) and going to Supplier1.

To correctly provide the credentials to the program, the two command line options `--private-key` and `--certificate` are available. In their simplest form, these two command line options are used to provide a certificate and private key pair that will be used to decrypt messages as follows:

```
--certificate=/path/to/certificate

--private-key=/path/to/privatekey
```

In both cases, DER and PEM encoded credentials are accepted. The tool will examine the file to identify the exact encoding. In the case of PEM, it is required that the credential begins with the `----BEGIN…`

header line. The user is recommended to validate their certificates using the `openssl` tool to ensure that it can read them, as the triage tool aligns with `openssl` in this respect.

Revisiting the example from Figure 3, if the crypto command line parameters are offered to the tool, it is possible to decrypt the message, as demonstrated in Figure 4. The credentials used are shipped as part of the GFI platform. Care should be taken that only the key agreement certificates are provided as the digital signature certificates are not relevant for encrypting data.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool GCS16a_4.6.1_SUCCESS_RESPONSE_GBCS.HEX \
> --certificate=/opt/gfi/conf/smki/device/device-ee/cert/00db1234567890A1-ka.der \
> --private-key=/opt/gfi/conf/smki/org/ZAZ1/key/Z1-supplier-ka.key
Processing [GCS16a_4.6.1_SUCCESS_RESPONSE_GBCS.HEX] ... ok, 1 messages identified
-----------------------------------------------------------------
Message Code .....................: 0077
Message Type .....................: Response
Business Originator ID ...........: 00db1234567890a1
Business Target ID ...............: 90b3d51f30010000
Originator Counter ...............: 00000000000003ee
Payload Parse ....................: Success
Plaintext:  (72 bytes):
00 00 00 00 10 26 35 1C 02 00 01 01 00 00 00 06 DA 02 00 00 00 00 04 3D 00 00 00 00 00 6A 00 00
00 00 00 52 00 00 00 00 00 36 00 00 00 00 00 14 D5 00 00 00 00 00 24 00 00 00 00 00 4A 00 00 00
00 00 68 00 00 00 00 00
Plaintext:  (72 bytes):
01 00 00 00 90 77 36 1C 02 00 01 01 00 00 00 06 17 04 00 00 00 00 04 AD 00 00 00 00 00 95 00 00
00 00 00 5F 00 00 00 00 00 3C 00 00 00 00 00 14 35 00 00 00 00 00 B5 00 00 00 00 00 C4 00 00 00
00 00 8C 00 00 00 00 00
-----------------------------------------------------------------

gfi@gfi:/opt/gfi$
```

**Figure 4 – Message Decryption (Simple)**

If the wrong certificate is provided, for example if A0 is provided instead of A1, then decryption will fail and a MAC check failure will arise. This is shown in Figure 5.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool GCS16a_4.6.1_SUCCESS_RESPONSE_GBCS.HEX \
> --certificate=/opt/gfi/conf/smki/device/device-ee/cert/00db1234567890A0-ka.der \
> --private-key=/opt/gfi/conf/smki/org/ZAZ1/key/Z1-supplier-ka.key
Processing [GCS16a_4.6.1_SUCCESS_RESPONSE_GBCS.HEX] ... ok, 1 messages identified
-----------------------------------------------------------------
Message Code .....................: 0077
Message Type .....................: Response
Business Originator ID ...........: 00db1234567890a1
Business Target ID ...............: 90b3d51f30010000
Originator Counter ...............: 00000000000003ee
Payload Parse ....................: Success
Error: Decryption failed: CCryptographyLow::GCMDecryption: HashVerificationFilter: message hash or MAC not valid
Error: Decryption failed: CCryptographyLow::GCMDecryption: HashVerificationFilter: message hash or MAC not valid
-----------------------------------------------------------------

gfi@gfi:/opt/gfi$
```

**Figure 5 – Decryption MAC Fail**

This raises issues if there are multiple xSMEs or multiple remote parties that are involved in transferring encrypted data. Namely, it is only possible to provide one certificate/private key pair to use. To alleviate this issue, the command line options allow for the EUI64 that the credential belongs to, to be specified. This allows for multiple certificates and private keys to be specified on the command line, then the application will decided based on the business originator and target IDs which certificate and private key to utilise. The format for using these options is as follows:

```
--certificate=EUI64,/path/to/certificate

--private-key=EUI64,/path/to/privatekey
```

Here, the name of the certificate/private key is prepended by the EUI64 of the associated entity and a comma. When using this method it is important that the EUI64 is given in hexadecimal format, and that the length is exactly 16 digits. Thus, the example from Figure 4 can be refactored as presented in Figure 6.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool GCS16a_4.6.1_SUCCESS_RESPONSE_GBCS.HEX \
>  --certificate=00db1234567890A1,/opt/gfi/conf/smki/device/device-ee/cert/00db1234567890A1-ka.der \
>  --private-key=90b3d51f30010000,/opt/gfi/conf/smki/org/ZAZ1/key/Z1-supplier-ka.key
Processing [GCS16a_4.6.1_SUCCESS_RESPONSE_GBCS.HEX] ... ok, 1 messages identified
-----------------------------------------------------------------
Message Code .....................: 0077
Message Type .....................: Response
Business Originator ID ...........: 00db1234567890a1
Business Target ID ...............: 90b3d51f30010000
Originator Counter ...............: 00000000000003ee
Payload Parse ....................: Success
Plaintext:  (72 bytes):
00 00 00 00 10 26 35 1C 02 00 01 01 00 00 00 06 DA 02 00 00 00 00 04 3D 00 00 00 00 00 6A 00 00
00 00 00 52 00 00 00 00 00 36 00 00 00 00 00 14 D5 00 00 00 00 00 24 00 00 00 00 00 4A 00 00 00
00 00 68 00 00 00 00 00
Plaintext:  (72 bytes):
01 00 00 00 90 77 36 1C 02 00 01 01 00 00 00 06 17 04 00 00 00 00 04 AD 00 00 00 00 00 95 00 00
00 00 00 5F 00 00 00 00 00 3C 00 00 00 00 00 14 35 00 00 00 00 00 B5 00 00 00 00 00 C4 00 00 00
00 00 8C 00 00 00 00 00
-----------------------------------------------------------------

gfi@gfi:/opt/gfi$
```

**Figure 6 – Message Decryption (Advanced)**

### 2.3.3 Working with Multiple Messages

When viewing log files that contain multiple messages, such as those contained in PCAP files, DCF files, or CH log files, it is often useful to first get a compacted version of the essential information. This is because the standard printout format will consume a number of lines of text, often obscuring the underlying network activity. For this, it is possible to specify the `--brief` command line option. This will result in a summary of each message being printed out in a single line, as demonstrated in Figure 7.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool sample1.pcap --brief
Processing [sample1.pcap] ... ok, 6 messages identified
13:49:47.030368 C 007e 70b3d51f30010000 0022a3000009a469 3f6a
13:49:51.026423 R 0073 0022a3000009a469 70b3d51f30010000 31b4
13:50:37.013802 R 007e 0022a3000009a469 70b3d51f30010000 3f6a
13:51:07.053724 R 007e 0022a3000009a469 70b3d51f30010000 3f6a
13:51:40.050157 R 0088 0022a3000009a469 70b3d51f30010000 31b1
13:52:08.046490 R 007e 0022a3000009a469 70b3d51f30010000 3f6a
gfi@gfi:/opt/gfi$
```

**Figure 7 – Brief Option**

The message summary for the brief option is composed of the following fields:

- Timestamp,

- CRA Flag,
- Message Code,
- Business Originator ID,
- Business Target ID,
- Least 16 bits of the originator counter.

A second feature that GFI Triage Tool offers for dealing with multiple messages is the ability to group the GBCS messages together into a transaction view of the log files. That is, commands and their associated responses are grouped together. This grouping is based on Originator and Target IDs, message code and the originator counter. An example of this functionality, when combined with the `--brief` option, is shown in Figure 8.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool sample1.pcap --brief --group
Processing [sample1.pcap] ... ok, 6 messages identified

- GCS25 (set billing calendar on the GSME):
13:49:47.030368 C 007e 70b3d51f30010000 0022a3000009a469 3f6a
13:50:37.013802 R 007e 0022a3000009a469 70b3d51f30010000 3f6a
13:51:07.053724 R 007e 0022a3000009a469 70b3d51f30010000 3f6a
13:52:08.046490 R 007e 0022a3000009a469 70b3d51f30010000 3f6a

- GCS11 (disable privacy PIN protection on GSME):
13:49:51.026423 R 0073 0022a3000009a469 70b3d51f30010000 31b4

- GCS44 (write contact details on GSME):
13:51:40.050157 R 0088 0022a3000009a469 70b3d51f30010000 31b1
gfi@gfi:/opt/gfi$
```

**Figure 8 – Message Grouping**

The example in Figure 8 shows a command (GCS25) being issued, and 3 responses being sent approximately 1 minute apart from each other. There are two response being sent, presumably from previously issued commands for the use cases: GCS11, GCS44.

### 2.3.4 Working with HAN logs

The GFI Triage Tool is able to identify and decode the GBCS messages in HAN logs, in PCAP or DCF file formats, produced by standard Zigbee capture tools such as Ember Desktop, Ubiqua or Daintree. For Ember Desktop or Ubiqua it is required to export the log file into either PCAP or DCF format. The GFI Triage Tool parses the 802.15.4 MAC frames in these files, decrypts the Zigbee Network (NWK) and Application (APS) payloads, reassembles the APS fragments, and decodes the GBCS messages in the ZSE tunneling cluster data frames.

To decrypt the Zigbee NWK and APS frames, the respective keys must be provided to GFI Triage Tool with the `--nwk-key` and `--aps-key` arguments. Multiple NWK keys can be provided by passing multiple `--nwk-key` arguments, one for each ZigBee network captured in the HAN log, and multiple APS (or Link) keys can be provided by passing multiple `--aps-key` arguments, one for each ZigBee device captured in the HAN log, as exemplified in Figure 9.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool \
>                 --nwk-key=000102030405060708090A0B0C0D0E0F \
>                 --aps-key=550BF7282BF4F3A64A5DFB9E641FF9D0 \
>                 --aps-key=26454DB9FF940A815115C97F9A87C71E \
>                 --aps-key=9830FB42F00989A7D37EF28607C2D8AA \
>                 han-capture.pcap
Processing [han-capture.pcap] ... ok, 87 messages identified
-----------------------------------------------------------------
Timestamp .......................: 2016/10/10 00:04:16.501192000
Message Code ....................: 0008
Usecase .........................: CS02a (provide security credentials details)
Message Type ....................: Command
Business Originator ID ..........: 90b3d51f30000002
Business Target ID ..............: 00db12345678904d
Originator Counter ..............: 0000000000000514
Payload Parse ...................: Success
-----------------------------------------------------------------

-----------------------------------------------------------------
Timestamp .......................: 2016/10/10 00:04:21.057390000
Message Code ....................: 0008
Usecase .........................: CS02a (provide security credentials details)
Message Type ....................: Response
Business Originator ID ..........: 00db12345678904d
Business Target ID ..............: 90b3d51f30000002
Originator Counter ..............: 0000000000000514
Payload Parse ...................: Success
-----------------------------------------------------------------

-----------------------------------------------------------------
Timestamp .......................: 2016/10/10 00:09:57.910917000
Message Code ....................: 000c
Usecase .........................: CS02e (provide device certificates from device)
Message Type ....................: Command
Business Originator ID ..........: 90b3d51f30010000
Business Target ID ..............: 00db12345678904d
Originator Counter ..............: 00000000000003e9
Payload Parse ...................: Success
-----------------------------------------------------------------

-----------------------------------------------------------------
Timestamp .......................: 2016/10/10 00:10:04.552635000
Message Code ....................: 000c
Usecase .........................: CS02e (provide device certificates from device)
Message Type ....................: Response
Business Originator ID ..........: 00db12345678904d
Business Target ID ..............: 90b3d51f30010000
Originator Counter ..............: 00000000000003e9
Payload Parse ...................: Success
-----------------------------------------------------------------
```

**Figure 9 – Working with HAN logs**

Whenever a ZigBee frame is found in a HAN log that fails to decrypt, either because the necessary NWK or APS keys were not provided in the command line arguments, or are incorrect, the GFI Triage Tool reports a warning message once for each different source IEEE address of the frames. The user should then locate the correct keys for the desired network (NWK key) and device link (APS key), and provide them to GFI Triage Tool on a subsequent run, as exemplified in Figure 10.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool capture.pcap
Processing [capture.pcap] ...
Warning: Failed to decrypt NWK frame from source address 22a30000019a3b (missing NWK key?).
Warning: Failed to decrypt NWK frame from source address db12345678904d (missing NWK key?).
Warning: Failed to decrypt NWK frame from source address db1234567890a0 (missing NWK key?).
ok, 0 messages identified

gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool capture.pcap \
>                 --nwk-key=000102030405060708090a0b0c0d0e0f
Processing [capture.pcap] ...
Warning: Failed to decrypt APS frame from source address db12345678904d (missing APS key?).
Warning: Failed to decrypt APS frame from source address 22a30000019a3b (missing APS key?).
ok, 0 messages identified

gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool capture.pcap \
>                 --nwk-key=000102030405060708090a0b0c0d0e0f \
>                 --aps-key=550bf7282bf4f3a64a5dfb9e641ff9d0
Processing [capture.pcap] ...  ok, 79 messages identified
-----------------------------------------------------------------
Timestamp .......................: 2016/10/10 00:04:16.501192000
Message Code ....................: 0008
Usecase .........................: CS02a (provide security credentials details)
Message Type ....................: Command
Business Originator ID ..........: 90b3d51f30000002
Business Target ID ..............: 00db12345678904d
Originator Counter ..............: 0000000000000514
Payload Parse ...................: Success

...
```

**Figure 10 – HAN log parse errors**

### 2.3.5 Error Reporting

In addition to decryption errors, the GFI Triage Tool is able to detect and report errors in the headers and payload of a message; it is also able to detect out-of-order messages. When an error is detected the tool attempts to provide the most accurate location and description as possible.

For messages with errors in the headers that prevent the payload from being parsed, the GFI Triage Tool displays a textual indication of what the erroneous value is, and in some cases the correct/expected

value is also displayed, see Figure 11 for an example where the length of the message in the General Ciphering message does not match the actual message size.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool ECS17d_4.1.2_SINGLE_SUCCESS_COMMAND_GBCS.invalid.HEX
Processing [ECS17d_4.1.2_SINGLE_SUCCESS_COMMAND_GBCS.invalid.HEX] ...
Failed to parse General Ciphering payload:
The size computed from the message contents doesn't match the size of the received GENERAL_CIPHERING_MESSAGE.
Computed message size = 158
GENERAL_CIPHERING_MESSAGE size = 157
failed

gfi@gfi:/opt/gfi$
```

**Figure 11 – Message with Error in Header**

For messages with errors in the payload, the GFI Triage Tool attempts to calculate the byte location of the error in the payload, as shown in Figure 12. This number is displayed in decimal, and is relative to the start of the payload, e.g. for a Grouping Header this will be relative to the start of the content field.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool GCS01a_1.1.1_FUTURE_DATED_BLOCK_SUCCESS_COMMAND_GBCS.nulled.HEX
Processing [GCS01a_1.1.1_FUTURE_DATED_BLOCK_SUCCESS_COMMAND_GBCS.nulled.HEX] ... ok, 1 messages identified
-----------------------------------------------------------------
Message Code ......................: 006b
Message Type ......................: Command
Business Originator ID ............: 90b3d51f30010000
Business Target ID ................: 00db1234567890a1
Originator Counter ................: 00000000000003ef
Payload Parse .....................: Fail at byte 109
-----------------------------------------------------------------

gfi@gfi:/opt/gfi$
```

**Figure 12 – Message with Error in Payload**

For every GBT message that is sent out-of-order, the GFI Triage Tool displays the GBT block number of the out of order GBT message and the GBT block number that was expected, see Figure 13 for an example.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool ecs22a_out_of_order_gbt_example.pcap
Processing [ecs22a_out_of_order_gbt_example.pcap] ... ok, 3 messages identified
-----------------------------------------------------------------
Message Code .....................: 0036
Message Type .....................: Response
Business Originator ID ...........: 00db1234567890a0
Business Target ID ...............: 90b3d51f30020000
Originator Counter ...............: 00000000000004b4
Block Number .....................: 0001
Block Number Ack .................: 0000
GBT Message Type .................: Data Transfer Block
Payload Parse ....................: N/A
-----------------------------------------------------------------

-----------------------------------------------------------------
Message Code .....................: 0036
Message Type .....................: Response
Business Originator ID ...........: 00db1234567890a0
Business Target ID ...............: 90b3d51f30020000
Originator Counter ...............: 00000000000004b4
Block Number .....................: 0003 (out of order; expected 0002)
Block Number Ack .................: 0000
GBT Message Type .................: Data Transfer Block
Payload Parse ....................: N/A
-----------------------------------------------------------------

-----------------------------------------------------------------
Message Code .....................: 0036
Message Type .....................: Response
Business Originator ID ...........: 00db1234567890a0
Business Target ID ...............: 90b3d51f30020000
Originator Counter ...............: 00000000000004b4
Block Number .....................: 0002
Block Number Ack .................: 0000
GBT Message Type .................: Data Transfer Block
Payload Parse ....................: N/A
-----------------------------------------------------------------

gfi@gfi:/opt/gfi$
```

**Figure 13 – Out of Order GBT Messages**

For every duplicated message found (a message where its originator counter value is the same as a previously sent message by the same business originator EUI64), the GFI Triage Tool displays an indication next to the originator counter value that is repeated, see Figure 14 for an example.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool repeated_originator_counter_example.pcap
Processing [repeated_originator_counter_example.pcap] ... ok, 3 messages identified
-------------------------------------------------------------------
Timestamp: .......................: 2016/06/04 14:49:47.630368000
Message Code .....................: 007e
Message Type .....................: Command
Business Originator ID ...........: 70b3d51f30010000
Business Target ID ...............: 0022a3000009a469
Originator Counter ...............: 019fb51162783f6a
Payload Parse ....................: Success
-------------------------------------------------------------------

-------------------------------------------------------------------
Timestamp: .......................: 2016/06/04 14:50:37.813802000
Message Code .....................: 007e
Message Type .....................: Response
Business Originator ID ...........: 0022a3000009a469
Business Target ID ...............: 70b3d51f30010000
Originator Counter ...............: 019fb51162783f6a
Payload Parse ....................: Success
-------------------------------------------------------------------

-------------------------------------------------------------------
Timestamp: .......................: 2016/06/04 14:51:07.853724000
Message Code .....................: 007e
Message Type .....................: Response
Business Originator ID ...........: 0022a3000009a469
Business Target ID ...............: 70b3d51f30010000
Originator Counter ...............: 019fb51162783f6a (repeated)
Payload Parse ....................: Success
-------------------------------------------------------------------

gfi@gfi:/opt/gfi$
```

**Figure 14 – Duplicated Messages**

# 3 HEX

The HEX format is an unofficial standard, but is a canonical representation of a GBCS message that is human readable, and was chosen as the format for RTDS payloads. For this reason it has been chosen to be supported within the triage tool, mainly when there is a singular message that needs to be checked or decrypted.

## 3.1 Examples

The format of the files are simple; they only contain hexadecimal characters represented in ASCII with optional whitespace. See the examples below:

**Example 1 (ECS40 Command)**

```
DD0000000000000521100000000DF09010000000000000003EA0890B3D51F300100000800DB1234
567890A00002004E1ED9200003EA000201000100005E2C220B020100010100000002FF020200
0000AE8894A0DEC0E916790127ED
```

**Example 2 (ECS22a GBT Response)**

```
DD 09 02 00 00 00 00 00 00 04 B4 08 00 DB 12 34 56 78 90 A0 08 90 B3 D5 1F 30
02 00 00 00 02 00 36 00 82 02 BA 01 00 00 00 00 E0 81 00 05 00 00 82 02 AC 00
00 00 A9 00 00 00 76 1B C8 E7 E8 00 00 00 9B 00 00 00 6C 1B C8 EE F0 00 00 00
62 00 00 00 44 1B C8 F5 F8 00 00 00 7B 00 00 00 56 1B C8 FD 00 00 00 00 F3 00
00 00 AA 1B C9 04 08 00 00 00 1E 00 00 00 15 1B C9 0B 10 00 00 00 E5 00 00 00
A0 1B C9 12 18 00 00 00 29 00 00 00 1C 1B C9 19 20 00 00 00 F3 00 00 00 AA 1B
C9 20 28 00 00 00 08 00 00 00 05 1B C9 27 30 00 00 00 E3 00 00 00 9E 1B C9 2E
38 00 00 00 59 00 00 00 3E 1B C9 35 40 00 00 00 59 00 00 00 3E 1B C9 3C 48 00
00 00 25 00 00 00 19 1B C9 43 50 00 00 00 A2 00 00 00 71 1B C9 4A 58 00 00 00
37 00 00 00 26 1B C9 51 60 00 00 00 EC 00 00 00 A5 1B C9 58 68 00 00 00 6B 00
00 00 4A 1B C9 5F 70 00 00 00 63 00 00 00 45 1B C9 66 78 00 00 00 EB 00 00 00
A4 1B C9 6D 80 00 00 00 5E 00 00 00 41 1B C9 74 88 00 00 00 63 00 00 00 45 1B
C9 7B 90 00 00 00 BF 00 00 00 85 1B C9 82 98 00 00 00 47 00 00 00 31 1B C9 89
A0 00 00 00 58 00 00 00 3D 1B C9 90 A8 00 00 00 8B 00 00 00 61 1B C9 97 B0 00
00 00 DC 00 00 00 9A 1B C9 9E B8 00 00 00 A0 00 00 00 70 1B C9 A5 C0 00 00 00
F7 00 00 00 AC 1B C9 AC C8 00 00 00 B0 00 00 00 7B 1B C9 B3 D0 00 00 00 AF 00
00 00 7A 1B C9 BA D8 00 00 00 A2 00 00 00 71 1B C9 C1 E0 00 00 00 4C 00 00 00
35 1B C9 C8 E8 00 00 00 12 00 00 00 0C 1B C9 CF F0 00 00 00 9D 00 00 00 6D 1B
C9 D6 F8 00 00 00 C0 00 00 00 86 1B C9 DE 00 00 00 00 AF 00 00 00 7A 1B C9 E5
08 00 00 00 83 00 00 00 5B 1B C9 EC 10 00 00 00 E9 00 00 00 A3 1B C9 F3 18 00
00 00 24 00 00 00 19 1B C9 FA 20 00 00 00 8B 00 00 00 61 1B CA 01 28 00 00 00
CD 00 00 00 8F 1B CA 08 30 00 00 00 FC 00 00 00 B0 1B CA 0F 38 00 00 00 E4 00
00 00 9F 1B CA 16 40 00 00 00 73 00 00 00 50 1B CA 1D 48 00 00 00 9F 00 00 00
6F 1B CA 24 50 00 00 00 9C 00 00 00 6D 1B CA 2B 58 00 00 00 DF 00 00 00 9C 1B
CA 32 60 00 00 00 8B 00 00 00 61 1B CA 39 68 00 00 00 00 00 00 00 00 1B CA 40
70 00 00 00 CB 00 00 00 8E 1B CA 47 78 00 00 00 E9 00 00 00 A3 1B CA 4E 80 00
00 00 E3 00 00 00 9E 1B CA 55 88 00 00 00 0B 00 00 00 07 1B CA 5C 90 00 00 00
31 00 00 00 22 1B CA 63 98 00 00 00 3C 00 00 00 2A 01 01 00 00 01 35 4E 71 2F
34 27 75 1F 40 39 94
```

# 4 PCAP

PCAP is the file format used by most WAN and HAN analysers, such as Wireshark[1], Ember Desktop[2], or Ubiqua[3], to store all the packets captured by a device connected to the network under analysis. The PCAP file format is specified in libpcap[4].

For WAN captures, the GFI Triage Tool is able to identify and decode the GBCS messages in a PCAP file with frames of any link-layer type (e.g. Ethernet, Wi-Fi), network layer (e.g. IPv4, IPv6), transport layer (e.g. UDP, TCP), and application layer (e.g. HTTP, custom), as long as each GBCS message is fully contained in a single PCAP packet and not encrypted or obfuscated.

For HAN captures, the GFI Triage Tool is able to identify and decode the GBCS messages in a PCAP file with 802.15.4 MAC frames with Zigbee payloads. It parses the MAC frames, decrypts the Zigbee NWK and APS payloads (the respective keys must be provided with the `--nwk-key` and `--aps-key` arguments), reassembles the APS fragments, and decodes the GBCS messages in the ZSE tunneling cluster data frames.

The PCAP files are passed to the tool as arguments. They must either have names that end with the `.pcap` extension or be passed with the `--pcap-file` option. On loading the file, the tool will state how many messages it has loaded from the file, and then enumerate the messages as described in Section 2.3, and exemplified in Figure 15 (WAN capture) and Figure 16 (HAN capture). While working with PCAP files, the user is recommended to read Section 2.3.3.

[1] https://www.wireshark.org/
[2] https://www.silabs.com/products/wireless/mesh-networking/zigbee/Pages/zigbee-ember-desktop-network-analyzer.aspx
[3] https://www.ubilogix.com/products/ubiqua
[4] http://www.tcpdump.org/manpages/pcap-savefile.5.html

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool sample1.pcap
Processing [sample1.pcap] ... ok, 6 messages identified
-----------------------------------------------------------------
Timestamp: .......................: 2016/06/04 14:49:47.630368000
Message Code .....................: 007e
Message Type .....................: Command
Business Originator ID ...........: 70b3d51f30010000
Business Target ID ...............: 0022a3000009a469
Originator Counter ...............: 019fb51162783f6a
Payload Parse ....................: Success
-----------------------------------------------------------------


-----------------------------------------------------------------
Timestamp: .......................: 2016/06/04 14:49:51.926423000
Message Code .....................: 0073
Message Type .....................: Response
Business Originator ID ...........: 0022a3000009a469
Business Target ID ...............: 70b3d51f30010000
Originator Counter ...............: 00001ea115e731b4
Payload Parse ....................: Success
-----------------------------------------------------------------
```

**Figure 15 – Decoding a PCAP file of a WAN capture**

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool \
>                 --nwk-key=000102030405060708090A0B0C0D0E0F \
>                 --aps-key=550BF7282BF4F3A64A5DFB9E641FF9D0 \
>                 --aps-key=26454DB9FF940A815115C97F9A87C71E \
>                 --aps-key=9830FB42F00989A7D37EF28607C2D8AA \
>                 han-capture.pcap
Processing [han-capture.pcap] ... ok, 87 messages identified
-----------------------------------------------------------------
Timestamp .......................: 2016/10/10 00:04:16.501192000
Message Code ....................: 0008
Usecase .........................: CS02a (provide security credentials details)
Message Type ....................: Command
Business Originator ID ..........: 90b3d51f30000002
Business Target ID ..............: 00db12345678904d
Originator Counter ..............: 0000000000000514
Payload Parse ...................: Success
-----------------------------------------------------------------

-----------------------------------------------------------------
Timestamp .......................: 2016/10/10 00:04:21.057390000
Message Code ....................: 0008
Usecase .........................: CS02a (provide security credentials details)
Message Type ....................: Response
Business Originator ID ..........: 00db12345678904d
Business Target ID ..............: 90b3d51f30000002
Originator Counter ..............: 0000000000000514
Payload Parse ...................: Success
-----------------------------------------------------------------

-----------------------------------------------------------------
Timestamp .......................: 2016/10/10 00:09:57.910917000
Message Code ....................: 000c
Usecase .........................: CS02e (provide device certificates from device)
Message Type ....................: Command
Business Originator ID ..........: 90b3d51f30010000
Business Target ID ..............: 00db12345678904d
Originator Counter ..............: 00000000000003e9
Payload Parse ...................: Success
-----------------------------------------------------------------

-----------------------------------------------------------------
Timestamp .......................: 2016/10/10 00:10:04.552635000
Message Code ....................: 000c
Usecase .........................: CS02e (provide device certificates from device)
Message Type ....................: Response
Business Originator ID ..........: 00db12345678904d
Business Target ID ..............: 90b3d51f30010000
Originator Counter ..............: 00000000000003e9
Payload Parse ...................: Success
-----------------------------------------------------------------
```

**Figure 16 – Decoding a PCAP file of a HAN capture**

# 5 DCF

DCF files contain HAN captures in Daintree SNA (Sensor Network Analyser) text format, where each line represents an 802.15.4 MAC frame with Zigbee payload. It is a format that many other Zigbee protocol analysers, such as Ember Desktop, or Ubiqua, can also export to.

The GFI Triage Tool is able to identify and decode all the GBCS messages in a DCF file, in both the original Daintree DCF format, or the Ubiqua DCF format. It parses the MAC frames, decrypts the Zigbee NWK and APS payloads (the respective keys must be provided with the `--nwk-key` and `--aps-key` arguments), reassembles the APS fragments, and decodes the GBCS messages in the ZSE tunneling cluster data frames.

The DCF files are passed to the tool as arguments. They must either have names that end with the .dcf extension or be passed with the `--dcf-file` option. On loading the DCF files, the tool will state how many GBCS messages it has identified, and then enumerate them as described in Section 2.3, and exemplified in Figure 17.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool \
>                 --nwk-key=000102030405060708090A0b0c0d0e0f \
>                 --aps-key=0123456789abcdef0123456789abcdef \
>                 capture.dcf
Processing [capture.dcf] ... ok, 2 messages identified
-----------------------------------------------------------------
Timestamp .........................: 2016/10/01 14:46:26.814236000
Message Code ......................: 0008
Usecase ...........................: CS02a (provide security credentials details)
Message Type ......................: Command
Business Originator ID ............: 70b3d51f30000003
Business Target ID ................: 847d50000000001a
Originator Counter ................: 0157807dfe120001
Payload Parse .....................: Success
-----------------------------------------------------------------

-----------------------------------------------------------------
Timestamp .........................: 2016/10/01 14:46:30.942508000
Message Code ......................: 0008
Usecase ...........................: CS02a (provide security credentials details)
Message Type ......................: Response
Business Originator ID ............: 847d50000000001a
Business Target ID ................: 70b3d51f30000003
Originator Counter ................: 0157807dfe120001
Payload Parse .....................: Success
-----------------------------------------------------------------

gfi@gfi:/opt/gfi$
```

**Figure 17 – Decoding a DCF file**

# 6 Log File

In some situations, where textual log files are produced it is possible to use these as input into the GFI Triage Tool. Currently, only log files produced by EDMI Communications Hubs are fully supported, however some of the parameters are tunable, allowing for similar files to also be interpreted.

Files in this format require:

1. A message delimiter (or header) line of text.
    a. This is specified by a regular expression, the only constraint is that it must be on a single line of text.
2. The GBCS message is printed in uppercase HEX format in rows of a fixed column width.
    a. The HEX bytes must be continuious within the line (but separated by a space), however, the lines may be prefixed, postfixed with extra text.
    b. Its possible to interleave the lines of HEX bytes with other text.

The GFI Triage Tool allows for both of these parameters to be tuned by means of command line arguments, if the default configuration is not acceptable.

Due to the flexibility of the parser, it is not possible to gauarentee that the GBCS message will be extracted successfully. As its acceptable to interleave lines of text with the message payload (see examples 2 and 3 below), the interleaved lines might have text that incorrectly gets parsed. For example, a message from a ZigBee stack printing out an encryption key (which is also 16 bytes in length).

## 6.1 Examples

**Example 1 (ECS40 Command, Nominal)**

Consider the following example of a log file (named `log.txt`):

```
Log message 1
Log message 2
Log message 3
Message Received from WAN:
DD 00 00 00 00 00 00 52 11 00 00 00 00 DF 09 01
00 00 00 00 00 00 03 EA 08 90 B3 D5 1F 30 01 00
00 08 00 DB 12 34 56 78 90 A0 00 02 00 4E 1E D9
20 00 03 EA 00 02 01 00 01 00 00 5E 2C 22 0B 02
01 00 01 01 00 00 00 02 FF 02 02 00 00 00 AE 88
94 A0 DE C0 E9 16 79 01 27 ED
Log message 4
Log message 5
Log message 6
```

The GFI Triage Tool can be invoked with this file as depicted in Figure 18.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool --log-file=log.txt \
>                                       --log-regex-delimiter="Message Received from (.*):" \
>                                       --log-columns=16
Processing [log.txt] ... ok, 1 messages identified
-----------------------------------------------------------------
Timestamp .........................: 2000/01/01 00:00:00.000000000
Message Code ......................: 004e
Message Type ......................: Command
Business Originator ID ............: 90b3d51f30010000
Business Target ID ................: 00db1234567890a0
Originator Counter ................: 00000000000003ea
Payload Parse .....................: Success
-----------------------------------------------------------------
```

**Figure 18 – Log File Invocation**

Here, the message delimiter is "`Message Received from (.*):`" and the column width is 16 bytes. Fixing both of these parameters, both of the following two examples would be accepted.

### Example 2 (ECS40 Command, Interleaved)

```
Log message 1
Log message 2
Message Received from HAN:
Log message 3
DD 00 00 00 00 00 00 52 11 00 00 00 00 DF 09 01
00 00 00 00 00 00 03 EA 08 90 B3 D5 1F 30 01 00
Log message 4
00 08 00 DB 12 34 56 78 90 A0 00 02 00 4E 1E D9
20 00 03 EA 00 02 01 00 01 00 00 5E 2C 22 0B 02
01 00 01 01 00 00 00 02 FF 02 02 00 00 00 AE 88
Log message 5
94 A0 DE C0 E9 16 79 01 27 ED
Log message 6
```

### Example 3 (ECS40 Command, Interline mixing)

```
Log message 1
Log message 2Message Received from CHF:
Log message 3DD 00 00 00 00 00 00 52 11 00 00 00 00 DF 09 01


00 00 00 00 00 00 03 EA 08 90 B3 D5 1F 30 01 00
Log message 400 08 00 DB 12 34 56 78 90 A0 00 02 00 4E 1E D9

20 00 03 EA 00 02 01 00 01 00 00 5E 2C 22 0B 02
01 00 01 01 00 00 00 02 FF 02 02 00 00 00 AE 88Log message 5
94 A0 DE C0 E9 16 79 01 27 ED

Log message 6
```

**Example 4 (ECS40 Command, Parse Fail)**

Due to the flexibility of the parser, it is possible that it incorrectly fails to identify the message from the text file. In this case, instead of silently ignoreing the partially obtained GBCS message it will print a warning, so that the issue can be further investiaged. As an example of this, consider example 1 but chaninging the number of columns to 15 (see Figure 19), and then 17 (see Figure 20).

In the first case, when set to 15, the parser will extract, according to the regex standard, the left-most match. Thus, it will consume the first 15 HEX bytes of each line, and finally result in a parse fail as it is reaches the end of the file without parsing the correct number of bytes. The number of bytes to parse is determined by extracting the length from the MAC Header (or Grouping Header). In the second case, when set to 17, none of the lines match the regex, as its expecting rows with 17 hex bytes.

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool --log-file=log.txt \
>                                       --log-regex-delimiter="Message Received from (.*):" \
>                                       --log-columns=15
Processing [log.txt] ...
Warning: Someting has gone wrong on line: 13
Warning: The partially obtained message is: dd 00 00 00 00 00 00 52 11 00 00 00 00 df 09 00 00 00 00
 00 00 03 ea 08 90 b3 d5 1f 30 01 00 08 00 db 12 34 56 78 90 a0 00 02 00 4e 1e 20 00 03 ea 00 02 01
00 01 00 00 5e 2c 22 0b 01 00 01 01 00 00 00 02 ff 02 02 00 00 00 ae

failed
```

Figure 19 – Log File Partial Parse

```
gfi@gfi:/opt/gfi$ ./bin/gfi-triage-tool --log-file=log.txt \
>                                       --log-regex-delimiter="Message Received from (.*):" \
>                                       --log-columns=17
Processing [log.txt] ... failed
```

Figure 20 – Log File Parse Fail

# 7 Glossary

Table 2 presents the list of definitions used throughout this document.

| Name | Description |
|---|---|
| Applicable Document | A document is considered applicable if it complements this document. All its content is directly applied as if it was stated as an annex of this document. |
| APS | ZigBee application frame, see ZigBee sepecification for more information. |
| HEX | Hexadecimal format file. I.e. text file with ASCII characters representing hexadecimal bytes, with optional whitespace. E.g. “A1A2 A3”. |
| NWK | ZigBee network frame, see ZigBee specification for more information. |
| PCAP | Packet Capture, a standard format used for recording network packets. |
| Reference Document | A document is considered a reference if it is referred but not applicable to this document. Reference documents are mainly used to provide further reading. |
| RegEx | Regular Expression |

**Table 2 – Definitions.**

Table 3 presents the list of acronyms used throughout this document.

| Acronym | Description |
|---|---|
| AD | Applicable Document |
| ASCII | American Standard Code for Information Interchange |
| CSW | Critical Software, S.A. |
| DUT | Device Under Test |
| ESME | Electricity Smart Metering Equipment |
| EUI-64 | The IEEE-defined 64-bit Extended Unique Identifier |
| GBCS | Great Britain Companion Specification |
| GBT | General Block Transfer |
| GFI | GIT For Industry |
| GIT | GBCS Interface Testing |

| Acronym | Description |
|---|---|
| GSME | Gas Smart Metering Equipment |
| HAN | Home Area Network |
| NA | Not Applicable |
| RD | Reference Document |
| RTDS | Reference Test Data Set |
| SMETS | Smart Metering Equipment Technical Specifications |
| SMKI | Smart Meter Key Infrastructure |
| TBC | To be confirmed |
| TBD | To be defined |
| WAN | Wide Area Network |
| ZSE | ZigBee Smart Energy |

**Table 3 – Acronyms.**