GFI Segmented Processing

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
13/04/2017	First version	NA	1.3.7E
25/08/2017	Update	NA	2.0.0 RC1
30/11/2017	Update	NA	2.0.0 RC2
02/09/2018	Update	NA	2.0.0
27/03/2018	Update	NA	2.0.1
30/04/2018	Update	NA	2.0.2
28/05/2018	Update	NA	2.0.3
05/07/2018	Update	NA	2.1.0
09/10/2018	Update	NA	2.1.1
16/01/2019	Update	NA	2.1.2
06/03/2019	Update	NA	2.1.3
24/04/2019	Update	NA	2.1.4
17/06/2019	Update	NA	2.1.5
06/09/2019	Update	NA	2.1.6
13/09/2019	Update	NA	3.0.0
25/10/2019	Update	NA	3.0.1
14/11/2019	Update	NA	3.0.2
09/01/2020	Update	NA	3.1.0
31/01/2020	Update	NA	3.1.1
13/03/2020	Update	NA	3.2.0
23/04/2020	Update	NA	3.2.1
08/05/2020	Update	NA	3.3.0
19/06/2020	Update	NA	4.0.0
26/06/2020	Update	NA	4.0.1
09/07/2020	Update	NA	4.1.0
24/072020	Update	NA	4.2.0
31/072020	Update	NA	4.2.1
12/08/2020	Update	NA	4.2.2
18/08/2020	Update	NA	4.2.3
31/08/2020	Update	NA	4.2.4

Revision Date	Summary of Changes	Changes Marked	Version Number
10/09/2020	Update	NA	4.2.5
15/09/2020	Update	NA	4.2.6
18/09/2020	Update	NA	4.2.7
06/10/2020	Update	NA	4.2.8
26/11/2020	Update	NA	4.2.9
28/01/2021	Update	NA	4.3.0
31/03/2021	Update	NA	4.4.0
31/05/2021	Update	NA	4.5.0

Reviewers

Name	Title / Responsibility	Release Date	Version Number
Andre Oliveira	Critical Software Project Engineer	13/04/2017	1.3.7E
Dinis Paes	Critical Software Technical Manager	13/04/2017	1.3.7E
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	02/09/2018	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
Helder Martins	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
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

Name	Title / Responsibility	Release Date	Version Number
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
Sara Alves	Critical Software GFI Team Member	23/04/2020	3.2.1
Sara Alves	Critical Software GFI Team Member	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
José Rodrigues	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	13/04/2017	1.3.7E
Dinis Paes		Critical Software Technical Manager	25/08/2017	2.0.0 RC1
Dinis Paes		Critical Software Technical Manager	04/12/2017	2.0.0 RC2
Pedro Almeida		Critical Software GFI Team Leader	02/09/2018	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

Name	Signature	Title / Responsibility	Release Date	Version Number
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
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 8

2 Configuration 9

3 UDP 10

4 TCP 11

5 Sample execution 12

6 GFI Interface Tool 14

6.1 Execution 14

6.1.1 Execution on another PC 16

6.2 Arguments 16

6.3 Commands 17

7 Glossary 17

1 Introduction

This document describes the GFI Segmented Processing functionality.

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 (GBCS) for smart meters 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 typical scenario when testing a physical meter is to use the GFI software to interact directly with the meter, as illustrated in Figure 1. The GFI software comprises the gfi-testing-tool and the gfi-comms-hub programs. The gfi-testing-tool emulates the Remote Party (RP) and the Access Control Broker (ACB), generates the GBCS commands, checks the GBCS responses and alerts, and generates the test reports. The gfi-comms-hub emulates the Communications Hub (CH), creating the ZigBee HAN for the physical meter to join and exchange GBCS messages as ZigBee frames.

Figure 1 – GFI software and physical meter

The new GFI Segmented Processing functionality described in this document adds the possibility to use the GFI software with a physical CH, as illustrated in Figure 2. In this scenario, the GFI software comprises only the gfi-testing-tool program, which sends every GBCS command generated by its ACB emulator directly to the physical CH, and delivers back to the ACB emulator every GBCS response and alert received from the CH, for checking and reporting. The communication between the gfi-testing-tool and the CH uses either TCP or UDP, over IPv4 or IPv6, which are described in Sections 3 and 4.

Figure 2 – GFI software, physical Comms Hub, and physical meter

In practice, it is desirable and recommended to have an intermediate external software interface module between the gfi-testing-tool and the physical CH, as illustrated in Figure 3. This module is implemented by the manufacturers, allowing them to use custom protocols and interfaces (RS232, optical probe, …) to interact with the CH.

Figure 3 – GFI software, interface module, CH, and meter

2 Configuration

To configure the gfi-testing-tool to use the GFI Segmented Processing functionality to communicate with an external CH please refer to the document Graphical User Interface Manual, Section 4.3.2.

3 UDP

When gfi-testing-tool is configured to use UDP to communicate with the CH, it sends each GBCS command message generated by the ACB emulator, in a separate datagram, to the address and port specified in the configuration, and with the message data in binary format exactly as specified in the GBCS. In the case of a Large Remote Party Message, the ACB emulator first splits it into the required General Block Transfer (GBT) messages and each one is sent in a separate datagram. Likewise, the CH must send each GBCS response, alert, or GBT message in a separate datagram, in the same binary format, but to the address and port of the source of the datagram previously received from the gfi-testing-tool, as illustrated in Figure 4, and in this direction the datagram data may contain extra bytes before and/or after the GBCS message, which are discarded by gfi-testing-tool.

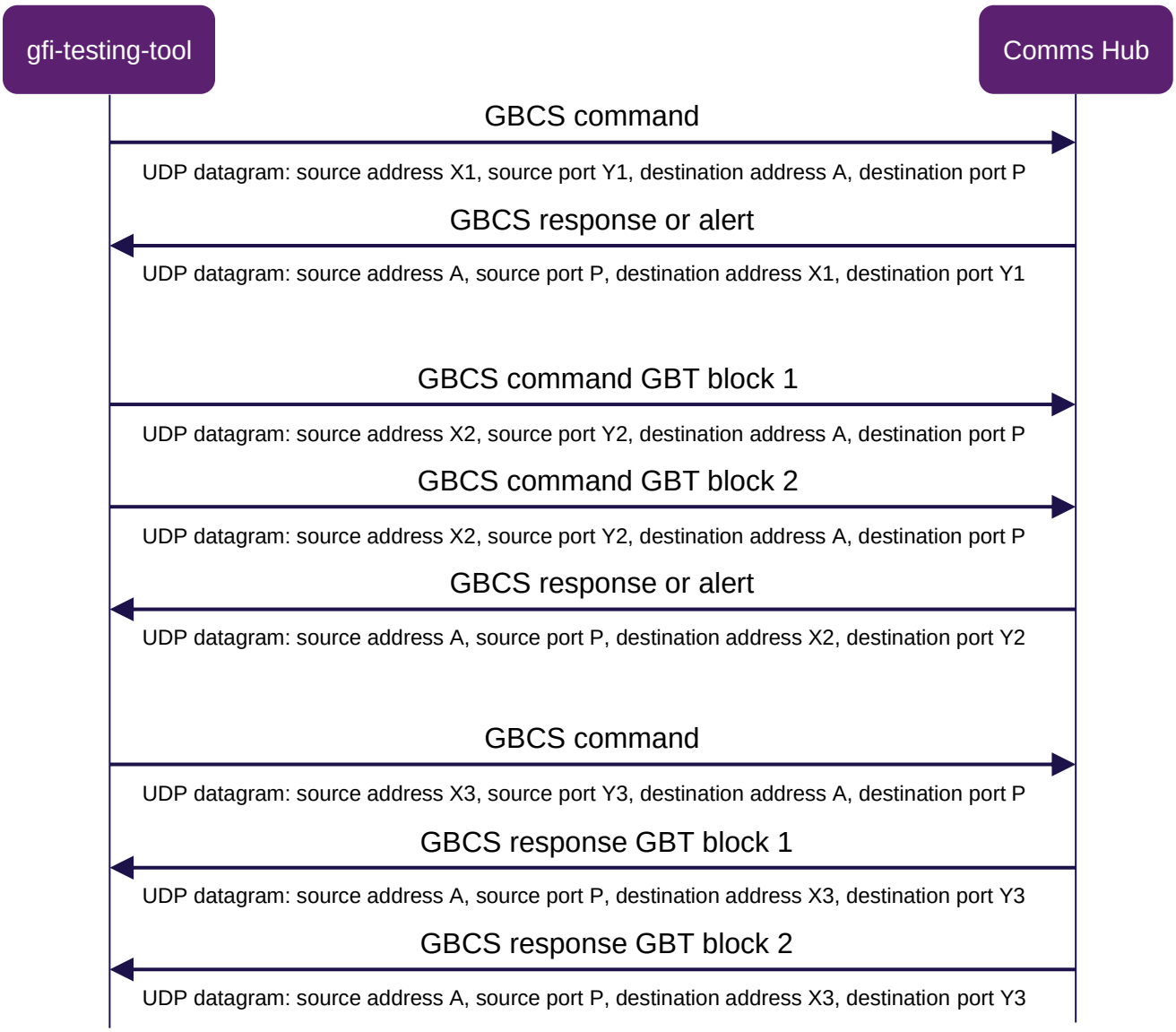

Figure 4 – UDP datagram sequence examples

4 TCP

When gfi-testing-tool is configured to use TCP to communicate with the CH, it will connect to the CH at the specified address and port at the start of each execution. The GBCS command messages generated by the ACB emulator are then sent to the CH in text format, one per line, using the following Standard Commands for Programmable Instruments (SCPI) syntax:

```
GBCS:DATA#<number>:<length>:<string of HEX value character representation of
                                                        the binary data>\r\n
```

In the case of a Large Remote Party Message, the ACB emulator first splits it into the required GBT messages and each one is sent in a separate line. Likewise, the CH sends each GBCS response, alert, or GBT message using the same SCPI syntax, also one message per line.

The `<number>` parameter is to indicate the sequence number of the order of each segment of the command/response/alert as they are sent or received (whether they are a single command/response, GBT, alert or a combination of them). It is a decimal number that is incremented for each message sent (the gfi-testing-tool starts counting from 0 for each new execution).

The `<length>` parameter is a decimal number that indicates how many binary bytes this GBCS command, response, alert, or GBT message has.

The `<string of HEX value character representation of the binary data>` parameter is a sequence of `<length>` times 2 uppercase hexadecimal characters (0123456789ABCDEF) that encode the `<length>` binary bytes of this GBCS command, response, alert, or GBT message.

The `\r` character is the ASCII carriage return character 13 (0x0D).

The `\n` character is the ASCII line feed character 10 (0x0A).

Example:

```
GBCS:DATA#0:72:DD0000000000004011000000000DF09020000000000000003EB0800DB1234567
890A00890B3D51F30010000000200210CDA200003EB0000010001020000F4E3A029790221B3B
6549478\r\n
```

5 Sample execution

The following is the log of an execution of the gfi-testing-tool configured to use the GFI Segmented Processing functionality described in this document, when testing the behaviour of a meter to the CS02b use case. The log messages generated by the CCHProxy equipment class are highlighted, to call attention to the interface between the gfi-testing-tool and the external CH: in iteration 1 the gfi-testing-tool RP emulator generates a Large Remote Party Message, which the ACB emulator splits into two GBT messages, which are sent to the external CH in separate packets, and then the CH sends a packet with the GBCS response message and then another packet with the GBCS alert message; iteration 2 is the typical case of a single GBCS command message and single GBCS response message; iteration 3 is another Large Remote Party Message, this one requiring four GBT messages to be sent to the external CH.

```
gfi@gfi:/opt/gfi$ /opt/gfi/bin/gfi-testing-tool gfi-scenario-chproxy.xml tests/ATG-TEST-CS02b-ESME.xml
-----------------------------------------------------------------------------------------------------
    TIME      | AREA  | LV | MESSAGE
-----------------------------------------------------------------------------------------------------
00:00:00.146 |  MNGE |  0 | Preparing Test Execution...
00:00:00.146 | CHPXY |  4 | CHFClient: Connecting to the CH at address 127.0.0.1 on TCP port 50002
00:00:00.147 | CHPXY |  4 | CHFClient: Connected to the CH at address 127.0.0.1 on TCP port 50002
00:00:00.149 | RPORT |  4 | Report location: /opt/gfi/tests/test-report.html
00:00:00.162 |  MNGE |  0 | Starting Test Execution...
00:00:00.164 |  MNGE |  2 | Testcase: ATG-TC-CS02b-ESME, Starting Iteration: 1 of 3
00:00:00.175 | EQUIP |  4 | KRP: Sending GENERAL_SIGNING_MESSAGE (1873 bytes)
00:00:00.176 |   ACB |  4 | ACBEmulator: GENERAL_SIGNING_MESSAGE received from KRP
...
00:00:00.181 | EQUIP |  4 | ACB: Sending GENERAL_CIPHERING_MESSAGE (1200 bytes)
00:00:00.181 | CHPXY |  4 | CHFClient: Sending 1200-byte GBCS message to CH
...
00:00:00.184 | EQUIP |  4 | ACB: Sending GENERAL_CIPHERING_MESSAGE (802 bytes)
00:00:00.184 | CHPXY |  4 | CHFClient: Sending 802-byte GBCS message to CH
...
00:00:04.803 | CHPXY |  4 | CHFClient: Received 115-byte GBCS message from CH
00:00:04.803 | CHPXY |  4 | CHFClient: Publishing 115-byte GBCS message
00:00:04.803 | EQUIP |  4 | ACB: Received GENERAL_SIGNING_MESSAGE (115 bytes)
00:00:04.804 |   ACB |  4 | ACBEmulator: Routing GENERAL_SIGNING_MESSAGE to KRP
00:00:04.804 | EQUIP |  4 | KRP: Received GENERAL_SIGNING_MESSAGE (115 bytes)
00:00:04.805 |   KRP |  4 | RPEmulator: Performing sanity checks
00:00:04.805 |  GBCS |  4 | GBCS: Performing sanity checks over GENERAL_SIGNING_MESSAGE
00:00:04.810 |   KRP |  4 | RPEmulator: Signature verification succeeded.
00:00:04.811 |   KRP |  4 | RPEmulator: Message accepted.
00:00:04.811 |   STN |  0 | New GBCS response received:CS02B_RESPONSE
...
00:00:05.904 | CHPXY |  4 | CHFClient: Received 331-byte GBCS message from CH
00:00:05.904 | CHPXY |  4 | CHFClient: Publishing 331-byte GBCS message
00:00:05.904 | EQUIP |  4 | ACB: Received GENERAL_SIGNING_MESSAGE (331 bytes)
00:00:05.905 |   ACB |  4 | ACBEmulator: Routing GENERAL_SIGNING_MESSAGE to KRP
00:00:05.905 | EQUIP |  4 | KRP: Received GENERAL_SIGNING_MESSAGE (331 bytes)
00:00:05.906 |   KRP |  4 | RPEmulator: Performing sanity checks
00:00:05.906 |  GBCS |  4 | GBCS: Performing sanity checks over GENERAL_SIGNING_MESSAGE
00:00:05.912 |   KRP |  4 | RPEmulator: Signature verification succeeded.
00:00:05.913 |   KRP |  4 | RPEmulator: Message accepted.
00:00:05.914 |   STN |  0 | New GBCS alert received:CS02B_ALERT
00:00:05.917 |  MNGE |  2 | Testcase: ATG-TC-CS02b-ESME, Complete Iteration: 1 of 3
...
00:00:05.917 |  MNGE |  2 | Testcase: ATG-TC-CS02b-ESME, Starting Iteration: 2 of 3
00:00:05.922 | EQUIP |  4 | KRP: Sending GENERAL_SIGNING_MESSAGE (902 bytes)
00:00:05.923 |   ACB |  4 | ACBEmulator: GENERAL_SIGNING_MESSAGE received from KRP
00:00:05.929 | EQUIP |  4 | ACB: Sending GENERAL_CIPHERING_MESSAGE (929 bytes)
00:00:05.929 | CHPXY |  4 | CHFClient: Sending 929-byte GBCS message to CH
...
00:00:08.729 | CHPXY |  4 | CHFClient: Received 182-byte GBCS message from CH
00:00:08.729 | CHPXY |  4 | CHFClient: Publishing 182-byte GBCS message
00:00:08.729 | EQUIP |  4 | ACB: Received GENERAL_SIGNING_MESSAGE (182 bytes)
00:00:08.729 |   ACB |  4 | ACBEmulator: Routing GENERAL_SIGNING_MESSAGE to KRP
00:00:08.729 | EQUIP |  4 | KRP: Received GENERAL_SIGNING_MESSAGE (182 bytes)
00:00:08.730 |   KRP |  4 | RPEmulator: Performing sanity checks
```

```
00:00:08.730 |  GBCS |  4 | GBCS: Performing sanity checks over GENERAL_SIGNING_MESSAGE
00:00:08.733 |   KRP |  4 | RPEmulator: Signature verification succeeded.
00:00:08.733 |   KRP |  4 | RPEmulator: Message accepted.
00:00:08.733 |   STN |  0 | New GBCS response received:CS02B_RESPONSE
00:00:08.734 |  MNGE |  2 | Testcase: ATG-TC-CS02b-ESME, Complete Iteration: 2 of 3
...
00:00:08.734 |  MNGE |  2 | Testcase: ATG-TC-CS02b-ESME, Starting Iteration: 3 of 3
00:00:08.740 |  TEST |  4 | CGBCSUseCase: Preparing to send GENERAL_SIGNING_MESSAGE
00:00:08.743 |   UNK |  4 | CC02BCommandUseCase: Adding APEX signature to GENERAL_SIGNING_MESSAGE
00:00:08.752 | EQUIP |  4 | KRP: Sending GENERAL_SIGNING_MESSAGE (4542 bytes)
00:00:08.755 |   ACB |  4 | ACBEmulator: GENERAL_SIGNING_MESSAGE received from KRP
...
00:00:08.759 | EQUIP |  4 | ACB: Sending GENERAL_CIPHERING_MESSAGE (1200 bytes)
00:00:08.759 | CHPXY |  4 | CHFClient: Sending 1200-byte GBCS message to CH
...
00:00:08.762 | EQUIP |  4 | ACB: Sending GENERAL_CIPHERING_MESSAGE (1200 bytes)
00:00:08.762 | CHPXY |  4 | CHFClient: Sending 1200-byte GBCS message to CH
...
00:00:08.765 | EQUIP |  4 | ACB: Sending GENERAL_CIPHERING_MESSAGE (1200 bytes)
00:00:08.765 | CHPXY |  4 | CHFClient: Sending 1200-byte GBCS message to CH
...
00:00:08.767 | EQUIP |  4 | ACB: Sending GENERAL_CIPHERING_MESSAGE (1146 bytes)
00:00:08.767 | CHPXY |  4 | CHFClient: Sending 1146-byte GBCS message to CH
...
00:00:19.904 | CHPXY |  4 | CHFClient: Received 667-byte GBCS message from CH
00:00:19.904 | CHPXY |  4 | CHFClient: Publishing 667-byte GBCS message
00:00:19.905 | EQUIP |  4 | ACB: Received GENERAL_SIGNING_MESSAGE (667 bytes)
00:00:19.905 |   ACB |  4 | ACBEmulator: Routing GENERAL_SIGNING_MESSAGE to KRP
00:00:19.906 | EQUIP |  4 | KRP: Received GENERAL_SIGNING_MESSAGE (667 bytes)
00:00:19.906 |   KRP |  4 | RPEmulator: Performing sanity checks
00:00:19.906 |  GBCS |  4 | GBCS: Performing sanity checks over GENERAL_SIGNING_MESSAGE
00:00:19.912 |   KRP |  4 | RPEmulator: Signature verification succeeded.
00:00:19.912 |   KRP |  4 | RPEmulator: Message accepted.
00:00:19.912 |   STN |  0 | New GBCS response received:CS02B_RESPONSE
00:00:19.915 |  MNGE |  2 | Testcase: ATG-TC-CS02b-ESME, Complete Iteration: 3 of 3
00:00:19.915 |  MNGE |  2 | Testcase: ATG-TC-CS02b-ESME, Complete
00:00:19.915 |  MNGE |  2 | Testcase: ATG-TC-CS02b-ESME, Overall Result = PASS
00:00:19.915 |  MNGE |  0 | Complete Run 1 of 1
00:00:19.915 |  MNGE |  0 | Complete Test Execution
00:00:19.915 | RPORT |  0 | ----------------------------------------------------------------------
00:00:19.915 | RPORT |  0 | OVERALL TEST RESULT: PASS
00:00:19.915 | RPORT |  0 | ----------------------------------------------------------------------
```

6 GFI Interface Tool

The GFI Interface Tool is command line utility that acts as a simple interface module suitable for passing GBCS messages between GFI and a CommsHub CLI interface.

Its main features include setting up a TCP server that will wait for a connection from GFI using the Segmented Processing API, receiving GBCS commands from GFI and sending back GBCS responses and alerts. Incoming messages are shown on the console and optionally saved to file. Outgoing messages can be written on the console or loaded from a file. Altough the Segmented Processing functionality uses the SCPI syntax for TCP message exchange, the GFI Interface Tool can also work with messages in hexadecimal, Base64 and binary formats.

6.1 Execution

The following steps show how to use the GFI Interface Tool with its default settings:

1) Open a Command Line Terminal.

2) Change directory: "`cd /opt/gfi/bin`".

3) Start the tool: "`./gfi-interface-tool.py`", see Figure 5.

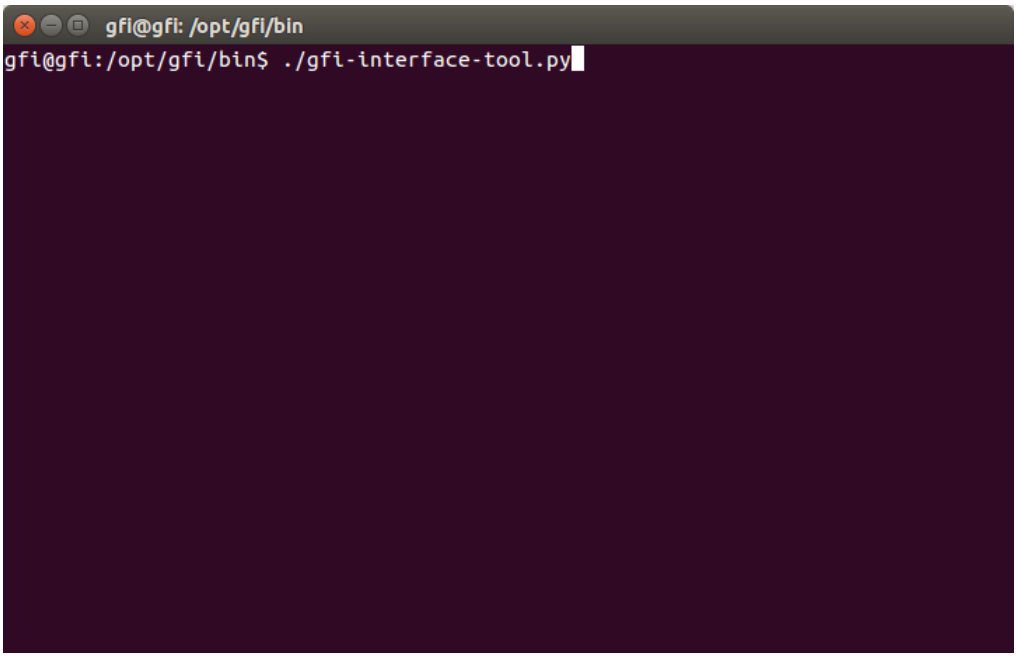

Figure 5 - Starting the GFI Interface Tool

4) On start-up the tool will listen for connection requests from GFI, see Figure 6.

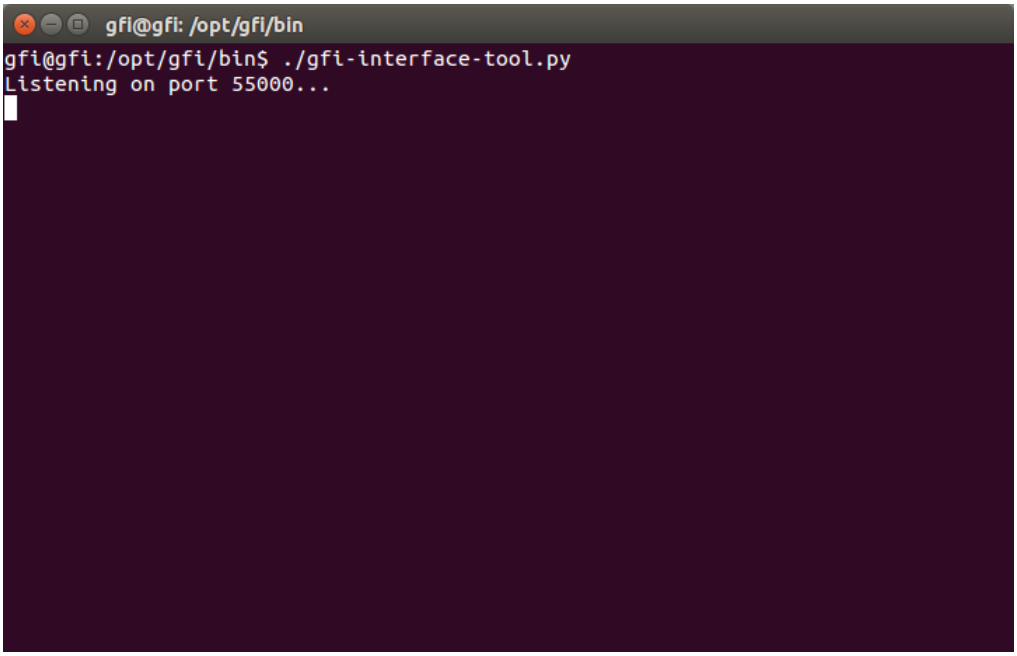

Figure 6 - Listening for connections

5) Start the test on GFI.

6) The GFI Interface Tool will accept the connection request from GFI, receive a message with a GBCS command and display the message on the console, see Figure 7.

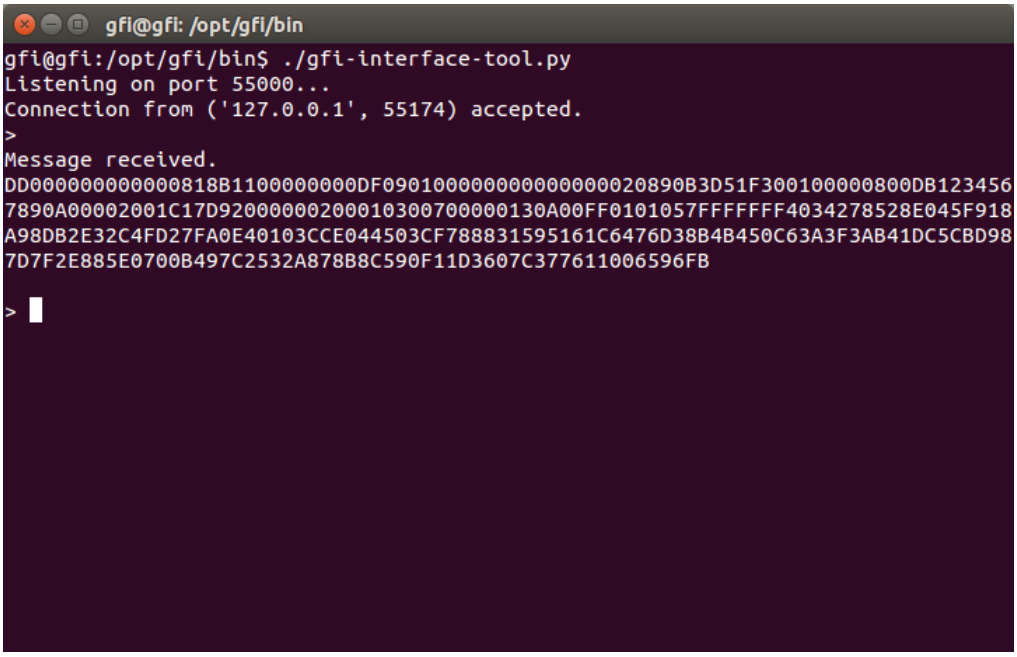

Figure 7 - Receiving a command

7) From the GFI Interface Tool send a message with a GBCS response back to GFI: "`m <message>`". GFI will receive the response and resume the test. After the test is completed GFI will close the connection and the GFI Interface Tool will exit, see Figure 8.

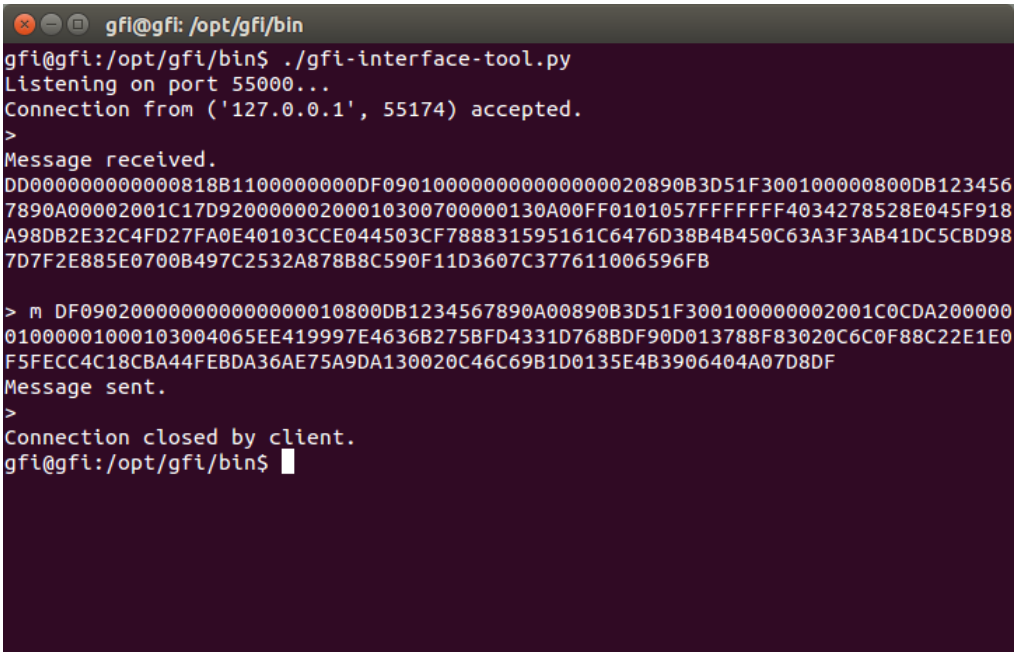

Figure 8 - Sending a response

6.1.1 Execution on another PC

The GFI Interface Tool is Python script and therefore can be executed on a PC with the suitable Python version (2.7) installed. On some operating systems (e.g. Microsoft Windows) the Python interpreter must be explicitely started as follows: “`python gfi-interface-tool.py`” .

6.2 Arguments

The GFI Interface Tool can be configured using the following command line arguments:

- **h, help**

  Displays the GFI Interface Tool version and the list of command line arguments.

- **p, port**

  The TCP port on which the tool will listen for a connection from GFI.

- **t, timeout**

  The time in seconds the tool waits for a connection. If zero no timeout is used. This parameter is optional and if not present 60 seconds is assumed.

- **f, format**

  The format used to display and enter messages in the console and save and load messages to and from file. Possible values are 'hex' (hexadecimal), 'b64' (Base64), 'scpi' (SCPI syntax) and 'bin' (raw binary). 'bin' is only supported in files and fallbacks to 'hex' in the console. This parameter is optional and if not present 'hex' is assumed.

- **s, save-dir**

The directory where received messages are saved. This parameter is optional and if not present received messages are not saved.

- **--hex-spacing**

  Adds a space between each byte when displaying or saving messages using the hexadecimal format. This parameter is optional and if not present spaces are not added.

6.3 Commands

Once a connection is accepted the GFI Interface Tool makes the following commands available:

- **h**

  Displays the list of available commands.

- **m <message>**

  Sends the specified message to GFI. Messages containing spaces should be enclosed in double-quotes.

- **f <file>**

  Sends the message in the specified file to GFI. File paths containing spaces should be enclosed in double-quotes.

- **q**

  Exits the GFI Interface Tool.

Note: When the format is other than 'scpi' the tool will calculate the outgoing messages sequence number starting with zero for the first message and incrementing the value by one for each successive message.

7 Glossary

Table 1 presents the list of acronyms used throughout this document.

Acronym	Description
ACB	Access Control Broker
ASCII	American Standard Code for Information Interchange
CH	Communications Hub
GBCS	Great Britain Companion Specification
GBT	General Block Transfer
GFI	GIT for Industry
GIT	GBCS Interface Testing

Acronym	Description
GUI	Graphical User Interface
HAN	Home Area Network
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
NA	Not Applicable
RP	Remote Party
SCPI	Standard Commands for Programmable Instruments
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
XML	Extensible Markup Language

Table 1 – Acronyms.