Tivoli Netcool Supports Guide to the EIF SDK SSL RCV1 Example

by

Jim Hutchinson

Document release: 2.0

Table of Contents

1Introduction....2

1.1Overview....2
1.2GSKit 7....2

2GSKit SSL....3

2.1EIFRCV1 SSL Certificate creation....3

3RCV1....5

3.1Configg....5
3.2eifrcv1_diag_config.txt....6

4Posteifmsg....7

4.1ssl.conf....7
4.2Example SSL posteifmsg command....7
4.3posteifmsg_diag_config.txt....8

5EIF MUX Example....9

5.1in.conf....9
5.2out.conf....9
5.3tivoli_eif.props....9
5.3.1mux_diag_config.txt....10
5.3.2eif_diag_config.txt....11

6Tracing....12

6.1Posteifmsg....12
6.2RCV1....14
6.3Example successful read....14
6.4Incorrect certificate....16
6.5No certificate....16

1 Introduction

1.1 Overview

The EIF SDK Supports SSL using GSKit 7 which can be downloaded from fix central.

The Type=SSL is used to set the SSL behaviour. In the examples provided, the basic SSL options are used to define the SSL KDB and TrustStore, which is the KDB by default. All SSL Ciphers are accepted by default.

Please refer to the EIF SDK manual for guidance on what other C EIF API settings are available.

1.2 GSKit 7

GSKit 7 needs to be installed on the system running the server and client applications.

GSKit V7 FAQ - Install, Uninstall and Upgrade instructions :

http://www-01.ibm.com/support/docview.wss?uid=swg21443730

Once installed the following files will be available [or their 64-bit equivalent]:

```
/bin/gsk7capicmd
/bin/gsk7cmd
/bin/gsk7ikm
/bin/gsk7ver
/usr/lib/libgsk7acmeidup.so
/usr/lib/libgsk7cms.so
/usr/lib/libgsk7dbfl.so
/usr/lib/libgsk7drld.so
/usr/lib/libgsk7iccs.so
/usr/lib/libgsk7kicc.so
/usr/lib/libgsk7kjni.so
/usr/lib/libgsk7km.so
/usr/lib/libgsk7krnc.so
/usr/lib/libgsk7krrb.so
/usr/lib/libgsk7krsw.so
/usr/lib/libgsk7msca.so
/usr/lib/libgsk7p11.so
/usr/lib/libgsk7ssl.so
/usr/lib/libgsk7sys.so
/usr/lib/libgsk7valn.so
```

2 GSKit SSL

2.1 EIFRCV1 SSL Certificate creation

In this example the GSKit 7 64-bit command [gsk7capicmd] is used to create the CA and ServerLocation KDB's.
The KDB's file location is given in the configg file.
The method presented can be used for other EIF SDK applications.

- Create a MASTER_CA databases to issue/sign the certificates [host#1]

```
mkdir gsk7-ssl
cd gsk7-ssl
/bin/gsk7capicmd_64 -keydb -create -db "./MASTER_CA.kdb" -type cms  -pw netcool
/bin/gsk7capicmd_64 -keydb -stashpw -db "./MASTER_CA.kdb" -type cms -pw netcool

/bin/gsk7capicmd_64 -cert -create -db "./MASTER_CA.kdb" -pw netcool \
-label "MASTER_CA" -size 1024 -ca true \
-dn "CN=MASTER_CA,O=IBM,OU=Support,L=SouthBank,ST=London,C=GB" \
-expire 1825 -x509version 3

/bin/gsk7capicmd_64 -cert -list -db "./MASTER_CA.kdb" -pw netcool
```

- Extract the MASTER_CA certificate to a file for use in other KDB's

```
/bin/gsk7capicmd_64 -cert -extract -db "./MASTER_CA.kdb" -pw netcool \
-label "MASTER_CA" \
-target "./MASTER_CA_CERT.arm"
```

- Create a KDB for the EIFRCV1 server

In this case its in the same directory and server as the MASTER_CA.

```
/bin/gsk7capicmd_64 -keydb -create -db "./eif.kdb"  -type cms -pw netcool
/bin/gsk7capicmd_64 -keydb -stashpw -db "./eif.kdb" -type cms -pw netcool

/bin/gsk7capicmd_64 -cert -add -db "./eif.kdb" -pw netcool \
-file "./MASTER_CA_CERT.arm" -label "MASTER_CA"

/bin/gsk7capicmd_64 -cert -list -db ./eif.kdb -pw netcool
```

- Create a certificate for the node FQDN

Use the ServerLocation's FQDN as the certificates label [**myserver.ibm.com**].

```
/bin/gsk7capicmd_64 -certreq -create -db "./eif.kdb" -pw netcool \
-label "myserver.ibm.com" \
-size 1024 \
-dn "CN=myserver.ibm.com,O=IBM,OU=Support,L=SouthBank,ST=London,C=GB" \
-file "./EIFRCV_req.arm"

/bin/gsk7capicmd_64 -cert -sign -db "./MASTER_CA.kdb" -pw netcool \
-label "MASTER_CA" \
-target "./EIFRCV_signed.arm" \
-expire 365 \
-file "./EIFRCV_req.arm"

/bin/gsk7capicmd_64 -cert -receive -db "./eif.kdb" -pw netcool \
-file "./EIFRCV_signed.arm"

/bin/gsk7capicmd_64 -cert -list -db ./eif.kdb -pw netcool
```

- Define the ServerLocation certificate as the default

```
/bin/gsk7capicmd_64 -cert -setdefault -label "redhat6.sbank.uk.ibm.com" -db ./eif.kdb
-pw netcool
```

Check it is set as the default certificate :

```
/bin/gsk7capicmd_64 -cert -getdefault -db ./eif.kdb -pw netcool
```

3 RCV1

3.1 Configg

The basic RCV1 SSL configg file is given below with <FQDN> being the hostname and label for the SSL certificate:

```
# C API Tracing
ed_diag_config_file=./eifrcv1_diag_config.txt
# RCV settings
BufferEvents=NO
RetryInterval=5
BufEvtPath=./cache.dat
# RCV1 SSL
TransportList=t1_
c1_nameSSLFIPSMode=OFF
t1_Type=SSL
t1_Channels=c1_
c1_Port=5555
c1_ServerLocation=<FQDN>
c1_SSLKeystore=./gsk7-ssl/eifrcv1.kdb
c1_SSLKeystorePW=netcool
#EOF
```

3.2 eifrcv1_diag_config.txt

Enabling the tracing is useful when trying to understand how the EIF SDK works.

```
Highest_level trace2
Truncate_on_restart true
tec_ed Highest_level trace2
# cache and buffering
tec_ed ed_buffer trace2 /tmp/eifrcv1.trace
tec_ed ed_cache trace2 /tmp/eifrcv1.trace
tec_ed ed_cache_controller trace2 /tmp/eifrcv1.trace
# cMain drivers
tec_ed ed_agent_comm trace2 /tmp/eifrcv1.trace
tec_ed ed_agent_util trace2 /tmp/eifrcv1.trace
tec_ed ed_tc trace2 /tmp/eifrcv1.trace
# TRANSPORTS
tec_ed ed_fwk_t trace2 /tmp/eifrcv1.trace
tec_ed ed_lcf_t trace2 /tmp/eifrcv1.trace
tec_ed ed_socket_t trace2 /tmp/eifrcv1.trace
# SOCKETS
tec_ed ed_eipc trace2 /tmp/eifrcv1.trace
tec_ed ed_socket_impl trace2 /tmp/eifrcv1.trace
tec_ed ed_socket_impl trace2 /tmp/eifrcv1.trace
tec_ed ed_socket_rcv trace2 /tmp/eifrcv1.trace
# API calls to eipc
tec_ed ed_c_recv_msg trace2 /tmp/eifrcv1.trace
tec_ed ed_c_ipc_poll trace2 /tmp/eifrcv1.trace
tec_ed ed_c_ipc trace2 /tmp/eifrcv1.trace
tec_ed ed_c_ipc_accept trace2 /tmp/eifrcv1.trace
tec_ed ed_c_pool trace2 /tmp/eifrcv1.trace
tec_ed ed_c_ipc_server trace2 /tmp/eifrcv1.trace
# API calls to ipc
tec_ed ed_recv_msg trace2 /tmp/eifrcv1.trace
tec_ed ed_ipc_poll trace2 /tmp/eifrcv1.trace
tec_ed ed_ipc trace2 /tmp/eifrcv1.trace
tec_ed ed_ipc_accept trace2 /tmp/eifrcv1.trace
tec_ed ed_pool trace2 /tmp/eifrcv1.trace
tec_ed ed_ipc_server trace2 /tmp/eifrcv1.trace
tec_ed ed_evmsg trace2 /tmp/eifrcv1.trace
# misc
tec_ed hn_target trace2 /tmp/eifrcv1.trace
# GSKit
tec_ed ed_gsk trace2 /tmp/eifrcv1.trace
#EOF
```

4 Posteifmsg

The posteifmsg.kdb can just be a copy of the MASTER_CA.kdb, otherwise you can create it using the same method used for the EIFRCV1 SSL KDB, provided they use the same MASTER CA.

4.1 ssl.conf

```
# EIF Tracing file
ed_diag_config_file=./posteifmsg_diag_config.txt
# EIF settings
BufferEvents=NO
RetryInterval=5
BufEvtPath=./cache.dat
# EIF Target port
TransportList=t1_
t1_Type=SSL
t1_Channels=c1_
c1_Port=5555
c1_ServerLocation=<FQDN>
c1_SSLKeystore=./gsk7-ssl/posteifmsg.kdb
c1_SSLKeystorePW=netcool
#EOF
```

4.2 Example SSL posteifmsg command

The posteifmsg command is located in the local directory for this example:

```
vi send_ssl.sh
./posteifmsg -f ssl.conf -m hello_ssl EVENT TEC
#EOF
:wq
chmod 755 send_ssl.sh
./send_ssl.sh
```

4.3 posteifmsg_diag_config.txt

```
Highest_level trace2
Truncate_on_restart true
tec_ed Highest_level trace2
# cache and buffering
tec_ed ed_buffer trace2 /tmp/posteifmsg.trace
tec_ed ed_cache trace2 /tmp/posteifmsg.trace
tec_ed ed_cache_controller trace2 /tmp/posteifmsg.trace
# cMain drivers
tec_ed ed_agent_comm trace2 /tmp/posteifmsg.trace
tec_ed ed_agent_util trace2 /tmp/posteifmsg.trace
tec_ed ed_tc trace2 /tmp/posteifmsg.trace
# TRANSPORTS
tec_ed ed_fwk_t trace2 /tmp/posteifmsg.trace
tec_ed ed_lcf_t trace2 /tmp/posteifmsg.trace
tec_ed ed_socket_t trace2 /tmp/posteifmsg.trace
# SOCKETS
tec_ed ed_eipc trace2 /tmp/posteifmsg.trace
tec_ed ed_socket_impl trace2 /tmp/posteifmsg.trace
tec_ed ed_socket_impl trace2 /tmp/posteifmsg.trace
tec_ed ed_socket_rcv trace2 /tmp/posteifmsg.trace
# API calls to eipc
tec_ed ed_c_recv_msg trace2 /tmp/posteifmsg.trace
tec_ed ed_c_ipc_poll trace2 /tmp/posteifmsg.trace
tec_ed ed_c_ipc trace2 /tmp/posteifmsg.trace
tec_ed ed_c_ipc_accept trace2 /tmp/posteifmsg.trace
tec_ed ed_c_pool trace2 /tmp/posteifmsg.trace
tec_ed ed_c_ipc_server trace2 /tmp/posteifmsg.trace
# API calls to ipc
tec_ed ed_recv_msg trace2 /tmp/posteifmsg.trace
tec_ed ed_ipc_poll trace2 /tmp/posteifmsg.trace
tec_ed ed_ipc trace2 /tmp/posteifmsg.trace
tec_ed ed_ipc_accept trace2 /tmp/posteifmsg.trace
tec_ed ed_pool trace2 /tmp/posteifmsg.trace
tec_ed ed_ipc_server trace2 /tmp/posteifmsg.trace
tec_ed ed_evmsg trace2 /tmp/posteifmsg.trace
# misc
tec_ed hn_target trace2 /tmp/posteifmsg.trace
#
tec_ed ed_gsk trace2 /tmp/posteifmsg.trace
#EOF
```

5 EIF MUX Example

The Tivoli EIF MUX is a C EIF API binary that allows Tivoli EIF events to be sent to a given host and port, and then forwarded to the Tivoli EIF probe.
In the example, the in.conf uses t2_ and c2_ to define the listening servers transport and channels.
With the out.conf using t1_ and c1_ defining the senders transport and channels.
The eif_mux.kdb can be created using the same method used for the EIFRCV1 KDB.

5.1 in.conf

The in.conf defines the server listening port with the SSL certificate.

```
BufEvtPath=./1cache.dat
BufferEvents=NO
RetryInterval=5
ed_diag_config_file=./mux_diag_config.txt
TransportList=t2_
t2_Type=SSL
t2_Channels=c2_
c2_ServerLocation=myserver.ibm.com
c2_Port=9990
c2_SSLKeystore=./gsk7-ssl/eif_mux.kdb
c2_SSLKeystorePW=netcool
#EOF
```

5.2 out.conf

The out.conf file defines the connection between the MUX and the Tivoli EIF probe.

```
BufEvtMaxSize=640000
BufEvtPath=mux_primary.dat
BufferEvents=NO
ConnectionMode=connection_oriented
ed_diag_config_file=./eif_diag_config.txt
FQDomain=NO
LogFileName=./eifout.log
LogLevel=NO
# Connecting to Tivoli EIF probe
c1_Port=9998
c1_ServerLocation=myserver.ibm.com
RetryInterval=5
t1_Channels=c1_
t1_Type=SOCKET
TestMode=NO
TransportList=t1_
#EOF
```

5.3 tivoli_eif.props

The Tivoli EIF probe needs the probe port to be set the same as the ServerLocation given in the out.conf.

```
PortNumber              : 9998
```

5.3.1 mux_diag_config.txt

```
Highest_level trace2
Truncate_on_restart true
tec_ed Highest_level trace2
# cache and buffering
tec_ed ed_buffer trace2 /tmp/eif_mux.trace
tec_ed ed_cache trace2 /tmp/eif_mux.trace
tec_ed ed_cache_controller trace2 /tmp/eif_mux.trace
# cMain drivers
tec_ed ed_agent_comm trace2 /tmp/eif_mux.trace
tec_ed ed_agent_util trace2 /tmp/eif_mux.trace
tec_ed ed_tc trace2 /tmp/eif_mux.trace
# TRANSPORTS
tec_ed ed_fwk_t trace2 /tmp/eif_mux.trace
tec_ed ed_lcf_t trace2 /tmp/eif_mux.trace
tec_ed ed_socket_t trace2 /tmp/eif_mux.trace
# SOCKETS
tec_ed ed_eipc trace2 /tmp/eif_mux.trace
tec_ed ed_socket_impl trace2 /tmp/eif_mux.trace
tec_ed ed_socket_impl trace2 /tmp/eif_mux.trace
tec_ed ed_socket_rcv trace2 /tmp/eif_mux.trace
# API calls to eipc
tec_ed ed_c_recv_msg trace2 /tmp/eif_mux.trace
tec_ed ed_c_ipc_poll trace2 /tmp/eif_mux.trace
tec_ed ed_c_ipc trace2 /tmp/eif_mux.trace
tec_ed ed_c_ipc_accept trace2 /tmp/eif_mux.trace
tec_ed ed_c_pool trace2 /tmp/eif_mux.trace
tec_ed ed_c_ipc_server trace2 /tmp/eif_mux.trace
# API calls to ipc
tec_ed ed_recv_msg trace2 /tmp/eif_mux.trace
tec_ed ed_ipc_poll trace2 /tmp/eif_mux.trace
tec_ed ed_ipc trace2 /tmp/eif_mux.trace
tec_ed ed_ipc_accept trace2 /tmp/eif_mux.trace
tec_ed ed_pool trace2 /tmp/eif_mux.trace
tec_ed ed_ipc_server trace2 /tmp/eif_mux.trace
tec_ed ed_evmsg trace2 /tmp/eif_mux.trace
# misc
tec_ed hn_target trace2 /tmp/eif_mux.trace
# GSKit
tec_ed ed_gsk trace2 /tmp/eif_mux.trace
#EOF
```

5.3.2 eif_diag_config.txt

```
Highest_level trace2
Truncate_on_restart true
tec_ed Highest_level trace2
# cache and buffering
tec_ed ed_buffer trace2 /tmp/eif_out.trace
tec_ed ed_cache trace2 /tmp/eif_out.trace
tec_ed ed_cache_controller trace2 /tmp/eif_out.trace
# cMain drivers
tec_ed ed_agent_comm trace2 /tmp/eif_out.trace
tec_ed ed_agent_util trace2 /tmp/eif_out.trace
tec_ed ed_tc trace2 /tmp/eif_out.trace
# TRANSPORTS
tec_ed ed_fwk_t trace2 /tmp/eif_out.trace
tec_ed ed_lcf_t trace2 /tmp/eif_out.trace
tec_ed ed_socket_t trace2 /tmp/eif_out.trace
# SOCKETS
tec_ed ed_eipc trace2 /tmp/eif_out.trace
tec_ed ed_socket_impl trace2 /tmp/eif_out.trace
tec_ed ed_socket_impl trace2 /tmp/eif_out.trace
tec_ed ed_socket_rcv trace2 /tmp/eif_out.trace
# API calls to eipc
tec_ed ed_c_recv_msg trace2 /tmp/eif_out.trace
tec_ed ed_c_ipc_poll trace2 /tmp/eif_out.trace
tec_ed ed_c_ipc trace2 /tmp/eif_out.trace
tec_ed ed_c_ipc_accept trace2 /tmp/eif_out.trace
tec_ed ed_c_pool trace2 /tmp/eif_out.trace
tec_ed ed_c_ipc_server trace2 /tmp/eif_out.trace
# API calls to ipc
tec_ed ed_recv_msg trace2 /tmp/eif_out.trace
tec_ed ed_ipc_poll trace2 /tmp/eif_out.trace
tec_ed ed_ipc trace2 /tmp/eif_out.trace
tec_ed ed_ipc_accept trace2 /tmp/eif_out.trace
tec_ed ed_pool trace2 /tmp/eif_out.trace
tec_ed ed_ipc_server trace2 /tmp/eif_out.trace
tec_ed ed_evmsg trace2 /tmp/eif_out.trace
# misc
tec_ed hn_target trace2 /tmp/eif_out.trace
# GSKit - not needed
# tec_ed ed_gsk trace2 /tmp/eif_out.trace
#EOF
```

6 Tracing

6.1 Posteifmsg

Posteifmsg : Correct certificate

```
Connected using SSLv3 Protocol
send_to()
62 bytes on send
Sender data   +5bytes:
Dumping [67] bytes.
     01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20
00000  3 45 56 45 4e 54 3b 73 6f 75 72 63 65 3d 54 45 43 3b 6d 73  | .EVENT;source=TEC;ms
00020 67 3d 68 65 6c 6c 6f 5f 67 73 6b 37 5f 73 73 6c 3b 6f 72 69  | g=hello_gsk7_ssl;ori
00040 67 69 6e 3d 39 2e 31 35 33 2e 35 35 3b 45 4e 44  | gin=9.999.999.55;END
00060  a  1  0 20  0  0 18                              | .......
send rtss
get_peer_response_timed fd 4 tv 0.0
peer response PEER_RESPONSE_NONE
handle is_ssl=1
send 40 bytes over ssl channel
send rv 0, length_written 40
Send the rest
handle is_ssl=1
send 58 bytes over ssl channel
send rv 0, length_written 58
62 bytes on send rc=0
```

Posteifmsg : Wrong or rejected certificate

```
ed_gsk_secure_soc_init failed, rc=414, error message = GSK_ERROR_BAD_CERT
SSL handshake failed, rv=414
```

Posteifmsg : No SSL

```
send rtss
get_peer_response_timed fd 4 tv 0.0
peer response PEER_RESPONSE_NONE
handle is_ssl=0
send 40 bytes
Send the rest
handle is_ssl=0
send 56 bytes
60 bytes on send rc=0
```

Posteifmsg : GSKit 7 is not installed

```
failed to load gskit library, rc=-1, error message = ld.so.1: posteifmsg: fatal:
libgsk7ssl.so: open failed: No such file or directory
Failed to load GSKit library
```

Posteifmsg : Wrong KeyStoreDB password
Command line shows:
```
tec_put_event failed, errno = 0
```
Log shows:
```
ed_gsk_environment_init failed, rc=408, error message =
GSK_ERROR_BAD_KEYFILE_PASSWORD
```

Posteifmsg : Missing KeyStoreDB file
Command line shows:
```
tec_put_event failed, errno = 0
```
Log shows:
```
ed_gsk_environment_init failed, rc=202, error message = GSK_KEYRING_OPEN_ERROR-
Keyring file did not open
```

Posteifmsg : Just MASTER_CA and no default certificate [**works**]
Log shows: Succesful logging details as seen with the correct client certificate

Posteifmsg : No MASTER_CA and no default
Command line shows:
```
tec_put_event failed, errno = 0
```
Log shows:
```
ed_gsk_secure_soc_init failed, rc=414, error message = GSK_ERROR_BAD_CERT
SSL handshake failed, rv=414
```

6.2 RCV1

6.3 Example successful read

```
ed_gsk.c:529: Connected using SSLv2 Protocol
edc_ipc.c:1427: EDC_IPC_timedIsReady_q 0
edc_ipc.c:428: eipc getsockopt
edc_ipc.c:441:         Nagle algorithm disabled
edc_ipc.c:1254: Creation of new IPC pool element succeeded
socket_imp.c:3012: connection info connection Made [2]
edc_pool.c:337: New iterator @ 15e848 (current = -1, pool = a3f90)
edc_pool.c:337: New iterator @ 15e848 (current = -1, pool = a3f90)
edc_ipc.c:1427: EDC_IPC_timedIsReady_q 1
socket_imp.c:2350: data ready: proceed  EDC_IPC_recv()
edc_pool.c:337: New iterator @ 15e848 (current = -1, pool = a3f90)
sockeif.c:319: fd_trace closing fd -1
edc_ipc.c:892: EDC_IPC_recv
eipc.c:1726: eipc_check_input()
edc_pool.c:337: New iterator @ 15e848 (current = -1, pool = a3f90)
edc_ipc.c:932: HANDLE COUNT 1
edc_ipc.c:827: EDC_IPC_recv_one
edc_ipc.c:856: Receiving data on channel E of type = I
edc_ipc.c:664: receive_data
edc_ipc.c:678: Reading IPC...
eipc.c:1627: eipc_recv()
eipc.c:1578: status 0   done 0  error 0
eipc.c:1382: eipc_recv_msg
eipc.c:1389: Allocating 36 bytes for header
eipc.c:1397: receiving header
eipc.c:1540: do_ssl_receive recv fd 5
eipc.c:1550: ed_gsk_read success buf=<START>> len=36
eipc.c:1478: receiving data 62 bytes
eipc.c:1540: do_ssl_receive recv fd 5
eipc.c:1550: ed_gsk_read success buf=EVE len=4
eipc.c:1604: status 0   done 0  error 0
eipc.c:1578: status 0   done 0  error 0
eipc.c:1382: eipc_recv_msg
eipc.c:1478: receiving data 58 bytes
eipc.c:1540: do_ssl_receive recv fd 5
eipc.c:1550: ed_gsk_read success buf=NT;source=TEC;msg=hello_gsk7_ssl;origin=9.999.999.99;END

eipc.c:1604: status 0   done 1  error 0
edc_ipc.c:681:         Got 62 bytes (ret = 0, error = 0)
edc_ipc.c:700: calling EDC_MSG_recv
evmsg.c:135: UTF8 interrogation
evmsg.c:145: Removing ED_IPC_UTF8_DELIMITOR
evmsg.c:285: evmsg_evaluate_nobuff
evmsg.c:301: Parsing @ 61 bytes

 13 14 15 16 17 18 19 20
 3d 54 45 43 3b 6d 73 67  | EVENT;source=TEC;msg
 73 73 6c 3b 6f 72 69 67  | =hello_gsk7_ssl;orig
 2e 35 35 3b 45 4e 44  a  | in=9.999.999.99;END.
                          | ......

evmsg.c:382: RESULT @ bufferSize 61

 13 14 15 16 17 18 19 20
 3d 54 45 43 3b 6d 73 67  | EVENT;source=TEC;msg
 73 73 6c 3b 6f 72 69 67  | =hello_gsk7_ssl;orig
 2e 35 35 3b 45 4e 44  a  | in=9.999.999.99;END.
                          | ......

edc_ipc.c:609: Setting input FDs and polling...
eipc.c:1726: eipc_check_input()
edc_ipc.c:866: EDC_IPC_recv_one returning
edc_ipc.c:867: EDC_IPC_recv_one [61][EVENT;source=TEC;msg=hello_gsk7_ssl;origin=9.999.999.99;END

edc_ipc.c:950: EDC_IPC_recv returning
edc_ipc.c:951: EDC_IPC_recv Returning package [EVENT;source=TEC;msg=hello_gsk7_ssl;origin=9.999.999.99;END
```

```
socket_imp.c:2359: transport package [EVENT;source=TEC;msg=hello_gsk7_ssl;origin=9.999.999.99;END

socket_imp.c:707: Receiver data   bytes:

 13 14 15 16 17 18 19 20
 3d 54 45 43 3b 6d 73 67  | EVENT;source=TEC;msg
 73 73 6c 3b 6f 72 69 67  | =hello_gsk7_ssl;orig
 2e 35 35 3b 45 4e 44  a  | in=9.999.999.99;END.
                          | .

tc.c:408: Parsing @ 61 bytes

 13 14 15 16 17 18 19 20
 3d 54 45 43 3b 6d 73 67  | EVENT;source=TEC;msg
 73 73 6c 3b 6f 72 69 67  | =hello_gsk7_ssl;orig
 2e 35 35 3b 45 4e 44  a  | in=9.999.999.99;END.
                          | ......

edc_pool.c:337: New iterator @ 15e848 (current = -1, pool = a3f90)
edc_pool.c:337: New iterator @ 15e848 (current = -1, pool = a3f90)
edc_ipc.c:1427: EDC_IPC_timedIsReady_q 1
socket_imp.c:2350: data ready: proceed  EDC_IPC_recv()
edc_pool.c:337: New iterator @ 15e848 (current = -1, pool = a3f90)
edc_ipc.c:892: EDC_IPC_recv
eipc.c:1726: eipc_check_input()
edc_pool.c:337: New iterator @ 15e848 (current = -1, pool = a3f90)
edc_ipc.c:932: HANDLE COUNT 1
edc_ipc.c:827: EDC_IPC_recv_one
edc_ipc.c:856: Receiving data on channel E of type = I
edc_ipc.c:664: receive_data
edc_ipc.c:678: Reading IPC...
eipc.c:1627: eipc_recv()
eipc.c:1578: status 0   done 0  error 0
eipc.c:1382: eipc_recv_msg
eipc.c:1389: Allocating 36 bytes for header
eipc.c:1397: receiving header
eipc.c:1540: do_ssl_receive recv fd 5
ed_gsk.c:581: ed_gsk_secure_soc_read failed, rc=406, error message = GSK_ERROR_IO
eipc.c:1546: recv socket returned 406
eipc.c:1402: No HEADER
eipc.c:1604: status -1   done 0  error 0
edc_ipc.c:681:         Got 0 bytes (ret = -1, error = 0)
edc_ipc.c:866: EDC_IPC_recv_one returning
edc_ipc.c:867: EDC_IPC_recv_one [0][NULL]
edc_recv_msg.c:156: Comm. with T/EC Agent broken (error 0)
edc_ipc.c:350: Shutdown communication with E ...
edc_ipc.c:330: Shutdown only communication with E ...
sockeif.c:273: _imp_eipc_shutdown fd 5  option 2
sockeif.c:277: _imp_eipc_shutdown fd 5  option 2 rc=0
edc_ipc.c:336: Shutdown Only succeeded
edc_ipc.c:353:         Destroying IPC connection
sockeif.c:319: fd_trace closing fd 5
sockeif.c:299: eipc_destroy_handle close fd 5
edc_ipc.c:357: Freeing IPC handle  private data
edc_ipc.c:367: Freeing IPC handle 0
edc_pool.c:283: Deleting item 161d60 from pool a3f90
edc_pool.c:296: Decrementing iterator 15e848 (current = 0, pool = a3f90)
edc_pool.c:304: Number of Elements from [1] to [0]
edc_ipc.c:371: Call safety signal
edc_pool.c:164: Safety Signal - curr [0] safety [102]  @[80] percent of [128]
edc_pool.c:171: Safety Met
edc_ipc.c:374: Shutdown communication succeeded
edc_ipc.c:950: EDC_IPC_recv returning
edc_ipc.c:951: EDC_IPC_recv Returning package [NULL]
socket_imp.c:2359: transport package [NULL]
```

6.4 Incorrect certificate

```
ssockeif.c:558: socket accepted
sockeif.c:571: Peer info retrieved.
edc_ipc.c:1427: EDC_IPC_timedIsReady_q 0
ed_gsk.c:524: ed_gsk_secure_soc_init failed, rc=420, error message = GSK_ERROR_SOCKET_CLOSED
sockeif.c:319: fd_trace closing fd 5
sockeif.c:299: eipc_destroy_handle close fd 5
```

6.5 No certificate

```
sockeif.c:558: socket accepted
sockeif.c:571: Peer info retrieved.
ed_gsk.c:524: ed_gsk_secure_soc_init failed, rc=410, error message = GSK_ERROR_BAD_MESSAGE
sockeif.c:319: fd_trace closing fd 5
sockeif.c:299: eipc_destroy_handle close fd 5
```