



Deliverable D3.4

5G-PPP Security Enablers Documentation (v1.0)

Generic Collector Interface Enabler

Project name	5G Enablers for Network and System Security and Resilience	
Short name	5G-ENSURE	
Grant agreement	671562	
Call	H2020-ICT-2014-2	
Delivery date	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	Orange: Jean-Philippe Warry, Ghada Arfaoui	

This part will be removed before the final edition and publication

Document Version	Date	Change(s)	Author(s)
0.1	28.06.2016	Created template	Felix Klaedtke

Foreword

5G-ENSURE belongs to the first group of EU-funded projects which collaboratively develop 5G under the umbrella of the 5G Infrastructure Public Private Partnership (5G-PPP) in the Horizon 2020 Programme. The overall goal of 5G-ENSURE is to deliver strategic impact across technology and business enablement, standardisation and vision for a secure, resilient and viable 5G network. The project covers research & innovation - from technical solutions (5G security architecture and testbed with 5G security enablers) to market validation and stakeholders engagement - spanning various application domains.

Situate the enabler's manual with respect to the project objectives and other documents of the project (one paragraph). E.g., it is part of D3.4 and

Disclaimer

The information in this document is provided 'as is', and no guarantee or warranty is given that the information is fit for any particular purpose.

The EC flag in this deliverable is owned by the European Commission and the 5G PPP logo is owned by the 5G PPP initiative. The use of the flag and the 5G PPP logo reflects that 5G-ENSURE receives funding from the European Commission, integrated in its 5G PPP initiative. Apart from this, the European Commission or the 5G PPP initiative have no responsibility for the content.

Copyright notice

© 2015-2017 5G-ENSURE Consortium

Contents

1	Introduction.....	5
2	Installation and Administration Guide	5
2.1	System Requirements.....	5
2.2	Enabler Configuration.....	6
2.3	Enabler Installation.....	8
2.4	Troubleshooting	8
3	User and Programmer Guide.....	9
3.1	User Guide	9
3.2	Programmer Guide	10
4	Unit Tests.....	10
4.1	Information about Tests	10
4.2	Unit Test 1.....	10
4.3	Unit Test 2.....	Erreur ! Signet non défini.
4.4	Unit Test n	Erreur ! Signet non défini.
5	Acknowledgements	11
6	Abbreviations.....	11
7	References	11

Note: In case the template does not fit for your enabler, please feel free to deviate from the proposed structure. However, you should have a good reason when you change the structure. Furthermore, please discuss your changes with your task leader or the deliverable editor.

Note: This is the template for *one* enabler. Please fill in a name for <X>. The manuals of all enablers will be compiled into a single document (PDF) later. This single PDF document will contain a global introduction and a conclusion.

Note: Store the final manual in the folder “D3.4” in the WP3 folder on VTT’s workspace (link: <https://workspace.vtt.fi/sites/5g-ensure/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2Fsites%2F5g%2Densure%2FShared%20Documents%2FWorkPackages%2FWP3%2Fshared%20documents%2FD3%2E4&FolderCTID=0x0120006FF9FDCDF197A64797E81A95BCBF2A6D&View={612A22FC-AEC9-4EE5-929B-2B2B682452A1}>)

Store your manual of the enabler in a file named T3N_<X>.docx, where (1) N is the task number and (2) <X> is the enabler name. No spaces and special characters! Also provide a PDF file (same naming but with the “.pdf” suffix).

1 Introduction

Provide a brief introduction of the enabler and an overview of this section. Recall the enablers's use case(s) (a small paragraph with a link to D3.1 should suffice).

In this chapter, we present the “Generic Collector interface” enabler documentation. This enabler mainly aims at collecting data from the 5G network elements / components. The “Generic Collector Interface” will provide to authorized parties / actors large amount of data namely logs, events and incidents related to virtualization, identity management, communication protocols / layers / stacks and some specific privileges escalation. This enabler leverages also the implementation of efficient FastData inside 5G Networks, in order to detect as soon as possible security and efficiency issues of the network.

At the writing time of this document, the Generic Collector interface is still under research. In the scope of R1, we plan to develop some features (namely the report exchange), of the open specifications described in D3.2, as a proof of concept.

2 Installation and Administration Guide

This subsection should describe how the enabler is installed, configured, and administrated.

2.1 System Requirements

Describe the requirements of the enabler. Describe the basic network architecture for your enable and reflect on the interaction with other network components, if any. You may want to recall/reuse some parts from the open specification of the enabler here.

In a nutshell, we recall a simplified functional overview of the “Generic Collector interface”. The Client engine sends XML reports to the server engine. Depending on the security policies, the server engine transmits these reports to the designated service provider.

As shown in Figure 1 , the “Generic Collector Interface” consists of three main software blocs namely the client engine software (cf. Open specification in D3.2), the server engine software (Open specification D3.2) and the service software. The first software, so called “monitoringClient.py”, should be installed on all the network components. This software should run with the root privileges in order to be able later to get necessary information. The second software, so called “monitoringServer.py”, should run in an external server (i.e., the collect server). The third software, so called “monitoringService.py” should run on the service provider side.

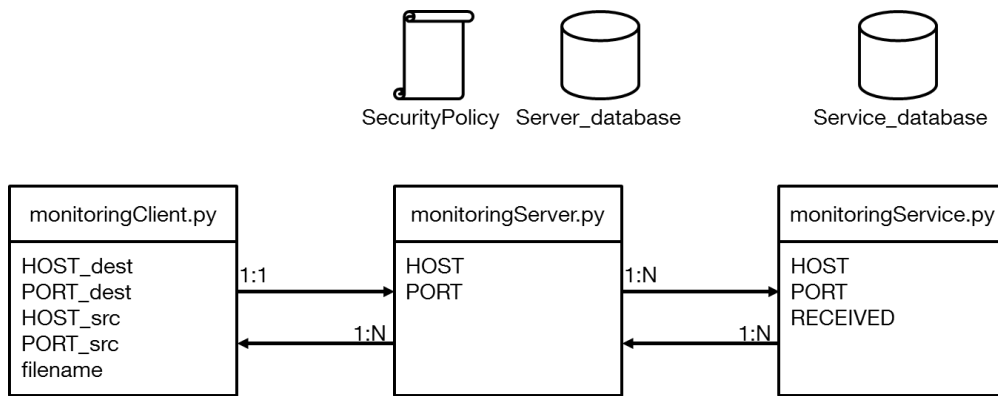


Figure 1: Generic Collector Interface diagram

2.2 Enabler Configuration

Describe how you configure your enabler, e.g., configuration files, command-line options, etc.

The “Generic Collector Interface” has a so called “securityPolicy” file (cf. Figure 1) and two database folders, i.e., “Server_database” and “Service_database” (cf. Figure 1). The “securityPolicy” file should be accessible only in “read” mode for this enabler, namely “monitoringServer.py”. This file contains the network configuration, i.e., “who can get which information”. For instance, this file can contain information like: data originated from the network components X must be transmitted only to the service A. This file could be simply organized in four fields per line as follows: <source IP address> <source port> <target IP address> <target port>. The “Server_database” folder should be available in “read” and “write” modes only for the server engine software. This folder will temporary contain the collected data. Once the data is correctly transmitted to the designated service, it is deleted from this folder. The “Service_database” folder should be available in “read” and “write” modes only for the service software.

The current main configurations of this enabler are

- The setup of the IP addresses and ports for the different running software (Figure 2, Figure 3, Figure 4).
- The setup of the storage folders and its access rights
- The setup of the security policy file and its access rights

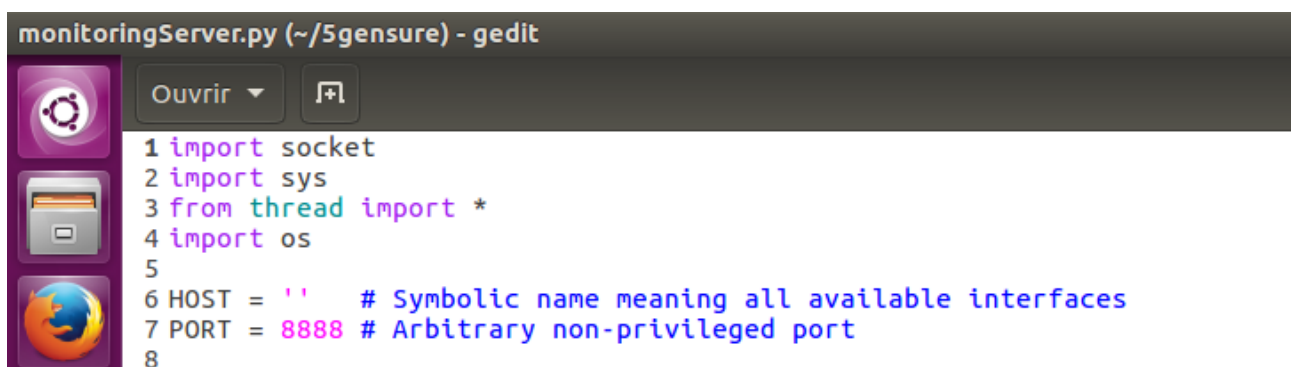
For “monitoringClient.py”, we need to precise the available IP address and port of the host running “monitoringServer.py” (lines 5 and 6 in Figure 2). Lines 8 and 9 specify the IP address and port from which data will be sent. HOST_src = ' ' means that “monitoringClient.py” will send data on the available interface. If we prefer to force using a given IP address, we need to replace ' ' by the given IP address.



```
monitoringClient.py (~/.5gensure) - gedit
1 import socket
2 import sys
3 from thread import *
4
5 HOST_dest = ' ' # Add the server IP address
6 PORT_dest = 8888 # Add the open port of the server
7
8 HOST_src = ' '
9 PORT_src = 4444
10
```

Figure 2: Configuration of "monitoringClient.py"

For “monitoringServer.py”, we also need to specify the IP address and port where to wait for inputs from the client engines (lines 6 and 7 in Figure 3). HOST = ' ' means that “monitoringServer.py” will wait for input on all the available interfaces. As previously mentioned, if we want to force “monitoringServer.py” to listen only to a given IP address, we should replace ' ' by the given IP address.



```
monitoringServer.py (~/.5gensure) - gedit
1 import socket
2 import sys
3 from thread import *
4 import os
5
6 HOST = ' ' # Symbolic name meaning all available interfaces
7 PORT = 8888 # Arbitrary non-privileged port
8
```

Figure 3: Configuration of "monitoringServer.py"

For “monitoringService.py”, we have almost the same configuration as “monitoringServer.py”. We also need to specify the IP address and port where to wait for inputs from the server engines (lines 5 and 6 in Figure 4). HOST = ' ' means that “monitoringService.py” will wait for input on all the available interfaces. As previously mentioned, if we want to force “monitoringService.py” to listen only to a given IP address, we should replace ' ' by the given IP address.

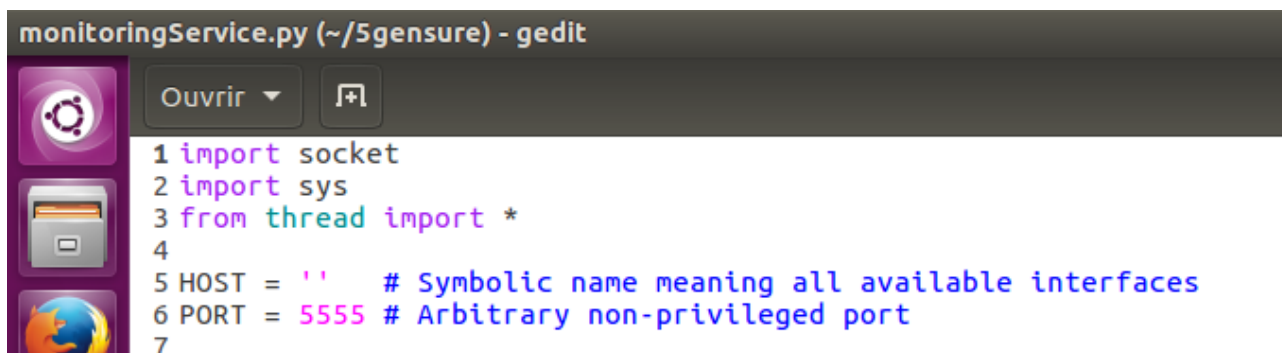


Figure 4: Configuration of "monitoringService.py"

2.3 Enabler Installation

Describe the process of installing your enabler.

In this link ([reference of the software in the wiki](#)) you can find compressed folder "GenericCollectorInterface.zip" containing three python scripts, i.e., "monitoringClient.py", "monitoringServer.py" and "moitoringService.py", an example of a security policy file "securityPolicy.txt", and an example of an XML report "report.xml".

The python script "monitoringClient.py" should be installed in all the network components that are supposed to collect and send reports to the Server engine. The python script "monitoringSrever.py" should be installed in a host that is accessible by the entire network component and that can access the host where installed the "moitoringService.py" script. "securityPolicy.txt" should be in the same host as the "monitoringsrever.py". "report.xml" can be duplicated in all the network components (This example can be used to facilitate the tests). Afterwards, "Server_database" (respectively. "Service_database") should be created in the machine hosting "monitoringServer.py" (respectively. "monitoringService.py").

2.4 Troubleshooting

Describe, e.g., error messages here and how to address such errors. Describe also problems with respect to the enabler's installation and its administration.

The message error, in Figure 5, can occur in the server engine side ("monitoringServer.py"). It is due to socket issues. This problem does not require any solution. It will be solved by the following report transmission process.

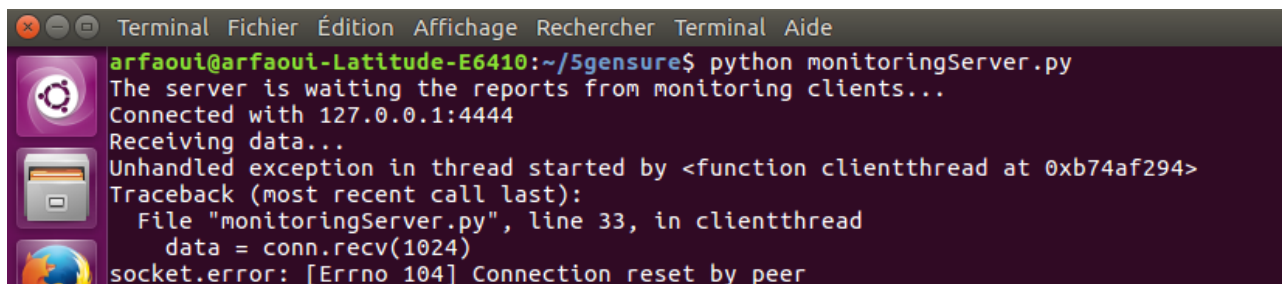


Figure 5: Socket error

3 User and Programmer Guide

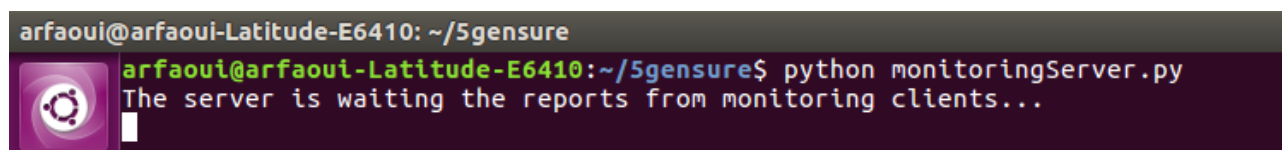
This subsection should describe how to use the enabler and how to “program” the enabler. In case your enabler is not “programmable,” the corresponding section (i.e., Section 3.2) should contain a note that this section is not applicable. For uniformity, do not delete this section. Feel free to introduce subsections.

3.1 User Guide

After installing the scripts

- 1- Start the server engine (Figure 6):

python monitoringServer.py



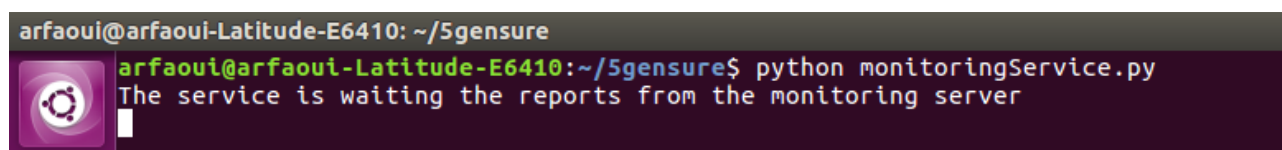
```
arfaoui@arfaoui-Latitude-E6410: ~/5gensure
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringServer.py
The server is waiting the reports from monitoring clients...
```

Figure 6: Start the server engine

At this moment, the server engine is ready to get any report from any client and then transmit it to the corresponding Service.

- 2- Start the service (Figure 7):

python monitoringService.py



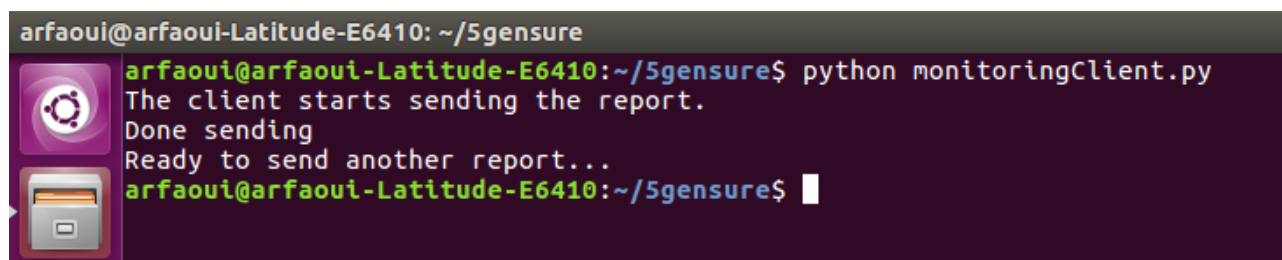
```
arfaoui@arfaoui-Latitude-E6410: ~/5gensure
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringService.py
The service is waiting the reports from the monitoring server
```

Figure 7: Start the service

At this moment, the service is ready to receive any report from any client.

- 3- Start a report transmission by the client engine (Figure 8):

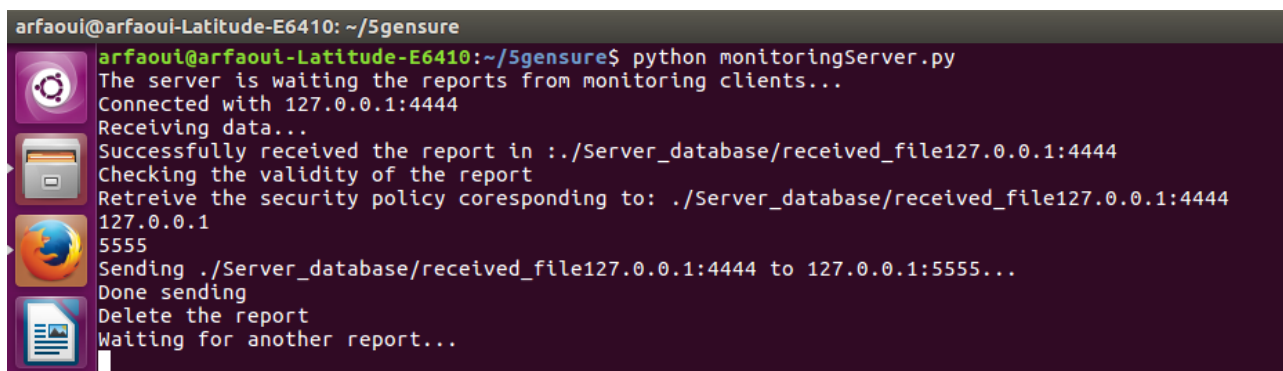
python monitoringClient.py



```
arfaoui@arfaoui-Latitude-E6410: ~/5gensure
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringClient.py
The client starts sending the report.
Done sending
Ready to send another report...
arfaoui@arfaoui-Latitude-E6410:~/5gensure$
```

Figure 8: Start a report transmission

This command triggers the transmission of a report from the client engine to the server engine (Figure 9 and Figure 10).

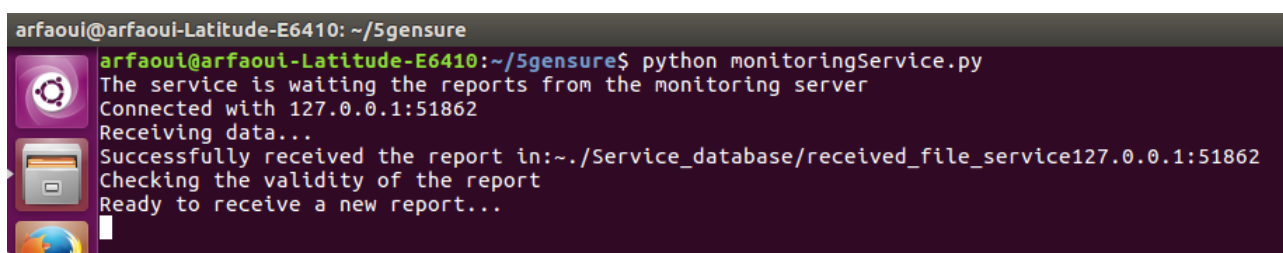


```

arfaoui@arfaoui-Latitude-E6410: ~/5gensure
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringServer.py
The server is waiting the reports from monitoring clients...
Connected with 127.0.0.1:4444
Receiving data...
Successfully received the report in ./Server_database/received_file127.0.0.1:4444
Checking the validity of the report
Retreive the security policy coresponding to: ./Server_database/received_file127.0.0.1:4444
127.0.0.1
5555
Sending ./Server_database/received_file127.0.0.1:4444 to 127.0.0.1:5555...
Done sending
Delete the report
Waiting for another report...

```

Figure 9: Server engine when receiving a report



```

arfaoui@arfaoui-Latitude-E6410: ~/5gensure
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringService.py
The service is waiting the reports from the monitoring server
Connected with 127.0.0.1:51862
Receiving data...
Successfully received the report in:~/Service_database/received_file_service127.0.0.1:51862
Checking the validity of the report
Ready to receive a new report...

```

Figure 10: Service when receiving a report

3.2 Programmer Guide

4 Unit Tests

This subsection should describe some simple tests for your enabler or components of your enabler. The section is not meant as an evaluation of your enabler; this is done in WP4. However, your tests should cover all the features of your enablers.

4.1 Information about Tests

Describe here the background on your tests and provide a brief overview on your tests.

4.2 Unit Test 1

Provide here the details of the test (what is tested? what is the input? what is the expected output? how do you run the test? etc.). If, e.g., the tests are run uniformly, feel free to describe this already in Section 4.1. The test should not depend on your specific environment in which you developed the enabler, i.e., it should be possible that others also run this test.

At this level, the goal of the test would be to check that the XML report is correctly transmitted from a client engine to the corresponding service. To do so, after following the three steps described in section 3.1. Check manually that the file “report.xml” has been transmitted only to the authorized service. We can also use Wireshark to trace the various exchanges between the different scripts.

5 Acknowledgements

Acknowledge relevant other work and projects, thank people for contributions, etc.

6 Abbreviations

List all used abbreviations here.

5G-PPP	5G Infrastructure Public Private Partnership

7 References

[1] W. Rudin, Functional Analysis, McGraw-Hill, 1973.